INSTITUTO TECNOLÓGICO DE COSTA RICA
ESCUELA DE COMPUTACIÓN
PROGRAMA DE MAESTRÍA EN COMPUTACIÓN

Mejora de la calidad basado en la gestión de la deuda técnica

Proyecto para optar por el grado de Maestría Profesional con énfasis en Sistemas de Información

Juan Daniel Sánchez Cambronero

Profesor Asesor: Jennier Solano

San José, Costa Rica
2017

Dedicatoria

El presente proyecto de graduación culmina una fase de mi vida que inició en el año 2008, la fase TEC. Ese año di el primer paso para lo que sería mi vida profesional y académica, no me arrepiento de haber dado ese paso, pero tampoco fue algo que haya hecho solo. Durante más de 8 años, hubo dos personas que siempre estuvieron a mi lado, me inculcaron a seguir mis ideales y me llevaron de la mano a través de los momentos más difíciles, mis padres. Agradezco a ellos, con todo mi corazón, el apoyo que me han brindado tanto físicamente como mentalmente. Han sido mis consejeros, mis sabios y mis orientadores.

Debo mencionar que el último año de esta etapa ha sido el más duro para mí. Hubo momentos en los que el trabajo disponía de mí el 100%, momentos en los que la única opción en mi cabeza era dejar la universidad y concentrarme en otras cosas. Sin embargo, a pesar de todas las dudas, cuestionamientos y altibajos de ánimo; una persona, en conjunto con mis padres, me mantuvo tranquilo, sereno y me devolvía al camino que yo mismo había estado forjando. Lucía, te agradezco desde el fondo de mi corazón el haberte conocido, pero sobretodo agradezco tu ayuda en los momentos en que más la necesite.

A ustedes tres les dedico este proyecto de graduación.

Agradecimientos

Agradezco a mi familia que ha sido un pilar para seguir adelante, mis primas, mi tía y mi abuela han sido las personas que, además de mis padres, han creído en mí en todo momento.

También agradezco a todos mis compañeros durante esta fase, aquellos que conocí durante el bachillerato y se mantuvo la amistad (Mariela y German), otros que estuvieron conmigo durante la maestría Beatriz y Gustavo, pero además agradezco una amistad que se mantuvo durante toda esta etapa, Soledad.

A los profesores les agradezco el conocimiento que me han brindado: Roberto, Ronald, Alejandro, Mauricio y también Jennier, que ha sido mi asesor durante la fase final, y la más importante a lo largo de mi carrera.

A todos los mencionados, y a muchas personas más, les agradezco todo lo que han hecho por mí para llegar hasta aquí.

Epígrafe

“Sólo hay dos medios de pagar las deudas: por el trabajo y por el ahorro.”

-Thomas Carlyle

Resumen

El presente documento trata sobre la mejora del proceso de desarrollo de software y la generación de un proyecto piloto en un grupo de desarrollo de software en la empresa Intel.
El proyecto tiene como objetivo integrar el manejo de deuda técnica a los procesos de desarrollo y verificación de calidad para fomentar el desarrollo de mejores productos.

Se presenta una introducción con la descripción de la empresa, el departamento, estado actual, el problema que se quiere enfrentar y los objetivos del proyecto.
A continuación, se presenta el marco teórico donde se detallan múltiples conceptos abarcando desde aspectos generales como las metodologías de desarrollo, hasta aspectos más específicos como la deuda técnica. Luego, se brinda el marco metodológico donde se define el tipo de investigación realizada y la obtención de datos para generar un diagnóstico, e identificar oportunidades de mejora con el fin de proponer recomendaciones, el plan piloto a ejecutar y la propuesta de aplicación.
Por último, se presenta el análisis de resultados, las conclusiones y recomendaciones.

APROBACIÓN DE PROYECTO FINAL

"Mejora de la calidad basado en la Gestión de la deuda Técnica"

TRIBUNAL EXAMINADOR

Dr. Jennier Solano Cordero.
Profesor Asesor

MSc. Freddy Ramírez Mora
Profesor Lector

Máster. Kattia Chavarría Carrillo
Profesor Externo

Dr. Roberto Cortés Morales
Coordinador del Programa
de Maestría en Computación

TEC | Tecnológico de Costa Rica
Maestría en Computación

Junio, 2017

Índice general

Índice de figuras

Índice de tablas

1 Introducción

Para lograr el éxito en una organización, los sistemas de información son un componente esencial que debe sobresalir en una serie de aspectos como seguridad, desempeño, modularidad, entre otros. El objetivo principal de dichos sistemas es habilitar a la organización para entregar productos que satisfagan sus necesidades con la máxima calidad aceptable; permitiendo a la compañía sobrevivir o sobresalir en la industria. Por lo tanto, tener sistemas que sean propensos a fallos o no demuestren un nivel de calidad aceptable para la industria, puede ocasionar serios problemas a la organización.

Dichos sistemas de información se desarrollan siguiendo un proceso de gestión de proyectos que ejecuten etapas como la recolección de requerimientos, el diseño, su implementación y finalice en la publicación y el soporte respectivo. La ejecución correcta de estas etapas permite a las organizaciones satisfacer las necesidades de sus clientes al ser ejecutadas de la mejor manera posible, pero no siempre se siguen las mejores prácticas generando sistemas defectuosos que llegan a complicar sus funciones como también las labores de soporte.

1.1 Descripción de la empresa

Intel es la creadora de la serie de microprocesadores x86, los cuales se pueden encontrar en la mayoría de las computadoras personales. Además de estos procesadores, Intel también fabrica placas madre, tarjetas de red, tarjetas de video, procesadores embebidos y otros dispositivos relacionados a comunicaciones y computación.
Fundada por Robert Noyce y Gordon Moore en 1968 bajo el nombre NM Electronics, iniciaron sus operaciones y tomaron el nombre actual (Intel, abreviatura de “Integrated Electronics”) un año después. Actualmente, el director ejecutivo es Brian Krzanich, y cuenta con más de 100.000 empleados distribuidos en 46 países por todo el mundo.

Intel divide sus organizaciones internamente por niveles, siendo el primer nivel la junta directiva de Intel. El segundo nivel se divide en 12 organizaciones, las cuales se pueden apreciar en el siguiente diagrama:

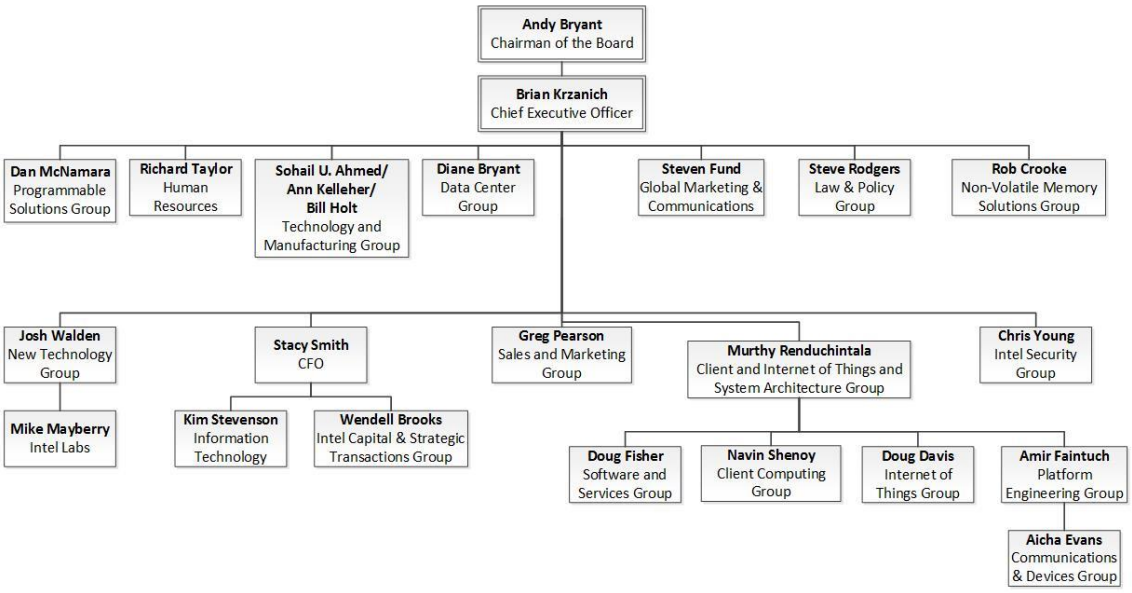

Figura 1- Organigrama de Intel. Fuente: publicado en la red de Intel

Cada una de las organizaciones al segundo nivel contiene una gran cantidad de sub-organizaciones de menor tamaño; por lo tanto para este proyecto se ha limitado el alcance a una organización específica llamada “Sistemas de calidad del negocio” o QBS por sus siglas en inglés, la cual se encuentra en el séptimo nivel bajo la estructura de la organización de segundo nivel llamada: “Grupo de Manufactura y Tecnología”.

QBS es una organización desarrolladora de sistemas de información de software que provee soporte a otras organizaciones dentro de Intel. Tiene más de 65 empleados directos y contratistas en Oregon, Folsom, Arizona y Costa Rica, con puestos como analistas de negocios, gestores de proyectos, ingenieros en sistemas, arquitectos empresariales, ingenieros en factores humanos, desarrolladores de software y administradores de base de datos.

Este grupo se rige bajo 4 objetivos estratégicos:

1. Satisfacer a los clientes
2. Asegurar el liderazgo en calidad al menor costo
3. Mejorar la velocidad en los métodos y sistemas de calidad
4. Construir una organización innovadora

Como misión, QBS busca entregar y manejar soluciones innovadoras automatizadas para la organización llamada “Red de Calidad Corporativa” (Organización padre de QBS en el cuarto nivel y cuyas siglas en inglés son CQN).

En el siguiente diagrama, se puede apreciar el modelo jerárquico de QBS:

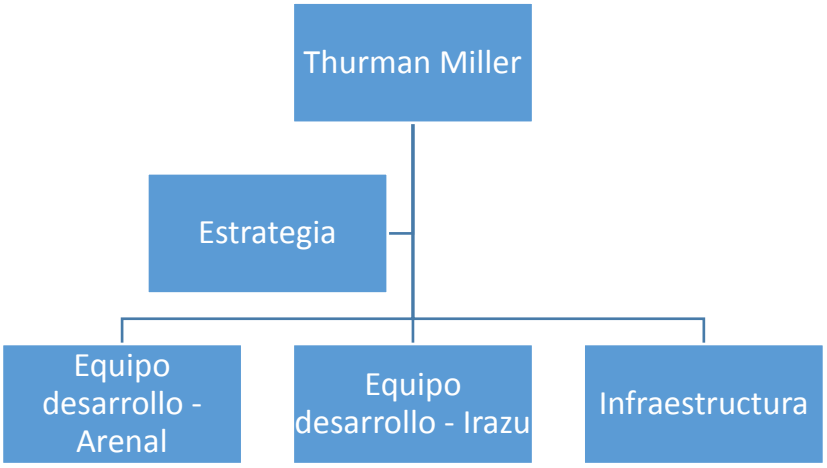

Figura 2 - Organigrama QBS. Fuente: Creación propia

QBS consta de 4 equipos principalmente y una figura de gerencia (Thurman Miller). El primer equipo en la jerarquía es el equipo de estrategia, conformado por los arquitectos de sistemas y analistas de información; este equipo busca liderar los esfuerzos técnicos mediante el establecimiento de las prácticas a seguir y apoyar el diseño de los sistemas desarrollados por QBS. El segundo equipo es el de infraestructura, conformado por todos los administradores de bases de datos e ingenieros de infraestructura; su principal objetivo es brindar todo el equipo necesario a los desarrolladores para ejecutar sus labores (servidores, bases de datos, entre otros). Por último, existen dos equipos de desarrolladores enfocados en la implementación de los sistemas.

El proyecto que se busca realizar será ejecutado dentro de esta última organización descrita, y se espera que impacte la totalidad de la organización.

Actualmente, QBS cuenta con un ciclo de vida de proyectos basado en el esquema de desarrollo en cascada, haciendo una excepción en la etapa de desarrollo donde se realiza una mezcla con las metodologías ágiles, particularmente Scrum.

El ciclo de vida se divide en 5 fases, iniciando en una fase de pre-exploración y finalizando con la fase de publicación. En el gráfico siguiente se puede observar cómo se divide dicho proceso de gestión:

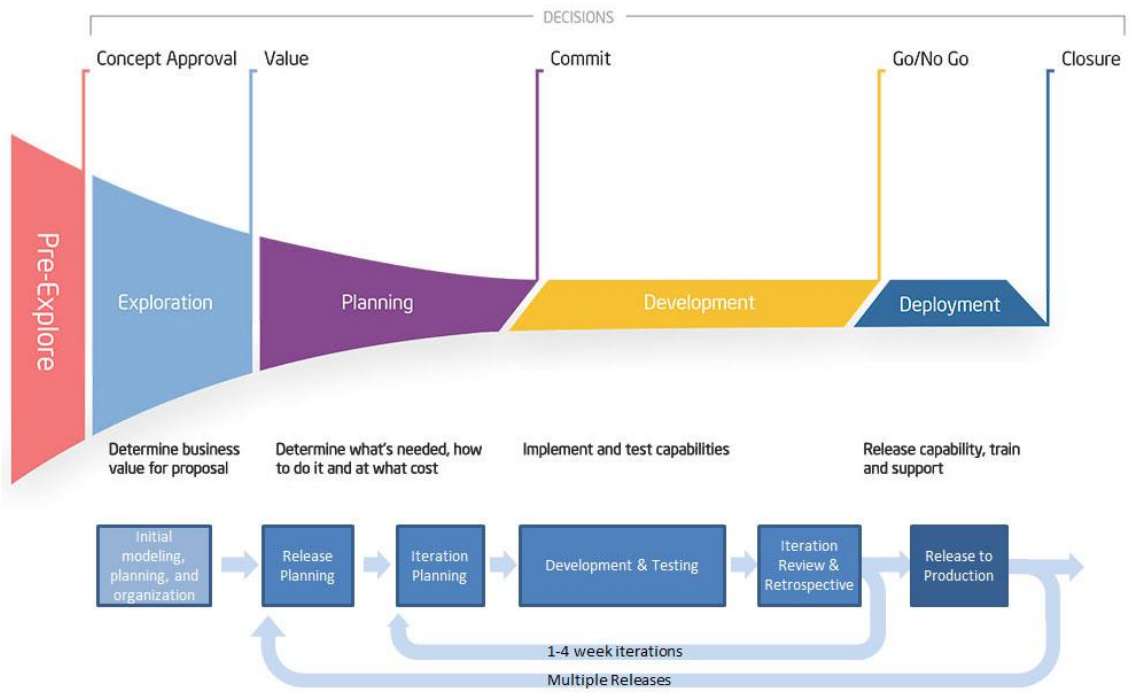

Figura 3 - Ciclo de vida de proyectos. Fuente: qbs.intel.com

Cada una de las fases mostradas en el gráfico anterior tiene un objetivo principal, estos son:

- Pre-exploración: se define el concepto y la viabilidad del proyecto.
- Exploración: se determina el valor a la organización y los beneficios esperados a brindar a los clientes.
- Planificación: se determinan los recursos a utilizar, los costos necesarios, las ganancias esperables y los requerimientos principales. Además se debe brindar un borrador de cómo se implementará la solución a partir de los requerimientos recolectados.
- Desarrollo: esta fase se desliga del modelo cascada y adopta una metodología ágil. Se utiliza una versión modificada de Scrum, adaptada a la organización para realizar iteraciones de desarrollo, que proveen a los usuarios productos parciales o completos para ser utilizados a modo de prueba o directamente en producción. En esta fase también se realizan todas las pruebas pertinentes para publicar el sistema desarrollado.

