UNIVERSIDAD HISPANOAMERICANA

ESCUELA DE INGENIERÍA INFORMÁTICA

Propuesta de una metodología de control de calidad del software adaptada a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el Área de Informática de Gestión del Poder Judicial, II Circuito Judicial de San José

Sustentante:
Aida Elena Siles Rojas

Tutor:
Ing. Cristian Campos Agüero. MGP

Febrero, 2018

Contenido

Índice de Figuras

Índice de Tablas

Glosario

- *Backlog*: Conjunto de historias de usuario especificadas por el *product owner* y que se encuentra dentro de un documento o herramienta de apoyo a la gestión de proyectos.
- Calidad: Es un conjunto de propiedades y de características de un producto o servicio que le confieren capacidad para satisfacer necesidades expresadas o implícitas.
- Defecto de software: Es el resultado de un fallo o deficiencia durante el proceso de creación de programas de ordenador, computadora u otro dispositivo.
- Historia de usuario: Es la descripción de una funcionalidad que debe incorporar un sistema de software.
- Metodología: Es un sistema de prácticas, técnicas, procedimientos y normas utilizados por quienes trabajan en una disciplina. Es una forma estratégica de trabajo para trabajar bien y en orden durante el desarrollo del proyecto.
- *Milestone:* Es una tarea de duración cero que simboliza el haber conseguido un logro importante en el proyecto, son una forma de conocer el avance del proyecto sin estar familiarizado con el proyecto.
- Proceso: Es un conjunto de fases sucesivas o serie de pasos organizados cuyo fin es alcanzar un objetivo determinado.
- *Product Owner*: En español se le conoce como dueño del producto. Es un rol de Scrum que representa al cliente, usuarios del software y todas aquellas partes interesadas en el producto y es quien decide las funcionalidades y características que tendrá el producto.
- *Release*: Es una versión de software donde se incluyen nuevas funcionalidades para distribuirla a los clientes, no solo incluye el código sino que también la documentación, archivos de configuración, archivos de datos, programa de instalación, etc.

- *Roadmap (Mapa de ruta en español):* Es una planificación del desarrollo de software con los objetivos a corto y largo plazo. Se suele organizar en hitos o *milestones,* que son fechas en las que supuestamente estará finalizado un paquete de nuevas funcionalidades. Se puede identificar el estado actual y el posible futuro del software, dando una visión general hacia dónde apunta a llegar el software.
- Scrum*:* Es un marco de trabajo de desarrollo ágil de software. Los requerimientos son especificados como historias de usuario dentro del *backlog* y se utilizan iteraciones cortas llamadas por el nombre de *sprint* que permiten entregar productos funcionales al cliente.
- *Scrum master*: En español se le conoce como el maestro de Scrum. Es el que ordena las reuniones diarias, rastrea el atraso del trabajo a realizar, se comunica con los clientes y administradores fuera del proyecto.
- *Software*: Se define software como un conjunto de programas que permiten el desarrollo de una actividad en una computadora.
- *Sprint*: Iteración o período de tiempo de trabajo normalmente de dos a cuatro semanas en donde un equipo multifuncional produce parte de un producto a partir de las historias de usuario que se establecieron en el sprint.
- *Stakeholders:* Son todas aquellas personas, grupos y entidades que tienen intereses de cualquier tipo en una empresa y se ven afectados por sus actividades. Por ejemplo: Propietarios, directivos, trabajadores, proveedores, clientes, accionistas, etc.
- *Tester:* Persona encargada de realizar el control de la calidad de los entregables, verifica que los entregables se encuentre debidamente desarrollados a como se especificó.
- *Timeboxing:* En la gestión de proyectos, el *timeboxing* es la asignación de un tiempo fijo para completar cada aspecto del proyecto en su conjunto.

- TMMi: Es un modelo de calidad especializado en pruebas cuyas raíces provienen del modelo TMM, el modelo busca cubrir todos los aspectos de la calidad del software.

Declaración Jurada

DECLARACIÓN JURADA

Yo Aída Elena Siles Rojas, mayor de edad, portador de la cédula de identidad número 1-1288-0127 egresado de la carrera de Ingeniería Informática de la Universidad Hispanoamericana, hago constar por medio de éste acto y debidamente apercibido y entendido de las penas y consecuencias con las que se castiga en el Código Penal el delito de perjurio, ante quienes se constituyen en el Tribunal Examinador de mi trabajo de tesis para optar por el título de Licenciatura en Ingeniería Informática, juro solemnemente que mi trabajo de investigación titulado: Propuesta de una metodología de control de calidad del software adaptado a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el área de Informática de Gestión del Poder Judicial, II Circuito Judicial de San José es una obra original que ha respetado todo lo preceptuado por las Leyes Penales, así como la Ley de Derecho de Autor y Derecho Conexos número 6683 del 14 de octubre de 1982 y sus reformas, publicada en la Gaceta número 226 del 25 de noviembre de 1982; incluyendo el numeral 70 de dicha ley que advierte; articulo 70. Es permitido citar a un autor, transcribiendo los pasajes pertinentes siempre que éstos no sean tantos y seguidos, que puedan considerarse como una producción simulada y sustancial, que redunde en perjuicio del autor de la obra original. Asimismo, quedo advertido que la Universidad se reserva el derecho de protocolizar este documento ante Notario Público.

En fe de lo anterior, firmo en la ciudad de San José, a los Veinte días del mes de Febrero del año dos mil dieciocho.

Firma del estudiante

Cédula: 1-1288-0127

Carta de aprobación del Tutor

CARTA DEL TUTOR

San José, 19 de febrero del 2018.

Ph.D Yenory Rojas Hernández
Escuela de Ingeniería en Informática
Universidad Hispanoamericana

Estimado señor:

La estudiante Aida Elena Siles Rojas, cédula de identidad número 1-1288-0127, me ha presentado, para efectos de revisión y aprobación, el trabajo de investigación denominado ***Propuesta de una metodología de control de calidad del software adaptada a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el Área de Informática de Gestión del Poder Judicial, II Circuito Judicial de San José,*** el cual ha elaborado para optar por el grado académico de licenciatura.

En mi calidad de tutor, he verificado que se han hecho las correcciones indicadas durante el proceso de tutoría y he evaluado los aspectos relativos a la elaboración del problema, objetivos, justificación; antecedentes, marco teórico, marco metodológico, tabulación, análisis de datos; conclusiones y recomendaciones.

De los resultados obtenidos por el postulante, se obtiene la siguiente calificación:

a)	ORIGINAL DEL TEMA	10%	9%
b)	CUMPLIMIENTO DE ENTREGA DE AVANCES	20%	20%
c)	COHERENCIA ENTRE LOS OBJETIVOS, LOS INSTRUMENTOS APLICADOS Y LOS RESULTADOS DE LA INVESTIGACION	30%	30%
d)	RELEVANCIA DE LAS CONCLUSIONES Y RECOMENDACIONES	20%	20%
e)	CALIDAD, DETALLE DEL MARCO TEORICO	20%	18%
	TOTAL		**97%**

En virtud de la calificación obtenida, se avala el traslado al proceso de lectura.

Atentamente,

Ing. Cristian Campos Agüero. MGP
Cédula identidad 160400100307
Carné CPIC 3568

Carta de aprobación del Lector

CARTA DE LECTOR

San José, 18 de Abril del 2018

Ing. Yenory Rojas Hernández PhD.
Ingeniería en Informática
Universidad Hispanoamericana

Estimada señora

El estudiante Aida Elena Siles Rojas, cédula de identidad 1-1288-0127, me ha presentado para efectos de revisión y aprobación, el trabajo de investigación denominado "Propuesta de una metodología de control de calidad del software adaptada a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el Área de Informática de Gestión del Poder Judicial, II Circuito Judicial de San José", el cual ha elaborado para obtener su grado de Licenciatura en Ingeniería Informática.

He revisado y he hecho las observaciones relativas al contenido analizado, particularmente lo relativo a la coherencia entre el marco teórico y análisis de datos, la consistencia de los datos recopilados y la coherencia entre éstos y las conclusiones; asimismo, la aplicabilidad y originalidad de las recomendaciones, en términos de aporte de la investigación.

Por consiguiente, este trabajo cuenta con mi aval para ser presentado en la defensa pública.

Atte.

Lic. Pedro I. Leiva Chinchilla.
1-1394-0453

Carta de revisión filológica

CARTA DE REVISIÓN FILOLÓGICA

San José, 25 de abril de 2018

Señores
Universidad Hispanoamericana
Escuela de Ingeniería Informática

Estimados señores:

La estudiante **Aida Elena Silos Rojas** me ha presentado, para efectos de corrección de estilo, en mi calidad de profesional graduado en Filología y Enseñanza del Español, la tesis denominada **Propuesta de una metodología de control de calidad del software adaptada a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el Área de Informática de Gestión del Poder Judicial, II Circuito Judicial de San José**, la cual ha sido elaborada como parte de los requisitos para optar por el grado de Licenciatura en Ingeniería Informática.

He revisado, de acuerdo con los lineamientos de la corrección de estilo señalados por la Universidad, los aspectos de estructura gramatical, acentuación, ortografía, puntuación y los vicios de dicción que se trasladan a lo escrito, y he verificado que se han realizado todas las correcciones indicadas en el documento.

Agradeciendo su atención,

Lic. Henry Rivera Morales
Céd. 1-1195-0430
N° 036633
Colegio de Licenciados y Profesores

Dedicatoria

A mis padres, hermanas y a mi sobrino Ismael.

Agradecimiento

A Dios por darme fortaleza, sabiduría y salud para culminar esta etapa en mi vida.

A mi padre, a mi madre y a mis hermanas por darme fuerzas a seguir adelante y apoyarme en todo momento.

A mi prima Laura por toda la ayuda brindada.

A mi tutor Cristian Campos por el conocimiento trasmitido y el apoyo brindado en el proceso de elaboración de este proyecto.

Abreviaturas

Abreviatura	**Significado**
ACS	Aseguramiento de la calidad del software
CMM	*Capability Maturity Model*
DTIC	Dirección de Tecnologías de la Información y Comunicaciones
FDD	*Feature Driven Development*
IEEE	*Institute of Electrical and Electronics Engineers*
INVEST	*Independent, Negotiable, Valuable, Estimable, Small, Testable.*
ISO	*International Organization for Standardization*
ISTQB	*International Software Testing Qualifications Board*
PETIC	Plan Estratégico de Tecnologías de la Información y Comunicaciones
QA	*Quality Assurance*
SIAGPJ	Sistema Integrado de Gestión Administrativa del Poder Judicial
TDD	*Test Driven Development*
TI	Tecnología de la Información
TMMi	*Testing Maturity Model integrated*
VSTS	*Visual Studio Team Services*
XP	*Extreme Programming*

CAPÍTULO I

1. PLANTEAMIENTO DEL PROBLEMA

1.1 ANTECEDENTES Y JUSTIFICACIÓN DEL PROYECTO

1.1.1 Antecedentes del contexto de la institución

La institución donde se realiza el proyecto es en el Poder Judicial de Costa Rica, esta institución fue creada en 1825 con la Ley Fundamental del Estado Libre de Costa Rica, tiene como obligación hacer respetar las leyes y administrar la justicia y se dirige por las directrices legales establecidas en la Ley Orgánica del Poder Judicial, ley número 7333 del 5 de mayo de 1993, que establece en el artículo I:

> Corresponde al Poder Judicial, además de las funciones que la Constitución le señala, conocer de los procesos civiles, penales, penales juveniles, comerciales, de trabajo, contencioso-administrativo y civiles de hacienda, constitucionales, de familia y agrarios, así como de las otras que establezca la Ley; resolver definitivamente sobre ellos y ejecutar las resoluciones que pronuncie, con la ayuda de la Fuerza Pública si fuere necesario (p.1).

1.1.1.1 Misión del Poder Judicial

La misión del Poder Judicial de Costa Rica es: “Administrar justicia en forma pronta, cumplida, sin denegación y en estricta conformidad con el ordenamiento jurídico, que garanticen calidad en la prestación de servicios para las personas usuarias que lo requieran.” (Dirección de Tecnologías de Información y Comunicaciones Poder Judicial, 2015, p.43).

1.1.1.2 Visión del Poder Judicial

La visión del Poder Judicial de Costa Rica es:

> Ser un Poder Judicial que garantice a la persona usuaria el acceso a la justicia y resuelva sus conflictos con modernos sistemas de organización y gestión; compuesto por personal orientado por valores institucionales compartidos, conscientes de su papel en el desarrollo de la nación y apoyados en socios estratégicos (Dirección de Tecnologías de Información y Comunicaciones Poder Judicial, 2015, p.42).

1.1.1.3 Funciones del Poder Judicial de Costa Rica

Algunas de las funciones del Poder Judicial de Costa Rica son:

- Informar a los otros Poderes del Estado en los asuntos en que la Constitución o las leyes determinen que sea consultado y darle su opinión cuando sea requerido acerca de los proyectos de reforma a la legislación codificada o que afecten la organización o funcionamiento del Poder Judicial.
- Proponer las reformas legislativas y reglamentarias que juzgue convenientes para mejorar la administración de la justicia.
- Aprobar el proyecto de presupuesto del Poder Judicial, el cual una vez promulgado por la Asamblea Legislativa, podrá ejecutar por medio del Consejo.
- Nombrar a los miembros propietarios y suplentes del Tribunal Supremo de Elecciones.
- Resolver las competencias que se susciten entre las Salas de la Corte, con excepción de lo dispuesto por la ley respecto de la Sala Constitucional.
- Designar, en votación secreta, al Presidente y Vicepresidente de la Corte por períodos de cuatro años y dos años respectivamente, quienes podrán ser reelectos por períodos iguales.

- Promulgar por iniciativa propia o a propuesta del Consejo Superior del Poder Judicial, los reglamentos internos de orden y de servicio que estime pertinentes.
- Conocer los recursos de casación y de revisión de las sentencias dictadas por las Salas Segunda y Tercera, cuando estas actúen como tribunal de juicio o de única instancia (Poder Judicial de Costa Rica, 2016).

La única excepción existente a la actividad jurisdiccional del Poder Judicial es la relativa a la materia electoral, cuyas decisiones corresponden exclusivamente al Tribunal Supremo de Elecciones, según el artículo 103 constitucional (Poder Judicial de Costa Rica, 2016).

1.1.1.4 Historia de la institución

Una vez recibida la noticia de la independencia, el pueblo costarricense se organizó políticamente para formar un gobierno propio. El 1° de diciembre de 1821, representantes de diferentes ciudades y pueblos formularon el primer documento constitucional de Costa Rica llamado Pacto de Concordia, en este se estableció la Junta Suprema Gubernamental para ejercer funciones de gobierno, se creó también un tribunal para administrar la justicia, conforme a las Leyes de Indias. En 1824 se da la división del estado en tres poderes: Ejecutivo, Legislativo y Judicial, y dos años después se instala la Corte. Los primeros magistrados fueron nicaragüenses y guatemaltecos, posteriormente se integraron costarricenses, y es hasta 1842 que el cargo de Presidencia de la Corte es ejercido por un costarricense, don Manuel Mora Fernández.

La administración de justicia se impartió con base en las Leyes de Indias por un periodo de 20 años. En 1843 se anuncia la nueva Constitución Política, en la cual se incluye por primera vez el nombre de Corte Suprema de Justicia y se aumenta a siete el número de magistrados. En 1851 se emitió la Ley Orgánica del Poder Judicial, que estableció la forma de integrar el Poder Judicial por medio de miembros electos popularmente no menores de 30 años, jurisconsultos o con

notoria preparación en Derecho Civil. En 1859 se creó un nuevo cargo llamado "Co-juez Nato", conocido actualmente como Magistrado Suplente.

En 1869 la corte quedó dividida en dos Salas: Primera y Segunda. El número de magistrados aumentó a nueve. En 1871 se dispuso la elección de los magistrados por cuenta del Congreso y no del Presidente de la República. El 29 de marzo de 1887 se estableció por primera vez la independencia del Poder Judicial y el 6 de setiembre de 1937 se aprobó la Ley Orgánica del Poder Judicial, la cual reafirmó el principio de autonomía que este tiene en el ejercicio de sus funciones, y estableció por primera vez la división de la Corte Suprema de Justicia en una Sala de Casación, una Civil y otra Penal. El número de Magistrados pasa de 9 a 11.

En 1940 la Corte se reorganizó nuevamente en cuatro Salas de apelaciones, dos civiles, dos penales y una Sala de Casación. Desde este momento quedó conformada la Corte por 17 magistrados. El 8 de mayo de ese año quedó sin efecto la Constitución Política de 1871. Para finales de 1949 la Asamblea Constituyente emitió la Constitución que rige la vida institucional del país hasta el momento, posteriormente se dio una serie de innovaciones respecto a la organización del Poder Judicial. En 1982 la Asamblea Legislativa aprobó la Ley de Reforma donde se establece que los magistrados de la Corte Suprema de Justicia obtienen el derecho de elegir al presidente de esta misma y en 1989 se creó la Sala Constitucional integrada por 7 magistrados, con lo que aumenta a 22. El 1 de enero de 1998 entró a regir la Ley de Reorganización Judicial que vino a reformar el reordenamiento jurídico Penal y Procesal Penal. Asimismo estableció una nueva organización y competencia de las oficinas judiciales, reformando parcialmente la Ley Orgánica del Poder Judicial y derogando otras leyes y artículos, realizando reformas parciales a la Ley de Tránsito y Código Procesal Civil, entre otros (Poder Judicial de Costa Rica, 2016).

1.1.1.5 Estructura organizacional

La Figura 1 ilustra el organigrama del Poder Judicial de Costa Rica, el cual se encuentra dividido en tres ámbitos: ámbito jurisdiccional, ámbito auxiliar de justicia y ámbito administrativo. A continuación se detalla brevemente cada uno de estos ámbitos.

1.1.1.5.1 Ámbito jurisdiccional

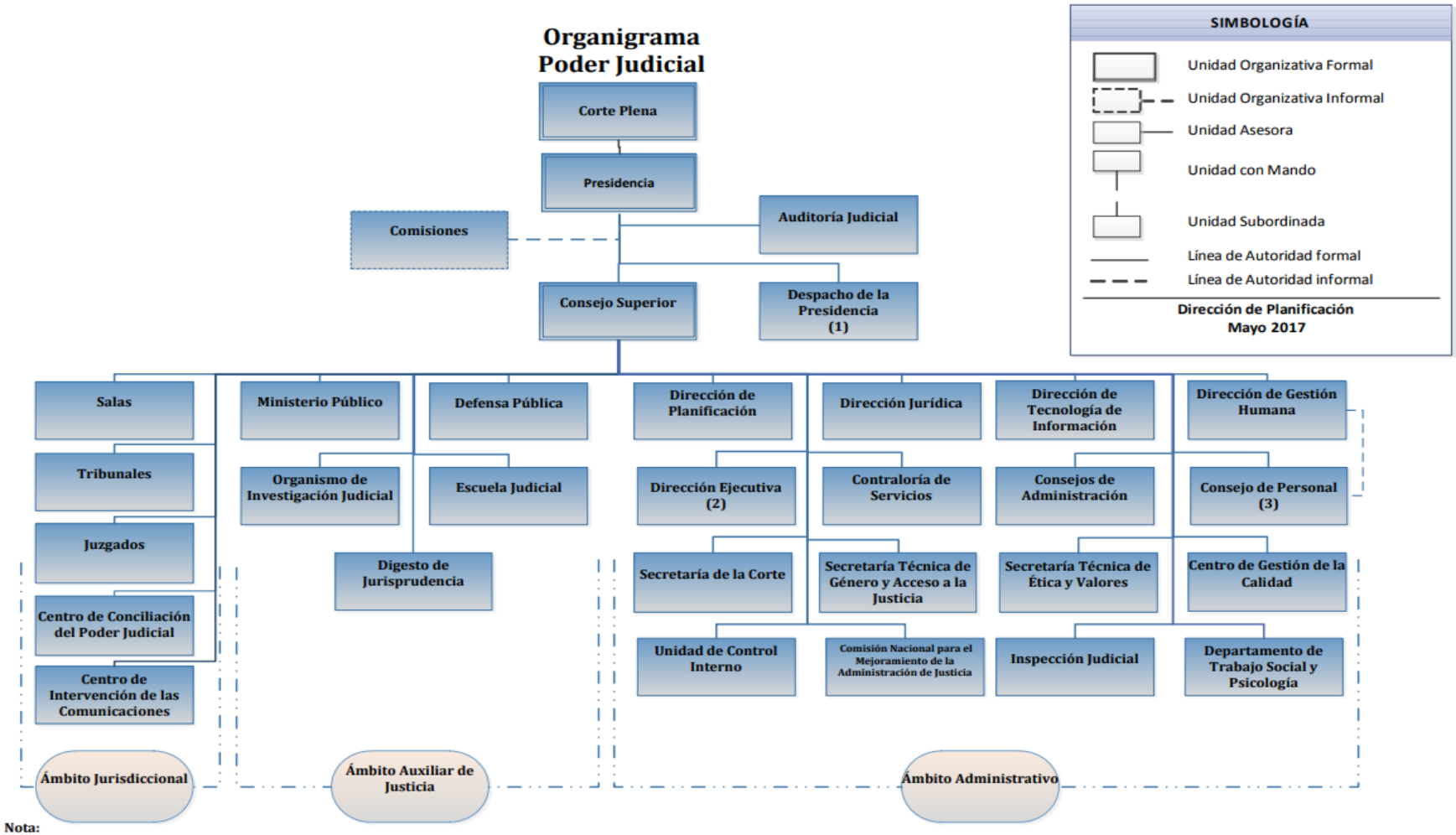
En este ámbito se encuentran las cuatro salas de la Corte Suprema de Justicia: Sala Primera, Sala Segunda, Sala Tercera y la Sala Constitucional. Posteriormente se encuentran los tribunales y juzgados, los cuales se dividen de acuerdo con la materia, el centro de conciliación y el centro de intervención de las comunicaciones.

1.1.1.5.2 Ámbito auxiliar de justicia

Está constituido por todos aquellos órganos y departamentos que coadyuvan diariamente en la labor de administrar justicia, en el cumplimiento de las funciones que constitucionalmente le están asignadas.

1.1.1.5.3 Ámbito administrativo

Le corresponde atender todos los aspectos logísticos, relacionados con el recurso humano, presupuesto, equipo, materiales, infraestructura, entre otros, del ámbito jurisdiccional y del ámbito auxiliar de justicia. En este ámbito encontramos la Dirección de Tecnología de Información.



Nota:

(1) Las dependencias adscritas al Despacho de la Presidencia se desempeñan en dos ámbitos. En el ámbito Jurisdiccional se encuentra el Centro de Apoyo, Coordinación y Mejoramiento de la Función Jurisdiccional; por su parte en el ámbito administrativo se tienen al Departamento de Prensa y Comunicación Organizacional; a la Sección de Gestión Administrativa-Despacho de la Presidencia; a la Sección de Cooperación y Relaciones Internacionales; así como a la Sección de Protocolo y Relaciones Públicas.

(2) La Dirección Ejecutiva tiene adscritos los departamentos de Proveeduría, Artes Gráficas, Servicios Generales, Financiero Contable y Seguridad; además la Biblioteca Judicial, el Archivo Judicial, el Registro Judicial, el Centro Infantil del Poder Judicial y las diferentes Administraciones Regionales.

(3) El Consejo de Personal es el órgano asesor del Consejo Superior en materia de administración de recursos humanos. Y por ello, debe mantener una estrecha relación con la Dirección de Gestión Humana.

Figura 1: Organigrama del Poder Judicial de Costa Rica

Fuente: www.poder-judicial.go.cr

1.1.2 Antecedentes de la Dirección de Tecnologías de la Información y Comunicación (DTIC)

La DTIC tiene sus orígenes en el año 1984 como una unidad de Informática que estaba adscrita a la Sección de Estadística. Dos años después se convirtió en una sección de informática y es en ese momento que se inician una serie de proyectos, como lo fue la adquisición de una computadora IBM/9375 modelo 60, con la que se incorporó el Sistema Integrado de Personal (SIP) y se adquirieron cerca de 100 computadoras. Posteriormente, se dio la primera instalación tanto de un equipo computacional en una oficina como la de una aplicación denominada Control de Expediente, cuyo objetivo era la automatización de los libros de la oficina.

Para el año 1993, la Sección de Informática pasó a ser el Departamento de Informática, y se mantuvo así por 18 años. En el año 94 se instaló la primera red de datos en el edificio de la corte y su cobertura fue de 80 microcomputadoras, adicionalmente se adquirieron 1000 computadoras en las cuales se instaló en varios despachos judiciales el sistema denominado Juzgado Mixto de Santa Cruz (JMS) en alusión al despacho donde se instaló el sistema por primera vez.

En el año 2011 la Corte Plena acordó en la sesión extraordinaria N° 28-11 Artículo XIX convertir el Departamento de Tecnología de la Información en una Dirección. Para el año 2014 se estableció que la Dirección de Tecnología de la Información dependerá del Consejo Superior y mantendrá una estrecha relación con la Dirección Ejecutiva, en virtud de que esta última mantendrá el apoyo logístico que se requiere para la implementación del trabajo que se desarrolla en materia de tecnología. Es de esta forma como se le confiere a la Dirección de Tecnología un rol de asesoría y participación en los temas estratégicos del Poder Judicial que anteriormente no poseía (Dirección de Tecnologías de Información y Comunicaciones Poder Judicial, 2015).

1.1.2.1 Misión de la DTIC

La misión de la DTIC del Poder Judicial de Costa Rica es:

> Proveer servicios de tecnologías de información y comunicaciones que brinden soporte a los procesos institucionales de Administración de Justicia y sus áreas de apoyo, dentro de un marco de referencia y calidad, y basados en las mejores prácticas de gestión tecnológica (Dirección de Tecnologías de Información y Comunicaciones Poder Judicial, 2015, p.44).

1.1.2.2 Visión de la DTIC

La visión de la DTIC del Poder Judicial de Costa Rica es:

> Ser un aliado estratégico que gestiona las TICs con excelencia, confianza, seguridad y oportunidad, para contribuir con la innovación y mejora de servicios institucionales, facilitar el acceso a la justicia a los diversos grupos de interés y apoyándonos en personal altamente capacitado y comprometido con los valores institucionales (Dirección de Tecnologías de Información y Comunicaciones Poder Judicial, 2015, p.45).

1.1.2.3 Estructura organizacional de la DTIC

La Figura 2 ilustra la estructura organizacional de la DTIC del Poder Judicial, la misma se compone de cinco secciones: Sección de Sistemas, Sección Informática Gestión, Sección de Apoyo a la Gestión Informática, Sección de Telemática y la Sección de Soporte Técnico.

Figura 2: Estructura Organizativa de la DTIC

Fuente: PETIC 2015-2020.

A continuación, se explica la función de cada una de las secciones mencionadas anteriormente:

1.1.2.3.1 Sección de Sistemas

Desarrollar y dar sostenibilidad a los sistemas de información del Poder Judicial, con el fin de apoyar los procesos de gestión judicial tanto en sus áreas sustantivas como áreas complementarias y de apoyo.

1.1.2.3.2 Sección Informática de Gestión

Esta sección se encarga de:

> Proveer servicios de desarrollo y aprovisionamiento de soluciones para el tratamiento de información de los procesos de la Administración de Justicia y sus diversos entes participantes, como el Ministerio Público, la Defensa Pública y los litigantes, para una adecuada gestión de casos (Dirección de Tecnologías de Información y Comunicaciones Poder Judicial, 2015, p.16).

1.1.2.3.2.1 Estructura de la sección de Informática de Gestión

La Figura 3 muestra la organización interna de la sección, la cual está compuesta por un jefe, un administrador de proyectos y dos coordinadores de proyectos quienes se encargan de coordinar a los equipos de normalización, equipo de inteligencia de negocios, desarrolladores, área de calidad y equipo de implantación.

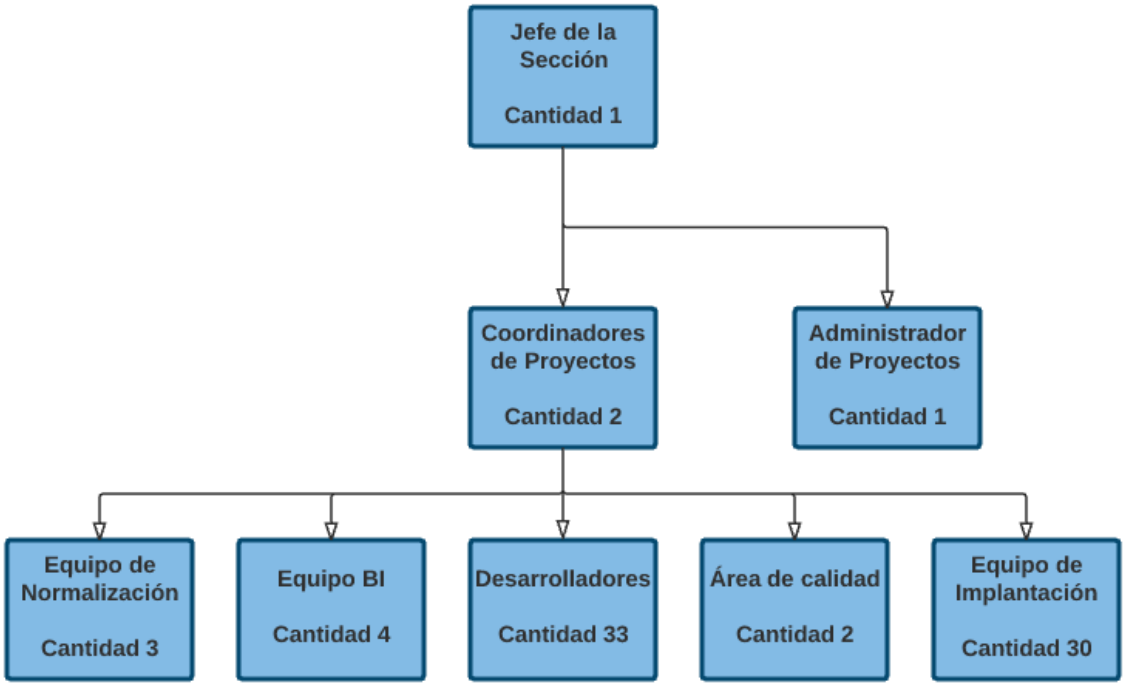


Figura 3: Organigrama de la sección informática de gestión

Fuente: Elaboración propia, 2017.

Seguidamente, se explica la función de cada uno de los equipos y áreas que conforman la sección de Informática de Gestión:

1.1.2.3.2.1.1 Equipo de normalización

El equipo de Normalización de Formatos Jurídicos tiene la responsabilidad de administrar, modificar e incluir los datos autorizados por la Sección de Estadística del Departamento de Planificación en los catálogos de la Plataforma Jurídica del Sistema Costarricense de Gestión de Despachos Judiciales, Agenda Cronos y otros sistemas, así como realizar mejoras a los documentos de todos los Despachos Judiciales de conformidad con las directrices emanadas por el Consejo Superior y la Comisión de Normalización. Preparar los formatos jurídicos que se utilizan en los Despachos Judiciales a nivel nacional, darles mantenimiento y atender problemas que se presenten en la ejecución de los mismos. Preparar la documentación necesaria que se utilizará para el manejo adecuado de la infraestructura jurídica y los formatos jurídicos a nivel nacional. Actualmente se encuentra conformado por tres personas.

1.1.2.3.2.1.2 Equipo de inteligencia de negocios

Es el equipo encargado de entender los sistemas transaccionales, intercalarlos con las “reglas de negocio” y de manera fiable, mostrar la información de forma ágil a las jefaturas, ya sea con indicadores, informes, espacios de información, etc.

1.1.2.3.2.1.3 Desarrolladores de software

Son profesionales de informática que se encargan de realizar análisis, diseño y desarrollo de proyectos nuevos o de dar mantenimiento a las aplicaciones que el área tiene a cargo. Actualmente se tienen 33 desarrolladores.

1.1.2.3.2.1.4 Área de calidad

Esta área se encuentra compuesta por dos profesionales en informática cuya función principal es la de realizar diferentes tipos de pruebas de software a los sistemas existentes o proyectos nuevos que la sección tiene a cargo, con el objetivo de asegurar la calidad del software. Entre los tipos de pruebas que esta área realiza se encuentran: pruebas de interfaz, pruebas de seguridad, pruebas de integridad de base de datos, pruebas de instalación, pruebas funcionales, pruebas de usabilidad, pruebas de regresión y pruebas de documentación.

1.1.2.3.2.1.5 Equipo de implantación

Equipo encargado de implantar los sistemas del área de informática de gestión en los diferentes circuitos judiciales del país. Se encargan de implementar toda la infraestructura necesaria para el uso de los sistemas, capacitan al personal y dan seguimiento para asegurar el uso correcto de los sistemas.

1.1.2.3.2.2 Proyectos de la sección

La Figura 4 ilustra los sistemas que la sección de Informática de Gestión tiene a cargo como lo son: aplicaciones web, aplicaciones cliente/servidor, aplicaciones móviles, aplicaciones *middleware*, aplicaciones *standalone* y las aplicaciones de inteligencia de negocios (BI).

1.1.2.3.2.3 Metodología de desarrollo Scrum en el Área de Informática de Gestión

En el año 2015 el Área de Informática de Gestión recibió una inducción sobre la metodología Scrum, la cual fue impartida por la empresa Arc1lab (Azofeifa, 2015). A finales de ese mismo año se incorporó el uso de la metodología Scrum específicamente para el desarrollo del proyecto SIAGPJ, el cual “surge como un proceso de reingeniería con el objetivo de desarrollar un único aplicativo que abarque todas las funcionalidades de tramitación de expedientes incluidas en los aplicativos soportados por el Área de Informática de Gestión” (Árticulo de Interés, 2016).

El proyecto SIAGPJ actualmente cuenta con catorce desarrolladores, dos personas del área de calidad, un administrador de proyectos y dos coordinadores de desarrollo de software. Poco a poco el área ha tratado de disminuir el desarrollo de mejoras a los sistemas existentes, ya que se pretende que el SIAGPJ reúna muchas de las funciones que actualmente se encuentran dispersas en los diferentes sistemas de la Figura 4.

La coordinadora de desarrollo, quien además representa el rol de dueño del producto en el proyecto SIAGPJ, establece que la metodología Scrum a pesar de su uso en la sección no se encuentra formalmente documentada por la DTIC, pero sí se establece y se documenta el uso de dicha metodología en el estudio de factibilidad del proyecto SIAGPJ. Azofeifa (2015) indica: "Para lograr materializar este proyecto se propone adoptar una de las metodologías de desarrollo ágil de aplicaciones, en este caso: Scrum" (p.45).

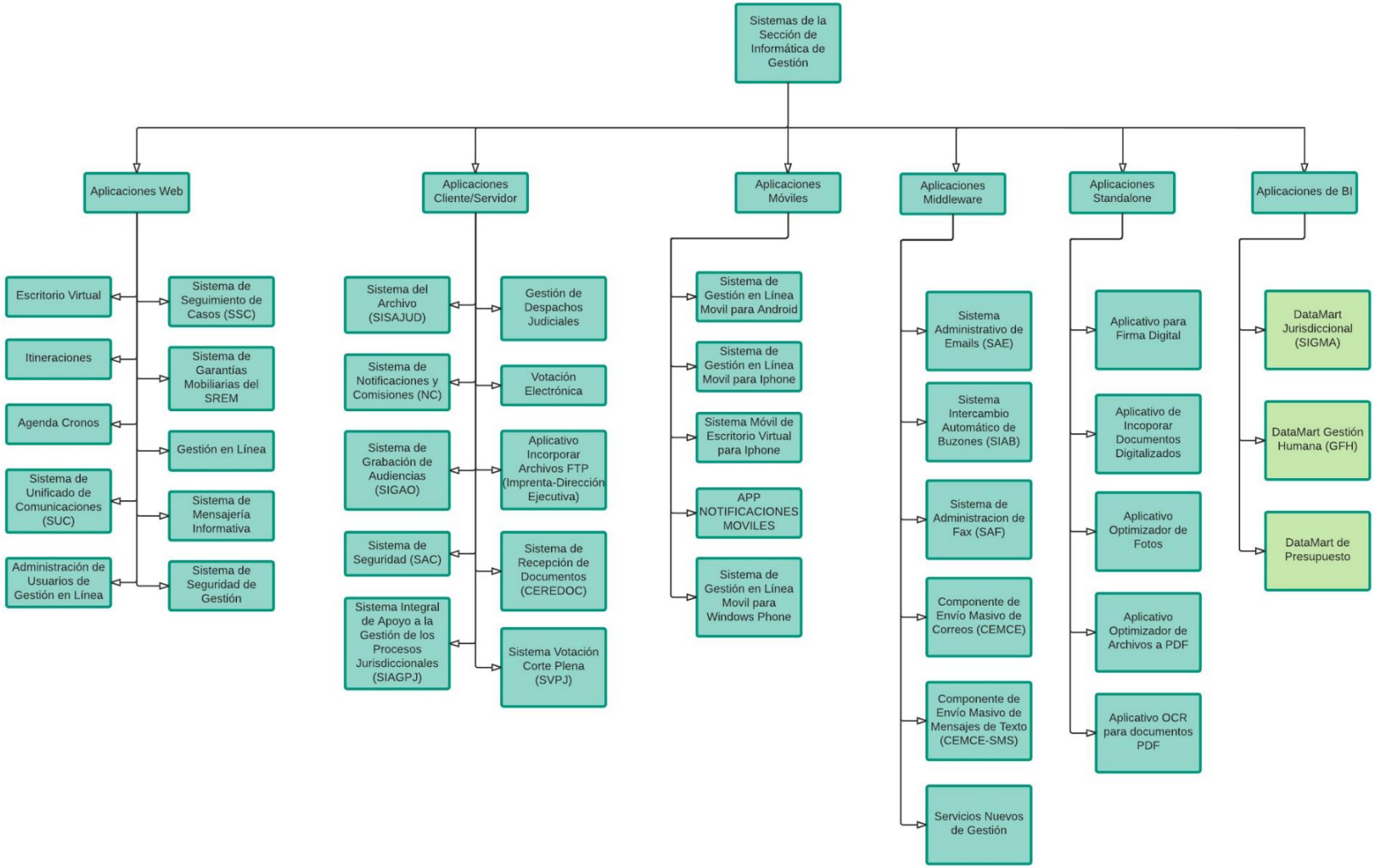


Figura 4: Sistemas a cargo de la sección de informática de gestión

Fuente: Elaboración Propia, 2017.

1.1.2.3.3 Sección Apoyo a la Gestión

Esta sección se encarga de atender incidentes y solicitudes de servicio relacionadas con las Tecnologías de la Información y comunicaciones, que se les presentan a los funcionarios y servidores del Poder Judicial. Adicionalmente llevan el control de toda adquisición, donación, instalación o movimiento de licencias (Dirección de Tecnologías de Información y Comunicaciones Poder Judicial, 2015).

1.1.2.3.4 Sección Telemática

La DTIC del Poder Judicial (2015) establece:

> El objetivo de esta Sección es Incorporar las tecnologías y las herramientas telemáticas y de comunicaciones en la operación y gestión del quehacer de la Administración de Justicia, a través de productos y servicios institucionales integrados, seguros, acordes con estándares de calidad y modernos que permitan mejorar e innovar los procesos jurisdiccionales, administrativos y auxiliares de justicia (p.15).

1.1.2.3.5 Sección Soporte Técnico

La sección de soporte técnico tiene como objetivo: “disponer de las condiciones adecuadas en el tema de infraestructura tecnológica de hardware y software, para ofrecer la plataforma necesaria para la prestación de servicios de TI y su respectiva continuidad” (Dirección de Tecnologías de Información y Comunicaciones Poder Judicial, 2015, p.15).

1.1.3 Justificación del proyecto

La calidad del software es un tema con el que todas las organizaciones que desarrollan aplicaciones informáticas deben lidiar, es por esto que muchas empresas adoptan metodologías ágiles para mejorar sus procesos. La calidad del software es uno de estos procesos.

El uso de la metodología Scrum en el Área de Informática de Gestión ha generado mejoras en cuanto a la productividad del equipo de desarrollo y el manejo de los requerimientos cambiantes (Dirección de Tecnologías de Información y Comunicaciones Poder Judicial, 2016). Sin embargo, acelerar la entrega del producto, la cual es la principal razón de las organizaciones que adoptan una metodología de desarrollo ágil (VersionOne, 2016) y tener un producto de calidad que pueda ser entregado al usuario en cada *sprint* (Schwaber, 2004), son aspectos importantes que a la fecha el área no ha logrado alcanzar. El equipo de calidad es un elemento fundamental para lograr la entrega continua y temprana de software valioso.

Para el mes de agosto del año 2017 se tenía que haber desarrollo un total de 85 módulos, es decir un 77% de avance del proyecto, sin embargo actualmente el porcentaje de avance del proyecto es de un 23%; por otra parte el primer entregable que se envió al área de calidad representa un atraso en tiempo de 16 meses (Rímola & Azofeifa, 2017). Algunas de las razones del atraso del proyecto de acuerdo con Rímola (2017) son: el equipo desarrolla en tecnologías nuevas, lo que para muchos desarrolladores representa una curva de aprendizaje, algunos desarrolladores no se encuentra a tiempo completo en el proyecto, y tener un equipo de calidad aislado al proceso de desarrollo en cada *sprint* dificulta la entrega de pequeñas liberaciones funcionales.

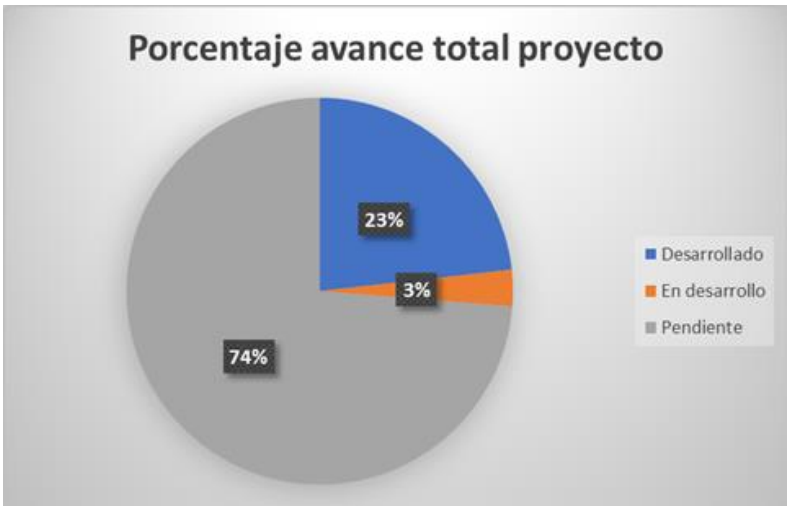


Figura 5: Porcentaje de avance proyecto SIAGPJ

Fuente: Informe Gerencial de Avance Sistema Gestión Nuevo, Agosto 2017.

Para que el proceso de desarrollo ágil sea exitoso, las pruebas de software "deben estar estrechamente enfocadas en las metodologías ágiles para cumplir con los plazos y servir mejor a los usuarios" (Varhol, 2010, p.6). Además las pruebas de software deben ser parte de un proceso metodológico de control de calidad que proporcione la evidencia adecuada de que el producto cumple con los requerimientos.

Por otra parte, la adopción de una metodología de desarrollo ágil como Scrum involucra una serie de cambios significativos en la forma de trabajo de cada miembro del equipo. La organización *International Software Testing Qualifications Board* [ISTQB] (2016) nos indica que un *tester* involucrado en un proyecto ágil trabajará diferente a como usualmente trabaja en un proyecto tradicional, y que además los miembros de control de calidad deben formar parte del equipo de desarrollo, siguiendo los valores del manifiesto ágil.

El objetivo de un equipo de calidad de software en una metodología ágil es evitar que se produzcan errores más que encontrarlos, por esto la importancia de la integración entre el equipo de desarrollo y el de calidad (Méndez, 2015). Por lo anterior, se desarrolla este proyecto para que

el equipo de calidad del área de Informática de Gestión pueda contar con una metodología de control de calidad adaptada a la metodología de desarrollo Scrum, que permita mejorar el proceso de control de calidad a través de técnicas ágiles, procedimientos, herramientas y métodos ágiles con base en las mejoras prácticas actuales logrando entregar software de calidad en el tiempo establecido a los usuarios internos y externos de la institución. De esta forma, contribuye al Poder Judicial mediante una de las metas de la DTIC presente en su plan estratégico 15-20, la cual es: "Entregar los proyectos tecnológicos dentro del tiempo, presupuesto y recursos estimado" (Dirección de Tecnologías de Información y Comunicaciones Poder Judicial, 2015, p.48).

1.2 DEFINICIÓN DEL PROBLEMA

Actualmente el área de calidad no cuenta con procedimientos para el planeamiento, ejecución y reporte de las pruebas bajo la metodología de desarrollo Scrum, lo que ha generado una serie de efectos negativos que a continuación se detallan:

- No se realiza una planeación para la ejecución de las pruebas. Cada integrante de calidad prueba el *software* sin conocer el alcance de las pruebas, los tipos de pruebas que debe aplicar o la prioridad que deben tener las pruebas. Siempre van a existir funcionalidades más críticas de probar que otras, sin embargo, al no realizar esta planeación y al estar en un entorno ágil, donde debido a las etapas definidas el tiempo es limitado, se presenta que el equipo de calidad pierde el tiempo probando minuciosamente partes del sistema que no son tan críticos. Se compromete al final las pruebas de lo que realmente tiene más peso en el sistema.
- Las pruebas realizadas no están basadas en casos de prueba previamente documentados, lo que genera que no se tenga evidencia de que el sistema cumple con los requerimientos. Al

finalizar las pruebas, el coordinador del área de calidad simplemente envía un correo a la jefatura y coordinadores del proyecto adjuntando un documento (ver Anexo 3: Plantilla control de aprobación de pruebas) en el que se indican las pruebas realizadas, pero no se tiene con certeza lo que realmente se probó o cuántos errores fueron encontrados.

- En muchas ocasiones los errores encontrados por el área de calidad son reportados al desarrollador personalmente, esto es un problema ya que al no registrar el error no se refleja la labor del *tester* ni tampoco se le puede dar un debido seguimiento al error.
- El equipo de calidad actualmente no realiza pruebas de rendimiento, carga o estrés, lo que afecta la calidad del software.

Por otra parte, el proceso de control de calidad actual no se encuentra alineado con la metodología de desarrollo Scrum, lo cual ocasiona que se presenten las siguientes situaciones:

- El equipo de calidad se encuentra aislado del proceso de desarrollo en cada *sprint*, lo que ocasiona la detección tardía de errores, aumentando las fases de estabilización que al final generan un re-trabajo para el área de calidad. Adicionalmente ocasiona brechas de comunicación entre el equipo de desarrollo y el equipo de calidad, lo cual provoca que este último reporte errores comportamientos que al final no lo son.
- El equipo de desarrollo y calidad no se encuentran sincronizados en los *sprints*, por cuanto los desarrolladores avanzan conforme a la metodología construyendo componentes individuales funcionales, pero dichos componentes no son inmediatamente probados por el equipo de calidad. Esto genera que la etapa de pruebas represente un efecto similar al conocido “cuello de botella” dado que la cantidad de desarrollo a la entrada de la fase de pruebas tiene tendencia a aumento, pero no así la salida de esta fase.

- La falta de experiencia y capacitación de la metodología Scrum hace que se sigan procedimientos de un desarrollo en cascada donde la etapa de pruebas se deja al final, y se compromete la calidad del software.

La situación descrita anteriormente genera retrasos en la entrega del producto, por cuanto es claro que no existe un alineamiento en los tiempos de ambas partes del proceso en la entrega y salida de sus productos. Por otra parte, al no tener una metodología para el control de calidad del software, no se tiene evidencia clara de las pruebas realizadas durante el *sprint*, por cuanto se desconoce si realmente el incremento de software cumple con los requerimientos. Adicionalmente, la calidad del software se ve afectada por la no realización de pruebas rendimiento, carga y estrés. Y finalmente aumenta el presupuesto del proyecto, dado que las fases de estabilización del producto tienden aumentar.

Por lo anterior, al realizar el análisis para este proyecto se determina que la propuesta de una metodología para el control de calidad del software adaptada a la metodología de desarrollo Scrum permitiría la inclusión de actividades proclives a una mejora en el tiempo, además de actualizar al Poder Judicial en las tendencias actuales.

1.3 OBJETIVOS DE LA INVESTIGACIÓN

1.3.1 Objetivo general

- Proponer una metodología de control de calidad del software adaptada a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el área de informática de Gestión del departamento de la DTIC del Poder Judicial.

