INSTITUTO UNIVERSITARIO AERONÁUTICO

Facultad de Ciencias de la Administración

Proyecto de Grado

Creación de un proceso de desarrollo aplicando las mejores prácticas de la industria.

Alberto Alejandro Guardo

DNI: 30456343 - aguardo343@alumnos.iua.edu.ar

María Belén Sosa Tosello

DNI: 34768951 - msosa951@alumnos.iua.edu.ar

Profesor tutor: Ing. María Alejandra Boggio

16 de Abril de 2018

Índice de Contenidos

AGRADECIMIENTOS

A nuestras familias y amigos que nos acompañaron a lo largo de todo el proceso alentándonos a nunca bajar los brazos y perseverar para alcanzar la meta.

A nuestros compañeros de carrera que, de alguna manera, directa o indirecta, colaboraron en que hayamos llegado hasta aquí.

A nuestros profesores que a lo largo de la carrera fueron formándonos en cada una de las herramientas que pudimos poner en práctica en nuestra vida profesional y en particular en este proyecto de grado.

A nuestra tutora la Ingeniera Alejandra Boggio, por su entera predisposición y entrega, guiándonos en cada detalle de este proyecto.

¡Gracias a todos y cada uno de ustedes!

Belén y Alberto

RESUMEN DEL PROYECTO DE GRADO

Las metodologías tradicionales proporcionan un marco más rígido para la documentación de los requerimientos, a su vez las metodologías ágiles brindan prácticas de fácil adopción que ayudan a que los proyectos de desarrollo de software finalicen de manera exitosa. Durante los primeros capítulos de este trabajo se recopilaran e investigaran las mejores prácticas tanto de las metodologías tradicionales como de las metodologías ágiles.

Luego se describirá la problemática actual de una software factory que brinda sus servicios de desarrollo a un cliente externo a la misma. Para esto, se realizará un análisis enfocado en determinar cuál es la causa de fondo de los distintos desafíos que afronta el equipo de desarrollo y como pueden ser solucionados.

Una vez identificado el problema de fondo y su posible solución se seleccionarán las mejores prácticas de las metodologías investigadas tomando aquellas que colaboren en la solución a la problemática tratada en este trabajo. A partir de estas prácticas se generara un nuevo proceso de desarrollo personalizando el mismo para que sea compatible con las necesidades de la compañía.

Finalmente, se realizará la implementación del nuevo proceso de desarrollo en un proyecto representativo para determinar cuál es el impacto (si lo hubiera) del nuevo proceso de desarrollo y en base a esto determinar si puede ser el nuevo modelo que se usará como estándar en para el cliente.

GLOSARIO

Software Factory: una empresa de la industria del software cuya misión es el desarrollo de software para sus clientes de acuerdo con los requisitos específicos que aquel le solicita.

Equipo de Testing o Equipo de QA: Equipo integrado por Testers

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

WIP: Del inglés Work In Progress. Utilizado para definir "Trabajo en Progreso" de una manera abreviada.

Desarrollador Senior: Desarrollador con más de 3 años de experiencia y un profundo conocimiento de la aplicación en la que trabaja.

Historia o User Story: es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario.

CAPÍTULO 1. INTRODUCCIÓN

1.1 Metodologías ágiles

Desde mediados de la década de 1990 las metodologías ágiles han ido evolucionando hasta convertirse actualmente en el método más utilizado por las empresas de desarrollo de software en sus proyectos, brindan un marco teórico y práctico que sirve de guía para encarar las distintas etapas del ciclo de vida del desarrollo. Si bien hay varias metodologías existentes, actualmente las más utilizadas en el mercado son Extreme Programming, Scrum, Kanban y Agile Inception, cada una con sus ventajas y debilidades.

1.2 Descripción

El caso de estudio que se tratara en este proyecto se origina en una software factory que presta servicios a una de las principales aerolíneas de los Estados Unidos. Dicha empresa utiliza una metodología ágil como guía para darle marco a su propio proceso de desarrollo de los proyectos de software que administran.

Estos proyectos consisten en nuevos módulos y modificaciones a los modulos ya existentes del sistema de reservas de la aerolínea utilizado en todo el mundo por sus agentes. Este sistema es altamente crítico ya que la mayoría de los ingresos de la aerolínea se generan a partir de las ventas realizadas a través del mismo. Los proyectos que se trabajan son de alta complejidad debido a que el sistema abarca todos los productos que la aerolínea tiene a la venta, así como también la interacción entre los mismos y su disponibilidad la cual varía según el país y los agentes.

A raíz de reiterados problemas de comunicación y defectos inyectados por el equipo de desarrollo que llegaron al ambiente previo a producción. Se identificó que, al no haber sido detectados por el equipo de Testing o de desarrollo, existe una necesidad de revisar el proceso de desarrollo ya que estos defectos podrían haber impactado directamente en la facturación de la compañía.

1.3 Justificación del trabajo

Se realizó un estudio de la causa raíz de los defectos encontrados y de porque pasaron las distintas instancias al punto de casi llegar al ambiente de producción. El estudio se basó en una investigación detallada de cada uno de los componentes del proyecto, por componentes la investigación se refirió a:

- Procesos
- Equipos técnicos
- Líderes
- Documentación

En un principio se atribuyeron los problemas a errores humanos, pero luego de una serie de reuniones con desarrolladores y analistas de calidad, así como también con los líderes de los distintos equipos se determinó que había una falla en la comunicación entre el equipo de negocio y los equipos técnicos que tienen que desarrollar los productos de software. El equipo de negocio cambiaba constantemente los requerimientos a la vez que no eran claro en los mismos, por otro lado, cuando los requerimientos no estaban claros, los equipos técnicos se basaban en suposiciones sobre cómo debía comportarse el sistema por su propia experiencia en el mismo y a su vez los equipos de Testing al no tener una definición clara de lo que debían probar realizaban pruebas generales y muchos errores pasaban sin ser detectados.

En función de esta investigación y partiendo de la base de que: "al mejorar el proceso se mejorará el producto", se determinó que era necesario modificar el proceso de desarrollo que se estaba implementando con el fin de obtener una documentación más detallada que brindara un mayor control sobre el código y los requerimientos del cliente, así como también un mayor respaldo y sobre todo una aceptación de dichos requerimientos por parte de cliente.

CAPÍTULO 2. OBJETIVOS

2.1 General

El objetivo general del proyecto será la generación de un proceso de desarrollo combinando las mejores prácticas de las metodologías ágiles y tradicionales que pueda responder de manera eficiente a los problemas que no está abordando correctamente el proceso de desarrollo actualmente utilizado. En otras palabras, se realizará una adaptación (o tailoring) de las mejores prácticas de las distintas metodologías, buscando que además de proveer respuesta a los problemas actuales, brinde una mayor eficiencia que el proceso de desarrollo previo en lo referido a calidad y rapidez de entrega de las distintas instancias del proyecto.

2.2 Específicos

- Disminuir la cantidad de defectos por errores del equipo de desarrollo.
- Disminuir el esfuerzo innecesario de los equipos de Testing.
- Brindar una documentación sólida sobre la cual no queden dudas para el equipo de desarrollo o para el equipo de negocio.
- Asegurarse que los representantes del negocio tengan un completo entendimiento de los requerimientos, así como de qué funcionalidades recibirán.
- Eliminar esfuerzos innecesarios y de retrabajo por fallas de la comunicación.
- Disminuir el tiempo total de los proyectos.
- Aumentar la calidad del producto de software final.
- Eliminar reuniones innecesarias y repetitivas para aclarar los requerimientos.
- Acotar el foco sobre el cual los equipos de Testing deben trabajar.
- Eliminar tiempos muertos por falta de información.

CAPÍTULO 3. ALCANCE DEL TRABAJO

El ámbito de actuación para este trabajo será en un equipo que desarrolla software para una importante aerolínea de Estados Unidos.

El alcance de este trabajo implicará definir un nuevo proceso de desarrollo aplicando diferentes herramientas que proveen las metodologías ágiles y las tradicionales. A partir de este nuevo proceso de desarrollo se realizará una Prueba de Concepto con la que se busca demostrar que al implementar el nuevo proceso se observará una reducción de los defectos inyectados a la vez que se obtiene un software de mayor calidad y con una mayor velocidad de desarrollo. Para esto se realizará un análisis “Post Mortem” donde se definirá de manera precisa que se medirá, cómo y también cuando se aceptará y cuando se rechazará.

Se debe destacar que el alcance del trabajo sólo incluirá las áreas de Investigación de Requerimientos, Diseño, Desarrollo e Implementación. Quedan fueran del alcance de este trabajo las áreas de Testing y Gestión del Proyecto.

CAPÍTULO 4. MARCO CONCEPTUAL

4.1 CMMI

"CMMI es un modelo de mejora de rendimiento de clase mundial para organizaciones competitivas que desean lograr operaciones de alto rendimiento. De eficacia probada en organizaciones y gobiernos a nivel mundial en los últimos 25 años, CMMI consiste en recopilar las mejores prácticas diseñadas para promover los comportamientos que conducen a un mejor rendimiento en cualquier organización"[1].

Niveles de Madurez

Existen 5 niveles de madurez en CMMI que sirven tanto para evaluar la capacidad de una organización como también compararla con otras. Estos niveles además brindan una guía hacia la mejora, identificando las áreas a mejorar y aplicando las mejores prácticas de cada nivel las organizaciones alcanzarán los niveles de madurez más altos. Los niveles más altos les brindarán ventajas competitivas ante otras organizaciones demostrando a sus stakeholders dedicación para llegar a la excelencia [1].

Los niveles de CMMI indican lo siguiente [1]:

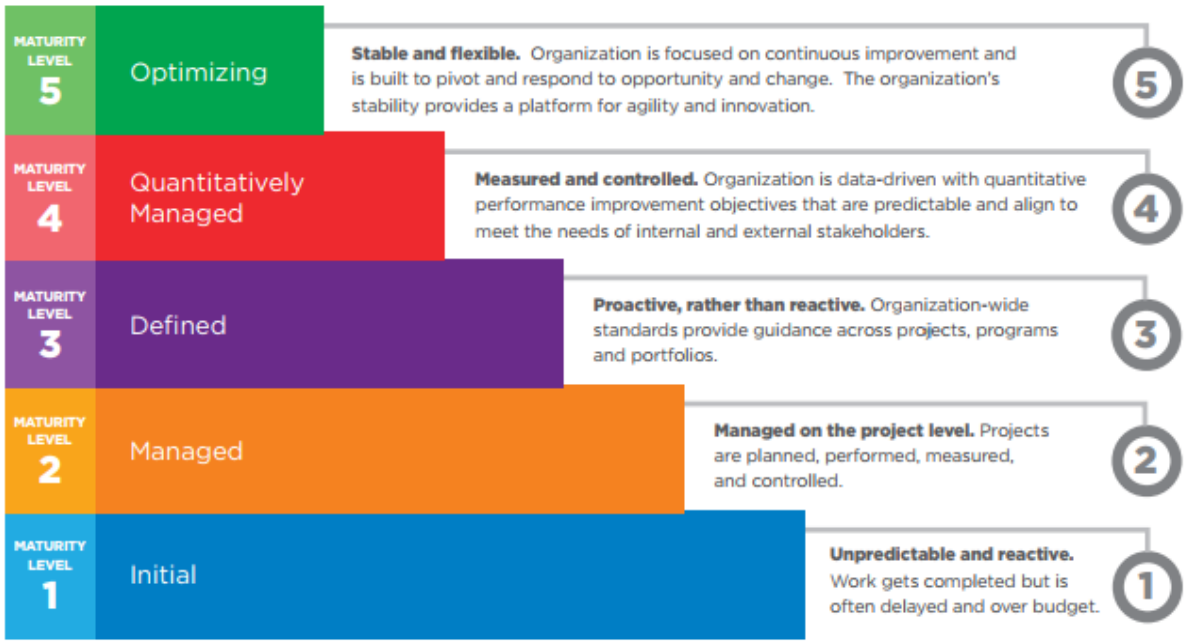

Figura 1. CMMI Maturity Levels, cmmiinstitute.com [Internet] CMMI® Institute LLC,2017 [citado 13 de Noviembre de 2017] Disponible en: http://cmmiinstitute.com/capability-maturity-model-integration

1) **Inicial. Impredecible y reactivo.** El trabajo se completa, pero es a menudo retrasado y por encima del presupuesto.
2) **Gestionado.** Gestionado en el nivel del proyecto. Proyectos son planeados, realizados, medidos, y controlado
3) **Definido.** Proactivo, en lugar de reactivo. Toda la organización las normas brindan orientación sobre proyectos, programas y carteras.
4) **Cuantitativamente gestionado.** Medido y controlado. La organización está basada en datos con objetivos de mejora del rendimiento que son predecibles y se alinean con satisfacer las necesidades de los interesados internos y externos.
5) **Optimizado.** Estable y flexible. La organización se enfoca en la mejora continua y está diseñado para pivotar y responder a las oportunidades y al cambio. La organización de la estabilidad proporciona una plataforma para la agilidad y la innovación.

Modelos CMMI

CMMI brinda 4 modelos para distintos tipos de organizaciones que tienen por objetivo indicar los puntos clave a mejorar para que crezca el rendimiento, la calidad y la rentabilidad en las mismas [1].

Figura 2. CMMI Models, cmmiinstitute.com [Internet] CMMI® Institute LLC, 2017 [citado 13 de Noviembre de 2017] Disponible en: http://cmmiinstitute.com/capability-maturity-model-integration

A lo largo del desarrollo de este proyecto de grado se tendrá en cuenta el modelo "CMMI for Development" que provee los siguientes beneficios [2]:

Figura 3. You need CMMI for development, cmmiinstitute.com [Internet] CMMI® Institute LLC, 2017 [citado 13 de Noviembre de 2017] Disponible en: http://cmmiinstitute.com/sites/default/files/resource_asset/Which_Model_2017.pdf

"El modelo CMMI-DEV proporciona una guía para mejorar la capacidad de la organización para desarrollar productos de calidad y servicios que satisfacen las necesidades de los clientes y usuarios finales.

Estas mejores prácticas ayudarán a su organización a mejorar eficiencia, velocidad y calidad del producto impulsada por un menor número de defectos" 2.

Áreas de Proceso

Cada modelo CMMI consta de 16 áreas de proceso principales que contienen los conceptos necesarios para mejorar los procesos de las áreas de interés (adquisición, desarrollo, servicios, personas).

El modelo CMMI-DEV contiene 22 áreas de proceso [3]:

Process Area	Categoria	Nivel de Madurez
Análisis y resolución causal (CAR)	Soporte	5
Gestión de configuración (CM)	Soporte	2
Análisis y resolución de decisiones (DAR)	Soporte	3
Gestión integrada de proyectos (IPM)	Project Management	3
Medición y análisis (MA)	Soporte	2
Definición del proceso organizacional (OPD)	Project Management	3
Enfoque del proceso organizacional (OPF)	Project Management	3
Gestión del rendimiento organizacional (OPM)	Project Management	5
Desempeño del proceso organizacional (OPP)	Project Management	4
Entrenamiento Organizacional (OT)	Project Management	3
Integración de producto (PI)	Ingenieria	3
Monitoreo y control del proyecto (PMC)	Project Management	2
Planificación de proyectos (PP)	Project Management	2
Aseguramiento de calidad de proceso y producto (PPQA)	Soporte	2
Gestión cuantitativa de proyectos (QPM)	Project Management	4
Desarrollo de requisitos (RD)	Ingenieria	3
Gestión de requisitos (REQM)	Project Management	2
Gestión de riesgos (RSKM)	Project Management	3
Gestión de acuerdos con proveedores (SAM)	Project Management	2
Solución técnica (TS)	Ingenieria	3
Validación (VAL)	Ingenieria	3
Verificación (VER)	Ingenieria	3

Figura 4. Process Areas, Categories, and Maturity Levels, CMMI Product Team. CMMI® for Development. Versión 1.3. Estados Unidos: Carnegie Mellon; 2010.

Áreas de proceso de Ingeniería

Estas áreas de proceso se encargan de todo lo referente a las actividades de mantenimiento y desarrollo intervienen en todos los ámbitos de la ingeniería, para facilitar su utilización en los ámbitos técnicos involucrados en los procesos de desarrollo de producto se utilizó terminología de ingeniería general para escribir estas áreas de proceso [3].

Los procesos de distintas áreas de ingeniería junto con estas áreas de proceso forman un único proceso de desarrollo de productos que apoya la estrategia de mejora de productos orientada al producto dirigida a objetivos comerciales [3].