- Publicación: esta fase incluye todos los entrenamientos, creación de manuales de usuario y soporte necesario a realizar. El soporte es brindado a través de un esquema de tiquetes, en el que una persona solicita arreglar o agregar una funcionalidad, este tiquete es asignado a un desarrollador que se encuentre familiarizado con la herramienta, finalizando en la ejecución según los tiempos establecidos a partir de la prioridad que la solicitud tenga.

La organización anteriormente ha realizado esfuerzos por mejorar el nivel de calidad en las aplicaciones, entre ellos podemos nombrar la incorporación de tareas para las revisiones de código, la incorporación de una herramienta de análisis de código o la modificación del proceso de calidad para adaptarse al modelo de desarrollo utilizado. No obstante, debido a que el ambiente de desarrollo de software es variable y se mueve a una gran velocidad, las soluciones implementadas no han logrado mostrar resultados consistentes. Además, los esfuerzos mencionados se han limitado a una porción de las aplicaciones, por lo que no están observando todo el panorama de la organización.

Por estas razones, se han encontrado deficiencias que deben ser tratadas con el fin de evitar que empeoren y generen resultados negativos para la organización.

A continuación, se presenta una descripción del problema que expone el proceso que se desea mejorar.

1.2 Descripción del problema

El proceso de gestión de proyectos utilizado por la organización cumple el objetivo de crear herramientas que satisfagan las necesidades de los clientes. En su ejecución, se pasan por todas las etapas necesarias como recolectar requerimientos, verificar viabilidad, implementar el proyecto, realizar pruebas y publicar la solución. Sin embargo, a pesar de que el proceso seguido brinda buenos resultados, se han identificado problemas como:

- **Cantidad de defectos en sistemas de producción**: durante el primer semestre del año 2016 se recibieron 1186 tiquetes que fueron catalogados como defectos en las aplicaciones, o solicitudes de

configuración; por lo tanto, se necesita reducir dicho número para reducir el tiempo invertido en labores de soporte.

- **Incumplimiento del tiempo de respuesta en los acuerdos de nivel de servicio**: durante el primer semestre del año 2016, el promedio de respuesta para cerrar los tiquetes fue de 11 días y medio de trabajo. La organización cuenta con acuerdos de nivel de servicio, en los que se establece que los tiquetes de soporte deben ser cerrados en un plazo menor a 14 días laborales, basado en esto se encontró que 255 incumplían dicho acuerdo, es decir un 23% de los tiquetes manejados.

Con base en estos problemas, se ha establecido que una de las principales razones por la que se han manifestado es la falta de cumplimiento en los estándares de calidad. Esto ha generado que se invierta tiempo en entender el diseño y el funcionamiento de las soluciones, no solamente para proveer el soporte necesario, sino también para realizar modificaciones que permitan mejorar las funcionalidades existentes.

Una de las razones por las que ocurre esto es que existen aplicaciones desarrolladas por otros grupos de personas, las cuales no tenían conocimiento en el desarrollo de sistemas de información, generando así fallas en los estándares de calidad. Esto provoca que se invierta la mayor cantidad de tiempo en la etapa de soporte provocando un incremento de los costos de operación asociados.

Por esta razón, se ha determinado que se debe realizar un esfuerzo para mejorar las soluciones existentes, y verificar que cualquier nueva solución mantenga un nivel de calidad aceptable; para lo cual, se opta por utilizar una estrategia de manejo de deuda técnica.

Para entender el significado de deuda técnica es necesario aclarar que a lo largo del ciclo de vida del desarrollo de software, los desarrolladores pueden introducir código incompleto, propenso a errores o que es simplemente una solución temporal. Esto puede ser realizado involuntariamente, pero existen ocasiones donde el desarrollador realizó lo que consideró correcto en el momento debido a un sacrificio, por ejemplo, tiempo. Este tipo de código es conocido comúnmente como deuda técnica y, de igual manera que las deudas financieras, debe ser

manejado apropiadamente o de lo contrario puede comprometer severamente la calidad de los sistemas, el desempeño, la seguridad y la estabilidad [1].

Ward Cunningham fue el primero en acuñar el término de deuda técnica, y el manejo de la misma es un área de investigación relativamente nueva en la industria del desarrollo de sistemas [2]. Como toda deuda, esta debe ser pagada de una u otra forma para asegurar el estado correcto del producto, mas no se han diseñado estrategias en la industria para incorporar el manejo de la deuda técnica en el ciclo de vida del desarrollo de sistemas de información.

Para manejar la deuda técnica es necesario una herramienta que realice análisis del código de los sistemas de información, específicamente análisis de código estático [3] que proveen a los equipos de desarrollo todas las métricas necesarias para estudiar los proyectos. Al realizar esto, se permitirá la identificación de las áreas problemáticas y determinar los costos de resolver dichos problemas. Además, permitirá a los grupos de desarrollo alinear sus esfuerzos con las buenas prácticas establecidas por los arquitectos, garantizando la generación de productos con estándares de calidad aprobados por los ingenieros de calidad.

Actualmente existe una herramienta dentro de la organización que provee este tipo de análisis, pero las tendencias de desarrollo han cambiado y la herramienta ha quedado obsoleta, ya que no genera un análisis completo de las tecnologías que usa con mayor frecuencia la organización.

Realizar un proyecto que implemente el análisis de la deuda técnica, conlleva una adaptación del proceso de desarrollo actual. Este proceso utiliza una metodología ágil para ejecutar las actividades de desarrollo, y comúnmente se utilizan iteraciones de 2 semanas de trabajo, que incorporan un proceso de verificación de calidad por parte de los clientes y los aseguradores de calidad.

El proceso adaptado, además de realizar las mismas tareas que se ejecutan actualmente, deberá tomar en cuenta un flujo de actividades que no solamente realicen pruebas funcionales del sistema y verifiquen los resultados, sino también que el código desarrollado cumpla con una serie de métricas de calidad definidas por la organización y la industria.

1.3 Objetivo general

Proponer un modelo para el manejo de deuda técnica a los procesos de desarrollo y aseguramiento de la calidad en la organización QBS de Intel.

1.4 Objetivos específicos

- Analizar el estado actual de las soluciones basándose en los estándares estipulados según la nueva herramienta de análisis de código a implementar.
- Conceptualizar un nuevo modelo de procesos para el desarrollo de aplicaciones y su aseguramiento de la calidad al incorporar la gestión de la deuda técnica.
- Implementar un proyecto piloto sobre una aplicación, que permita analizar los efectos del manejo de la deuda técnica en un período no mayor a 16 semanas.
- Proponer un plan de implementación para incorporar las modificaciones al proceso de desarrollo actual, empezando en enero del 2017 y con una duración de 10 meses para aplicarse a las aplicaciones con mayor prioridad.

1.5 Alcance

El proyecto a realizar deberá ser finalizado en un periodo no mayor a 16 semanas, concluyendo en una presentación con los resultados obtenidos a partir de la implementación del proyecto piloto, y los cambios necesarios a realizar para incorporarlo a toda la organización de QBS.

Como proyecto piloto, se tomará una aplicación que se encuentre en proceso de finalización o soporte. Se espera comprobar que los beneficios obtenidos engloben aspectos como la calidad de código medido por la deuda técnica y el tiempo de respuesta y la cantidad de tiquetes en un sistema en producción.

El análisis a realizar sobre el proceso de desarrollo, implica el conocimiento de temas como el manejo de deuda técnica, la utilización de herramientas para su gestión, y también tener conocimiento sobre metodologías ágiles de desarrollo como Scrum.

Queda fuera del alcance del proyecto, la implementación a todas las aplicaciones existentes en la organización. Sin embargo, el plan de implementación deberá incorporar las aplicaciones que tengan la mayor necesidad de ser remediadas.

2 Marco Referencial

En el presente capítulo se definen los diferentes conceptos base para desarrollar la metodología a ejecutar en el departamento de desarrollo de software. Dichos conceptos abarcan temas tales como las metodologías de desarrollo, el aseguramiento de la calidad y la administración de procesos de negocio.

2.1 Metodologías de desarrollo

Una metodología, según la Real Academia Española, se define como un conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal [4]. Estas metodologías sirven como guía para la elaboración de un producto final, y en el área de desarrollo de software este producto final son los diferentes sistemas de información creados exclusivamente para satisfacer las necesidades de un grupo de usuarios.

En la actualidad, existen ciertas metodologías que son consideradas las más populares por su gran adopción en la industria. Entre ellas podemos mencionar dos que han sido las más utilizadas: cascada y las metodologías ágiles de desarrollo.

2.1.1 Cascada

El desarrollo con la metodología cascada organiza las principales actividades que componen la construcción de un sistema de información de manera secuencial, su fundamento se basa en que ninguna etapa puede iniciarse antes de que haya finalizado la anterior [5].

Se divide en 6 etapas, las cuales se enumeran seguidamente:

1. Análisis de requisitos: el usuario detalla las funcionalidades deseadas por el sistema, es necesario explicar todos los procesos y es obligatorio llegar a acuerdos para verificar el total entendimiento de lo esperado. En esta etapa los resultados serán los documentos oficiales con las especificaciones de los requerimientos.
2. Diseño del sistema: se dividen y organizan los componentes del sistema con el objetivo de construir el sistema por bloques. Además, se deben

analizar aspectos de arquitectura, seguridad, rendimiento, entre otros. En esta etapa el resultado es la generación de un documento de diseño con las principales pautas a seguir por los desarrolladores.

3. Codificación: se crea la solución y se obtendrá el código fuente del sistema.
4. Pruebas: se debe verificar y validar que cada componente funcione como es debido y cumpla las expectativas de los usuarios finales.
5. Implantación o liberación: se libera en el ambiente de producción para su ejercicio normal, es decir, ya los clientes reciben el sistema para su uso real.
6. Mantenimiento o soporte: en esta etapa se realizan mejoras, labores de mantenimiento y la corrección de defectos encontrados.

En este tipo de metodología de desarrollo, las pruebas se realizan una vez finalizada la etapa de implementación, acarreando problemas debido a que un error identificado podría requerir un rediseño de la solución o incluso levantamiento de nuevos requerimientos, retrasando la finalización del sistema.

2.1.2 Metodologías ágiles de desarrollo

Las metodologías ágiles de desarrollo son alternativas a la forma tradicional de desarrollar. El Manifiesto para el desarrollo ágil de software es una iniciativa surgida en el 2001 por un conjunto de consultores y desarrolladores, con el fin de promover el uso de metodologías ágiles. Dicho manifiesto especifica que se valora con mayor peso ciertas prácticas sobre otras, por ejemplo [6]:

- Individuos e interacciones sobre procesos y herramientas.
- Software funcional sobre documentación extensiva.
- Colaboración con el cliente sobre negación contractual.
- Respuesta ante el cambio sobre seguir el plan.

Las metodologías ágiles buscan realizar desarrollos más rápidos y que eviten elementos innecesarios que puedan retrasar los proyectos, tomando en cuenta la valoración del cliente y enfocando la colaboración para la obtención de los objetivos.

Algunos de los tipos de metodologías de desarrollo ágil son:

1. **Programación extrema (XP, “Extreme Programming”):**

Es una metodología que busca la simplicidad y los cambios son esperados durante la ejecución. Sus principales características son:

- Hacer pequeñas entregas
- Simplificar la etapa de diseño
- Probar sobre el camino
- Esperar cambios sobre el diseño
- Trabajo con el cliente constante

En el siguiente diagrama, se puede observar el flujo de ejecución bajo este esquema:

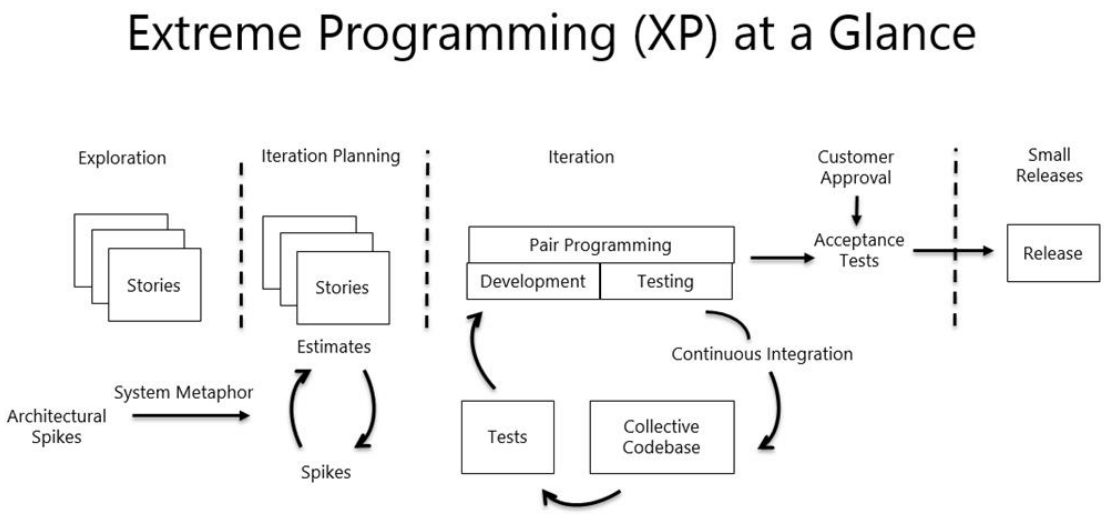

Figura 4 - Diagrama Flujo de trabajo Extreme Programming. Fuente: Blog J.D. Meiers

Se siguen 12 prácticas de desarrollo, estas son [7]:

- Seguir estándares de programación: los desarrolladores deben buscar el uso de las mejores prácticas de programación que la industria establezca, según el ambiente sobre el cual se trabaje.
- Dominio colectivo: el equipo es dueño de todo el producto y conoce todo el negocio para el cual se trabaja.
- Integración continua: todo el proceso de desarrollo se encuentra conectado y automatizado.

- Clientes en el sitio: se procura tener los clientes involucrados en la totalidad del proceso.
- Programación en parejas: los desarrolladores deben trabajar en parejas.
- Juego de planificación: esta se conoce como la etapa donde se realiza la planificación de la versión del proyecto que se desarrollará.
- Refactorización: proceso en el cual se mejora el código desarrollado para tener mejores resultados o eliminar errores pasados.
- Publicaciones cortas: el sistema se desarrolla en múltiples publicaciones cortas que provean valor al cliente, y se puedan ver resultados en un período corto de tiempo.
- Desarrollo simple: la simplicidad se busca en el desarrollo buscando el diseño más simple que cumpla con las necesidades planteadas.
- Ritmo sostenible: buscar mantener las 40 horas de desarrollo por recurso.
- Metáfora del sistema: el equipo debe buscar el uso de símiles para describir el sistema. Se busca relacionar conceptos de una manera diferente y simple.
- Desarrollo enfocado en pruebas: a diferencia de la tendencia regular por desarrollar primero las funcionalidades y realizar las pruebas posteriormente; XP busca primero la creación de los casos de prueba o pruebas unitarias previo al desarrollo del sistema.

2. Scrum:

Scrum es una forma de trabajo aplicada por equipos de alto desempeño. Se utilizan una serie de buenas prácticas que tienen como objetivo el desarrollo de soluciones de software de manera ágil, realizando entregas parciales pero funcionales para el cliente.

Estas entregas son priorizadas de acuerdo a las necesidades de los usuarios, facilitando su adopción en entornos muy cambiantes o poco claros, debido a que se pueden obtener resultados rápidamente.

Scrum se basa en iteraciones cortas (de 15 a 30 días), denominadas sprints, y fijas que generan un resultado de utilidad para el usuario, lo cual se logra al involucrar al usuario con la priorización de los requerimientos y con la distribución de la carga de trabajo en las planificaciones de las iteraciones.

Cada día se realiza una reunión de sincronización, con una duración de alrededor de 15 minutos. Durante estas sesiones cada miembro del equipo informa su avance o bloqueo, si se presenta, con esto todo el equipo se informa de manera directa y general del avance del proyecto.
En una reunión de sincronización cada miembro del equipo debe responder tres preguntas:

- ¿Qué he hecho desde la última reunión de sincronización?
- ¿Qué voy a hacer a partir de este momento?
- ¿Qué impedimentos tengo o voy a tener?