1.3.2 Objetivos específicos

- Definir la situación actual del proceso de control de calidad del software que se realiza en los proyectos desarrollados bajo Scrum del Área de Informática de Gestión.
- Definir la brecha entre el proceso de control de calidad actual del Área de Informática de Gestión contra lo establecido en las metodologías de calidad de software enfocadas a ambientes ágiles.
- Diseñar un marco de trabajo de control de la calidad de software conforme a la metodología Scrum y adaptado al proceso de desarrollo ágil de la institución.
- Implementar un plan piloto de la metodología ágil de control de calidad en el área de informática de gestión para el proyecto SIAGPJ.

1.4 ALCANCES Y LIMITACIONES

1.4.1 Alcances

El alcance del proyecto consiste en la elaboración de una metodología de control de calidad adaptada a la metodología de desarrollo Scrum. Una vez realizada la propuesta se llevará a cabo un plan piloto en el Área de Informática de Gestión del II Circuito Judicial de San José.

A continuación se detallan una serie de entregables que están relacionados con los objetivos planteados:

- El primer entregable del proyecto es un diagnóstico de la situación actual del proceso de pruebas en el proyecto SIAGPJ, el cual incluye los resultados de la encuesta, entrevistas y lo que se haya captado a través del método de la observación. La información recolectada se utilizará para realizar el próximo entregable.

- El segundo entregable del proyecto comprende el análisis de brechas, donde se identificará de acuerdo con la información recopilada en el primer entregable las falencias del proceso actual contra lo establecido por la metodología Scrum. Lo anterior permitirá definir una metodología de control de calidad pero enfocada a la metodología de desarrollo.
- El tercer entregable del proyecto es el diseño de un marco de trabajo de control de calidad del software basado en la metodología de desarrollo Scrum y adaptado al proceso de desarrollo ágil de la institución. Este marco incluirá roles, responsabilidades del equipo de calidad, actividades, herramientas, pasos a seguir, métricas y demás elementos que componen un marco de trabajo de *testing*.
- El cuarto entregable del proyecto es la implementación de un plan piloto en el área de informática de gestión de Goicoechea. Allí se realizará una planificación, coordinando con la jefatura del área, y los integrantes del proyecto SIAGPJ, seguidamente se ejecutará el plan piloto preferiblemente en un *sprint* y por último se documentarán los resultados obtenidos.

1.4.2 Limitaciones

- A pesar de que el proyecto en el capítulo dos describe diferentes tipos de pruebas, es importante mencionar que en la etapa de implementación del plan piloto no se aplicarán todos los tipos de pruebas mencionados, ya que esto depende del entregable que se vaya a probar en el momento que se ejecute el plan piloto.
- El área de Informática de Gestión se basa en la metodología Scrum para el desarrollo de proyectos, sin embargo, este marco de trabajo no se encuentra formalmente documentado por la DTIC del Poder Judicial.

1.5 CRONOGRAMA DE ACTIVIDADES

A continuación se detallan las actividades y los tiempos necesarios para la realización del presente proyecto universitario, además las fechas clave de entregas:

Tabla 1: Cronograma de actividades

Nombre de tarea	Duración días	Comienzo	Fin
Seminario	56	08/05/2017	03/07/2017
Elaboración capítulo I	13	09/05/2017	22/05/2017
Elaboración capítulo II: Marco Teórico	20	23/05/2017	12/06/2017
Elaboración capítulo III: Marco Metodológico	13	13/06/2017	26/06/2017
Ajustar el anteproyecto del seminario al formato de la Escuela de Ingeniería Informática	9	03/07/2017	12/07/2017
Revisión del anteproyecto por parte del tutor	12	17/07/2017	29/07/2017
Realizar modificaciones de los Capítulos I, II y III de acuerdo con las recomendaciones realizadas por el tutor	77	30/07/2017	15/10/2017
Realizar y aplicar la encuesta y la entrevista.	8	16/10/2017	24/10/2017
Desarrollo Capítulo IV: Diagnóstico de la situación actual	25	25/10/2017	19/11/2017
Desarrollo Capítulo V: Análisis de la brecha	21	20/11/2017	11/12/2017
Desarrollo Capítulo VI: Propuesta del proyecto	27	12/12/2017	08/01/2018
Desarrollo Capítulo VII: Implementación plan piloto	15	08/01/2018	23/01/2018
Desarrollo Capítulo VII: Conclusiones y recomendaciones del proyecto	2	23/01/2018	25/01/2018
Revisión del tutor y correcciones	24	26/01/2018	19/02/2018
Ultimo día de entrega del proyecto con el visto bueno del tutor	1	20/02/2018	20/02/2018
Inicio de revisión y aprobación por parte del lector	Por definir	23/02/2018	Por definir
Correcciones en caso de presentarse	Por definir	Por definir	20/05/2018
Último día para entrega de informes finales con el visto bueno del tutor, lector, filólogo y declaración jurada	1	20/05/2018	20/05/2018
Inicio de defensa del proyecto	1	23/05/2018	23/05/2018

Fuente: Elaboración propia, 2017

CAPÍTULO II

2. MARCO TEÓRICO

2.1 TEORÍAS REFERENTES A LA TEMÁTICA A DESARROLLAR

A continuación se describen algunas teorías referentes a la temática a desarrollar en este proyecto.

En el año 2014 en la empresa Intel en Costa Rica se realizó un proyecto llamado “Propuesta de un modelo metodológico para la planificación y la gestión de pruebas del control de calidad de software” realizada por un estudiante de la Universidad Fidélitas. El mismo explicaba que la problemática radica en que el área de control de calidad no contaba con una metodología para la realización de pruebas de software, lo que generaba en ese momento un alto consumo de tiempo y de recursos (Garita, 2014).

Por otra parte, en el año 2015 en la empresa New Line Consultants S.A. se realizó un proyecto que proponía el mejoramiento de proceso de administración de la calidad del software basado en el estándar ISTQB, la problemática de esta empresa era la ausencia de una metodología para el aseguramiento de la calidad (Villalobos, 2015).

La calidad del software en Scrum es un tema que cada vez más preocupa a los interesados de un proyecto. En el año 2013 se realizó una tesis llamada *Software Quality Assurance in Scrum* por Tiisetso Khalane de la Universidad de Ciudad del Cabo de Sudáfrica, la misma expone la problemática de la falta de orientación en cuanto a estrategias, herramientas y técnicas de aseguramiento de la calidad en Scrum, y que al no tener personal dedicado en un proyecto de Scrum se presentan desafíos como demandas de capacidad, problemas de calidad y pruebas y la falta de experiencia en pruebas (Khalane, 2013).

El experto en Scrum Damian Buonamico, en su artículo llamado *Estrategias de Testing en Equipos Scrum* elaborado en el año 2016, expone una serie de estrategias que encontró en equipos de desarrollo de software. Una de estas estrategias se acerca un poco a la problemática que se expone en este proyecto, la cual se muestra en la Figura 6 (Buonamico, 2016).

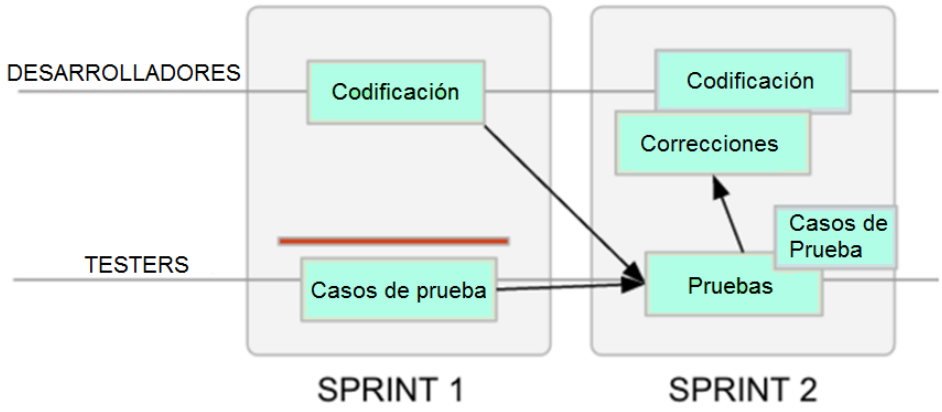


Figura 6: Ciclo de dos sprint estilo cascada

Fuente: www.caminoagil.com. Adaptado al español.

El equipo de desarrollo programa un conjunto de funcionalidades, pero en ese mismo *sprint* estas funcionalidades no son probadas. Buonamico (2016) nos indica que esta estrategia es anti-scrum, ya que el *sprint* no termina con software terminado ni con software potencialmente entregable. Algo similar sucede actualmente en el proyecto SIAGPJ, a diferencia que en dicho proyecto pasa más de un *sprint* en el que no se aplican pruebas. Adicionalmente la empresa Knight Errant, quien proporciona servicios de *testing* y soluciones de TI, indica que esta estrategia va en contra de lo que es una metodología ágil ya que crea separación entre el equipo de pruebas y el resto del equipo Scrum, reduce la comunicación, y retrasa el tiempo para dar una realimentación del código, ya que los desarrolladores han pasado a desarrollar una nueva funcionalidad, entre otros (Knight Errant, 2012).

2.2 MARCO TEÓRICO REFERENCIAL

A continuación, en este capítulo se mencionan los términos relevantes a los conceptos utilizados durante el proceso de desarrollo del proyecto, con el objetivo de contar con un conocimiento previo, y sumamente necesario para los capítulos futuros.

2.2.1 Ingeniería del Software

El experto Sommerville (2011) establece que:

> La ingeniería del software es una disciplina de la ingeniería que se interesa por todos los aspectos de la producción de software desde las primeras etapas de la especificación del sistema, hasta el mantenimiento del sistema después de que se pone en operación (p.7).

De la definición anterior, Sommerville (2011) especifica que es una disciplina de la ingeniería debido a que los ingenieros aplican teorías, métodos y herramientas para hacer que algo funcione. Además comprende procesos técnicos del desarrollo de software, procesos de gestión de proyectos de software y el desarrollo de métodos, herramientas y teorías que apoyan el proceso de desarrollo de software.

El ingeniero Pressman (2010) establece que la ingeniería del software es una tecnología que se presenta en estratos que abarca un proceso, métodos y herramientas, donde la base de estos estratos es el compromiso con la calidad (véase Figura 7).

Figura 7: Estratos de la ingeniería de software

Fuente: Libro Ingeniería del Software – Un enfoque práctico

La ingeniería del software comprende un proceso de cuatro actividades que permiten el desarrollo o evolución del software:

1. **Especificación del software,** donde los clientes e ingenieros definen el software a producir y las restricciones sobre su operación.
2. **Desarrollo del software,** donde el software se diseña y programa.
3. **Validación del software,** donde el software se válida para asegurar qué es lo que el cliente requiere.
4. **Evolución del software,** donde el software se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado. (Sommerville, 2011, p.28)

Por otra parte, el experto en ingeniería de software Pressman (2010) indica otras actividades que se complementan con las mencionadas anteriormente y que se aplican a lo largo de un proyecto de software facilitando la administración, el control, la calidad, el cambio y el riesgo del proyecto:

a. **Seguimiento y control del proyecto de software:** Se evalúa el progreso comparándolo con el plan del proyecto, y se compara lo realizado con las metas y

planes para detectar desviaciones y así ejecutar acciones correctivas que permitan mantener el proyecto bajo control.

b. **Administración del riesgo:** Se evalúan los riesgos que puedan afectar el proyecto o la calidad del producto.
c. **Aseguramiento de la calidad del software:** Define y ejecuta las actividades necesarias para garantizar la calidad del software.
d. **Revisiones técnicas:** Se evalúan los productos del trabajo de ingeniería de software con el objetivo de descubrir y eliminar errores antes de que se propaguen a la siguiente actividad.
e. **Medición:** Establece mediciones del proceso, del proyecto y del producto con el objetivo de ayudar al equipo a entregar el software que satisfaga las necesidades del cliente.
f. **Administración de la reutilización:** Se definen criterios para volver a utilizar el producto del trabajo y se establecen mecanismos para el uso de componentes reutilizables.
g. **Preparación y producción del producto del trabajo:** Agrupa actividades requeridas para la creación de productos, como modelos, documentos, registros, formatos y listas.

2.2.2 Proyecto

El PMI (2013) establece: “Un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único” (p.3). Todo proyecto tiene su principio y fin definidos, en donde el fin de un proyecto puede darse por varios motivos, ya sea cuando se han conseguido los objetivos del proyecto, cuando el cliente desea terminar el proyecto, cuando se ha

identificado que los objetivos no se cumplirán, o cuando ya no exista una necesidad para que el proyecto continúe (PMI, 2013).

2.2.3 Proyecto de software

Un proyecto de software se compone de cuatro P: personal, producto, proceso y proyecto (Pressman, 2010).

Figura 8: Cuatro P de un proyecto de software

Fuente: Elaboración propia, 2017

2.2.3.1 El personal

Un proyecto de software se encuentra compuesto por participantes como gerentes ejecutivos, los cuales definen los temas estratégicos, gerentes de proyecto, quienes planifican, motivan, organizan y controlan a los profesionales que realizan el software, profesionales de la ingeniería del software, clientes, quienes especifican los requerimientos y usuarios finales, quienes harán uso del software a desarrollar. Es importante que el personal sea motivado y calificado para lograr el éxito del proyecto (Pressman, 2010).

2.2.3.1.1 Equipos ágiles

El desarrollo de software ágil ha surgido como una solución a muchos de los problemas que se dan en un proyecto de software. Las metodologías ágiles se enfocan en la satisfacción del cliente y la entrega incremental temprana de software, se tienen equipos ágiles motivados y auto organizados que tienen autonomía para planificar y tomar decisiones técnicas (Pressman, 2010).

2.2.3.2 El producto

En un proyecto de software se deben establecer los objetivos y el ámbito del producto ambos se deben definir en conjunto con los participantes del proyecto (Pressman, 2010).

2.2.3.3 El proceso

Para que un producto pueda ser creado se debe seguir un proceso que comprende un conjunto de actividades, tareas y acciones sin importar el tamaño o la complejidad del proyecto, el proceso debe adaptarse al personal y al producto (Pressman, 2010).

2.2.3.4 El proyecto

Un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto o servicio. Un proyecto de software se debe planear y controlar para poder manejar la complejidad (Pressman, 2010).

2.2.3.4.1 Ciclo de vida de un Proyecto

El PMI (2013) establece:

> El ciclo de vida de un proyecto es la serie de fases por las que atraviesa un proyecto desde su inicio hasta su cierre. Las fases son generalmente secuenciales y sus nombres y números se determinan en función de las necesidades de gestión y control de la

organización u organizaciones que participan en el proyecto, la naturaleza propia del proyecto y su área de aplicación (p.37).

Todos los proyectos sin importar su tamaño o complejidad pueden adaptarse dentro de la siguiente estructura del ciclo de vida (véase Figura 9):

- Inicio
- Organización y preparación
- Ejecución del trabajo
- Cierre

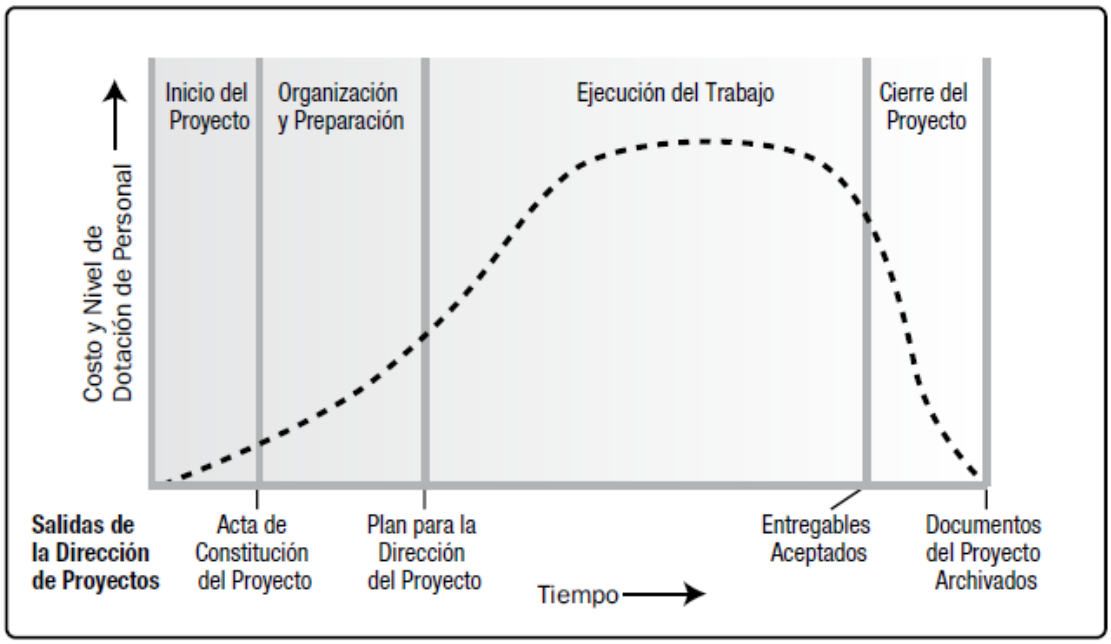


Figura 9: Niveles típicos de costo y dotación de personal durante el ciclo de vida del proyecto

Fuente: PMBOK ®, 2013.

2.2.4 Modelos de proceso de software

El experto Somerville (2011) establece que los procesos de software se podrían clasificar en dos tipos: dirigidos por un plan o como procesos ágiles, el primero se refiere a aquellos procesos en donde se definen todas las actividades por anticipado, un plan y el avance del proceso se mide

contra dicho plan, mientras que en el segundo la planeación es de forma incremental y se tiene un mejor manejo de los requerimientos cambiantes. Existen varios tipos de modelos, sin embargo este proyecto se enfoca en el de procesos ágiles.

2.2.5 Metodologías para el desarrollo de software

Una metodología para el desarrollo de software se refiere a un marco de trabajo utilizado para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información (CMS, 2005). Existe una gran variedad de metodologías que han surgido a través de los años, sin embargo para este proyecto solo se mencionará el modelo en cascada o tradicional y la metodología Scrum, ya que son necesarios para el desarrollo de los próximos capítulos del proyecto.

2.2.5.1 Metodología de desarrollo en cascada

Es un modelo de desarrollo considerado como el tradicional, el cual consiste de una serie de fases secuenciales donde cada fase debe completarse antes de iniciar la siguiente fase. En la Figura 10 se muestran las etapas del modelo clásico cascada.

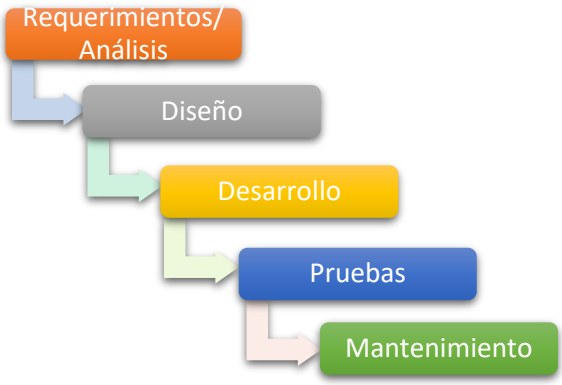


Figura 10: Modelo de desarrollo cascada

Fuente: Elaboración propia, 2017.

Como se puede apreciar en la Figura 10, la fase de pruebas se inicia una vez que la etapa de desarrollo se haya completado. Lo anterior ha generado que este enfoque sea blanco de críticas, ya que dejar el proceso de pruebas hasta el final genera como consecuencia un alto grado de riesgo en que el proyecto no cumpla con las expectativas del cliente o que no funcione correctamente. Como resultado de esto ocurre el retrabajo y la realización de las pruebas nuevamente, todo por la detección tardía de los defectos (Watkins, 2009).

El costo de corregir de los defectos del software aumenta conforme a las etapas, como se puede apreciar en la Figura 11.

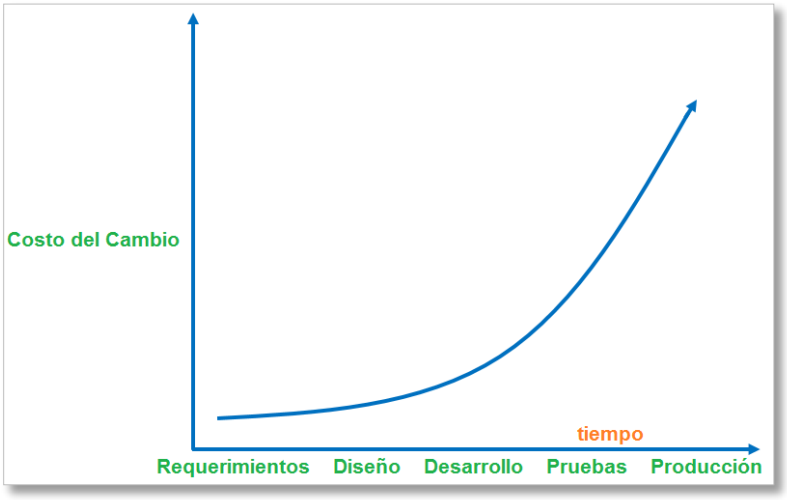


Figura 11: Costo de corrección de defectos

Fuente: Elaboración Propia, 2017.

Esta metodología de desarrollo también ha sido criticada por la falta de respuesta ante los requerimientos cambiantes durante el desarrollo del sistema, debido a que los requerimientos prácticamente son escritos en piedra en la fase de análisis y la entrega del sistema puede tomarse meses o inclusive años. Para el momento de entrega ya las necesidades del cliente son otras (Watkins, 2009).

Pressman (2010) afirma:

> Hoy en día, el trabajo de software es acelerado y está sujeto a una corriente sin fin de cambios (en las características, funciones y contenido de información). El modelo de la cascada suele ser inapropiado para este tipo de labor. No obstante, sirve como un modelo de proceso útil en situaciones en las que los requerimientos son fijos y el trabajo avanza en forma lineal hacia el final (p.34).

2.2.5.2 Metodología de desarrollo ágil

El desarrollo ágil de software se basa en el modelo de desarrollo incremental, el cual consiste en aplicar secuencias lineales en forma escalonada y al final de cada secuencia se genera un incremento de software ver Figura 12: Modelo IncrementalFigura 12.

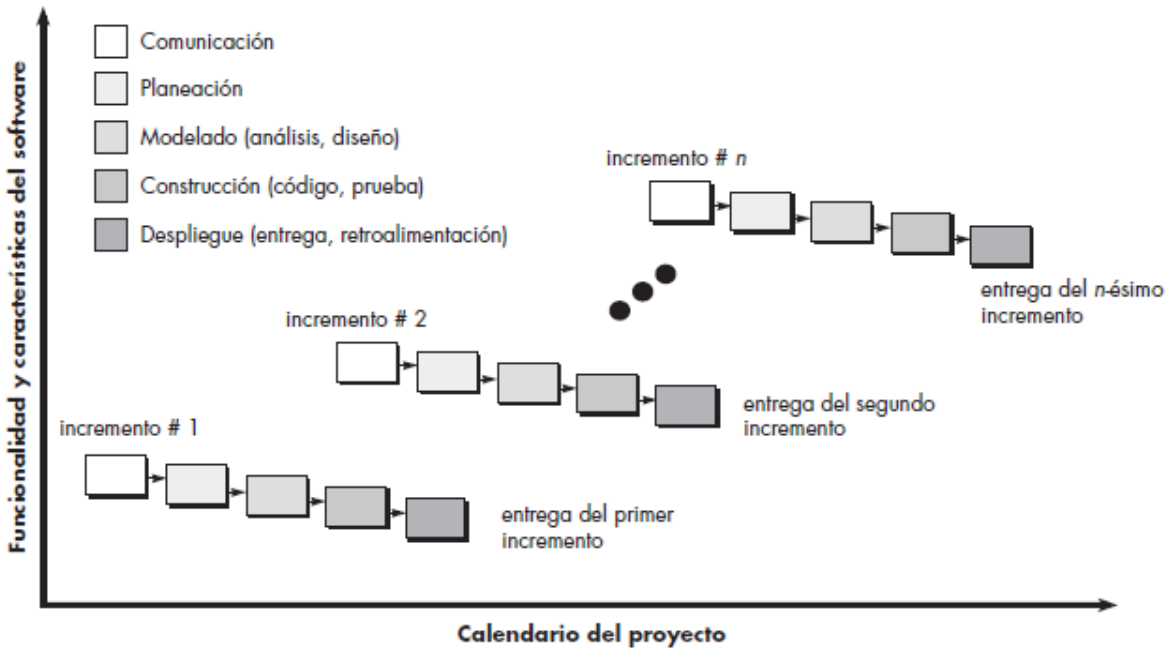


Figura 12: Modelo Incremental

Fuente: Pressman, 2010.

En el desarrollo de software el término ágil se utiliza para referirse a varias metodologías, prácticas o procesos. En el año 2001 un grupo de especialistas en desarrollo de software

establecieron principios comunes para los métodos de desarrollo como Extreme Programming (XP), Scrum y Feature Driven Development (FDD), entre otros. Como resultado de lo anterior se creó lo que se conoce actualmente como la Alianza Ágil o bien *Agile Alliance* y se estableció el Manifiesto Ágil, que son los valores bajo los cuales las metodologías ágiles trabajan (Alexander Menzinsky, 2016).

El Manifiesto Ágil estableció cuatro valores y doce principios para todas las metodologías ágiles. Los valores que conforman el Manifiesto Ágil son:

- Valorar más a los individuos y su interacción que a los procesos y las herramientas.
- Valorar más el software que funciona que la documentación exhaustiva.
- Valorar más la colaboración con el cliente que la negociación contractual.
- Valorar más la respuesta al cambio que el seguimiento de un plan (Alexander Menzinsky, 2016).

Las metodologías ágiles, aunque cada una de ellas tiene sus prácticas o procedimientos y fases diferentes, todas comparten las siguientes características: la documentación no es tan extensiva a diferencia de los desarrollos tradicionales, es un proceso de desarrollo incremental o iterativo y la comunicación, entre otros (Fuentes, 2014).

Una definición más clara de desarrollo ágil es la siguiente:

> Una metodología ágil es un proceso en donde la funcionalidad del software es desarrollada en un ciclo pequeño el cual es llamado como iteración. Esta funcionalidad es basada en características las cuales son también divididas en pequeñas secciones las cuales pueden ser fácilmente desarrolladas y probadas en iteraciones pequeñas (Pandey, 2014).

2.2.5.2.1 Scrum

Scrum es un marco de trabajo ágil iterativo que se ha utilizado para el desarrollo de productos complejos, lo anterior desde los noventas. Algunos expertos que han contribuido significativamente a la evolución de Scrum. Schwaber & Sutherland (2016) indican que “Scrum no es un proceso o una técnica para construir productos; más bien, es un marco de trabajo dentro del cual se pueden emplear varios procesos y técnicas.” (p.3).

Scrum se compone de equipos, roles, eventos, y artefactos, cada uno de esos elementos tienen un propósito específico y son esenciales para el éxito de esta metodología (Schwaber & Sutherland, 2016).

De acuerdo con ciertas encuestas realizadas en los últimos tres años, la principal razón por la que las organizaciones adoptan una metodología de desarrollo ágil es para acelerar la entrega del producto, Scrum es la metodología más utilizada (VersionOne, 2015).

2.2.5.2.1.1 Prácticas de Scrum

Scrum se compone de roles, eventos, y artefactos, cada uno de esos elementos tienen un propósito específico y son esenciales para el éxito de esta metodología (Schwaber & Sutherland, 2016). Seguidamente en la Figura 13 se muestra cuáles son los roles, eventos y los artefactos que conforman el marco de trabajo Scrum.

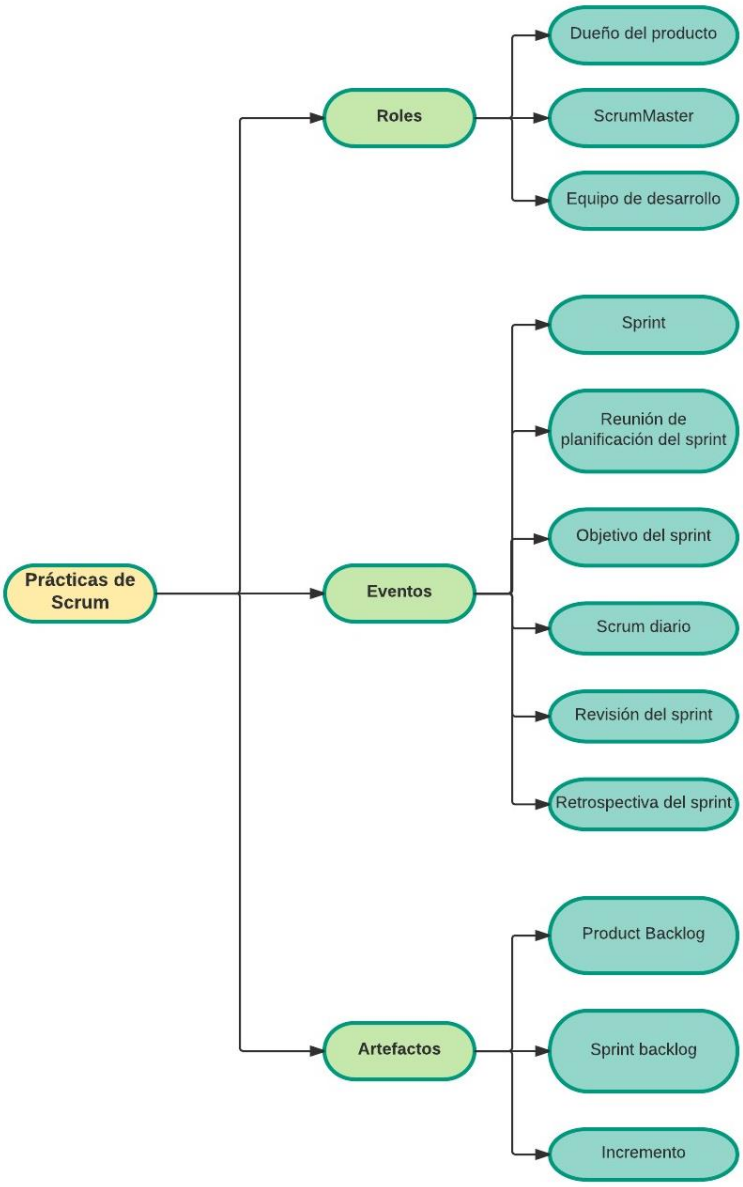


Figura 13: Prácticas de Scrum

Fuente: Elaboración propia, 2017.

2.2.5.2.1.1.1 Roles en Scrum

Scrum está conformado por tres roles principales: dueño del producto, el equipo de desarrollo y un Scrum Master. A continuación se detalla cada uno de estos roles.

2.2.5.2.1.1.1.1 Dueño del producto

Es el miembro del equipo que toma las decisiones, y conoce muy bien el negocio del cliente y su visión del producto. Es responsable de decidir cuáles funcionalidades construir y en qué orden se construirán, además mantiene y comunica a todos los participantes una visión clara de lo que el equipo de Scrum está tratando de lograr. En resumen, es responsable del éxito en general de la aplicación que se está desarrollando o manteniendo (Rubin, 2013).

El dueño del producto debe tener una serie de cualidades. Cohn (2010) indica cinco atributos que se describen a continuación: como primer atributo la 1) disponibilidad, debido a que es una metodología de desarrollo rápido cuando surgen preguntas o dudas es necesario que el dueño del producto esté disponible para dar las respuestas al equipo de forma oportuna. El próximo atributo es la 2) comprensión del negocio, de los clientes y usuarios, lo que ayuda a tomar mejores decisiones en lo que debe o no desarrollarse. El siguiente atributo es 3) comunicativo, debido a que el dueño del producto interactúa constantemente con los usuarios, socios y partes interesadas, adicionalmente debe escuchar a dichas partes así como también al equipo de desarrollo en cuanto a riesgos técnicos o desafíos del proyecto o recomendaciones que haga el equipo de desarrollo en cuanto a la priorización del trabajo basados en factores técnicos. Continuando con los atributos, el siguiente es 4) decisivo, ya que si el equipo de desarrollo se dirige al dueño del producto para exponer el problema, es porque requieren de una solución donde el dueño del producto debe tomar decisiones de gran peso. Por último 5) delegar o empoderado, es decir toma las decisiones y se hace responsable de esas decisiones, es alguien que debe estar lo suficientemente alto en la organización para recibir tal responsabilidad.

2.2.5.2.1.1.1.2 Equipo de desarrollo

El equipo de desarrollo son los que realizan el trabajo de entregar software potencialmente entregable al final de cada *sprint* (Schwaber & Sutherland, 2016). Muchas organizaciones dividen los roles de trabajo en equipos especializados, por ejemplo, en una empresa de desarrollo de software pueden estar divididos en analistas de sistemas, diseñadores, programadores, especialistas en base de datos, probadores de software, entre otros; cada uno de esos equipos entregan el trabajo uno al otro cuando está completo y prácticamente funcionan de forma independiente uno del otro. En Scrum en equipo de desarrollo debe realizar todos esos roles, por lo que se requiere un equipo que sea experto en todas esas áreas (Cohn, 2010). Es un grupo de personas multifuncionales que tienen las habilidades necesarias para entregar cada incremento del producto.

El experto Linz (2014) establece que el equipo requiere al menos de un miembro con experiencia en pruebas de software y que este se encuentre dedicado tiempo completo a las pruebas. Adicionalmente va ser responsable de la planeación, diseño y ejecución de las pruebas en todos los *sprints*.

2.2.5.2.1.1.1.3 Scrum Master

Los expertos Schwaber & Sutherland (2016) indican que: “El scrum master es responsable de asegurar de que Scrum se entienda y se promulgue. Los Scrum Masters hacen eso para asegurar de que el equipo de Scrum se adhiera a la teoría de Scrum, prácticas y reglas.” (p.6).

Adicionalmente el Scrum Master ayuda al dueño del producto, al equipo de desarrollo y a la organización de diferentes maneras. Los creadores de Scrum (Schwaber & Sutherland, 2016) indican que al dueño del producto le ayuda a encontrar técnicas para una administración efectiva de lista del producto, le ayuda a comprender y practicar la agilidad, entre otros; al equipo de

desarrollo le ayuda de forma que se creen productos de alto valor, también colabora a eliminar impedimentos para el progreso del equipo, entre otros; finalmente colabora a la organización con el entrenamiento y liderazgo para la adopción de Scrum en la organización, realiza cambios que incrementan la productividad del equipo Scrum, entre otros.

Linz (2014) indica que el Scrum master también es responsable de organizar las pruebas. Si las herramientas de pruebas o la infraestructura requieren de un mejoramiento o un reemplazo o si las habilidades necesarias no están disponibles dentro del equipo, el Scrum master es responsable de asegurar que esos impedimentos sean resueltos. (Linz, 2014).

2.2.5.2.1.1.2 Eventos de Scrum

La finalidad de los eventos en Scrum es crear regularidad y minimizar la necesidad de reuniones, todos los eventos son bloques de tiempo (*time-boxes*) los cuales tienen una duración máxima. Se dice que la falta de alguno de estos eventos que se explicarán a continuación reduce la transparencia y además imposibilita la inspección y adaptación de alguno de los aspectos del proceso, del producto o del progreso (Schwaber & Sutherland, 2013).

2.2.5.2.1.1.2.1 El *sprint*

Una metodología ágil, como bien se indicó anteriormente, trabaja en ciclos pequeños o iteraciones que en Scrum se llaman *Sprints*. La duración de un *Sprint* puede ser un mes o menos, en donde se crea un incremento de producto utilizable y potencialmente liberable (Schwaber & Sutherland, 2016).

"Los *sprints* contienen y consisten de la planificación del *sprint*, Scrum diarios, el trabajo de desarrollo, la revisión del *sprint* y el retrospectivo del *sprint*" (Schwaber & Sutherland, 2016, pag.8).

Un *sprint* siempre tiene una fecha de inicio y una fecha final y normalmente todos los *sprint* tienen la misma duración. Un *sprint* nuevo inicia inmediatamente cuando el *sprint* anterior finalizó (Cohn, 2010).

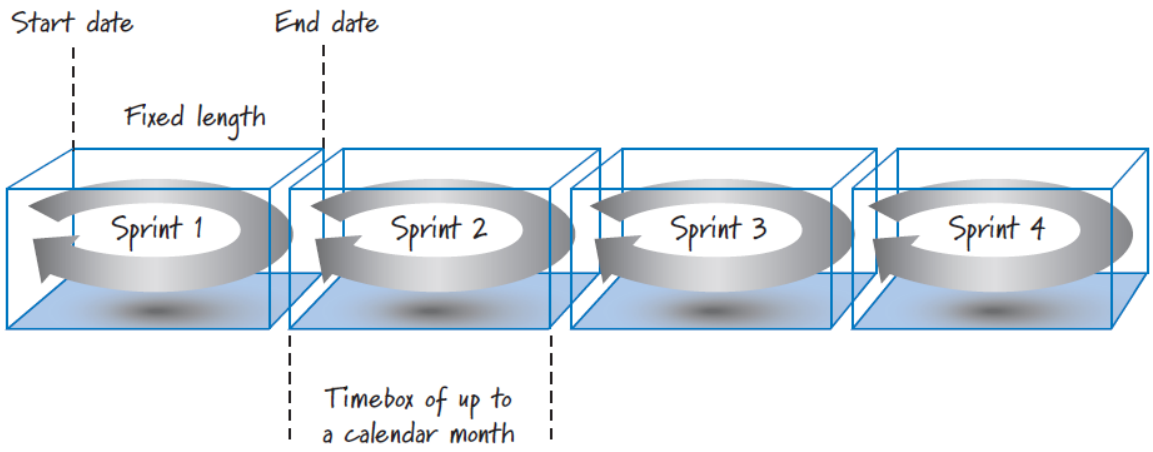


Figura 14: Características de un sprint

Fuente: (Rubin, 2013)

Adicionalmente en cada *sprint* se realizan todas las actividades necesarias para crear un incremento del producto, por supuesto que no todo, pero sí una parte. En la Figura 15 podemos ver que en cada *sprint* se pueden ver actividades como análisis, diseño, desarrollo, integración y pruebas (Rubin, 2013).

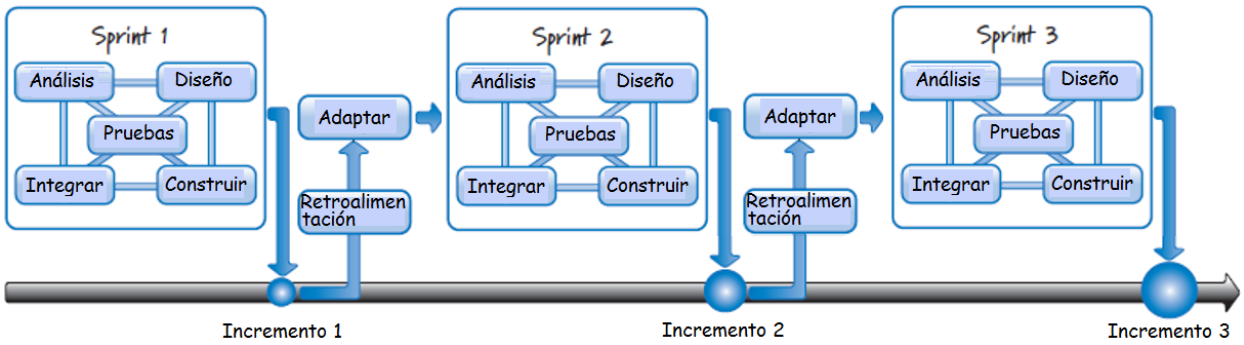


Figura 15: Actividades de un sprint

Fuente: (Rubin, 2013) Adaptado al español

2.2.5.2.1.1.2.2 Reunión de planificación de un *sprint*

En esta etapa se planea el trabajo que se realizará en el *sprint*, todo el equipo de Scrum se reúne, revisan el *product backlog* y establecen prioridades. Las funcionalidades a desarrollarse se descomponen en tareas y el equipo de desarrollo se encarga de establecer un estimado del tiempo normalmente en horas para cada tarea. “El descomponer *product backlog ítems* en tareas es una forma de diseño y planificación justo a tiempo para obtener las funcionalidades finalizadas” (Rubin, 2013, p.22).

El especialista en software Linz (2014) afirma: “El Scrum master tiene que asegurarse que el trabajo de las pruebas de software no se descuide ni sea olvidado” (p.38). Por otra parte, el profesional en QA y Scrum Master certificado Giardina (2013) indica una serie de tareas que el *tester* debe realizar durante la planificación del próximo *sprint*:

- Debe ser parte de la reunión de planificación para el próximo *sprint*. El *tester* necesita validar si el creará cualquier entorno complejo, o cualquier mejora importante para la automatización del marco de trabajo, antes de iniciar el próximo sprint (Giardina, 2013).
- El *tester* crea criterios de aceptación para las historias de usuario del próximo *sprint*, ayudando al analista del negocio o programadores mediante las sugerencias desde la perspectiva de QA con respecto a normas, experiencia del usuario, posibles problemas de rendimiento y errores futuros (Giardina, 2013).
- Crear los casos de prueba para el próximo *sprint*.

2.2.5.2.1.1.2.3 Ejecución del *sprint*

Una vez que todo el equipo de Scrum haya finalizado con la planificación del *sprint* y de que todos estén de acuerdo en el trabajo que viene para el próximo *sprint*, se inicia con el desarrollo de todas las tareas definidas en la planificación del *sprint*, posteriormente se establecen dichas

tareas como finalizadas o done como se indica en la mayoría de las literaturas. “El significado de *done* se refiere a que hay un alto grado de confianza en que todo el trabajo necesario para producir funcionalidades de buena calidad han sido completadas” (Rubin, 2013, p.23).

Durante la ejecución del *sprint* para el caso de los *testers*, el experto Giardina (2013) nos indica:

- Si la aplicación está dividida en capas, se tendrán APIs o servicios, por lo que se podría comenzar a escribir las pruebas antes de que los programadores terminen de programar.
- Los casos de prueba deberían ejecutarse manualmente una vez que la historia de usuario esté finalizada.
- Ser parte de la reunión de revisión, el objetivo es mostrar lo que se ha alcanzado en el actual *sprint*.
- Ser parte de la reunión retrospectiva, donde cada uno participara, identificando las cosas que se han realizado bien, que cosas mejorar y que acciones se deben aplicar.

2.2.5.2.1.1.2.4 Scrum diario

Es una actividad diaria de 15 minutos o menos, normalmente a la misma hora y lugar, donde se reúne el equipo de desarrollo para explicar lo realizado el día anterior, lo que se realizará el día de hoy y si existen impedimentos que me impiden el progreso. Esta reunión ayuda a que los miembros del equipo se mantengan actualizados en cuanto al progreso del *sprint* y optimiza la probabilidad de que el equipo de desarrollo logre el objetivo del *sprint*. En ocasiones los miembros del equipo se reúnen luego del Scrum diario para discutir detalles o ya sea para adaptar o replantear el resto del trabajo del *sprint* (Schwaber & Sutherland, 2016).

El Scrum master se encarga de que las reuniones se realicen cada día, pero es responsabilidad del equipo de desarrollo llevar a cabo la reunión diaria Scrum.

Los originadores de Scrum, Schwaber & Sutherland (2016), establecen que:

> Las reuniones diarias de Scrum mejoran las comunicaciones, eliminan otras reuniones, identifica impedimentos de desarrollo para eliminarlos, destacan y promueven la toma de decisiones rápidas y mejoran el nivel de conocimiento del equipo de desarrollo (p.11).

En resumen, la reunión diaria de Scrum es una actividad de inspección, sincronización y adaptación diaria que ayuda a un equipo autoorganizado realice su trabajo mejor.

2.2.5.2.1.1.2.5 Revisión del *sprint*

Es una reunión informal que se realiza al final del *sprint* donde se inspecciona el incremento o funcionalidad desarrollada y se adapta la lista de producto (*product backlog)* en caso de ser necesario. El equipo de Scrum y las partes interesadas (stakeholders) se reúnen para ver lo realizado en el *sprint*. Nuevamente la responsabilidad el Scrum Master en esta parte es asegurar que la actividad se lleve a cabo (Schwaber & Sutherland, 2016).

Los creadores del marco de trabajo Scrum, (Schwaber & Sutherland, 2016) establecen una serie de elementos que se realizan en esta reunión. Por ejemplo el dueño del producto explica los elementos de la lista de producto (*product backlog ítems)* que han sido terminados y cuáles no, también el equipo desarrollo discute lo que se llevó a cabo bien durante el *sprint*, los problemas que tuvieron y como esos problemas se resolvieron, entre otros.

2.2.5.2.1.1.2.6 Reunión de retrospectiva del *Sprint*

Ocurre normalmente después de la revisión del *sprint* y antes de la planificación del próximo *sprint*. Es una reunión de tres horas para el caso de los *sprint* de un mes, para *sprint* más pequeños es una reunión más corta. Igual que en las reuniones previas el Scrum master asegura

que este evento se realice y que los asistentes comprendan su propósito (Schwaber & Sutherland, 2016).

Los expertos en Scrum Schwaber & Sutherland (2016) indican el propósito de esta reunión:

> Inspeccionar como fue el último *sprint* con respecto a las personas, las relaciones, el proceso y las herramientas. Identificar y ordenar los ítems principales que salieron bien y mejoras potenciales. Crear un plan para implementar mejoras en la forma en que el equipo de Scrum hace su trabajo (p.12).

Una vez completada la reunión, se vuelve a repetir todo el ciclo nuevamente (Rubin, 2013).

2.2.5.2.1.1.3 Artefactos en Scrum

Schwaber & Sutherland (2016) nos dicen que:

> Los Artefactos en Scrum representan trabajo o valor añadido que aportan transparencia y oportunidades para la revisión y adaptación. Los Artefactos están diseñados específicamente para facilitar la transparencia de la información clave y unificar los criterios de compresión de dicho artefacto (p.13).

2.2.5.2.1.1.3.1 *Product backlog*

Es una lista ordenada que contiene lo que se considera necesario para el desarrollo del producto, ya sean características, funcionalidades, requisitos, mejoras o correcciones y es el dueño del producto responsable de su contenido, disponibilidad y ordenamiento (Schwaber & Sutherland, 2016).

Como se muestra en la Figura 16, la lista debe estar ordenada donde las de mayor prioridad son las primeras, hasta bajar a los ítems con prioridad baja:

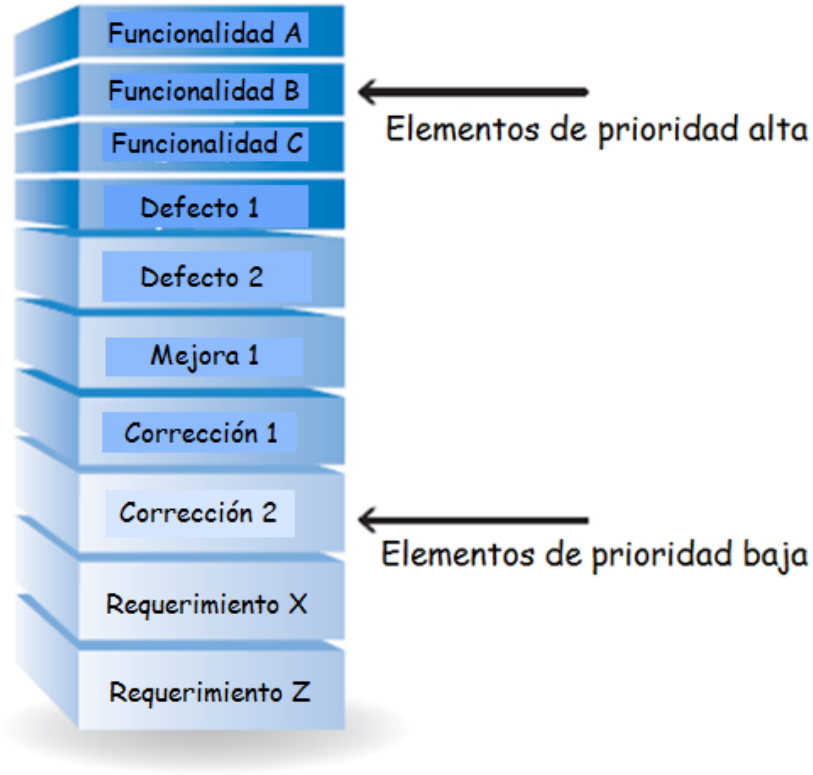


Figura 16: Product Backlog

Fuente: Elaboración propia, 2017.

La lista de *product backlog* me permite visualizar los elementos que se encuentran pendientes de realizar y cuáles ya fueron terminados.