En CMMI-DEV hay 5 áreas de proceso [3]:

- Integración de Producto (PI)
- Desarrollo de requisitos (RD)
- Solución técnica (TS)
- Validación (VAL)

- Verificación (VER)

Abajo se puede ver la interacción de las áreas de ingeniería [3]:

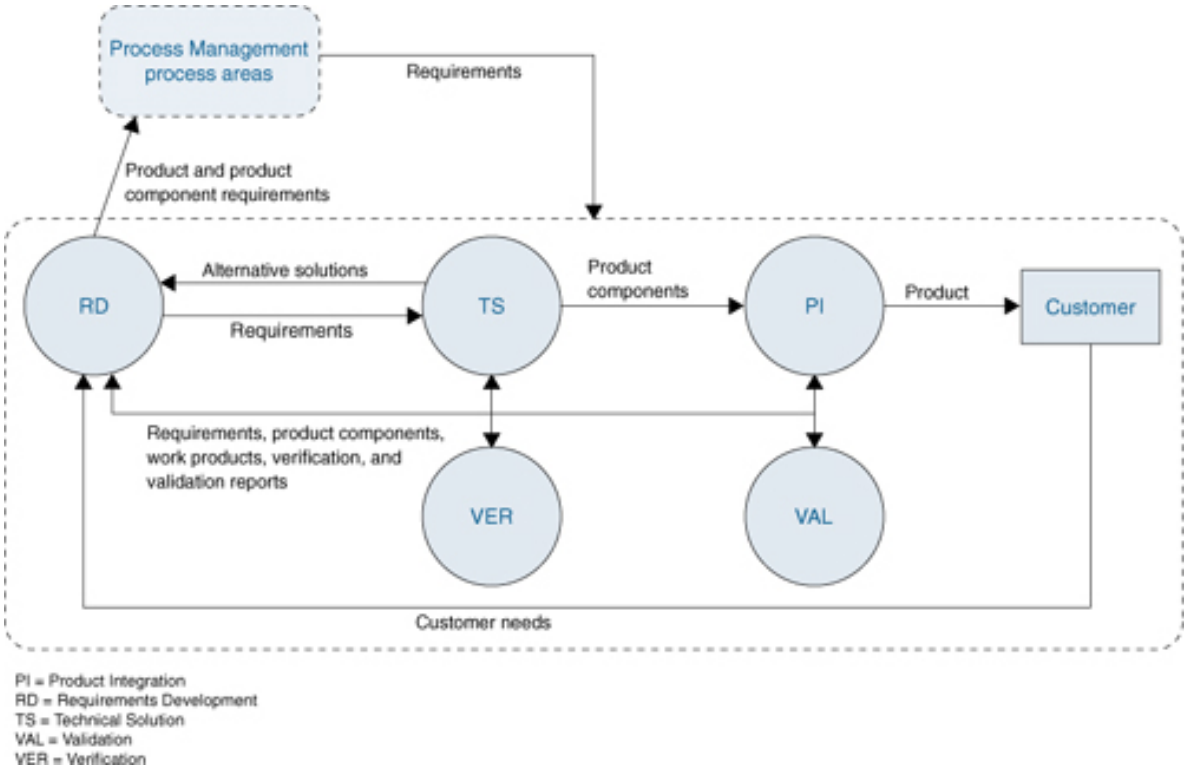

Figura 5. Engineering Process Areas, Categories, and Maturity Levels, CMMI Product Team. CMMI® for Development. Versión 1.3. Estados Unidos: Carnegie Mellon; 2010.

Integración de productos - Nivel de madurez 3

La integración de productos (PI) tiene como fin unir los componentes de un producto acoplando los mismos para crear el producto y que el mismo tenga el comportamiento deseado así como también la entrega de este [3].

Esta área de proceso busca ensamblar completamente el producto mediante una o varias etapas incrementales de ensamblaje de sus componentes, siguiendo procedimientos y estrategias predefinidas. Además de las tareas de ensamblaje se debe prestar especial atención para la integración del producto a la gestión de las interfaces externas e internas con las cuales el producto interactuara [3].

PRÁCTICAS ESPECÍFICAS POR META

Meta Específica 1: Preparación para la integración de productos.
Se refiere a creación de la estrategia de integración, el entorno en donde se procederá a integrar los componentes y fijar criterios y procedimientos de dicha integración 3.

Práctica Específica 1.1 Establecer una estrategia de integración.

Además de la estrategia de integración se detallará un tratamiento para la recepción, ensamblaje y evaluación de los componentes del producto 3.

El plan de integración de productos contiene el desarrollo de la estrategia de integración de los productos, este plan es inspeccionado, comprendido y luego aceptado por todas las partes interesadas [3].

Práctica Específica 1.2 Establecer el entorno de integración de productos.

Establecimiento y mantenimiento del entorno donde se realizará la integración de los componentes [3].

Será necesario contar con los requisitos para la compra o desarrollo de los recursos (hardware, software, etc.) necesarios para el entorno, el mismo puede ser desarrollado por la organización o bien adquirido total o parcialmente (ya que se pueden reutilizar recursos preexistentes)[3].

Práctica Específica 1.3 Constituir procedimientos y criterios de integración de productos.

Se constituirán y mantendrán procedimientos y principios de integración de componente. En los procedimientos se pueden detallar las iteraciones incrementales, detalles de cómo realizar las pruebas, resultados esperados de pruebas, verificación de los componentes y comportamientos esperados, preparación de los componentes, detalles de admisibilidad de los mismos, etc. [3].

Meta Específica 2: Garantizar compatibilidad de interfaz.
Debe haber una administración de requisitos, especificaciones y diseños de interfaz eficaz para asegurarse que los productos y las interfaces (internas y externas) sean completamente compatibles y evitar problemas de integración 3.

Práctica Específica 2.1 Revisión de las descripciones de la interfaz.

Se deberán revisar las descripciones de las interfaces de componentes del producto así como también las interfaces del entorno de integración de este [3].

Práctica Específica 2.2 Gestionar interfaces.

La gestión de las interfaces internas y externas incluye las definiciones, diseños, cambios y mantenimiento de la consistencia de las mismas a lo largo de la vida del producto. Hay que tener en cuenta que las modificaciones en las interfaces afectan tanto a los componentes del producto como a sistemas externos y entornos de verificación y validación 3.

Meta Específica 3: Ensamblar los componentes del producto y entregar el producto.

Los componentes que pasaron la verificación se ensamblan en componentes más grandes y complejos y luego estos se verifican para comprobar su correcta interacción con el resto de los componentes. Este proceso se repite hasta que todos los componentes están ensamblados y el producto esté totalmente integrado. Una vez completado el proceso de integración del producto, se valida su funcionamiento total y se entrega integrado, verificado y validado [3].

Práctica Específica 3.1 Confirmar la disponibilidad de los componentes del producto para la integración.

Antes de ensamblar el producto se deberá confirmar que sus componentes y las interfaces de estos tienen el comportamiento esperado en base a su descripción. Asegurándose de esta manera no haya problemas más adelante cuando se integren en componentes más grandes y los mismos tengan que interactuar al ser ensamblados (de acuerdo con los procedimientos y estrategias) entre sí para formar el producto final [3].

Práctica Específica 3.2 Montar los componentes del producto.

Las actividades de montar los componentes se realizan iterativamente empezando desde los componentes iniciales, ensamblajes intermedios y terminando con el producto final [3].

Práctica Específica 3.3 Evaluar componentes de productos ensamblados.

Esta evaluación comprende verificar los componentes ensamblados y determinar si su comportamiento se condice con las especificaciones de los mismos y su compatibilidad con la interfaz es la esperada. De la evaluación se obtiene el

rendimiento, idoneidad y preparación de los componentes. Según los procedimientos y la estrategia de integración del producto se puede definir el momento en el que se realizarán las pruebas, pudiendo ser una por cada integración, una sola al final, etc. [3].

Práctica Específica 3.4 Empaquetar y entregar el producto o componente del producto.

El producto puede ser empacado totalmente ensamblado o puede ser un componente del producto el que sea ensamblado y enviado al cliente. Algunos productos pueden tener requisitos de empaque especiales detallados en sus especificaciones y criterios de evaluación. También se detallarán los ajustes requeridos para unir los componentes del producto en el sitio operacional en caso de que sean diferentes a los ajustes para unir los componentes en la fábrica [3].

Desarrollo de requisitos - Nivel de madurez 3

El objetivo del Desarrollo de Requisitos (RD) es obtener y analizar los requisitos del cliente y del producto [3].

Esta área de proceso describe tres tipos de requisitos: requisitos del cliente, requisitos del producto y requisitos del componente del producto. En conjunto tienen en cuenta las necesidades de los interesados incluidas aquellas relacionadas con las fases del ciclo de vida del producto y con los atributos del producto. Los requisitos también incluyen las limitaciones causadas por la selección de soluciones de diseño. Todos los proyectos de desarrollo tienen requisitos, estos son la base del diseño [3].

Los requisitos se identifican y depuran durante las fases del ciclo de vida del producto. Las decisiones de diseño, las posteriores correcciones y los comentarios se analizan para determinar el impacto en los requisitos derivados [3].

El área de "Desarrollo de requisitos" tiene tres objetivos específicos desarrollar requisitos del cliente, desarrollar requisitos del producto y analizar y validar requisitos. Las prácticas específicas de la tercera meta específica están destinadas a ayudar a las prácticas específicas en los primeros dos objetivos específicos. Los procesos asociados con el área de proceso de Desarrollo de Requisitos y los procesos

asociados con el área de proceso de Solución Técnica pueden interactuar recursivamente entre sí [3].

Los análisis se utilizan para comprender, definir y seleccionar los requisitos en todos los niveles de las alternativas de la competencia. Algunos atributos de calidad surgirán como arquitectónicamente significativos y, por lo tanto, impulsarán el desarrollo de la arquitectura [3].

Estos análisis ocurren recursivamente en capas sucesivamente más detalladas de la arquitectura del producto hasta que haya suficientes detalles disponibles para permitir que el diseño detallado, la adquisición y las pruebas del producto continúen. Como resultado del análisis de los requisitos y el concepto operacional, el concepto de fabricación o producción produce más requisitos derivados [3].

Una jerarquía de entidades lógicas se establece con el concepto operativo en evolución. Los requisitos se refinan, derivan y asignan a estas entidades lógicas. Los requisitos y las entidades lógicas se asignan a productos, componentes de productos, personas o procesos asociados. En el caso de iterativo o desarrollo incremental, los requisitos también se asignan a iteraciones o incrementos [3].

Se implica a los interesados que corresponde en el desarrollo de ambos requisitos y el análisis les da visibilidad sobre la evolución de los requisitos. El estar involucrados continuamente les asegura que los requisitos están siendo definidos correctamente [3].

PRÁCTICAS ESPECÍFICAS POR META

Meta específica 1: Desarrollar los requisitos del cliente.

Las necesidades y expectativas de los interesados son la base para determinar los requisitos del cliente [3].

Con frecuencia estas necesidades están mal identificadas o son conflictivas. Para comprender claramente las necesidades, expectativas y limitaciones de las partes interesadas se utiliza un proceso iterativo a lo largo del proyecto. Para facilitar este proceso, un sustituto del usuario final o cliente se involucra de forma cercana para representar sus necesidades y así ayudar a resolver los conflictos que vayan surgiendo [3].

Práctica Específica 1.1 Obtener las necesidades.

Se recopilan los requisitos solicitados y aquellos no proporcionados por los clientes de manera explícita. Los requisitos deben abordar además el impacto en el producto [3].

Práctica Específica 1.2 Transformar las necesidades de las partes interesadas en los requisitos del cliente.

Se consolidan y completan los aportes relacionados a las necesidades resolviendo los posibles conflictos existentes. En algunas situaciones, el cliente define ciertos requisitos, o los requisitos existen como un resultado de un proyecto anterior aquí podrían presentarse conflictos que deben resolverse de manera apropiada [3].

Los requisitos que aquí se definen deben incluir funciones comerciales y técnicas. Los requisitos del cliente son el resultado de decisiones informadas sobre el negocio, así como los efectos técnicos de sus requisitos [3].

***Meta específica 2*:** Desarrollar los requisitos del producto.

Se analizan los requisitos del cliente para obtener requisitos más detallados llamados "requisitos de productos y componentes del producto". Estos incluyen las necesidades asociadas a cada fase del ciclo de vida del producto. Los requisitos se vuelven a examinar con cada conjunto de requisitos y arquitectura sucesivos de menor nivel y se refinan [3].

La trazabilidad de los requisitos para funciones, objetos, pruebas, problemas u otras entidades está documentada. Los requisitos y funciones asignados son la base para la síntesis de la solución técnica; sin embargo, a medida que la arquitectura se define sirve como la base para nuevos requisitos. A medida que se desarrollan los componentes internos, se definen interfaces adicionales y se establecen requisitos de interfaz [3].

Práctica Específica 2.1 Establecer requisitos de componentes de productos y productos.

Los requisitos funcionales y de calidad del cliente se pueden expresar de una manera no técnica. Los requisitos del producto son la expresión de estos, en términos técnicos que luego pueden usarse para decisiones de diseño [3].

La modificación de los requisitos debido a cambios de requisitos aprobados está cubierta por el aspecto "mantener" de esta práctica específica; mientras que la administración de los cambios de requisitos está cubierta por el área de proceso de "Gestión de requisitos" [3].

Práctica Específica 2.2 Asignación de los requisitos del componente del producto.

La arquitectura del producto brinda la base para asignar los requisitos del producto a los componentes del producto [3].

Si un requisito especifica un atributo que será responsabilidad de más de un componente del producto, el atributo puede ser particionado para la asignación única a cada componente del producto como requisito derivado, sin embargo, otras veces el requisito compartido debería en su lugar ser asignado directamente a la arquitectura [3].
Este concepto de requisitos compartidos puede extenderse a otros atributos de calidad arquitectónicamente significativos [3].

Práctica Específica 2.3 Identificar los requisitos de interfaz.

Interfaces entre funciones (o entre objetos u otras entidades lógicas) son identificados. Las interfaces pueden impulsar el desarrollo de soluciones alternativas descrito en el área de proceso de Solución Técnica [3].

Se definen los requisitos de interfaz entre productos o componentes de productos identificados en la arquitectura del producto. Ellos están controlados como parte de integración de productos y componentes de productos y son una parte integral de la definición de arquitectura [3].

Meta específica 3: Analizar y validar los requisitos.

Abarca el análisis y la validación de los requisitos con respecto al entorno previsto del usuario final [3].

Los análisis se realizan para determinar qué impacto tendrá el entorno operativo previsto en la capacidad de satisfacer las necesidades, expectativas, limitaciones e interfaces de los interesados. Los atributos de calidad arquitectónicamente significativos se identifican en función de los controladores de la misión y del negocio. También se establece una definición de funcionalidad requerida y atributos de calidad.

Se consideran todos los modos de uso especificados para el producto.[3] Los objetivos de los análisis son determinar los requisitos de los candidatos para los conceptos del producto y luego convertir estos conceptos en requisitos. Paralelamente a esta actividad, los parámetros que se utilizarán para evaluar la efectividad del producto se determinan con base en la opinión del cliente y el concepto preliminar del producto [3].

Los requisitos se validan para aumentar la probabilidad de que el producto resultante funcione según lo previsto en el entorno de uso [3].

Práctica Específica 3.1 Establecer Conceptos y Escenarios Operacionales.

Un escenario es una secuencia de eventos que pueden ocurrir en el desarrollo, uso o mantenimiento del producto, que se utiliza para hacer explícitas algunas de las necesidades funcionales o de atributos de calidad de los interesados. En cambio, un concepto operacional para un producto generalmente depende tanto de la solución de diseño como del escenario. Los conceptos operativos se refinan a medida que se toman las decisiones de la solución y se desarrollan requisitos detallados de menor nivel.[3]

Así como una decisión de diseño para un producto puede convertirse en un requisito para un componente del producto, el concepto operacional puede convertirse en los escenarios (requisitos) para los componentes del producto. Los conceptos y escenarios operativos documentan la interacción de los componentes del producto con el entorno, los usuarios finales y otros componentes del producto, independientemente de la disciplina de ingeniería. Deben documentarse para todos los modos y estados dentro de las operaciones, el desarrollo del producto, la implementación, la entrega, el soporte, la capacitación y la eliminación [3]. Los escenarios se pueden desarrollar para abordar secuencias operativas, de sostenimiento, desarrollo u otras secuencias de eventos [3].

Práctica Específica 3.2 Establecer una definición de funcionalidad requerida y atributos de calidad.

Analizar los escenarios usando un "análisis funcional" para describir lo que se pretende que el producto haga. La descripción resultante de funciones, agrupaciones

lógicas de funciones y su asociación con requisitos se denomina arquitectura funcional [3].

Algunos atributos de calidad surgirán como arquitectónicamente significativos y, por lo tanto, impulsarán el desarrollo de la arquitectura del producto. Una comprensión clara de los atributos de calidad y su importancia en función de las necesidades de la misión o del negocio es un muy importante para el proceso de diseño [3].

Práctica Específica 3.3 Analizar requisitos.

Teniendo en cuenta el concepto operacional y los escenarios, los requisitos para un nivel del producto se analizan para determinar si son necesarios y suficientes para cumplir los objetivos de niveles más altos del producto. Los requisitos analizados proporcionan la base para requisitos más detallados y precisos para niveles inferiores de la jerarquía de productos [3].

A medida que se definen los requisitos, debe entenderse su relación con los requisitos de nivel superior y la definición de nivel superior de la funcionalidad requerida y los atributos de calidad [3].

Práctica Específica 3.4 Analizar los requisitos para lograr el equilibrio.

Las necesidades y limitaciones de los interesados pueden incluir aspectos tales como el costo, el cronograma, el rendimiento del producto o proyecto, la funcionalidad, las prioridades, los componentes reutilizables, la capacidad de mantenimiento o el riesgo [3].

Práctica Específica 3.5 Valida los requisitos.

La validación de los requisitos se realiza al principio del esfuerzo de desarrollo con los usuarios finales para ganar la confianza de que los requisitos son capaces de guiar un desarrollo que dé como resultado una validación final exitosa. Esta actividad debe integrarse con las actividades de gestión de riesgos. Las organizaciones maduras generalmente realizarán la validación de los requisitos de una manera más sofisticada utilizando múltiples técnicas y ampliarán la base de la validación para incluir otras necesidades y expectativas de los interesados [3].

Solución Técnica - Nivel de madurez 3

En el área de solución técnica (TS) se seleccionará, diseñará e implementar la solución que se aplicará en función de los requisitos, diseños e implementaciones con los cuales se crearán los productos [3].

Además de aplicar las prácticas específicas al producto y sus componentes también se aplicarán a los procesos del ciclo de vida del producto. Se aplican las prácticas específicas tanto para modificar procesos existentes como para crear nuevos [3].