Scrum se caracteriza por poseer roles definidos, los cuales son fundamentales para la aplicación del método de trabajo. Dentro de estos roles se puede mencionar:

- Dueño del producto: Ocupa el puesto principal, y representa a todas las personas interesadas (consumidores finales del producto). Esta persona, o grupo, es la que define los objetivos a cumplir.
- Maestro Scrum: facilita las reuniones de la metodología y valida que se ejecuten de la manera correcta.
- Miembros del equipo: Grupo responsable de ejecutar el proyecto y desarrollar el producto.
- Interesados: personas que hacen posible el proyecto y para quienes se producirá el beneficio acordado.

Finalmente, existe otra sesión de trabajo denominada inspección y adaptación, la cual consiste en una sesión al final de cada iteración dividida en dos partes:

- Demostración: El equipo de desarrollo muestra el resultado de la iteración al usuario, quien realiza las observaciones pertinentes. El producto debe ser fácilmente liberado en producción.
- Retrospectiva: Se revisa la forma en la que se trabajó durante la iteración, así como los problemas que se presentaron, para que se procure eliminarlos.

En la siguiente figura, se puede observar todo el flujo de trabajo en esta metodología:

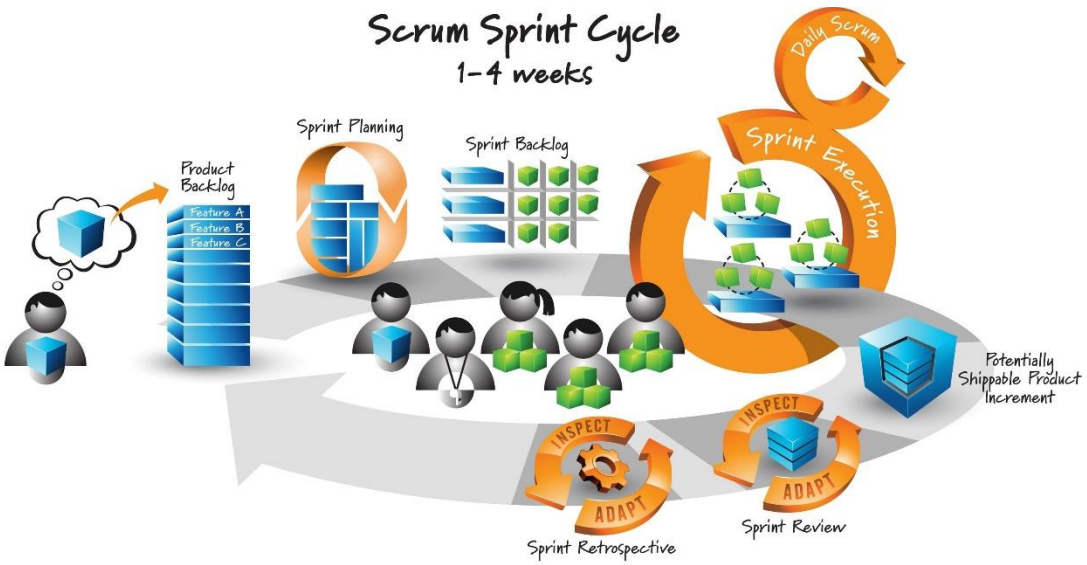

Figura 5 - Diagrama de flujo de Scrum. Fuente: Springtimesoft [8]

2.2 Calidad del software

La calidad del software es un concepto que engloba la validación de un producto a través de un proceso que verifique su funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.

A lo largo de la historia se han dado grandes contribuciones en este campo, entre ellas podemos nombras las siguientes [9]:

1. **Walter Shewhart:** considerado el fundador del proceso de control estadístico, desarrolló un conjunto de gráficos utilizados para el control de procesos donde se establecen ciertos límites para controlar el desempeño de un proceso específico. Posteriormente, sus contribuciones fueron incorporadas como parte del desarrollo de software con los modelos de madurez (CMM por sus siglas en inglés, “capability maturity model”).
2. **Edwards Demming:** escritor del libro “Out of the crisis”, donde se exponen 14 principios para transformar el proceso de una organización para enfocarse en la calidad y sus clientes.
3. **Joseph Juran:** define la calidad como la aptitud para el uso y propone que los problemas de calidad son responsabilidad directa de las áreas administrativas. También creó un programa de 10 pasos para la mejora de la calidad en una organización.

4. **Watts Humphrey:** considerado como el padre de la calidad del software. Dedicó la mayor parte de su carrera en solucionar los problemas como los retrasos, excesos en los costos y la mejora de calidad y la productividad. Su mayor contribución fue la creación del programa del proceso de software en el instituto de ingeniería de software, lo que posteriormente lideró en la creación de los modelos de madurez (CMM).

2.2.1 Aseguramiento de la calidad

En el ámbito del desarrollo de productos y servicios, el proceso de aseguramiento de la calidad es todo proceso sistemático que verifique si un producto o servicio cumple con los requerimientos establecidos. A pesar de que los pasos en el proceso de aseguramiento puedan cambiar, ya que dependen de los requerimientos y los objetivos de las compañías o los productos en desarrollo, el principal objetivo se mantiene [10].

El proceso de aseguramiento de la calidad es importante, y debe ser aplicado en todas las fases del ciclo de vida de un producto, siendo el control de calidad el método más común para el aseguramiento de la calidad. Implementar este proceso incrementará la confianza de los clientes y la credibilidad de la compañía, lo que mejora los procesos de trabajo y su eficiencia [11].

Los principales beneficios del aseguramiento de la calidad incluyen:

1. Brindar visibilidad de los procesos seguidos y los productos producidos por la organización a los cargos administrativos.
2. Asegurar la efectividad de los procesos.
3. Verificar que se cumplan los estándares definidos por la industria y que sean seguidos por la organización.

2.2.2 Pruebas de calidad del software

Una de las principales actividades del aseguramiento de calidad, es la ejecución de las pruebas del software. Esta actividad puede considerarse como constructiva, puesto que verifica que las funcionalidades se ejecuten correctamente, sin embargo, también es una actividad destructiva ya que su objetivo es encontrar errores antes de que el producto sea liberado.

Además, la norma ISO 29119 define las pruebas de software como [12]:

"…*las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada o stakeholder*."

Conforme avanza el área de desarrollo de software, las pruebas o el testeo también adquieren especializaciones para cada una de las funciones que se estén desarrollando. Esto ha generado que se creen categorías como las siguientes:

2.2.3 Pruebas unitarias

Este tipo de pruebas son ejecutadas por los desarrolladores y su objetivo principal es verificar que cada componente o módulo (métodos, funciones, objetos, entre otros) funcione correctamente e independientemente del resto de los componentes.

Una prueba unitaria debe cumplir con ciertas características para ser considerada dentro de esta categoría. Estas características son:

- Automatizable: deben poder ser ejecutadas sin la necesidad de la intervención humana, posibilitando la ejecución automática.
- Repetibles o reutilizables: deben ser ágiles y rápidas, y deben permitir al desarrollador ejecutarlas tantas veces como sea necesario.
- Completas: deben abarcar la mayor cantidad de líneas de código posible, con el fin de probar de forma independiente todo el programa.
- Independientes: no se deben de ver afectadas por la ejecución de otras pruebas, debe poseer autonomía de ejecución. En el caso de que una prueba requiera de otras dependencias, estas deben ser simuladas para no alterar su ejecución.

2.2.4 Pruebas de integración

La IEEE (Instituto de Ingeniería Eléctrica y Electrónica por sus siglas en inglés) define las pruebas de integración como [13]:

"*las pruebas donde los componentes de software, componentes de hardware o ambos son combinados y probados para evaluar la interacción entre ellos*"

Las pruebas de integración son realizadas después de que las pruebas unitarias hayan concluido exitosamente. Estas pruebas corresponden a tomar un grupo de componentes relacionados entre sí y verificar su correcta funcionalidad, con el principal objetivo de verificar la funcionalidad de varios componentes que se comunican o interactúan con datos en común.

2.2.5 Pruebas funcionales

Estas pruebas de software tienen por objetivo validar que los sistemas cumplan con las funciones específicas para los cuales han sido creados, y son ejecutadas por los ingenieros de calidad en conjunto con los usuarios finales.

Esta modalidad de pruebas se ejecutan manualmente, sin embargo, existen herramientas que proveen la capacidad de automatizar este tipo de pruebas para controlar la ejecución de las mismas y comparar los resultados obtenidos con los esperados. Además, la automatización de las pruebas reduce el tiempo de mano de obra invertida, agilizando el proceso de verificación de calidad.

Las pruebas automatizadas se pueden abarcar con dos enfoques: pruebas manejadas por el código (o pruebas unitarias, las cuales ya se han descrito anteriormente), y pruebas de interfaz de usuario donde se graba una secuencia de operaciones y se repiten automáticamente.

Entre las principales herramientas para la automatización de pruebas se pueden destacar las siguientes:

- **Selenium:** esta herramienta open-source gratuita proporciona un entorno de pruebas para aplicaciones basadas en web. Selenium provee una herramienta para grabar y reproducir interacciones y de esta forma realizar las pruebas correspondientes [14].
- **HP QuickTest Professional:** actualmente conocido como Unified Functional Testing, proporciona la capacidad de automatizar pruebas funcionales y pruebas de regresión para software y ambientes de prueba. Su modalidad de uso es bajo licencias vendidas por Hewlett-Packard.

2.2.6 Desarrollo guiado por pruebas (TDD)

TDD es un enfoque para el desarrollo de sistemas de información, que consiste en desarrollar primero el código que pruebe un requerimiento deseado del sistema, antes del código que implementa su funcionalidad [15].

El desarrollo guiado por pruebas busca convertir los requerimientos del cliente en casos de prueba, asegurando que cuando se presente al usuario el producto, éste haya sido probado completamente. Se establece una metodología de trabajo secuencial, que brinda orden a las actividades que se realizan, de esta forma se crea el denominado ritmo TDD, el cual consiste en los siguientes pasos de manera simplificada:

1. Seleccionar el requerimiento del usuario.
2. Escribir una prueba que demuestre la necesidad de escribir código.
3. Escribir el mínimo código para que la prueba compile.
4. Implementar exclusivamente la funcionalidad demandada por las pruebas.
5. Ejecución de la prueba en la medida de lo posible de forma automatizada.
6. Mejorar el código sin añadir funcionalidad.
7. Volver al primer paso.

2.2.7 Estándares de calidad

En la industria también se han manifestado algunos estándares para la verificación de la calidad, entre ellos podemos encontrar:

2.2.7.1 Estándar IEEE 730-1984

Este estándar fue publicado en 1984 y su principal propósito es proveer requerimientos uniformes y aceptables para la preparación de un plan de aseguramiento de calidad en las etapas de iniciación, planificación, control y ejecución. Se enfoca en el desarrollo y mantenimiento de sistemas de software y ha obtenido múltiples revisiones desde su publicación, actualmente la versión en vigencia es la 730-2014 [16].

El proceso de aseguramiento de calidad, en conjunto con el proceso de desarrollo, como se mencionó anteriormente, varía de acuerdo al contexto organizacional en el que se encuentre. Por lo tanto, es necesario utilizar de herramientas que permitan gestionar dichos procesos, controlarlos y aplicarles mejoras.

2.2.7.2 Estándar ISO/IEC 25000: Requerimientos de calidad y evaluación

El estándar 25000 provee una guía para el uso de una nueva serie de estándares internacionales conocidos como SQUARE (por sus siglas en inglés, Systems and software Quality Requirements and Evaluation) [17].

Su objetivo principal es organizar y centralizar otros dos procesos de calidad en un solo marco de referencia, estos procesos son la especificación de requisitos de calidad del software y la evaluación de la calidad del software.

Este estándar tiene 6 divisiones, estas son:

1. **ISO/IEC 25000n:** la división de la gestión de calidad donde se explican los principales conceptos básicos de SQUARE.
2. **ISO/IEC 25001n:** la división del modelo de calidad y se presentan características de un proceso de calidad con requisitos internos como externos.
3. **ISO/IEC 25002n:** la división de mediciones de calidad, donde se detallan ciertas definiciones matemáticas aplicadas a métricas y una guía práctica para su correcta aplicación.
4. **ISO/IEC 25003n:** la división de requisitos de calidad, donde se explica cómo se deben detallar los requisitos de calidad y los principales componentes que deben ser detallados para ser utilizados posteriormente en la realización de pruebas.
5. **ISO/IEC 25004n:** la división de la evaluación de la calidad, donde se proveen guías para la evaluación correcta de la calidad de un producto. También se detallan las diferentes responsabilidades de la verificación de la calidad para los distintos roles como: desarrolladores, clientes y evaluadores.
6. **ISO/IEC 25050-25099:** son extensiones de las bases de SQUARE.

2.2.7.3 Estándar ISO/IEC 29119 Pruebas de software

ISO/IEC/IEEE 29119 Pruebas de software, es un conjunto de estándares internacionalmente acordados para las pruebas de software, uno de sus objetivos es que pueda ser utilizado con cualquier metodología o ciclo de vida de desarrollo de software [12].

El estándar ISO/IEC/IEEE 29119 se compone de 5 estándares específicos, los cuales se mencionan seguidamente:

- **ISO/IEC 29119-1:** se refiere a los conceptos y definiciones básicas que permiten comprender los otros cuatro estándares. Fue publicada en setiembre del 2013.
- **ISO/IEC 29119-2:** publicado en setiembre del 2013, el objetivo de este estándar es definir un modelo de pruebas que pueda ser utilizado con cualquier metodología de desarrollo de software.
- **ISO/IEC 29119-3:** Publicado también en setiembre del 2013, busca definir las plantillas para la documentación de las pruebas que abarquen todo el ciclo de vida de las pruebas.
- **ISO/IEC 29119-4:** publicado en diciembre del 2014, tiene como objetivo cubrir el desarrollo de un método aceptado internacionalmente para las técnicas del diseño y planeamiento de las pruebas a realizar.
- **ISO/IEC 29119-5:** Este estándar aún no se ha publicado oficialmente y se encuentra en desarrollo, su objetivo será definir un vocabulario estándar que permita generar casos de prueba con un lenguaje fácilmente comprensible, lo que facilita su mantenimiento y automatización.

La verificación de la calidad en el software provee beneficios a todos los involucrados en un proceso de desarrollo, pero no existen procesos infalibles ya que la incorporación de otros factores puede influir en la ejecución de dichos procesos. Es en estos casos donde se puede incurrir en la incorporación de fallos o malas prácticas, un concepto también conocido como deuda técnica.

2.2.8 Deuda técnica

En 1992, Ward Cunningham acuñó el término de deuda técnica [18]. Este concepto engloba una dicotomía en la toma de decisiones cuando se enfocan los resultados en soluciones a largo plazo con buena calidad, contra soluciones de corto plazo pero que reducen el tiempo al mercado e incrementan el valor del negocio.

En el artículo escrito por Cunningham, se hace referencia a la acumulación de la deuda por las desviaciones que se hacen de los diseños por múltiples factores, y la principal solución para disminuir dicha deuda es mediante la refactorización

de los sistemas. Estas decisiones de diseño, una vez que se han detectado como deuda técnica, es necesario registrarlas y seguirlas para evitar que sigan produciendo mayores consecuencias. Al igual que su contraparte financiera: la deuda es aceptable siempre y cuando se busque la manera de pagarla.