2.2.5.2.1.1.3.2 Sprint backlog

Es el conjunto de elementos del *product backlog* seleccionados para el *sprint*, hace visible todo el trabajo que el equipo de desarrollo identifica como necesario de cumplir para lograr el objetivo del *sprint* (Schwaber & Sutherland, 2016).

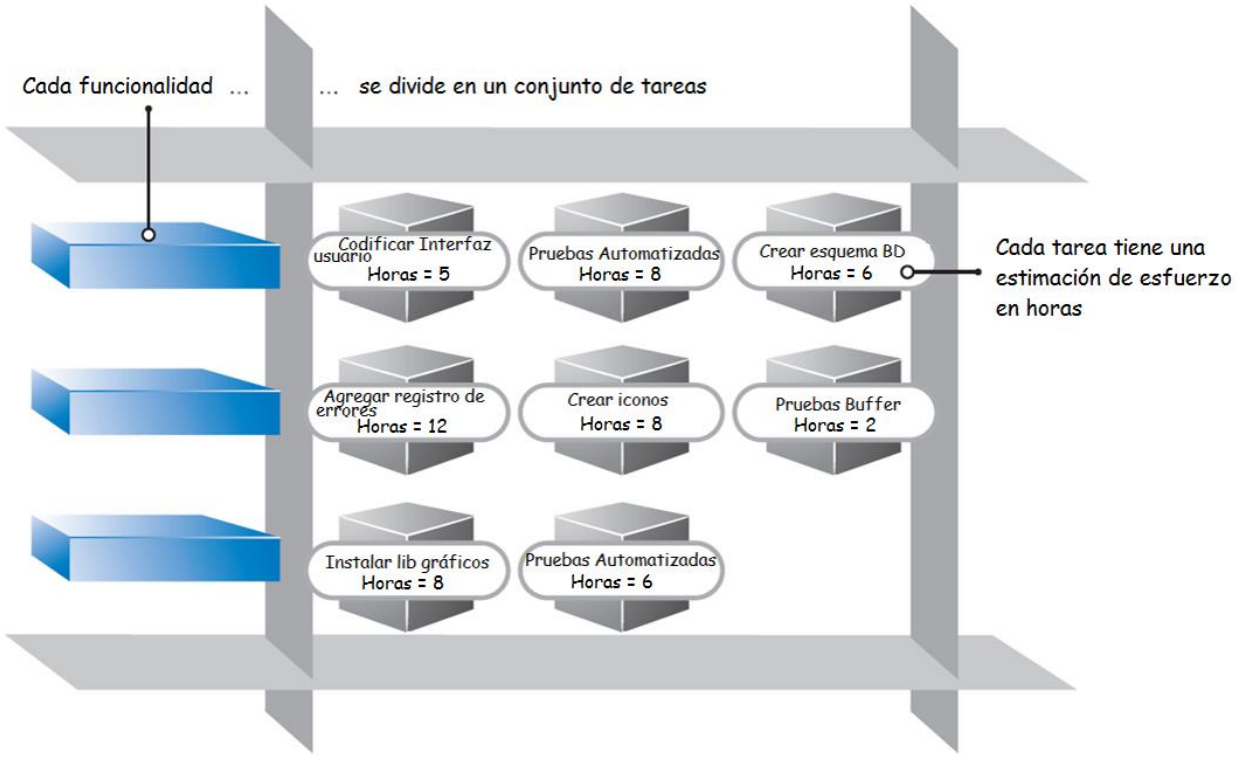


Figura 17: Sprint Backlog

Fuente: Libro Essential Scrum. Adaptado al español

2.2.5.2.1.1.3.3 Incremento del producto

"El incremento es la suma de todos los elementos de la lista de producto realizados durante un *sprint* y el valor de los incrementos de todos los *sprints* anteriores." (Schwaber & Sutherland, 2016, pag.15).

En otras palabras, es la parte del producto que se realiza en un *sprint* y que fue terminada y probada, de manera que se considera como un producto potencialmente entregable en cualquier momento que el dueño del producto lo desee liberar a producción (Menzinsky, López, & Palacio, 2016). Es importante mencionar que el dueño del producto puede decidir en cualquier momento poner el producto en producción, por lo que en buena teoría cada vez que se produce un incremento en un *sprint*, este debe ser utilizable (Schwaber & Sutherland, 2016).

2.2.5.2.1.1.3.4 Flujo de trabajo

Una vez comprendidos todos los roles, actividades y artefactos del marco de trabajo Scrum, podemos describir el flujo de trabajo que comprende Scrum. A continuación en la Figura 18 se muestra el flujo de trabajo:

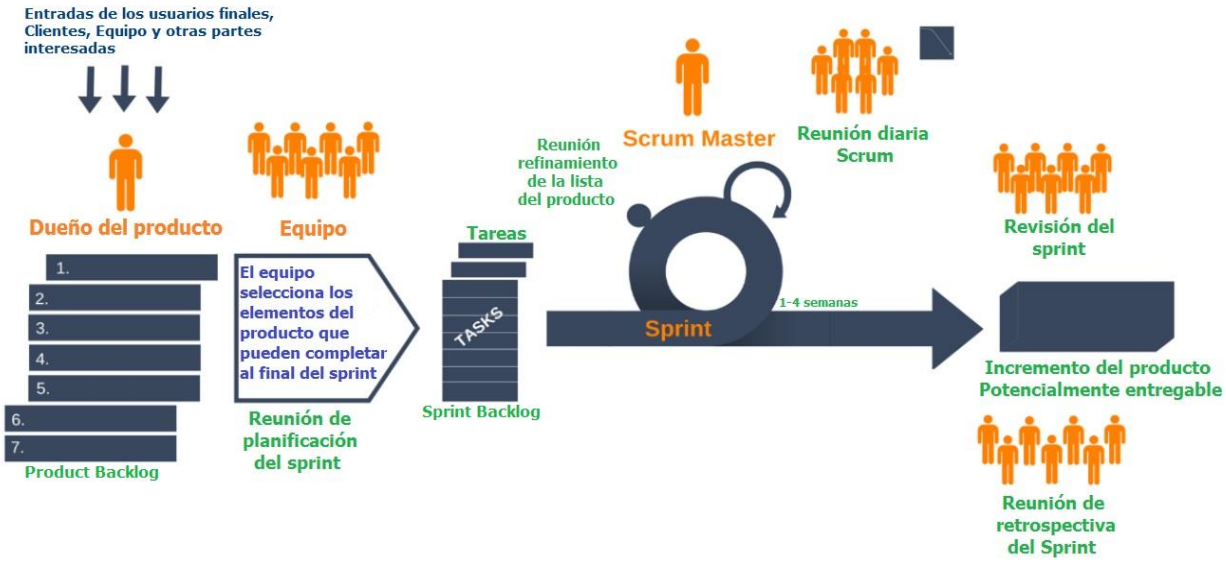


Figura 18: Scrum flujo de trabajo

Fuente: www.uwa.edu.au .Adaptado al español.

Todo inicia con los requerimientos y necesidades de los usuarios, clientes, equipo y otras partes interesadas, posteriormente a partir de la visión del dueño del producto. Este establece lo que se quiere crear, seguidamente se descompone el producto deseado en pequeñas características que se priorizarán en la lista llamada *product backlog*. Se inicia el *sprint* con la planificación del mismo. Es probable que el número de ítems en el *product backlog* sean más de los que el equipo de desarrollo puede completar, es por esto que el equipo de desarrollo debe determinar cuáles elementos del *product backlog* pueden completar en el *sprint* y crean un *sprint backlog* donde estarán todos esos elementos del producto seleccionados previamente, en este se

establecen las tareas necesarias para cada elemento del *product backlog* como por ejemplo tareas de diseño, desarrollo, pruebas, etc.

Seguidamente viene la ejecución del *sprint*, en donde el equipo de desarrollo realiza las tareas de cada elemento del *product backlog*, cada día se realiza una reunión diaria de inspección, sincronización y planificación que permite a los miembros del equipo administrar el flujo de trabajo. Luego de esta etapa si todo salió bien se tendrá un incremento del producto potencialmente entregable, el cual representa una parte de la visión del dueño del producto. Finalmente el equipo de Scrum completa el *sprint* con la realización de dos actividades, las cuales son 1) la revisión del *sprint* donde el equipo de Scrum y las partes interesadas inspeccionan el producto que se ha construido, y 2) la retrospectiva del *sprint*, aquí el equipo de Scrum lo que hace es inspeccionar el proceso de Scrum que está siendo usado para crear el producto, es probable que resultado de estas dos actividades surgen cosas que hay que corregir y agregar al proceso de desarrollo. Por último, el ciclo vuelve a repetirse para el próximo *sprint* (Schwaber & Sutherland, 2016).

2.2.6 Comparación de Modelos de desarrollo

De acuerdo con Linz (2014), los modelos de desarrollo deben tratar con cuatro aspectos: gestión de proyectos, gestión de la calidad, técnicas de desarrollo y gestión del recurso humano. En la Tabla 2 se realiza una comparación de los modelos de desarrollo Scrum y el tradicional enfocándose en estos cuatro aspectos antes mencionados.

Tabla 2: Comparación de modelo Scrum y Tradicional

		Scrum	Tradicional
GESTIÓN DE PROYECTOS	**Planeación del producto**	Visión del producto, *Roadmap*, Lista del producto (*Product Backlog)*	*Roadmap*
	Planeación del proyecto	Lista del producto (*Product Backlog)*	Plan de proyecto con milestones

D E P		Scrum	Tradicional
	Planeación de tareas	Lista de pendientes del sprint (*Sprint backlog*) para cada iteración	Fases específicas
	Iteración	Duración de iteración predefinida (*Sprints*)	De acuerdo con el plan del proyecto
	Duración de iteración	Normalmente de 1-4 semanas	De acuerdo con el plan del proyecto
	Monitoreo de estado	Diario con todo el equipo o en una pizarra	Por el administrador de proyectos, basados en milestones
	Timeboxing	Si	El administrador de proyecto decide
	Entrega	Al final de cada sprint	Con cada liberación o al finalizar el proyecto
	Gestión del cambio	A través de la actualización del backlog	A través del sistema de gestión de cambios existente
	Métricas	Gráfico de trabajo pendiente, en inglés llamado *Burndowm Chart*	Adhesión a plazos, esfuerzo, costos
GESTIÓN DE LA CALIDAD	**Optimización del proceso**	Revisión del proyecto, inspeccionar y adaptar (bottom-up)	Auditorías internas y externas (de acuerdo con el ISO 9001). Programas de optimización de procesos (top-down)
	Verificación/Pruebas	Continuas dentro de cada sprint	Pruebas contra las especificaciones del proyecto durante las fases de prueba
	Validación/Aprobación	Demo al final del sprint (cumplimiento de los criterios de aceptación)	Pruebas de aceptación al finalizar el proyecto
HERRAMIENTAS DE DESARROLLO	**Técnicas**	Común pero no obligatorio: Programación de pares, integración continua, probar primero, programación, diseño incremental, código limpio, refactorización	Depende del modelo en uso. Algunos incluyen codificación predefinida, directrices de herramientas
RECURSOS HUMANOS	**Valores y principios**	Manifiesto Ágil, Valores de Scrum: Compromiso, foco, apertura, respeto, coraje.	Administrador de proyectos, modelo de proceso, optimización continua (ISO 9000)
	Organización	Equipo Scrum, Product owner, Scrum master	Líder del equipo, administrador de proyecto, administrador de la organización
	Capacitación	Por propia iniciativa, deseo de mejorar, aprender con el equipo	Plan de formación de la empresa, desarrollo de recursos humanos
	Asignación de trabajo	Interdisciplinario (multifuncional)	Altamente especializado

Fuente: (Linz, 2014). Adaptado al español

2.2.7 Calidad del Software

La calidad del software juega un papel muy importante dentro del desarrollo de aplicaciones y su objetivo es satisfacer las necesidades de los usuarios. Existen diferentes puntos de vista para definir calidad del software. Pressman (2010) define la calidad de software como: “Proceso

eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes lo producen y a quienes lo utilizan" (p.340). Por otra parte, el glosario IEEE para la ingeniería de software define la calidad del software como: "El grado con el cual un sistema, componente o proceso cumple con los requerimientos y con las necesidades y expectativas del usuario" (p.60).

La calidad del software es responsabilidad de todo el equipo, desde el levantamiento de los requerimientos hasta la implementación del sistema, y no es solo cumplir con los requisitos funcionales sino también los no funcionales donde entra el tema de las expectativas del usuario, es decir temas como la seguridad, rendimiento, entre otros. De acuerdo con el experto Garzás (2012), la calidad del software se subdivide en tres tipos de calidad: la calidad del proceso, la calidad del producto y la calidad del equipo. A continuación se detalla cada uno de estos.

2.2.7.1 La calidad del proceso

Desde el punto de vista de los procesos se dice que la calidad de un producto de software está determinada por la calidad del proceso, entiéndase proceso a las actividades, procedimientos, tareas, entradas, salidas que se realizan para desarrollar y mantener el software. Algunas normas, modelos y metodologías que se enfocan en la calidad del proceso son CMMI, ISO 15504 / ISO 12207, el ciclo de vida utilizado e inclusive las metodologías ágiles entran aquí. Los buenos procesos tienen más probabilidad de conducir a software de buena calidad, pero también es importante aclarar que el tener un modelo de procesos reconocido no siempre asegura que se genere productos de calidad (Garzás, 2012).

2.2.7.2 Calidad del producto

Garzás (2012) refiere a que en IEEE software comentan que existe poca evidencia de que cumplir con un modelo de procesos como CMMI, ISO 12207, o una metodología de desarrollo

sea ágil o no conlleva a producir software de calidad y que las evaluaciones de calidad deberían estar basadas en evidencias del producto y no en evidencias circunstanciales o suposiciones. Es por lo anterior que surgieron los modelos de calidad como el ISO 9126 o la nueva serie ISO 25000 (Garzás, 2012).

2.2.7.3 Calidad del equipo

De acuerdo con Garzás (2012), las personas son uno de los factores más determinantes para lograr el éxito de un proyecto debido a que son los que desarrollan el software. Existen estrategias de motivación, modelos como el *Team Software Process* (TSP), el cual es un método de establecimiento y mejora del trabajo en equipo para procesos de software. El modelo *Personal Software Process* (PSP) comprende un conjunto de prácticas disciplinarias para administrar el tiempo y mejorar la productividad de los ingenieros de software.

2.2.8 Factores de la Calidad

Los factores de calidad proporcionan una base útil para hacer mediciones indirectas y una lista de comprobación excelente para evaluar la calidad del sistema. El estándar ISO 9126 se elaboró con el objetivo de identificar los atributos clave del software, de acuerdo con este estándar se identifican seis atributos (Pressman, 2010).

- **Funcionalidad**. Grado en que el software satisface las necesidades planteadas según las establecen los atributos siguientes: adaptabilidad, exactitud, interoperabilidad, cumplimiento y seguridad.
- **Confiabilidad**. Cantidad de tiempo que el software se encuentra disponible para su uso, según lo indican los siguientes atributos: madurez, tolerancia a fallas y recuperación.
- **Usabilidad**. Grado en el que el software es fácil de usar, según lo indican los siguientes subatributos: entendible, aprendible y operable.

- **Eficiencia**. Grado en el que el software emplea óptimamente los recursos del sistema, según lo indican los subatributos siguientes: comportamiento del tiempo y de los recursos.
- **Facilidad de recibir mantenimiento.** Facilidad con la que pueden efectuarse reparaciones al software, según lo indican los atributos que siguen: analizable, cambiable, estable, susceptible de someterse a pruebas.
- **Portabilidad**. Facilidad con la que el software puede llevarse de un ambiente a otro según lo indican los siguientes atributos: adaptable, instalable, conformidad y sustituible (Pressman, 2010, p.343).

Pressman (2010) establece que para lograr la calidad del software hay que primero lograr una buena administración del proyecto y una buena práctica de la ingeniería del software, lo anterior mediante cuatro actividades principales que ayudan a lograr una alta calidad:

a. **Métodos de la ingeniería de software:** Entender el problema a resolver, y crear un diseño de acuerdo con el problema y con los factores de calidad anteriores.

b. Técnicas de administración de proyectos: Hacer estimaciones para verificar que las fechas pueden cumplirse, planeación del riesgo y que el plan del proyecto incluya técnicas explícitas para la administración de la calidad y el cambio.

c. Control de calidad: Conjunto de actividades que ayudan asegurar que todo producto cumpla sus metas de calidad.

d. **Aseguramiento de la calidad:** Establece la infraestructura de apoyo a los métodos sólidos de la ingeniería de software, la administración racional de proyectos y las acciones de control de calidad. También consiste en un conjunto de funciones de

auditoría y reportes para evaluar la eficacia y completitud de las acciones de control de calidad.

2.2.9 Aseguramiento de la calidad del software (ACS)

Son un conjunto de actividades esenciales que se aplican en todo el proceso del desarrollo de software, el ACS de acuerdo con Pressman (2010) incluye:

1) Un proceso de aseguramiento de la calidad del software.
2) Tareas específicas de aseguramiento y control de la calidad (incluidas revisiones técnicas y una estrategia de pruebas relacionadas entre sí).
3) Prácticas eficaces de ingeniería de software (métodos y herramientas).
4) Control de todos los productos del trabajo de software y de los cambios que sufren
5) Un procedimiento para garantizar el cumplimiento de los estándares del desarrollo de software (cuando sea aplicable).
6) Y mecanismos de medición y reporte (p.369).

Algunas de las actividades o elementos del ACS que se centran en la administración de la calidad del software de acuerdo con el experto Pressman (2010) son:

- **Estándares**

  Existen organizaciones que establecen estándares para la ingeniería de software como por ejemplo el IEEE y el ISO. El aseguramiento de la calidad del software asegura que estos estándares se sigan en caso de haberse adoptado.

- **Revisiones y auditorías**

  Las revisiones técnicas son una actividad del control de calidad cuyo objetivo es detectar errores, y las auditorías garantizan que se sigan los lineamientos de calidad en el trabajo de la ingeniería de software.

- **Pruebas**

  Las pruebas de software son una función del control de calidad que tienen como objetivo principal detectar errores. El ACS garantiza que las pruebas se planeen en forma apropiada y se realicen con eficiencia.

- **Colección y análisis de los errores**

  El ACS reúne y analiza errores y datos acerca de los defectos para entender mejor cómo se comenten los errores (p.370).

2.2.10 Control de la Calidad

De acuerdo con la Norma ISO 9000 (2005), "el control de la calidad es la parte de la gestión de la calidad orientada al cumplimiento de los requisitos de la calidad" (p.10). Lo anterior se logra mediante el empleo de técnicas y actividades de carácter operativo como revisiones, inspecciones y pruebas, cuyos objetivos son: (1) mantener bajo control un proceso y (2) eliminar las causas de los defectos en las diferentes fases del ciclo de vida.

Un aspecto importante dentro de lo que es el control de calidad son las pruebas de software, que permiten validar y verificar el software mediante técnicas y herramientas de pruebas de software.

2.2.10.1 Técnicas de Revisión

Las revisiones permiten descubrir errores y defectos, entre más rápido se pueda encontrar un error es menos caro corregirlo. El resultado de una revisión es una lista de errores o defectos descubiertos.

2.2.10.1.1 Revisiones Técnicas

El objetivo principal de esta técnica es “encontrar errores durante el proceso a fin de que no se conviertan en defectos después de liberar el software.” (Pressman, 2010, p.355). Pressman (2010) establece que existen dos categorías de revisiones técnicas: las revisiones informales y las revisiones técnicas más formales.

2.2.10.1.1.1 Revisiones Informales

Es una simple verificación donde no existe una planeación por adelantado ni se da un seguimiento a los errores encontrados. La eficacia de estas revisiones es menor con respecto a las revisiones formales, pero de igual manera es una revisión que descubre errores y que evita que se propaguen en el desarrollo de software. Las verificaciones de escritorio y la programación por parejas forman parte de esta revisión (Pressman, 2010).

2.2.10.1.1.2 Revisiones Técnicas Formales

Es una actividad del control de la calidad del software. Los objetivos de estas revisiones de acuerdo con Pressman (2010) son:

> 1) descubrir los errores en funcionamiento, lógica o implementación de cualquier representación del software;
>
> 2) verificar que el software que se revisa cumple sus requerimientos;
>
> 3) garantizar que el software está representado de acuerdo con estándares predefinidos;
>
> 4) obtener software desarrollado de manera uniforme y
>
> 5) hacer proyectos más manejables (p.362).

En esta categoría se incluye *walkthroughs* e inspecciones.

2.2.10.2 Pruebas de Software

Las pruebas de software son ejecutadas para verificar que los requerimientos funcionales y no funcionales son cumplidos, también se realizan con la intención de encontrar defectos para corregirlos posteriormente. Lo anterior contribuye a mejorar la calidad del software.

Las pruebas de software tienen dos objetivos:

1) Demostrar al desarrollar y al usuario que el software satisface sus requerimientos. Debería haber al menos una prueba para cada requerimiento, esto en el caso de software a medida y para productos de software genéricos debería haber pruebas para todas las funcionalidades o características del sistema que se incorporan en la entrega del producto.

2) Descubrir defectos en el software, como por ejemplo que el comportamiento del sistema es incorrecto o no deseable o no cumple con su especificación. Algunos ejemplos de estos defectos: caídas del sistema, interacciones no permitidas con otros sistemas, cálculos incorrectos y corrupción de datos (Sommerville, 2011).

2.2.11 Actividades de un proceso de pruebas de software

De acuerdo con el estándar ISTQB (2014), un proceso de pruebas de software comprende al menos cinco actividades que se indican en la Figura 19.

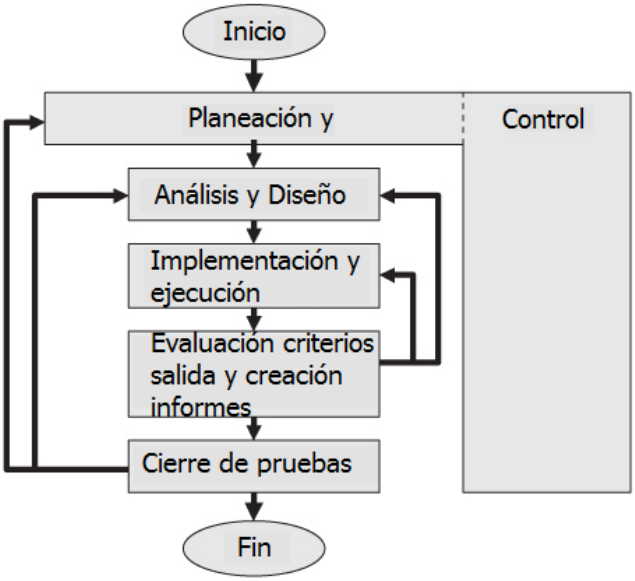


Figura 19: Proceso fundamental de pruebas ISTQB

Fuente: (Schaefer, Linz, & Spillner, 2014). Adaptado al español.

A continuación se detalla cada una de las actividades de la Figura 19.

2.2.11.1 Planeación y control

Un proceso de pruebas de software no debe llevarse a cabo si no se tiene un plan, la planeación del proceso de pruebas se da al principio del desarrollo del proyecto y posterior a la creación el plan se actualiza y se ajusta (Spillner, Linz & Schaefer, 2014).

De acuerdo con los expertos, en la planeación se realizan las siguientes actividades:

a) Planeación de recurso, tiempo.
b) Determinar la estrategia de pruebas, una de las principales tareas de la planeación.
c) Definir la intensidad de las pruebas para subsistemas y diferentes aspectos
d) Priorización de las pruebas
e) Herramientas a utilizar (Spillner, Linz & Schaefer, 2014)

Una vez realizado todo lo anterior citado, se tendrá como resultado un entregable denominado plan de pruebas.

2.2.11.2 Análisis y diseño de pruebas

Se analiza toda la documentación existente con respecto al sistema, para iniciar el diseño de los casos de prueba. Para iniciar este diseño se pueden utilizar: casos de uso, historias de usuario, arquitectura del sistema, diseños, manuales de usuario, manuales técnicos. Se recomienda considerar casos de prueba negativos y positivos, donde los casos de prueba negativos permiten validar cómo se comporta el sistema en circunstancias atípicas. Finalmente, en esta etapa, se definen los datos de prueba necesarios para la ejecución de los casos de prueba diseñados (Sánchez, 2013).

2.2.11.3 Implementación y ejecución de pruebas

Inicia con la creación de los datos de prueba necesarios para ejecutar los casos de prueba previamente diseñados, ya sea de forma manual o automatizada. En cualquiera de los dos casos, si se presenta un fallo en el sistema, es importante que sea documentado y registrado en una herramienta que permita gestionar los defectos como por ejemplo Bug Tracker. Una vez que el defecto sea corregido por el desarrollador, el *tester* debe verificar que el defecto fue solucionado volviendo a ejecutar el caso de prueba y hacer pruebas de regresión que permitan verificar que el cambio realizado no haya afectado a otras funcionalidades (Sánchez, 2013).

2.2.11.4 Evaluación de criterios de salida y generación de informes

Determinar la finalización de las pruebas o si es necesario ejecutar casos de prueba adicionales (Spillner, Linz & Schaefer, 2014). Se define una serie de métricas por ejemplo número de defectos críticos y mayores detectados, lo que va permitir comparar los resultados obtenidos del proceso de pruebas contra lo definido en las métricas (Sánchez, 2013).

2.2.11.5 Cierre del proceso

Evaluar el proceso de pruebas de acuerdo con los recursos utilizados y los resultados alcanzados, para identificar posibles mejoras que deban realizarse y aplicarse para futuros entregables o proyectos (Spillner, Linz & Schaefer, 2014).

El orden de estas actividades depende del modelo de desarrollo utilizado, ya sea secuencial o iterativo. La Tabla 3 muestra cómo el proceso de pruebas debería aplicarse a un sistema desarrollado bajo un modelo en V o cascada.

Tabla 3: Proceso de pruebas en modelo V e iterativo

Actividad del proceso fundamental de pruebas	**Actividad pruebas en Modelo V**	**Actividad de pruebas en Modelo Iterativo**
Planeación	Concurrente con la planificación del proyecto	Al inicio de cada iteración
Control	A lo largo del proyecto desde el inicio de las pruebas hasta la finalización	Se realiza por iteración para todo el proyecto
Análisis y diseño	Concurrente con la especificación de los requerimientos, diseño alto nivel (sistema y arquitectura), y diseño bajo nivel (componente)	Por elementos diseñados para la iteración en particular.
Implementación	Implementación inicia durante el diseño del sistema, el resto se realiza durante la codificación y las pruebas de componentes. Debe concluir antes del inicio de las pruebas del sistema.	Implementación generalmente está limitada a lo que se necesita para la iteración en particular, aunque la planificación a largo plazo para todo el entorno debería comenzar al comienzo del proyecto para que cada iteración tenga el entorno necesario disponible.
Ejecución	Comienza cuando se cumplen (o se excluyen) los criterios de entrada, generalmente cuando se completan las pruebas de integración y de componentes. Continúa hasta que se cumplan los criterios de salida.	Generalmente comienza tan pronto como se complete la prueba de componentes y generalmente se combina con las pruebas de integración. Los criterios de entrada formales podrían no usarse.
Evaluación de criterios de salida y generación de informes	A lo largo de las pruebas, cada vez más frecuente a medida que se acerca el final del proyecto.	A lo largo de las pruebas y particularmente al final de cada iteración, así como al final del proyecto.
Actividades de cierre	Una vez que se cumplen los criterios de salida y cuando se finalizan las pruebas del sistema, aunque puede posponerse hasta que se completen todas las prueba (por ejemplo, pruebas de aceptación)	Al final del proyecto, cuando todas las iteraciones se han completado pero se pueden posponer hasta que se completen todas las pruebas (por ejemplo, pruebas de aceptación)

Fuente: (McKey & Bath, 2014). Tabla adaptada al español

2.2.12 Estrategia de prueba de software

Pressman (2010) establece que:

> Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados (p.383).

Una estrategia de prueba de software tiene las siguientes características genéricas:

- Realizar revisiones técnicas efectivas ayuda a eliminar muchos errores antes de comenzar la prueba.
- La prueba comienza en los componentes o bien pruebas de bajo nivel y opera “hacia afuera”, hacia la integración de todo el sistema.
- Diferentes técnicas son adecuadas para distintos enfoques de ingeniería de software y en diferentes momentos en el tiempo (Pressman, 2010).

2.2.13 Niveles de Pruebas de Software

Al igual que las pruebas en entornos tradicionales, las pruebas ágiles también requieren cubrir todos los niveles de pruebas. El Modelo V que se muestra en la Figura 20 muestra las pruebas que idealmente deben realizarse en cada fase del ciclo de desarrollo.

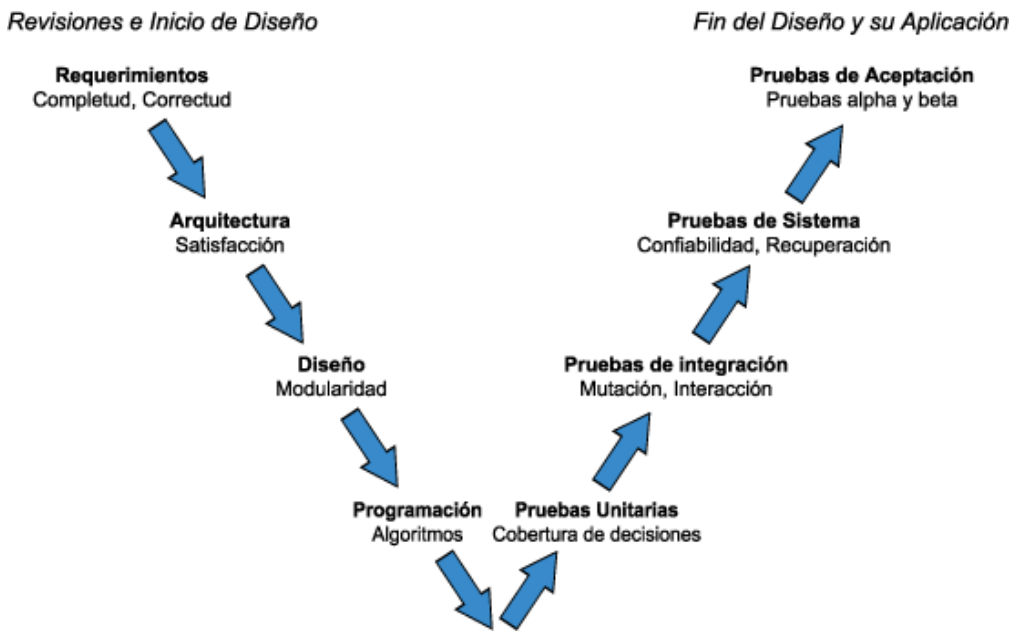


Figura 20: Niveles de pruebas. Modelo V.

Fuente: www.sg.com.mx.

2.2.13.1 Pruebas Unitarias

Son el primer nivel de pruebas y se enfocan en probar unidades pequeñas unidades de código individuales. Las pruebas unitarias siempre se realizan antes de las pruebas de integración. El propósito de esta prueba es aislar cada una de las partes individuales y asegurarse de que todas las partes funcionen correctamente, son realizadas por los desarrolladores y permiten encontrar errores/defectos en una etapa temprana (Daniel, 2014).

2.2.13.2 Pruebas de Integración

Esta prueba se realiza para probar las partes combinadas de la aplicación para comprobar que funcionan correctamente juntas. El objetivo es identificar problemas que puede ocurrir cuando se combinan diferentes unidades. Este tipo de pruebas son realizadas tanto por desarrolladores como *testers* y dentro de las ventajas que se tienen al realizar las pruebas de integración es la prevención y la reducción de errores funcionales y de rendimiento (Daniel, 2014).

2.2.13.3 Pruebas de Sistema

Se realizan para probar un sistema como un todo. Esto significa que aquí es necesario probar todo el funcionamiento de la aplicación de software en una forma detallada. En esta prueba, verificamos que el sistema funciona bien con otros elementos con los que se está interactuando. La prueba de sistema solo se completa cuando cumple con los requisitos de la empresa: requisitos funcionales y no funcionales. Este nivel de prueba tiene como objetivo el verificar las especificaciones del sistema de acuerdo con los requisitos del negocio. Normalmente son realizadas por los *testers* y este tipo de prueba asegura que el sistema funcione acorde a su especificación (Daniel, 2014).

2.2.13.4 Pruebas de Aceptación

Esta prueba se realiza para aceptar el software e iniciar su uso. En esta etapa se han corregido todos los errores máximos para aceptar el software e iniciar su uso. Esta prueba se lleva a cabo en dos partes: pruebas alfa y pruebas beta. Aquí el usuario final prueba el software como un todo y experimenta su experiencia mientras lo usa. El objetivo de las pruebas de aceptación es probar que el sistema o software cumpla con los requisitos de usuario. Son pruebas realizadas por los usuarios o clientes, y una de sus ventajas es que permite un alto nivel de satisfacción de los usuarios (Daniel, 2014).

2.2.14 Técnicas de pruebas de software

Existen dos maneras de probar el software, una de ellas es probando el funcionamiento interno, es decir se prueba el código para verificar que todas las operaciones internas funcionan de acuerdo con lo especificado, a esta técnica se le llama prueba de caja blanca. La otra forma es aplicando pruebas que demuestran la correcta operación de cada función del sistema, a esta última se llama prueba de caja negra (Pressman, 2010).

2.2.14.1 Pruebas de Caja Negra

Los expertos Amo, Martínez y Segovia (2005) establecen que estas pruebas:

> Se llevan a cabo sobre la interfaz del software y pretenden demostrar que el software funciona adecuadamente; es decir, que las entradas se aceptan de forma adecuada y que se produce una salida correcta. Estas pruebas no tienen en cuenta la estructura lógica interna del software. (p.88)

2.2.14.2 Pruebas de Caja Blanca

Esta técnica se basa en la revisión minuciosa de los detalles procedimentales, se revisan bucles, sentencias, funciones, etc. (Amo, Normand, & Pérez, 2005). Se debe tener conocimiento en programación, un ejemplo de esta técnica es la prueba unitaria. La prueba de caja blanca se realiza al inicio del proceso de prueba, mientras que la prueba de caja negra se realiza en las últimas etapas de la prueba (Pressman, 2010).

2.2.15 Tipos de Pruebas de Software

La organización ISTQB (2008) ha establecido cuatro tipos de pruebas:

a) Pruebas funcionales
b) Pruebas no funcionales
c) Pruebas de Estructura de Software (Arquitectura)
d) Pruebas relacionadas con cambios

A continuación, se detalla cada una de estas pruebas.

2.2.15.1 Pruebas Funcionales

Son pruebas que se realizan de acuerdo con los requerimientos especificados. Evalúan la aplicación y validan si la aplicación se está comportando según los requerimientos. Las

funciones o características se prueban alimentándolos con entradas y examinando las salidas. Las pruebas funcionales garantizan que la aplicación satisfaga adecuadamente los requisitos. Este tipo de pruebas no se refiere a como se produce el procesamiento, sino más bien a los resultados del procesamiento.

Durante las pruebas funcionales, la técnica de caja negra es usada en donde el *tester* no conoce la lógica interna del sistema que está siendo probada (Software Testing Fundamentals, s.f.). Hay dos categorías principales de pruebas funcionales:

- Pruebas Funcionales positivas
- Pruebas Funcionales negativas

La primera consiste en introducir entradas válidas para ver cómo responde la aplicación a estas y también para determinar si las salidas son las correctas (Stark, 2014). Por ejemplo:

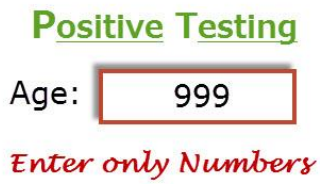


Figura 21: Ejemplo de prueba funcional positiva

Fuente: www.softwaretestingclass.com.

La segunda implica el uso de diferentes entradas inválidas (Stark, 2014). Chequea si una aplicación se comporta como se espera con entradas negativas (Software Testing Class, 2013). Por ejemplo:

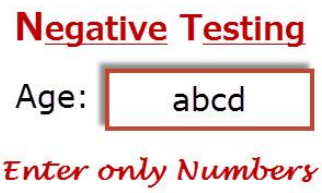


Figura 22: Ejemplo de prueba funcional negativa

Fuente: www.softwaretestingclass.com

2.2.15.2 Pruebas no Funcionales

Son pruebas de los atributos o características no funcionales de un sistema o componente. En esta parte nos interesa ver qué tan rápido o qué tan bien se realizan ciertas acciones en el sistema. Se prueba algo que sea medible, por ejemplo el tiempo de respuesta. Se realiza en todos los niveles de las pruebas.

2.2.15.2.1 Tipos de pruebas no funcionales

El estándar ISO 25010 define ocho características de calidad que deben ser evaluadas para poder obtener un producto de calidad, siete de esas características corresponden a atributos no funcionales que deben ser evaluados mediante algunas de los siguientes tipos de pruebas.

2.2.15.2.1.1 Prueba de rendimiento

El experto Erinle (2013) establece: “La prueba de rendimiento es un tipo de prueba destinada para determinar la capacidad de respuesta, fiabilidad, rendimiento, interoperabilidad y escalabilidad de un sistema bajo una carga de trabajo determinada” (p.8).

2.2.15.2.1.2 Prueba de carga

Se realizan para entender el comportamiento de una aplicación, componente o servicio bajo una carga específica, esta carga va aumentando ya sean usuarios o transacciones para lograr determinar la carga que puede manejar el sistema o componente (ISTQB, 2016).

2.2.15.2.1.3 Prueba de estrés

Es una prueba en que se somete el sistema a una carga por encima de los límites requeridos de funcionamiento, buscando que los recursos como el acceso a la memoria o a los servidores se vuelva inaccesible (ISTQB, 2016).

2.2.15.2.1.4 Pruebas de compatibilidad

Es una prueba de tipo no funcional, que asegura la compatibilidad del sistema o aplicación con otros objetos, como por ejemplo navegadores web, plataformas de hardware, sistemas operativos y entornos de red, entre otros.

2.2.15.2.1.5 Prueba de usabilidad

Es una técnica de caja negra, "que determina la medida en que el producto de software se entiende, es fácil de aprender, fácil de operar y atractivo para los usuarios en condiciones especificadas" (ISTQB, 2016).

2.2.15.2.1.6 Prueba de accesibilidad

"Prueba para determinar la facilidad con que los usuarios con discapacidades pueden usar un componente o sistema" (ISTQB, 2016). Este tipo de prueba es mejor trabajarla reuniendo un grupo de personas que presenten alguna discapacidad y que estos interactúen con la aplicación para poder identificar errores o deficiencias en la pantalla y hacer recomendaciones.

2.2.15.2.1.7 Pruebas de Seguridad

Verifica que los mecanismos de protección que se desarrollan en un sistema, realmente protejan cualquier penetración indebida (Pressman, 2010).

2.2.15.2.1.8 Pruebas de Interfaz

Es un tipo de prueba que verifica la interfaz de la aplicación, en cuanto a controles como los menús, botones, iconos, todo tipo de barras, alineación y posición de los controles, etc.

2.2.15.2.1.9 Pruebas de Documentación

Son pruebas que evalúan la calidad de la documentación, como manuales de usuario y manual de configuración, entre otros (ISTQB, 2016).

2.2.15.2.1.10 Pruebas de Portabilidad

Es un proceso que prueba la facilidad con la que el producto de software se puede mover de un entorno a otro (ISTQB, 2016).

2.2.15.3 Pruebas de Estructura de Software (Arquitectura)

Estas pruebas son normalmente llamadas pruebas de caja blanca debido a que estamos interesados en ver lo que pasa dentro de la caja. Pueden ocurrir en cualquier nivel de pruebas, pero normalmente se realizan más en los niveles de componente y de integración (Graham et al., 2008).

2.2.15.4 Pruebas Relacionadas con Cambios

Estas pruebas se realizan cuando se corrigió un defecto reportado, se verifica que se haya corregido lo que genera un re testeo de la aplicación o componente y se busca que este cambio no haya afectado otros módulos o funcionalidades del sistema. Es aquí donde entra el concepto de pruebas de regresión.

2.2.15.4.1 Prueba de Confirmación

Cuando una prueba falla y se determina que la causa de la falla es un defecto del software entonces se reporta, cuando se corrija este defecto y el *tester* vuelve a probar para confirmar que

el defecto ha sido corregido se dice que se realizan pruebas de confirmación (Graham et al., 2008).

2.2.15.4.2 Prueba de Regresión

Cuando se ejecutan casos de prueba que han sido ejecutados anteriormente se dice que se realizan pruebas de regresión, con el objetivo de verificar que las modificaciones en el software cumplan los requerimientos y además que no ocasionen efectos adversos en el software. (Graham et al., 2008).

Las pruebas de software se pueden realizar de dos formas, de forma manual o de forma automática, dependiendo de los requerimientos del proyecto, la experiencia y el tiempo, entre otros.

2.2.15.5 Pruebas Manuales

Se ejecutan los casos de prueba manualmente utilizando el sistema como lo haría un usuario, no se utiliza ninguna herramienta de apoyo para ejecutar las pruebas.

2.2.15.6 Pruebas Automatizadas

Se ejecutan los casos de prueba sin intervención humana, es decir se utiliza alguna herramienta para ejecutarlas. Estas pruebas ayudan a reducir tiempo que se tarda la ejecución, y permite a los analistas de pruebas dedicar su tiempo a planear y ejecutar pruebas más creativas.

2.2.16 Modelos, estándares y normativas existentes relacionadas con calidad del software

2.2.16.1 ISO

Es una organización no gubernamental compuesta por expertos de todo el mundo que se encargan de crear documentos que especifican requerimientos, guías o características que pueden ser empleadas en organizaciones para garantizar que el producto o servicio sean seguros, de

confianza y de calidad (ISO, 2017). Algunos estándares de la norma ISO que hacen referencia a la calidad del software:

2.2.16.1.1 ISO/IEC 9126 – ISO/IEC 25010

La norma ISO 9126 fue una importante referencia para evaluar la calidad del software, ya que reunía una serie de características y subcaracterísticas relacionadas con la calidad del software. La norma ISO 9126 fue actualizada y actualmente forma parte de la familia de normas ISO/IEC 25000, el ISO/IEC 25010 fue el primer estándar llamado Requisitos y Evaluación de Calidad de Productos de Software (*Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models*). La ISO 25010 define un modelo de calidad del producto el cual se encuentra compuesto por ocho características, cada una de estas tiene subcaracterísticas que se muestran en la .

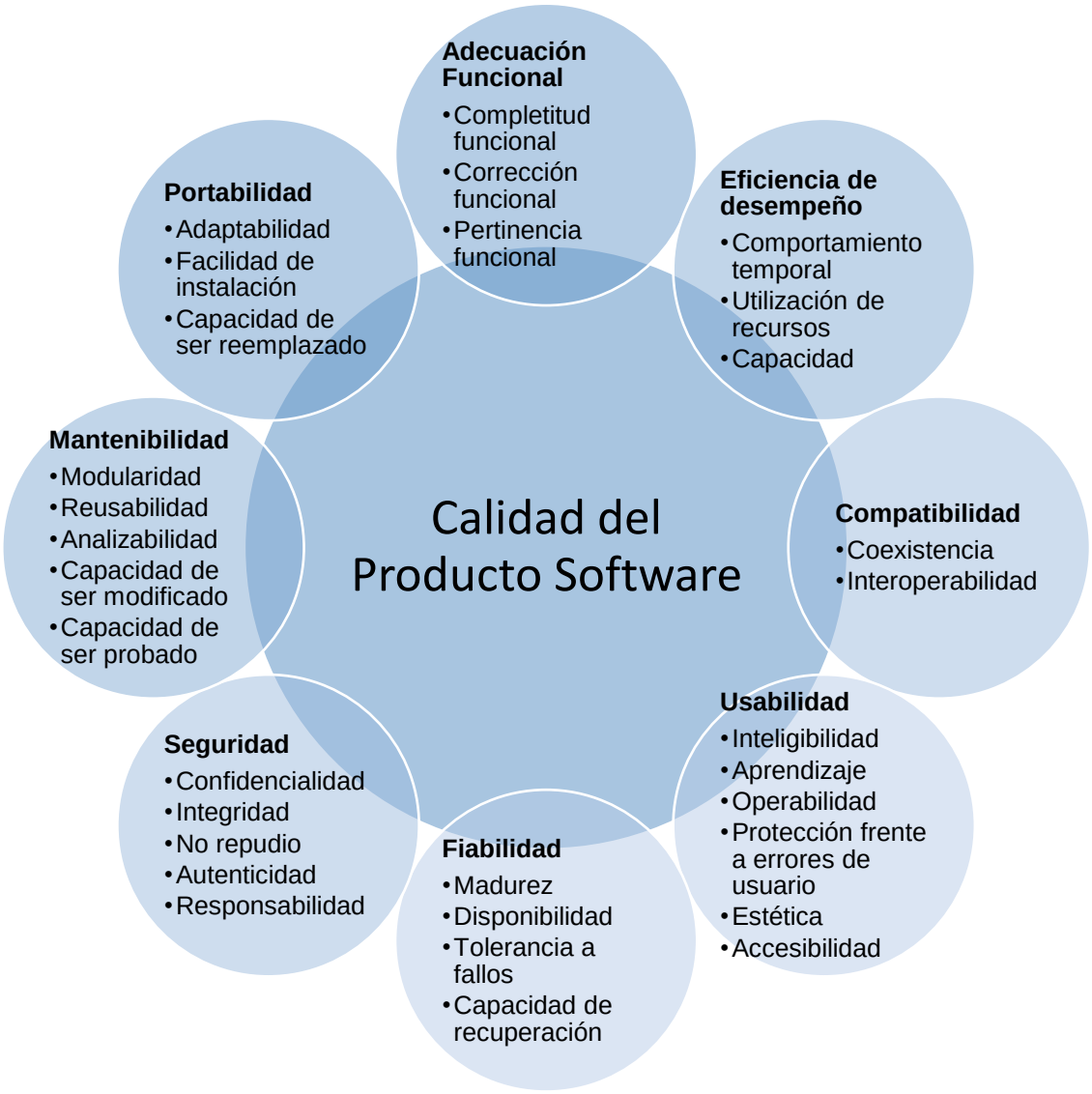


Figura 23: Modelo de calidad del producto ISO/IEC 25010

Fuente: www.iso25000.com. Diseño elaboración propia.

2.2.16.1.2 ISO/IEC 14598 – ISO/IEC 25040

La norma ISO 14598 ha sido actualizada y ahora forma parte de la familia de normas ISO/IEC 25000, específicamente de la ISO/IEC 25040. Esta norma "contiene requerimientos y recomendaciones para la evaluación de un producto de calidad de software y establece los requisitos para la aplicación de este proceso." (ISO, 2011).

El proceso de evaluación comprende cinco actividades que se describen en la Figura 24.

Figura 24: Actividades de la norma ISO/IEC 25040

Fuente: www.iso25000.com.

2.2.16.2 ISTQB

Es una organización de certificación de la calidad del software fundada en el 2002 en Edimburgo, se encarga de soportar y definir un esquema de certificación internacional (Testeando Software, 2014). Surgió debido a la gran importancia que han tomado las pruebas dentro del proceso de aseguramiento de la calidad y dentro del proceso de desarrollo. ISTQB proporciona diferentes niveles de certificación, donde en el 2014 se incorporó al nivel básico la certificación ISTQB *Foundation Level Agile Tester* a razón del creciente uso de metodologías de desarrollo ágiles. Más allá de la certificación, ISTQB nos proporciona estandarización en cuanto

a terminologías, procesos y manuales que es conveniente seguir para tener una metodología de pruebas robusta (ISTQB, 2016).

2.2.16.3 TMMi

Para mejorar la calidad del software se han creado modelos para mejorar los procesos de desarrollo, tal es el caso del modelo CMM y su sucesor CMMi, sin embargo, ambos modelos prestan poca atención al tema de las pruebas de software. Debido esto, la comunidad de pruebas ha creado sus propios modelos de mejora como lo es el modelo TMMi, el cual es un modelo que surge como complemento del modelo CMMi pero enfocado en la mejora de procesos de pruebas de software. Contempla 5 niveles de madurez y dentro de su estructura se definen 16 áreas clave de proceso (TMMi Foundation, 2015).