PRÁCTICAS ESPECÍFICAS POR META

Meta específica 1: Seleccione Soluciones de Componente de Producto.

Antes de elegir una solución se comparan y evalúan otras soluciones alternativas y las ventajas relativas de estas, teniendo en cuenta las opciones arquitectónicas y los atributos de calidad. También se tienen en cuenta componentes de productos que pueden conseguirse en el mercado evaluando los beneficios y contras [3].

Generalmente las soluciones para cada componente se definen cuando se define toda la capa de componentes que estarán en el conjunto. El fin de esto es mejorar el rendimiento del conjunto como un todo y no como cada componente individual [3].

Práctica Específica 1.1 Desarrollar soluciones alternativas y criterios de selección.

Se desarrollan soluciones alternativas con el fin de poder elegir la solución que mejor aplique al producto teniendo en cuenta diferentes criterios de selección como costos, cronograma, desempeño y riesgo así como también calidad. Las soluciones alternativas muchas veces pueden ser componentes de producto disponibles en el mercado que apliquen mejor a los criterios de selección que los desarrollados internamente [3].

Práctica Específica 1.2 Seleccione las soluciones de componentes del producto.

Se seleccionarán las soluciones de componentes que mejor cumplan con los criterios de requisitos del producto. Se dejará documentada la descripción de las soluciones así como también la razón por la cual se seleccionó dicha solución [3].

Meta específica 2: Desarrollar el diseño.

Es necesario que el diseño de los productos y sus componentes además de contener la información necesaria para la implementación contenga también información para otras fases como modificación, reprocesamiento, mantenimiento e instalación. Esta documentación no solo brinda un entendimiento del diseño sino que será de ayuda para los futuros cambios en el diseño [3].

Práctica Específica 2.1 Diseñar el producto o componente del producto

El diseño del producto o componente del producto se divide en 2 fases, preliminar detallado. En la fase preliminar se constituyen las capacidades y la arquitectura del producto. Por otro lado, en la fase detallada se determinan completamente la estructura y capacidades de los componentes del producto [3].

Práctica Específica 2.2 Establecer un paquete de datos técnicos.

Al establecer un paquete de datos técnicos se está brindando una descripción detallada del producto o componente del producto en el momento que se desarrolla. Aquí se guardan detalles esenciales del diseño, la definición de la configuración, procedimientos, datos técnicos, dibujos, especificaciones, descripciones de diseño, etc. también incluye una solución alternativa preseleccionada [3].

Práctica Específica 2.3 Diseño de interfaces usando criterios.

Los criterios utilizados para el diseño de interfaces suelen reflejar parámetros que necesitan ser definidos para asegurarse que sean aplicables. Estos parámetros son característicos de un tipo de producto y normalmente asociados a seguridad, durabilidad y misión crítica [3].

Práctica Específica 2.4 Realizar análisis de fabricación, compra o reutilización.

En este análisis que comienza en etapas tempranas del proyecto se determina cuáles son los productos o componentes del producto que se compraran, cuales se fabricarán y cuáles serán productos existentes que se utilizaran todo esto se decidirá en base a las necesidades del proyecto [3].

Meta Específica 3: Implementar el diseño del producto.

La implementación del componente del producto se realiza una vez que se completa el diseño. Esta actividad incluye pruebas unitarias antes de enviar el componente a ser integrado con el producto y realizar la documentación de usuario [3].

Práctica Específica 3.1 Implementar el diseño.

Al finalizar el diseño se implementa el mismo se implementa como componente del producto. Cada componente del producto debe estar especificado en los niveles inferiores para poder implementar el diseño en el nivel de jerarquía superior del producto [3].

Práctica Específica 3.2 Desarrollar documentación de soporte del producto

La documentación de instalación, operación y mantenimiento de producto será desarrollada en esta práctica [3].

Validación - Nivel de madurez 3

"El propósito de la Validación (VAL) es demostrar que un producto o componente del producto cumple su uso previsto cuando se coloca en su entorno previsto" [3].

La validación es aplicable a todos los aspectos del producto en cualquiera de sus entornos previstos [3].

El entorno de validación debe representar el entorno previsto para el producto y los componentes del producto, así como representar el entorno previsto adecuado para las actividades de validación con productos de trabajo [3].

La validación asegura que lo que se construyó fue lo que se solicitó, lo que se esperaba. A diferencia de la verificación, la cual asegura que fue bien construido. Las actividades de validación utilizan enfoques similares a la verificación. Ambas actividades de validación y verificación a menudo se ejecutan simultáneamente y pueden usar partes del mismo entorno. Algunos problemas de validación pueden descubrirse temprano en la vida del proyecto utilizando productos de trabajo al involucrar a las partes interesadas relevantes [3].

Cuando se identifican problemas de validación, se refieren a procesos asociados con el Desarrollo de Requisitos, Solución Técnica, o Área de proceso de Monitoreo y Control del proyecto para su resolución [3].

PRÁCTICAS ESPECÍFICAS POR META

Meta específica 1: Prepárese para la validación.

La preparación incluye seleccionar productos y componentes de productos para la validación y establecer y mantener el entorno de validación, procedimientos y criterios. Cualquier producto o componente del producto puede estar sujeto a validación [3].

El entorno requerido para validar el producto o componente del producto es preparado (se puede comprar o se puede construir). Los entornos utilizados para la integración y verificación de productos se pueden considerar como ambiente de validación para reducir costos y mejorar la eficiencia o la productividad [3].

Práctica Específica 1.1 Seleccione productos para validación.

Los productos y los componentes del producto se seleccionan en función de su relación con las necesidades del usuario final. Para cada componente del producto, se debe determinar el alcance de la validación [3].

Primero se recopilan los requisitos y las restricciones para realizar la validación. Luego, los métodos de validación se seleccionan en función de su capacidad para demostrar que las necesidades del usuario final están satisfechas. Los métodos de validación también impulsan las necesidades de las instalaciones, equipo y ambientes. El enfoque de validación y las necesidades pueden resultar en la generación de requisitos de componentes de productos de menor nivel que son manejados por los procesos de desarrollo de requisitos [3].

Los métodos de validación deben seleccionarse temprano en la vida del proyecto para que sean entendidos y acordados por las partes interesadas pertinentes [3].

Práctica Específica 1.2 Establecer el entorno de validación.

Los requisitos para el entorno de validación dependen del producto o los componentes del producto seleccionados, por el tipo de productos de trabajo y por

los métodos de validación. Estas selecciones pueden generar requisitos para la compra o desarrollo de equipo, software u otros recursos. El entorno de validación puede incluir la reutilización de recursos existentes; sin embargo, se deben hacer arreglos para el uso de estos recursos [3].

El entorno de validación debe controlarse cuidadosamente para prever replicación, análisis de resultados y revalidación de áreas problemáticas [3].

Práctica Específica 1.3 Establecer procedimientos y criterios de validación.

Los procedimientos y criterios de validación se definen para garantizar que el producto o el componente del producto cumplirá con su uso previsto cuando se coloque en su entorno previsto. Los casos de prueba y los procedimientos para las pruebas de aceptación pueden ser utilizadas para procedimientos de validación [3].

Meta específica 2: Validar productos o componentes del producto

Los métodos de validación, procedimientos y criterios se utilizan para validar el producto seleccionado y componentes del producto y cualquier servicio asociado utilizando la validación adecuada. Las actividades de validación se realizan durante todo el ciclo de vida del producto [3].

Práctica Específica 2.1 Realizar validación.

Para ser aceptable para las partes interesadas, un producto o componente del producto debe funcionar como se espera en su entorno operativo previsto [3].

Las actividades de validación se realizan y los datos resultantes se recopilan. Los procedimientos de validación durante la ejecución deben documentarse y las desviaciones que se producen durante la ejecución deben anotarse según corresponda [3].

Práctica Específica 2.2 Analizar resultados de validación.

Los datos resultantes se analizan contra criterios de validación definidos. Los informes indican si se cumplieron las necesidades. En el caso de deficiencias, estos informes documentan el grado de éxito o fracaso y clasifican las causas probables de falla. Los resultados recopilados se comparan con los criterios de evaluación establecidos para determinar si procede o se abordan los requisitos o problemas de diseño [3].

Los informes de análisis o la documentación de validación también pueden indicar que los resultados de las pruebas incorrectas se deben a un problema en el procedimiento de validación o a un problema del entorno de validación [3].

Verificación - Nivel de madurez 3

El área de proceso de verificación implica: preparación de la verificación, verificación del rendimiento e identificación de la acción correctiva [3].

La verificación incluye la verificación del producto y de productos de trabajo intermedios contra todos los requisitos seleccionados [3].

La verificación es necesariamente un proceso incremental porque ocurre durante todo el desarrollo del producto y los productos de trabajo, comenzando con la verificación de los requisitos, progresando a través de la verificación de productos de trabajo en evolución y culminando en la verificación del producto terminado [3].

Las áreas del proceso de Verificación y Validación son similares, pero abordan diferentes problemas tal como se mencionó en el apartado anterior (4.1.4.4 Validación - Nivel de madurez 3). La verificación asegura que "lo construiste bien"; mientras que la validación asegura que "construiste lo correcto" [3].

Las evaluaciones por homólogos son una parte importante de la verificación y son un mecanismo comprobado para la eliminación efectiva de defectos [3].

Por otro lado, las revisiones interpares implican un examen metódico de los productos de trabajo por parte de compañeros de los productores para identificar los defectos y otros cambios que se necesitan [3].

PRÁCTICAS ESPECÍFICAS POR META

Meta específica 1: Prepárese para la verificación.

La preparación inicial es necesaria para garantizar que las disposiciones de verificación estén integradas en cada componente del proceso de desarrollo. La verificación incluye la selección, inspección, prueba, análisis y demostración de productos de trabajo [3].

Los métodos de verificación incluyen, entre otros, inspecciones, revisiones por pares, auditorías, etc. La preparación también implica la definición de herramientas de soporte, equipos de prueba y software, simulaciones, prototipos e instalaciones [3].

Práctica Específica 1.1 Seleccione los productos de trabajo para la verificación.

Serán seleccionados para la verificación los productos de trabajo que cumplan los objetivos y requisitos del proyecto y los que abordan riesgos para el mismo. Se incluirán los métodos de verificación junto con los requisitos a ser verificados [3].

Práctica Específica 1.2 Establecer el entorno de verificación.

El entorno para la verificación puede desarrollarse, adquirirse, reutilizarse, etc. en función de las necesidades del proyecto. El tamaño y tipo del entorno será en función a las verificaciones que se realizarán a los productos de trabajo y diferirá si son pruebas simples o de integración, etc. [3].

Práctica Específica 1.3 Establecer procedimientos y criterios de verificación.

Para asegurarse que los requisitos del producto de trabajo sean cumplidos se definirán los criterios de verificación [3].

Meta específica 2: Realizar revisiones por pares.

La revisión por pares consiste en una inspección del producto de trabajo realizada por un compañero de trabajo que realiza actividades similares a la del desarrollador del producto. Mediante esta inspección se encuentran y eliminan defectos y se proponen mejoras necesarias [3].

Práctica Específica 2.1 Prepararse para revisiones por pares.

Se identifica al personal que participara en la revisión por pares de los productos de trabajo y se actualizan los materiales utilizados en las revisiones por pares como listas de verificación y criterios de revisión, etc. [3].

Práctica Específica 2.2 Conducir las revisiones por pares.

Las revisiones por pares tienen como finalidad encontrar de manera temprana defectos y eliminarlos. Las mismas se realizan de manera gradual y están estructuradas. Si se detectan defectos a ser eliminados se debe contactar al desarrollador principal del producto de trabajo para que este los corrija [3].

Práctica Específica 2.3 Analizar datos de revisión por pares.
Se analizarán los datos obtenidos de la revisión por pares [3].

Meta específica 3 Verificar productos de trabajo seleccionados.

Para la verificación de los productos de trabajo seleccionados se utilizarán los métodos de verificación, los procedimientos y los criterios provistos. Estas verificaciones serán realizadas durante todo el ciclo de vida del producto [3].

Práctica Específica 3.1 Realizar verificación

Se ahorran considerablemente los costos de retrabajo al realizar la verificación de productos progresivamente ya que la misma facilita la detección y eliminación temprana de defectos [3].

Práctica Específica 3.2 Analizar los resultados de la verificación

Se analizarán y compararan los resultados de la verificación con los criterios de aceptabilidad de los mismos y se registraran como evidencia de la verificación. Todos los resultados son analizados de manera incremental para cada producto incluidos los resultados de la verificación por pares [3].

4.2 Proceso Unificado de Desarrollo de Software

El proceso unificado de desarrollo de software está dirigido por los casos de uso, centrado en la arquitectura, es iterativo e incremental. El mismo utiliza el Lenguaje Unificado de Modelado (UML). El proceso pone en práctica el basar gran parte del proyecto de desarrollo en componentes reutilizables [4].

El proceso unificado se repite a lo largo de una serie de ciclos que concluyen con una versión del producto, a su vez cada ciclo consta de 4 fases: Inicio, Elaboración, Construcción y transición [4]:

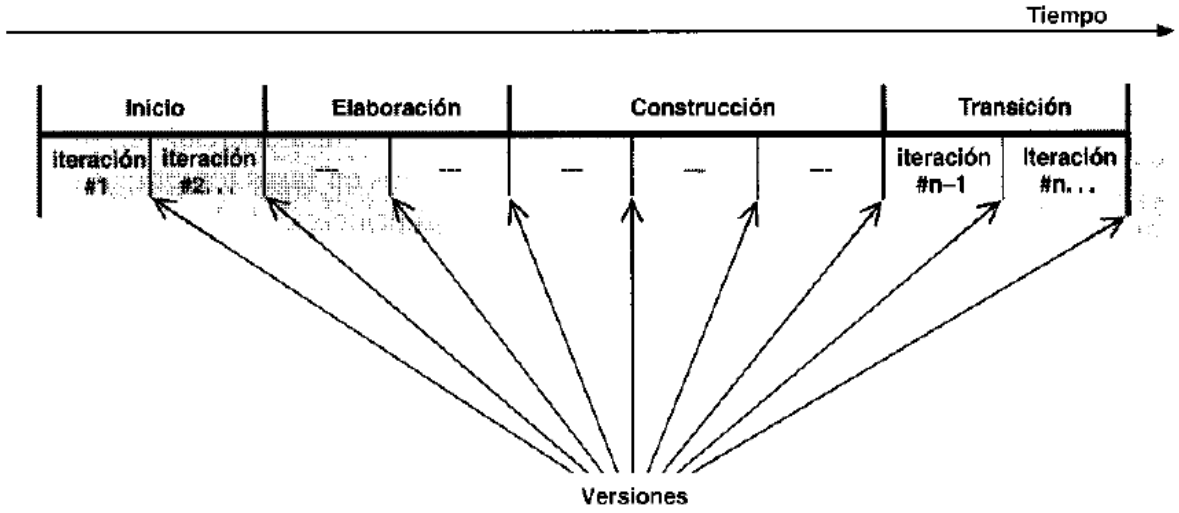

Figura 6. Un ciclo con sus fases e iteraciones, Ivar Jacobson, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

El producto terminado incluye además de los ejecutables, los requisitos, casos de uso, especificaciones no funcionales y casos de prueba, modelo de arquitectura y modelo visual y artefactos modelados con UML [4]:

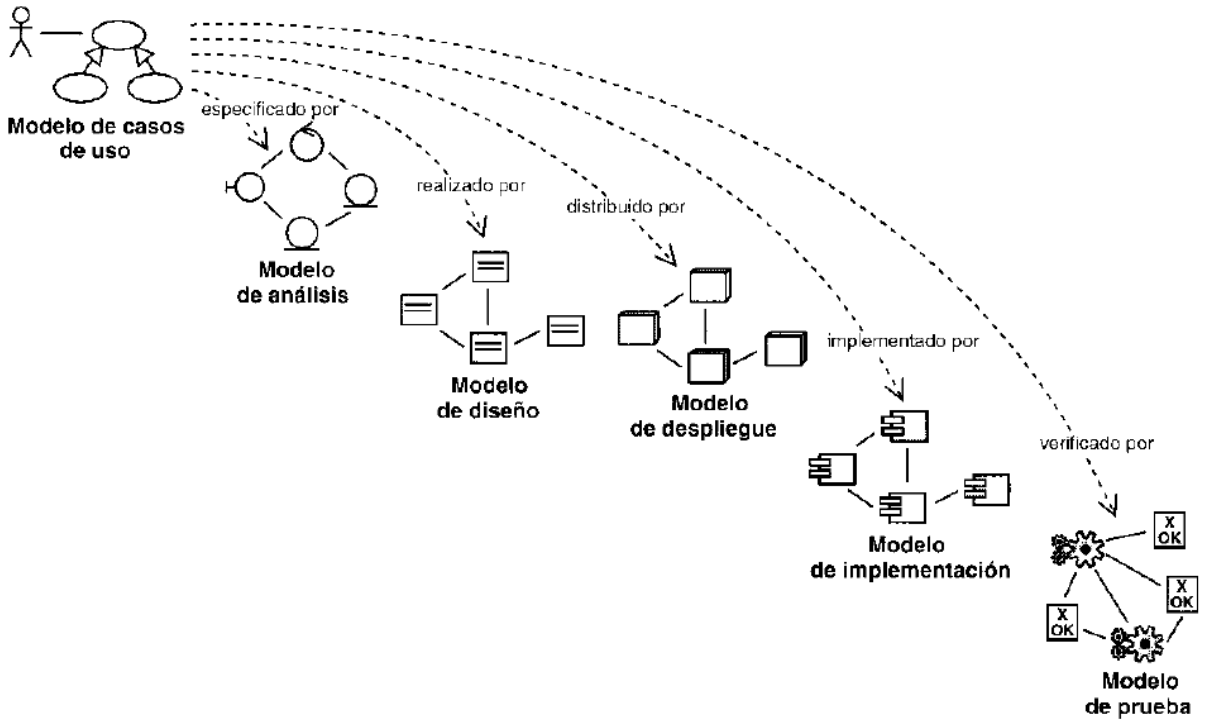

Figura 7. Modelo del proceso unificado, Ivar Jacobson, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