En el ambiente ágil, uno de los principales problemas que se presenta es la verificación y validación de métodos para el manejo de la deuda técnica; su aplicación tradicional puede conllevar a costos altos o tomar más tiempo, el cual no se tenía previsto, a pesar de que en los últimos años se han realizado esfuerzos por incorporar en los procesos de desarrollo el manejo de la deuda técnica. Durante el 2014, un grupo de investigadores decidieron realizar una simulación de manera que se pueda determinar e implementar la mejor estrategia de gestión de deuda técnica en un ambiente ágil real [19]. En dicha investigación, se presentan distintos métodos de gestión pero se evalúan solamente los siguientes:

- **Listado de deuda técnica:** en este modelo solamente se lleva un listado de los elementos de deuda técnica encontrados y se busca su corrección por sprint (un sprint dedicado a solucionar la deuda técnica) o de manera porcentual (se reduce un porcentaje de deuda durante el sprint y el tiempo restante es utilizado para incorporar nuevos requerimientos).
- **Listado de deuda técnica con manejo automatizado:** en este modelo no solamente se lleva un listado de los elementos encontrados sino que su monitorización se realiza de manera automatizada. Además, su remediación se hace de igual manera que el modelo anterior.
- **Manejo de deuda técnica por umbrales de aceptación:** en este modelo se mide la deuda técnica de manera automatizada pero solamente es solucionada cuando se ha llegado a un límite predefinido.

El proceso de simulación generó más de 8000 replicaciones y determinó que de los modelos definidos anteriormente, el mejor es el listado de deuda técnica con manejo automatizado, y siguiendo una estrategia de remediación por sprint. En los resultados se aclara que, aunque esta sea la mejor solución para el manejo de la deuda técnica, en efectos prácticos puede que no sea aplicable debido a que existen factores como el tiempo y los recursos disponibles para realizar dicha estrategia.

Las herramientas utilizadas para el manejo de deuda técnica ejecutan una revisión conocida como análisis estático de código fuente [20]. En esta revisión se buscan vulnerabilidades mediante el uso de técnicas como los análisis de flujo de datos. Algunos ejemplos de herramientas son:

- **SonarQube:** es una plataforma de código libre utilizada para la evaluación del código fuente de las aplicaciones. Creada por la empresa SonarSource, cuyas oficinas principales se encuentran ubicadas en Suiza. En el anexo 1 se podrá encontrar una explicación detallada de las principales características, pero entre ellas podemos encontrar:
    - Indicadores de desempeño para niveles administrativos.
    - Soporte de múltiples lenguajes como c#, java, php, entre otros.
    - Integración con herramientas de integración continúa.
    - Posee más de 60 extensiones para complementar los análisis básicos que provee la herramienta.
    - Provee una lista de más de 100 métricas para la evaluación de la deuda técnica [21].
- **Checkmarx SAST:** checkmarx es una empresa dedicada a la elaboración de soluciones enfocadas en identificar y corregir vulnerabilidades de seguridad en aplicaciones web y móviles. Es una herramienta comercial y tiene un costo para su uso. Entre sus principales características podemos encontrar:
    - Soporte de más de 20 lenguajes de programación y no se debe realizar ninguna configuración para escanear cada uno de dichos lenguajes.
    - Utiliza estándares de seguridad como OWASP 10 o SANS 25.
    - Algoritmos de remediación automáticos.
    - Fácil de utilizar e incorporar en las organizaciones.

Desde un punto de vista administrativo, utilizar herramientas como las mencionadas anteriormente se consideran aceptables si implican un beneficio económico. Por ende, es necesario incorporar al manejo de la deuda técnica ciertas variables como [22]:

- ¿Cuánta deuda técnica es aceptable?
- ¿Es el momento correcto para reducir la deuda técnica?

- ¿Se tienen los recursos necesarios para remediar los problemas?

Determinar las respuestas a las preguntas anteriores permitirá brindar una idea más clara a la organización de cómo ejecutar las tareas de remediación. Sin embargo, una vez determinado que si se pueden realizar los esfuerzos, se tienen otras barreras. Por ejemplo, en Scrum existen factores que no son tomados en cuenta dentro de la metodología como [23]:

1. ¿Quién es el responsable de la deuda técnica: el equipo de desarrollo, el "Scrum Master" o el dueño del producto?
2. Es común que el dueño del producto no entienda los beneficios y la necesidad de reducir la deuda técnica.
3. Los problemas y metas en relación a la deuda técnica no se encuentran estructurados ni documentados.

2.3 Administración de procesos de negocios (BPM)

La administración de procesos de negocio o BPM, por sus siglas en inglés, es definida por Jhon Jeston como una disciplina enfocada en la utilización de los procesos de las organizaciones para lograr sus objetivos, además, se aplican mejoras que buscan obtener el mejor desempeño continuo. Sin embargo, esta definición se complementa con lo propuesto por la organización Gartner, donde BPM no se limita a buscar mejoras, sino que además busca modelar el mejor proceso, analizar su comportamiento y medir resultados para poder llegar a la optimización más efectiva [24] [25].

La implementación correcta de un proceso en una organización depende de factores como la cultura organizacional, la madurez de los procesos, la comunicación entre todos los niveles jerárquicos, entre otros; pero de todos, el más importante es tener definido una estrategia empresarial que asegure la ejecución de los procesos hacia un fin específico. Utilizar BPM proporciona una serie de metodologías y resultados que brindan una serie de lineamientos para ejecutar el trabajo, y cómo alinear dichas tareas hacia la estrategia definida. Para ello, es necesario definir los objetivos a aplicar mediante BPM, donde Susan Page define 3 características, estas son [26]:

- **Efectividad:** enfocado en los clientes y la manera en la que se les provee valor.

- **Eficiencia:** enfocado en los encargados de los procesos y la mejor forma en que puedan ejecutar sus labores.
- **Adaptabilidad:** enfocado solamente en evaluar las más sencillas soluciones para adaptarse a las necesidades del negocio.

BPM permite gestionar y sostener los principales aspectos enlazados al mejoramiento de la organización, además establece una priorización de procesos con el fin de determinar cuáles son aquellos con mejores posibilidades de mejora. Entre sus aspectos importantes, es necesario resaltar que BPM permite un rediseño de procesos, definiendo un punto de partida (conocido como situación actual), sobre el cual se realizarán todos los cambios pertinentes a las mejoras.

Esta disciplina ha comenzado a formar parte de muchas empresas en todo el mundo, y ha sido la base de éxito para las operaciones de estas empresas ya que se ha caracterizado por utilizar las mejores prácticas que la industria ha identificado, logrando así la reducción de los tiempos de ejecución y el manejo de excepciones con una mayor velocidad, resultando finalmente en la toma de mejores decisiones.

3 Metodología del proyecto

Para efectos de este proyecto, es vital trabajar con una metodología que defina los pasos a seguir y marque los objetivos para obtener los resultados deseados. Dicha metodología se basa en los temas planteados en el marco referencial, tomando como base los principales aspectos que se acoplan a los objetivos específicos.

3.1 Recopilación de información

Para el desarrollo del proyecto, se busca llevar a cabo una captura de datos; de forma tal que se ha establecido seguir una estrategia de observación para los datos cualitativos, como se menciona en el libro “Metodología de la investigación” [27]. Para efectos de este proyecto, este tipo de estrategia busca:

1. Comprender los procesos
2. Describir las actividades que se desarrollan
3. Identificar problemas

Entre los datos cualitativos obtenidos se encuentran las actividades desarrolladas en los procesos de desarrollo, los roles involucrados, estimaciones de tiempo, entre otros factores.

Para conseguir esto, y respaldando la observación realizada en la organización, se realizaron entrevistas a 6 integrantes del departamento para obtener dicha información. Como base para dichos cuestionarios, se utiliza la siguiente plantilla:

Nombre:			
Profesión:			
Fecha de entrevista:			

Pregunta 1:	
Respuesta:	
Pregunta 2:	
Respuesta:	

Tabla 1 - Plantilla para cuestionario. Fuente: creación propia

Para la recolección de los datos cuantitativos, se utilizó una herramienta de automatización de manejo de deuda técnica, en este caso SonarQube. Y se tomaron las siguientes métricas como relevantes para la ejecución del proyecto [21]:

- **Complejidad ciclomática:** se calcula como el número de posibles caminos en el código fuente. Toda función tiene un mínimo de complejidad de 1.
- **Bloques duplicados:** número de bloques de líneas duplicados.
- Archivos duplicados: número de archivos involucrados en los bloques duplicados.
- **Densidad de líneas duplicadas:** porcentaje de la duplicación en todo el código.
- **Complejidad promedio por función:** complejidad ciclomática promedio por función en el proyecto.
- **Proporción de deuda técnica:** es la proporción entre el costo de desarrollar una línea de código y el costo para corregir el error que se encuentre. El valor base para desarrollar una línea de código es de 1.4 horas.
- **Calificación de mantenibilidad:** calificación obtenida a partir de la proporción de la deuda técnica. La escala se divide de la siguiente manera:

Calificación	Porcentaje deuda técnica
A	<5%
B	>6% - <10%

C	>11% - <20%
D	>21% - <50%
E	>50%

Tabla 2 - Calificación deuda técnica según Sonarqube. Fuente: Sonarqube

- **Defectos:** número de defectos
- **Días de deuda técnica:** la cantidad de días establecida por la herramienta para solucionar todos los problemas encontrados.
- **Vulnerabilidades:** número de vulnerabilidades de seguridad
- **Líneas de código:** número de líneas de código

Una vez finalizada la recopilación de información, se realiza un análisis del estado actual de las soluciones y el proceso. Según menciona Hernández [27], la idea central de capturar datos es poder guiar la toma de decisiones para generar cambios, para ello se utiliza la metodología definida por Susan Page para la administración de procesos de negocio [26].

3.2 Análisis del proceso de desarrollo

Las compañías están constantemente buscando como hacer sus negocios más simples y eficientes, sin embargo la mayoría busca un ente externo para ayudar con dicho proceso.

Actualmente existen en el mercado diferentes metodologías que pueden ser utilizadas para la mejora de procesos de negocios. Entre estas podemos encontrar [28]:

- **Six Sigma**: es un método estadístico que busca reducir la variación en el desempeño de un proceso. Su nombre hace referencia a los 6 niveles de desviaciones estándar que pueden haber en los tiempos de entrega en un proceso.
- **10 pasos para la mejora de procesos**: metodología creada por Susan Page que divide el proceso de mejora en 10 pasos y busca entender los procesos de una organización y eliminar todas aquellas actividades que no son necesarias.

En la tabla siguiente, se presenta una comparación de las principales características que ambas metodologías presentan:

	Six Sigma	10 pasos para la mejora de procesos
Enfoque	Se enfoca en una estrategia analítica para la generación de un beneficio económico	Se enfoca en la automatización y optimización de la mejora de un proceso
Datos utilizados	Utiliza análisis estadísticos como claves para determinar las oportunidades de mejora	Utiliza la información recolectada a través de otros sistemas, la organización y los empleados para determinar las oportunidades de mejora
Diseño de la mejora del proceso	Las mejoras sobre los procesos son obtenidas a través de análisis de causa-efecto. Se utilizan todas las áreas de la organización para obtener el mayor beneficio.	Provee un diseño visual del proceso y a través de los flujos e interacciones del proceso se determinan las actividades de mejora utilizando las 6 técnicas de mejora.
Mediciones	Se utilizan controles predefinidos de las métricas claves utilizadas.	Se establecen los controles de mejora continua por cada proceso mejorado.

Tabla 3 - Comparación de metodologías. Fuente: creación propia

Entre las dos, se ha optado por usar la metodología de Susan Page ya que presenta un enfoque flexible y adaptable de cómo mejorar un proceso, lo que permite su pronta aplicación en el contexto de este proyecto. La metodología utiliza una serie de pasos simples que no requieren incurrir en gastos por obtener un consultor, lo cual se adapta a las necesidades del proyecto puesto que no se posee del capital económico para realizarla. Paralelamente, al ser los mismos empleados de la organización los que trabajan con la metodología, permite que dichos empleados obtengan un mejor entendimiento de cómo funciona el proceso de inicio a fin. Por último, se ha escogido la metodología de Susan Page

por la familiaridad y la sólida base adquirida con esta a través de los cursos de formación en la maestría.

Otras metodologías como “Six Sigma”, al ser más comprensivas en su enfoque, requieren procesos más formales de entrenamiento, certificación y aplicación de sus herramientas, lo cual puede en ocasiones incrementar costos, causar retrasos y reducir la creatividad. Esto hace que sean más aplicables a procesos más grandes y/o complejos en los que la inversión económica y de tiempo se vea más justificada. También, Six Sigma se enfoca en cómo reducir el tiempo de entrega de un producto al cliente final, puede crear rigidez y burocracia en el proceso, lo cual no se adapta al contexto del proyecto ya que se necesita la flexibilidad de poder adaptarse a cualquier equipo de desarrollo y sistema que se necesite crear [29].

La metodología de Susan Page utiliza 10 pasos para rediseñar los procesos. Como se mencionó anteriormente, Page se centra en tres aspectos principales: efectividad, eficiencia y adaptabilidad.

Su principal característica es su enfoque en el desarrollo del rediseño y análisis de todos los pasos que se trabajan, y no presenta grupos de trabajo como otras metodologías.

En la siguiente figura se representan los 10 pasos que Susan Page define:

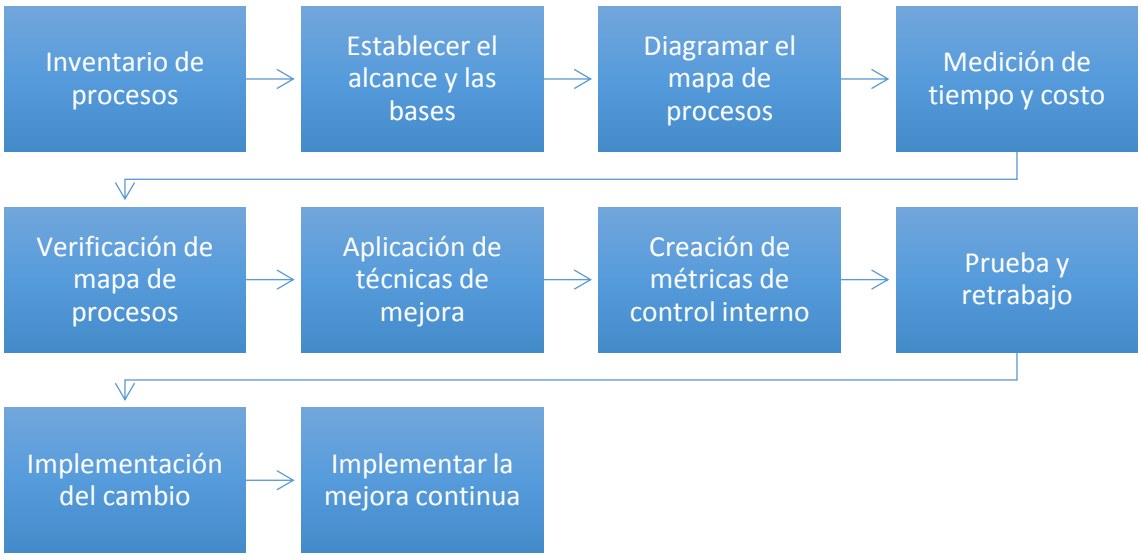

Figura 6 - Pasos de Susan Page. Fuente: creación propia

A continuación se detallan los pasos de la metodología:

1. **Desarrollo del inventario de procesos:** En este paso se crea una hoja de ruta donde se detalla el estado actual de los procesos existentes. Es necesario priorizar los procesos para determinar el nivel de importancia entre ellos. Algunos criterios que Page identifica para determinar la prioridad son: impacto, implementación, estado actual y el valor del proceso; cada uno de estos criterios responde a una serie de preguntas definidas.
2. **Establecer el alcance y las bases:** En este paso se define qué proceso se rediseñará, determinando a su vez el alcance sobre el cual se realizarán las modificaciones.
3. **Diagramar mapa de procesos actuales:** se realiza un diagrama de proceso para clarificar el funcionamiento y qué elementos se encuentran involucrados.
4. **Medir tiempo estimado y costo:** Se identifican los tiempos y el costo usando los recursos y herramientas disponibles obteniendo como resultado datos cuantitativos.
5. **Verificar mapa de procesos:** se revisa el proceso ubicando elementos importantes como usuarios, roles y funciones primordiales. Es necesario el involucramiento de todos los interesados y los patrocinadores.
6. **Aplicar técnicas de mejora:** en esta fase, se aplican técnicas para analizar los procesos desde la perspectiva de los clientes externos e internos. Los procesos analizados se evalúan con las siguientes características:
    a. Análisis de burocracia
    b. Valor agregado al cliente
    c. Duplicidad de información
    d. Simplicidad del proceso
    e. Ciclo de tiempo
    f. Automatización
7. **Creación de controles internos y métricas:** se plantean controles que respalden la efectividad y la eficiencia mediante el control de la información y los resultados.