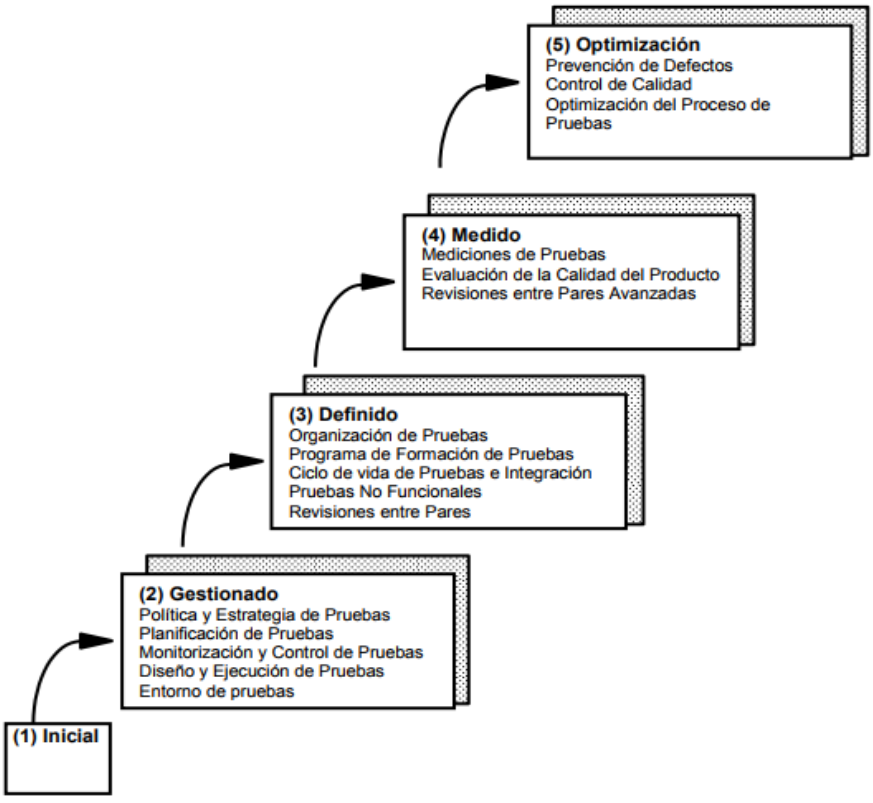


Figura 25: Niveles de madurez y áreas de proceso de TMMi

Fuente: www.tmmifoundation.org.

2.2.17 Pruebas ágiles

Las pruebas ágiles son una práctica que sigue los principios de desarrollo de software ágil y que significan probar dentro de un proceso ágil como Scrum.

2.2.17.1 Principios de Pruebas Ágiles

Las pruebas ágiles siguen los principios del Manifiesto Ágil. De acuerdo con la experta Hendrickson (2008), son nueve principios:

- Las pruebas hacen avanzar el proyecto: Las pruebas ágiles proporcionan retroalimentación continua, permitiendo corregir el rumbo continuamente durante el desarrollo del software.
- Las pruebas no son una fase: Las pruebas se realizan continuamente junto con el desarrollo de software y demás actividades.
- Todo el equipo realiza pruebas: Los analistas de negocio y desarrolladores de software también ejecutan pruebas, no solo los *testers*, como ocurre en metodologías tradicionales.
- Reducir el tiempo para recibir retroalimentación: Con la realización de pruebas continuas durante el *sprint* se reduce el tiempo de retroalimentación y el costo de correcciones también es menor.
- Las pruebas representan expectativas: Antes de gastar tiempo probando algún tipo de riesgo, es mejor tener claras las expectativas con las partes interesadas del proyecto.
- Mantener el código limpio: Los defectos en el código se corrigen en la misma iteración, por lo que se mantiene el código limpio.
- Reducir la documentación de pruebas: Los probadores ágiles usan listas de chequeo reusables en lugar de documentación extensa, se enfocan en la esencia de la prueba en lugar de detalles, capturan ideas de pruebas con las pruebas exploratorias.

- Las pruebas son parte de la definición "*Done*": *Done* significa que ha sido probado e implementado.
- De *Test-Last* a *Test-Driven*: Ya las pruebas no serán al final del proyecto, sino que se realizarán conforme se vaya desarrollando.

2.2.17.2 Métodos de Pruebas Ágiles

Existen una serie de prácticas que se aplican tanto a desarrollos ágiles como tradicionales que permiten producir productos de calidad, asegurando que los tipos de pruebas correctos se ejecuten en el tiempo oportuno y como parte del nivel de pruebas correcto (Black et al., 2014).

2.2.17.2.1 Desarrollo dirigido por pruebas

"Una de las técnicas de pruebas de software que más está alineada con Scrum y con el desarrollo ágil es la de desarrollo conducido por pruebas conocida como *Test Driven Development* (TDD)" (PMOinformatica.com, 2012).

Esta técnica entrelaza el desarrollo de pruebas y el de código, es decir el código se va desarrollando incrementalmente y a la vez se va probando, no se avanza al siguiente incremento hasta que este pase las pruebas respectivas (Sommerville, 2011).

El proceso fundamental del TDD se ilustra en la Figura 26, donde los pasos son:

1) Primero se identifica el incremento de funcionalidad, el cual debe ser pequeño y aplicable en pocas líneas de código.
2) Se escribe la prueba para la funcionalidad identificada.
3) Se corre la prueba, la cual fallará la primera vez debido a que la funcionalidad no está implementada.

4) Se implementa la funcionalidad y se realiza nuevamente la prueba. En caso de fallar se restructura el código para perfeccionarlo.
5) Si todas las pruebas dieron éxito, entonces se avanza a la implementación de la siguiente funcionalidad (Sommerville, 2011).

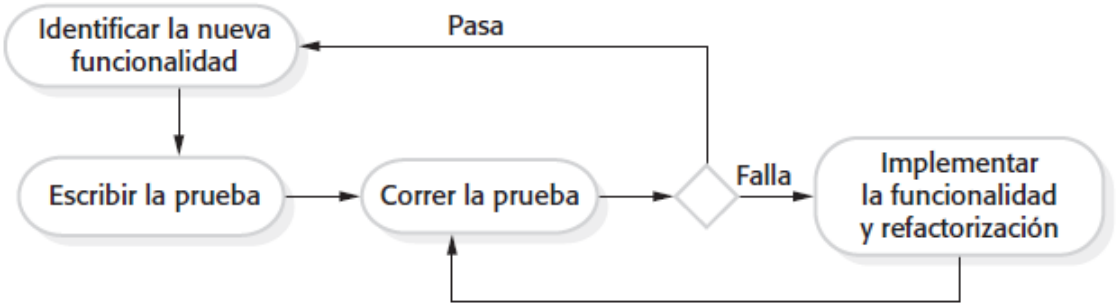


Figura 26: Desarrollo dirigido por pruebas

Fuente: (Sommerville, 2011)

Algunos beneficios de esta técnica son: permite asegurar de forma temprana que lo que se está programando es lo correcto, reduce los costos de las pruebas de regresión donde se verifica que los cambios no hayan introducido nuevos errores en el sistema. Además se agiliza la depuración ya que se conoce de antemano donde se encuentra el error o defecto y por último facilita la comprensión del código debido a que la documentación de las pruebas describen lo que debe hacer el código (Sommerville, 2011).

2.2.17.3 Cuadrantes de Pruebas Ágiles

Los cuadrantes de pruebas ágiles aseguran que el equipo pueda cubrir todos los diferentes tipos de pruebas que se necesitan hacer, por supuesto que para esto se requiere de herramientas, más adelante se explicarán algunas. Los cuadrantes de pruebas alinean los niveles de pruebas con los tipos de pruebas apropiados en la metodología ágil. Los cuadrantes de pruebas ágiles ayudan

a asegurarse que todos los tipos de pruebas importantes y los niveles de pruebas se incluyan durante el desarrollo (Crispin & Gregory, 2009).

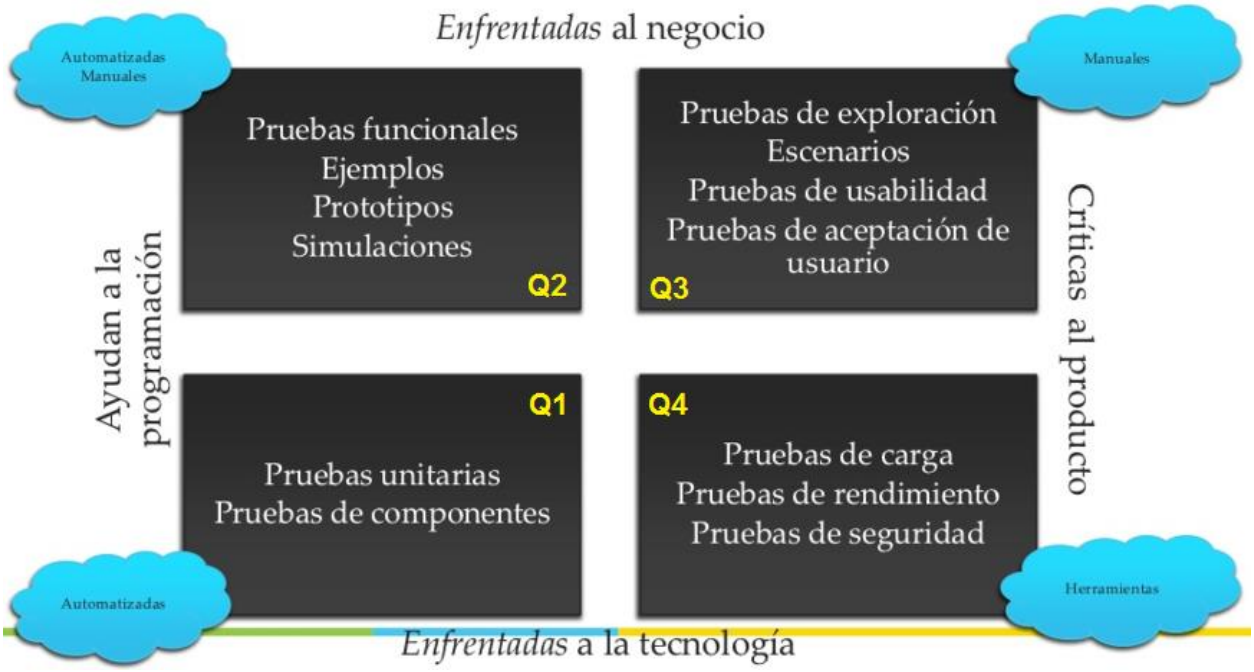


Figura 27: Cuadrantes Pruebas Ágiles

Fuente: www.slideshare.net/globetesting.

Cuadrante 1: Se realizan pruebas unitarias para verificar la funcionalidad de un pequeño subconjunto del sistema como un objeto o método y además se realizan pruebas de componentes para verificar el comportamiento de una parte más larga del sistema, por ejemplo como un grupo de clases de un servicio. Ambos tipos de pruebas son automatizadas y son pruebas que realizan los desarrolladores y que al final se enfrentan a la tecnología. En este cuadrante se puede implementar el método de desarrollo dirigido por pruebas (Crispin & Gregory, 2009).

Cuadrante 2: Se realizan pruebas que apoyan el trabajo del equipo de desarrollo, donde se define la calidad externa y las funcionalidades que el cliente desea. Pueden ser pruebas automatizadas o manuales, también se implementa el método de desarrollo dirigido por pruebas pero en un nivel más alto (Crispin & Gregory, 2009).

Cuadrante 3: Las pruebas de aceptación normalmente las realizan los usuarios y los clientes, las pruebas exploratorias son realizadas por el equipo de QA, quienes simultáneamente diseñan y realizan las pruebas, usando su creatividad e intuición. Se dice que los errores más serios son encontrados mediante las pruebas exploratorias. Las pruebas de usabilidad son realizadas tanto por el equipo de QA como por un grupo seleccionado de personas del negocio donde son observadas por el equipo de QA para analizar el uso del sistema, en este cuadrante las pruebas se realizan de forma manual (Crispin & Gregory, 2009).

Cuadrante 4: Se realizan los tipos de pruebas que verifican los atributos no funcionales de un sistema usualmente por medio de herramientas (Crispin & Gregory, 2009).

2.2.17.4 Pruebas Exploratorias

Este tipo de pruebas son muy importantes en los desarrollos ágiles ya que ayudan a los probadores de software a mantenerse al día con el ritmo de desarrollo rápido de los proyectos de software ágil (Ghahrai, 2015).

Las pruebas exploratorias son aquellas que son realizadas con total libertad sin casos de prueba, más que una prueba es una actividad de pensamiento que depende mucho de las habilidades del *tester*. Dentro de las ventajas se tiene: se descubren errores que normalmente son ignorados por otras técnicas de prueba, amplían la imaginación de los *testers* mediante la ejecución de más y más casos de prueba, se prueba hasta la parte más pequeña de la aplicación, entre otras (Guru99, n.d.).

2.2.18 Herramientas

Una herramienta de pruebas es utilizada como apoyo a las actividades del proceso de pruebas, permite que dicho proceso sea más eficiente y eficaz (ISTQB, 2016). Existen diferentes tipos de

herramientas, algunas de ellas se especializan en soportar una o varias de las actividades del proceso de pruebas (ver Figura 19) y otras dan soporte a todas las actividades del mencionado proceso de pruebas. A continuación se describen dos herramientas que serán fundamentales para el desarrollo del proyecto.

2.2.18.1 Microsoft Test Manager

Es una herramienta de Microsoft que facilita la creación y ejecución de casos de prueba, realizar pruebas exploratorias, grabar y reproducir las pruebas manuales, dar seguimiento de la calidad del software, o especificar las plataformas de prueba, entre otros (Microsoft, 2017).

2.2.18.2 Visual Studio 2017

La licencia Enterprise de Visual Studio 2017 o 2015 permite crear pruebas de rendimiento y pruebas de carga tanto aplicaciones web como servicios. Las pruebas se pueden realizar en la nube o a nivel local (Microsoft, 2017).

2.2.18.3 Jmeter

Es una herramienta 100% Java gratuita de código abierto que permite realizar pruebas de carga y medir el rendimiento de diferentes aplicaciones, servidores y diferentes protocolos como HTTP, HTTPS, servicios web de tipo SOAP / REST, entre otros (The Apache Foundation, 2017).

CAPÍTULO III

3. MARCO METODOLÓGICO

3.1 TIPO Y ENFOQUE DE LA INVESTIGACIÓN

Finalidad

El proyecto a desarrollar corresponde a una investigación aplicada para el desarrollo de una propuesta para la solución de un problema concreto.

Marco

Se va realizar una investigación micro, la cual está orientada solamente a un área dentro de la Dirección de Tecnologías de la Información y Comunicación del Poder Judicial de Costa Rica, llamada Informática de Gestión, donde se va desarrollar una metodología de control calidad basada en la metodología de desarrollo Scrum.

Naturaleza

La naturaleza de este proyecto es cualitativa, ya que mediante las técnicas de observación, entrevista no estructurada y encuestas sobre los fenómenos que son estudiados se podrán obtener datos importantes para la investigación que demostrarán los defectos encontrados que serán de insumo para la propuesta del proyecto.

Carácter

El propósito del estudio correlacional según Hernández, Fernández y Baptista (2010) es: “conocer la relación o grado de asociación que exista entre dos o más conceptos, categorías o variables en un contexto en particular” (p.81). Por lo anterior, se establece que el presente estudio es correlacional ya que se relacionan variables para encontrar los factores que están generando el problema del proceso de pruebas de software en Scrum.

3.2 FUENTES Y SUJETOS DE INFORMACIÓN

Fuentes Primarias

De acuerdo con Hernández et al. (2010), las fuentes primarias "proporcionan datos de primera mano, pues se trata de documentos que incluyen los resultados de los estudios correspondientes." (p.53).

Las fuentes primarias que se utilizaron para el desarrollo del proyecto son las siguientes:

- Personal del área de calidad y desarrolladores de la sección informática de gestión.
- Libros de ingeniería del software relacionados con la gestión de la calidad del software, con Scrum y con pruebas ágiles.
- Documentos, estudios del proyecto SIAGPJ del área de Informática de Gestión.

Fuentes Secundarias

Para efectos de esta investigación se utilizaron las siguientes fuentes secundarias:

- Artículos y Testimonio de Expertos en cuanto a pruebas de software en Scrum
- Proyectos de graduación: Marco Conceptual y Metodológico
- Normas: Norma APA Quinta Edición

Sujetos de Información

Los sujetos de información son personas que colaboran con la investigación proporcionando información clave y necesaria del caso actual de investigación. Algunos sujetos de información consultados para la realización de este proyecto fueron los siguientes:

Tabla 4: Sujetos de información

Puesto Laboral o Descripción general	Profesión u Oficio	Experiencia	Relación con el tema
Coordinador del área de	Coordinador del área de	6 años como	Forma parte del área

Puesto Laboral o Descripción general	Profesión u Oficio	Experiencia	Relación con el tema
calidad	calidad	coordinador de calidad de software	donde se está realizando el estudio
Tester del área de calidad	*Tester* del área de calidad	3 años como *tester* de software	Forma parte del área donde se está realizando el estudio
Desarrolladores de software del área informática de gestión	Análisis y desarrollo software	Varía dependiendo del desarrollador. Pero la experiencia mínima es 1 año desarrollando bajo metodología Scrum	Forman parte del equipo Scrum, e interactúan con el área de calidad.
Product Owner del proyecto SIAGPJ	Coordinar el desarrollo del proyecto SIAGPJ	2 años como coordinador del proyecto SIAGPJ	Forma parte del equipo Scrum, e interactúan con el área de calidad.
Administrador de proyectos	Gestión de proyectos del SIAGPJ	2 años como gestor de proyectos	Forma parte del equipo Scrum, e interactúan con el área de calidad.
Jefe del área	Jefe del Área de Informática de Gestión	3 años como jefe de sección sistemas 15 años como coordinador de desarrollo de software	Jefe del área donde se realiza la propuesta.

Fuente: Elaboración propia, 2017.

3.3 TÉCNICAS Y HERRAMIENTAS

Un instrumento de medición, según establecen Hernández et al. (2010), es “un recurso que utiliza el investigador para registrar información o datos sobre las variables que tiene en mente” (p. 200).

A continuación se detallan las herramientas que se utilizaron para la recopilación de información.

Observación

Hernández et al. (2010) establecen que: “Este método de recolección de datos consiste en el registro sistemático, válido y confiable de comportamientos y situaciones observables, a través de un conjunto de categorías y subcategorías.” (pag. 260).

Se utilizará este método como apoyo a la definición de la situación actual del proceso de pruebas en el proyecto SIAGPJ.

Entrevistas

Se elaborará una entrevista al Jefe del Área de Informática de Gestión (véase Anexo 6), con el fin de obtener información del tema a tratar desde la perspectiva de este. La entrevista se compone de nueve preguntas abiertas que tratan temas de calidad del software, priorización de proyectos y Scrum, entre otros. Adicionalmente se realizarán preguntas al coordinador del área de calidad, a los desarrolladores y administradores de proyecto para tener un mejor entendimiento de la situación actual.

Encuestas

Se realizarán dos encuestas, la primera al área de desarrollo (véase Anexo 4) y la segunda al coordinador de calidad (véase Anexo 5). Lo anterior para obtener un panorama más real de la situación del proceso de control de calidad así como de la metodología Scrum en el proyecto SIAGPJ, con base en cuestionarios con preguntas cerradas, abiertas o mixtas con el objetivo de recolectar información necesaria y específica. Ambas encuestas se encuentran estructuradas en tres secciones: la primera corresponde a preguntas relacionadas con la información de la persona encuestada, la segunda refiere al uso y conocimiento de la metodología Scrum y la tercera corresponde a preguntas relacionadas directamente con el proceso de control de calidad.

Población y Muestra

Poder Judicial II Circuito Judicial de San José, Área Informática de Gestión. Se aplicará la encuesta al personal de los siguientes puestos: desarrolladores, jefes de proyecto y al coordinador de calidad software.

3.4 VARIABLES DE INVESTIGACIÓN

3.4.1 Variables de investigación objetivo general

Objetivo general: Proponer una metodología de control de calidad del software adaptada a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el área de informática de Gestión del departamento de la DTIC del Poder Judicial.

Tabla 5: Variables de investigación objetivo general

Variables asociadas	Descripción	Fuentes de Información	Técnica o Herramienta
• Identificar los defectos y causas del procedimiento de control de calidad actual. • Procesos con brechas o defectos. • Proponer una metodología de control de la calidad del software adaptada al marco de trabajo Scrum. • Mejoras en los procesos de pruebas en la etapa de desarrollo bajo la metodología Scrum.	Definición de casos de prueba, uso de herramientas, tipos de pruebas, duración de las pruebas, reportes de las pruebas. Identificar qué tan apegado se encuentra el equipo a la metodología Scrum. Identificar si los procedimientos de control de calidad se realizan de acuerdo con estándares como el ISTQB. Diseñar la metodología. Defectos encontrados durante la ejecución	Procesos del área de calidad en el proyecto SIAGPJ. Procesos del equipo Scrum. Estándar ISTQB, estándares ISO, modelo calidad TMMi, pruebas ágiles. Metodología Scrum. Metodología propuesta.	Entrevista Encuesta Observación Revisión documental Metodología propuesta.

Variables asociadas	Descripción	Fuentes de Información	Técnica o Herramienta
	del plan piloto y pruebas realizadas.		

Fuente: Elaboración propia, 2017.

3.4.2 Variables de investigación objetivos específicos

Objetivo específico 1: Definir la situación actual del proceso de control de calidad del software que se realiza en los proyectos desarrollados bajo Scrum del Área de Informática de Gestión.

Tabla 6: Variables de investigación objetivo específico 1

Variables asociadas	Descripción	Fuentes de Información	Técnica o Herramienta
Identificar los defectos y causas del procedimiento de control de calidad actual.	Definición de casos de prueba. Uso de herramientas Tipos de pruebas Duración de las pruebas Reportes de las pruebas	Procesos del área de calidad en el proyecto SIAGPJ. Procesos del equipo Scrum.	Entrevista, encuesta, Observación.

Fuente: Elaboración propia, 2017.

Objetivo específico 2: Definir la brecha entre el proceso de control de calidad actual del Área de Informática de Gestión contra lo establecido en las metodologías de calidad de software enfocadas a ambientes ágiles.

Tabla 7: Variables de investigación objetivo específico 2

Variables asociadas	Descripción	Fuentes de Información	Técnica o Herramienta
Procesos con brechas o defectos.	Identificar qué tan apegado se encuentra el equipo a la metodología Scrum.	Estándar ISTQB, estándares ISO, modelo calidad	Revisión documental, entrevista, encuesta.

Variables asociadas	Descripción	Fuentes de Información	Técnica o Herramienta
	Identificar si los procedimientos de control de calidad se realizan de acuerdo con estándares como el ISTQB.	TMMi, pruebas ágiles. Metodología Scrum.	

Fuente: Elaboración propia, 2017.

Objetivo específico 3: Diseñar un marco de trabajo de control de la calidad de software conforme a la metodología Scrum y adaptado al proceso de desarrollo ágil de la institución.

Tabla 8: Variables de investigación objetivo específico 3

Variables asociadas	Descripción	Fuentes de Información	Técnica o Herramienta
Proponer una metodología de control de la calidad del software adaptada al marco de trabajo Scrum.	Diseñar la metodología.	Estándar ISTQB, estándares ISO, modelo calidad TMMi, pruebas ágiles. Metodología Scrum.	Revisión documental.

Fuente: Elaboración propia, 2017.

Objetivo específico 4: Implementar un plan piloto de la metodología ágil de control de calidad en el área de informática de gestión para el proyecto SIAGPJ.

Tabla 9: Variables de investigación objetivo específico 4

Variables asociadas	Descripción	Fuentes de Información	Técnica o Herramienta
Mejoras en los procesos de pruebas en la etapa de desarrollo bajo la metodología Scrum.	Defectos encontrados durante la ejecución del plan piloto Pruebas realizadas.	Metodología propuesta.	Metodología propuesta.

Fuente: Elaboración propia, 2017.

3.5 DISEÑO DE LA INVESTIGACIÓN

En esta sección se muestra el proceso que conlleva el proyecto, las distintas etapas indicando cuáles técnicas y herramientas se aplican en cada etapa, y se exponen los resultados que se espera obtener en cada una de las etapas.

Tabla 10: Diseño de la investigación

Etapa	Descripción	Técnicas y herramientas	Resultados esperados
Etapa I	Se define el estado de la situación actual del proceso de control de calidad.	Se aplican dos encuestas. Una para el equipo de desarrollo y la otra para el coordinador de calidad. Se utiliza el método de la observación para definir procedimientos actuales del área.	Establecer el procedimiento actual del área de calidad.
Etapa II	Realizar un análisis que permita identificar las principales brechas relacionadas con el proceso de control de calidad en una metodología ágil.	Revisión documental. Resultados de la entrevista y la encuesta permitirán identificar las principales brechas.	Lista detallada de los procesos del equipo de calidad y de la metodología scrum que presentan brechas o defectos.
Etapa III	Elaborar la propuesta de metodología.	Revisión documental. Marco teórico del proyecto.	Metodología para el control de la calidad del software adaptada a Scrum.
Etapa IV	Implementar plan piloto de la metodología.	Metodología propuesta.	Reporte de los resultados de las pruebas. Mejora de los procesos de calidad. Automatización de tareas.

Fuente: Elaboración propia, 2017.

CAPÍTULO IV

4. DIAGNÓSTICO DE LA SITUACIÓN ACTUAL

4.1 Análisis de situación actual

Dentro de los objetivos específicos del proyecto se encuentra definir la situación actual del proceso de control de calidad del software que se realiza en los proyectos desarrollados bajo Scrum del área de informática de gestión. Para esto se realizaron dos encuestas, una al equipo de desarrollo y la otra al coordinador del área de calidad. Ambas encuestas se encuentran divididas en tres temas específicos: información sobre la persona encuestada, información sobre la metodología de desarrollo e información general sobre el proceso de control de calidad. Adicionalmente se realizó una entrevista al jefe del área para conocer su perspectiva sobre la metodología Scrum y el equipo de calidad.

4.1.1 Tabulación de los datos e interpretación

4.1.1.1 Encuesta 1. Equipo Desarrollo

Se realizó esta encuesta a 15 personas que forman parte del proyecto SIAGPJ, entre los cuales se encuentran desarrolladores, el dueño del producto, el Scrum Master y el administrador de proyectos. A continuación se detallan los resultados obtenidos de la encuesta.

4.1.1.1.1 Información sobre las personas encuestadas

De las 15 personas encuestadas, 12 (80%) tienen licenciatura mientras que 3 (20%) tienen maestría. Lo anterior permite evidenciar que todo el equipo de desarrollo que se encuentra trabajando en el proyecto SIAGPJ son profesionales en ingeniería del software.

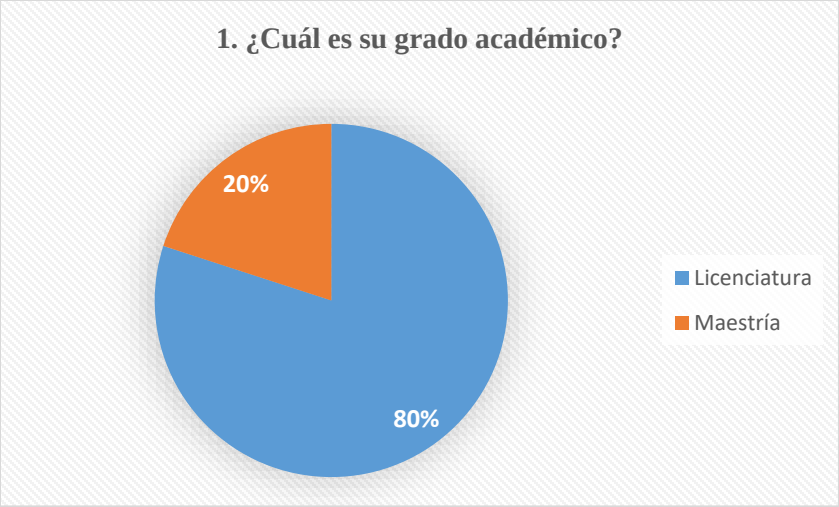


Figura 28: Grado académico del equipo desarrollo

Fuente: Elaboración propia, 2017.

Por otra parte, se solicitó a los encuestadores que seleccionaran un rango de años de experiencia laboral en el ámbito del software, en donde 12 personas respondieron que tienen más de 7 años mientras que el resto tienen entre 5 y 7 años de experiencia. Lo anterior demuestra que el equipo se encuentra compuesto por profesionales con una larga trayectoria y experiencia en el área de software.

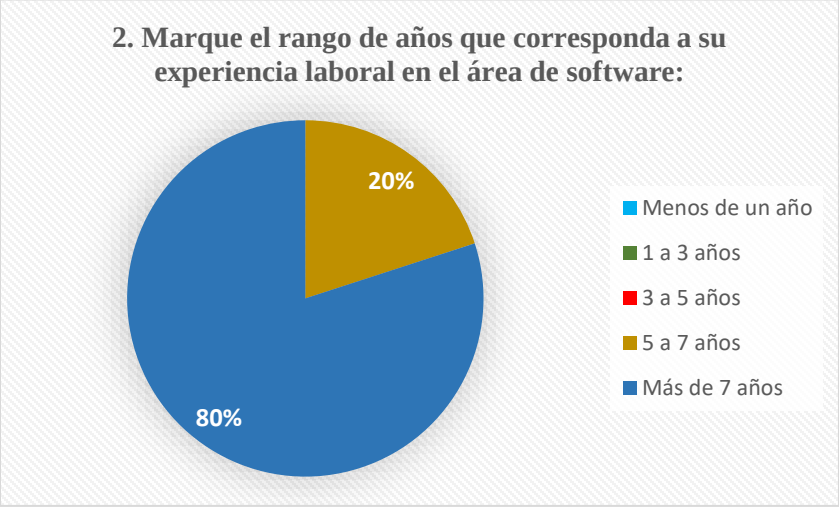


Figura 29: Años de experiencia laboral del equipo desarrollo

Fuente: Elaboración propia, 2017.

4.1.1.1.2 Información general sobre la metodología de desarrollo

Como se aprecia en la Figura 30, 14 (93%) de las personas encuestadas conocen la metodología Scrum y solamente una persona respondió que no la conoce, lo cual puede ser explicado porque se incorporó un desarrollador nuevo al equipo y el mismo no ha recibido la inducción sobre la metodología para la fecha en la que se aplicó la encuesta.

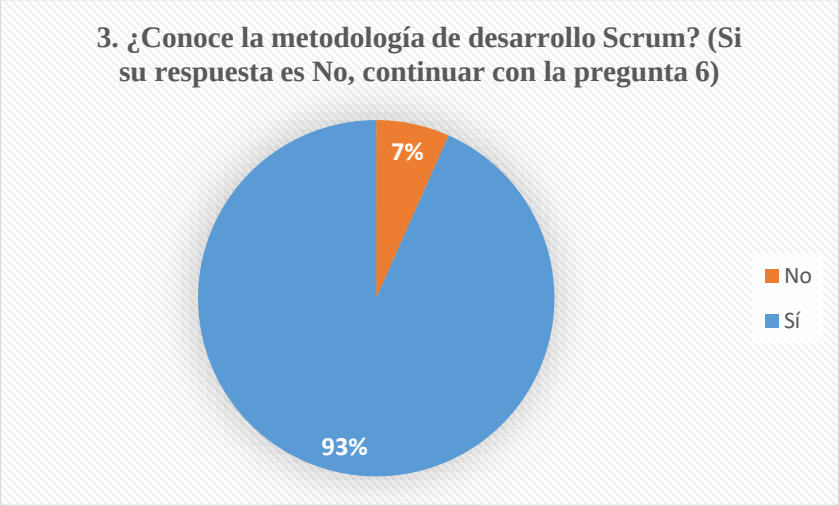


Figura 30: Conocimiento de la metodología Scrum

Fuente: Elaboración propia, 2017.

De las 14 personas que indicaron conocer la metodología Scrum, se quiso identificar si las mismas hacían uso de la metodología en el trabajo diario, para lo cual 13 (93%) respondieron que sí mientras que solo 1 persona indicó que no.

Figura 31: Uso de la metodología Scrum en el trabajo diario

Fuente: Elaboración propia, 2017.

De las personas que hacen uso diario de la metodología Scrum en el trabajo, se logró identificar que la frecuencia con la que la utilizan es de manera constante. A continuación se describen las actividades que se realizan actualmente:

a) Se realiza una planificación del sprint durante todo un día; se reúnen los desarrolladores, el dueño del producto, el scrummaster y en ocasiones el administrador de proyectos. En esta reunión se definen las funcionalidades que se desarrollarán en el próximo sprint así como las tareas necesarias para poder desarrollarlas.
b) Durante la ejecución del sprint, el equipo se reúne a diario para exponer lo que cada uno hizo el día anterior, si se presentaron impedimentos y lo que se realizará durante el día.
c) Al finalizar el sprint, el equipo se reúne para exponer lo realizado en el sprint.

Figura 32: Frecuencia de uso de la metodología Scrum

Fuente: Elaboración propia, 2017.

Por último en esta sección sobre la metodología de desarrollo, se logró identificar que la mayoría 13 (87%) de los encuestados opinan que Scrum sí es el marco de trabajo adecuado que el área necesita para el desarrollo de proyectos. Lo anterior evidencia que el equipo se encuentra satisfecho trabajando sobre este marco de trabajo.

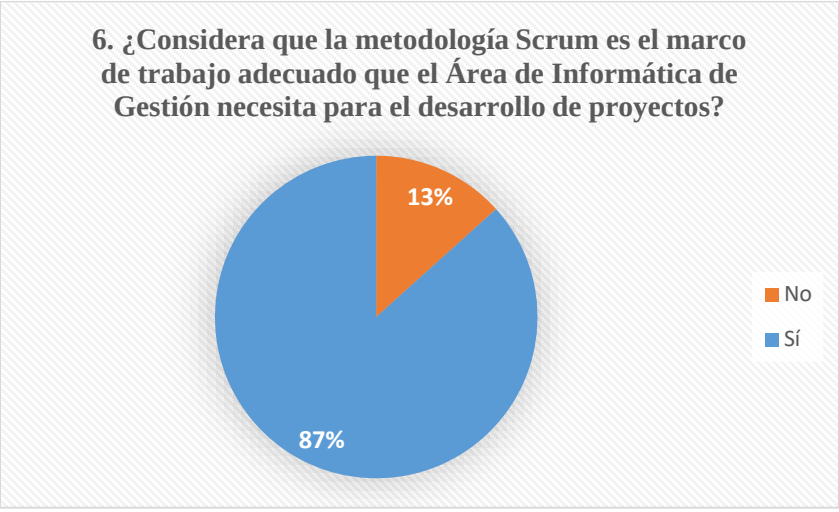


Figura 33: Metodología Scrum marco de trabajo adecuado

Fuente: Elaboración propia, 2017.

4.1.1.1.3 Información general sobre el proceso de control de calidad

En esta sección de la encuesta se logró identificar aspectos importantes relacionados con el proceso de control de calidad, lo anterior desde el punto de vista de los desarrolladores, el dueño del producto, Scrummaster y el administrador de proyectos. En la Figura 34 se puede apreciar que 9 (60%) de los encuestados desconoce si existe una metodología de control de calidad, mientras que 5 (33%) indicaron que no existe y solo una persona indicó que sí existe. Este punto nos permite identificar que la mayoría de la población encuestada desconoce si el equipo de calidad trabaja bajo una guía metodológica propia para el control de la calidad del software.

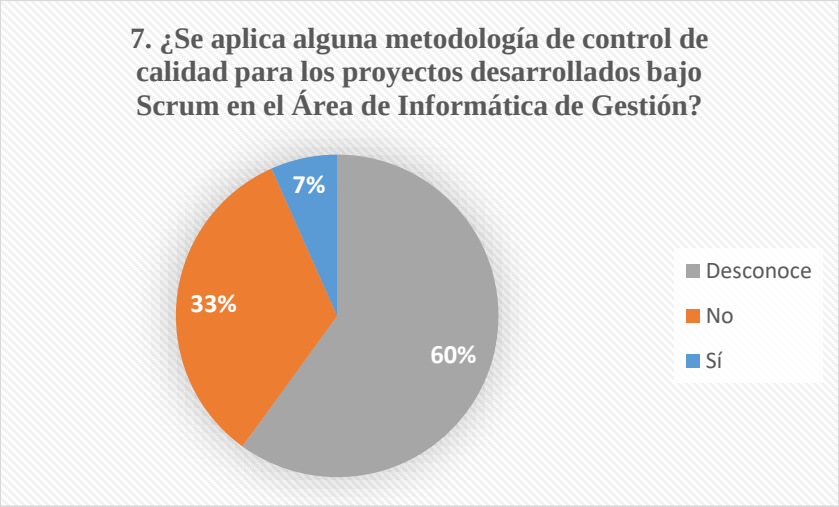


Figura 34: Existencia de una metodología de control de calidad

Fuente: Elaboración propia, 2017.

Por otra parte, se quiso conocer si las tareas de control de calidad se realizan previamente a una planificación de las mismas. En la Figura 35 vemos que 9 (60%) de la población encuestada desconoce si existe un proceso de planificación, 4 (27%) indicaron que sí existe y 2 (13%) indicaron que no. El desconocimiento puede ser explicado porque la mayoría de los desarrolladores no intervienen en la planificación del proyecto y menos en las tareas de control de calidad.

Figura 35: Planificación de las tareas de control de calidad

Fuente: Elaboración propia, 2017.

Se preguntó a los participantes de la encuesta si se aplican pruebas de software en cada sprint que comprenda la liberación de un producto funcional, para lo que 10 (67%) respondieron que no y 5 (33%) indicaron que sí. Con esto se logra identificar que el equipo de desarrollo programa funcionalidades o componentes que no pasan inmediatamente por el proceso de control de calidad en el sprint, sino que se van acumulando las funcionalidades para que en determinado momento se pasen a la etapa de pruebas. Esto es una especie de metodología en cascada donde las pruebas se realizan al final. Con Scrum la idea es probar funcionalidades, componentes, pantallas y código aunque no se tenga un producto funcional para entregar. "Las pruebas deberían ocurrir dentro del sprint desde el inicio hasta el final, iniciando con las pruebas unitarias, y finalizando con las pruebas de aceptación o pruebas de caja negra, todo dentro del sprint" (XBOSoft, 2012, p.6). Por otra parte, en la encuesta *The XBOSoft 2012 Scrum Testing Survey* se logró identificar que la mayoría de las compañías encuestadas manejan el proceso de pruebas en cada sprint de forma que los programadores escriben un pequeño código y el mismo es probado inmediatamente (XBOSoft, 2012).

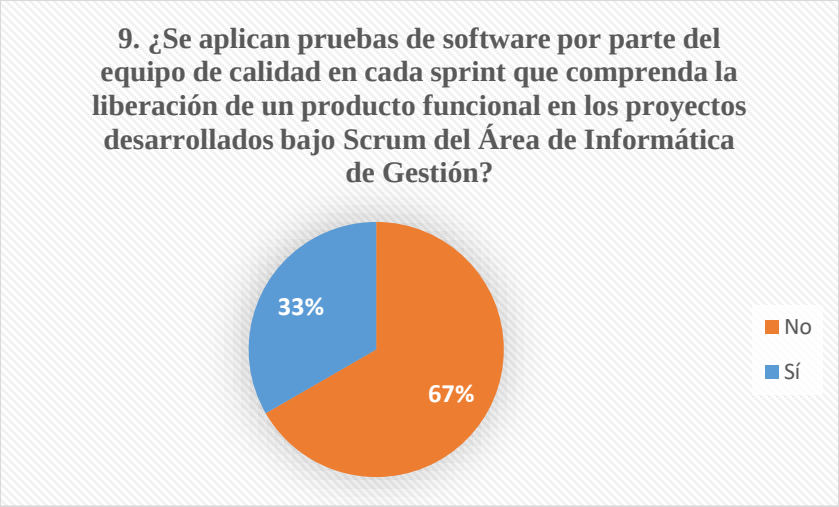


Figura 36: Aplicación de pruebas en cada sprint

Fuente: Elaboración propia, 2017.

En la Figura 37 se puede apreciar que 12 (80%) de la población encuestada indicaron no conocer los tipos de pruebas que el equipo de calidad aplica en los *sprints*, mientras que 3 (20%) indicaron sí tener conocimiento. Es importante informar a todo el equipo las pruebas realizadas en cada *sprint* y brindar un reporte de resultados que sea de conocimiento de todo el equipo, esto ayudaría a tener una retroalimentación de todos los miembros del equipo y fomentar la mejora continua tanto para el proceso de desarrollo de software como para el de *testing*.

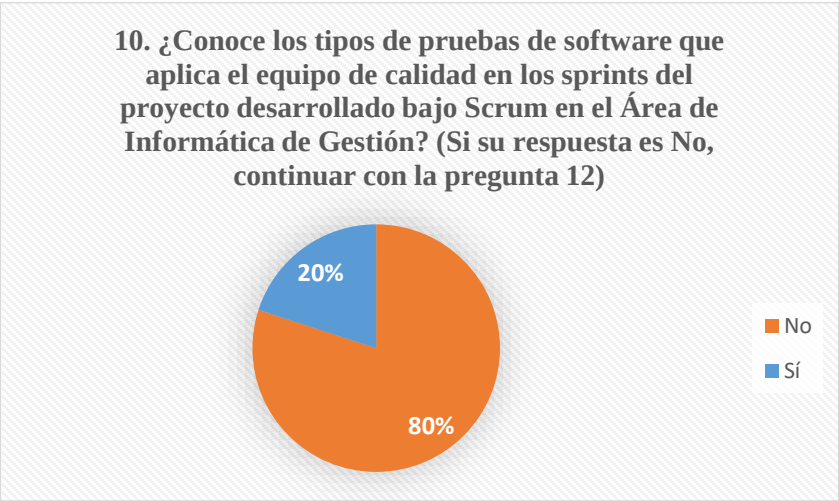


Figura 37: Conocimiento de los tipos de pruebas de software

Fuente: Elaboración propia, 2017.

Entre quienes indicaron conocer los tipos de pruebas que el área de calidad realiza, se quiso identificar cuáles eran esos tipos de pruebas, por lo que un dato curioso que se muestra en la Figura 38 es que las pruebas de estrés, rendimiento y carga fueron las que obtuvieron menos valor. Así se evidencia que actualmente este tipo de pruebas son prácticamente omitidas por el área, a pesar de ser muy importantes para brindar un producto de calidad a la institución.

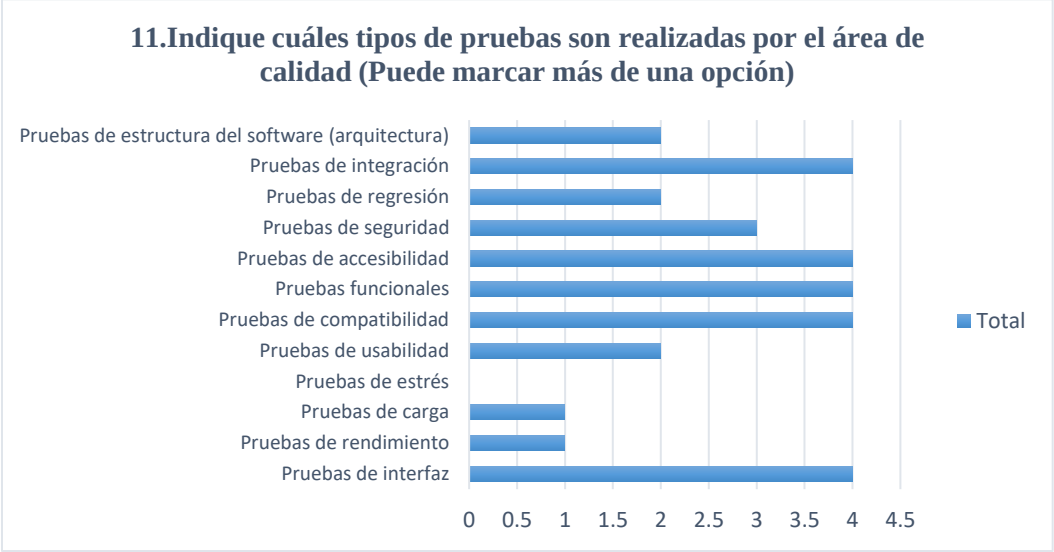


Figura 38: Identificación de los tipos de pruebas

Fuente: Elaboración propia, 2017.

De acuerdo a la Figura 39 podemos identificar que 14 integrantes del equipo Scrum (93%) consideran que el proceso de control de calidad actual no se encuentra alineado con la metodología de desarrollo Scrum. Algunas de las razones por las que indicaron lo anterior son: el equipo de calidad no asiste a las sesiones de planificación del sprint, tampoco forman parte de la reunión diaria, no asisten a las reuniones de retrospectiva al final del sprint y el proceso de pruebas no se encuentra integrado con el de desarrollo.

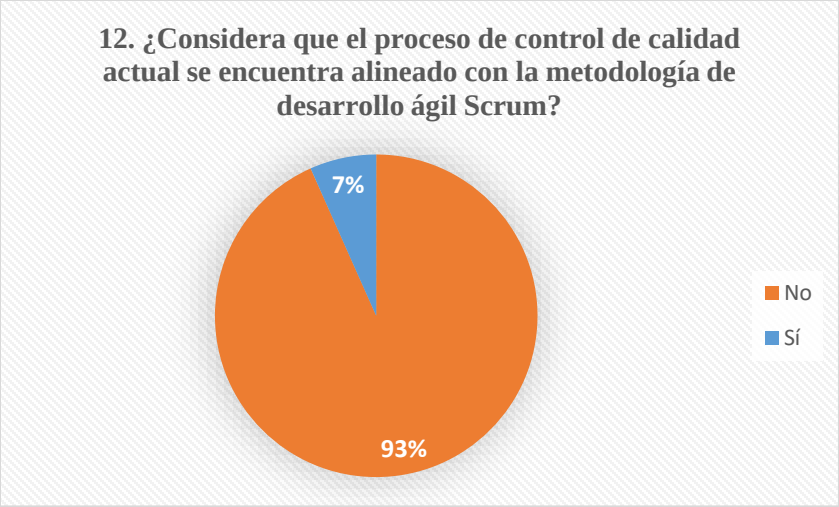


Figura 39: Proceso de control de calidad alineado a Scrum

Fuente: Elaboración propia, 2017

Más de la mitad de la población encuestada considera que la fiabilidad del proceso de control de calidad es regular, como se aprecia en la Figura 40. Esta calificación puede ser explicada ya que los casos de prueba no se documentan, por lo tanto se desconoce los escenarios de prueba que se aplicaron para considerar que el incremento o producto cumple con los requerimientos.

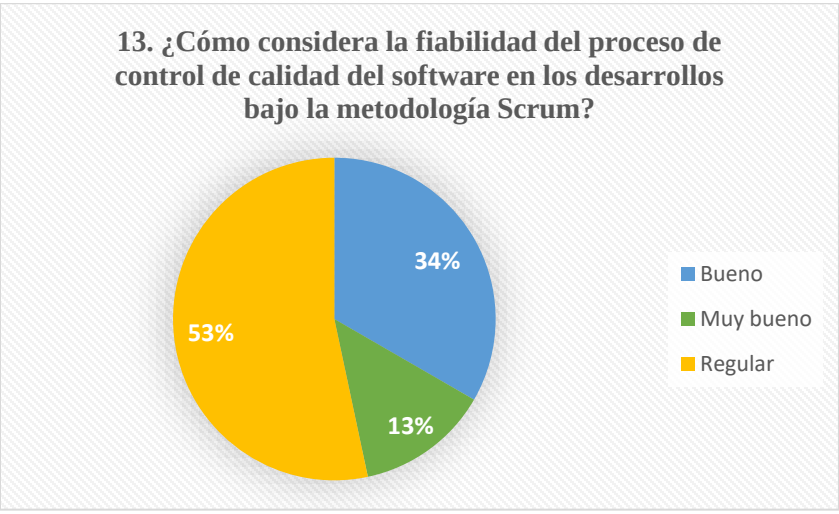


Figura 40: Fiabilidad del proceso de control de calidad

Fuente: Elaboración propia, 2017.

A pesar de que existen más desarrolladores que *testers* en el proyecto del SIAGPJ y que en cierta parte esto ocasiona que el proceso de control de calidad se vuelva más lento, 7 personas (47%) consideran que la eficiencia del proceso de control de calidad es bueno, mientras que 6 (40%) indicaron que es regular y únicamente 2 indicaron muy bueno, como se puede apreciar en la Figura 41.