Cada ciclo además de tener sus 4 fases tiene 5 flujos de trabajo que se repiten en cada fase y ocupan distintas cantidades de tiempo [4]:

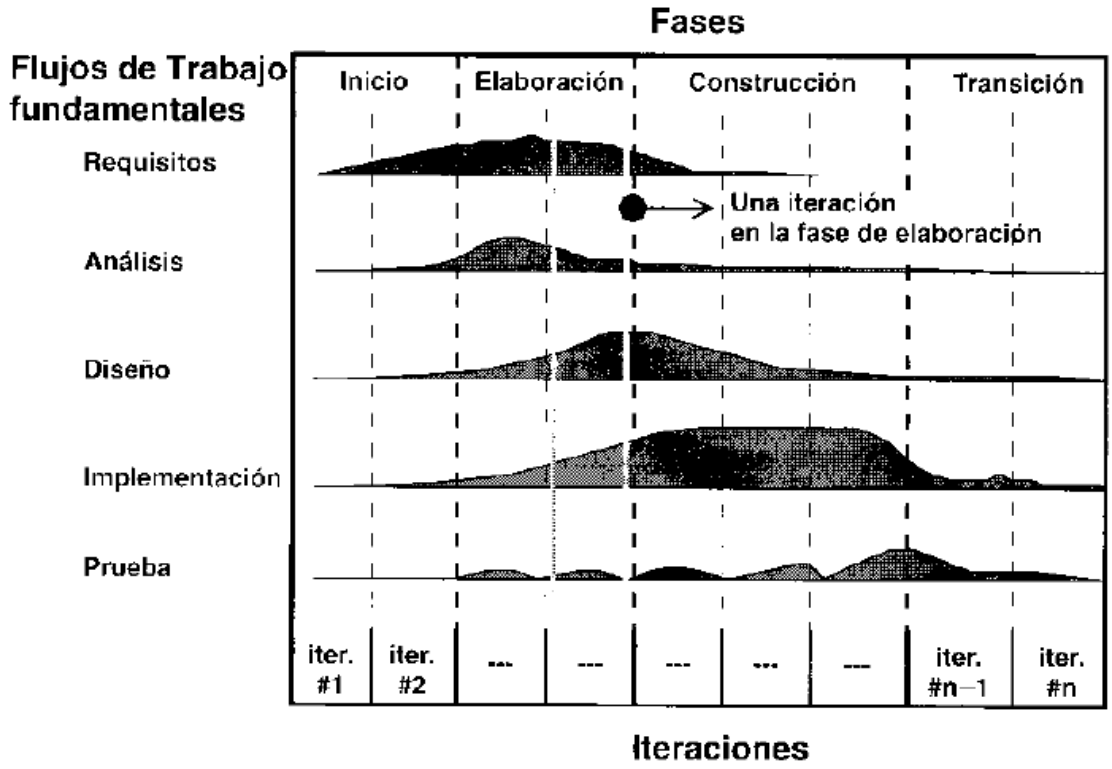

Figura 8. Los cinco flujos de trabajo, Ivar Jacobson, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

Las cuatro "P" en el desarrollo de software: Personas, Proyecto, Producto y Proceso [4].

Persona: arquitectos, desarrolladores, ingenieros de prueba, personal de gestión, usuarios y otros interesados.

Proyecto: elemento organizativo a través del cual se gestiona el desarrollo de software.

Producto: artefactos creados durante la vida del proyecto.

Proceso: conjunto de actividades necesarias para transformar los requisitos de usuario en un producto.

Un proceso dirigido por casos de uso.

Los casos de uso dirigen el proceso de desarrollo en su totalidad, son la entrada fundamental cuando se identifican clases, subsistemas e interfaces y nos guían a través del conjunto completo del flujo de trabajo [4]:

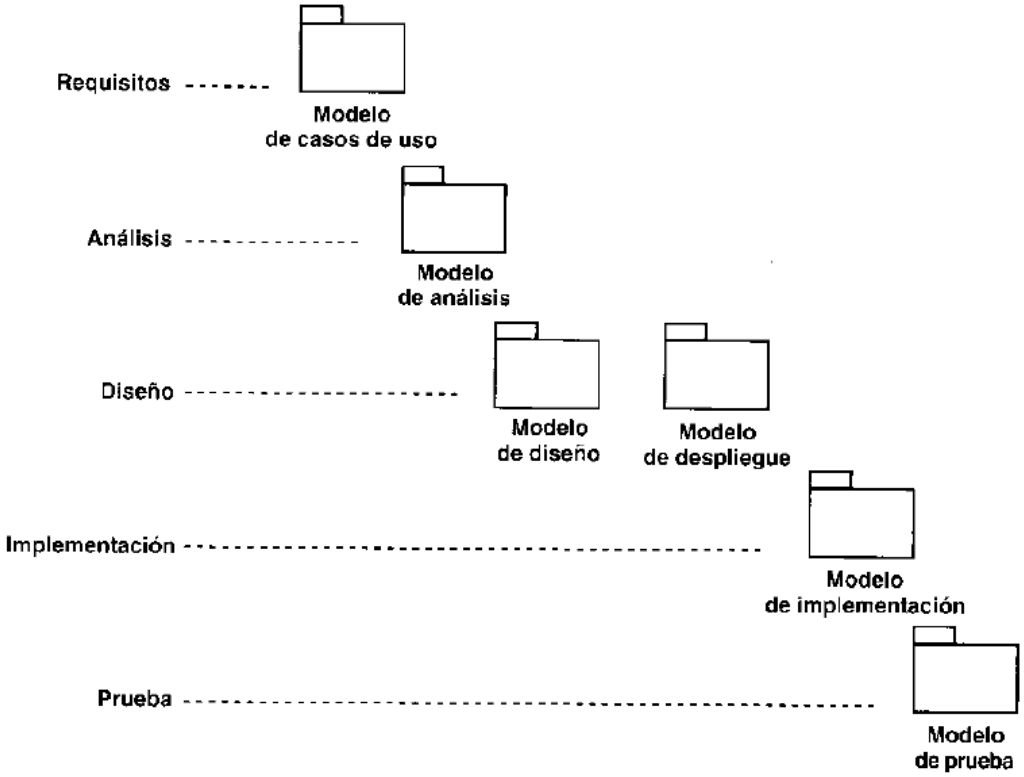

Figura 8. El proceso unificado consiste en una serie de flujos de trabajo que van desde los requisitos hasta las pruebas, Ivar Jacobson, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

Captura de requisitos: de la visión a los requisitos.

Para capturar los requisitos los analistas necesitan un conjunto de técnicas y artefactos que los ayuden a obtener una visión del sistema para avanzar con los flujos de trabajo subsiguientes, a estos artefactos se los llama colectivamente conjunto de requisitos. Los artefactos necesarios para establecer el contexto del sistema son el modelo de dominio y el modelo de negocio [4].

El modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos de dominio representan las cosas que existen o los eventos que suceden en el entorno que trabaja el sistema [4].

El modelo de negocio describe los procesos de negocio de una empresa en términos de casos de uso del negocio y actores del negocio que se corresponden con los procesos del negocio y los clientes, respectivamente. Este modelo presenta un sistema desde la perspectiva de su uso y esquematiza como proporcionar valor a sus usuarios [4].

Captura de requisitos como casos de uso.

Aquí se desarrolla el modelo de sistema que se va a construir. Los casos de uso proporcionan un medio intuitivo y sistemático para capturar los requisitos funcionales con un énfasis especial en el valor añadido para cada usuario individual o para cada sistema externo. Se describe el flujo de trabajo en 3 pasos [4] :

- Los artefactos del flujo de trabajo de los requisitos
- Los trabajadores participantes de este flujo
- El flujo de trabajo de captura de requisitos

A continuación, se pueden ver los actores y los artefactos [4] :

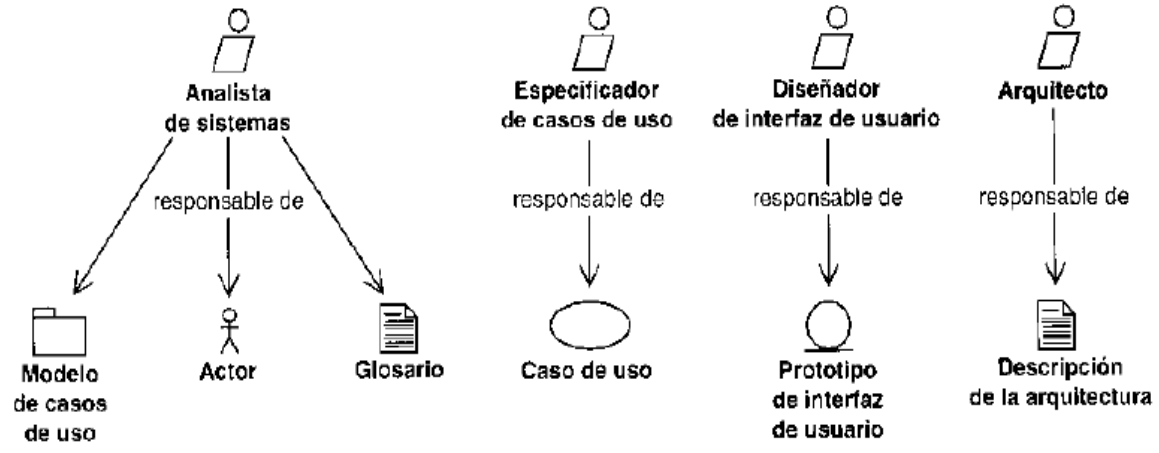

Figura 9. Los trabajadores y artefactos implicados durante la captura de requisitos mediante casos de uso, Ivar Jacobson, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

Y también el flujo de trabajo [4] :

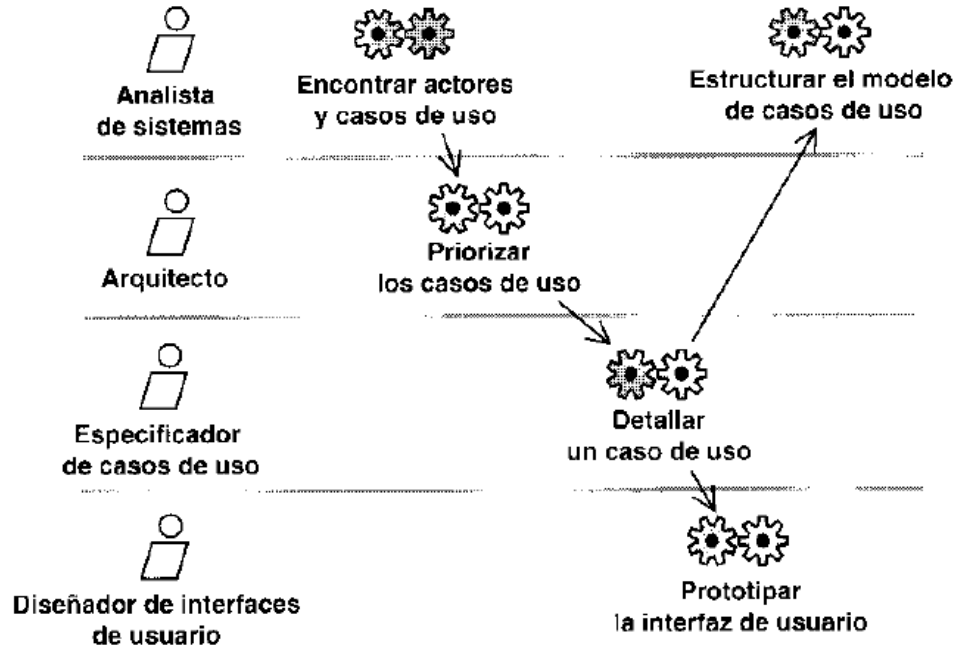

Figura 10. El flujo de trabajo para la captura de requisitos en forma de casos de uso, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

Análisis

El propósito fundamental del análisis es analizar los requisitos con mayor profundidad y resolver los siguientes temas:

- Mantener los casos de uso independientes unos de otros.
- Escribir los casos de uso utilizando el lenguaje del cliente.
- Estructurar cada caso de uso para que forme una especificación de funcionalidad completa e intuitiva [4] .

Pero la gran diferencia es que puede utilizarse el lenguaje de los desarrolladores para escribir los resultados, razonando más sobre los aspectos internos del sistema y resolviendo aspectos relativos a la interferencia de casos de uso [4] .

El análisis cuenta con los siguientes actores y artefactos [4] :

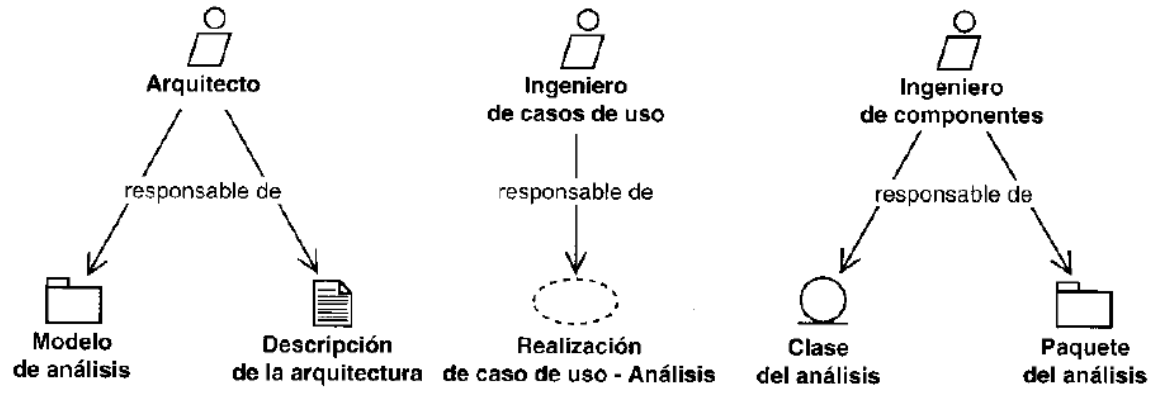

Figura 11. Los trabajadores y artefactos implicados en el análisis, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

El flujo de trabajo de análisis con los trabajadores participantes y sus actividades [4] :

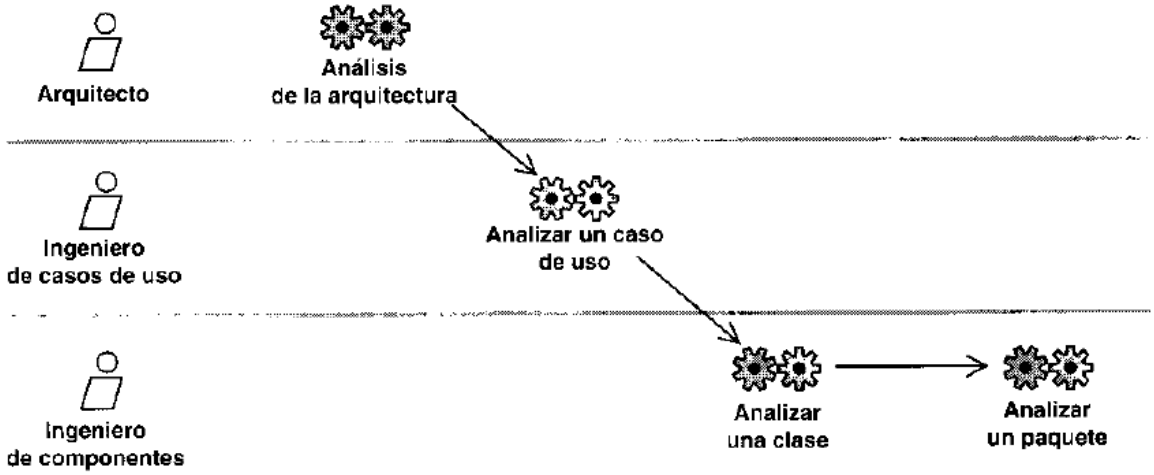

Figura 12. El flujo de trabajo en el análisis, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

Diseño.

En el diseño se modela el sistema y se encuentra su forma para que soporte todos los requisitos (funcionales y no funcionales) para esto utilizamos como entrada esencial el modelo de análisis que proporciona una comprensión detallada de los requisitos e impone una estructura del sistema que hay que esforzarse en conservar los más fielmente posible [4].

Los propósitos del diseño son :

- Adquirir una comprensión en profundidad de los aspectos relacionados con los no funcionales y las restricciones de los lenguajes de programación, componentes reutilizables, sistemas operativos, etc.
- Crear una entrada apropiada y punto de partida para actividades de implementación subsiguientes.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo [4].