8. **Prueba y retrabajo:** Se crea un plan de pruebas que verifique el funcionamiento que se busca. Los datos obtenidos por estas pruebas permiten optimizar el proceso para garantizar con certeza que el proceso rediseñado cumple con todos los requerimientos establecidos.
9. **Implementación del cambio:** en esta fase se introduce el nuevo proceso a la organización, tomando en cuenta temas como la comunicación y el entrenamiento para todos los recursos.
10. **Implementación de la mejora continua:** se debe mantener evolucionando el proceso y adaptándolo a las nuevas necesidades que se manifiesten en el tiempo.

Como base para el análisis del proceso se toman los pasos definidos por Susan Page, donde se parte del hecho de que se conoce el estado actual del proceso de desarrollo de Software, por lo que es necesario desarrollar un inventario de procesos, establecer las bases del proceso y diagramar el proceso según su estado.

Posteriormente, se realizan evaluaciones de calidad al proceso para determinar cuáles son las mejores técnicas de mejora aplicables, y se finaliza con el rediseño del proceso tomando todos los datos anteriormente evaluados.

Se presenta además, un plan piloto que toma como insumo el diseño del proceso y se aplica a un caso de estudio por un período de 5 semanas, finalizando con la recolección de los datos obtenidos a partir de su implementación.

En el siguiente diagrama, se puede apreciar el flujo completo de los pasos a seguir durante esta metodología:

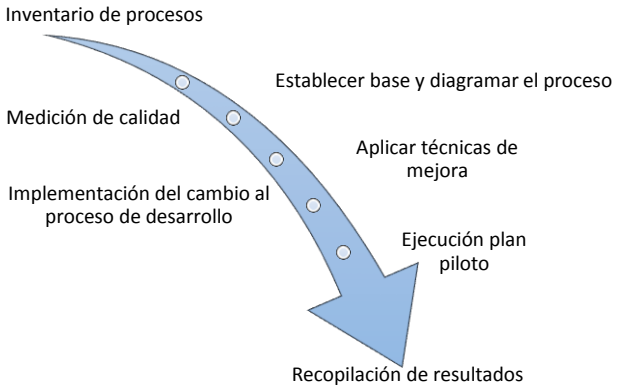

Figura 7 - Metodología utilizada. Fuente: creación propia

A continuación se detalla el flujo del diagrama anterior.

3.2.1 Realizar el inventario de procesos

En la primera etapa de la Metodología, es importante poder conocer con lo que se va trabajar y de esta forma tener una base que respalde el resto de pasos. Por ser el primer paso es muy importante que genere los insumos necesarios para sustentar el resto de las etapas.

Este inicio consta de 3 fases que según menciona Susan Page [26] crean la estructura para trabajar como punto de partida, estos son:

1. Crear el inventario de procesos.
2. Priorización de los procesos por criterio
3. Crear la tabla de inventario y criterios

Las 3 fases permiten priorizar las labores de mejora utilizando una herramienta provista por Susan Page llamada “Tabla de inventario de procesos”. Esta tabla tiene la siguiente forma:

Proceso	Criterio 1	Criterio 2
Proceso 1		
Proceso 2		

Tabla 4 - Inventario de procesos. Fuente: Susan Page

Los criterios que se presentan en la tabla pueden ser tantos como sean convenientes para buscar la solución al problema a resolver. Cada criterio debe poseer una escala de valores para poder dar los resultados del análisis de la mejor manera, además, los criterios deben poseer un valor de peso en una escala de 1 a 100 para poder realizar la priorización.

3.2.2 Establecer la base y diagramar el proceso

El objetivo de este paso es crear un diagrama de flujo que muestre el funcionamiento del proceso que va a ser rediseñado. Según Page, este diagrama es una representación de las actividades que proveen valor a la organización, por lo que se puede observar, de manera general, qué se debe mejorar y cómo se verán los posibles cambios.

3.2.3 Medición de calidad

La medición de la calidad en el proceso permite ubicar problemas y clasificarlos, de manera que se logre identificar la raíz del problema. Para medir los problemas de calidad se utilizan los diagramas de pescado; los cuales permiten analizar las causas de un problema y representarlas de forma clara. Un diagrama de pescado sigue la estructura planteada en la siguiente figura:

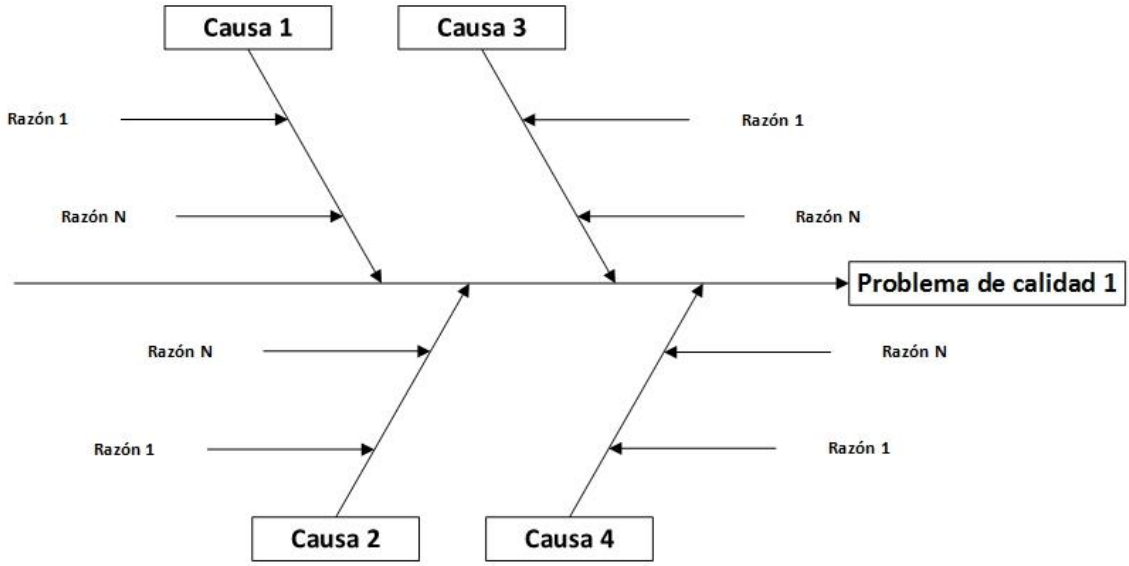

Figura 8 - Ejemplo diagrama de pescado. Fuente: creación propia

3.2.4 Aplicar técnicas de mejora

Existen 6 técnicas de mejoras propuestas por Susan Page, las cuales se aplican según la rueda de las técnicas de mejora (herramienta creada por Susan Page en su libro). Las 6 técnicas son:

1. **Burocracia:** se buscan todas las actividades que retrasan un proceso efectivo y eficiente. Normalmente, estas actividades reflejan la necesidad por tener un control excesivo o por el temor a cometer un error.
2. **Valor agregado:** evaluar si cada una de las actividades del proceso contribuyen con algún valor al cliente o consumidor.
3. **Duplicidad:** la duplicidad o redundancia se genera cuando múltiples grupos de trabajo se involucran en un proceso sin ninguna integración

entre ellos; ocasionando así la duplicación de información o la realización de tareas innecesarias.

4. **Simplificación:** se busca reducir o eliminar la complejidad de una actividad en un proceso de negocio.
5. **Reducir el tiempo de ciclo:** el tiempo de ciclo se define como el tiempo que toma la finalización de un proceso en su totalidad, incluyendo los tiempos de espera. En esta técnica se busca reducir los tiempos de cada una de las tareas.
6. **Automatización:** en esta actividad, se adaptan herramientas automatizadas a las necesidades del negocio, con el objetivo de maximizar los beneficios obtenidos de las inversiones realizadas y mejorar el proceso.

3.2.5 Implementación del cambio al proceso de desarrollo

En el paso final, se presenta el resultado del análisis de mejora, de forma tal que se pueda evaluar el progreso de los pasos anteriores. En resumen, se presentan los cambios realizados al proceso de dos maneras: la presentación del nuevo proceso mediante un diagrama de flujo y se utiliza una tabla para denotar los cambios realizados y sus razones. A continuación se muestra una plantilla de la tabla a utilizar:

Subproceso 1		
Actividad	Mejora realizada	Detalle
Actividad 1		
Actividad 2		

Tabla 5 - Detalle de mejoras realizadas. Fuente: creación propia

3.2.6 Ejecución plan piloto

Con la meta de verificar que el proceso modificado presente resultados, se aplica la ejecución de un plan piloto con el nuevo proceso.

En este plan piloto, se trabaja sobre una de las aplicaciones que fueron analizadas durante la recopilación de información, y durante 5 semanas el equipo de desarrollo trabajó utilizando el nuevo proceso en lugar del existente. Como

medida de desempeño, se utilizan criterios como la cantidad de tiquetes obtenidos en soporte y el estado de la deuda técnica del sistema.

3.2.7 Recopilación de resultados

Como último paso, se recopilan los resultados obtenidos, analizando el estado del proyecto utilizado en el plan piloto previo a su ejecución, y después de las 5 semanas de prueba.

Además, se presentan las recomendaciones y pasos a seguir por la organización para implementar el nuevo proceso en su totalidad.

4 Análisis de resultados

En esta sección, se muestran los resultados que se han obtenido a partir de todas las acciones realizadas durante el proyecto.
La primera fase realizada fue obtener una descripción de los procesos de desarrollo y gestión de calidad dentro de la organización. Para ello se elaboró un cuestionario con 10 preguntas (revisar anexo 2 para ver el documento utilizado); tomando como premisa que ya existe dicho proceso documentado y que se utiliza Scrum como metodología de desarrollo.

El cuestionario fue provisto a 6 integrantes de la organización, con la siguiente distribución de puestos de trabajo:

Puesto	Cantidad
Gerente de proyecto	1
Desarrollador	3
Analista de calidad	2

Tabla 6 - Distribución de roles en cuestionario. Fuente: creación propia

Luego de aplicar las encuestas a la población escogida, se obtuvieron las siguientes observaciones principales:

1. **Pregunta #1**: existen múltiples roles como se mencionaron anteriormente, sin embargo, el rol de desarrollador es el que se ve involucrado de mayor manera con el enfoque del presente proyecto. Los desarrolladores tienen una categorización interna donde un grupo reducido tienen la posición de líderes técnicos. Estos últimos son los encargados de realizar las revisiones de código y son asignados normalmente por aplicación (usualmente el mismo líder técnico cumple esta función en más de una aplicación a la vez). Las revisiones de código son los puntos de verificación para cada desarrollador para poder aprobar los cambios realizados, y si una revisión no aprueba los cambios, estos deben ser modificados para poder ser aprobados y publicados.

2. **Pregunta #2**: Los sprints tienen una duración de dos semanas en promedio, pero se varía dependiendo del proyecto. En algunos casos se utilizan 3 semanas pero no es una ocurrencia usual.
3. **Pregunta #3**: Cada producto puede tener una nueva versión, cada 2 semanas para aquellos proyectos de desarrollo activo. Pero para poder liberar dicha nueva versión, es necesario esperar por la aprobación del ingeniero de infraestructura de la organización y que todas las revisiones de código hayan sido aprobadas por el líder técnico. En caso de no tener las aprobaciones correspondientes y que no se hayan realizado las pruebas de calidad, no se podrá liberar dicha versión.
4. **Pregunta #4**: existe un proceso de calidad que es implementado de manera manual. Se utilizan pruebas funcionales realizadas por los aseguradores de calidad con las que se determinan que todos los requerimientos de un proyecto se han cumplido. También se realizan pruebas de regresión que implican una inversión de tiempo mayor por cada recurso de aseguramiento de calidad. Como contramedida a dicha inversión de tiempo, se han entrenado a algunos desarrolladores para realizar ciertas actividades de los aseguradores de calidad.
5. **Pregunta #5**: se registran errores en ambas etapas pero se ha notado que la cantidad de defectos en las etapas de soporte sobrepasan el límite aceptado. Proporcionalmente, se ha determinado que se detecta un 85% de errores en desarrollo y un 15% en soporte, pero realmente se espera que el número no sea mayor a un 5% de defectos en soporte.
6. **Pregunta #6**: Se posee una herramienta automatizada que analiza código, pero se encuentra limitada a código desarrollado en C#. Por lo tanto, una gran porción de código desarrollado en Javascript no se le aplica ningún análisis, y con las últimas tendencias, el porcentaje de código desarrollado en esta tecnología ha incrementado a llegar inclusive a tener una proporción del 50%. La herramienta utilizada actualmente para el análisis de código es conocida como NDepend y se tuvo que realizar un proyecto interno para poder tener un dashboard que muestre los resultados brindados por NDepend.
7. **Pregunta #7**: Las pruebas unitarias son implementadas para código C# pero no para javascript. Las pruebas unitarias en javascript no se realizan

debido a que los desarrolladores no tienen experiencia en su creación, y la infraestructura de integración continua no posee los recursos necesarios para automatizar la ejecución de dichas pruebas.

8. **Pregunta #8**: Los tiquetes de soporte pueden durar en promedio 1 semana abiertos, y el mayor factor que influye en dicho tiempo es la complejidad del código en ciertos sistemas, en otros casos se atribuye a otros factores como falta de requerimientos, incremento de alcance o limitaciones técnicas.
9. **Pregunta #9**: se manejan alrededor de 15 tiquetes por mes, lo cual incurre en que los desarrolladores inviertan una cantidad considerable de tiempo en la etapa de soporte.
10. **Pregunta #10**: Se utilizan múltiples herramientas para manejar la carga de trabajo entre soporte y nuevos desarrollos. Entre ellas se utiliza TFS (Team foundation service) para registrar las tareas a los desarrolladores y los defectos encontrados por los aseguradores de calidad. También se utiliza Sharepoint como medio de interacción entre los consumidores finales y los desarrolladores para que se registren errores de las aplicaciones que ya se encuentran en producción.

Las observaciones obtenidas muestran ciertas deficiencias que deben ser corregidas, sin embargo no todas se abarcan con el proyecto planteado. A continuación se describen las acciones que se toman para las observaciones encontradas:

1. Debido a una falta de guía para efectuar las revisiones de código, la herramienta SonarQube pudo establecer una serie de criterios a evaluar para cada desarrollador que cada líder técnico podrá verificar si se están cumpliendo. Esto, en conjunto con los estándares de la organización como nomenclatura, estructura y diseño, satisfará los aspectos necesarios para garantizar que cada sistema tenga los criterios de calidad adecuados.
2. SonarQube facilita la labor de los ingenieros de infraestructura y los arquitectos al tener un reporte inmediato del estado actual de la aplicación y disminuirá el tiempo invertido en la actividad de control de estos roles.

3. Al analizar código en Javascript en conjunto con el código en C#, en comparación con la herramienta anterior, se obtiene un reporte con una exactitud mayor al proporcionar el estado real de las aplicaciones.
4. Al gestionar la deuda técnica, se influye en la creación de mejores soluciones para los sistemas creados por la organización, lo cual repercute en la disminución del tiempo de respuesta y la cantidad de los tiquetes de producción.

Por otro lado, se encontraron ciertas observaciones que no se tomaron medidas para la ejecución del proyecto, estas son:

1. Las pruebas unitarias para el código javascript no son incorporadas en el presente proyecto. Sin embargo, se espera mantener que el porcentaje de código cubierto por las pruebas unitarias en C# no baje del 75% por proyecto (indicador impuesto previamente por la organización).
2. No se propone un cambio con las herramientas para el manejo de la carga de trabajo, a excepción del reemplazo de una de las herramientas por SonarQube.