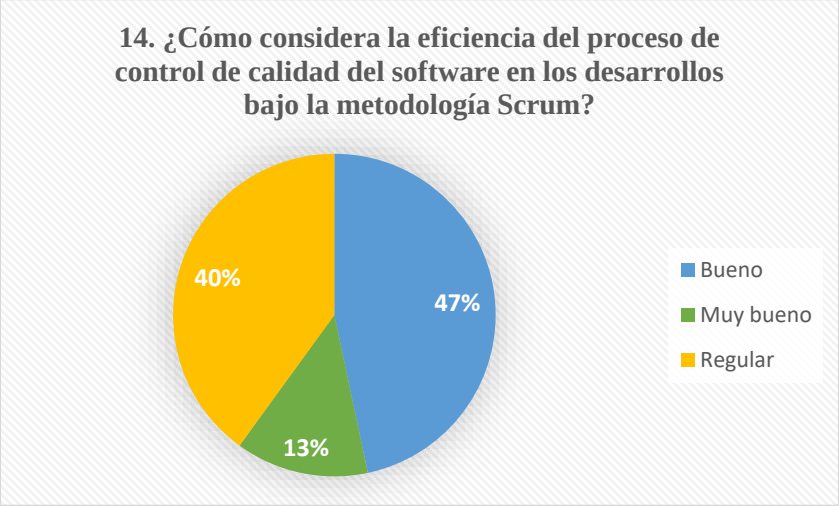


Figura 41: Eficiencia del proceso de control de calidad

Fuente: Elaboración propia, 2017.

De acuerdo con la población encuestada, el equipo de calidad debe fortalecer las pruebas de carga, pruebas de rendimiento y las pruebas de seguridad. Como se puede apreciar en la Figura 42, estas tres pruebas fueron las que tuvieron mayor puntaje.

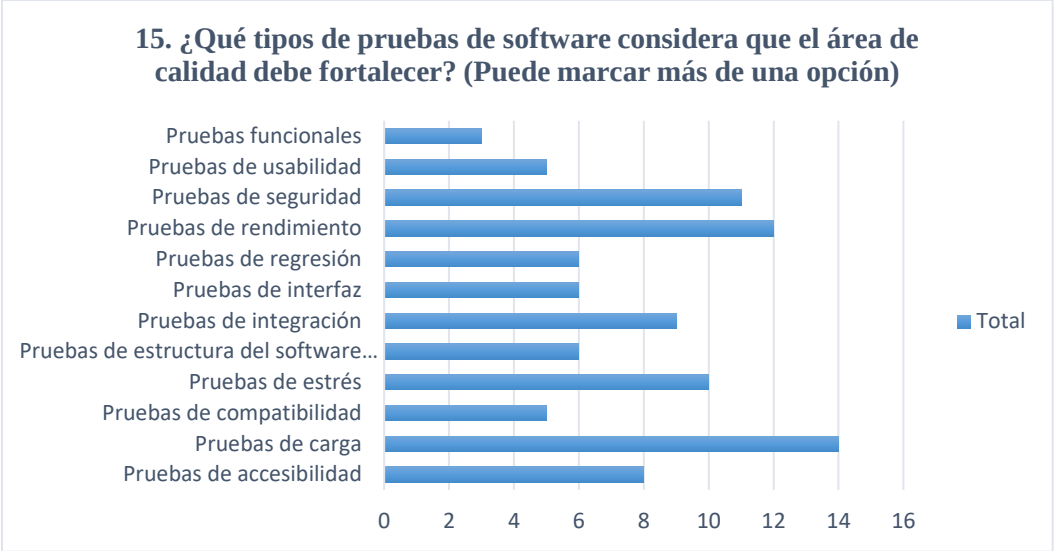


Figura 42: Tipos de pruebas que el área de calidad debe fortalecer

Fuente: Elaboración propia, 2017.

La comunicación es uno de los aspectos más importantes en Scrum, es por esto que al inicio, durante y al final de un *sprint* se realizan diferentes reuniones. Una buena comunicación entre los miembros del equipo permite que estos se centren en completar el trabajo para el sprint, además de mantener informados a los interesados del proyecto (Kasturi, 2014). De acuerdo con la población encuestada la comunicación entre los desarrolladores y el equipo de calidad es regular, lo que puede ser explicado ya que el equipo de calidad no forma parte de ninguna reunión de Scrum y adicionalmente la mayoría de los desarrolladores se encuentra físicamente en otro lugar.

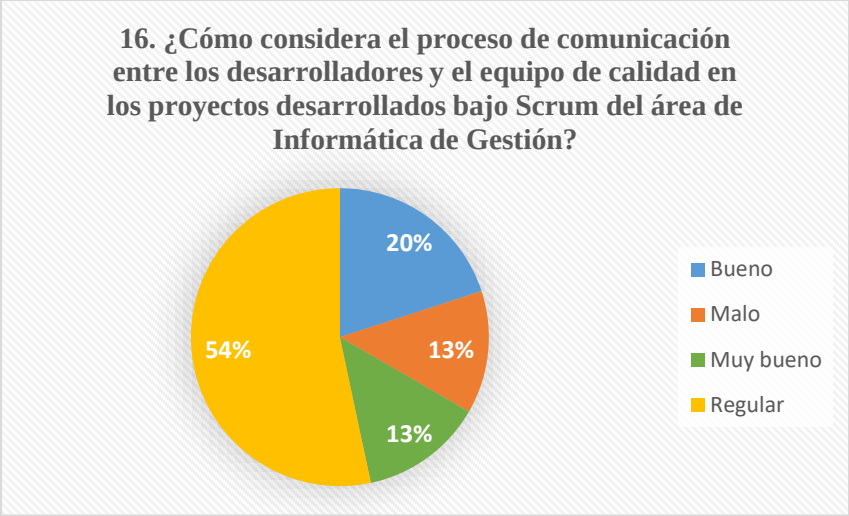


Figura 43: Comunicación entre desarrolladores y equipo de calidad

Fuente: Elaboración propia, 2017.

Por último, en la encuesta se quiso identificar cómo era el reporte de los defectos por parte del equipo de calidad tomando en cuenta que fueran completos, concisos, precisos y objetivos, para lo cual la mayoría de la población encuestada considera que es regular, mientras que 6 (40%) indicaron que es bueno. Dicha calificación puede ser explicada debido a que en muchas ocasiones los defectos reportados no son claros y los desarrolladores deben contactar al *tester* para que este les explique, o inclusive para identificar los datos o pasos que se utilizaron.

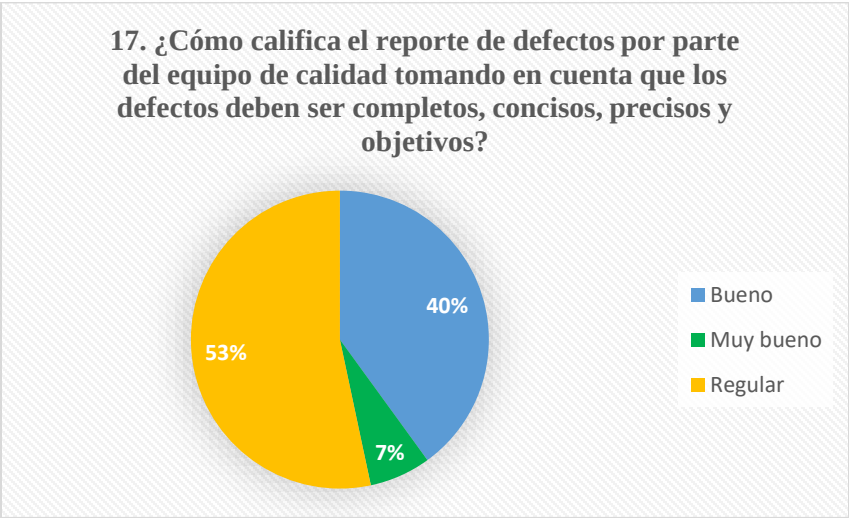


Figura 44: Calificación del reporte de defectos por parte del equipo calidad
Fuente: Elaboración propia, 2017.

4.1.1.2 Encuesta 2. Coordinador equipo calidad

Se realizó esta encuesta al coordinador del equipo de calidad. A continuación se detallan los resultados obtenidos de la encuesta.

4.1.1.2.1 Información sobre las personas encuestadas

En esta sección de la encuesta se identificó el grado académico que el coordinador del área de calidad obtiene, el cual corresponde a licenciatura. Con respecto a la experiencia laboral del encuestado, se logró identificar que el mismo tiene más de 7 años de experiencia laboral en el área del software, la mayoría de este tiempo se ha desempeñado en el área de calidad del software.

4.1.1.2.2 Información general sobre la metodología de desarrollo

Con respecto a la metodología Scrum, se logró identificar que el coordinador de calidad sí conoce Scrum y que además de conocerla hace uso de ella en su trabajo, a pesar de que no siempre utiliza o aplica la metodología. Como se explicó anteriormente, el equipo no forma parte de ninguna reunión de Scrum. Por otra parte, considera que la metodología Scrum es el marco de trabajo adecuado para el área de Informática de Gestión.

Por último, en esta sección de la encuesta, se realizó la siguiente pregunta: ¿Conoce bien sus roles y responsabilidades dentro de cada proyecto desarrollado bajo la metodología Scrum? El coordinador de calidad respondió sí conocer sus roles y responsabilidades. Sin embargo, es importante tener claro que el rol de un *tester* en una metodología ágil no es solo encontrar pulgas, va más allá que eso, es mejorar el software proporcionando información valiosa a lo largo del ciclo del desarrollo (uTest, 2011).

4.1.1.2.3 Información general sobre el proceso de control de calidad

En cuanto al proceso de control de calidad, se evidencia en la encuesta que actualmente el área de calidad no cuenta con una metodología para el control de la calidad del software adaptada a la metodología Scrum. Sí existe una planificación para llevar a cabo las tareas de control de calidad, sin embargo esta planificación solo se realiza cuando se va liberar un paquete para que el área de calidad realice su proceso de control de calidad. Scrum sugiere que todas las tareas relacionadas a la calidad sean completadas durante el *sprint*, para obtener al final un producto o incremento listo para ser puesto en producción.

Actualmente el área de Informática de Gestión tiene a cargo diferentes tipos de proyectos, de acuerdo con el coordinador de calidad el único tipo de aplicaciones a las cuales no se aplica control de calidad son las de inteligencia de negocios (véase Figura 45).

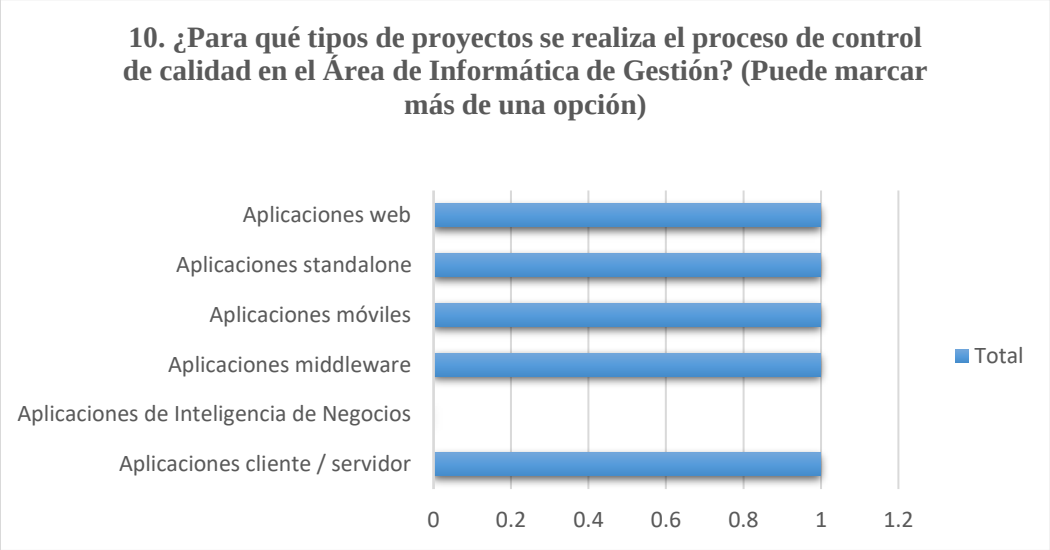


Figura 45: Proyectos a los que se aplica el control de calidad

Fuente: Elaboración propia, 2017.

Por otra parte, el coordinador de calidad indicó que sí existe comunicación entre el equipo de calidad y el equipo que planifica el proyecto, lo cual es uno de los aspectos más importantes en

una metodología ágil como Scrum. En cuanto a la existencia de una planeación formal documentada sobre la estrategia, técnicas, tipos de pruebas, herramientas y la estimación el encuestado, indicó que no existe dicha planeación formal documentada, pero sí se realiza una planeación informal, es decir no documentada, de las funcionalidades a probar.

Los casos de prueba se definen por varios aspectos: para asegurar la calidad, para tener una mejor cobertura de la pruebas, para depender de un proceso y no de una persona, para tener consistencia en la ejecución de las pruebas, para usarlos como prueba al cliente de las áreas de pruebas cubiertas, ayudan a proporcionar evidencia documental de lo que exactamente se ha probado (Sinhg, 2014). A pesar de que las metodologías ágiles se basan en el manifiesto ágil, donde uno de sus valores habla sobre valorar más el software que funciona que la documentación exhaustiva, en el Poder Judicial la documentación no puede dejarse a un lado. Las Normas de Control Interno para el Sector Público de la Contraloría General de la República (2009) indican:

> 4.2 Requisitos de las actividades de control
>
> Las actividades de control deben reunir los siguientes requisitos:
>
> […]
>
> e. Documentación. Las actividades de control deben documentarse mediante su incorporación en los manuales de procedimientos, en las descripciones de puestos y procesos, o en documentos de naturaleza similar. Esa documentación debe estar disponible, en forma ordenada conforme a criterios previamente establecidos, para su uso, consulta y evaluación (Contraloría General de la República, 2009, p.14).

De acuerdo con lo indicado por el coordinador de calidad, los casos de prueba casi nunca se realizan ni se documentan. Cada integrante del área de calidad prueba a criterio propio para detectar defectos o errores en el software.

Scrum recomienda que en cada sprint se realicen pruebas, sin embargo, actualmente de acuerdo con el coordinador de calidad, las pruebas solo se aplican cuando se tiene un módulo o una funcionalidad grande completa. La pregunta 14 del cuestionario es: ¿Se aplican pruebas de software por parte del equipo de calidad en cada sprint que comprenda la liberación de un producto funcional en los proyectos desarrollados bajo Scrum en el Área de Informática de Gestión? El coordinador indicó que sí se aplican pruebas en cada *sprint* que comprenda la liberación de un producto funcional.

Las pruebas de que se realizan actualmente por parte del equipo de calidad son las que se muestran en la Figura 46, donde se puede evidenciar que las pruebas de rendimiento, carga y estrés no se realizan.

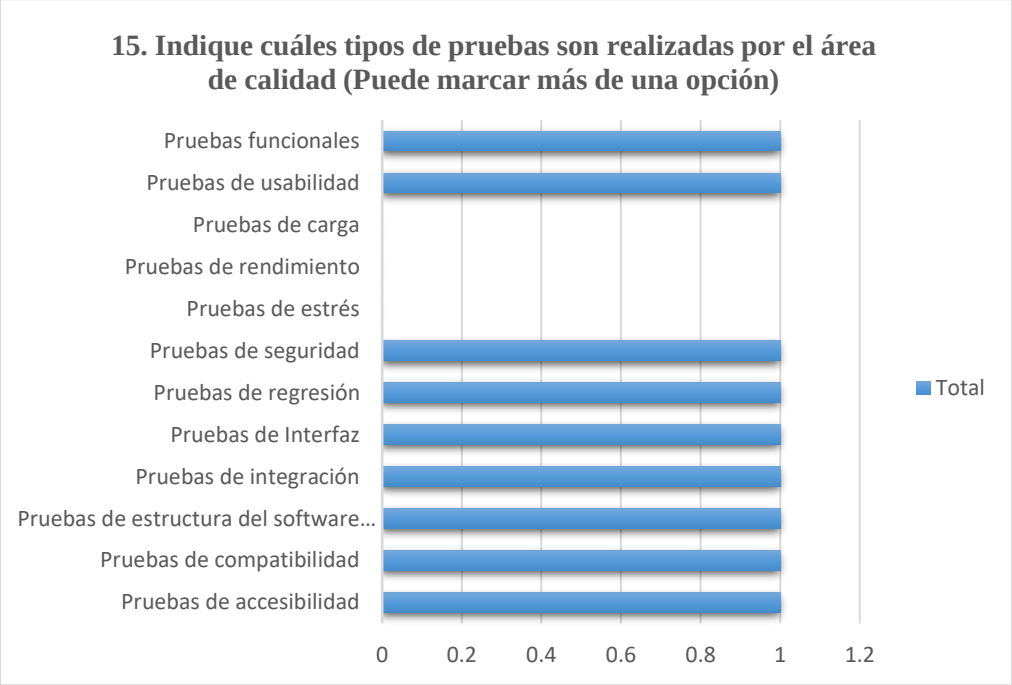


Figura 46: Identificación de los tipos de pruebas. Opinión coordinador de calidad

Fuente: Elaboración propia, 2017.

Otro aspecto importante que se recolectó en la encuesta fue a través de la pregunta 16, la cual indica: ¿Considera que el proceso de control de calidad actual se encuentra alineado con la

metodología de desarrollo ágil Scrum? El coordinador de calidad respondió que no, ya que dicho proceso se está realizando al final luego de que se ha desarrollado toda una funcionalidad y no como debería realizarse con Scrum, que recomienda realizar dicho proceso en cada sprint con pequeñas funcionalidades.

De acuerdo con el coordinador de calidad se deben fortalecer las pruebas de usabilidad, pruebas de rendimiento, pruebas de carga y las pruebas de estrés, como se puede distinguir en la Figura 47. A pesar de que ya se realizan pruebas de usabilidad, el coordinador considera que no se tiene un buen manejo sobre dichas pruebas, ya que se realizan a criterio propio.

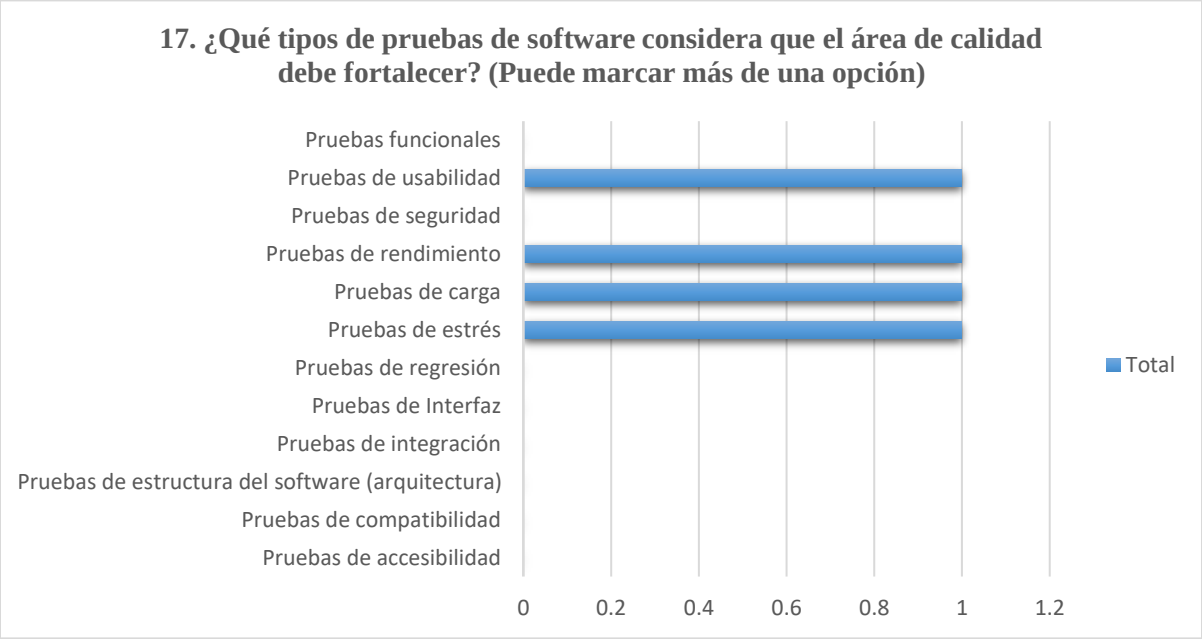


Figura 47: Tipos de pruebas que el área de calidad debe fortalecer. Opinión coordinador de calidad

Fuente: Elaboración propia, 2017.

Con respecto a la comunicación entre el equipo de calidad y los desarrolladores, el coordinador indicó que dicha comunicación es regular, ya que algunos de los desarrolladores realizan teletrabajo y en algunas ocasiones se hace más difícil contactarlos. Otro aspecto importante es que muchas veces los desarrolladores detectan ciertos defectos en el software a

nivel de código y no se le informa al equipo de calidad de dicho cambio. Por otra parte, se identificó que el equipo de calidad no tiene contacto con los usuarios, en el caso de las pruebas de usabilidad y de accesibilidad es importante que el equipo de calidad tenga contacto con usuarios para facilitar este tipo de pruebas.

El área de calidad cuenta con la herramienta de Microsoft llamada Test Manager, en la encuesta se quiso conocer la frecuencia de uso de esta herramienta, para lo cual el coordinador respondió que a veces la utilizan más que todo para registrar las pulgas que se encuentren en los entregables. Una herramienta de pruebas es aquella que da apoyo a una o más actividades, tales como la planeación y control, especificación, ejecución y análisis de la prueba. Hace que el proceso de pruebas sea más eficiente al reducir el tiempo de las tareas repetitivas y permitir que el *tester* se centre en tareas como la planeación, el análisis y el diseño de la prueba (Hambling, Morgan, Samaroo, Thompson, & Williams, 2010).

Una de las últimas preguntas realizadas en la encuesta fue: ¿Se utiliza algún sistema de información para el registro de las pruebas? El coordinador respondió que no existe un sistema de información como tal, pero que sí se registran propiamente en una base de datos de Access. Por último, el encuestado indicó que no existe un proceso de validación del trabajo realizado por el equipo de calidad, esto es un problema ya que no existen métricas o procesos que aseguren que el trabajo realizado por el equipo de calidad está bien.

4.1.1.3 Entrevista. Jefe del área

De acuerdo con la entrevista realizada al jefe del área de Informática de Gestión, se obtienen los siguientes resultados (para más detalle ver Anexo 7: Respuestas de la entrevista realizada al jefe del área).

El área de calidad surgió como una necesidad del departamento para minimizar los errores que se presentaban en producción y mejorar la calidad de los sistemas del área. A raíz de los problemas que se generaban con la metodología de desarrollo en cascada, la DTIC inició un plan piloto utilizando la metodología Scrum para el desarrollo del sistema SIAGPJ. Se logró evidenciar en la entrevista que la metodología Scrum no se encuentra oficialmente documentada dentro de los procedimientos de la DTIC, sin embargo, sí se encuentra debidamente documentada en el estudio de factibilidad del proyecto SIAGPJ. Por otro lado, se determinó que la jefatura considera que Scrum es el marco de trabajo adecuado que el área necesita, ya que se ha obtenido mejoras en cuanto a la productividad del equipo y el tener una mejor estimación de los tiempos.

Uno de los problemas mencionados por el jefe en cuanto al proceso de control de calidad en el proyecto SIAGPJ es el no tener dicho proceso integrado al proceso de desarrollo en cada *sprint*, y que este proceso de calidad se aplique de forma correctiva en lugar de ser preventivo. Por otra parte, algunos de los aspectos que se deben mejorar en el área de calidad es el tema de las pruebas de rendimiento, carga y estrés, y adicionalmente el tema de investigación e innovación por parte del equipo de calidad. Por otro lado, la jefatura espera que la propuesta de la metodología del presente proyecto de investigación traiga beneficios como lo es mejorar la gestión de la calidad, incorporar la aplicación de pruebas de rendimiento y carga, que el equipo trabaje con procedimientos ágiles que se adapten a la metodología de Scrum.

Otro aspecto importante recolectado en la entrevista es que el área no tiene forma de identificar si los procesos de calidad son caros o no. Los costos de la calidad son todos aquellos costos o actividades en las que se busca la calidad. El costo de la calidad se asocia con la prevención, la evaluación y la falla, pero sin duda, la mala calidad también tiene su costo. Es

importante que las organizaciones cuenten con unidades de medición que proporcionen el fundamento del costo de la calidad para facilitar la ejecución de mejoras tendientes a la reducción de los costos (Pressman, 2010).

4.1.2 Análisis FODA del proceso de control de calidad

De acuerdo con los datos suministrados en ambas encuestas, entrevista y a través del método de observación, en la Tabla 11 se presenta el análisis FODA del proceso de control de calidad actual en el Área de Informática de Gestión.

Tabla 11: Análisis FODA del proceso de control de calidad

Fortalezas	**Debilidades**
• El equipo de calidad cuenta con un alto conocimiento del negocio, lo que facilita la realización de pruebas funcionales. • Compromiso del equipo en el trabajo diario. • Conocimiento de la metodología Scrum. • Coordinador de calidad con más de 7 años de experiencia en el área del software.	• No existe una buena comunicación entre el equipo de calidad y desarrolladores. • No existe un proceso de validación del trabajo realizado por el equipo de QA. • La fiabilidad del proceso de control de calidad es regular. • El proceso de control de calidad no se encuentra alineado con Scrum. • El equipo de calidad no forma parte de la planificación del sprint ni de las reuniones de Scrum. • El equipo de calidad no tiene contacto con los usuarios de la institución. • Poco recurso humano de QA. • No existe una planeación documentada de las pruebas. • El no documentar los casos de prueba hace que no se tenga evidencia de lo que realmente se probó. • No existen unidades de medición para los costos de calidad. • Los defectos reportados no siempre son claros. • Cada integrante del equipo de QA define el mejor proceso a seguir para ejecutar el control de la calidad, no es un proceso integral. • Falta de capacitación en pruebas de usabilidad, seguridad, carga, rendimiento y estrés.

Oportunidades	**Amenazas**
• Proceso de control de calidad preventivo en lugar de correctivo para disminuir los costos de las reparaciones. • El área cuenta con presupuesto para que el equipo de calidad pueda recibir capacitaciones o consultorías en temas con falencias. • Aplicar pruebas en cada sprint para funcionalidades pequeñas permite tener una mejor calidad en el software. • El uso de la metodología Scrum en el área. • Tener un área de calidad que abarque todo lo que sale del área de informática de gestión. • Equipo de calidad cuenta con la licencia Enterprise de Microsoft, lo que permite el uso de la herramienta Test Manager y Visual Studio para las pruebas. • Apoyar la mejora continua a través de las lecciones aprendidas.	• Un control de calidad que no toma en cuenta las pruebas de rendimiento o carga aumenta la probabilidad de defectos e inconformidades de los usuarios del Poder Judicial.

Fuente: Elaboración propia, 2017.

CAPÍTULO V

5. Análisis de brechas

En este capítulo se realiza un análisis de brechas entre el proceso de control de calidad actual del Área de Informática de Gestión contra lo establecido en las metodologías de calidad de software enfocadas a ambientes ágiles. El análisis se realiza con base en el ciclo de vida de Scrum, el cual se relaciona el ciclo de Deming conocido también como ciclo PDCA (o ciclo PHVA), siglas que corresponden a sus cuatro etapas: Planificar, Hacer, Verificar y Actuar. El ciclo PHVA es una estrategia de mejora continua de la calidad, que como se aprecia en la Figura 48 se adapta a un sprint de la metodología Scrum.

Dicho ciclo inicia con la fase de planificación donde se establecen los objetivos y los procesos necesarios para lograr los resultados de acuerdo con lo solicitado por el cliente y con las políticas de la empresa, la fase hacer corresponde al desarrollo de los pasos de acuerdo con la planificación. Seguidamente en la fase verificar se evalúan los resultados obtenidos y se verifica que los procesos realizados se hayan ejecutado de acuerdo con lo planeado, y finalmente en la fase actuar se toman decisiones, se estandarizan los cambios, formar y entrenar. Una vez acabada la fase actuar se debe volver a la primera fase y repetir el ciclo nuevamente.

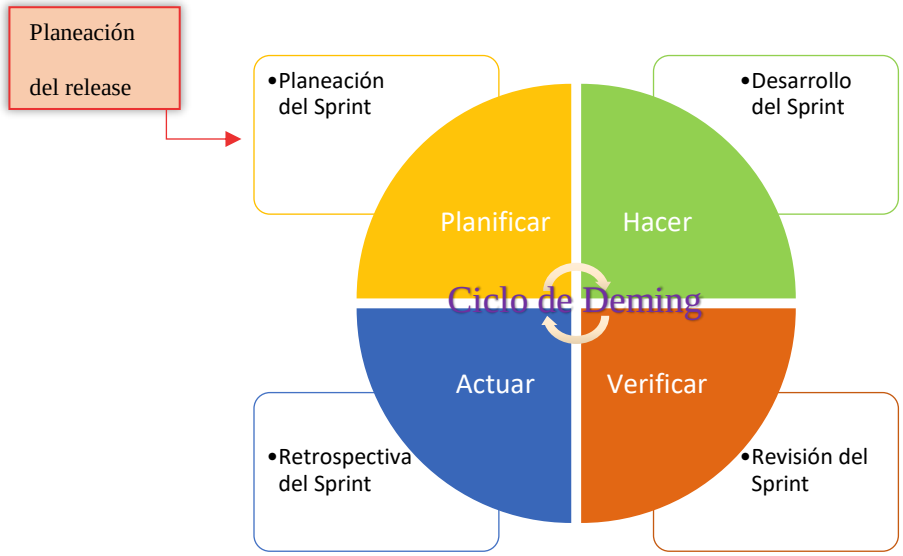


Figura 48: Ciclo de sprint de Scrum y su relación con el Ciclo de Deming

Fuente: Elaboración propia, 2017.

De acuerdo con las cinco etapas de Scrum de la Figura 48, se realiza el análisis del procedimiento actual de control de calidad contra el procedimiento que establece la metodología ágil de *testing* basado en el estándar ISTQB.

Tabla 12: Planeación del release

Etapa en Scrum: Planeación del release o producto	
Metodología Ágil de Pruebas ISTQB	Procedimiento actual del equipo de calidad del área de informática de gestión.
Se define los miembros de *testing* que serán responsables de asegurar la calidad del software al proyecto.	Se define los miembros de *testing* que serán responsables de asegurar la calidad del software al proyecto.
El equipo de *testing* realiza un plan de pruebas de alto nivel que abarca varias iteraciones.	No se realiza ningún plan de pruebas, ni se documenta.
Se realiza un rápido análisis de riesgos de las historias de usuario y se planea un enfoque de pruebas para abordar esos riesgos.	El equipo de calidad no realiza análisis de riesgos sobre las historias de usuario.
Se debe especificar el ambiente de pruebas, crear los datos de prueba que serán usados para propósitos de pruebas.	El ambiente de pruebas se planea informalmente, muchas veces los datos de prueba no son contemplados hasta el momento de la ejecución de las pruebas, lo que genera atrasos en la entrega del producto.
Determinar la estrategia de pruebas a alto nivel para todo el *release*.	No se elabora una estrategia de pruebas.

Etapa en Scrum: Planeación del release o producto	
Metodología Ágil de Pruebas ISTQB	Procedimiento actual del equipo de calidad del área de informática de gestión.
Definir los niveles de pruebas a realizar.	No se definen los niveles de pruebas.

Fuente: Elaboración propia, 2017.

Tabla 13: Planeación del sprint

Etapa en Scrum: Planeación del Sprint	
Metodología Ágil de Pruebas ISTQB	**Procedimiento actual del equipo de calidad del área de informática de gestión**
Conocer la meta u objetivo del sprint y seleccionar las historias de usuario que se probarán en dicho sprint.	El equipo de calidad no prueba por sprint.
Identificar los riesgos de calidad asociados a cada historia de usuario, con el resto del equipo Scrum.	No se identifican riesgos.
Evaluar cada riesgo identificado: - Categorizar el riesgo - Determinar su nivel de riesgo en función del impacto y la probabilidad de defectos.	No se identifican riesgos.
Definir los tipos de pruebas necesarios.	Los tipos de pruebas se planean durante la ejecución de las pruebas.
Identificar los datos necesarios de pruebas, el ambiente de pruebas, la infraestructura necesaria y las herramientas para la ejecución de las pruebas.	En muchas ocasiones se llegan a identificar los datos necesarios de pruebas, el ambiente de pruebas, la infraestructura necesaria y las herramientas durante la ejecución de las pruebas.
Definir, para cada historia de usuario seleccionada, las tareas necesarias que debe desempeñar el equipo de calidad para cumplir con los objetivos del sprint, lo que tiene como resultado el *sprint backlog*.	Actualmente el equipo de calidad no registra las tareas que se deben realizar para completar el proceso de pruebas.
Estimar el esfuerzo para todas las tareas de pruebas.	La estimación se realiza a nivel general y no por las tareas.
Apoyar y participar en pruebas de automatización en los diferentes niveles de pruebas.	El equipo de calidad no realiza pruebas automatizadas.
Actualizar el plan o la estrategia de pruebas	No se realiza una planeación previa documentada de las pruebas a realizar.

Fuente: Elaboración propia, 2017.

Tabla 14: Desarrollo del sprint

Etapa en Scrum: Desarrollo del Sprint	
Metodología Ágil de Pruebas ISTQB	**Procedimiento actual del equipo de calidad del área de informática de gestión**

Etapa en Scrum: Desarrollo del Sprint	
Metodología Ágil de Pruebas ISTQB	**Procedimiento actual del equipo de calidad del área de informática de gestión**
Creación y ejecución de casos de pruebas.	Los casos de prueba no se documentan, las pruebas no se basan en casos de pruebas previamente diseñados.
Ejecutar los diferentes tipos de pruebas planeados en la etapa anterior en los diferentes niveles, para cada historia de usuario seleccionada.	Se realizan pruebas funcionales, sin embargo las pruebas de carga, rendimiento o estrés no se están realizando. Adicionalmente estas pruebas no se realizan dentro del ciclo de un sprint.
Participar en la reunión diaria Scrum, comunicar los impedimentos y las tareas próximas por hacer. Compartir conocimiento e identificar riesgos.	QA no participa en la reunión diaria de Scrum. No comunica las tareas próximas por hacer ni las realizadas.
Documentar los defectos.	Los defectos se documentan.

Fuente: Elaboración propia, 2017.

Tabla 15: Revisión del sprint

Etapa en Scrum: Revisión del Sprint	
Metodología Ágil de Pruebas ISTQB	**Procedimiento actual del equipo de calidad del área de informática de gestión**
El equipo de calidad presenta lo que ha logrado durante el sprint.	El equipo de calidad no forma parte de la reunión de revisión del sprint.

Fuente: Elaboración propia, 2017.

Tabla 16: Retrospectiva del sprint

Etapa en Scrum: Retrospectiva del Sprint	
Metodología Ágil de Pruebas ISTQB	**Procedimiento actual del equipo de calidad del área de informática de gestión**
El equipo de calidad comparte con el resto del equipo Scrum los cambios que se podrían hacer para mejorar en cuanto a personas, procesos, relaciones, herramientas de pruebas en el próximo sprint de acuerdo con el sprint que recién ha finalizado.	El equipo de calidad no forma parte de la reunión de retrospectiva. No se fomenta la mejora continua en el área de calidad.

Fuente: Elaboración propia, 2017.

5.1 Brechas encontradas en la etapa de planeación del release

Tabla 17: Brechas encontradas etapa planeación del release

Brecha encontrada	Riesgo

Brecha encontrada	Riesgo
La no participación del equipo de calidad en la etapa de planificación del *release*.	El tener un equipo de calidad que desconoce la visión común sobre lo que se debe lograr y cuándo, puede repercutir en un proceso de pruebas de baja calidad.
No se realiza un plan de pruebas a alto nivel para el proyecto a probar.	El no tener un plan de pruebas del proceso de *testing* que brinde apoyo al ciclo del desarrollo del software de un proyecto genera que el equipo no tenga claros los diferentes tipos de pruebas que deben realizarse en el sistema, y existe una mayor probabilidad de que las funcionalidades críticas o esenciales del sistema no se prueben adecuadamente.
El equipo de calidad no participa en la identificación y evaluación de riesgos de calidad y del proyecto como tal.	Entrega de un producto no completo y que no cumple con las necesidades del cliente.
No se definen los niveles de pruebas.	Al no definir los niveles de pruebas se corre el riesgo de que algunas partes del sistema no sean probadas desde diferentes aspectos.
Planeación pobre sobre el entorno de pruebas y los datos de pruebas.	Atraso en los entregables.
Falta de una estrategia de pruebas de alto nivel, que contemple varias iteraciones del proyecto.	Aplicación de pruebas de forma desorganizada, lo que afecta la fiabilidad del proceso de pruebas y por consiguiente la calidad del producto.

Fuente: Elaboración propia, 2017.

5.2 Brechas encontradas en la etapa de planeación del sprint

Tabla 18: Brechas encontradas etapa planeación del sprint

Brecha encontrada	Riesgo
El equipo de calidad no forma parte de la reunión que conlleva la planificación del sprint.	Mala interpretación de las historias de usuario por parte del equipo de calidad, lo cual perjudica el tiempo de entrega y la calidad del producto.
No se identifican ni se evalúan los riesgos de calidad que puedan tener las historias de usuario.	Ejecución de pruebas exhaustivas en aspectos del sistema de menor impacto al negocio, se dejan de lado las pruebas de mayor impacto.
No se realiza una planeación de los tipos de pruebas a realizar.	Probabilidad de que se omitan tipos de pruebas durante la ejecución de las pruebas.
La estimación del proceso de control de calidad en los *sprints* se realiza a nivel general y no de tareas.	Al no realizar una estimación basada y fundamentada en las tareas que se deben ejecutar, se da la probabilidad de que los tiempos establecidos por el equipo de calidad no se

	cumplan.

Fuente: Elaboración propia, 2017.

5.3 Brechas encontradas en la etapa de desarrollo del sprint

Tabla 19: Brechas encontradas etapa desarrollo del sprint

Brecha encontrada	Riesgo
Las pruebas realizadas no se basan en casos de prueba previamente diseñados por el equipo de QA.	No se tiene evidencia de las pruebas realizadas por el equipo de QA.
Carencia de un proceso de pruebas de rendimiento, carga y estrés.	Afecta la calidad del software y la imagen de la institución.
La no participación del equipo de calidad en la reunión diaria de Scrum.	No se tiene un control sobre las labores diarias del equipo de calidad, adicionalmente la no participación afecta el tema de la comunicación entre desarrolladores y equipo QA.

Fuente: Elaboración propia, 2017.

5.4 Brechas encontradas en la etapa de revisión del sprint

Tabla 20: Brechas encontradas etapa revisión del sprint

Brecha encontrada	Riesgo
La no participación del equipo de calidad en la reunión de revisión del sprint, en donde es partícipe el cliente para dar retroalimentación de lo desarrollado.	Poca visión por parte del equipo de calidad, al momento de crear y ejecutar casos de prueba. Esto afecta el tiempo de entrega del producto.

Fuente: Elaboración propia, 2017

5.5 Brechas encontradas en la etapa de retrospectiva del sprint

Tabla 21: Brechas encontradas etapa retrospectiva del sprint

Brecha encontrada	Riesgo
El equipo de calidad no participa en la reunión de retrospectiva del sprint.	La resistencia al cambio genera que los integrantes de calidad sigan utilizando procedimientos que no se ajustan a una metodología ágil, lo que afecta la entrega del producto. La reunión de retrospectiva permite que un equipo de QA revise sus procedimientos utilizados, ver si hay que mejorarlos o no; actualmente esta revisión no se realiza a lo interno en el propio equipo sino que las sugerencias de mejoras de procesos, usualmente provienen de la

	jefatura.

Fuente: Elaboración propia, 2017.

CAPÍTULO VI

6. Propuesta del proyecto

De acuerdo con los resultados obtenidos presentes en el capítulo IV y el análisis de brechas realizado en el capítulo V se presenta la propuesta de la metodología de control de calidad del software adaptada a Scrum. La metodología propuesta incorpora recomendaciones y principios del estándar ISTQB, así como del modelo de calidad TMMi.

6.1 Visión general de la metodología propuesta

En la Figura 49 se observa de forma esquemática la metodología de control de calidad de software basada en Scrum que se propone para el Área de Informática de Gestión del Poder Judicial.

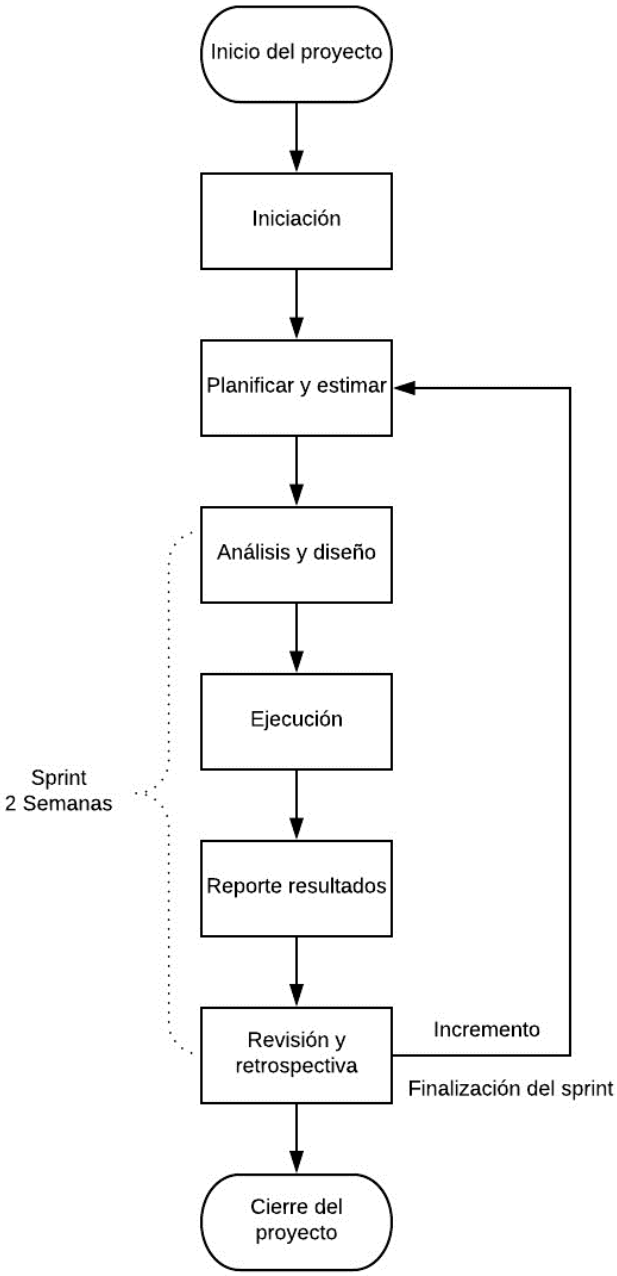


Figura 49: Vista general de la metodología de control de calidad propuesta

Fuente: Elaboración propia, 2018.

En las siguientes secciones se explicarán en detalle cada uno de los procesos que componen la metodología.

6.2 Definición de roles

6.2.1 Líder de pruebas

Este rol corresponde a aquel especialista con amplio conocimiento en el proceso de pruebas de software, administración de la calidad, administración de proyectos y en la gestión de personal. Entre las tareas que este rol debe realizar se mencionan las siguientes:

- Desarrollar el enfoque de las pruebas y el plan de pruebas.
- Gestionar los recursos de pruebas.
- Seleccionar e introducir estrategias y métodos de prueba adecuados.
- Seleccionar o mejorar herramientas de prueba.
- Establecer el entorno de pruebas.
- Introducir u optimizar procesos de pruebas
- Comunicar los resultados de las pruebas.

Cabe mencionar que el líder de pruebas también realiza las tareas del rol de *tester*, el cual se menciona a continuación.

6.2.2 Tester

El *tester* es un analista y experto en la definición de las pruebas con amplio conocimiento en el campo de pruebas de software e ingeniería del software. Se encarga de revisar los requerimientos y especificaciones para el diseño de los casos de prueba y la preparación de los datos de prueba. Adicionalmente se encarga de instalar, operar, configurar y dar mantenimiento al ambiente de pruebas. Ejecuta los casos de prueba definidos previamente y todas aquellas pruebas definidas en el plan de pruebas, utiliza herramientas de pruebas y de monitoreo, documenta los resultados de las pruebas así como los defectos encontrados.

Tanto el líder de pruebas como el *tester* se encuentran dentro del rol equipo de desarrollo definido por la metodología Scrum.

6.3 Iniciación

Esta etapa comprende el inicio del proyecto y la preparación de los insumos necesarios para los procesos de planificación. Se genera una visión general del proyecto, se establecen cronogramas de entregas, entrenamiento o talleres necesarios para el comienzo del desarrollo del proyecto y se conforma el equipo Scrum, el cual comprende tres roles principales: el *product owner*, el *scrum master* y un equipo de desarrollo. Este último comprende un grupo de personas expertas en diferentes ámbitos, por ejemplo: desarrolladores de software, administrador de proyectos, analistas de sistemas, especialistas en arquitectura del software, usuarios expertos del negocio, administradores de bases de datos, representantes del negocio, analistas de calidad del software, *testers* de software, etc. En la Figura 50 se muestra de manera general los procesos involucrados en la etapa de iniciación para el área de calidad.

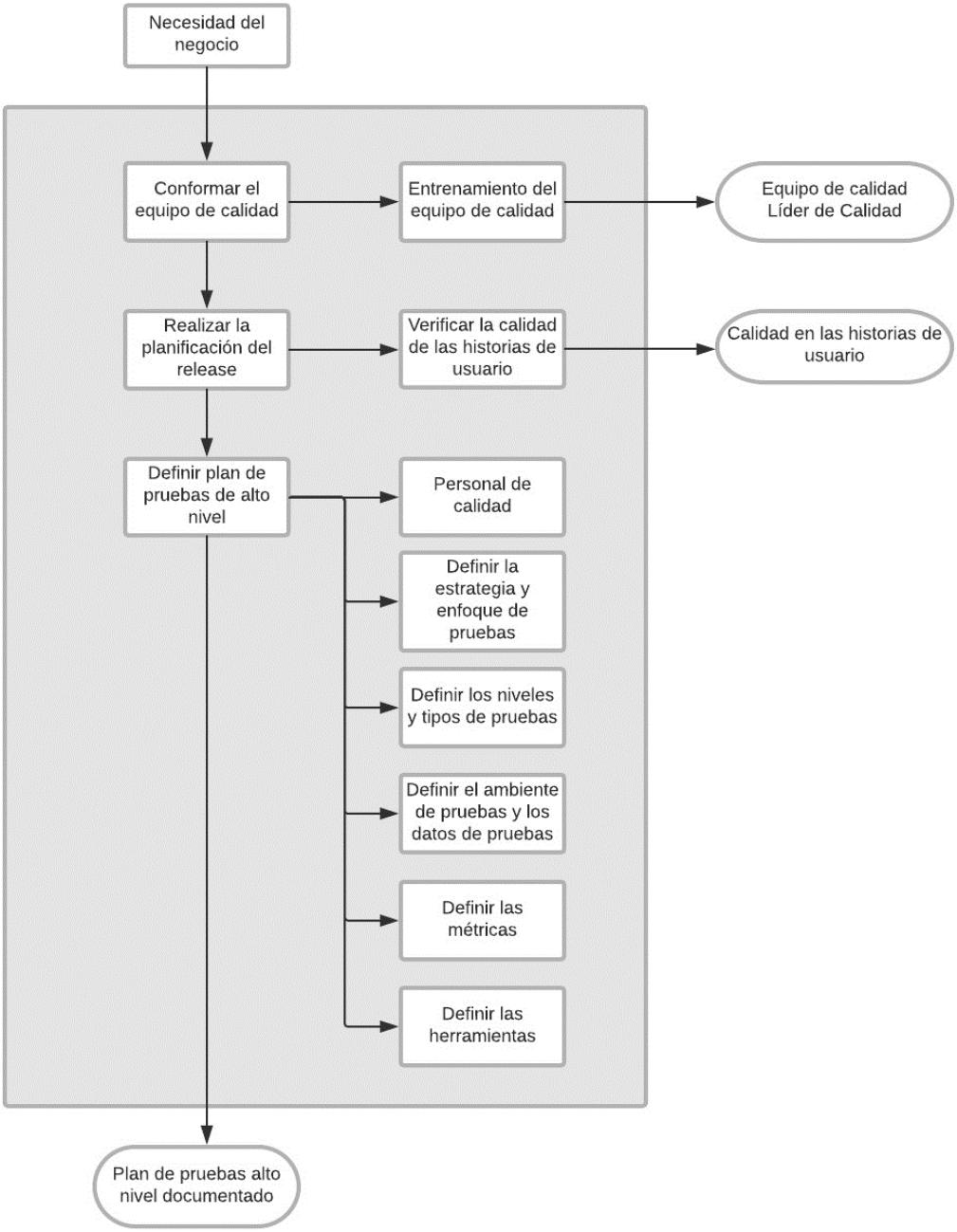


Figura 50: Procesos de calidad en la etapa de iniciación

Fuente: Elaboración propia, 2018.