Trabajadores y artefactos involucrados en el diseño [4] :

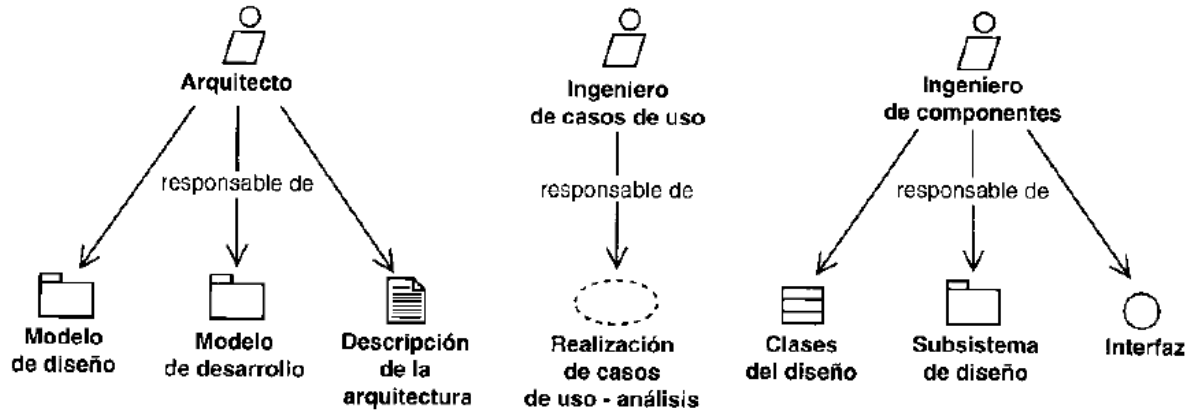

Figura 13. Trabajadores y artefactos involucrados en el diseño, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

Flujo de trabajo del diseño con sus participantes y actividades [4] :

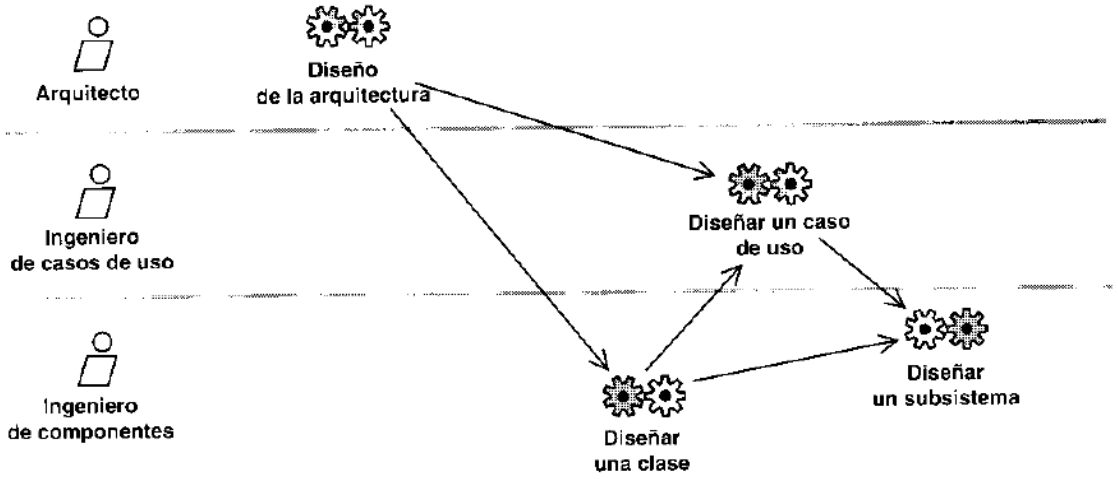

Figura 14. El flujo de trabajo en el diseño, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

Implementación [4].

En la implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes (código fuente, scripts, binarios, ejecutables, etc.). El propósito principal es desarrollar la arquitectura y el sistema como un todo, de forma más específica:

- Planificar las integraciones de sistema necesarias en cada iteración
- Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue.
- Implementar las clases y subsistemas encontrados durante el diseño.
- Probar los componentes individualmente y a continuación integrarlos compilarlos y enlazarlos en uno o más ejecutables antes de ser enviados a ser integrados y llevar a cabo las comprobaciones del sistema
- Trabajadores y artefactos de la implementación:

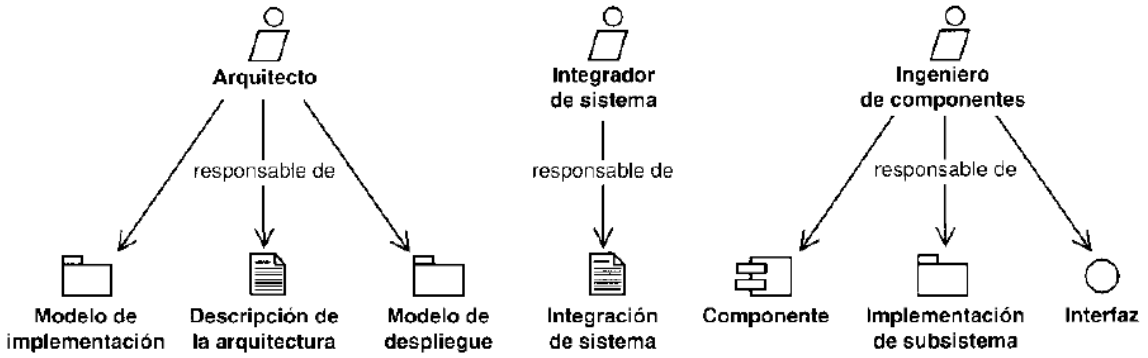

Figura 15. Los trabajadores y artefactos involucrados en la implementación, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

- Flujo de trabajo, participantes y actividades:

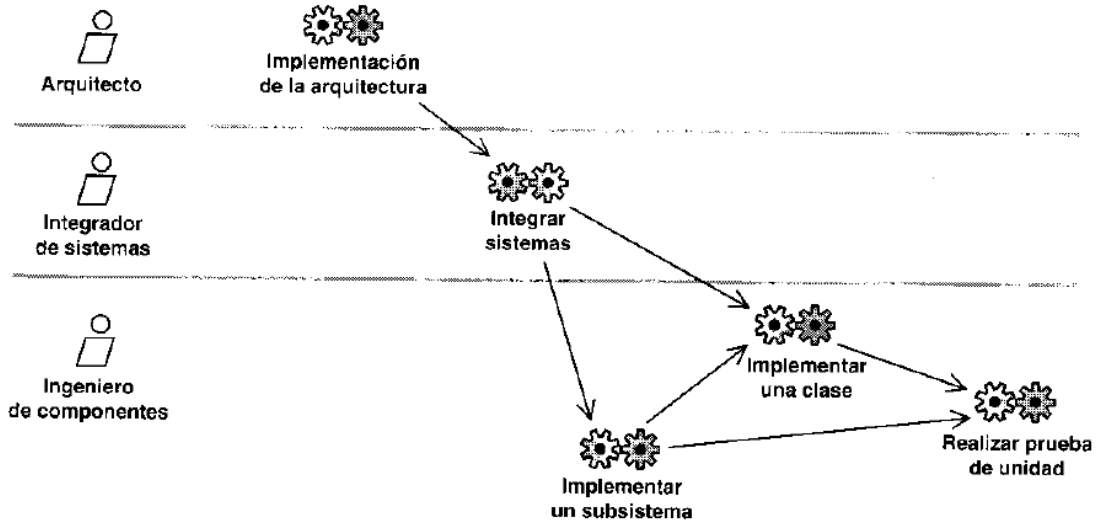

Figura 16. Flujo de trabajo de la etapa de implementación, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

4.3 Metodologías Ágiles

A fines de los 90, diversas metodologías con combinaciones diferentes de ideas nuevas y antiguas comenzaron a volverse el foco de atención. Estas hacían énfasis en una colaboración cercana entre el equipo de desarrollo y los interesados en el negocio, entregas frecuentes de valor, equipos auto organizados y formas inteligentes de crear y entregar código [5].

El término "ágil" se aplicó a esta colección de metodologías a principios de 2001 cuando un grupo de profesionales del desarrollo de software se reunieron en Snowbird, Utah, para analizar sus ideas compartidas y diferentes enfoques para el desarrollo de software [5].

Esta colección conjunta de valores y principios se expresó en el Manifiesto para el desarrollo de software ágil y los doce principios correspondientes [5].

Desde entonces las metodologías ágiles han evolucionado basándose en prácticas que probaron su éxito en pequeños equipos de desarrollo [5].

Según Paul Mc Mahon [6], los métodos ágiles más populares incluyen Scrum, Crystal, Extreme Programming y Agile Modeling.

Principios y Prácticas Ágiles

Los principios del Manifiesto Ágil [7] indican lo siguiente:

1) Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2) Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3) Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4) Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5) Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6) El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7) El software funcionando es la medida principal de progreso.
8) Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

9) La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10) La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11) Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12) A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

El Manifiesto Ágil [8] además identifica cuatro valores:

- Valoramos las personas y las interacciones sobre los procesos y herramientas.
- Valoramos el software en funcionamiento sobre la documentación.
- Valoramos la colaboración del cliente sobre la negociación del contrato.
- Valoramos la respuesta al cambio sobre seguir un plan.

También aclara en referencia a los cuatro valores, "Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda" [8].

4.3.2 Scrum

"Scrum es un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente."[9]

Scrum consiste en los equipos y sus roles, eventos, artefactos y reglas asociadas[9]. Tres **pilares** soportan toda la implementación del control de procesos empírico en el que se basa Scrum:

- **Transparencia:** Los aspectos más importantes del proceso deben ser visibles para los responsables del resultado. Estos aspectos deben seguir un estándar común para que todos los observadores lo comprendan de igual manera.

- **Inspección**: Se deben inspeccionar los artefactos de Scrum y el progreso hacia el objetivo para detectar variaciones, aunque estas no deben ser demasiado frecuentes ya que pueden interferir en el trabajo.

- **Adaptación**: Si se determina que uno o más aspectos de un proceso se desvían de límites aceptables y que el producto resultante será inaceptable, el proceso debe ajustarse cuanto antes para minimizar las desviaciones. Scrum prescribe cuatro eventos formales, contenidos dentro del Sprint, para la inspección y adaptación (se describen más adelante)[9].

Scrum define además **cinco valores**: compromiso, coraje, foco, apertura y respeto. Cuando el Equipo Scrum los incorpora y vivencia, los pilares se materializan y se obtiene un uso exitoso de Scrum [9].

El Equipo Scrum

El Equipo Scrum consiste en:

- ***Un Dueño de Producto*** *(Product Owner):* el responsable de maximizar el valor del producto y el trabajo del Equipo de Desarrollo. Es la única persona responsable de gestionar la Lista del Producto (Product Backlog).

- ***El Equipo de Desarrollo*** *(Development Team):* Los profesionales que realizan entregas incrementales de producto "Terminado" que se puede poner en producción al final de cada Sprint. Tienen las siguientes características: son auto organizados, son multifuncionales, Scrum no reconoce títulos (todos son Desarrolladores, independientemente del trabajo que realice cada persona), Scrum no reconoce subequipos en los equipos de desarrollo y la responsabilidad recae en el Equipo de Desarrollo como un todo (independientemente de las áreas en las que están enfocados). El tamaño óptimo del Equipo de Desarrollo es lo suficientemente pequeño como para permanecer ágil y lo suficientemente grande como para completar una cantidad de trabajo significativa.

- ***Un Scrum Master.*** Es el responsable de asegurar que Scrum se entienda y se adopte. Los Scrum Masters hacen esto asegurándose de que el Equipo Scrum trabaja ajustándose a la teoría, prácticas y reglas de Scrum [9].

Eventos de Scrum

El Sprint

Es un bloque de tiempo de un mes o menos en el que se crea un incremento de producto "Terminado" utilizado y potencialmente desplegable. Es más conveniente que la duración de los sprints sea consistente, cada nuevo Sprint comienza al finalizar el anterior [9].

Los Sprints contienen y consisten en la Planificación del Sprint (Sprint Planning), los Scrums Diarios (Daily Scrums), el trabajo de desarrollo, la Revisión del Sprint (Sprint Review), y la Retrospectiva del Sprint (Sprint Retrospective) [9].

Planificación de Sprint (Sprint Planning)

Aquí se planifica el trabajo a realizar durante el sprint de manera colaborativa del Equipo Scrum completo. La planificación puede durar un máximo de ocho horas para un sprint de un mes o menos tiempo para sprints más cortos. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito, además enseña al equipo a mantenerse dentro del bloque de tiempo [9].

Scrum Diario (Daily Scrum)

El Scrum Diario es una reunión de 15 minutos para que el Equipo de Desarrollo sincronice sus actividades y cree un plan para las siguientes 24 horas. Se realiza a la misma hora y en el mismo lugar todos los días para reducir la complejidad. Durante la reunión, cada miembro del Equipo de Desarrollo explica:

- ¿Qué hice ayer que ayudó al Equipo a lograr el Objetivo del Sprint?
- ¿Qué haré hoy para ayudar al Equipo a lograr el Objetivo del Sprint?
- ¿Veo algún impedimento que evite que el Equipo o yo logremos el Objetivo del Sprint?[9].

Revisión de Sprint (Sprint Review)

Se realiza al finalizar el Sprint para inspeccionar el Incremento y adaptar la Lista de Producto si fuese necesario. Durante la revisión el Equipo Scrum y los interesados

colaboran acerca de lo que se hizo durante el Sprint. Basándose en esto y en los cambios a la Lista de Producto que se hicieron durante el Sprint, los asistentes colaboran para determinar las cosas que podrían hacerse a continuación para seguir incrementando el valor [9].

Se trata de una reunión informal, no una reunión de seguimiento, y la presentación del Incremento tiene como objetivo facilitar la retroalimentación de información y fomentar la colaboración. Se trata de una reunión de no más de cuatro horas para Sprints de un mes o menos para Sprints más cortos [9].

Retrospectiva de Sprint (Sprint Retrospective)

La Retrospectiva de Sprint busca que el Equipo Scrum se inspeccione a sí mismo y cree un plan de mejoras a abordar el siguiente Sprint. Esta reunión se lleva a cabo después de la Revisión y antes de la siguiente Planificación, puede durar un máximo de tres horas para Sprints de un mes [9].

El propósito de la Retrospectiva de Sprint es:

- Inspeccionar cómo fue el último Sprint en cuanto a personas, relaciones, procesos y herramientas;
- Identificar y ordenar los elementos más importantes que salieron bien y las posibles mejoras; y,
- Crear un plan para implementar las mejoras a la forma en la que el Equipo Scrum desempeña su trabajo [9].

Artefactos de Scrum

Los artefactos de Scrum representan trabajo o valor en diversas formas[9].

Lista de Producto (Product Backlog)

Es una lista ordenada de todo lo que podría ser necesario en el producto y es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El Dueño de Producto (Product Owner) es el responsable de la Lista de Producto, incluyendo lo que contiene y como está ordenado [9].

Esta lista nunca está completa, es dinámica y cambia constantemente para identificar lo que el producto necesita para ser adecuado, competitivo y útil [9].

Lista de Pendientes del Sprint (Sprint Backlog)

La Lista de Pendientes del Sprint es el conjunto de elementos de la Lista de Producto seleccionados para el Sprint, junto con un plan para entregar el Incremento de producto y conseguir el Objetivo del Sprint. La Lista de Pendientes del Sprint es una predicción hecha por el Equipo de Desarrollo acerca de qué funcionalidad formará parte del próximo Incremento y del trabajo necesario para entregar esa funcionalidad en un Incremento "Terminado" [9].

4.3.3 Extreme Programming (XP)

"La Programación Extrema o Extreme Programming (XP) es un marco de desarrollo de software ágil que tiene como objetivo producir software de mayor calidad y una mejor calidad de vida para el equipo de desarrollo. XP es el más específico de los marcos ágiles con respecto a las prácticas de ingeniería apropiadas para el desarrollo de software" [10].

El éxito de la Programación Extrema reside en su énfasis en la satisfacción del cliente ya que le va ofreciendo el software que necesita a medida que lo necesita. Este marco de desarrollo permite responder a los requisitos cambiantes de los clientes incluso a fines del ciclo de vida [10].

La Programación Extrema además fomenta el trabajo en equipo ya que tanto clientes como desarrolladores y gerentes trabajan juntos en un equipo colaborativo; y gracias al entorno simple que implementa se logran equipos altamente productivos [11].

XP es aplicable en un contexto que cumpla las siguientes características según Don Wells:

- Requisitos del software dinámicamente cambiantes
- Riesgos causados por proyectos con tiempos fijos que usan nueva tecnología
- Equipo de desarrollo extendido, pequeño y en la misma locación

- La tecnología que se está utilizando permite pruebas unitarias y funcionales automatizadas [10].

La Programación Extrema (XP) adopta cinco valores para guiar el desarrollo:

- **Comunicación:** Todos forman parte del equipo y se comunican cara a cara diariamente para trabajar juntos desde los requisitos hasta el código.
- **Simplicidad:** Se hace lo que se necesita y es solicitado, pero no más. Se toman pequeños pasos simples hacia el objetivo y se mitigan las fallas a medida que ocurren.
- **Retroalimentación:** Se entrega lo comprometido en cada iteración entregando software en funcionamiento. Se escucha y hacen los cambios necesarios adaptando el proceso al proyecto y no al revés.
- **Coraje:** Se dice la verdad acerca del progreso y las estimaciones.
- **Respeto:** Todos dan y sienten el respeto que merecen como un valioso miembro del equipo. Los desarrolladores respetan la experiencia de los clientes y viceversa. La gerencia respeta el derecho de los desarrolladores a aceptar responsabilidad y recibir autoridad sobre su propio trabajo [12].

Las prácticas de la Programación Extrema han cambiado un poco desde que se introdujeron inicialmente. A continuación, están las descripciones de las prácticas descritas en la segunda edición de Extreme Programming Explained, Embrace Change[13]. Estas descripciones incluyen mejoras basadas en las experiencias de aquellos que practican la programación extrema:

- **Sentarse juntos:** Hacer que el equipo se siente en el mismo espacio sin barreras de comunicación, como las paredes de un cubículo.
- **Todo el equipo:** Un grupo funcional cruzado con las funciones necesarias para un producto forma un solo equipo.
- **Espacio de trabajo informativo:** Organizar el espacio del equipo para facilitar la comunicación, permitir que las personas tengan algo de privacidad cuando lo necesiten, y hacer que el trabajo del equipo sea transparente comunicando activamente información actualizada.
- **Trabajo Energizado:** Tomar medidas para asegurarse de que pueda física y mentalmente entrar en un estado enfocado. No trabajar demasiado o

dejar que los demás lo hagan, mantenerse saludable y mostrar respeto a los compañeros de equipo para mantenerlos sanos.