La segunda fase en la recolección de información es el análisis utilizando SonarQube sobre un conjunto de herramientas de la organización. En total se realizó el análisis en 6 aplicaciones, 5 de ellas en soporte y desarrollo de nuevos requerimientos, mientras que la última es una aplicación solamente en desarrollo puesto que es un nuevo sistema. Debido a temas de confidencialidad, se excluyen los nombres exactos para cada sistema y se utiliza un alias utilizando la letra “S” como prefijo y un número secuencial del 1 al 6.

A continuación, se presenta de manera comparativa una tabla con los datos obtenidos al analizar cada sistema. Se toma un subconjunto de todas las métricas disponibles por SonarQube para realizar la evaluación.

Criterio	S-1	S-2	S-3	S-4	S-5	S-6

Complejidad ciclomática acumulada	5,711	7,893	1,778	2,983	6,121	573
Bloques duplicados	425	626	1,469	241	565	11
Archivos duplicados	51	74	22	40	52	9
Densidad de líneas duplicadas	13.3%	11.7%	11.0%	11.5%	13.6%	7.6%
Complejidad promedio por función	3.3	4.3	4.5	4.9	4.7	3.4
Proporción de deuda técnica	1.8%	1.5%	1.4%	1.3%	2.0%	0.8%
Calificación de mantenibilidad	A	A	A	A	A	A
Defectos	1,750	2,653	615	935	2,673	149
Días de deuda técnica	43d	45d	10d	15d	37d	2d
Vulnerabilidades	40min	1h 55min	35min	5min	5h 35min	0
Líneas de código	39,035	50,342	11,476	20,195	29,320	5,424

Tabla 7 - Estado actual de aplicaciones. Fuente: creación propia

Todas las herramientas analizadas tienen al menos 5 años de haber sido creadas, a excepción del sistema S-6, el cual es un sistema nuevo que se encuentra en etapas temprana de desarrollo y no ha sido liberado a producción todavía.

Con los datos obtenidos se puede observar que existen 3 sistemas potenciales a ser remediados: S-1, S-2 y S-5. Estos 3 sistemas son los de mayor tamaño, tomando como criterio las líneas de código. También son los que tienen un mayor número de días acumulados de deuda técnica y su complejidad ciclomática acumulada se sobrepone con los demás sistemas.

En relación proporcional, el sistema S-5 es el que se encuentra en peores condiciones al llegar a un 2% cuando los sistemas S-1 y S-2 tienen un 1.8% y 1.5%, respectivamente. Además S-5 es el que tiene el mayor número de defectos detectados y el mayor tiempo encontrado en vulnerabilidades. Sin embargo, se ha optado por utilizar el sistema S-2 como proyecto piloto a remediar, debido a que su número de días acumulado de deuda técnica es el mayor con 45 días, 8 días más que el sistema S-5.

El tercer paso fue realizar el inventario de procesos con la información obtenida a partir de las encuestas realizadas a los empleados y la observación realizada sobre la organización.

Etapa	Actividad	Dueño
Gestión de requerimientos	Recolectar requerimientos	Analista de negocio
	Establecer estimaciones	Líder técnico
	Asignar responsabilidades a desarrolladores	Líder técnico
	Priorizar labores	Líder técnico
Desarrollo	Desarrollar funcionalidades y correcciones	Desarrollador
	Crear pruebas unitarias	Desarrollador
	Realizar revisión de código	Líder técnico
	Inclusión de código en repositorio	Desarrollador
	Análisis de código estático y ejecución de pruebas unitarias	Automatizado

Verificación de calidad	Probar nueva funcionalidad	Analista de calidad
	Realizar pruebas de regresión	Analista de calidad
	Corrección de errores	Desarrollador
Publicación	Publicar código en ambiente de producción	Líder técnico

Tabla 8 - Inventario del proceso. Fuente: creación propia

A partir del inventario anterior, se logra diagramar el proceso de desarrollo con todos los roles y actividades necesarias.

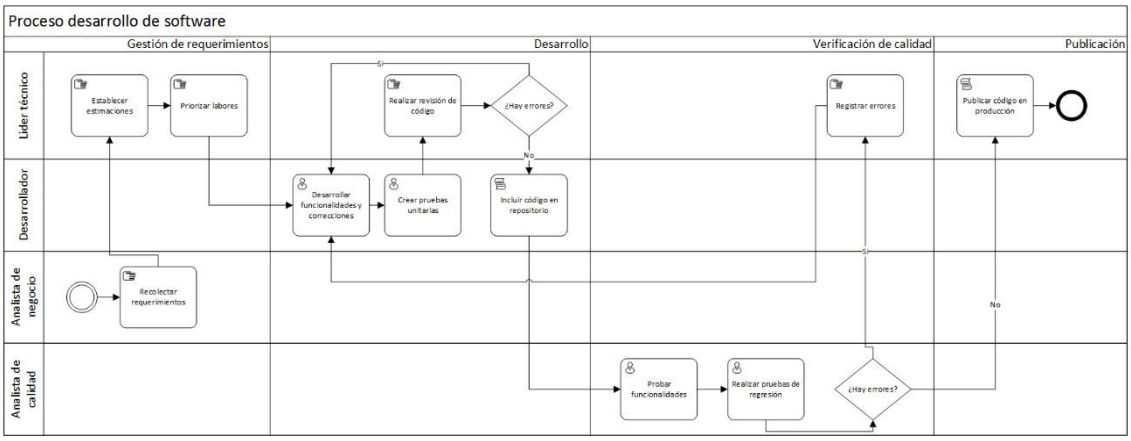

Figura 9 - Diagrama proceso desarrollo QBS. Fuente: creación propia

Con toda la información recolectada, se realizaron mediciones de calidad para encontrar los principales problemas y sus causas.

Figura 10 - Diagrama de causa y efecto para el incumplimiento con los acuerdos de servicios de tiquetes. Fuente: creación propia.

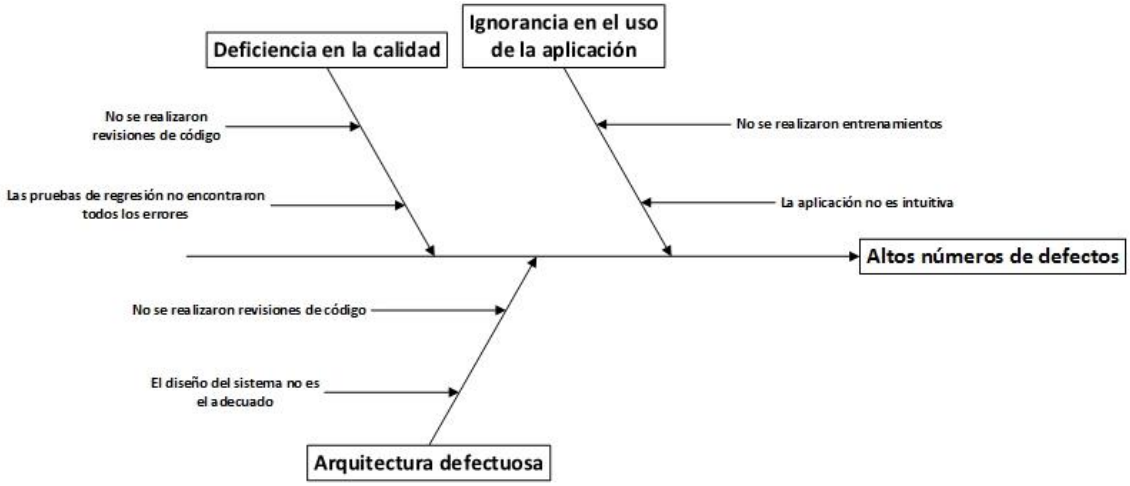

Figura 11 - Diagrama de causa y efecto para el alto número de defectos. Fuente: creación propia

De acuerdo a los diagramas mostrados anteriormente, existen razones que se ubican en múltiples causas en ambos problemas, es decir que estos son los principales puntos a solucionar. Además se pueden categorizar las razones de la siguiente manera:

- **Deuda técnica**
  - El diseño del sistema no es el adecuado
  - Código complejo
  - Incorrecta refactorización de código
  - La aplicación no es intuitiva
- **Exclusión de pasos o actividades**
  - No se realizaron revisiones de código

    - Las pruebas de regresión no encontraron todos los errores
    - No se realizaron entrenamientos
    - No se documentan componentes esenciales
- **Deficiencias en el proceso de soporte y aseguramiento de la calidad**
    - Un solo proceso de desarrollo y soporte
    - Pocos recursos de analistas de calidad
    - Proceso de aseguramiento de la calidad extenso

Basado en la categorización, es necesario encontrar soluciones de manera que la deuda técnica se mejore, se evite la exclusión de pasos y se realicen ajustes que faciliten el proceso y automaticen ciertas labores.

Al aplicar las técnicas de mejora de Susan Page, se obtuvieron los siguientes resultados:

- La técnica de burocracia no puede ser utilizada debido a que las actividades de aprobación y verificación encontradas en el proceso no pueden ser eliminadas, ya que son los principales puntos de decisión para continuar con el flujo de trabajo.
- La técnica de duplicidad no puede ser utilizada porque no es posible identificar puestos o labores duplicadas, como tampoco silos de información. Los roles involucrados son esenciales, se enfocan en actividades distintas y mantienen una comunicación activa entre ellos.
- La simplificación de actividades no es viable puesto que los procesos en análisis poseen labores sencillas que buscan ser mejoradas.
- La reducción de los tiempos de ciclos tampoco es factible ya que el tiempo de ciclo depende del negocio y el sistema sobre el cual se esté laborando y no a actividades con duraciones fijas.

Al aplicar la técnica de aumentar el valor agregado, se determinó que es necesario establecer metas para cada métrica con la cual se evalúan las aplicaciones. Estas metas permiten establecer límites de trabajo para asegurar el nivel mínimo aceptable de cada una de las aplicaciones desarrolladas por la organización. Con este objetivo, se ha optado por agregar una tarea de

validación a los roles de líder técnico para evaluar los resultados obtenidos de las herramientas de análisis y decidir las acciones correctivas en caso de ser necesarias.

Aplicando la técnica de mejora de automatización, se encontraron 2 puntos de mejora. La realización de revisiones de código debe realizarse de dos maneras: una automatizada que ejecute tareas de análisis de código estático sin tener que pasar por una revisión manual. De esta manera, las revisiones de código se enfocan en el negocio del sistema en lugar de rasgos completamente técnicos.

El segundo punto es la automatización de las pruebas de regresión para agilizar la revisión de cada versión del sistema a liberar.

A partir de las modificaciones identificadas, se han aplicado las siguientes modificaciones al proceso actual:

Proceso de desarrollo de software			
Actividad	Mejora realizada	Justificación	Detalle
Primera	Automatización de análisis de código	Al necesitar de una herramienta para el control de la deuda técnica, se opta por la utilización de una herramienta automatizada que provea dicha funcionalidad.	Se implementa una solución de análisis de código estático en el servidor que permita identificar problemas de código. Esta solución se ejecuta con cada modificación realizada por los desarrolladores y provee un reporte con las acciones necesarias a tomar.
Segunda	Automatización de pruebas de regresión	Con el objetivo de reducir el tiempo involucrado por los aseguradores de calidad, se opta por realizar automatizaciones de pruebas de regresión	Estas pruebas permiten agilizar la verificación del comportamiento de las soluciones y reducen los tiempos de ejecución.
Tercera	Validación de métricas con metas establecidas	Es necesario revisar los indicadores de las métricas analizadas para verificar que se estén entregando las soluciones con los niveles de calidad esperados.	Se incorpora una tarea de validación de métricas contra las metas de evaluación para poder asegurar que todas las aplicaciones tengan un estado mínimo aceptable antes de ser publicadas.

Tabla 9 - Resumen de modificaciones al proceso. Fuente: creación propia

A partir de lo recolectado anteriormente, el nuevo diagrama para el proceso de desarrollo ha quedado de la siguiente manera:

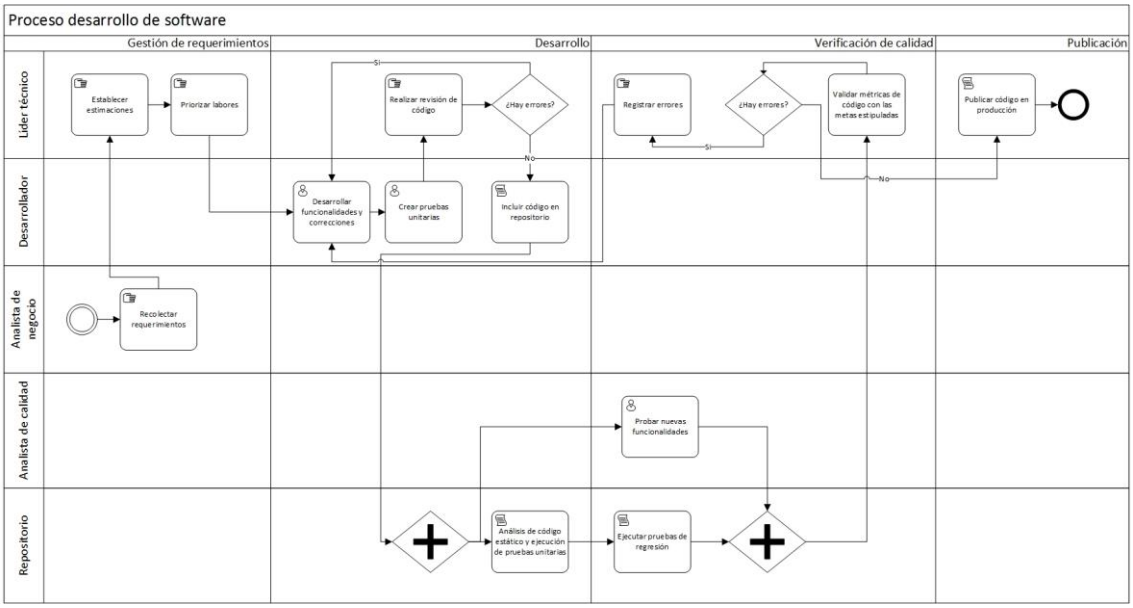

Figura 12 - Diagrama proceso modificado desarrollo en QBS. Fuente: creación propia

La inclusión de la automatización del análisis de código es el paso que conlleva la mayor cantidad de beneficios. Tener una herramienta que permita obtener un estado real de las soluciones, en comparación al sistema existente, facilitará la estimación de trabajo para la corrección de defectos; tanto en términos de tiempo como económicos, al tener una idea clara del esfuerzo necesario de ejecutar.
Esta herramienta también acelera el proceso de revisiones de código. Los líderes técnicos pueden enfocar sus esfuerzos en la lógica del negocio durante dichas revisiones, y utilizar los reportes de la SonarQube para determinar si la calidad del código implementado cumple con los estándares esperados.

La automatización de las pruebas de regresión, permite la disminución de tiempos en las labores manuales de los aseguradores de calidad. Esta labor implica una inversión de tiempo para crear las pruebas automatizadas, principalmente en los casos de uso con mayor utilización; sin embargo cuando los proyectos se encuentran en etapas de soporte, se disminuye el involucramiento de los aseguradores de calidad al tener un sistema de pruebas que detecte un cambio erróneo de manera temprana.

Esta automatización también permite que los nuevos desarrolladores que sean involucrados a un proyecto, sin tener conocimiento de todas las reglas de negocio con las cuales se rige una aplicación, puedan realizar cambios y tener una red de seguridad que cubra dichas deficiencias.

Por último, la verificación de las métricas se convierte en un paso de autorización necesario en las responsabilidades de un líder técnico. Esta verificación se incorpora a los criterios de evaluación en las revisiones técnicas, y permite establecer un control de la aplicación para mantenerla en un estado aceptable según los estándares de la organización.