Seguidamente se describe cada uno de los procesos:

6.3.1 Conformar el equipo de calidad

Se establece el equipo y el líder de calidad que se encargará del proceso de control de calidad en cada sprint. Es importante establecer el tiempo que este equipo dedicará al proyecto, para más adelante poder realizar la estimación de tiempos en cada tarea de *testing*.

6.3.2 Realizar la planificación del release

La planeación del *release* es un proceso de inicio de planificación, que se realiza con frecuencia a lo largo del proyecto, por lo general, una vez cada cuatro o seis *sprints*, siempre que se inicia un proyecto o alguna de las fases del mismo. En la reunión de planeación del *release* se llevan a cabo los siguientes eventos:

- Se define la duración de cada *sprint*, en el caso del área de informática de gestión se tiene una duración de dos semanas.
- El equipo Scrum y los involucrados claves del proyecto discuten los ítems que se encuentran en el *product backlog* para determinar las funcionalidades que serán entregadas en el siguiente *release*.
- Se crea un cronograma del *release*, en el que se describe para cada sprint cuáles son las funcionalidades que serán entregadas al cliente.

Una vez que se tiene como insumos los elementos priorizados que se desarrollarán en los siguientes *sprints*, a continuación se detalla cada tarea que conlleva la planificación del *release* desde la perspectiva de un equipo de calidad:

6.3.2.1 Definir plan de pruebas de alto nivel

El plan de pruebas a realizar en la planificación del *release* permite dar una guía general del proceso de pruebas durante el proyecto y a la vez ayuda a que los individuos fuera del equipo de

pruebas puedan entender el proceso a nivel general de control de calidad. Este plan debe actualizar cada vez que haya algún cambio en la planeación de posteriores *releases*. En el Anexo 8 se puede observar la plantilla de plan de pruebas que se propone, ya sea que se utilice como un documento impreso o electrónico, lo importante es establecer un estándar de los elementos mínimos que debe contener un plan de pruebas. La plantilla contiene los siguientes elementos:

a) Proporcionar una visión general del plan mediante los siguientes datos:
    i. Proporcionar un identificador al plan de pruebas
    ii. Fecha de elaboración del plan
    iii. Dar una descripción del sistema
b) Describir el personal de calidad asignado al proyecto o al menos para el *release* planeado: Proporcionar nombre, el rol asignado y el tiempo que dedicará al proyecto.
c) Describir la estrategia y el enfoque de las pruebas.
d) Describir niveles y tipos de pruebas
    i. Especificar los niveles de pruebas que se realizarán
    ii. Definir los tipos de pruebas que se ejecutarán
e) Definir el alcance de la prueba, especificando lo que se va probar: Indicar los módulos o entregables
f) Definir las métricas que permitan monitorear y controlar el proceso de pruebas. Las métricas ayudan a mejorar un proceso para lograr una mejor productividad, calidad en los entregables y con eso satisfacer al cliente.
        i. Número de casos de pruebas realizados
        ii. Número de defectos encontrados

g) Especificar las nuevas características o requerimientos que el ambiente de pruebas necesita para el sprint: Definir requerimientos de software, hardware o propiamente de espacio físico.

h) Definir las herramientas para la gestión de la calidad. Se deben determinar las herramientas que apoyen los procesos de control de calidad para lograr los objetivos del área. Para este caso se propone la utilización de tres herramientas para el proceso de pruebas, una de ellas ya se encuentra disponible en la institución pero no es aprovechada por completo en el área de calidad, la cual es *Microsoft Test Manager* o bien *Microsoft Visual Studio Team Services* para la gestión de defectos, creación y ejecución de casos de prueba, ejecución de pruebas exploratorias. La otra herramienta que también se encuentra disponible para el área de calidad y que no se le da un aprovechamiento efectivo es *Microsoft Visual Studio Enterprise 2017*, esta herramienta se propone para la ejecución de pruebas de rendimiento, carga y estrés.

i) Por último, se propone que el plan de pruebas sea aprobado ya sea por el *product owner* o algún encargado del proyecto.

6.3.2.2 Verificar la calidad de las historias de usuario

La calidad de un producto nace desde el levantamiento de los requerimientos, es por esto que en la etapa inicial se recomienda que el equipo de calidad realice la verificación de las historias de usuario utilizando una lista de chequeo que permite evaluar la calidad de las mismas. Para esto se propone la plantilla que se encuentra en la sección de anexos (ver Anexo 9: Plantilla de verificación de historias de usuario), la cual se basa en la técnica INVEST por sus siglas en inglés. La plantilla permite registrar las historias de usuario que no cumplan con alguna

característica y que las mismas sean evaluadas por el resto del equipo junto con el *product owner*.

6.4 Planificar y estimar

Esta es la etapa donde el equipo de calidad forma parte del evento de Scrum llamado planificación del *sprint*, donde se realizan todas aquellas actividades relacionadas con la planeación de las pruebas para el *sprint*. Es importante mencionar que la base o insumo de esta etapa son las historias de usuarios. A continuación en la Figura 51 se muestra de manera general los procesos involucrados en esta etapa.

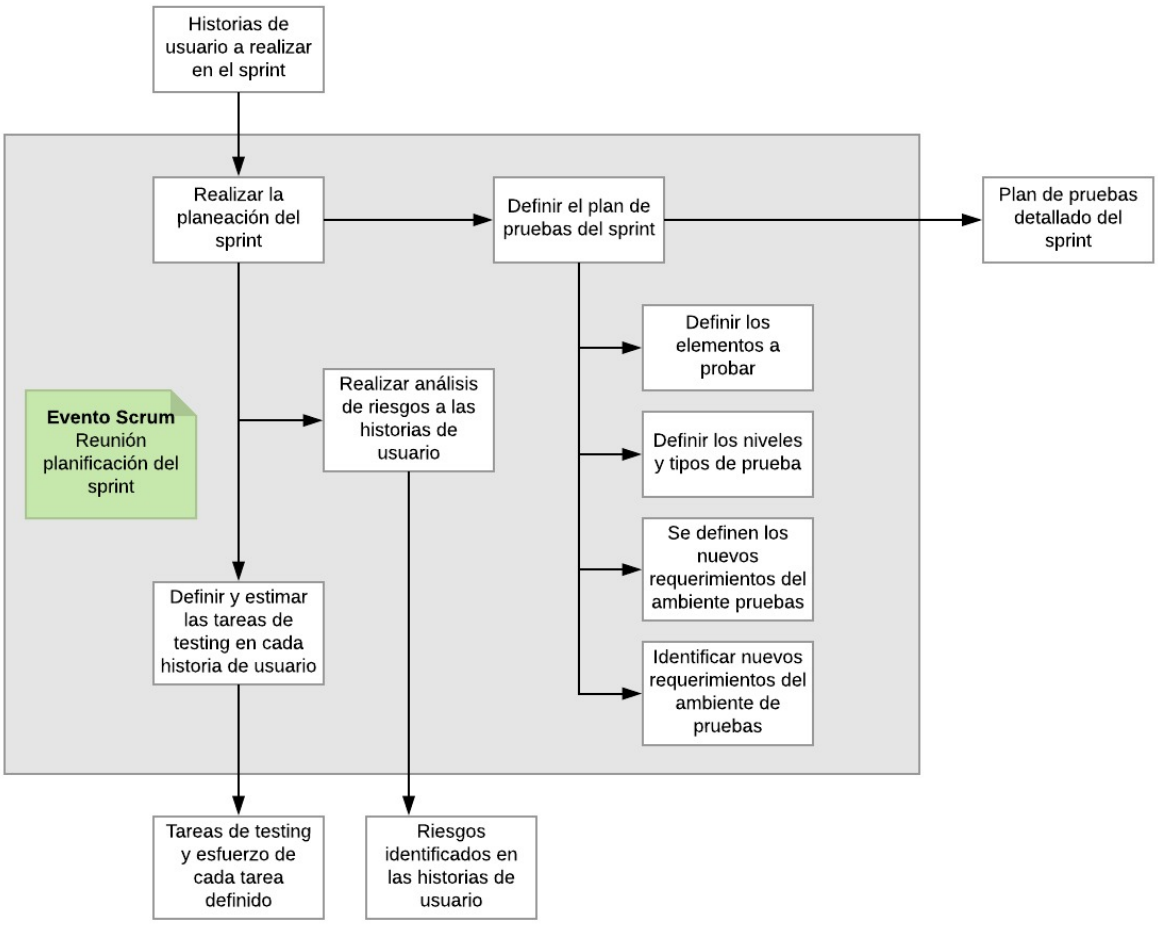


Figura 51: Procesos de calidad en la etapa de planificar y estimar

Fuente: Elaboración propia, 2018.

6.4.1 Realizar la planeación del sprint

Este evento de Scrum se compone de tres actividades principales, la primera corresponde a la definición del plan de pruebas del *sprint*, posteriormente se realiza un análisis de riesgos a las historias de usuario del *sprint* y por último se definen las tareas y la estimación de esfuerzo correspondiente al equipo de *testing*. Seguidamente se describe cada una de las actividades de *testing* que comprenden el proceso de planeación del sprint.

6.4.1.1 Definir el plan de pruebas del sprint

El objetivo de este proceso es poder verificar la calidad de los entregables de manera objetiva y planificada. Este plan de pruebas, a diferencia del realizado en la etapa de iniciación, es un poco más detallado ya que comprende la planeación de un sprint. En el Anexo 10 se muestra la plantilla de plan de pruebas del *sprint* propuesta para el área de calidad, dicha plantilla se definió con los siguientes elementos:

a) Proporcionar una visión general del plan mediante los siguientes datos:
    a. Definir el *sprint* al que corresponde el plan de pruebas.
    b. Fecha de elaboración del plan.
    c. Dar una descripción del sistema.
b) Describir niveles y tipos de pruebas.
    a. Especificar los niveles de pruebas que se realizarán.
    b. Definir los tipos de pruebas que se ejecutarán.
c) Definir el alcance de la prueba, especificando lo que se va probar.
    a. Indicar los módulos o entregables.
d) Especificar las nuevas características o requerimientos que el ambiente de pruebas necesita para el *sprint.*

a. Definir requerimientos de software, hardware o propiamente de espacio físico.

6.4.1.2 Realizar análisis de riesgos

El realizar un análisis de riesgos ayuda a priorizar los elementos que deben probarse y permite que el equipo de calidad pueda determinar lo que se debe probar primero y qué elementos requieren una mayor atención durante el proceso de control de calidad. Para determinar el riesgo de cada ítem primero se determina el impacto que puede generar a la institución y luego se estima la probabilidad en que puede darse, para finalmente multiplicar el impacto por la probabilidad y tener como resultado el riesgo total. Una vez teniendo lo anterior, hará que la estimación de las tareas sea más fácil de realizar y más realista. Para la elaboración del análisis de riesgos se propone la plantilla que se encuentra en el Anexo 11.

6.4.1.3 Definir y estimar las tareas

Este proceso comprende la definición de las tareas para cada historia de usuario a probar en el *sprint*. Estas tareas se registrarán con la herramienta de *Microsoft Visual Studio Team Services,* la cual es la oficial para el registro de dichas tareas en el área. En el Anexo 12 se muestra la plantilla de agregar nueva tarea, la cual comprende elementos como:

a) Título: Ingresar el título de la tarea.
b) Asignar integrante: Se asigna el integrante del equipo que realizará la tarea.
c) Estado: El estado por defecto cuando se crea una tarea es “Por realizar”.
d) Iteración: En caso de estar sobre un sprint en específico se carga el mismo, caso contrario, se le debe seleccionar el *sprint* donde se completará dicha tarea.
e) Descripción: Se ingresa la descripción o el detalle de la tarea.
f) Prioridad: Debido a que habrán tareas con más prioridad que otras, la plantilla permite asignar dicha prioridad.

g) Tiempo: Se establece el tiempo en horas que se considera necesario para realizar la tarea.

h) Actividad: Para esta metodología se asigna la actividad llamada *testing*.

i) Trabajo relacionado: Se pueden agregar otras tareas relacionadas.

Se propone la creación de tres tareas de *testing* para cada historia de usuario. La primera es el análisis de la historia de usuario, esta tarea corresponde a la revisión de la documentación, arquitectura, diseño, etc., relacionada con la historia de usuario. La segunda tarea corresponde al diseño de las pruebas, una vez que el integrante de calidad inicie esta tarea, el mismo procederá a diseñar las pruebas basadas en el análisis realizado en la tarea anterior. Por último está la tarea de ejecución de las pruebas, que comprende la ejecución, el reporte de defectos y la documentación de las pruebas realizadas.

También se propone la realización de dos tareas como parte del cierre del proceso de pruebas en el *sprint*. La primera tarea corresponde a la realización de pruebas de regresión y la segunda tarea corresponde a la realización del reporte de resultados de las pruebas del *sprint*.

6.5 Análisis y diseño

El integrante de calidad inicialmente realiza un análisis de documentación de los requerimientos, del diseño, de la arquitectura y de los criterios de aceptación, entre otros. Lo anterior le permitirá identificar los escenarios de prueba que se requieren diseñar a través de los casos de prueba, así como los datos de prueba necesarios. Es importante mencionar que el equipo de calidad debe escribir las pruebas de acuerdo con las especificaciones o requerimientos, y no con lo que alguien dijo extraoficialmente. El diseño de los casos de prueba se puede realizar por módulos y por los tipos de pruebas siguiendo lo establecido en el plan de pruebas.

Por otra parte, los miembros del equipo de calidad deben formar parte de la reunión de Scrum diario, donde cada integrante expone al resto del equipo lo que se hizo el día anterior, lo que se va realizar en el día actual y si tiene impedimentos. A continuación en la Figura 52 se muestra de manera general los procesos involucrados en esta etapa.

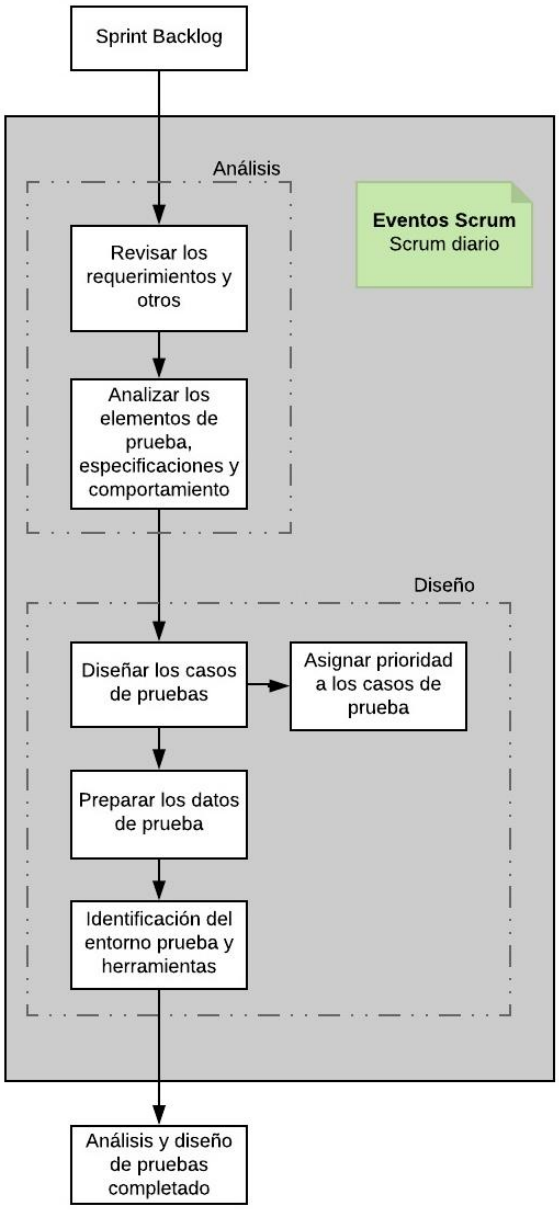


Figura 52: Procesos de calidad en la etapa análisis y diseño

Fuente: Elaboración propia, 2018.

Se debe registrar cada caso de prueba en una herramienta de administración de casos de prueba, lo anterior para automatizar ciertas tareas, reutilizar los casos de prueba para futuras versiones del producto y por último para llevar un seguimiento. Para esto se propone hacer uso de la herramienta con la que cuenta actualmente el área, la cual es *Microsoft Visual Studio Team Services*, esta herramienta cuenta con un apartado de pruebas donde se pueden crear los casos de prueba, correrlos y reutilizarlos en futuros *sprints*. En el Anexo 13 se muestra la plantilla de nuevo caso de prueba, la cual comprende elementos como:

a) Título: Ingresar el título del caso de prueba.
b) Asignar *tester*: Se asigna el *tester* que ejecutará el caso de prueba.
c) Estado: El estado por defecto cuando se crea un caso de prueba es “Diseño”.
d) Iteración: En caso de estar sobre un *sprint* en específico se carga el mismo, caso contrario, se le debe seleccionar el *sprint* donde se completará.
e) Prioridad: Debido a que habrán casos de prueba con más prioridad que otros, la plantilla permite asignar dicha prioridad.
f) Tiempo: Se establece el tiempo en horas que se considera necesario para realizar la tarea.
g) Trabajo relacionado: Se pueden agregar otras tareas relacionadas.
h) Pasos: Se agregan los pasos que comprende el caso de prueba así como los resultados esperados en cada paso. En cada paso se puede adjuntar archivos si fuese necesario.

Adicionalmente se deben preparar los datos de prueba así como el entorno donde se realizarán las pruebas, y evaluar si se requiere alguna herramienta adicional para la ejecución de las pruebas.

6.6 Ejecución

Esta etapa corresponde al proceso de ejecución de las pruebas, ya sea a través de casos de prueba previamente diseñados en la etapa anterior o pruebas no basadas en casos de prueba, como lo son las pruebas exploratorias, pruebas de humo u otras. El equipo de calidad durante esta etapa debe formar parte de la reunión diaria de Scrum, donde cada integrante expone al resto del equipo lo que se hizo el día anterior, lo que se va realizar en el día actual y si tiene impedimentos. A continuación en la Figura 53 se muestra de manera general los procesos involucrados en esta etapa.

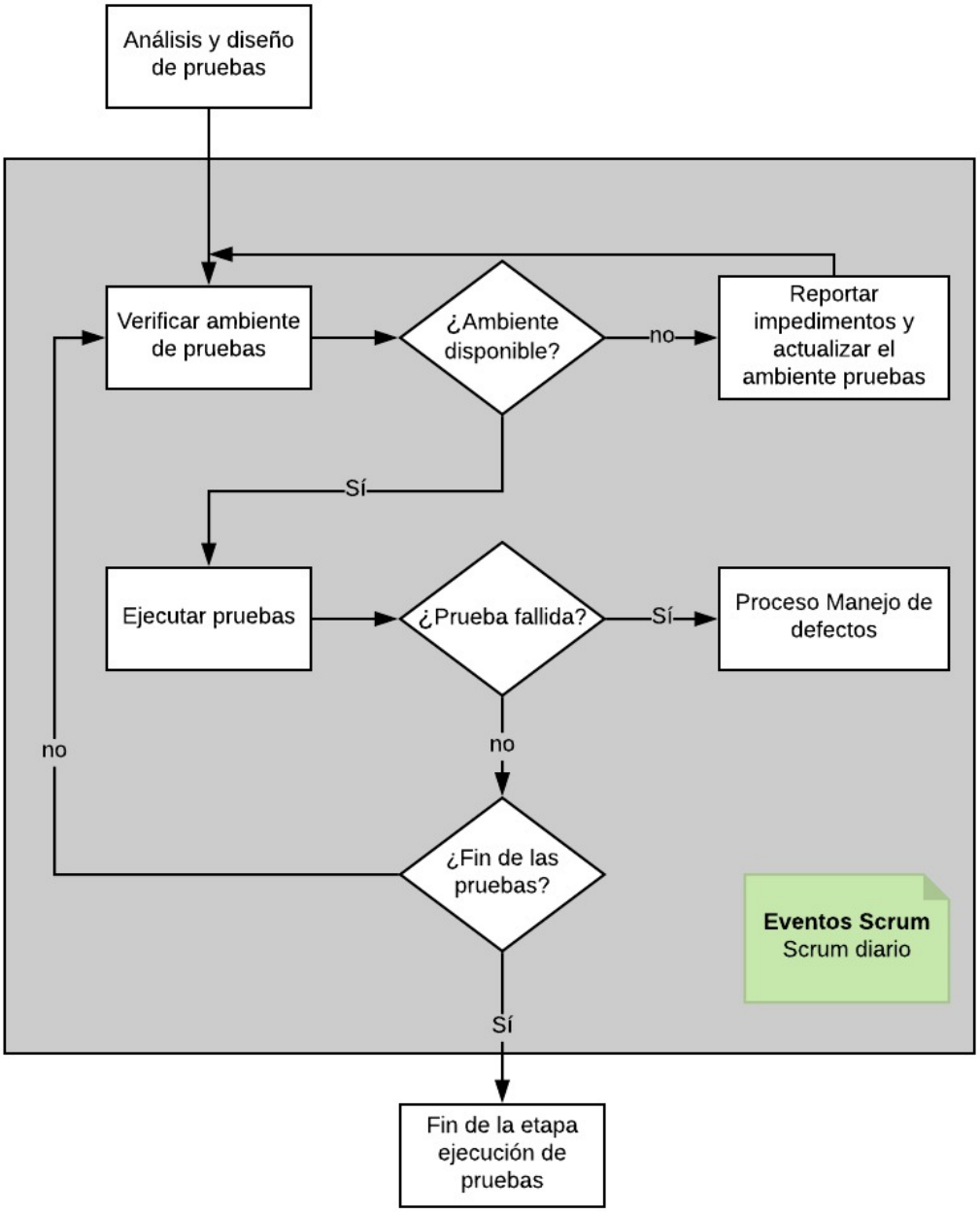


Figura 53: Procesos de calidad de la etapa ejecución

Fuente: Elaboración propia, 2018.

6.6.1 Verificar el ambiente de pruebas

El equipo de calidad debe validar el ambiente de pruebas y la existencia de los datos en la base de datos antes de iniciar con la ejecución de las pruebas.

6.6.2 Ejecutar las pruebas

Una vez que se tiene disponible lo anterior, se comienza la ejecución de las pruebas, para esto se recomienda inicialmente realizar pruebas de humo cuyo objetivo es el de dar una revisión rápida de las funcionalidades principales del producto para comprobar que están funcionando y que no se presentan fallas. En caso de darse algún problema, no se continúan con las pruebas hasta que sean corregidos los fallos.

Si el plan de pruebas del *sprint* comprende la ejecución de pruebas en todos los niveles, se deben ejecutar dichas pruebas orden respetando los niveles de las pruebas. Dependiendo de las pruebas que se vayan a realizar, puede ser necesario el uso de alguna herramienta para la ejecución de la pruebas, caso contrario solo se aplican pruebas manuales. Una vez ejecutada la prueba o caso de prueba, se verifica su resultado, si pasó o falló, en caso de fallar se debe crear el defecto o error que se obtuvo en alguna herramienta administradora de errores con el fin de automatizar y facilitar las tareas de *testing*, lo cual permite que se le pueda dar seguimiento al reporte y corrección de los mismos. Para la verificación de las pruebas se ha creado una lista de verificación con el objetivo de mantener un estándar de elementos básicos que deben revisarse.

El integrante de calidad evalúa si el criterio de aceptación se cumple y determina si es necesario efectuar más pruebas.

6.6.2.1 Proceso de manejo de defectos

Este punto describe el flujo del proceso del manejo de defectos durante la etapa de ejecución. En la siguiente figura se presenta el flujo del proceso de manejo de defectos reportados por el equipo de calidad.

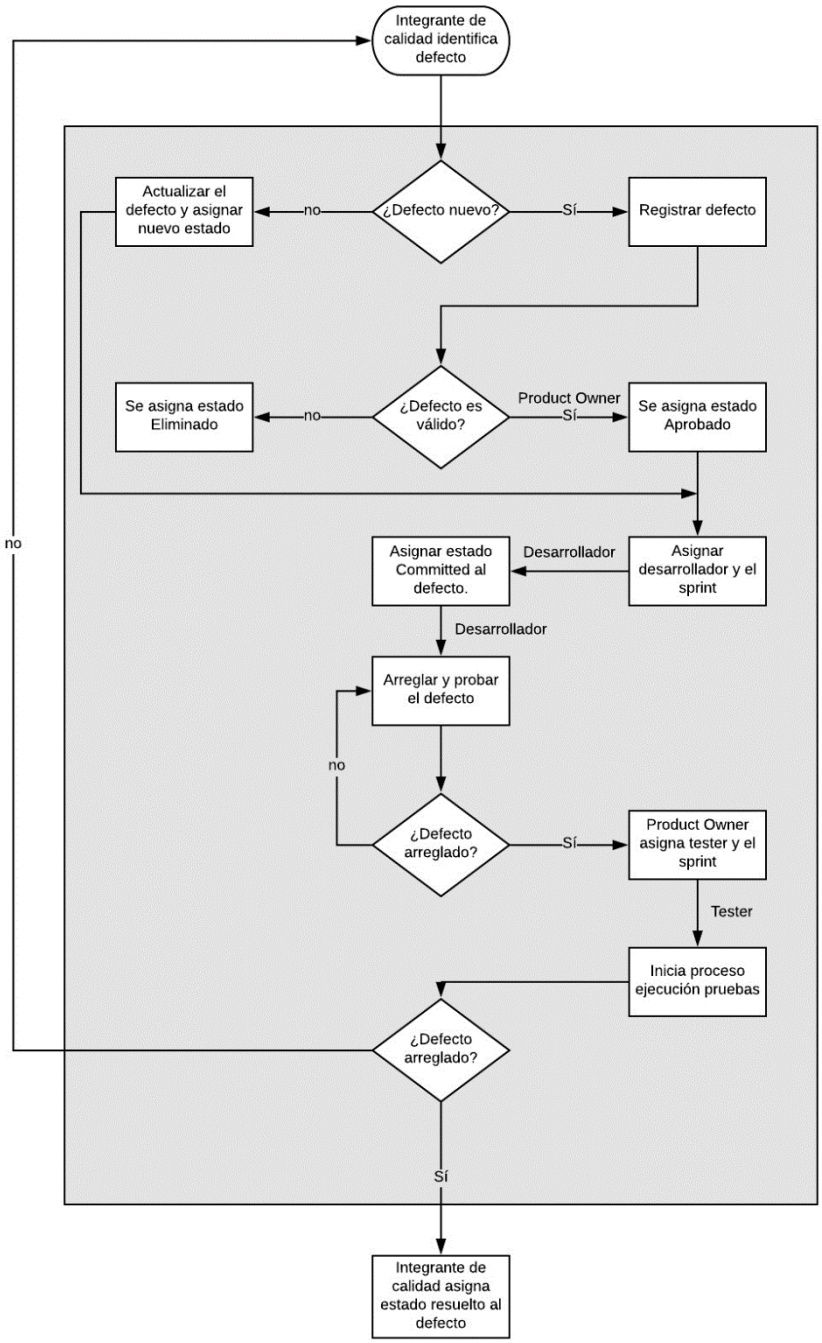


Figura 54: Procesos del manejo de defectos

Fuente: Elaboración propia, 2018.

Para el registro o actualización de un *bug* se utilizará la plantilla de la herramienta *Microsoft Visual Studio Team Services* (ver Anexo 14: Plantilla creación de nuevo defecto (bug)). Dicha plantilla comprende elementos como:

a) Título: Ingresar el título del *bug*.

b) Asignar responsable: Se asigna el responsable al *bug*.

c) Estado: El estado por defecto cuando se crea un nuevo *bug* es "Nuevo".

   Una vez que el *product owner* ha identificado el defecto y considera que si procede, este lo asigna a un desarrollador para la investigación y corrección.

   Cuando el desarrollar finaliza la corrección asigna el estado "commited", y es el *product owner* quien asigna nuevamente el defecto a un integrante de calidad para la revisión. Si el reporte está corregido el integrante de calidad actualiza el estado a terminado.

d) Iteración: En caso de estar sobre un *sprint* en específico se carga el mismo, caso contrario, se le debe seleccionar el *sprint* al que pertenece el *bug*.

e) Repro steps: Sección donde se especifican los pasos y los resultados esperados, se marcan los pasos que fallaron.

f) Información del sistema: Se detallan las características de los sistemas operativos, navegadores web, arquitectura, etc., donde se realizaron las pruebas.

g) Criterio de aceptación: Se describe el criterio de aceptación para el bug registrado.

h) Prioridad: Es un atributo que determina la prioridad con la que el defecto debe corregirse. El valor 1 es de mayor prioridad, es decir, es un defecto que provoca que el producto no pueda entregarse o ponerse en producción.

i) Tiempo: En este caso el tiempo no se asigna, ya que esto lo determina el *product owner* o bien el equipo de desarrollo encargado de resolver el bug.

j) Trabajo relacionado: Se pueden agregar otros elementos relacionados al bug, por ejemplo otros *bugs*, tareas, *backlog items*, etc.

k) Adjuntar archivos: Se pueden adjuntar archivos al *bug*, que permitan evidenciar o mostrar las pantallas donde se genera el error. Esto es de gran ayuda para el desarrollador y se le facilita la reproducción del error, así como la corrección del mismo.

Una vez que un reporte de error o defecto ha sido corregido, el equipo de calidad debe realizar pruebas de confirmación para determinar que el reporte ha sido solucionado, y posteriormente debe realizar pruebas de regresión para validar que el cambio realizado no haya afectado la funcionalidad existente del sistema.

6.6.2.2 Tipos de pruebas a realizar y aspectos generales de evaluación

El equipo de calidad realizará las pruebas de acuerdo con el orden de los niveles de pruebas que se especificaron en la sección 2.2.13. Dentro de cada nivel el integrante de calidad establece los tipos de pruebas a realizar, los cuales se mencionaron en el apartado 2.2.15. Las pruebas de regresión deben realizarse en cada cambio en el software y al final de cada *sprint*, para asegurar la estabilidad de la aplicación antes de que sea puesta en producción.

Se realizó una lista de chequeo para cada tipo de prueba con el objetivo de que el equipo de calidad cuente con un estándar de aspectos básicos a evaluar y que a la vez quede evidencia de que se ha revisado cada elemento indicado en la lista. Estas listas de chequeo se pueden encontrar en el Anexo 15: Listas de chequeo para los diferentes tipos de pruebas.

Para la realización de las pruebas de rendimiento, carga y estrés se propone el siguiente proceso para que sea aplicado por el equipo de calidad.

6.6.2.3 Pruebas de rendimiento, carga y estrés

El proceso para ejecutar este tipo de pruebas de una mejor manera es el siguiente:

a. Identificar el entorno de pruebas

Es importante que el equipo de calidad se familiarice o conozca los ambientes físicos de producción y de pruebas y conozca aspectos como el hardware, software, configuraciones de red del entorno. Esto ayuda a obtener un plan de pruebas efectivo e identificar desafíos de las pruebas al comienzo.

b. Identificar el criterio de aceptación

Identificar las metas en cuanto a: tiempo de respuesta, rendimiento y la utilización de recursos. En esta fase, del criterio de aceptación podría lograrse simplemente observando las características de desempeño que los usuarios comparten con un buen desempeño.

c. Planear y diseñar las pruebas

En esta fase se planean los escenarios clave, se definen los datos de prueba y se establecen las métricas que se recolectarán.

d. Configurar el entorno de pruebas

En esta fase el equipo de calidad debe preparar el entorno de pruebas, las herramientas que se van a utilizar, ya sea Visual Studio o la aplicación JMETER, y demás recursos necesarios para ejecutar la prueba.

e. Ejecución de las pruebas

En esta fase el equipo de calidad realiza las pruebas de rendimiento de acuerdo con el diseño de prueba, además de ejecutar las pruebas debe monitorear las pruebas, validar dichas pruebas y verificar los resultados.

f. Analizar los resultados y reportar

Se analizan los resultados obtenidos y se reportan los resultados a los interesados.

6.7 Reporte de resultados

Una vez completadas las pruebas, se debe generar un reporte con los resultados de las pruebas del *sprint*. Para esto se propone que la generación de los resultados se realice de manera ágil mediante la herramienta de *Microsoft Visual Studio Team Services*.

Con la herramienta *Visual Studio Team Services* se pueden crear consultas y a partir de estas generar gráficos que muestren datos de interés, por ejemplo:

- Defectos: Defectos creados, defectos corregidos/no corregidos.
- Casos de prueba: Casos de prueba ejecutados, casos de prueba fallidos/no fallidos.

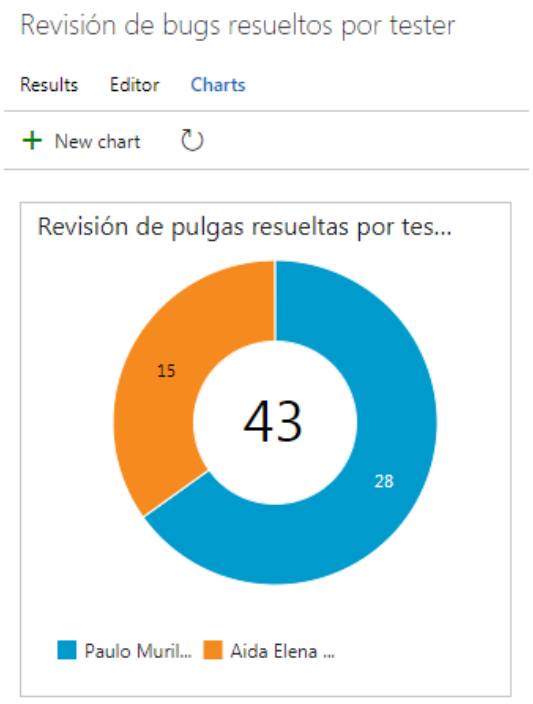


Figura 55: Reporte cantidad defectos resueltos en un sprint

Fuente: Elaboración propia mediante herramienta VSTS, 2018.

Adicionalmente se propone una plantilla que detalla las pruebas realizadas durante el *sprint*, así como los principales aspectos evaluados y los defectos que se registraron en el *sprint* (ver Anexo 16: Plantilla reporte de pruebas realizadas de un sprint). Una vez finalizada esta etapa, el integrante de calidad debe finalizar la tarea que corresponde a la realización de los reportes del *sprint* y adjuntar la plantilla a dicha tarea.

6.8 Revisión y retrospectiva

Esta fase comprende los procesos orientados a reflexionar sobre los procedimientos realizados y el incremento o producto generado en el *sprint* que está finalizando, lo anterior con el objetivo de proponer mejoras y detectar defectos.

6.8.1 Revisión del sprint

El equipo de calidad debe formar parte la reunión de revisión del *sprint*, mostrar al *product owner* y a las partes interesadas lo realizado durante el *sprint* para recibir retroalimentación sobre la calidad del producto, que luego se pueden convertir en historias de usuario nuevas.

6.8.2 Retrospectiva del sprint

El equipo de calidad se reúne con el resto del equipo Scrum para discutir las lecciones aprendidas desde la etapa de planificación hasta el cierre de las actividades de *testing* para el *sprint* correspondiente y la retroalimentación recibida en la reunión de revisión del *sprint*. Se analizan las métricas definidas por el equipo de calidad para evaluar el desempeño en el *sprint*. También se presentará propuestas de mejora al proceso de control de calidad. Finalmente se genera una lista de mejoras a los procesos, que son revisados para posteriormente seleccionar las propuestas de mejora que se desean implementar. Para este proceso se propone una plantilla que

permite registrar los aspectos que el equipo considera salieron bien o que deben mejorarse (ver Anexo 18: Plantilla retrospectiva del sprint).

CAPÍTULO VII

7. Implementación plan piloto de la metodología

Una vez que se ha definido la metodología de control de calidad, se requiere implementar como plan piloto dicha metodología en el Área de Informática de Gestión del Poder Judicial, tomando en consideración la cultura organizacional y la situación actual del área. Para esto se propone las siguientes etapas de implementación:

a. Autorización y selección del *sprint* para la implementación de la metodología.
b. Capacitación al equipo del proyecto, principalmente al equipo de calidad.
c. Implementación de la metodología.
d. Análisis de resultados y mejora de la metodología.
e. Cierre de implementación de la metodología.

En las siguientes secciones se explicará a detalle cada una de las etapas que forma parte de la implementación de la metodología de control de calidad en el área de informática de gestión.

7.1 Autorización y selección del sprint para la implementación de la metodología

En esta etapa, el *product owner* del proyecto SIAGPJ autoriza la implementación y establece el *sprint* para la implementación. Se asigna el encargado de implementar la metodología. Adicionalmente, en esta etapa, se asigna el tiempo que dedicará el integrante de calidad para llevar a cabo cada paso de la metodología propuesta.

7.2 Capacitación al equipo del proyecto

Una vez que se autoriza y se establece el *sprint* de implementación, así como el encargado de implementar y capacitar el equipo, se procede a realizar la capacitación al equipo Scrum de la metodología de control de calidad propuesta. La capacitación se enfoca en los siguientes temas:

- Scrum (1 hora)
- Pruebas ágiles (1 hora)
- Metodología de control de calidad propuesta en esta investigación (2 horas)

El principal objetivo de esta etapa es preparar al equipo del proyecto, principalmente al de calidad, para que apliquen lo mejor posible la metodología propuesta. Como parte del entregable en esta etapa corresponde el visto bueno del implementador indicando que el equipo del SIAGPJ ha sido capacitado satisfactoriamente.

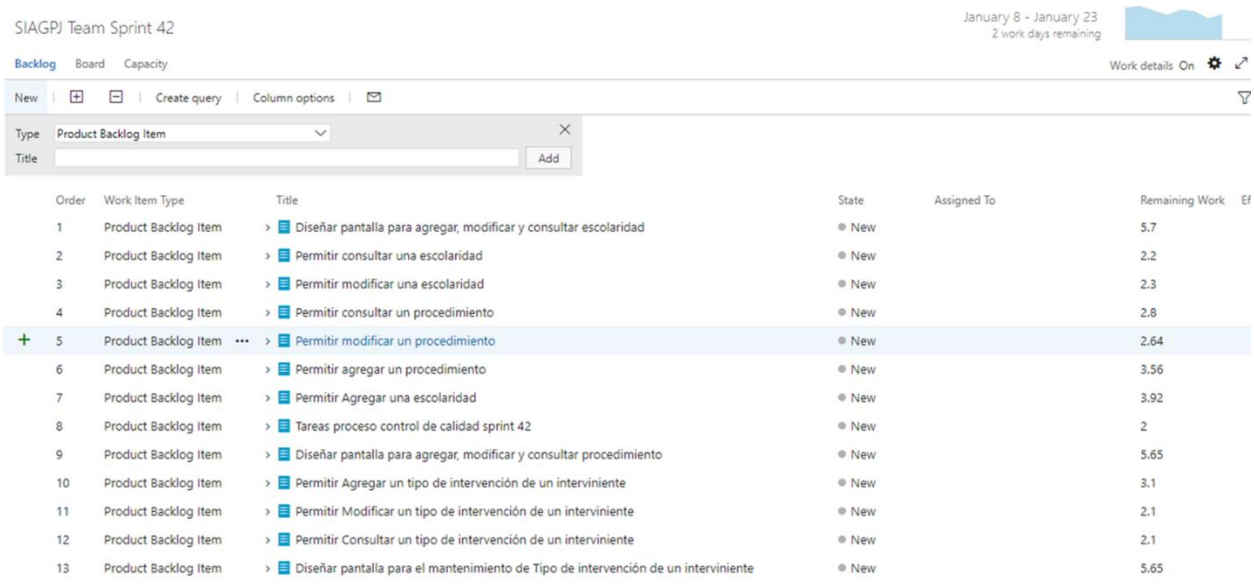
7.3 Implementación de la metodología

Una vez que el equipo de calidad está capacitado en la metodología de control de calidad, se procede con la implementación de la metodología para el *sprint* establecido en la primera fase. Para la implementación se requerirán de historias de usuario para realizar los procesos de planificar, ejecutar y reportar. Se requerirá del apoyo de los desarrolladores y del equipo de calidad para poder ejecutar las pruebas establecidas y los métodos ágiles de *testing*.

Se estableció que la metodología de control de calidad propuesta se implementará en el *sprint* 42 con fecha de inicio el 8 de enero del 2018 y finalizando el *sprint* el 23 de enero del 2018, se tiene como días hábiles de trabajo un total de 12 días; lo anterior debido a que la última semana de diciembre y la primera de enero no se laboró en el área de calidad. Por otra parte, se decidió por parte del *product owner* que el integrante de calidad dedicaría una hora diaria al proyecto

utilizando la metodología propuesta en esta investigación, lo que tiene como resultado un total de 12 horas asignadas.

Adicionalmente se seleccionaron las siguientes historias de usuario para que se les aplique el proceso de control de calidad bajo la metodología propuesta en esta investigación para el *sprint* 42.



Order	Work Item Type	Title	State	Assigned To	Remaining Work
1	Product Backlog Item	Diseñar pantalla para agregar, modificar y consultar escolaridad	New		5.7
2	Product Backlog Item	Permitir consultar una escolaridad	New		2.2
3	Product Backlog Item	Permitir modificar una escolaridad	New		2.3
4	Product Backlog Item	Permitir consultar un procedimiento	New		2.8
5	Product Backlog Item	Permitir modificar un procedimiento	New		2.64
6	Product Backlog Item	Permitir agregar un procedimiento	New		3.56
7	Product Backlog Item	Permitir Agregar una escolaridad	New		3.92
8	Product Backlog Item	Tareas proceso control de calidad sprint 42	New		2
9	Product Backlog Item	Diseñar pantalla para agregar, modificar y consultar procedimiento	New		5.65
10	Product Backlog Item	Permitir Agregar un tipo de intervención de un interviniente	New		3.1
11	Product Backlog Item	Permitir Modificar un tipo de intervención de un interviniente	New		2.1
12	Product Backlog Item	Permitir Consultar un tipo de intervención de un interviniente	New		2.1
13	Product Backlog Item	Diseñar pantalla para el mantenimiento de Tipo de intervención de un interviniente	New		5.65

Figura 56: Lista de historias de usuario a probar en el sprint 42

Fuente: https://siagpj.visualstudio.com.

Seguidamente se detalla el proceso de implementación de la metodología.

7.3.1 Planificar y estimar

Como bien se menciona en la metodología, esta etapa comprende inicialmente la realización de un plan de pruebas específico del *sprint*. Para proporcionar un estándar y un documento ágil de llenar, se propuso la **¡Error! No se encuentra el origen de la referencia.**, la cual se muestra a continuación ya elaborada.

Tabla 22: Plan de pruebas sprint 42

Plan de Pruebas del Sprint 42

Identificador:	**Fecha de elaboración:**
PPS41	05/12/2017

Nombre del proyecto:

SIAGPJ

Descripción:

El plan de pruebas corresponde a una guía de lo que se probará en el sprint y qué tipos de pruebas se aplicarán.

Niveles de pruebas: <Indicar los niveles de pruebas>

☐Pruebas unitarias	☒Pruebas de sistema
☒Pruebas de integración	☐Pruebas de aceptación

Tipos de pruebas:

☒Pruebas funcionales	☒Pruebas de usabilidad
☐Pruebas de rendimiento	☐Pruebas de accesibilidad
☐Pruebas de carga	☐Pruebas de seguridad
☐Pruebas de estrés	☒Pruebas de interfaz
☐Pruebas de compatibilidad	☒Pruebas de documentación
☐Pruebas de portabilidad	☐Pruebas de confirmación
☒Pruebas de regresión	

Alcance (Elementos a probar):

<Descripción de los elementos o módulos del sistema a probar>

Entregable	**Descripción**
Mantenimiento de catálogos: Escolaridad, Procedimiento y Tipo de Intervención.	Las funcionalidades a probar corresponden a: Agregar, modificar y buscar. También el diseño de las pantallas.

Ambiente de pruebas:

Se requiere la actualización de los servicios web API.

Ejecución de nuevos procedimientos almacenados.

Instalación del nuevo *release* en la máquina a probar.		
Aprobado por:	Paulo Murillo Esquivel	
Fecha aprobación:	05/01/2018	

Fuente: Elaboración propia, 2018.

Una vez realizado el plan de pruebas, se procede a realizar un breve análisis de riesgos de las historias de usuario que se van a probar en el *sprint*.

Tabla 23: Evaluación de riesgos historias de usuario sprint 42

Evaluación de Riesgos Historias de Usuario Sprint 42 Informática de Gestión – Área de Calidad y Testing					
Id Historia Usuario	**Descripción historia de usuario**	**Descripción Riesgo**	**Impacto**	**Probabilidad**	**Riesgo**
180	Diseñar pantalla para agregar, modificar y consultar escolaridad	Pantalla con falta de descripciones emergentes, alineación de los objetos desordenada.	1	1	1
176	Permitir consultar una escolaridad	Al consultar por una descripción existente, muestra cero resultados.	2	1	2
172	Permitir modificar una escolaridad	Al modificar una escolaridad con el valor máximo de caracteres permitidos en el campo, que la descripción modificada se corte.	3	2	6
166	Permitir Agregar una escolaridad	Al agregar una escolaridad el sistema está permitiendo ingresar registros en blanco.	2	1	3
201	Diseñar pantalla para agregar, modificar y consultar un procedimiento	Pantalla con falta de descripciones emergentes, alineación de los objetos desordenada.	1	1	1
196	Permitir consultar un procedimiento	Al consultar por código un procedimiento, muestra otros códigos de procedimiento.	2	2	4
192	Permitir modificar un procedimiento	Al modificar un procedimiento, el mismo no se está modificando.	3	3	9
186	Permitir agregar un procedimiento	Al agregar un procedimiento el sistema no registra la fecha de creación.	3	2	6
234	Permitir Agregar un tipo de intervención de un interviniente	Que el sistema permita registrar tipos de intervención Nulos.	2	1	2

Evaluación de Riesgos Historias de Usuario Sprint 42 Informática de Gestión – Área de Calidad y Testing					
Id Historia Usuario	**Descripción historia de usuario**	**Descripción Riesgo**	**Impacto**	**Probabilidad**	**Riesgo**
244	Permitir Modificar un tipo de intervención de un interviniente	Que el sistema no registre la fecha de modificación en la base de datos.	1	3	3
248	Permitir Consultar un tipo de intervención de un interviniente	Que al consultar un tipo de intervención por el código existente, el sistema muestre cero resultados.	2	1	2
252	Diseñar pantalla para el mantenimiento de Tipo de intervención de un interviniente	Pantalla con falta de descripciones emergentes, alineación de los objetos desordenada.	1	1	1

Valores de Impacto y Probabilidad

Impacto	Descripción		Probabilidad	Descripción
1	Bajo impacto		1	Baja probabilidad
2	Medio Impacto		2	Media probabilidad
3	Alto impacto		3	Alta probabilidad

Fecha de elaboración:	05/01/2018			
Elaborado por:	Aida Elena Siles Rojas			

Fuente: Elaboración propia, 2018.

Una vez realizado el análisis de riesgos de las historias de usuario, se procede a definir y estimar las tareas de *testing* en cada historia de usuario, así como las tareas generales del *sprint*.

a) Tareas registradas en cada historia de usuario:
    a. Análisis de la historia de usuario
    b. Diseño de las pruebas
    c. Ejecución de las pruebas
b) Creación de un *product backlog ítem* llamado: Tareas proceso control de calidad *sprint* 42. Dentro de este *backlog* se crearon dos tareas principales:
    a. Realizar pruebas de regresión

b. Realizar reporte de resultados

Después de haber definido las tareas necesarias para llevar a cabo el proceso de control de calidad en el *sprint* 42, se realizó la estimación de cada tarea, tomando en cuenta el análisis de riesgos realizado.

7.3.1.1 Definición y estimación de tareas del catálogo escolaridad

A continuación en las siguientes figuras se muestran las tareas y los tiempos que se estimaron para realizar el proceso de control de calidad, esto para las historias de usuario que corresponden al catálogo de escolaridad del *sprint* 42 en el proyecto SIAGPJ.