- **Programación de pares:** El software es desarrollado por dos personas sentadas en la misma máquina. Se obtiene una revisión continua del código y una respuesta más rápida a los problemas que pueden detener o bloquear a una persona.
- **Historias:** Describa qué debe hacer el producto en términos significativos para clientes y usuarios.
- **Ciclo Semanal:** Sinónimo de iteración. El objetivo para el final de la semana es ejecutar funciones probadas que realicen las historias seleccionadas.
- **Ciclo Trimestral:** El objetivo es mantener el trabajo detallado de cada ciclo semanal en el contexto del proyecto general.
- **Holgura:** La idea detrás de la holgura en términos de XP es agregar algunas tareas o historias de baja prioridad en sus ciclos semanales y trimestrales que pueden descartarse si el equipo se atrasa en tareas o historias más importantes.
- **Creación de build en 10 minutos:** Construir automáticamente todo el sistema y ejecutar todas las pruebas en diez minutos. Un build que lleve más de 10 minutos será usado con menos frecuencia.
- **Integración Continua:** Los cambios de código se prueban inmediatamente cuando se agregan a una base de código más grande.
- **Desarrollo guiado por pruebas:** Reduce el ciclo de retroalimentación para que los desarrolladores identifiquen y resuelvan problemas, disminuyendo así la cantidad de errores que se introducen.
- **Diseño Incremental** [10]**.**

4.3.4 Kanban

Según Agile Alliance [14], el Método Kanban es un medio para diseñar, administrar y mejorar los sistemas de flujo para el trabajo de conocimiento. El método también permite a las organizaciones comenzar con su flujo de trabajo existente y dirigirlo hacia el cambio evolutivo. Pueden hacerlo visualizando su flujo de trabajo, limitando el trabajo en progreso y dejando de comenzar y comenzando a terminar.

El Método Kanban recibe su nombre del uso de kanban (mecanismos de señalización visual para controlar el trabajo en curso para productos de trabajo intangibles)[14].

Kanban se puede usar en cualquier entorno de trabajo de conocimiento, y es particularmente aplicable en situaciones donde el trabajo llega de manera impredecible y / o cuando se desea implementar el trabajo tan pronto como está listo, en lugar de esperar por otros elementos de trabajo [14].

Valores[14]

Los equipos que aplican Kanban para mejorar los servicios que entregan adoptan los siguientes valores:

- **Transparencia:** Compartir información abiertamente usando un lenguaje claro y directo.
- **Equilibrio:** Diferentes aspectos, puntos de vista y capacidades deben equilibrarse para lograr efectividad.
- **Colaboración:** Kanban se creó para mejorar la forma en que las personas trabajan juntas.
- **Enfoque al cliente:** los sistemas Kanban apuntan a optimizar el flujo de valor para los clientes que son externos al sistema pero que pueden ser internos o externos a la organización en la que existe el sistema.
- **Flujo:** el trabajo es un flujo continuo de valor.
- **Liderazgo:** el liderazgo es necesario en todos los niveles para lograr mejoras continuas y ofrecer valor.
- **Comprensión:** El autoconocimiento individual y organizacional desde el comienzo es necesario para avanzar y mejorar.

- **Acuerdo:** todas las personas involucradas en un sistema se comprometen a mejorar y acuerdan moverse juntas hacia objetivos respetando y aceptando las diferencias de opinión y enfoque.
- **Respeto:** valorar, comprender y mostrar consideración por las personas.

Prácticas[14]

Las siguientes prácticas son actividades esenciales para administrar un sistema kanban.

- **Visualizar:** Los sistemas Kanban utilizan mecanismos tales como un tablero Kanban para visualizar el trabajo y el proceso por el que pasa. Para que la visualización sea la más efectiva, debe mostrar:
    - Donde en el proceso, un equipo acuerda hacer un elemento de trabajo específico (punto de compromiso)
    - Donde el equipo entrega la pieza a un cliente (punto de entrega)
    - Políticas que determinan qué trabajo debe existir en una etapa particular
    - Límites de **WIP**
- **Limitar el trabajo en progreso:** Se puede suavizar el flujo de trabajo y reducir los plazos de entrega, mejorar la calidad y entregar con más frecuencia al establecer límites en la cantidad de trabajo que se tiene en progreso.
- **Administrar flujo:** El flujo de trabajo debe maximizar la entrega de valor, minimizar los plazos de entrega y ser lo más predecible posible. Un aspecto clave es identificar y abordar los cuellos de botella y bloqueadores.
- **Hacer políticas explícitas:** Las políticas explícitas ayudan a explicar un proceso más allá del simple listado de diferentes etapas en el flujo de trabajo. Las políticas deben ser escasas, simples, bien definidas, visibles, siempre aplicadas y fácilmente modificables.
- **Implementar circuitos de retroalimentación.**
- **Mejorar colaborativamente, evolucionar experimentalmente:** Kanban comienza con el proceso tal como existe actualmente y aplica mejoras

continuas e incrementales en lugar de tratar de alcanzar un objetivo predefinido.

4.3.5 Crystal

Según Alistair Cockburn[15], Crystal es una familia de metodologías que enfatizan la entrega frecuente, la comunicación cercana y la mejora reflexiva. No hay una sola metodología Crystal. Existen diferentes metodologías Crystal para diferentes tipos de proyectos. Cada proyecto u organización usa las características de Crystal para generar nuevas metodologías.

El nombre "Crystal" viene de la caracterización de Cockburn[15] de proyectos en dos dimensiones, tamaño y criticidad, que coinciden con los de los minerales, el color y la dureza.

Alistair Cockburn[15] caracteriza las metodologías de Crystal por color, según el número de personas coordinadas: Clear (Transparente) es para equipos de 8 personas o menos, Yellow (Amarillo) es para equipos de 10-20 personas, Orange (Naranja) es para 20-50 personas, Red (Rojo) es para 50-100 personas, y así sucesivamente, a través de Maroon (Marrón), Blue (Azul) y Violet (Violeta). Se puede observar que los proyectos más grandes, que requieren más coordinación y comunicación, se asignan a colores más oscuros (claro, amarillo, naranja, rojo, etc.).

Crystal tiene el equipo del proyecto orientado hacia siete propiedades de seguridad, cuyas tres primeras propiedades son fundamentales para Crystal. Los demás se pueden agregar en cualquier orden para aumentar el margen de seguridad. Estas propiedades son las siguientes:

1) Entrega frecuente.
2) Mejora reflexiva (o Mejora Continua).
3) Comunicación cercana.
4) Seguridad personal (el primer paso en la confianza).
5) Atención.
6) Fácil acceso a usuarios expertos.
7) Entorno técnico con pruebas automatizadas, gestión de configuraciones e integración frecuente [15].

Los principios de Crystal se describen en detalle en Agile Software Development [16]. Entre ellos se encuentran algunas ideas centrales:

- La cantidad de detalles necesarios en los requisitos, diseño y documentos de planificación varía según las circunstancias del proyecto, específicamente la magnitud del daño que pueden ser causados por defectos no detectados y que tan frecuentemente colabora el equipo.
- Tal vez no sea posible eliminar todos los productos de trabajo intermedio pero pueden reducirse siempre que el equipo disponga de rutas de comunicación cortas, ricas e informales, y el software probado y en funcionamiento se entregue temprano y con frecuencia.
- El equipo ajusta continuamente sus convenciones de trabajo para adaptarse a las personalidades del equipo, el entorno de trabajo local actual, y las peculiaridades de la asignación específica [15].

Debido a que cada compañía y proyecto es ligeramente diferente, incluso Crystal Clear no está completamente especificado. Los primeros pasos para adoptar Crystal Clear son descubrir los puntos fuertes y las debilidades de su organización y ajustarse a las recomendaciones de Crystal Clear para capitalizar los puntos fuertes y cubrir las debilidades [15].

Para algunas organizaciones, esto es demasiado trabajo. Crystal Clear no es para esas organizaciones, es para grupos que desean construir su propia manera personal, fuerte y efectiva de entregar software repetidamente [15].

Crystal Clear comparte algunas características con XP, pero generalmente es menos exigente. Puede pensar que es una alternativa más relajada, una alternativa si XP no funciona para el grupo, o un punto de partida para implementar algunas prácticas ágiles antes de saltar a XP [15].

4.3.6 Lean

La idea central de Lean es maximizar el valor para el cliente y minimizar el desperdicio. Simplemente, crear más valor para los clientes con menos recursos [17].

Una organización Lean comprende el valor para el cliente y enfoca sus procesos para aumentarlo continuamente. El objetivo final es proporcionar un valor perfecto

para el cliente a través de un proceso de creación de valor perfecto que no tiene desperdicio [17].

Principios Lean [18]

Lean se ha basado en cinco principios y una amplia gama de prácticas que se han extraído del sistema de producción de Toyota y las experiencias de otras compañías que han seguido el ejemplo de Toyota. Estos cinco principios se identifican como:

- **Valor:** El valor lo define el cliente. ¿Qué valora el cliente en el producto? Se debe comprender qué es y qué no es valioso en el ojo del cliente.
- **Flujo de valor:** Una vez que se sabe lo que el cliente valora en su producto, se puede crear un flujo de valor que identifique la serie de pasos necesarios para producir el producto. Cada paso se puede clasificar como: de valor agregado; sin valor agregado pero necesario o sin valor agregado.
- **Fluir:** El proceso de producción debe estar diseñado para fluir continuamente. Si la cadena de valor deja de avanzar (por cualquier motivo), se producen desperdicios.
- **Halar:** Dejar que los pedidos de los clientes tiren del producto (valor). Este pull regresa en cascada a través del flujo de valores y garantiza que no se haga nada antes de que sea necesario, eliminando así la mayoría del inventario en proceso.
- **Perfección:** Es necesario esforzarse por la perfección identificando y eliminando continuamente los desechos.

En 2003, Mary y Tom Poppendieck publicaron un primer listado de principios Lean para el desarrollo de software en su libro *Lean Software Development: An Agile Toolkit for Software Development Managers* (Addison-Wesley Professional), y refinaron este mapeo en su segundo libro, *Implementación del desarrollo de software Lean: del concepto al efectivo* (Addison-Wesley Professional, 2006)[18].

La mayoría de las publicaciones posteriores sobre Lean Software Development han seguido el ejemplo de Poppendieck y han utilizado los siete principios que ellos identificaron [18]:

1) **Eliminar residuos:**

   a) Los defectos provocan un costoso retrabajo, que siempre es un desperdicio sin valor agregado. Lean busca prevenir defectos, mientras que el desarrollo tradicional se centra en encontrar defectos una vez que ya se han producido. Los defectos son especialmente costosos cuando se detectan tarde.
   b) Cada línea de código cuesta dinero. Por esto la presencia del código debe tenerse en cuenta en cada cambio o mejora futura del producto. La regla 80/20 se aplica en la mayoría de los productos de software: el 20% de las características del producto proporciona el 80% de las necesidades reales del usuario. El otro 80% de las características de un producto rara vez (o nunca) se utilizan.
   c) Se pierde una gran cantidad de conocimiento en cada entrega entre subequipos simplemente porque no es posible registrar todo lo que se aprendió, descubrió, creó y conoció en forma escrita. Es importante evitar transferencias siempre que sea posible.
   d) Las decisiones se toman casi constantemente, el desarrollador no puede saber todo y necesitará hacer preguntas a otras personas. Si estas personas están disponibles de inmediato, no hay demora y el desarrollo continúa a toda velocidad.
   e) En lugar de dejar que el trabajo parcialmente hecho se acumule en las colas, el enfoque Lean usa flujo de una sola pieza para llevar una característica a la implementación lo más rápido posible.
   f) El cambiar de tareas y las interrupciones matan la productividad.
   g) Los procesos innecesarios son desperdicio. Esto incluye tareas manuales que podrían automatizarse y procedimientos que dificultan tareas simples.

2) **Embeber la calidad:** Hacer un código a prueba de errores escribiendo pruebas mientras codifica las características. Estas pruebas evitan que los cambios posteriores al código introduzcan defectos no detectados.

3) **Crear conocimiento:** Encontrar formas de registrar el conocimiento del equipo para poder ubicarlo fácilmente cuando se lo necesite aprendiendo así de los errores.

4) **Aplazar el compromiso:** Las mejores decisiones se toman cuando se tiene la mayor cantidad de información disponible. Si no urge tomar la decisión lo mejor es postergarla pero sin esperar demasiado, la falta de una decisión no debería retrasar otros aspectos del proyecto.

5) **Entregar rápido:** Desarrollar funciones en pequeños lotes que se entregan al cliente rápidamente, en iteraciones cortas. Esto le permite al cliente probar ciertas características que pueden cambiar otros requisitos antes de ser implementados.

6) **Respeto a las personas:** Hace referencia a confiar en las personas para saber cuál es la mejor manera de hacer su trabajo, involucrarlas para exponer fallas en el proceso actual y alentarlas a encontrar formas de mejorar sus trabajos y los procesos que los rodean. Respetar a las personas significa reconocerlas por sus logros y solicitar activamente su consejo.

7) **Optimizar el todo:** Estos es muy importante en Lean, cada vez que optimiza un proceso local, casi siempre se hace a expensas de todo el flujo de valores (y esto no es óptimo).

4.4 Tailoring

Tailoring: *"El acto de ajustar las definiciones y / o particularizar los términos de una descripción general para derivar una descripción aplicable a un entorno alternativo (menos general)"* [19].

Los criterios y pautas de Tailoring [3] describen lo siguiente:

- Cómo se usan los procesos estándar y los activos de los procesos de la organización para crear procesos definidos.

- Requisitos que deben cumplir los procesos definidos.
- Opciones que se pueden ejercerse y criterios para seleccionar entre opciones.
- Procedimientos que se deben seguir para realizar y documentar la adaptación de procesos.

La flexibilidad en el tailoring y en la definición de procesos se equilibra con la garantía de la consistencia apropiada de los procesos a lo largo de toda la organización. La flexibilidad es necesaria para abordar variables contextuales como el dominio; la naturaleza del cliente; costo, cronograma y compensaciones de calidad; la dificultad técnica del trabajo; y la experiencia de las personas que implementan el proceso. Se necesita consistencia en toda la organización para que los estándares, objetivos y estrategias de la organización se aborden de manera adecuada, y se puedan compartir los datos de proceso y las lecciones aprendidas [3].

El Tailoring es una actividad crítica que permite cambios controlados en los procesos debido a las necesidades específicas de un grupo de trabajo o una parte de la organización. Los procesos y los elementos del proceso que están directamente relacionados con los objetivos comerciales críticos generalmente deben definirse como obligatorios, pero los procesos y los elementos del proceso que son menos críticos o que solo afectan indirectamente los objetivos del negocio pueden permitir una mayor adaptación [3].

La cantidad de personalización también podría depender del modelo de ciclo de vida del grupo de trabajo, el uso de proveedores y otros factores [3].

CAPÍTULO 5. SITUACIÓN ACTUAL

El caso de estudio se origina en una software factory que presta servicios a una de las principales aerolíneas de los Estados Unidos. Dicha empresa utiliza una metodología ágil como guía para darle marco a su propio proceso de desarrollo de los proyectos de software que administran.

5.1 Metodología Utilizada

Se utiliza como metodología de base **Scrum** pero la misma se encuentra adaptada a las necesidades internas del proyecto.

Al igual que en Scrum, el equipo consiste en:

- **Un dueño de producto** (Product Owner): el mismo se encuentra del lado del cliente y es quien posee el mayor conocimiento del producto y quien define las prioridades de la lista de producto (Product Backlog) así como también provee la información de negocio necesaria para que el equipo de desarrollo realice su trabajo.

- **Un Scrum Master**: que en este caso es llamado Project Manager, el mismo se encarga de asegurarse que se realicen todos las eventos de Scrum (Daily Scrum, Sprint Planning, Sprint Review, Sprint Retrospective) necesarios así como también intervenir y ayudar a destrabar bloqueos en el desarrollo. El Project Manager divide su tiempo entre varios equipos de desarrollo los cuales están afectados a distintos productos (sub-proyectos).

- **El Equipo de Desarrollo**: el equipo de desarrollo normalmente está compuesto de 3 desarrolladores, pero a diferencia de Scrum, hay diferencia en las tareas que realizan. Hay 2 desarrolladores cuya tarea principal es desarrollar el código y validarlo y un desarrollador Senior o Líder Técnico (con más experiencia en la aplicación y en proyectos de desarrollo) que brinda soporte a los desarrolladores y realiza las revisiones de código así como también despliega los builds para que sean testeados por el cliente. Este desarrollador Senior puede estar 100% dedicado a un sub-proyecto o dividir su tiempo entre varios. El equipo de desarrollo no realiza tareas de Testing, solo hace un test unitario manual de la funcionalidad indicada en

la User Story, si el sistema tiene el comportamiento requerido en la misma, se asume que el código es correcto y se manda a testear por el equipo de Testing que pertenece al cliente y es externo al equipo de desarrollo.

Además, basándose en Scrum sus eventos consisten en:

- **Daily Scrum interna:** simplemente llamada "Daily", el PM se reúne con todos sus equipos de desarrollo y pregunta a cada uno de los integrantes que actividades realizaron el día anterior, que actividades van a realizar en el día y si tienen algún bloqueo o impedimento para continuar con su trabajo. En esta reunión no participa el dueño del producto ni tampoco ningún interesado del lado del cliente. Los participantes permanecen en la reunión hasta que todos los integrantes de todos los equipos de desarrollo hayan completado su turno para hablar.