4.1 Resultados obtenidos

Como proyecto piloto, se utilizó una de las herramientas analizadas para trabajar en 5 semanas las mejoras del nuevo proceso, para ello se solicitó 1 recurso para poder trabajar con los cambios.

Durante las 5 semanas se trabajó con Scrum, generando 3 sprints de 1 semana. En cada sprint se publicaban los cambios en producción y se registraban los nuevos tiempos de respuesta a tiquetes.

A continuación, se presenta una comparación de las métricas de deuda técnica entre el estado de la solución al finalizar el período de cambios con el estado inicial:

Criterio	S-2 (Inicial)	S-2 (Final)	Mejora
Complejidad ciclomática acumulada	7,893	7,764	-129
Bloques duplicados	626	628	+2
Archivos duplicados	74	72	-2
Densidad de líneas duplicadas	11.7%	11.5%	-0.2%

Criterio	S-2 (Inicial)	S-2 (Final)	Mejora
Complejidad promedio por función	4.3	4.3	=
Proporción de deuda técnica	1.5%	1.1%	-0.4%
Calificación de mantenibilidad	A	A	=
Defectos	2,653	1,862	-791
Días de deuda técnica	45d	34d	-11d
Vulnerabilidades	1h 55min	15min	-1h 40min
Líneas de código	50,342	50,267	-75

Tabla 10 - Comparación de deuda técnica al finalizar el piloto. Fuente: creación propia

Como se puede observar, los principales esfuerzos se concentraron en la reducción de defectos y el tiempo de deuda técnica. Las principales mejoras obtenidas fueron:

1. La cantidad de defectos eliminados equivale a un 30% de los que había originalmente, se eliminaron cerca de 800 defectos detectados por la herramienta de SonarQube. Dichos defectos se dividían en múltiples categorías como fallas en desempeño, seguimiento de convenciones y código confuso. Además, según su severidad se corrigieron todos los errores bloqueadores, críticos y más de 600 en menores severidades.
2. La cantidad de días de deuda técnica se redujo en un 25%, pasando de 45 días de esfuerzo estimado inicialmente a 34 días. Los 11 días eliminados han permitido que los desarrolladores fortalecieran la herramienta, simplificaran la complejidad de la solución y, como valor agregado, obtuvieran un mayor conocimiento de sus principales funcionalidades. A su vez, esto repercute en el tiempo de respuesta de los tiquetes al tener desarrolladores que trabajan sobre una herramienta conocida y que tiene un mejor diseño al que se poseía anteriormente.
3. Se redujeron las vulnerabilidades detectadas en un 86%. Las vulnerabilidades son defectos que impactan directamente la seguridad de un sistema. Es decir, entre menor sea el número de vulnerabilidades, su

nivel de seguridad será mayor. Con la reducción del 86%, los desarrolladores han fortalecido la herramienta al eliminar potenciales problemas como la inyección de código SQL o “Cross Site Scripting”. Este aspecto era de vital importancia a mejorar ya que al manejar información confidencial de la organización, se podía tener en riesgo datos que no deben ser accedidos por cualquier persona.

4. Se ha simplificado la complejidad ciclomática debido a la eliminación de código confuso encontrado por SonarQube. Como ya se mencionó, la complejidad ciclomática equivale al número de posibles caminos o flujos en un sistema, por lo tanto, la reducción de 129 flujos en un sistema equivale a una gran reducción en la complejidad del sistema.
5. Gracias a la implementación de la herramienta, hubo otros factores que fueron afectados positivamente como la reducción de líneas código que no estaban siendo utilizadas, influyendo así en la velocidad de carga del sistema, y la proporción de la deuda técnica en comparación al tamaño de la aplicación la cual ha disminuido en un 0.4%.

En relación a las labores de soporte, se observó una disminución en la cantidad de tiquetes creados por los clientes y el promedio en el tiempo de resolución. Inicialmente, se había medido un promedio de 15 tiquetes creados por mes para la aplicación S-2, cuyo promedio de resolución rondaba los 8 días.
Posterior a la implementación del proyecto piloto, se midieron los tiquetes creados en un lapso de 4 meses observando las siguientes mejoras:

1. Se disminuyó la cantidad de tiquetes creados a 6 solamente.
2. El tiempo de resolución de tiquetes a bajo a 4 días.

Analizando estos datos, se ha hecho una reducción del 80% del esfuerzo invertido en las operaciones de la etapa de soporte. Esto ha permitido reducir la cantidad de recursos asignados a un solo proyecto para soporte, y redirigir dichos recursos a proyectos activos (en desarrollo) que provean mayores beneficios. Además, los desarrolladores han mostrado un tiempo menor de adaptación a la herramienta, la cual es una de las principales quejas, lo cual se ha demostrado en los tiempos de resolución disminuidos.

Por otro lado, la automatización de las pruebas de regresión demostró la reducción de tiempo por los aseguradores de calidad. Al inicio del período del proyecto piloto, se midió el tiempo promedio utilizado por un asegurador de calidad para ejecutar una serie de casos de pruebas predefinidos. Estos casos de prueba se escogieron basado en las funcionalidades más utilizadas por los clientes. La creación de las pruebas automatizadas se realizó durante el mismo período de mejora de la deuda técnica por un recurso adicional solicitado con este fin. En total se invirtieron 55 horas, ya que el recurso no se utilizó a tiempo completo.

En el siguiente cuadro, se comparan los tiempos iniciales con los tiempos obtenidos, para cada flujo de pruebas, luego de haber finalizado el esfuerzo de automatización:

Flujo de pruebas	Tiempo inicial	Tiempo automatizado
Creación de solicitud de análisis genérica	20 minutos	2 minutos
Actualización de datos a través del módulo de "Queue"	60 minutos	10 minutos
Administración de los valores personalizados para un laboratorio	30 minutos	4 minutos
Enviar notificación de pasar una solicitud de análisis a la cola de trabajo	5 minutos	30 segundos

Tabla 11 - Comparativa de tiempos de pruebas. Fuente: creación propia

Utilizando 4 de los flujos de pruebas, se observa una mejora en promedio de un 85% en los tiempos de ejecución por cada iteración de pruebas de regresión. Y además, el involucramiento de los aseguradores se ve reducido en su totalidad ya que no hay dependencia de un recurso fijo para la ejecución de dichas pruebas.

Esta reducción conlleva a una serie de beneficios como:

- Reducción del tiempo de entrega del producto a los clientes: en este proyecto piloto se logran automatizar 4 flujos de pruebas, pero es necesario seguir ampliando dicho catálogo. La ejecución de todas las pruebas funcionales de un sistema y las pruebas de regresión pueden tomar semanas para validar que todos los escenarios estén cubiertos, sin embargo este tipo de automatizaciones podrá hacer que la validación de un sistema se haga en horas o días en lugar de tener que esperar semanas.
- Exponer defectos en un menor tiempo: similar al beneficio anterior, encontrar defectos se dará de una manera expedita ya que el sistema de pruebas alertará a los desarrolladores cuando fallen las pruebas de regresión, evitando que se deba detener o atrasar la entrega por defectos inesperados al final del desarrollo.
- Realizar un número mayor de pruebas: al realizar las pruebas de manera más eficiente y rápida, se puede brindar más flexibilidad a los aseguradores de calidad. De esta manera, se libera tiempo que pueden utilizar para incorporar escenarios más complejos al catálogo de pruebas, y realizar pruebas de desempeño las cuales aseguran que el sistema se comporte en las mejores condiciones, incluso cuando haya un número alto de usuarios de manera concurrentes.

5 Propuesta de implementación

Al observar los resultados obtenidos, se propuso implementar los cambios al proceso de desarrollo a la organización QBS en 3 etapas, iniciando en enero del año 2017.

La primera etapa es la incorporación de la herramienta de análisis de deuda técnica en el proceso de desarrollo con los siguientes pasos:

1. Implantar la herramienta de análisis SonarQube en el servidor usado como repositorio para el código fuente de las aplicaciones.
2. Habilitar el tablero de resultados de SonarQube a todos los desarrolladores.
3. Configurar el sistema de integración continúa para ejecutar el análisis de código estático en todas las aplicaciones.
4. Informar a los desarrolladores y líderes técnicos del nuevo flujo en el proceso de desarrollo.
5. Calendarizar revisiones periódicas por los arquitectos para verificar que el nivel de deuda técnica en las aplicaciones esté mejorando.

Esta primera etapa solo incluye asignar un porcentaje de esfuerzo a los arquitectos y los ingenieros de sistemas para poder implantar la nueva herramienta. Se estima una duración de 8 semanas, tomando en cuenta la carga de trabajo estimada por dichos recursos.

La segunda etapa consiste en la aplicación del análisis de deuda técnica en todo el portafolio de aplicaciones, la estimación de recursos a utilizar y el plan de acción a ejecutar en QBS. Esta etapa se compone de los siguientes pasos:

1. Ejecutar análisis de aplicaciones por programa.
2. Establecer prioridades basado en dos variantes:
    a. **Aplicaciones heredadas (“*Legacy*”):** todas aquellas aplicaciones que no tienen una versión en desarrollo actualmente y solo se les encuentra dando soporte.
    b. **Nuevos desarrollos:** todas las aplicaciones que los equipos de desarrollo estén creando y no hayan estado en etapa de soporte previamente.

3. Basado en el análisis obtenido, definir el porcentaje de mejora para las aplicaciones definidas como prioridad y establecer un porcentaje de los recursos para cada programa en el que sus esfuerzos se enfoquen en la disminución de dicha deuda técnica.
4. No todas las aplicaciones escogidas muestran un retraso en las fechas de entrega, sin embargo, para aquellas que si son afectadas se debe discutir con los gerentes de cada programa para realizar una recalendarización de las aplicaciones y proveer la ayuda necesaria con la discusión de los clientes.
5. Para todas las aplicaciones nuevas, se incorporan las labores de gestión de deuda técnica en las listas de tareas para desarrollo.

En esta etapa se deben incluir los líderes técnicos y los gerentes de programas. Ambos roles deben trabajar en conjunto para recolectar los datos de los análisis y realizar la priorización. Además, los líderes técnicos deben informar también a los desarrolladores los cambios en la asignación de trabajo. Se estima una duración de 10 semanas para proveer el estado y el plan de ejecución en todos los sistemas, y es ejecutado posterior a la primera etapa.

La tercera, y última etapa, consiste en la automatización de las pruebas de regresión y ejecución del plan de mejora de deuda técnica, con los siguientes pasos:

1. Asignar un recurso, con experiencia en la creación de pruebas automatizadas utilizando Selenium, para trabajar con las aplicaciones heredadas y crear una serie de pruebas basadas en el flujo básico de uso de cada una de las aplicaciones.
2. Asignar un porcentaje de tiempo para los recursos de calidad existentes asociados a los nuevos desarrollos para la creación de las pruebas automatizadas.
3. Combinar la creación de las pruebas automatizadas con las labores de reducción de deuda técnica.

Los roles involucrados en esta etapa son los ingenieros de calidad y los líderes técnicos. Es necesario realizar la contratación de un nuevo recurso para trabajar con las aplicaciones heredadas y se propone que se contrate con la modalidad de subcontratación. Tanto los ingenieros de calidad y los líderes técnicos, una vez se hayan definido las aplicaciones principales a mejorar, deben trabajar en conjunto para definir todas las pruebas automatizadas a crear.

Implementar los pasos de la propuesta descrita, exige la necesidad de gestionar todas las fases como un proyecto organizacional, por ende existen riesgos o vulnerabilidades que puedan ocasionar retrasos durante la ejecución. A continuación se presentan los principales riesgos detectados:

1. Existen herramientas que no pueden ser analizadas por la herramienta SonarQube: este riesgo prevé la posibilidad de encontrar herramientas dentro de la organización que no puedan ser analizadas; por ejemplo aquellas desarrolladas en lenguajes como Visual Basic 6.0 donde los complementos para su análisis poseen un costo. Se categoriza como un riesgo de probabilidad alta pero impacto bajo. Para tratar este riesgo existen dos soluciones, eliminar los sistemas que cumplan dichas características de los esfuerzos de remediación, o realizar la inversión para la compra de dichos complementos.
2. Rotación de personal y pérdida de conocimiento: uno de los principales riesgos es la pérdida del conocimiento si una de las personas deja la organización. El área donde se daría el mayor impacto es con los aseguradores de calidad, ya que la automatización de pruebas es realizada por un recurso asignado. Se categoriza como un riesgo de probabilidad media pero impacto alto. Para tratar este riesgo, se incorporan medidas de mitigación como entrenamientos simultáneos donde existan al menos 2-3 personas que conozcan la herramienta para la automatización de pruebas.
3. Cambios en las prioridades de gerencia: al ser un proyecto interno a la organización, los gerentes de la organización pueden asignar recursos a proyectos que tengan mayores beneficios. Realizar esto puede retrasar el proyecto de manera indefinida. Se categoriza como un riesgo de probabilidad baja, con un impacto medio. Para tratar este riesgo, se busca

la reducción de la probabilidad incorporando a la gerencia a reuniones donde se pueda observar el beneficio obtenido y la negociación del tiempo asignado por recurso para dichos esfuerzos.

4. La curva de aprendizaje para la nueva herramienta es más larga de lo esperado: los desarrolladores deberán aprender las principales funcionalidades a utilizar dentro de la herramienta y pueden ocasionar que hayan retrasos por falta de conocimiento en su uso. Se categoriza como un riesgo de probabilidad baja, con un impacto bajo. Para tratar este riesgo se busca su eliminación al realizar entrenamientos internos o la utilización de medios externos como tutoriales en línea.

La propuesta de implementación se espera que finalice el 10 de noviembre del 2017, donde ya se han completado la primera etapa en su totalidad y se espera terminar la segunda etapa en la segunda semana de mayo; esto supone un retraso de dos semanas en comparación a la estimación original. A continuación se muestra dicho cronograma:

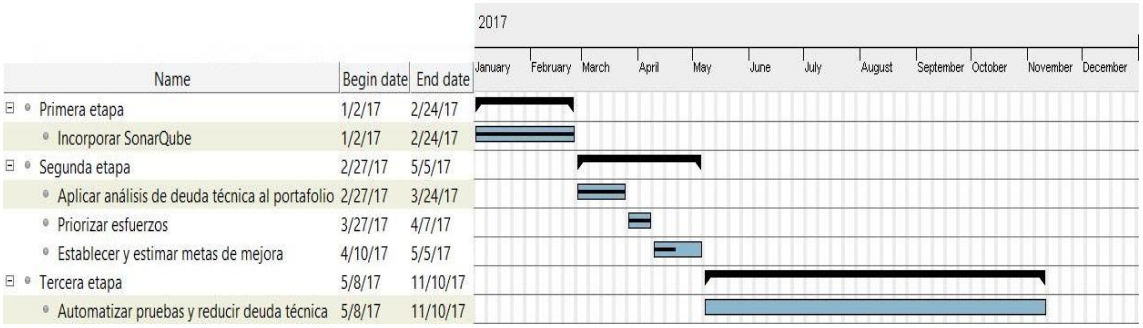

Figura 13 - Cronograma propuesta de implementación.

Para finalizar, en el cuadro siguiente se exponen las responsabilidades para cada rol durante la ejecución del proyecto:

	Arquitectos	Líderes técnicos	Ingeniero de calidad	Gerente de programa
Incorporar SonarQube a los ambientes de desarrollo	R/E	I	I	I
Establecer revisiones periódicas de la deuda técnica para toda la organización	R/E	I	I	I
Ejecutar análisis inicial de las aplicaciones		R/E		I
Establecer prioridades de trabajo		R/E		R/E
Definir acciones de mejora		R/E	I	I
Establecer nuevas fechas de entrega				R/E
Crear pruebas automatizadas		I	R/E	

Tabla 12 - Matriz de roles y responsabilidades. Fuente: creación propia

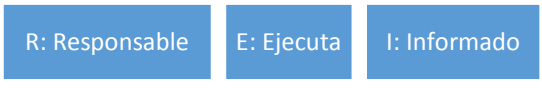

Figura 14 - Simbología matriz de roles y responsabilidades

6 Conclusiones

Como cierre de este proyecto, es necesario destacar que se alcanzaron los objetivos y entregables planteados inicialmente. Cada etapa de la metodología empleada reflejaba los resultados esperados, y a partir de ellos se logró iniciar la implementación de un proyecto de mayor escala dentro de la organización.