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Diseñar pantalla para agregar, modificar y consultar escolaridad	New		5.7	
Task	Diseñar la pantalla principal para el catalogo escolaridad (menú y consulta)	To Do	Alejandro Villalta	1.5	Development
Task	Diseñar la pantalla para agregar y modificar escolaridad	To Do	Alejandro Villalta	1.5	Development
Task	Llamar las operaciones insertar de los servicios e integrar con la pantalla	To Do	Alejandro Villalta	0.5	Development
Task	Llamar las operaciones modificar de los servicios e integrar con la pantalla	To Do	Alejandro Villalta	0.5	Development
Task	Llamar las operaciones listar de los servicios e integrar con la pantalla	To Do	Alejandro Villalta	1	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.25	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.25	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing

Figura 57: Tareas historia de usuario diseño pantalla escolaridad

Fuente: https://siagpj.visualstudio.com

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Permitir Agregar una escolaridad	New		3.92	
Task	Crear tabla y procedimiento almacenado para agregar escolaridad	To Do	Henry Mendez	0.5	Development
Task	Crear entidad escolaridad	To Do	Henry Mendez	0.5	Development
Task	Crear acceso a datos y método agregar escolaridad	To Do	Henry Mendez	0.5	Development
Task	Crear lógica de negocio y método agregar escolaridad a la lógica de negocio	To Do	Henry Mendez	0.5	Development
Task	Crear servicio y crear método agregar escolaridad	To Do	Henry Mendez	0.5	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.5	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.5	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.42	Testing

Figura 58: Tareas historia de usuario agregar una escolaridad

Fuente: https://siagpj.visualstudio.com

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Permitir consultar una escolaridad	New		2.2	
Task	Agregar método consultar escolaridad al acceso a datos y crear procedimiento almacenado	To Do	Henry Mendez	0.5	Development
Task	Agregar método consultar a la lógica de negocio	To Do	Henry Mendez	0.5	Development
Task	Agregar método Consultar al servicio	To Do	Henry Mendez	0.5	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.25	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.25	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing

Figura 59: Tareas historia de usuario consultar escolaridad

Fuente: https://siagpj.visualstudio.com

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Permitir modificar una escolaridad	New		2.3	
Task	Agregar método modificar escolaridad al acceso a datos y crear procedimiento almacenado	To Do	Henry Mendez	0.5	Development
Task	Agregar método modificar a la lógica de negocio	To Do	Henry Mendez	0.5	Development
Task	Agregar método modificar al servicio	To Do	Henry Mendez	0.5	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.3	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.3	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing

Figura 60: Tareas historia de usuario modificar escolaridad

Fuente: https://siagpj.visualstudio.com

7.3.1.2 Definición y estimación de tareas del catálogo procedimiento

A continuación en las siguientes figuras se muestran las tareas y los tiempos que se estimaron para realizar el proceso de control de calidad para las historias de usuario que corresponden al catálogo de procedimiento del *sprint* 42 en el proyecto SIAGPJ.

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Diseñar pantalla para agregar, modificar y consultar procedimiento	New		5.65	
Task	Diseñar la pantalla principal para el catalogo procedimiento (menú y consulta)	To Do	Uriel García R.	1.5	Development
Task	Diseñar la pantalla para agregar y modificar procedimiento	To Do	Uriel García R.	1.5	Development
Task	Llamar las operaciones insertar de los servicios e integrar con la pantalla	To Do	Uriel García R.	0.5	Development
Task	Llamar las operaciones modificar de los servicios e integrar con la pantalla	To Do	Uriel García R.	0.5	Development
Task	Llamar las operaciones listar de los servicios e integrar con la pantalla	To Do	Uriel García R.	1	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.25	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.25	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.15	Testing

Figura 61: Tareas historia de usuario diseño pantalla procedimiento

Fuente: https://siagpj.visualstudio.com

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Permitir agregar un procedimiento	New		3.56	
Task	Crear tabla Procedimiento	To Do	Uriel García R.	0.3	Development
Task	Crear acceso a datos	To Do	Uriel García R.	0.5	Development
Task	Crear la entidad procedimiento	To Do	Uriel García R.	0.5	Development
Task	Crear la lógica de negocio	To Do	Uriel García R.	0.5	Development
Task	Crear servicio para agregar procedimientos	To Do	Uriel García R.	0.5	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.42	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.42	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.42	Testing

Figura 62: Tareas historia de usuario agregar procedimiento

Fuente: https://siagpj.visualstudio.com

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Permitir consultar un procedimiento	New		2.8	
Task	Agregar el método de consultar en el acceso a datos	To Do	Pablo Alvarez	0.5	Development
Task	Agregar el método de consultar a la lógica de negocio	To Do	Pablo Alvarez	0.5	Development
Task	Agregar operación en el servicio de consultar procedimiento	To Do	Pablo Alvarez	0.5	Development
Task	Modificar el acceso a datos, negocio y servicio del método consultar procedimientos para ut...	To Do	Alejandro Villalta	0.5	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.3	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.3	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing

Figura 63: Tareas historia de usuario consultar procedimiento

Fuente: https://siagpj.visualstudio.com

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Permitir modificar un procedimiento	New		2.64	
Task	Agregar el método modificar al acceso a datos	To Do	Uriel García R.	0.5	Development
Task	Agregar el método modificar a la lógica de negocio	To Do	Uriel García R.	0.5	Development
Task	Agregar la operación modificar en el servicio	To Do	Uriel García R.	0.5	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.42	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.42	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.3	Testing

Figura 64: Tareas historia de usuario modificar procedimiento

Fuente: https://siagpj.visualstudio.com

7.3.1.3 Definición y estimación de tareas del catálogo tipo de intervención

A continuación en las siguientes figuras se muestran las tareas y los tiempos que se estimaron para realizar el proceso de control de calidad para las historias de usuario que corresponden al catálogo de tipo de intervención del *sprint* 42 en el proyecto SIAGPJ.

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Diseñar pantalla para el mantenimiento de Tipo de intervención de un interviniente	New		5.65	
Task	Diseñar la pantalla principal para el catalogo tipo de intervención (menú y consulta)	To Do	Johan Acosta	1.5	Development
Task	Diseñar la pantalla para agregar y modificar un tipo de intervención	To Do	Johan Acosta	1.5	Development
Task	Llamar las operaciones insertar de los servicios e integrar con la pantalla	To Do	Johan Acosta	0.5	Development
Task	Llamar las operaciones modificar de los servicios e integrar con la pantalla	To Do	Johan Acosta	0.5	Development
Task	Llamar las operaciones listar de los servicios e integrar con la pantalla	To Do	Johan Acosta	1	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.25	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.25	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.15	Testing

Figura 65: Tareas historia de usuario diseño pantalla tipo de intervención

Fuente: https://siagpj.visualstudio.com

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Permitir Agregar un tipo de intervención de un interviniente	New		3.1	
Task	Crear tabla de tipo intervención y procedimiento almacenado de agregar	To Do	Johan Acosta	0.5	Development
Task	Crear entidad de tipo intervención	To Do	Johan Acosta	0.5	Development
Task	Crear acceso a datos de tipo de intervención	To Do	Johan Acosta	0.5	Development
Task	Crear lógica y de negocio de tipo de tipo intervención	To Do	Johan Acosta	0.5	Development
Task	Crear servicio y la operación de tipo de intervención	To Do	Johan Acosta	0.5	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.2	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing

Figura 66: Tareas historia de usuario agregar tipo de intervención

Fuente: https://siagpj.visualstudio.com

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Permitir Consultar un tipo de intervención de un interviniente	New		2.1	
Task	Crear metodo en el acceso a datos y procedimiento almacenado	To Do	Johan Acosta	0.5	Development
Task	Crear método de lógica de negocio para consultar	To Do	Johan Acosta	0.5	Development
Task	Agregar operación de consultar al servicio	To Do	Johan Acosta	0.5	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.2	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing

Figura 67: Tareas historia de usuario consultar tipo de intervención

Fuente: https://siagpj.visualstudio.com

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Permitir Modificar un tipo de intervención de un interviniente	New		2.1	
Task	Crear metodo acceso de datos y procedimiento almacenado modificar	To Do	Johan Acosta	0.5	Development
Task	Crear método en lógica de negocio	To Do	Johan Acosta	0.5	Development
Task	Agregar operación de modificar al servicio	To Do	Johan Acosta	0.5	Development
Task	Análisis de la historia de usuario	To Do	Aida Elena Siles Rojas	0.2	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing

Figura 68: Tareas historia de usuario modificar tipo de intervención

Fuente: https://siagpj.visualstudio.com

7.3.2 Análisis y diseño

Inicia el *sprint* 42, y con esto el equipo de calidad inicia la etapa de análisis y diseño estableciendo la tarea correspondiente “en progreso”, como se muestra en la Figura 69. Una vez terminado el análisis, se procede a terminar la tarea cambiándole el estado a “Terminado”.

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Diseñar pantalla para agregar, modificar y consultar escolaridad	New		0.7	
Task	Diseñar la pantalla principal para el catalogo escolaridad (menú y consulta)	Done	Alejandro Villalta		Development
Task	Diseñar la pantalla para agregar y modificar escolaridad	Done	Alejandro Villalta		Development
Task	Llamar las operaciones insertar de los servicios e integrar con la pantalla	Done	Alejandro Villalta		Development
Task	Llamar las operaciones modificar de los servicios e integrar con la pantalla	Done	Alejandro Villalta		Development
Task	Llamar las operaciones listar de los servicios e integrar con la pantalla	Done	Alejandro Villalta		Development
Task	Análisis de la historia de usuario	In Progress	Aida Elena Siles Rojas	0.25	Testing
Task	Diseño de las pruebas	To Do	Aida Elena Siles Rojas	0.25	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing

Figura 69: Tarea análisis de la historia de usuario estado en progreso

Fuente: https://siagpj.visualstudio.com

Seguidamente se inicia la tarea diseño de pruebas como se muestra en la Figura 70.

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Diseñar pantalla para agregar, modificar y consultar escolaridad	New		0.45	
Task	Diseñar la pantalla principal para el catalogo escolaridad (menú y consulta)	Done	Alejandro Villalta		Development
Task	Diseñar la pantalla para agregar y modificar escolaridad	Done	Alejandro Villalta		Development
Task	Llamar las operaciones insertar de los servicios e integrar con la pantalla	Done	Alejandro Villalta		Development
Task	Llamar las operaciones modificar de los servicios e integrar con la pantalla	Done	Alejandro Villalta		Development
Task	Llamar las operaciones listar de los servicios e integrar con la pantalla	Done	Alejandro Villalta		Development
Task	Análisis de la historia de usuario	Done	Aida Elena Siles Rojas		Testing
Task	Diseño de las pruebas	In Progress	Aida Elena Siles Rojas	0.25	Testing
Task	Ejecución de las pruebas	To Do	Aida Elena Siles Rojas	0.2	Testing

Figura 70: Tarea diseño de la historia de usuario estado en progreso

Fuente: https://siagpj.visualstudio.com

Mediante la herramienta VSTS se comienzan a crear los casos de prueba de cada historia de usuario, cada funcionalidad tiene su caso de prueba tanto de forma positiva como negativa. En la Figura 71 se muestran los casos de prueba del catálogo procedimiento.

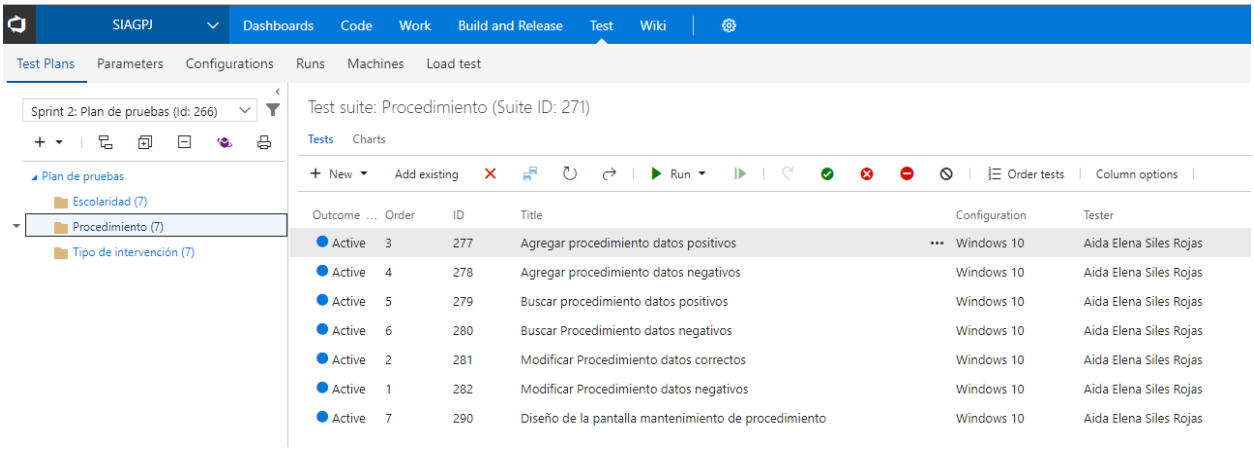


Figura 71: Casos de prueba por funcionalidad sprint 42

Fuente: https://siagpj.visualstudio.com

A continuación se indican las tareas realizadas por día durante la ejecución del plan piloto de proyecto y que corresponden a la etapa de análisis y diseño.

Día 1. Sprint 42. Fecha 08/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Permitir agregar un procedimiento	Análisis de la historia de usuario	0.42
Permitir agregar un procedimiento	Diseño de las pruebas	0.42
		0.84

Impedimentos

Ninguno

Día 2. Sprint 42. Fecha 09/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo
Permitir agregar un procedimiento	Análisis de la historia de usuario	0.42
Permitir agregar un procedimiento	Diseño de las pruebas	0.42
		0.84

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Permitir consultar un procedimiento	Análisis de la historia de usuario	0.3
Permitir consultar un procedimiento	Diseño de las pruebas	0.3
Permitir modificar un procedimiento	Análisis de la historia de usuario	0.42
Permitir modificar un procedimiento	Diseño de las pruebas	0.42
		1.44

Impedimentos

Ninguno

Día 3. Sprint 42. Fecha 10/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo
Permitir consultar un procedimiento	Análisis de la historia de usuario	0.3
Permitir consultar un procedimiento	Diseño de las pruebas	0.3
Permitir modificar un procedimiento	Análisis de la historia de usuario	0.42
Permitir modificar un procedimiento	Diseño de las pruebas	0.42
		1.44

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Diseñar pantalla para agregar, modificar y consultar procedimiento	Análisis de la historia de usuario	0.25
Diseñar pantalla para agregar, modificar y consultar procedimiento	Diseño de las pruebas	0.25
Permitir agregar una escolaridad	Análisis de la historia de usuario	0.5
		1

Impedimentos

Ninguno

Día 4. Sprint 42. Fecha 11/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo
Diseñar pantalla para agregar, modificar y consultar procedimiento	Análisis de la historia de usuario	0.25
Diseñar pantalla para agregar, modificar y consultar procedimiento	Diseño de las pruebas	0.25

Permitir agregar una escolaridad	Análisis de la historia de usuario	0.5
		1

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Permitir agregar una escolaridad	Diseño de las pruebas	0.5
Permitir consultar una escolaridad	Análisis de la historia de usuario	0.25
Permitir consultar una escolaridad	Diseño de las pruebas	0.25
		1

Impedimentos

Ninguno

Día 5. Sprint 42. Fecha 12/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo
Permitir agregar una escolaridad	Diseño de las pruebas	0.5
Permitir consultar una escolaridad	Análisis de la historia de usuario	0.25
Permitir consultar una escolaridad	Diseño de las pruebas	0.25
		1

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Permitir modificar una escolaridad	Análisis de la historia de usuario	0.3
Permitir modificar una escolaridad	Diseño de las pruebas	0.3
Diseñar pantalla para agregar, modificar y consultar escolaridad	Análisis de la historia de usuario	0.25
Diseñar pantalla para agregar,	Diseño de las pruebas	0.25

modificar y consultar escolaridad		
		1.1

Impedimentos

Ninguno

Día 6. Sprint 42. Fecha 15/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo
Permitir modificar una escolaridad	Análisis de la historia de usuario	0.3
Permitir modificar una escolaridad	Diseño de las pruebas	0.3
Diseñar pantalla para agregar, modificar y consultar escolaridad	Análisis de la historia de usuario	0.25
Diseñar pantalla para agregar, modificar y consultar escolaridad	Diseño de las pruebas	0.25
		1.1

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Permitir Agregar un tipo de intervención de un interviniente	Análisis de la historia de usuario	0.2
Permitir Agregar un tipo de intervención de un interviniente	Diseño de las pruebas	0.2
Permitir Consultar un tipo de intervención de un interviniente	Análisis de la historia de usuario	0.2
Permitir Consultar un tipo de intervención de un interviniente	Diseño de las pruebas	0.2
Permitir Modificar un tipo de intervención de un interviniente	Análisis de la historia de usuario	0.2
		1

Impedimentos

Ninguno

Día 7. Sprint 42. Fecha 16/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo
Permitir Agregar un tipo de intervención de un interviniente	Análisis de la historia de usuario	0.2
Permitir Agregar un tipo de intervención de un interviniente	Diseño de las pruebas	0.2
Permitir Consultar un tipo de intervención de un interviniente	Análisis de la historia de usuario	0.2
Permitir Consultar un tipo de intervención de un interviniente	Diseño de las pruebas	0.2
Permitir Modificar un tipo de intervención de un interviniente	Análisis de la historia de usuario	0.2
		1

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Permitir Modificar un tipo de intervención de un interviniente	Diseño de las pruebas	0.2
Diseñar pantalla para el mantenimiento de Tipo de intervención de un interviniente	Análisis de la historia de usuario	0.25
Diseñar pantalla para el mantenimiento de Tipo de intervención de un interviniente	Diseño de las pruebas	0.25
		0.7

Impedimentos

Ninguno

Una vez completadas las tareas de análisis y diseño de cada historia de usuario, se procede con la siguiente etapa, la cual corresponde a la ejecución de las pruebas.

7.3.3 Ejecución de las pruebas

Se procede a establecer la tarea ejecución de pruebas “en progreso” para la historia de usuario que corresponda.

Work Item Type	Title	State	Assigned To	Remaining Work	Activity
Product Backlog Item	Permitir agregar un procedimiento	New		0.42	
Task	Crear tabla Procedimiento	Done	Uriel García R.		Development
Task	Crear acceso a datos	Done	Uriel García R.		Development
Task	Crear la entidad procedimiento	Done	Uriel García R.		Development
Task	Crear la lógica de negocio	Done	Uriel García R.		Development
Task	Crear servicio para agregar procedimientos	Done	Uriel García R.		Development
Task	Análisis de la historia de usuario	Done	Aida Elena Siles Rojas		Testing
Task	Diseño de las pruebas	Done	Aida Elena Siles Rojas		Testing
Task	Ejecución de las pruebas	In Progress	Aida Elena Siles Rojas	0.42	Testing

Figura 72: Tarea ejecución de las pruebas de la historia de usuario estado en progreso

Fuente: https://siagpj.visualstudio.com

Se inicia con la ejecución de los casos de prueba, mediante la herramienta VSTS en el apartado o *branch* de pruebas, como se muestra en la Figura 73.

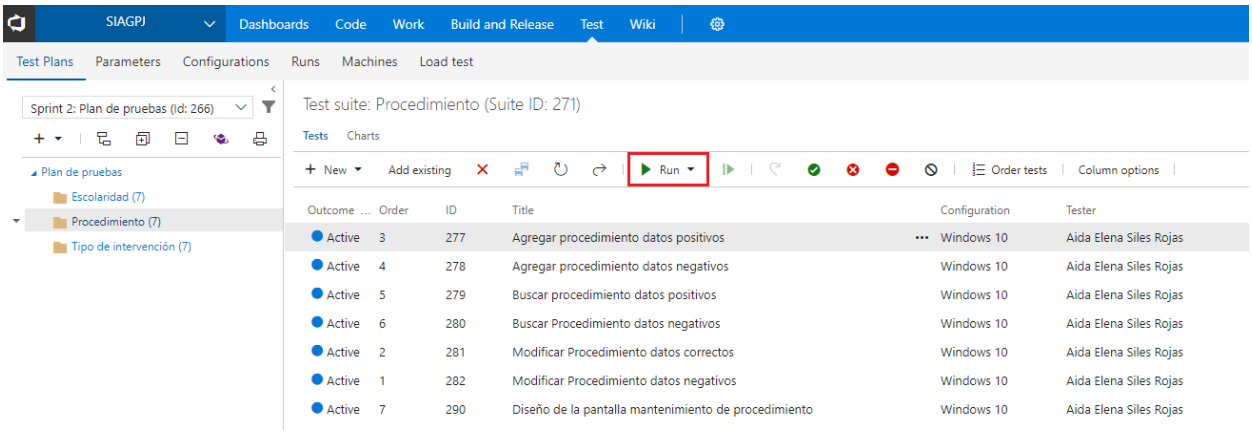


Figura 73: Ejecución de casos de prueba con la herramienta VSTS

Fuente: https://siagpj.visualstudio.com

Se comienzan a validar los pasos, si todos los pasos pasaron se considera la prueba como pasada, caso contrario la prueba se establece como fallida. A continuación en la Figura 74 se

muestra cómo se verifican los pasos. Durante la ejecución del plan piloto del proyecto, el siguiente caso de prueba se determinó como fallido, ya que al momento de agregar un procedimiento la descripción se está cortando a 100 caracteres.

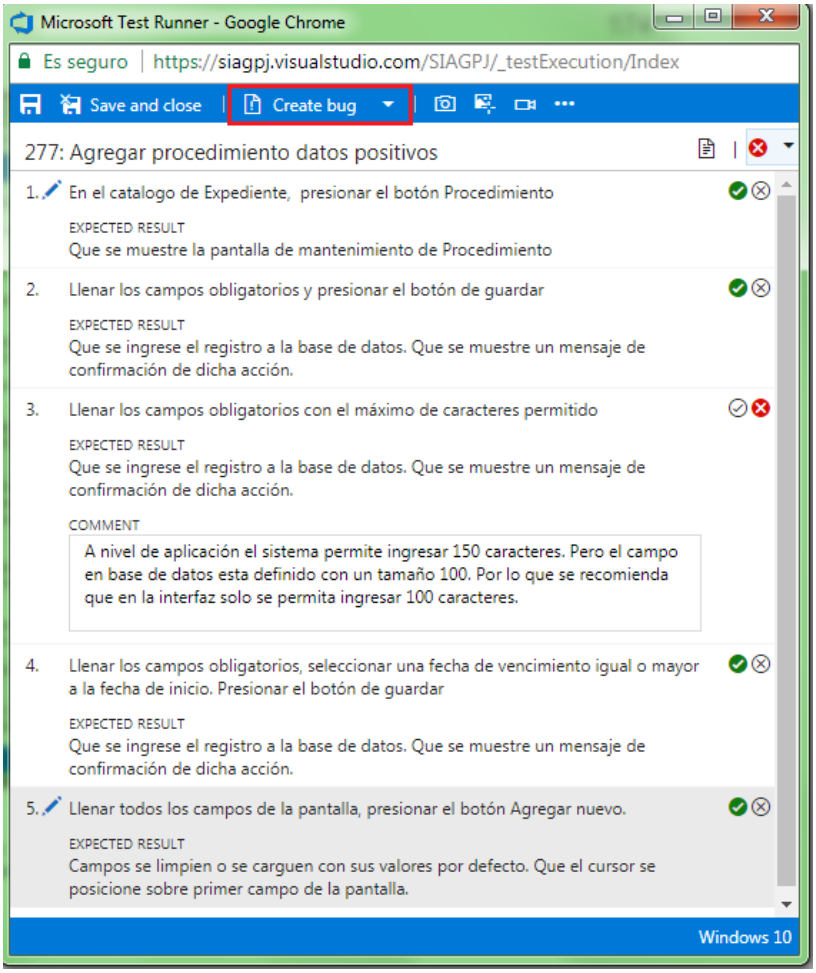


Figura 74: Verificación pasos de un caso de prueba

Fuente: https://siagpj.visualstudio.com

Se creó la pulga o bien el defecto que se mencionó anteriormente.

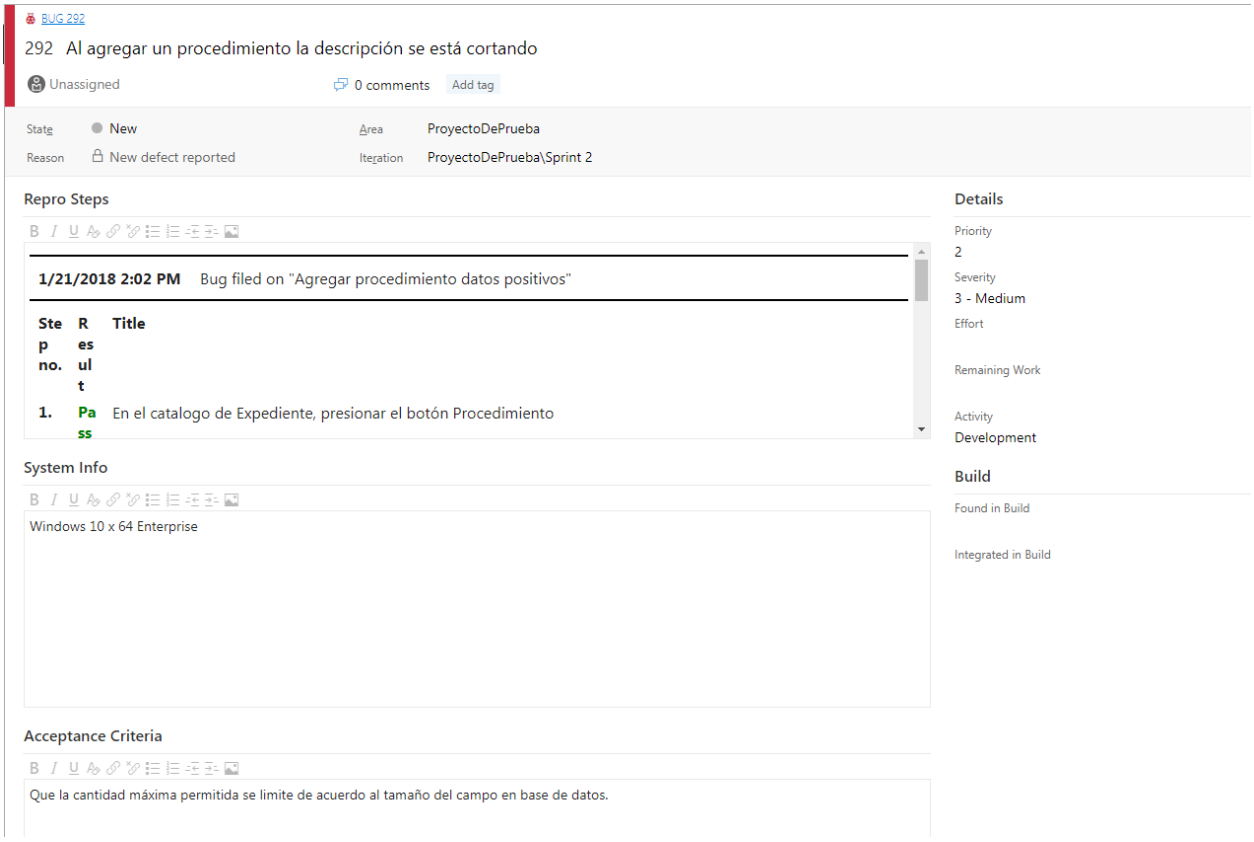


Figura 75: Defecto registrado sprint 42

Fuente: https://siagpj.visualstudio.com

A continuación se indican las tareas realizadas por día durante la ejecución del plan piloto de proyecto y que corresponden a la etapa de ejecución de las pruebas.

Día 8. Sprint 42. Fecha 17/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo
Permitir Modificar un tipo de intervención de un interviniente	Diseño de las pruebas	0.2
Diseñar pantalla para el mantenimiento de Tipo de intervención de un interviniente	Análisis de la historia de usuario	0.25

Diseñar pantalla para el mantenimiento de Tipo de intervención de un interviniente	Diseño de las pruebas	0.25
		0.7

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Permitir agregar un procedimiento	Ejecución de las pruebas	0.42
Permitir consultar un procedimiento	Ejecución de las pruebas	0.2
Permitir modificar un procedimiento	Ejecución de las pruebas	0.2
Diseñar pantalla para agregar, modificar y consultar procedimiento	Ejecución de las pruebas	0.15
		0.97

Impedimentos

Ninguno

Día 9. Sprint 42. Fecha 18/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo
Permitir agregar un procedimiento	Ejecución de las pruebas	0.42
Permitir consultar un procedimiento	Ejecución de las pruebas	0.2
Permitir modificar un procedimiento	Ejecución de las pruebas	0.2
Diseñar pantalla para agregar, modificar y consultar procedimiento	Ejecución de las pruebas	0.15
		0.97

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Permitir Agregar una escolaridad	Ejecución de las pruebas	0.42
Permitir Consultar una escolaridad	Ejecución de las pruebas	0.2
Permitir Modificar una escolaridad	Ejecución de las pruebas	0.2
Diseñar pantalla para agregar, modificar y consultar escolaridad	Ejecución de las pruebas	0.2
		1.02

Impedimentos

Ninguno

Día 10. Sprint 42. Fecha 19/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo
Permitir Agregar una escolaridad	Ejecución de las pruebas	0.42
Permitir Consultar una escolaridad	Ejecución de las pruebas	0.2
Permitir Modificar una escolaridad	Ejecución de las pruebas	0.2
Diseñar pantalla para agregar, modificar y consultar escolaridad	Ejecución de las pruebas	0.2
		1.02

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Permitir Agregar un tipo de intervención de un interviniente	Ejecución de las pruebas	0.2
Permitir Consultar un tipo de intervención de un interviniente	Ejecución de las pruebas	0.2
Permitir Modificar un tipo de intervención de un interviniente	Ejecución de las pruebas	0.2
Diseñar pantalla para el mantenimiento de Tipo de intervención de un interviniente	Ejecución de las pruebas	0.15
		0.75

Impedimentos

Ninguno

Día 11. Sprint 42. Fecha 22/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo
Permitir Agregar un tipo de intervención de un interviniente	Ejecución de las pruebas	0.2
Permitir Consultar un tipo de intervención de un interviniente	Ejecución de las pruebas	0.2
Permitir Modificar un tipo de intervención de un interviniente	Ejecución de las pruebas	0.2
Diseñar pantalla para el mantenimiento de Tipo de intervención de un interviniente	Ejecución de las pruebas	0.15
		0.75

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Tareas proceso control de calidad sprint 42	Realizar pruebas de regresión sprint 42	1

Impedimentos

Ninguno

Día 12. Sprint 42. Fecha 23/01/2018

El integrante de calidad se hace presente en la reunión diaria de Scrum, y expone sus tareas realizadas, por realizar y los impedimentos.

Tareas realizadas

Product Backlog Item	Tareas	Tiempo
Tareas proceso control de calidad sprint 42	Realizar pruebas de regresión sprint 42	1
		1

Tareas a realizar

Product Backlog Item	Tareas	Tiempo
Tareas proceso control de calidad sprint 42	Realizar reporte de resultados de las pruebas para el sprint 42	1
		1

Impedimentos

Ninguno

7.3.4 Reporte de resultados

Como resultado de la planeación, diseño y ejecución de las pruebas, se obtiene la siguiente plantilla que se propuso en el capítulo VI.

Tabla 24: Plantilla reporte de pruebas realizadas sprint 42

Plantilla Reporte de Pruebas Realizadas Sprint 42	
Identificador: RSS42	**Fecha de elaboración:** 23/01/2018
Nombre del proyecto: SIAGPJ	
Integrante(s) del equipo de calidad:	

Aida Elena Siles Rojas

Descripción:

El siguiente reporte muestra los resultados obtenidos durante el proceso de control de calidad realizado en el sprint 42.

Niveles de pruebas: <Indicar los niveles de pruebas realizados>

☐Pruebas unitarias	☒Pruebas de sistema
☒Pruebas de integración	☐Pruebas de aceptación

Tipos de pruebas:

Tipos de pruebas	Prueba Satisfactoria			Observaciones
	Sí	No	N/A	
☐Pruebas de instalación	☒	☐	☐	
☐Pruebas funcionales	☒	☐	☐	
☐Pruebas de rendimiento	☐	☐	☐	
☐Pruebas de carga	☐	☐	☐	
☐Pruebas de estrés	☐	☐	☐	
☐Pruebas de compatibilidad	☐	☐	☐	
☐Pruebas de portabilidad	☐	☐	☐	
☐Pruebas de usabilidad	☒	☐	☐	
☐Pruebas de accesibilidad	☐	☐	☐	
☐Pruebas de seguridad	☐	☐	☐	
☐Pruebas de Interfaz	☒	☐	☐	
☐Pruebas de documentación	☒	☐	☐	
☐Pruebas de confirmación	☐	☐	☐	
☐Pruebas de regresión	☒	☐	☐	

Pruebas de Instalación		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica la publicación del *deploy* o *release* en los servidores de pruebas.	☒	☐
2. Se verifica la instalación de los componentes en Windows 10 x64 de escritorio y de Tablet.	☒	☐
3. Se verifica la instalación y actualización de los servicios web API.	☒	☐
4. Se verifica que la documentación de los detalles de implementación y el proceso de instalación sean consistentes entre sí.	☒	☐

Pruebas Funcionales		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica la funcionalidad de cada requerimiento y que cumpla con el criterio de aceptación.	☐	☒
2. Se verifica que los mensajes de advertencia, error, y de confirmación de acciones concuerden con la funcionalidad del sistema.	☒	☐
3. Se verifica la funcionalidad en diferentes contextos del sistema.	☒	☐
4. Se verifica la funcionalidad con diferentes roles del sistema.	☒	☐

Pruebas de Usabilidad		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica que la funcionalidad sea fácil de aprender y de utilizar.	☒	☐
2. Se verifica que exista orden tabular entre los campos para facilidad de navegación.	☒	☐
3. Se verifica que exista un manejo de colores adecuado entre los diferentes controles y objetos del sistema.	☒	☐
4. Se verifica que la lectura de los títulos, botones, tooltip, y demás controles no sea difícil.	☐	☒
5. Se verifica que los campos obligatorios se distingan.	☒	☐
6. Se verifica que los campos y controles del sistema muestren la descripción emergente (tooltip) al posicionar el cursor sobre los objetos.	☐	☒
7. Se verifica que las diferentes pantallas del sistema brinden un entorno agradable y se facilite el entendimiento de la información presentada.	☒	☐
8. Se verifica que los campos obligatorios cuenten con mensajes de advertencia.	☒	☐

Pruebas de Interfaz		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica la navegación a través de los objetos de cada pantalla evaluada.	☒	☐
2. Se verifica la alineación de cada uno de los objetos evaluados de las pantallas del sistema.	☒	☐
3. Se verifica la ortografía de cada título, botón, tooltip y mensaje, entre otros.	☐	☐

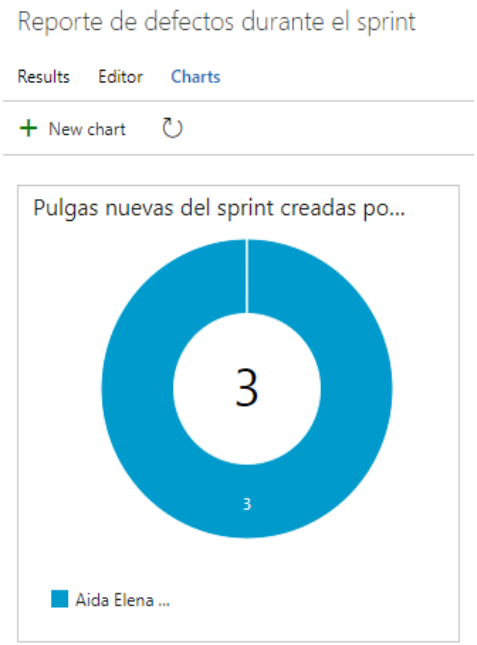
Pruebas de Documentación		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica que la documentación de usuario y el comportamiento del sistema sean consistentes entre sí.	☒	☐
2. Se verifica que sea legible, que tenga buena redacción y que sea comprensible.	☒	☐

Pruebas de Regresión		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica que el cambio realizado no haya afectado la funcionalidad existente.	☒	☐

Impedimentos:

No se presentaron impedimentos.

Nuevos defectos registrados:



Fuente: Elaboración propia, 2018.

7.3.5 Revisión y retrospectiva

Se participa de la revisión del sprint, se expone al equipo las pruebas realizadas durante el *sprint* así como los resultados obtenidos. Adicionalmente se participa en la retrospectiva del *sprint*, evaluando el proceso de control de calidad realizado para proponer futuras mejora. En este caso únicamente se propuso que en el siguiente *sprint* se agregara, al reporte de resultados de *testing*, los casos de prueba realizados en el *sprint*.

Tabla 25: Retrospectiva sprint 42

Retrospectiva Sprint #42			
Fecha:	24/01/2018		
Proyecto:	SIAGPJ		
Participantes:	Product Owner – Vivian Rímola Scrum Master – Gian Muir Desarrolladores: Andres Díaz, Isaac Dobles, Jonathan Aguilar, Johan Acosta, Roger Lara, Pablo Álvarez. Tester – Aida Siles Rojas		
Aspectos que salieron bien durante el sprint - La definición de las tareas y la estimación del tiempo. - Ejecución de las pruebas con las listas de chequeo.			
Aspectos que deben mejorarse para el siguiente sprint - Se debe mejorar el tema de los reportes de testing, se quiere agregar los casos de prueba realizados en el sprint. - Se debe mejorar el tema de pruebas automatizadas y el uso de la herramienta Visual Studio Team Services.			
Elaborado por:	Aida Elena Siles Rojas		
Aprobado por:	Product Owner y Scrum Master equipo SIAGPJ		

Fuente: Elaboración propia, 2018.

7.4 Resultados obtenidos con la metodología

Producto de la implementación de la metodología se obtienen los siguientes resultados:

a. La realización de un análisis de riesgos sobre las historias de usuario permitió facilitar la estimación del esfuerzo de pruebas para cada una de las historias de usuario.
b. La definición y ejecución de las tareas de *testing* durante el *sprint* permitieron que el *product owner* y el *scrum master* tengan un mejor control sobre estado actual del proceso de *testing*.
c. De acuerdo con el *product owner* del proyecto, la definición y ejecución de los casos de prueba dan confiabilidad de que el incremento o producto a liberar es de calidad.
d. La integración del *tester* al proceso de desarrollo permite que los desarrolladores tengan una retroalimentación rápida de lo elaborado, con la ventaja de que los defectos sean corregidos en el mismo *sprint*, para al final tener un incremento libre de defectos.
e. Las listas de chequeo permiten evaluar aspectos básicos fundamentales en el software y brindar un apoyo para aquellos nuevos recursos que ingresan al área de calidad.
f. El reporte de resultados informa tanto a la jefatura como al equipo del proyecto qué pruebas fueron aplicadas, y cuáles fueron satisfactorias y cuáles no. Actualmente se desconoce los tipos de pruebas que fueron aplicadas.

CAPÍTULO VIII

8. Conclusiones y recomendaciones

8.1 Conclusiones

Conforme a los resultados de la investigación realizada, se pueden agrupar las conclusiones según corresponden a los objetivos mencionados en el Capítulo I.

Objetivo general: Proponer una metodología de control de calidad del software adaptada a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el área de informática de Gestión del departamento de la DTIC del Poder Judicial.

a. Se encontró que existen muchas carencias en el proceso de control de calidad actual, como lo es la falta de una planeación adecuada de las pruebas y el registro de los casos de prueba.
b. Se concluye con base en las entrevistas realizadas que el área de Informática de Gestión trabaja bajo la metodología Scrum, pero no existe una documentación formal dentro de la institución que respalde dicha metodología.
c. Con base en las encuestas, entrevistas y el diagnóstico de la situación actual, se logró identificar una serie de brechas y defectos del proceso de control de calidad actual del área de Informática de Gestión. Dicha información se utilizó como insumo para diseñar la metodología de control de calidad ajustada a Scrum y cerrar las brechas de calidad encontradas.

Objetivo específico 1: Definir la situación actual del proceso de control de calidad del software que se realiza en los proyectos desarrollados bajo Scrum del Área de Informática de Gestión.

d. Con base en las encuestas se determinó que un 93% de las respuestas relacionadas con el proceso de control de calidad y su alineamiento con la metodología Scrum fueron

negativas, por lo que se concluye que el proceso de control de calidad que actualmente realiza el equipo de *testing* no se encuentra alineado con la metodología de desarrollo ágil Scrum. La presente investigación propone una metodología de control de calidad que permite alinear los procesos de calidad con Scrum, incorporando en sus procesos las recomendaciones del marco metodológico Scrum, las buenas prácticas de la ingeniería del software y los conceptos de mejora continua de la calidad.

e. Con base en la pregunta 7 de la encuesta al equipo de desarrollo y la pregunta 8 al coordinador de calidad se logró identificar que el área de *testing* no cuenta con una metodología de control de calidad que establezca los procedimientos a seguir y su respectiva documentación.
f. Se identificó mediante la pregunta 21 de la encuesta realizada al coordinador de calidad que actualmente no existe un proceso de validación de las pruebas, por lo que se desconoce si el trabajo realizado por el equipo de calidad es correcto.
g. Se encontró con base en la pregunta 11 de la encuesta al equipo de desarrollo y la pregunta 15 de la encuesta al coordinador de calidad que el área de *testing* actualmente no realiza pruebas de rendimiento, carga y estrés.

Objetivo específico 2: Definir la brecha entre el proceso de control de calidad actual del Área de Informática de Gestión contra lo establecido en las metodologías de calidad de software enfocadas a ambientes ágiles.

h. Se determina que el equipo de calidad no forma parte de ninguna reunión de Scrum, lo cual genera que este equipo desconozca aspectos importantes del proyecto como procesos, análisis, tareas realizadas, impedimentos y mejoras a realizar.

i. Se determinó que el área carece de una etapa de planeación de pruebas, lo que ocasiona que etapas posteriores a esta se vean afectadas durante de la ejecución, y se generan atrasos de entrega del producto.
j. Se encontró que el equipo de calidad no registra las tareas que se deben realizar para completar el proceso de pruebas. Lo anterior dificulta el tema de costos de calidad y a la vez de evaluación del desempeño.

Objetivo específico 3: Diseñar un marco de trabajo de control de la calidad de software conforme a la metodología Scrum y adaptado al proceso de desarrollo ágil de la institución.

k. Se concluye que uno de los aspectos fundamentales de un proceso de control de calidad basado en Scrum es la participación del equipo de *testing* en la definición de requerimientos.
l. Se determinó que la elaboración de un análisis de riesgos en etapas tempranas permite que la estimación de las tareas de *testing* sean más precisas y a la vez ayudan a determinar la cobertura y esfuerzo de las pruebas.
m. Con base en la pregunta 12 de la encuesta realizada al coordinador de calidad, se concluye que no existe una documentación adecuada que evidencie las pruebas realizadas, por lo tanto la presente investigación se va enfocar en el desarrollo de dichos documentos.

Objetivo específico 4: Implementar un plan piloto de la metodología ágil de control de calidad en el área de informática de gestión para el proyecto SIAGPJ.

n. Se determina que el Área de Informática de Gestión actualmente desarrolla el proyecto SIAGPJ bajo la metodología Scrum, y que dicho proyecto se utilizará como un plan

piloto para la implementación de la metodología de control de calidad de la presente investigación.

o. Se concluye que el equipo de calidad no cuenta con un proceso estandarizado para la evaluación de las pruebas, por lo tanto en la implementación del plan piloto se proporciona unas listas de chequeo que brindan un modelo a seguir para la evaluación del software.

8.2 Recomendaciones

Conforme a los resultados de la investigación realizada, se pueden agrupar las recomendaciones según correspondan a los objetivos mencionados en el Capítulo I.

Objetivo general: Proponer una metodología de control de calidad del software adaptada a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el área de informática de Gestión del departamento de la DTIC del Poder Judicial.

a. Se recomienda a la Jefatura del área de Informática de Gestión poner en práctica la metodología de control de calidad propuesta en esta investigación para corregir los principales problemas y cerrar las brechas que afectan el proceso de calidad y de entrega de los productos, y retroalimentarla para su mejoramiento.
b. Se recomienda a la Jefatura y al coordinador de calidad, ambos del área de Informática de Gestión, integrar la metodología de control de calidad de forma preventiva y no reactiva. En caso que está investigación sea efectiva, se recomienda que la metodología sea un estándar para el área.
c. Se recomienda continuar con la implementación de la metodología del control de calidad adicionando mejoras.

Objetivo específico 1: Definir la situación actual del proceso de control de calidad del software que se realiza en los proyectos desarrollados bajo Scrum del Área de Informática de Gestión.

d. Se recomienda a la Jefatura del área de Informática de Gestión asignar a una persona con cualidades de *tester* y con conocimiento en Scrum por tiempo completo al proyecto SIAGPJ, que permita realizar un proceso de aseguramiento de calidad bajo la metodología propuesta en la presente investigación para obtener mejoras, como lo es la reducción de los costos en cuanto a corrección de defectos y mejora en la calidad del software.

Objetivo específico 2: Definir la brecha entre el proceso de control de calidad actual del Área de Informática de Gestión contra lo establecido en las metodologías de calidad de software enfocadas a ambientes ágiles.

e. Se recomienda al coordinador del equipo de calidad y al *product owner* del área de Informática de Gestión implementar una estrategia de pruebas basadas en riesgos, para asegurar que las funcionalidades críticas del negocio han sido probadas.

Objetivo específico 3: Diseñar un marco de trabajo de control de la calidad de software conforme a la metodología Scrum y adaptado al proceso de desarrollo ágil de la institución.

f. Se recomienda al *product owner* del área de Informática de Gestión fomentar la participación del equipo de calidad en la etapa de planificación del *release* para validar la calidad de las historias de usuario, así como ayudar en la elaboración de los criterios de aceptación de cada historia de usuario.

Objetivo específico 4: Implementar un plan piloto de la metodología ágil de control de calidad en el área de informática de gestión para el proyecto SIAGPJ.

g. Se recomienda al equipo de calidad del área de Informática de Gestión almacenar los resultados del plan piloto para utilizarlos como punto de comparación para futuros resultados de otras pruebas de calidad, y con esto afinar la metodología de control de calidad propuesta.

9. Bibliografía

Alexander Menzinsky, G. L. (2016). *Scrum Manager.*

Amo, F. A., Normand, L. M., & Pérez, F. J. (2005). *Introducción a la ingeniería del software.* Delta Publicaciones.

Árticulo de Interés. (22 de 7 de 2016). *Sistema Integrado de Apoyo a la Gestión de Procesos Jurisdiccionales*. Costa Rica.

Azofeifa, Y. (2015). *Estudio de Factibilidad Sistema Integral de Apoyo a la Gestión de los Procesos Jurisdiccionales (SIAGPJ).* Goicoechea, San Jose, Costa Rica.

Black, R., Claesson, A., Coleman, G., Cornanguer, B., Forgacs, I., Linetzki, A., . . . Weber, S. (2014). *Foundation Level Extension Syllabus Agile Tester.*

Buonamico, D. (06 de 08 de 2016). Obtenido de http://www.caminoagil.com/2016/08/estrategias-de-testing-en-equipos-scrum.html

CMS. (2005). *Selecting a development approach.*

Cohn, M. (2010). *Succeeding With Agile.*

Contraloría General de la República. (2009). *Normas de control interno para el Sector Público.*

Crispin, L., & Gregory, J. (2009). *Agile Testing: A Practical Guide for Testers and Agile Teams.*

Daniel, B. (22 de 10 de 2014). *coderewind*. Obtenido de http://www.coderewind.com/2014/10/software-testing-levels-follow/

Dirección de Tecnologías de Información y Comunicaciones Poder Judicial. (2015). *Plan Estratégico de Tecnologías de Información y Comunicaciones 2015-2020.* Plan Estratégico Informática, San José.

Dirección de Tecnologías de la Información y Comunicaciones, Poder Judicial Costa Rica. (2016). Artículos de Interés - Scrum. *Artículos de Interés*. Recuperado el 14 de 08 de 2017

Erinle, B. (2015). *Performance Testing with JMeter* (2 ed.). Packt Publishing Ltd.

Fuentes, J. R. (2014). *Desarrollo de Software Ágil: Extreme Programming y Scrum.* Createspace Independent Publishing Platform.

Garita, E. M. (2014). *Propuesta de un Modelo Metodológico para la Planificación y la Gestión de Pruebas del Control de Calidad de Software.* San José, San José, Costa Rica. Recuperado el 09 de 09 de 2017

Garzás, J. (1 de 08 de 2012). *javiergarzas.com*. Obtenido de javiergarzas.com: http://www.javiergarzas.com/2012/08/calidad-del-producto-software-proceso-equipo.html

Ghahrai, A. (03 de 01 de 2015). *Testing Excellence*. Obtenido de https://www.testingexcellence.com/exploratory-testing-important-agile-projects/

Giardina, M. (7 de 06 de 2013). *ScrumAlliance*. Obtenido de ScrumAlliance: https://www.scrumalliance.org/community/articles/2013/june/testers-working-in-an-agile-team

Graham, D., Veenendaal, E. v., Evans, I., & Black, R. (2008). *Software Testing ISTQB Certification.* Cengage Learning EMEA.