- **Daily Scrum con el cliente:** en esta reunión participan el equipo de desarrollo, el Dueño de Producto, el equipo de Testing y otros interesados del lado del cliente. Solo participan los equipos afectados al producto en cuestión.

- **Sprint Planning:** se seleccionan las User Stories de la Lista de Producto (en adelante Backlog) que serán trabajadas en el sprint. Las mismas deberán tener toda la información necesaria para ser completadas por el equipo de desarrollo, serán puntuadas y estimadas de acuerdo a su esfuerzo. La cantidad de User Stories que entraran en el Sprint será de acuerdo al puntaje de las mismas y a la velocidad del equipo de desarrollo.

- **Sprint Review:** No se lleva a cabo una revisión del Sprint como lo indica Scrum. Se entiende que este evento tiene lugar en las oficinas del cliente entre los Testers, el Dueño de Producto y otros interesados; pero ni el Project Manager ni el equipo de desarrollo participan en el mismo

- **Backlog grooming:** este no es un evento de Scrum sin embargo es un evento que se realiza en este equipo de desarrollo para revisar las User Stories que se encuentran en el Backlog e intentar aclarar dudas y faltantes

de información. Participan en este evento el Dueño del Producto, el Project Manager y el equipo de desarrollo.

- **Sprint Retrospective:** se realiza una revisión de que salió bien y que salió mal durante el sprint así como también que puede mejorarse para el próximo sprint. Participan en este evento el Project Manager, el Dueño del Producto, el equipo de desarrollo, el equipo de Testing y otros interesados de parte del cliente.

- **Sprint:** la duración del sprint es de 2 a 3 semanas dependiendo del producto desarrollado.

5.2 Proceso de desarrollo

El desarrollo de las User Stories se realiza de la siguiente manera, una vez que la User Story es aceptada para entrar en el Sprint, uno de los desarrolladores del equipo de desarrollo se la asigna y carga las tareas que debe realizar para completar la User Story. En cada tarea se carga una pequeña descripción y una estimación en horas de lo que le llevará completar la misma.

El desarrollador realiza su código basándose en la información que se encuentra detallada en la User Story y prueba la misma en función de los puntos de verificación que se encuentran en esta.

Si el desarrollador encuentra algún bloqueo por falta de información o necesidad de clarificación de parte del cliente esto es informado durante la daily y el dueño del producto se encarga de conseguir la información necesaria. Si por otro lado el desarrollador se encuentra bloqueado por cuestiones de conocimiento técnico, el mismo puede pedir soporte al desarrollador Senior para seguir avanzando con la User Story sin necesidad de involucrar al Dueño del Producto.

Una vez que el desarrollador termina de programar el código, y realiza los testeos manuales, este lo sube al ambiente de test utilizando un versionador de código e informa al desarrollador Senior que su código está listo para la revisión.

El desarrollador Senior realiza una revisión técnica del código en busca de errores de programación y/o código ineficiente, si encuentra alguna mejora para realizarle al

código se lo informa al desarrollador para que este haga la corrección necesaria y vuelva a subir el código, si todo está bien, el desarrollador Senior despliega un nuevo build e informa al equipo de Testing que puede realizar los tests necesarios.

El equipo de Testing realiza sus pruebas y si todo esta correcto informa que está listo para recibir un build en el ambiente de Test, entonces el desarrollador Senior despliega ahora el build en el ambiente de Test e informa a todos los interesados. Luego el equipo de Testing realiza tests de integración sobre el último build entregado y si todo funciona como fue solicitado la User Story es aceptada.

Si hay una User Story lista para ser trabajada en el Backlog y hay suficiente tiempo en el Sprint se agrega la misma al Sprint Backlog y el desarrollador que terminó su User Story previa se la asigna y este proceso se repite hasta finalizar el Sprint.

5.3 Ambientes de Desarrollo

El cliente cuenta con 2 ambientes de desarrollo y 1 ambiente de producción, asimismo los desarrolladores realizan su trabajo en sus propias máquinas donde emulan el ambiente de desarrollo. Los ambientes con los que cuenta el cliente son:

- **Desarrollo:** este ambiente es donde se sube en primera instancia el código que de los productos que están siendo desarrollados. Es un ambiente que intenta emular el ambiente de producción pero que puede tener ciertas diferencias con el mismo por diversos motivos.

- **Testing:** este ambiente es donde se probará el código aceptado del ambiente de Desarrollo. Este ambiente emula exactamente el ambiente de producción, lo cual asegura que lo que funcione bien aquí también funcionará en producción.

- **Producción:** es el ambiente donde está desplegado el sistema que utilizan los agentes para realizar las actividades comerciales de la empresa. Aquí es donde termina todo el código de los productos desarrollados que fueron probados y aceptados. Cualquier error o problema en el código que llegue a este ambiente significa pérdidas económicas para el cliente y asimismo multas para la compañía que desarrolla el sistema para el cliente asi como tambien pérdida de confianza del cliente en el equipo de desarrollo.

CAPÍTULO 6. DIAGNÓSTICO DE LA SITUACIÓN ACTUAL

6.1 Introducción

Este proyecto de investigación tiene su origen en reiterados problemas de comunicación y defectos inyectados por el equipo de desarrollo que llegaron al ambiente previo a producción. Se identificó que al no haber sido detectados por el equipo de Testing ni de desarrollo, existe una necesidad de revisar en detalle el proceso de desarrollo ya que estos defectos podrían haber impactado directamente en la facturación de la compañía.

Se realizó un estudio de la causa raíz de los defectos encontrados y de porque pasaron las distintas instancias al punto de casi llegar al ambiente productivo. El estudio se basó en una investigación detallada de cada uno de los componentes del proyecto, por componentes la investigación se refirió a:

- Procesos
- Equipos técnicos
- Líderes
- Documentación

En un principio se atribuyeron los problemas a errores humanos, pero luego de una serie de reuniones con desarrolladores y Analistas de Calidad así como también con los líderes de los distintos equipos se determinó que había una falla en la comunicación entre el equipo de negocio y los equipos técnicos que tienen que desarrollar los productos de software. El equipo de negocio no era claro en los requerimientos y los cambiaba constantemente aun habiendo comenzado el desarrollo. Por otro lado, cuando los requerimientos no estaban claros, los equipos técnicos se basaban en suposiciones sobre cómo debía comportarse el sistema por su propia experiencia en el mismo y a su vez los equipos de Testing al no tener una definición clara de lo que debían probar realizaban pruebas generales y muchos errores pasaban sin ser detectados.

En función de esta investigación y partiendo de la base de que: "al mejorar el proceso se mejorará el producto"[20], se determinó que era necesario modificar el proceso de desarrollo que se estaba implementando con el fin de obtener una

documentación más detallada que brindara un mayor control sobre el código y los requerimientos del cliente, así como también un mayor respaldo y sobre todo una aceptación de dichos requerimientos por parte de cliente.

6.2 Análisis de defectos

Se realizó un análisis de la cantidad total de defectos encontrados en un producto representativo, cuyo proyecto constó de 70 Historias. En este proyecto fue en donde apareció el defecto que casi llega a producción, y se seleccionaron 4 patrones de búsqueda para el análisis. Cada patrón será analizado a continuación con sus valores de referencia.

6.2.1 Tipo de Problema:

Tipo de Issue	Cantidad	Porcentaje
New Development	86	81,13%
Defect	11	10,38%
Merge	9	8,49%
Total	106	100%

Figura 17. Tabla de tipos de problemas, elaboración propia basada en la información extraída de Jira en el proyecto utilizado para el análisis

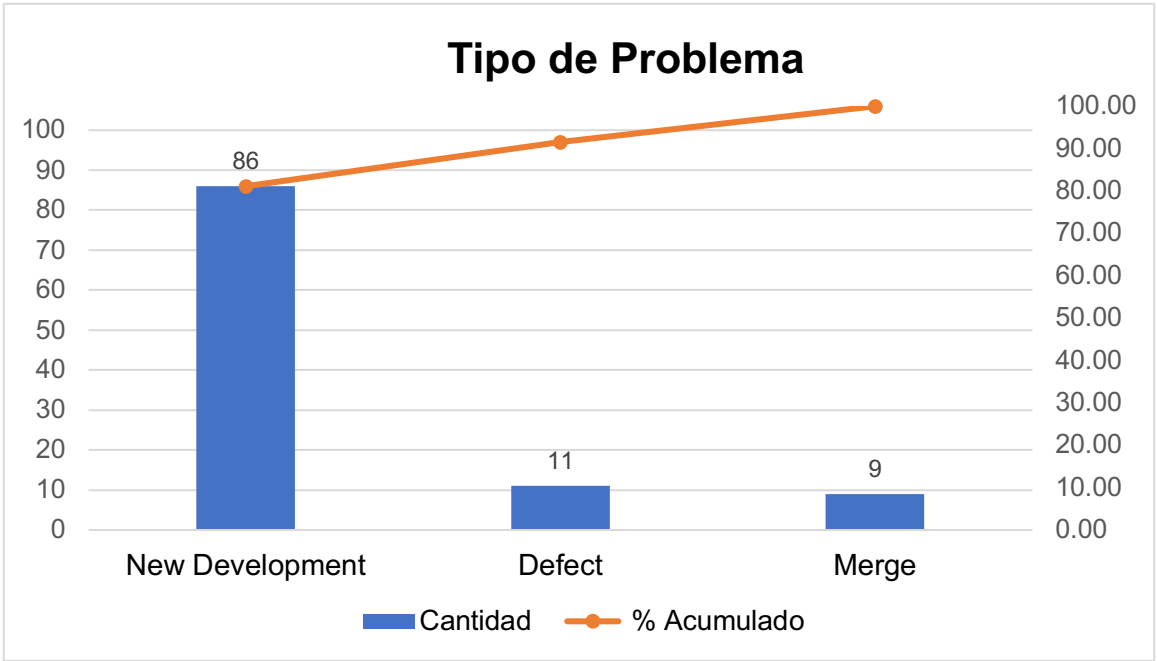

Figura 18. Histograma de tipos de problemas, elaboración propia basada en la información extraída de Jira en el proyecto utilizado para el análisis

En base a los valores de referencia se puede ver que hay un gran porcentaje de defectos en los cuales el tipo de problema fue debido al nuevo desarrollo seguido por defectos preexistentes en la aplicación y luego problemas de fusión (***merge***) de código nuevo con el código heredado y compartido con otros productos. Se puede analizar entonces que la gran mayoría de los defectos (81,13%) podrían haberse evitado en la etapa del desarrollo, y otro 8,49% al momento de mergear con otros proyectos. Teniendo en cuenta esto, en un análisis inicial se podría decir que el 95% de los defectos podría haberse evitado antes de llegar siquiera a una etapa de Testing. Esto aún no brinda un entendimiento de cómo disminuir la cantidad de defectos pero es un indicador de que hay que hacer foco en el área de desarrollo. No interesan los defectos heredados, que no tienen que ver con este producto, en cambio interesa analizar los defectos inyectados con este producto y como fueron solucionados. Estos defectos pueden haber sido introducidos por **merge** (fusión de código) o por **nuevos desarrollos**.

6.2.2 Tipo de resolución de los defectos:

Primeramente fue necesario eliminar los datos relacionados a los 11 defectos heredados de los 106 problemas que se están analizando ya que como se explicó en el párrafo anterior, estos no son relevantes para el estudio que se está llevando a cabo. Considerando solo los defectos introducidos por **merge** (fusión) o **nuevos desarrollos** se elaboró el gráfico a continuación en función de los tipos de resolución.

Tipo de Resolución	Cantidad	Porcentaje
Funcionando como fue diseñado	24	25,26%
Reparado (Código Actualizado)	63	66,32%
No es posible reproducir el error	8	8,42%
Total	95	100%

Figura 19. Tabla de tipo de resolución, elaboración propia basada en la información extraída de Jira en el proyecto utilizado para el análisis

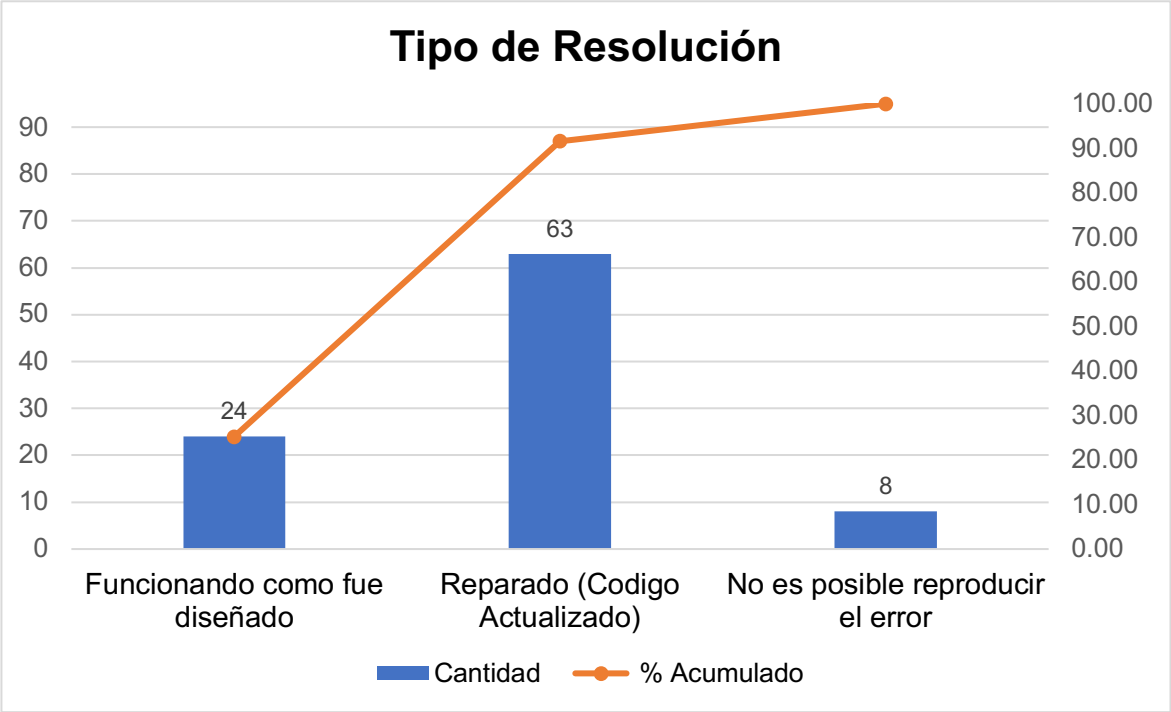

Figura 20. Histograma de tipo de resolución, elaboración propia basada en la información extraída de Jira en el proyecto utilizado para el análisis

Analizando estos datos podemos ver que un mínimo número de defectos entra en la categoría de "**No es posible reproducir el error**", no interesa analizar estos defectos ya que pueden darse por errores temporales, los cuales están fuera del alcance de este trabajo de investigación. Estos errores temporales pueden abarcar: problemas en la disponibilidad de servicios, datos de pruebas expirados y la imposibilidad de volver a replicarlos, entre otros.

En cambio este trabajo se centrara en los otros 2 tipos de resolución. Se puede ver que hay un porcentaje importante de defectos que luego de ser analizados por el área de desarrollo se determinó que estaban funcionando como fueron pedidos, de aquí se puede deducir que los requerimientos fueron mal elaborados o hubo un desconocimiento del área de Testing al crear el defecto. Por otro lado se puede ver que el mayor número de defectos entra en la categoría de "**Reparado (Código Actualizado)**", lo que está indicando que hubo un retrabajo ya sea porque no cumplía

con el requerimiento en algún escenario específico o por otras razones a determinar, para esto es importante analizar la causa raíz del defecto.