El desarrollo del proyecto permitió satisfacer el objetivo principal, proponiendo un modelo para el manejo de la deuda técnica, cuyas mejoras impactaron los sistemas existentes de la organización de manera positiva.
Para desarrollar dicho modelo, se siguieron una serie de etapas u objetivos específicos necesarios que culminaron en la aplicación del modelo modificado en un sistema determinado.

Como primer objetivo, se necesitó analizar el estado actual de los sistemas de la organización. Para ello se decidió utilizar la herramienta de análisis de código estático conocida como SonarQube, donde para cada aplicación se realizó una evaluación y se recopilaron todos los resultados de la calidad del código. Luego de analizar todos los resultados, se evaluó cuál de las aplicaciones necesitaba con mayor urgencia realizar una remediación; donde se utilizaron un conjunto de métricas predefinidas para determinar dicho resultado. Además, como parte de la recopilación de información, se realizó una encuesta a un grupo de personas de los equipos de desarrollo para poder analizar el proceso existente y las responsabilidades de cada rol existente.

El segundo objetivo correspondía a generar el nuevo modelo de desarrollo. Esta etapa incorporó la aplicación de una metodología de mejora de procesos, específicamente la propuesta por Susan Page; sin embargo se estableció una serie de modificaciones que permitirían realizar la ejecución de un plan piloto durante el desarrollo del proyecto. La metodología utilizada finalmente estaba constituida por 7 pasos, los cuales se enumeran a continuación:

1. Realizar el inventario de procesos
2. Establecer la base y diagramar el proceso

3. Medición de calidad
4. Aplicar técnicas de mejora
5. Implementación del cambio al proceso de desarrollo
6. Ejecución del plan piloto
7. Recopilación de resultados

Los primeros 5 pasos de la metodología generaron el nuevo modelo, el cual proponía 3 nuevas modificaciones que incorporan la automatización de ciertas actividades y la adición de una nueva actividad que cumple como verificación y control de la calidad de los productos creados.

A partir del quinto paso de la metodología, se inicia la ejecución del tercer objetivo, el cual establecía demostrar los resultados del nuevo modelo al aplicarlo sobre un sistema existente en la organización. El sistema utilizado fue escogido durante el primer objetivo mencionado anteriormente, y durante un período de 5 semanas se estableció el nuevo flujo de trabajo, finalizando con la recolección de los resultados. Entre ellos podemos mencionar los siguientes como los de mayor relevancia:

- Disminución del tiempo de respuesta promedio por cada tiquete en la etapa de soporte.
- Disminución de los tiempos de ejecución para las pruebas de regresión.
- Eliminación de un 30% de los defectos encontrados y de un 85% de las vulnerabilidades encontradas en un sistema.

Es importante detallar que todos los resultados medidos con controlables, y como parte del piloto se demostró a la organización cómo realizar dichas mediciones para proyectos futuros.

Como último objetivo, y gracias a los resultados obtenidos, se propuso la necesidad de implementar el nuevo modelo a otros sistemas que se encuentren en las mismas condiciones. De esta manera, se presentó una propuesta de implementación durante el año 2017 que tendría una duración de 10 meses.

7 Recomendaciones

Dentro de las principales recomendaciones a lo largo de la ejecución del proyecto se encontraron las siguientes:

1. Las revisiones técnicas deben ser estandarizadas y no depender del juicio de los líderes técnicos. A pesar de tener confianza en los líderes técnicos, es necesario que se tenga un listado de elementos a revisar o una guía a seguir como método de evaluación, con el objetivo de asegurar que todas las revisiones hayan pasado por el mismo control de calidad.
2. Se recomienda incluir un análisis de costos, además del análisis de calidad, para agregar un punto de vista extra en la priorización de aplicaciones a trabajar. Paralelamente, es necesario incorporar un análisis de riesgo en la implementación a la organización, permitirá también determinar las posibles amenazas y eventos no deseados que puedan ocurrir y afectar la incorporación del manejo de la deuda técnica.
3. Se recomienda contar con un plan de comunicación amplia y precisa, que no solamente incluya al equipo de desarrollo, sino que también incorpore los principales clientes de la organización y establezca de manera concisa los nuevos objetivos agregados, permitiendo hacer conciencia en los clientes del nuevo rumbo a tomar y los beneficios obtenidos a partir de ello.
4. Es importante destacar que, el involucramiento de los gestores de programas es vital para poder lograr realizar una implementación exitosa en toda la organización.
5. Utilizar los resultados de los análisis de deuda técnica para encontrar los principales aspectos a mejorar en los equipos de desarrollo. Esto se puede lograr detectando los grupos de métricas más afectados y buscando entrenamientos que mitiguen esa debilidad.
6. Incrementar el alcance de análisis de SonarQube mediante la incorporación de una licencia para encontrar fallas en más lenguajes de programación, SQL principalmente. El analizador para SQL tiene un costo de 700 dólares que deberá ser evaluado para determinar si es necesario comprar.

8 Anexos

8.1 Anexo 1 – Descripción de la herramienta SonarQube

SonarQube, conocido anteriormente solamente como Sonar, es una plataforma de código abierto utilizado para la medición y análisis de la calidad del código fuente de los sistemas de información. Se encuentra desarrollada en Java pero es apta para el análisis de más de 20 lenguajes de programación distintos.
La herramienta puede ser utilizada de manera individual, ejecutando los análisis de manera manual, sin embargo sus mayores beneficios se obtienen al integrarlo con sistemas de integración continua como Jenkins o “Team Foundation Server” de Microsoft.

La primera versión de la herramienta fue liberada el 14 de diciembre del año 2007; y actualmente se liberó su última versión el 2 de junio del 2017. SonarQube es desarrollada por la compañía SonarSource, especializada en la creación de herramientas para la mejora continua de la calidad del código.
SonarSource ofrece un programa empresarial para todas aquellas organizaciones que deseen pagar por el soporte de la herramienta, como también por consultorias y asesorías al implementar la solución.

Las principales funcionalidades que provee la herramienta de análisis de código se pueden dividir en 5 categorías, estas son [30]:

1. Inspección continua:
    a. Mediante el uso de gráficos, tablas e indicadores, se presenta el estado general de una aplicación de manera rápida, obteniendo así una sensación de los resultados obtenidos a través del tiempo.
    b. Utilizando el paradigma de la fuga de agua, se puede medir la calidad del código en términos del código nuevo en comparación al existente. Al detectar los principales puntos donde haya una fuga de calidad, se pueden tomar medidas de control para remediar los problemas.

c. Se permite reforzar el cumplimiento de los indicadores de calidad al establecer puertas de calidad, las cuales son metas a cumplir para poder aprobar un sistema.
d. Cada error posee una descripción detallada y se puede observar el código fuente donde se encontró. SonarQube permite determinar los niveles de criticidad para cada error, como también mostrar posibles maneras para remediarlos. Además, se provee un tiempo estimado para aplicar la solución lo que permite calcular el tiempo total de esfuerzo a utilizar.

2. Detección de errores:
    a. Se detectan errores basados en un catálogo de errores comunes entre los que se encuentran errores de lógica, fugas de memorias, problemas de desempeño, de-referencias de punteros nulos, vulnerabilidades de seguridad, entre otras. Cada error es catalogado en 5 niveles de importancia: bloqueadores, críticos, mayores, menores e informativos.
    b. Los olores de código (“code smells”) son informados a los desarrollados donde se pueden encontrar duplicidades de código, excesivas complejidades o código no cubierto por las pruebas unitarias.
3. Múltiples lenguajes:
    a. Se analizan más de 20 lenguajes de programación entre los que se pueden encontrar C, C++, Javascript, C#, Java, SQL, PHP, Python, entre otros.
    b. Los proyectos de múltiples lenguajes presentan sus resultados por cada lenguaje de programación empleado.
4. Integración con DevOps:
    a. Se provee la integración con múltiples herramientas de integración continua como Jenkins, Bamboo, TeamCity o TFS.
    b. Se provee un sistema de notificaciones con múltiples acciones a ejecutarse cuando se cumplen condiciones en cada compilación de código con los sistemas de integración continua.

5. Centralización de la calidad:
    a. Se utiliza un sistema como punto de control para todos los proyectos y métricas de calidad utilizadas.
    b. Las reglas de calidad pueden ser compartidas entre múltiples proyectos pero se pueden realizar modificaciones cuando se necesiten aplicar excepciones en proyectos específicos.

8.2 Anexo 2 – Cuestionario de estado actual de la organización

Nombre:			
Profesión:			
Fecha de entrevista:			
Pregunta 1:	¿Qué roles tienen y cuáles son sus responsabilidades?		
Respuesta:			
Pregunta 2:	¿Qué duración tienen los sprints?		
Respuesta:			
Pregunta 3:	¿Con qué frecuencia liberan nuevas versiones del producto? ¿Existe algún limitante que restrinja esta frecuencia?		
Respuesta:			
Pregunta 4:	¿Existe un proceso de verificación de calidad implementado?		
Respuesta:			
Pregunta 5:	¿En qué etapas se registran los errores de la aplicación: desarrollo, soporte, ambas?		
Respuesta:			
Pregunta 6:	¿Se realizan revisiones de código para cada nueva versión? Mencione si existe alguna herramienta que automatice esto.		
Respuesta:			
Pregunta 7:	¿Se realizan pruebas unitarias en el código?		
Respuesta:			
Pregunta 8:	¿Cuánto se dura resolviendo un tiquete de soporte normalmente y cuáles son las razones por las que se toma dicho tiempo?		
Respuesta:			
Pregunta 9:	¿Cuántos tiquetes manejan semanalmente?		
Respuesta:			
Pregunta 10:	¿Qué herramientas utilizan para gestionar la carga de trabajo?		
Respuesta:			

9 Bibliografía

[1] J. Holvitie, V. Leppanen y S. Hyrynsalmi, «Technical Debt and the Effect of Agile Software Development Practices on it,» *IEEE Computer Society,* pp. 35-42, 2014.

[2] W. Cunningham, «The WyCash portfolio management system,» *OOPSLA*, vol. 18, nº 22, pp. 29-30, 1992.

[3] P. Kruchten, «Strategic Management of Technical Debt,» *IEEE Computer Society*, pp. 282-284, 2012.

[4] Real Academia Española, «Definición de Metodología,» [En línea]. Available: http://dle.rae.es/?id=P7eTCPD.

[5] R. Pressman, Ingeniería del Software: Un enfoque práctico, McGraw-Hill Education, 1989.

[6] M. Beedle, A. Cockburn, W. Cunningham , M. Fowler, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, K. Swchaber, J. Sutherland y D. Thomas, «Manifiesto por el Desarrollo Ágil de Software,» 2001. [En línea]. Available: http://agilemanifesto.org/iso/es/manifesto.html. [Último acceso: 15 Setiembre 2016].

[7] J. Meier, «Extreme Programming (XP) at a Glance (Visual),» Microsoft, 6 Junio 2014. [En línea]. Available: https://blogs.msdn.microsoft.com/jmeier/2014/06/06/extreme-programming-xp-at-a-glance-visual/. [Último acceso: 23 Octubre 2016].

[8] Springtimesoft, «Agile software development through Scrum,» Springtimesoft, [En línea]. Available: https://springtimesoft.co.nz/agile-software-development-scrum/.

[9] G. O'Reagan, Introduction to software quality, Springer, 2014.

[10] A. Khanjani y R. Sulaiman, «The process of quality assurance under open source software development,» de *IEEE symposium on Computers and Informatics*, 2011.

[11] R. T. d. Sousa, F. E. G. d. Deus, B. A. d. Sousa, A. P. F. Araújo, M. Holanda, W. M. C. Silva, H. Freitas, S. S. A. N. Vidal, R. M. G. d. Santos y A. Moraes, «A methodology for quality assurance for business process modeling with BPMN: A

case study for the SIGEPE software,» de *Iberian Conference on Information Systems and Technologies* , Spain, 2016.

[12] «ISO/IEC/IEEE 29119 Software Testing,» 10 Setiembre 2014. [En línea]. Available: http://www.softwaretestingstandard.org/.

[13] IEEE, «IEEE,» [En línea]. Available: www.ieee.org.

[14] Selenium, «What is Selenium?,» [En línea]. Available: http://docs.seleniumhq.org/. [Último acceso: 14 09 2016].

[15] Microsoft, «TDD in .NET Framework,» 25 Julio 2005. [En línea]. Available: https://msdn.microsoft.com/es-es/library/bb932285.aspx.

[16] IEEE, «730-2014 - IEEE Standard for Software Quality Assurance Processes,» IEEE, 23 Junio 20114. [En línea]. Available: http://ieeexplore.ieee.org/document/6835311/?arnumber=6835311. [Último acceso: 4 Noviembre 2016].

[17] ISO, «ISO/IEC 25000:2014,» 15 Marzo 2014. [En línea]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=64764.

[18] W. Cunningham, «The WyCash portfolio management system,» *ACM SIGPLAN OOPS Messenger*, vol. IV, nº 2, pp. 29-30, 1993.

[19] I. Griffith, C. Izurieta, H. Taffahi y D. Claudio, «A SIMULATION STUDY OF PRACTICAL METHODS FOR TECHNICAL DEBT MANAGEMENT IN AGILE SOFTWARE DEVELOPMENT,» de *Winter Simulation Conference*, Georgia, 2014.

[20] OWASP, «Static Code Analysis,» 5 Julio 2016. [En línea]. Available: https://www.owasp.org/index.php/Static_Code_Analysis.

[21] SonarSource, «Metric Definitions,» 27 Mayo 2016. [En línea]. Available: http://docs.sonarqube.org/display/SONAR/Metric+definitions.

[22] C. Fernandez-Sanchez, J. Garbajosa y A. Yagüe, «A Framework to Aid in Decision Making for,» de *2015 IEEE 7th International Workshop on Managing Technical Debt*, Bremen, Alemania, 2015.

[23] F. Oliveira, A. Goldman y V. Santos, «Managing Technical Debt in Software Projects Using Scrum: An Action Research,» de *Agile Conference*, Washington DC, 2015.

[24] J. Jeston y J. Nelis, Business Process Management, Practical Guidelines to Succesful Implementations, New York: Routledge, 2008.

[25] Gartner, «Business Process Management (BPM),» [En línea]. Available: http://www.gartner.com/it-glossary/business-process-management-bpm/. [Último acceso: 21 09 2016].

[26] S. Page, Business Process Improvement, 10 Simples Steps to Increase Effectiveness, Efficiency and Adaptability, New York: AMACOM, 2010.

[27] R. Hernandez, C. Fernandez y P. Baptista, Metodología de la Investigación, Mexico DF: McGraw-Hill, 2006.

[28] S. Page, «Is There a Difference Between 6 Sigma, Lean, Kaizen, BPM, Continuous Improvement, and Reengineering?,» Ezine Articles, 15 Mayo 2011. [En línea]. Available: http://ezinearticles.com/?Is-There-a-Difference-Between-6-Sigma,-Lean,-Kaizen,-BPM,-Continuous-Improvement,-and-Reengineering?&id=6270998.

[29] H. Zala, «What is Six Sigma?Its Advantages and Disadvantages,» Linkedin, 2 Septiembre 2015. [En línea]. Available: https://www.linkedin.com/pulse/what-six-sigmaits-advantages-disadvantages-harkisan-zala.

[30] Sonarqube, «Features,» SonarSource, [En línea]. Available: https://www.sonarqube.org/.

[31] G. Myers, C. Sandler y T. Badgett, The Art of Software Testing, Wiley John + Sons, 2011.

[32] QBS, «Welcome to QBS,» June 2016. [En línea]. Available: qbs.intel.com.