Guru99. (s.f.). *Guru99*. Obtenido de https://www.guru99.com/exploratory-testing.html

Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2010). *Software Testing An ISTQB-ISEB Foundation Guide* (2 ed.).

Hendrickson, E. (11 de 8 de 2008). Agile Testing Nine Principles and Six Concrete Practices for Testing.

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2010). *Metodología de la investigación* (5 ed.). McGrawHill.

International Software Testing Qualifications Board. (2016). *ISTQB*. Obtenido de http://www.istqb.org/certification-path-root/agile-tester-extension/agile-tester-extension-in-a-nutshell.html

ISO. (03 de 2011). *International Organization for Standardization*. Obtenido de International Organization for Standardization: https://www.iso.org/standard/35765.html

ISO. (09 de 07 de 2017). *International Organization for Standardization*. Obtenido de www.iso.org

iso25000. (s.f.). *iso25000*. Obtenido de http://iso25000.com/index.php/normas-iso-25000/iso-25040

ISTQB. (2016). *ISTQB*. Obtenido de www.istqb.org

ISTQB. (2016). *Standard Glossary of Terms used in Software Testing Version 3.1* (3.1 ed.).

Kasturi, R. (2014). *Scrum Communication.* Obtenido de https://www.scrumalliance.org/community/articles/2014/october/scrum-communication

Khalane, T. (2013). *Software Quality Assurance in Scrum.* Ciudad del Cabo, Sudáfrica.

Knight Errant. (2012). *The importance of testing in an AGILE development context.*

Linz, T. (2014). *Testing in Scrum: A guide for Software Quality Assurance in the Agile World.* Rocky Nook Inc.

Lorenzo, F. S., & Parra, J. G. (2014). *I Jornada sobre Calidad del Producto Software e ISO 25000.*

McKey, J., & Bath, G. (2014). *The Software Test Engineer's Handbook* (2 ed.). Rocky Nook.

Méndez, Ó. (22 de 12 de 2015). *paradigmadigital*. Recuperado el 16 de 09 de 2017, de https://www.paradigmadigital.com/techbiz/stack-de-lujo-para-control-de-calidad-en-entornos-agiles-12/

Menzinsky, A., López, G., & Palacio, J. (2016). *Scrum Manager.*

Microsoft. (26 de 09 de 2017). *Microsoft*. Obtenido de https://docs.microsoft.com/en-us/vsts/load-test/run-performance-tests-app-before-release

Microsoft. (2017). *Microsoft Developer Network*. Obtenido de https://msdn.microsoft.com/es-es/library/jj635157.aspx

Oktaba, H. (s.f.). *SG Buzz*. Obtenido de https://sg.com.mx/content/view/990

Paez, N., Cyment, A., Tortorella, P., Salías, M., Peix, C., Fontdevila, D., . . . Christie, T. (2015). *Experiencias Agiles*. Obtenido de https://nicopaez.gitbooks.io/libroagileaoc2015/10-testing-con-scrum.html

Pandey, A. (16 de 02 de 2014). *Anand Pandey*. Obtenido de http://www.anandpandey.com/post/2014/02/16/scrum-testing-methodology

PMI. (2013). *Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK)* (5 ed.).

PMOinformatica.com. (19 de 09 de 2012). *PMOinformatica.com*. Obtenido de http://www.pmoinformatica.com/2012/09/test-driven-development-scrum.html

Poder Judicial de Costa Rica. (12 de 04 de 2016). *Poder Judicial República de Costa Rica.* Recuperado el 22 de 09 de 2017, de Poder Judicial República de Costa Rica: https://pj.poder-judicial.go.cr/images/documentos/generalidades/historia-organizacion-funcionamiento.pdf

Pressman, R. S. (2010). *Ingeniería del Software Un enfoque práctico* (7 ed.). McGrawHill.

Rajani, R. (8 de 11 de 2016). Testing in Agile Development and State of Agile Adoption. Obtenido de https://www.capgemini.com/blog/capping-it-off/2016/11/testing-in-agile-development-and-state-of-agile-adoption

Rex Black, E. v. (2012). *Foundations of Software Testing ISTQB Certification.*

Rímola, V., & Azofeifa, Y. (2017). *Informe Gerencial Avance Sistema Gestion Nuevo Agosto 2017.* San Jose, Costa Rica. Recuperado el 09 de 24 de 2017

Rubin, K. S. (2013). *Essential Scrum. A practical Guide to the Most Popular Agile Process.* Pearson Education, Inc.

Sánchez, J. Z. (13 de 1 de 2013). Obtenido de https://pruebasdelsoftware.wordpress.com/tag/metodologia-de-pruebas/

Schaefer, H., Linz, T., & Spillner, A. (2014). *Software Testing Foundations* (4 ed.). Rocky Nook.

Schwaber, K. (2004). *Agile Project Management with Scrum.* Washington: Microsoft Press.

Schwaber, K., & Sutherland, J. (2016). *The Scrum Guide.*

Sinhg, S. (22 de 09 de 2014). *SlideShare*. Obtenido de https://es.slideshare.net/nidhisaroj/writing-test-cases-in-agile-1

Software Testing Class. (16 de 10 de 2013). Obtenido de http://www.softwaretestingclass.com/positive-and-negative-testing-in-software-testing/

Software Testing Fundamentals. (s.f.). *Software Testing Fundamentals*. Obtenido de http://softwaretestingfundamentals.com/functional-testing/

Sommerville, I. (2011). *Ingeniería de Software* (9 ed.). Pearson Educación.

Stark, M. (11 de 08 de 2014). *iBeta Quality Assurance*. Obtenido de http://www.ibeta.com/functional-vs-non-functional-testing-whats-difference/

Testeando Software. (14 de 01 de 2014). *testeandosoftware*. Obtenido de https://testeandosoftware.com/istqb-que-es-cuales-son-los-niveles-de-certificacion/

The Apache Foundation. (10 de 21 de 2017). *Apache Jmeter*. Obtenido de http://jmeter.apache.org/

TMMi Foundation. (2015). *Test Maturity Model Integration.*

uTest. (2011). *Agile Software Testing.* Massachusetts, United States.

Varhol, P. (2010). *Integrating Testing into Agile Development.*

VersionOne. (2015). *The 10th Annual State of Agile.*

VersionOne. (2016). *VersionOne 11th Annual State of Agile Report.*

Villalobos, C. C. (2015). *Propuesta de mejoramiento para el proceso de administración de la calidad del software basado en el estándar ISTQB en la Compañía New Line Consultants S.A.* San José. Recuperado el 09 de 09 de 2017

Watkins, J. (2009). *Agile Testing How to Succeed in an Extreme Testing Environment.* Cambridge University Press.

XBOSoft Inc. (2012). *The XBOSoft 2012 Scrum Testing Survey.*

10. ANEXOS

Anexo 1: Artículo de Interés Scrum en el Poder Judicial

Fuente: Página Intranet del Poder Judicial

Anexo 2: Artículo de interés Sistema Integrado de Apoyo a la Gestión de Procesos Jurisdiccionales (SIAGPJ)

Fuente: Página Intranet del Poder Judicial

Anexo 3: Plantilla control de aprobación de pruebas

<table>
<tr><th colspan="2">Control de Aprobación de Pruebas</th></tr>
<tr><td colspan="2">Fecha: Fecha de la prueba</td></tr>
<tr><td>NOMBRE</td><td>[Nombre de la persona que realiza la prueba]</td></tr>
<tr><td>SISTEMA</td><td>[Nombre del sistema que se está probando]</td></tr>
<tr><td>VERSIÓN</td><td>[versión a la que corresponde la mejora-error-versión]</td></tr>
<tr><td>NÚMERO</td><td>[NUMERO DE MEJORA-ERROR-VERSION]</td></tr>
<tr><td>Detalle de Mejora
(Desarrollo)</td><td>[Descripción de la mejora-error-versión que se envía a pruebas normalmente es la descripción de la solicitud]</td></tr>
<tr><td>Observación
(Tester)</td><td>[Descripción detallada de los Escenarios o resultado de las pruebas]</td></tr>
<tr><td>Lista de chequeo de la prueba</td><td>
<table>
<tr><th>Tipos de Pruebas</th><th colspan="2">Prueba satisfactoria</th><th>Observaciones</th></tr>
<tr><td></td><td>SI</td><td>NO</td><td></td></tr>
<tr><td>INTEGRACIÓN</td><td></td><td></td><td></td></tr>
<tr><td>INTEFAZ DE USUARIO</td><td></td><td></td><td></td></tr>
<tr><td>SISTEMA:</td><td></td><td></td><td></td></tr>
<tr><td>Validación</td><td></td><td></td><td></td></tr>
<tr><td>Seguridad</td><td></td><td></td><td></td></tr>
<tr><td>Integridad de las bases de datos</td><td></td><td></td><td></td></tr>
<tr><td>Instalación</td><td></td><td></td><td></td></tr>
<tr><td>FUNCIONALES:</td><td></td><td></td><td></td></tr>
<tr><td>Funcional</td><td></td><td></td><td></td></tr>
<tr><td>Usabilidad</td><td></td><td></td><td></td></tr>
<tr><td>Instalación</td><td></td><td></td><td></td></tr>
<tr><td>REGRESIÓN</td><td></td><td></td><td></td></tr>
<tr><td>DOCUMENTACIÓN DE USUARIO</td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Resultado de la Prueba</td><td>☐ Reprobada ☐ Aprobada</td></tr>
</table>

Fuente: Documento proporcionado por el coordinador del área de calidad

Anexo 4: Encuesta para el área de desarrollo

Encuesta # 1

Para el área de desarrollo

La siguiente encuesta forma parte de los instrumentos de investigación del proyecto final llamado “Propuesta de una metodología de control de calidad del software adaptada a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el Área de informática de Gestión del Poder Judicial, II Circuito Judicial de San José en el III Cuatrimestre de 2017”, elaborado por la estudiante Aida Elena Siles Rojas para optar por el Grado de Licenciatura en Sistemas de Información de la Universidad Hispanoamericana. El objetivo de esta encuesta es identificar la situación actual del proceso de control de calidad llevado a cabo en los desarrollos bajo la metodología Scrum en el Área de Informática de Gestión del Poder Judicial de Costa Rica.

Información general de la persona encuestada

1. ¿Cuál es su grado académico?

   ( ) Bachiller

   ( ) Licenciatura

   ( ) Maestría

2. Marque el rango de años que corresponda a su experiencia laboral en el área de software:

   ( ) Menos de un año

   ( ) 1 a 3 años

   ( ) 3 a 5 años

   ( ) 5 a 7 años

   ( ) Más de 7 años

Información general sobre la metodología de desarrollo

3. ¿Conoce la metodología de desarrollo Scrum? (Si su respuesta es No, continuar con la pregunta 6)

   ( ) Sí

   ( ) No

4. ¿Hace uso de la metodología en su trabajo diario? (Si su respuesta es No, continuar en la pregunta 6)

   ( ) Sí

   ( ) No

5. ¿Qué tan frecuente aplica la metodología Scrum en su trabajo?

   ( ) Siempre

   ( ) A veces

   ( ) Rara vez

   ( ) Nunca

6. ¿Considera que la metodología Scrum es el marco de trabajo adecuado que el Área de Informática de Gestión necesita para el desarrollo de proyectos?

   ( ) Sí

   ( ) No

Información sobre el proceso de control de calidad

7. ¿Se aplica alguna metodología de control de calidad para los proyectos desarrollados bajo Scrum en el Área de Informática de Gestión?

   ( ) Sí

   ( ) No

( ) Desconoce

8. ¿Existe un proceso de planificación para llevar a cabo las tareas de control de calidad en los proyectos desarrollados bajo Scrum en el Área de Informática de Gestión?

   ( ) Sí

   ( ) No

   ( ) Desconoce

9. ¿Se aplican pruebas de software por parte del equipo de calidad en cada *sprint* que comprenda la liberación de un producto funcional en los proyectos desarrollados bajo Scrum en el Área de Informática de Gestión?

   ( ) Sí

   ( ) No

10. ¿Conoce los tipos de pruebas de software que aplica el equipo de calidad en los *sprints* del proyecto desarrollado bajo Scrum en el Área de Informática de Gestión? (Si su respuesta es No, continuar con la pregunta 12)

    ( ) Sí

    ( ) No

11. Indique cuáles tipos de pruebas son realizadas por el área de calidad (Puede marcar más de una opción)

    ( ) Pruebas de interfaz

    ( ) Pruebas de rendimiento

    ( ) Pruebas de carga

    ( ) Pruebas de estrés

    ( ) Pruebas de usabilidad

    ( ) Pruebas de compatibilidad

    ( ) Pruebas funcionales

    ( ) Pruebas de accesibilidad

( ) Pruebas de seguridad

( ) Pruebas de regresión

( ) Pruebas de integración

( ) Pruebas de estructura del software (arquitectura)

12. ¿Considera que el proceso de control de calidad actual se encuentra alineado con la metodología de desarrollo ágil Scrum?

( ) Sí

( ) No

13. ¿Cómo considera la fiabilidad del proceso de control de calidad del software en los desarrollos bajo la metodología Scrum?

( ) Malo

( ) Regular

( ) Bueno

( ) Muy bueno

14. ¿Cómo considera la eficiencia del proceso de control de calidad del software en los desarrollos bajo la metodología Scrum?

( ) Malo

( ) Regular

( ) Bueno

( ) Muy bueno

15. ¿Qué tipos de pruebas de software considera que el área de calidad debe fortalecer? (Puede marcar más de una opción)

( ) Pruebas de interfaz

( ) Pruebas de rendimiento

( ) Pruebas de carga

( ) Pruebas de estrés

( ) Pruebas de usabilidad

( ) Pruebas de compatibilidad

( ) Pruebas funcionales

( ) Pruebas de accesibilidad

( ) Pruebas de seguridad

( ) Pruebas de regresión

( ) Pruebas de integración

( ) Pruebas de estructura del software (arquitectura)

16. ¿Cómo considera el proceso de comunicación entre los desarrolladores y el equipo de calidad en los proyectos desarrollados bajo Scrum del área de Informática de Gestión?

( ) Malo

( ) Regular

( ) Bueno

( ) Muy bueno

17. ¿Cómo califica el reporte de defectos por parte del equipo de calidad, tomando en cuenta que los defectos deben ser completos, concisos, precisos y objetivos?

( ) Malo

( ) Regular

( ) Bueno

( ) Muy bueno

Anexo 5: Encuesta para el coordinador de calidad

Encuesta # 2

Para el coordinador de calidad

La siguiente encuesta forma parte de los instrumentos de investigación del proyecto final llamado “Propuesta de una metodología de control de calidad del software adaptada a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el Área de informática de Gestión del Poder Judicial, II Circuito Judicial de San José en el III Cuatrimestre de 2017”, elaborado por la estudiante Aida Elena Siles Rojas para optar por el Grado de Licenciatura en Sistemas de Información de la Universidad Hispanoamericana. El objetivo de esta encuesta es identificar la situación actual del proceso de control de calidad llevado a cabo en los desarrollos bajo la metodología Scrum en el Área de Informática de Gestión del Poder Judicial de Costa Rica.

Información general de la persona encuestada

1. ¿Cuál es su grado académico?

( ) Bachiller

( ) Licenciatura

( ) Maestría

2. Marque el rango de años que corresponda a su experiencia laboral en el área de software:

( ) Menos de un año

( ) 1 a 3 años

( ) 3 a 5 años

( ) 5 a 7 años

( ) Más de 7 años

Información general sobre la metodología de desarrollo

3. ¿Conoce la metodología de desarrollo Scrum? (Si su respuesta es No, continuar con la pregunta 6)

( ) Sí

( ) No

4. ¿Hace uso de la metodología en su trabajo diario? (Si su respuesta es No, continuar en la pregunta 6)

( ) Sí

( ) No

5. ¿Qué tan frecuente aplica la metodología Scrum en su trabajo?

( ) Siempre

( ) A veces

( ) Rara vez

( ) Nunca

6. ¿Considera que la metodología Scrum es el marco de trabajo adecuado que el Área de Informática de Gestión necesita para el desarrollo de proyectos?

( ) Sí

( ) No

7. ¿Conoce bien sus roles y responsabilidades dentro de cada proyecto desarrollado bajo la metodología Scrum?

( ) Sí

( ) No

Información sobre el proceso de control de calidad

8. ¿Se aplica alguna metodología de control de calidad para los proyectos desarrollados bajo Scrum en el Área de Informática de Gestión?

( ) Sí

( ) No

9. ¿Existe un proceso de planificación para llevar a cabo las tareas de control de calidad en los proyectos desarrollados bajo Scrum en el Área de Informática de Gestión?

( ) Sí

( ) No

( ) Desconoce

10. ¿Para qué tipos de proyectos se realiza el proceso de control de calidad en el Área de Informática de Gestión? (Puede marcar más de una opción)

( ) Aplicaciones Web

( ) Aplicaciones Cliente/Servidor

( ) Aplicaciones Móviles

( ) Aplicaciones Middleware

( ) Aplicaciones Standalone

( ) Aplicaciones de Inteligencia Negocios

11. ¿Existe comunicación entre el equipo de calidad y el equipo que planifica el proyecto bajo la metodología de Scrum?

( ) Sí

( ) No

12. ¿Se realiza una planeación de pruebas que incluya la estrategia, técnicas, tipos de pruebas, herramientas y la estimación del esfuerzo para las pruebas?

( ) Sí

( ) No

13. ¿Se definen casos de prueba por parte del equipo de calidad en los proyectos desarrollados bajo Scrum del área de Informática de Gestión?

( ) Sí

( ) No

14. ¿Se aplican pruebas de software por parte del equipo de calidad en cada *sprint* que comprenda la liberación de un producto funcional en los proyectos desarrollados bajo Scrum en el Área de Informática de Gestión?

( ) Sí

( ) No

15. Indique cuáles tipos de pruebas son realizadas por el área de calidad. (Puede marcar más de una opción)

( ) Pruebas de interfaz

( ) Pruebas de rendimiento

( ) Pruebas de carga

( ) Pruebas de estrés

( ) Pruebas de usabilidad

( ) Pruebas de compatibilidad

( ) Pruebas funcionales

( ) Pruebas de accesibilidad

( ) Pruebas de seguridad

( ) Pruebas de regresión

( ) Pruebas de integración

( ) Pruebas de estructura del software (arquitectura)

16. ¿Considera que el proceso de control de calidad actual se encuentra alineado con la metodología de desarrollo ágil Scrum?

( ) Sí

( ) No

17. ¿Qué tipos de pruebas de software considera que el área de calidad debe fortalecer? (Puede marcar más de una opción)

( ) Pruebas de interfaz

( ) Pruebas de rendimiento

( ) Pruebas de carga

( ) Pruebas de estrés

( ) Pruebas de usabilidad

( ) Pruebas de compatibilidad

( ) Pruebas funcionales

( ) Pruebas de accesibilidad

( ) Pruebas de seguridad

( ) Pruebas de regresión

( ) Pruebas de integración

( ) Pruebas de estructura del software (arquitectura)

18. ¿Cómo considera el proceso de comunicación entre los desarrolladores y el equipo de calidad en los proyectos desarrollados bajo Scrum del área de Informática de Gestión?

( ) Malo

( ) Regular

( ) Bueno

( ) Muy bueno

19. ¿El equipo de calidad tiene contacto con los usuarios de la institución?

( ) Sí

( ) No

20. ¿Qué tan frecuente hace uso de la herramienta Microsoft Test Manager durante el proceso de control de calidad bajo el desarrollo con Scrum?

( ) Siempre

( ) A veces

( ) Rara vez

( ) Nunca

21. ¿Se utiliza algún sistema de información para el registro de las pruebas?

( ) Sí

( ) No

22. ¿Existe un proceso de validación del trabajo realizado por el equipo de calidad?

( ) Sí

( ) No

Anexo 6: Preguntas entrevista al jefe del área

Entrevista # 1

Para la Jefatura del Área de Informática de Gestión

La siguiente entrevista forma parte de los instrumentos de investigación del proyecto final llamado “Propuesta de una metodología de control de calidad del software adaptada a la metodología de desarrollo Scrum para mejorar el proceso de pruebas en el Área de informática de Gestión del Poder Judicial, II Circuito Judicial de San José en el III Cuatrimestre de 2017”, elaborado por la estudiante Aida Elena Siles Rojas para optar por el Grado de Licenciatura en Sistemas de Información de la Universidad Hispanoamericana. El objetivo de esta encuesta es identificar la situación actual del proceso de control de calidad llevado a cabo en los desarrollos bajo la metodología Scrum en el Área de Informática de Gestión del Poder Judicial de Costa Rica.

Para la Jefatura

1. ¿Por qué se creó el área de calidad de software?

2. ¿Existe un documento oficial por parte del departamento de TI del Poder Judicial sobre la metodología de desarrollo Scrum?

3. ¿Considera que la metodología Scrum es el marco de trabajo adecuado que el Área de Informática de Gestión necesita para el desarrollo de proyectos?

4. ¿Qué problemas detecta a nivel de QA en el proyecto SIAGPJ?

5. ¿Qué aspectos considera que debe mejorarse en el área de calidad?

6. ¿Qué beneficios se espera con la implementación de una metodología para el área de calidad?

7. ¿Qué se espera a largo plazo del área de calidad?

8. ¿Hay procesos de priorización que determinan cuál proyecto debe probarse?

9. ¿Conoce la relación de costos entre las actividades de control de calidad versus tareas de desarrollo?

Anexo 7: Respuestas de la entrevista realizada al jefe del área

Pregunta 1: ¿Por qué se creó el área de calidad de software?

De acuerdo con lo indicado por el jefe del área, se creó debido a que en producción se estaban generando muchos errores, generando insatisfacción en los usuarios cada vez que algo se actualizaba en producción y afectando la imagen de la DTIC.

Pregunta 2: ¿Existe un documento oficial por parte del departamento de TI del Poder Judicial sobre la metodología Scrum?

Actualmente no. La metodología Scrum inició como un plan piloto con el inicio del desarrollo del proyecto SIAGPJ, en el cual se documentó el uso de esta metodología en el estudio de factibilidad del proyecto como tal.

Pregunta 3: ¿Considera que la metodología Scrum es el marco de trabajo adecuado que el Área de Informática de Gestión necesita para el desarrollo de proyectos?

El Jefe del área considera que si es el marco de trabajo adecuado, ya que se tiene una mejor estimación de los tiempos, la productividad del equipo ha mejorado se tiene un mejor control sobre el día a día del equipo.

Pregunta 4: ¿Qué problemas detecta a nivel de QA en el proyecto SIAGPJ?

Uno de los principales problemas que menciona el jefe del área, es el seguir recibiendo mejoras de otros sistemas, a pesar de que el proyecto SIAGPJ es de prioridad alta en ocasiones no se puede dedicar los recursos de calidad a un 100%.

Otro de los problemas es el no tener integrado el proceso de calidad con el de desarrollo en cada *sprint* como lo indica Scrum, esto genera que al área de calidad le lleguen módulos o

componentes muy grandes cuando ya se tiene toda una funcionalidad completa, dejando por último el proceso de calidad donde se detectan bastantes errores/ defectos que al final generan que el proceso de liberación del producto se vuelva más lento debido a las constantes pruebas de confirmación para verificar que los defectos hayan sido corregidos. En otras palabras, el tener un proceso de calidad que funciona de forma correctiva en lugar de preventiva es uno de los problemas.

Pregunta 5: ¿Qué aspectos considera que debe mejorarse en el área de calidad?

Que el equipo tenga más iniciativa, más propuestas que ayuden a mejorar el área, el software, etc. Otro aspecto es el tema de las pruebas de rendimiento y carga, se debe mejorar en ese tema.

Pregunta 6: ¿Qué beneficios se espera con la implementación de una metodología para el área de calidad?

Tener una mejor gestión del proceso de calidad, incorporar la aplicación de pruebas de carga y rendimiento, mejorar las pruebas de seguridad, mejores procedimientos de *testing* que se adapten a Scrum.

Preguntar 7: ¿Qué se espera a largo plazo del área de calidad?

Que sea un área integral, es decir, que todo lo que se genera en esta área pase por el proceso de calidad. Que sea un área más consolidada con un integrante más en el área.

Pregunta 8: ¿Hay procesos de priorización que determinan cuál proyecto debe probarse?

De acuerdo con el jefe del área si existen, las prioridades se ven con las comisiones de los diferentes ámbitos del Poder Judicial, pero es el jefe del área junto con la coordinadora de

desarrollo quienes establecen las prioridades a todos los equipos del área de Informática de Gestión.

Pregunta 9: ¿Conoce la relación de costos entre las actividades de control de calidad versus tareas de desarrollo?

El costo se maneja a nivel general, por lo que actualmente no se puede identificar si los procesos de calidad son caros o no, sin embargo, el jefe si ve la importancia de conocer los costos a nivel de solo calidad, y nos indica que en este proyecto se sugiera en la parte de las recomendaciones, el tema de los costos de calidad en un proyecto.

Anexo 8: Plantilla plan de pruebas alto nivel

<table>
<tr><th colspan="2">Plan de Pruebas</th></tr>
<tr><td>Identificador:
<Identificador único de plan de pruebas></td><td>Fecha de elaboración:
<Fecha de creación del plan de pruebas></td></tr>
<tr><td colspan="2">Nombre del proyecto:
<Indicar el nombre del proyecto al que pertenece el plan de pruebas></td></tr>
<tr><td colspan="2">Descripción:
<Descripción breve acerca del plan de pruebas y su objetivo para con el proyecto></td></tr>
<tr><td colspan="2">Recurso humano asignado al proyecto y su rol respectivo:
<Se indican los nombres de las personas asignadas al proyecto para realizar las labores de control de calidad así como su rol dentro del proyecto y el tiempo>
<table>
<tr><th>Nombre</th><th>Rol</th><th>Tiempo %</th></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td colspan="2">Estrategia y enfoque de pruebas:
<Indicar la estrategia de pruebas a utilizar para las posteriores iteraciones en el proyecto. Determinar si es preventiva o reactiva, y definir el enfoque del proceso de testing></td></tr>
<tr><td colspan="2">Niveles de pruebas: <Indicar los niveles de pruebas>
<table>
<tr><td>☐Pruebas unitarias</td><td>☐Pruebas de sistema</td></tr>
<tr><td>☐Pruebas de integración</td><td>☐Pruebas de aceptación</td></tr>
</table></td></tr>
<tr><td colspan="2">Tipos de pruebas:
<table>
<tr><td>☐Pruebas funcionales</td><td>☐Pruebas de usabilidad</td></tr>
<tr><td>☐Pruebas de rendimiento</td><td>☐Pruebas de accesibilidad</td></tr>
<tr><td>☐Pruebas de carga</td><td>☐Pruebas de seguridad</td></tr>
<tr><td>☐Pruebas de estrés</td><td>☐Pruebas de Interfaz</td></tr>
<tr><td>☐Pruebas de compatibilidad</td><td>☐Pruebas de documentación</td></tr>
</table></td></tr>
</table>

☐Pruebas de portabilidad	☐Pruebas de confirmación
☐Pruebas unitarias	☐Pruebas de regresión

Alcance (Elementos a probar):

<Descripción de los elementos o módulos del sistema a probar a nivel general>

Entregable	Descripción

Definición de métricas:

<Establecer las métricas que se utilizaran en el proyecto, con el fin de monitorear y controlar el trabajo realizado por el equipo de calidad>

Ambiente de pruebas:

<Definir el ambiente de pruebas, establecer las características y requisitos>

Herramientas a utilizar:

<Listar las herramientas que utilizará el equipo de calidad en el proyecto>

Aprobado por:		
Fecha aprobación:	Haga clic aquí para escribir una fecha.	

Fuente: Elaboración propia, 2018

Anexo 9: Plantilla de verificación de historias de usuario

Verificación de historias de usuario	
Historia de usuario ID:	
Descripción: <Descripción de la historia de usuario>	
La historia de usuario no cumple con la(s) siguiente(s) característica(s):	

Característica	**Observación**
☐Independiente	
☐Negociable	
☐Valiosa	
☐Estimable	
☐Pequeña	
☐Testeable	

Nombre del tester:	
Fecha de verificación:	Haga clic aquí para escribir una fecha.
Aprobado por:	
Fecha Aprobación:	Haga clic aquí para escribir una fecha.

Fuente: Elaboración propia, 2018

Anexo 10: Plantilla plan de pruebas del sprint

Plan de Pruebas del Sprint #	
Identificador: <Identificador único de plan de pruebas>	**Fecha de elaboración:** <Fecha de creación del plan de pruebas>
Nombre del proyecto: <Indicar el nombre del proyecto al que pertenece el plan de pruebas>	
Descripción: <Descripción breve acerca del plan de pruebas y su objetivo para el sprint correspondiente>	

Niveles de pruebas: <Indicar los niveles de pruebas>

☐Pruebas unitarias	☐Pruebas de sistema
☐Pruebas de integración	☐Pruebas de aceptación

Tipos de pruebas:

☐Pruebas funcionales	☐Pruebas de usabilidad
☐Pruebas de rendimiento	☐Pruebas de accesibilidad
☐Pruebas de carga	☐Pruebas de seguridad
☐Pruebas de estrés	☐Pruebas de Interfaz
☐Pruebas de compatibilidad	☐Pruebas de documentación
☐Pruebas de portabilidad	☐Pruebas de confirmación
	☐Pruebas de regresión

Alcance (Elementos a probar):

<Descripción de los elementos o módulos del sistema a probar>

Entregable	Descripción

Ambiente de pruebas:

<Definir si para el *sprint* correspondiente se requiere adicionar elementos al entorno de pruebas.

Hardware, software, espacio físico, etc.>		
Aprobado por:		
Fecha aprobación:	Haga clic aquí para escribir una fecha.	

Fuente: Elaboración propia, 2018.

Anexo 11: Plantilla de evaluación de riesgos de las historias de usuario del sprint

Evaluación de Riesgos Historias de Usuario Sprint # **Informática de Gestión – Área de Calidad y Testing**					
Id Historia Usuario	**Descripción historia de usuario**	**Descripción Riesgo**	**Impacto**	**Probabilidad**	**Riesgo**
#####					

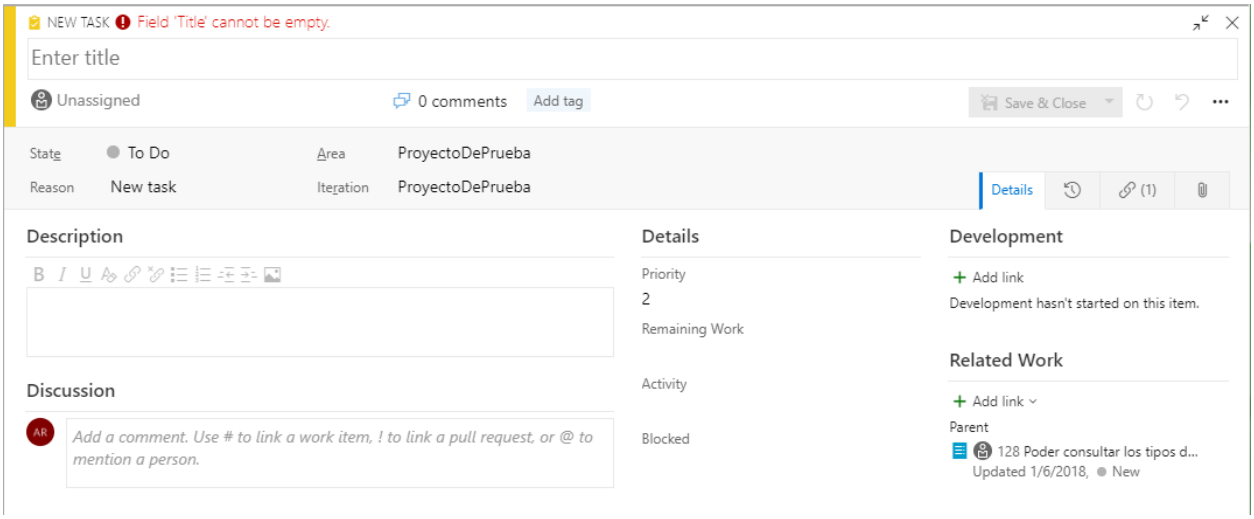
Valores de Impacto y Probabilidad

Impacto	Descripción		Probabilidad	Descripción
1	Bajo impacto		1	Baja probabilidad
2	Medio Impacto		2	Media probabilidad
3	Alto impacto		3	Alta probabilidad

Fecha de elaboración:	Haga clic aquí para escribir una fecha.			
Elaborado por:				

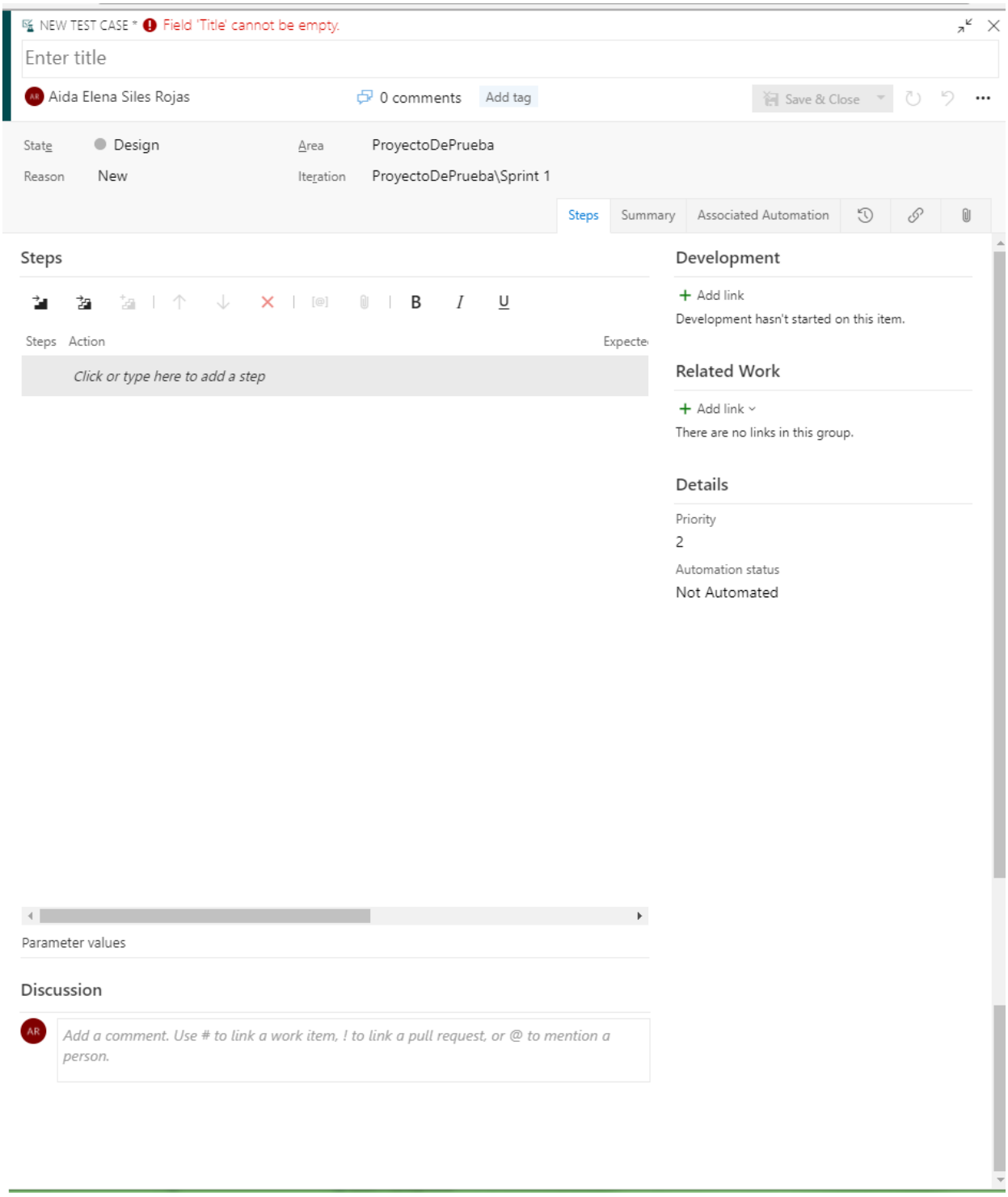
Fuente: Elaboración propia, 2018.

Anexo 12: Plantilla nueva tarea de la herramienta Visual Studio Team Services



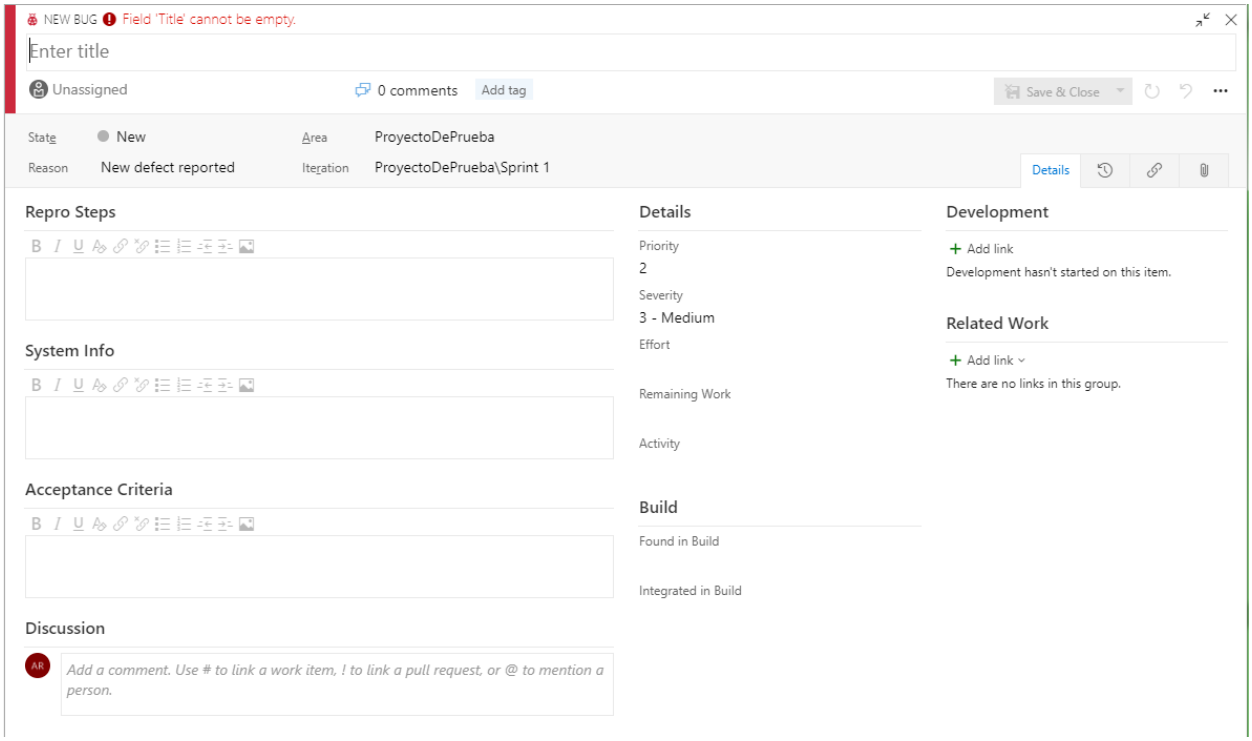
Fuente: www.visualstudio.com

Anexo 13: Plantilla nuevo caso de prueba de la herramienta Visual Studio Team Services



Fuente: www.visualstudio.com

Anexo 14: Plantilla creación de nuevo defecto (bug)



Fuente: www.visualstudio.com

Anexo 15: Listas de chequeo para los diferentes tipos de pruebas

Tabla 26: Lista de chequeo para las pruebas de instalación

Pruebas de Instalación		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
5. Se verifica la publicación del *deploy* o *release* en los servidores de pruebas.	☐	☐
6. Se verifica la instalación de los componentes en Windows 10 x64 de escritorio y de *Tablet*.	☐	☐
7. Se verifica la instalación y actualización de los servicios web API.	☐	☐
8. Se verifica que la documentación de los detalles de implementación y el proceso de instalación sean consistentes entre sí.	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 27: Lista de chequeo para las pruebas funcionales

Pruebas Funcionales		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
5. Se verifica la funcionalidad de cada requerimiento y que cumpla con el criterio de aceptación.	☐	☐
6. Se verifica que los mensajes de advertencia, error, y de confirmación de acciones concuerden con la funcionalidad del sistema.	☐	☐
7. Se verifica la funcionalidad en diferentes contextos del sistema.	☐	☐
8. Se verifica la funcionalidad con diferentes roles del sistema.	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 28: Lista de chequeo para las pruebas de rendimiento

Pruebas Rendimiento		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica el tiempo de respuesta de cada funcionalidad.	☐	☐
2. Se verifica el tiempo de respuesta del servicio API.	☐	☐
3. Se verifica el consumo de bytes de cada solicitud.	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 29: Lista de chequeo para las pruebas de carga

Pruebas de Carga		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica la funcionalidad del sistema bajo diferentes cargas de usuarios concurrentes.	☐	☐
2. Se verifica que el comportamiento del CPU no exceda el 95%.	☐	☐
3. Se verifica que el comportamiento de la memoria no exceda el 95%.	☐	☐
4. Se verifica el comportamiento del sistema durante la ejecución de pruebas de carga de 1000 usuarios virtuales constantes.	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 30: Lista de chequeo para las pruebas de estrés

Pruebas de Estrés		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica el comportamiento del sistema bajo una carga más allá de su nivel de aceptación.	☐	☐
2. Se verifica el uso de la memoria de los servidores.	☐	☐
3. Se verifica el uso del CPU de los servidores.	☐	☐
4. Se verifica la cantidad de usuarios concurrentes soportados por la aplicación.	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 31: Lista de chequeo para las pruebas de compatibilidad

Pruebas de Compatibilidad		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica la funcionalidad del sistema en Windows 10 x64 Escritorio.	☐	☐
2. Se verifica la funcionalidad del sistema en Windows 10 x 64 Tablet.	☐	☐
3. Se verifica la funcionalidad del sistema utilizando diferentes navegadores. Internet Explorer 11, Chrome, Mozilla Firefox.	☐	☐
4. Se verifica la compatibilidad del sistema con la base de datos SQL SERVER 2012.	☐	☐
5. Se verifica la compatibilidad del sistema con versiones anteriores.	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 32: Lista de chequeo para las pruebas de portabilidad

Pruebas de Portabilidad		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica que el sistema sea fácil de instalar en otros sistemas operativos.	☐	☐
2. Se verifica el comportamiento del sistema en otros sistemas operativos.	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 33: Lista de chequeo para las pruebas de usabilidad

Pruebas de Usabilidad		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
9. Se verifica que la funcionalidad sea fácil de aprender y de utilizar.	☐	☐
10. Se verifica que exista orden tabular entre los campos para facilidad de navegación.	☐	☐
11. Se verifica que exista un manejo de colores adecuado entre los diferentes controles y objetos del sistema.	☐	☐
12. Se verifica que la lectura de los títulos, botones, tooltip, y demás controles no sea difícil.	☐	☐
13. Se verifica que los campos obligatorios se distingan.	☐	☐
14. Se verifica que los campos y controles del sistema muestren la descripción emergente (tooltip) al posicionar el cursor sobre los objetos.	☐	☐
15. Se verifica que las diferentes pantallas del sistema brinden un entorno agradable y se facilite el entendimiento de la información presentada.	☐	☐
16. Se verifica que los campos obligatorios cuenten con mensajes de advertencia.		

Fuente: Elaboración propia, 2018.

Tabla 34: Lista de chequeo para las pruebas de accesibilidad

Pruebas de Accesibilidad		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica que la aplicación proporcione equivalente de teclado para todas las operaciones que se realizan con el mouse.	☐	☐
2. Es fácil de operar y entender el uso de la aplicación utilizando la documentación proporcionada en el paquete.	☐	☐
3. Se verifica que las imágenes e iconos sean usados apropiadamente, de manera que sea entendible para los usuarios.	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 35: Lista de chequeo para las pruebas de seguridad

Pruebas de Seguridad		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Se verifica el proceso de autenticación de usuarios del sistema.	☐	☐
2. Se verifica que la información proporcionada al usuario sea la correcta y se encuentre actualizada.	☐	☐
3. Se verifica que la transferencia de información a otro sistema sea la correcta.	☐	☐
4. Se verifica que la información esté disponible a las personas autorizadas cuando sea requerida.	☐	☐
5. Verificar las opciones habilitadas dependiendo del tipo de usuario	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 36: Lista de chequeo para las pruebas de interfaz

Pruebas de Interfaz		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
4. Se verifica la navegación a través de los objetos de cada pantalla evaluada.	☐	☐
5. Se verifica la alineación de cada uno de los objetos evaluados de las pantallas del sistema.	☐	☐
6. Se verifica la ortografía de cada título, botón, tooltip y mensaje, entre otros.	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 37: Lista de chequeo para las pruebas de documentación

Pruebas de Documentación		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
3. Se verifica que la documentación de usuario y el comportamiento del sistema sean consistentes entre sí.	☐	☐
4. Se verifica que sea legible, que tenga buena redacción y que sea comprensible.	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 38: Lista de chequeo para las pruebas de confirmación

Pruebas de Confirmación		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
1. Los defectos reportados han sido corregidos y cumplan con el criterio de aceptación.	☐	☐

Fuente: Elaboración propia, 2018.

Tabla 39: Lista de chequeo para las pruebas de regresión

Pruebas de Regresión		
Aspectos a evaluar	Cumple con el requisito	
	Sí	No
2. Se verifica que el cambio realizado no haya afectado la funcionalidad existente.	☐	☐

Fuente: Elaboración propia, 2018.

Anexo 16: Plantilla reporte de pruebas realizadas de un sprint

Plantilla Reporte de Pruebas Realizadas Sprint #

Identificador:	**Fecha de elaboración:**
<Identificador único del reporte>	<Fecha de creación del reporte de pruebas>

Nombre del proyecto:

<Indicar el nombre del proyecto al que pertenece el reporte de resultados>

Integrante(s) del equipo de calidad:

<Nombre del(los) integrante(s) de calidad que realizaron las pruebas>

Descripción:

<Descripción breve acerca de los resultados de las pruebas del sprint>

Niveles de pruebas: <Indicar los niveles de pruebas realizadas>

☐Pruebas unitarias	☐Pruebas de sistema
☐Pruebas de integración	☐Pruebas de aceptación

Tipos de pruebas:

Tipos de pruebas	Prueba Satisfactoria			Observaciones
	Sí	No	N/A	
☐Pruebas de instalación	☐	☐	☐	
☐Pruebas funcionales	☐	☐	☐	
☐Pruebas de rendimiento	☐	☐	☐	
☐Pruebas de carga	☐	☐	☐	
☐Pruebas de estrés	☐	☐	☐	
☐Pruebas de compatibilidad	☐	☐	☐	
☐Pruebas de portabilidad	☐	☐	☐	
☐Pruebas de usabilidad	☐	☐	☐	
☐Pruebas de accesibilidad	☐	☐	☐	
☐Pruebas de seguridad	☐	☐	☐	
☐Pruebas de Interfaz	☐	☐	☐	
☐Pruebas de documentación	☐	☐	☐	
☐Pruebas de confirmación	☐	☐	☐	
☐Pruebas de regresión	☐	☐	☐	

Impedimentos durante el sprint:

<Indicar los impedimentos que se presentaron durante el sprint>

Nuevos defectos registrados:

<Pegar la consulta que genera el VSTS sobre los bugs creados en el sprint correspondiente>

Fuente: Elaboración propia, 2018.

Anexo 17: Plantilla retrospectiva del sprint

Anexo 18: Plantilla retrospectiva del sprint

Retrospectiva Sprint #			
Fecha:	Haga clic aquí para escribir una fecha.		
Proyecto:			
Participantes:			
Aspectos que salieron bien durante el sprint <Se listan los aspectos que salieron bien en el sprint, en cuanto a procesos, tecnología, requerimientos, gestión del proyecto, problemas de negocio>			
Aspectos que deben mejorarse para el siguiente sprint <Se listan los aspectos que deben mejorarse para futuros sprints, en cuanto a procesos, tecnología, requerimientos, gestión del proyecto, problemas de negocio>			
Elaborado por:			
Aprobado por:			

Fuente: Elaboración propia, 2018.