6.2.3 Causa raíz de los defectos:

Causa del defecto	Cantidad	Porcentaje
Desarrollo	52	59,77%
Requerimientos	17	19,54%
Testing	8	9,20%
Dependencias Externas	10	11,49%
Total	87	100%

Figura 21. Tabla de causa raíz de los defectos, elaboración propia basada en la información extraída de Jira en el proyecto utilizado para el análisis

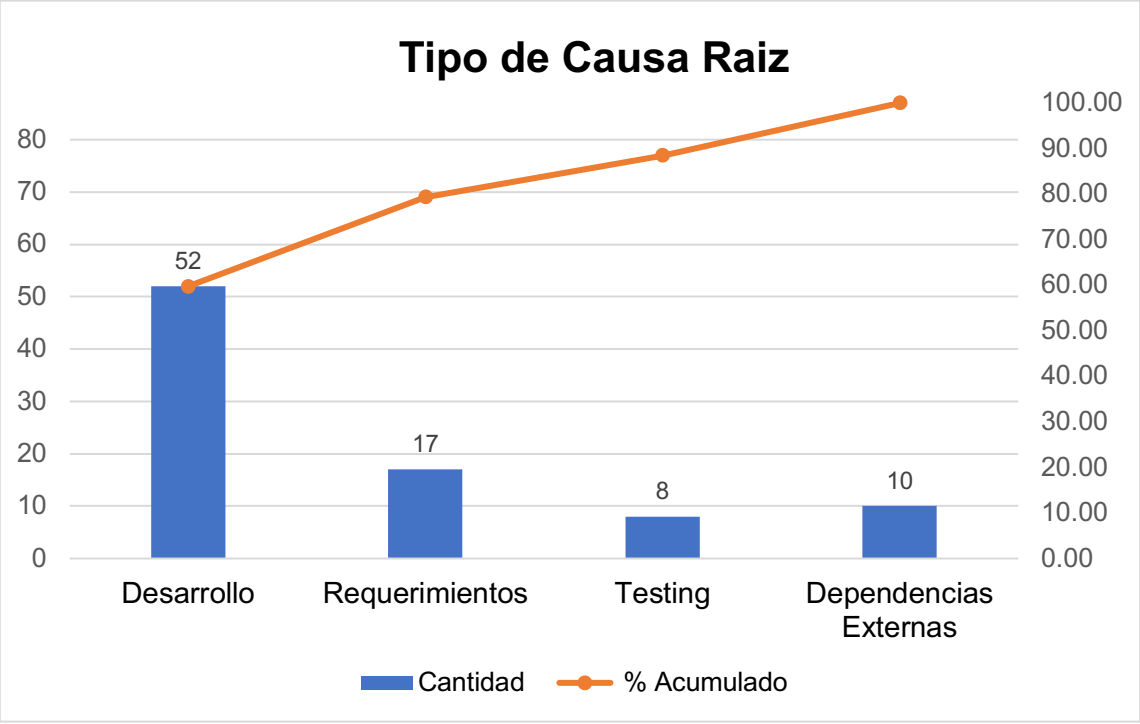

Figura 22. Histograma de causa raíz de los defectos, elaboración propia basada en la información extraída de Jira en el proyecto utilizado para el análisis

Analizando las distintas causas de los defectos se puede determinar que más de la mitad de ellos se dio por errores en el "**Desarrollo**", ya sea porque no se tuvieron en cuenta todos los escenarios posibles, errores en la programación, mala interpretación de los requerimientos, etc. El grupo de causas que sigue al desarrollo es "**Requerimientos**", estos tipos de defectos se dan por requerimientos ambiguos, poco claros, información faltante, etc. Luego siguen en menor medida las "**Dependencias Externas**", estas vienen de la mano de cambios de otras áreas que impactan directamente en el producto pero de las cuales no tienen control los equipos internos por ejemplo, servicios web de otros proveedores, aplicaciones externas, etc. Por último y en menor medida vemos que hay defectos ligados al "**Testing**" que son, por mala interpretación de los requerimientos de Testing, poca claridad en lo que se debe testear y cómo hacerlo, falta de escenarios de testeo, etc.

6.2.4 Impacto de los defectos:

Antes de realizar un análisis final de estos datos también se considerara importante determinar cuál fue el impacto real de los defectos y en qué medida podían haber afectado el desempeño de la aplicación. Se considerara para este análisis lo que implica cada impacto:

- **Crítico**: impacto en una funcionalidad importante sin una solución alternativa posible. Por ejemplo fallas del sistema con pérdidas de datos irrecuperables.
- **Alto:** impacta en una funcionalidad importante y desviación de los requerimientos. En este caso existe una solución alternativa disponible.
- **Medio:** impacta funcionalidades no críticas, existe una solución alternativa simple que permite al sistema funcionar correctamente. Implicaría molestias e inconvenientes.
- **Bajo:** sería simplemente un problema cosmético. Una diferencia en lo que se esperaba en lo que respecta a la experiencia del usuario.

En base a esto se tendrá para analizar los siguientes valores:

Impacto de los defectos	Cantidad	Porcentaje
Crítico	0	0%
Alto	14	16,09%
Medio	41	47,13%
Bajo	32	36,78%
Total	87	100%

Figura 23. Tabla de impacto de los defectos, elaboración propia basada en la información extraída de Jira en el proyecto utilizado para el análisis

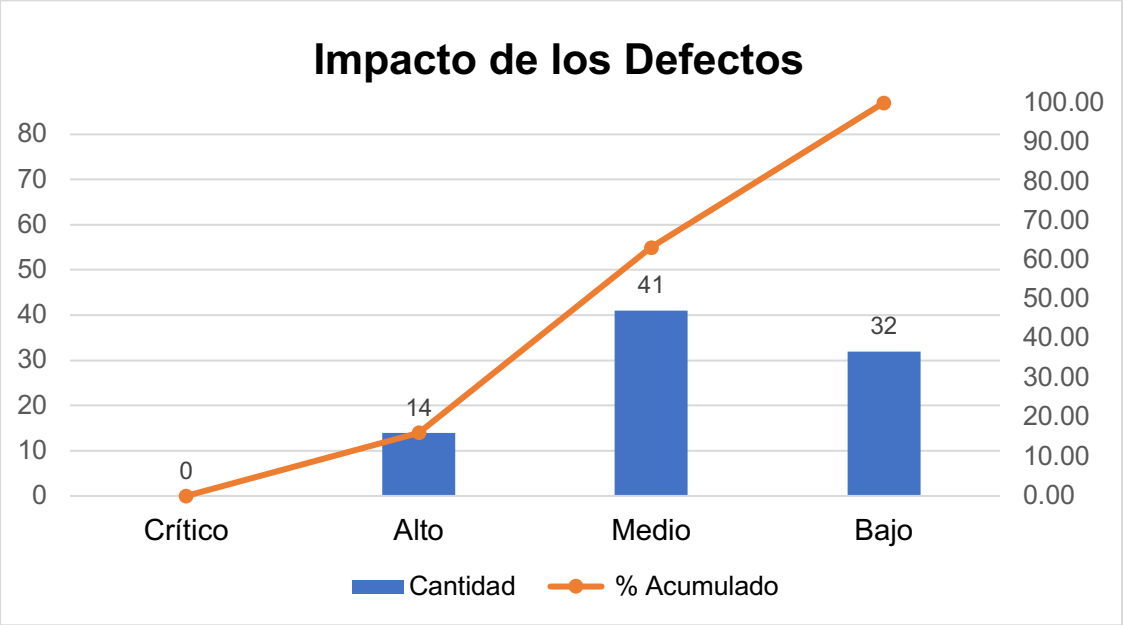

Figura 24. Histograma de impacto de los defectos, elaboración propia basada en la información extraída de Jira en el proyecto utilizado para el análisis

Analizando los datos provistos se puede concluir que solo existe un pequeño número de defectos de impacto "**Alto**" sin embargo los mismos podrían significar graves pérdidas económicas para el cliente y lo esperado es que no haya defectos de este tipo al final del desarrollo del producto. La mayor cantidad de defectos tiene un impacto **"Medio"**, esto puede indicar que al momento de desarrollo no se tuvieron en

cuenta todas las variantes o faltaron requerimientos para determinados escenarios que luego se hicieron evidentes al momento de hacer la demostración del producto. Por último se evalúan los defectos con impacto "**Bajo**" los cuales se ve que hay una gran cantidad y puede estar relacionado al hecho de falta de detalle en el requerimiento, por ejemplo imágenes con Interfaces de Usuario, o interfaces de usuario desactualizadas o bien por falla de desarrollo. No se contabilizaron defectos con impacto "**Crítico**".

6.3 Hipótesis

En base al análisis realizado se puede determinar que una gran cantidad de los defectos que se generan en la situación actual posiblemente tenga su origen en un conjunto de situaciones a mejorar que se detallaran a continuación:

- Información incompleta en la documentación de las User Stories.
- Fallas en la comunicación entre equipos internos y proveedores externos.
- Errores, ambigüedad y/o información faltante en la redacción de los requerimientos por parte del dueño del producto.
- Errores en la interpretación de los requerimientos por parte de los equipos de desarrollo.
- Errores en la interpretación de los requerimientos por parte de los equipos de Testing.
- Errores en que y como testear por parte de los equipos de Testing.

Partiendo de la base de que toda mejora en el proceso de desarrollo de un producto proporcionará una mejora en el producto final, la hipótesis es que al mejorar el proceso de desarrollo centrándose en los puntos anteriores y aplicando las mejores prácticas de la industria se mejorara el producto final.

El objetivo es que al aplicar el nuevo proceso de desarrollo definido en este proyecto se observará una disminución de alrededor de un 40% de defectos al momento de entregar el producto.

Se espera que al detallar los requerimientos sin errores, faltantes ni ambigüedad, disminuyan alrededor de un 50% la cantidad de los defectos de los nuevos desarrollos. También se estima que teniendo la información detallada de qué y cómo

se debe testear así como también cuales son todos los escenarios donde testearlo los errores de los Testers deberían disminuir, se considera que si llegan a disminuir los del tipo “funcionando como fue diseñado” alrededor de un 60% de la cantidad actual sería un éxito.

Por otro lado, mejorando la comunicación entre equipos internos y externos y manteniendo a todos los equipos informados se espera que disminuyan alrededor de un 40% la cantidad de los defectos del tipo de dependencias externas.

Por último, si se mejora la comunicación y el proceso de identificación y documentación de requerimientos, y a su vez, se asegura que todos los requerimientos fueron correctamente entendidos por todos los interesados y no hay ninguna duda de que es lo que se debe realizar, entonces los defectos relacionados a los requerimientos es esperable que al menos disminuyan alrededor de un 80% de la cantidad total actual.

CAPÍTULO 7. PROPUESTA DE SOLUCIÓN

Para brindar una solución acorde al complejo escenario que se enfrenta se cree que no alcanzaría simplemente con aplicar uno de los modelos que se vieron en el marco conceptual sino que se decidió tomar las mejores prácticas de cada uno de los mismos y adaptarlas para llegar a un proceso de desarrollo que pueda solventar los complejos problemas que se vieron en el punto anterior.

7.1 CMMI: mejores prácticas que se tomaran para la solución

En el caso de CMMI se realizara un análisis de la solución desde el modelo que propone, tal como se vio en el marco teórico. El modelo de CMMI para Desarrollo contiene 22 áreas de proceso de las cuales 5 corresponden a ingeniería:

- **Integración de Producto (PI):** la integración del producto brindara procedimientos y estrategias para ensamblar el producto en etapas incrementales, las cuales aseguran que el mismo cumpla con el comportamiento deseado.

- **Desarrollo de requisitos (RD):** como se vio en el diagnóstico, una de las áreas más importantes a mejorar es la de desarrollo de requisitos, esta área de proceso dará las bases para identificar, desarrollar, depurar, analizar y validar los requisitos del cliente y del producto así como el impacto de los requisitos derivados de los mismos.

- **Solución técnica (TS):** otra de las grandes falencias que se detectaron fue en el área de desarrollo (que es la que brinda la solución técnica del proyecto), se cree que las prácticas pertenecientes al área de solución técnica (que selecciona, diseña e implementa la solución en función de los requisitos) será fundamental para solventar los problemas.

- **Validación (VAL):** será muy importante también para evitar la creación de defectos innecesarios mejorar la validación que se hace a los entregables, en este punto juega un papel fundamental la validación que se asegura que el producto sea lo que se solicitó, lo que se esperaba más allá de la calidad que pueda tener.

- **Verificación (VER):** la verificación asegurará que el producto final cuenta con la calidad requerida, es un punto importante que evitará defectos futuros ya que esta verificación se da durante todo el desarrollo del producto desde la identificación de requisitos hasta el producto final terminado.

A partir de cada una de estas áreas de proceso se puede ver que son un buen punto de partida para sentar las bases de una solución ya que estas 5 áreas de proceso apuntan a solucionar puntos en los que hoy en día se encuentran serias falencias en el proceso de desarrollo actual.

7.2 PUDS: prácticas que se aplicaran a la solución

Como se detalló en el punto 4.2, el proceso unificado de desarrollo de software es un proceso más tradicional para crear productos de software que tiene componentes más rígidos que las metodologías ágiles actuales, pero se cree que justamente es esta característica la que puede servir en el escenario actual al que se está enfrentando. Como bien se dijo es un proceso que pone su foco en:

- **Personas:** es muy importante tener en cuenta a todas las personas que participan en el producto ya que son las que lo llevan adelante y serán las personas las que, además de construir el producto, testearlo, utilizarlo, etc., apliquen las mejores prácticas del proceso.
- **Proyecto:** este punto también es fundamental ya que se deberá hacer foco en el proyecto y todas sus partes e integrantes para poder eliminar los puntos de falla del mismo así como disminuir el número de defectos que se generan a lo largo del desarrollo del proyecto.
- **Producto:** centrarse en el producto es otro de los elementos fundamentales del PUDS que será de vital importancia, como se vio una de las causas de los defectos era el hecho de que el producto no cumplía con los requerimientos necesarios. Con este punto del PUDS se cree que se mejorara sustancialmente el producto final que será entregado.
- **Proceso:** el proceso es uno de los puntos más importantes a mejorar y como se dijo con anterioridad mejorando el proceso se mejorará el producto. Este es uno de los puntos en los que más se debe trabajar ya que como se vio en la sección "Diagnóstico de la solución", tiene muchas

falencias y es una de las grandes causas de los problemas que surgen en todo el proyecto.

Para poder aprovechar al máximo las mejores prácticas del PUDS será de vital importancia centrarse en crear un proceso dirigido por los casos de uso, estos se obtendrán a partir de los requerimientos que se capturaran del cliente en las reuniones con ellos, como así también en las Historias.

Se contará con un tiempo prudencial para realizar el análisis de los requerimientos capturados así como también transformar los mismos en casos de uso. Una vez obtenidos los casos de uso se procederá a realizar el diseño de la solución utilizando los mismos.

Se revisará el diseño y se procederá a la implementación, la cual usará el diseño como guía y deberá atenerse a este respetando los detalles del mismo.

7.3 Metodologías ágiles aplicadas a la solución

En el punto anterior se dejó en claro que eran necesarias algunas prácticas más tradicionales y rígidas con respecto a la documentación de los requerimientos debido a los problemas específicos que se enfrentan en este trabajo de investigación. No obstante, no se pueden ignorar las ventajas y beneficios de las metodologías ágiles y sería un gran error dejarlas afuera de este proceso que se está desarrollando. Se tomaran algunas de las mejores prácticas de las metodologías que vimos en el punto 4.3 las cuales serán detalladas a continuación.

7.3.1 Scrum

Se tomaran conceptos del equipo de Scrum como:

- **Dueño del producto:** este rol será ejercido por una persona del cliente.
- **Equipo de desarrollo:** el Equipo de Desarrollo estará compuesto por los desarrolladores asignados a un producto determinado.
- **Scrum Master:** este rol será ejercido por uno de los project managers internos de compañía que desarrolla el sistema, este project manager podrá estar asignado como Scrum Master de varios productos a la vez.

Se tomaran también los siguientes eventos de Scrum:

- **Sprint:** serán bloques de tiempo de entre 2 y 3 semanas dependiendo el producto en el cual se creará un incremento de funcionalidad potencialmente desplegable, el proyecto constará de una cantidad limitada de sprints definida antes de comenzar a trabajar en el proyecto.
- **Sprint planning:** aquí se determinará que Historias (User Stories) serán trabajadas en el Sprint en base a la capacidad del equipo de desarrollo. También se evaluará si las Historias contienen toda la información para empezar a trabajarlas o si es necesario obtener más información para su desarrollo.
- **Daily Scrum:** se revisará que tareas realizaron el día anterior los integrantes del equipo de desarrollo, que tareas realizarán durante el día y si hay algún bloqueo que les impida avanzar con sus tareas
- **Sprint Review:** al finalizar el Sprint se realizará una revisión de los incrementos de funcionalidad que se entregaron durante el Sprint, aquí participaran el equipo de Scrum y otros interesados.
- **Sprint Retrospective:** luego de la Sprint Review cada miembro del equipo de Scrum dará su opinión de que salió bien y que salió mal durante el Sprint y que cosas se podrán mejorar para el siguiente Sprint

Se tomaran los siguientes artefactos de Scrum:

- **Product Backlog:** este artefacto es visualizado como una lista de las Historias ordenada por una prioridad de importancia de la funcionalidad del producto para el negocio
- **Sprint Backlog:** este artefacto es visualizado como lista de Historias y sus tareas que funcionan a modo de plan de trabajo. Las mismas deben completarse durante el Sprint para obtener un incremento de funcionalidad entregable.

7.3.2 Extreme Programming

Se tomara de Extreme Programming el concepto de incrementar la satisfacción del cliente mediante la entrega del producto a medida que lo necesita y fomentar el trabajo de equipo. Se intentará incorporar los siguientes valores:

- **Simplicidad:** se realizará lo que se pidió y nada más, evitando así asunciones incorrectas por parte del desarrollador.
- **Retroalimentación:** se entregará lo comprometido en cada iteración y se escucharán los comentarios y sugerencias con las cuales se harán las modificaciones necesarias para adaptarse a las necesidades del producto.
- **Coraje:** los desarrolladores tendrán la potestad sobre las estimaciones de las Historias. Se comunicará el progreso real de las tareas sin temores.
- **Respeto:** habrá un respeto mutuo de las opiniones entre los desarrolladores y el cliente y líderes.

También se aplicaran las siguientes prácticas de XP para mejorar el proceso de desarrollo:

- **Diseño incremental:** se agregan funcionalidades al producto a medida que avance el proyecto y estas funcionalidades se irán entregando incrementalmente y no todas juntas.
- **Integración continua:** el código nuevo se irá integrando al código ya existente mientras es desarrollado, probado y entregado, utilizando herramientas de versionamiento de código.
- **Creación de un build en 10 minutos:** se utilizara una herramienta para automatizar los builds. Dejando fuera todas las tareas manuales y errores humanos se pueden aumentar la velocidad de los builds una vez que se tiene el código listo para ser entregado y testeado
- **Historias:** la historia deberá contener una descripción de qué debe hacer, en términos significativos, cada pieza de funcionalidad que se le agregue al producto.

7.3.3 Kanban

Se intentara aplicar la premisa de Kanban de empezar a implementar el trabajo tan pronto como esté listo para hacerlo, para poder avanzar con el mismo y no generar bloqueos mientras se espera por otros elementos.

Se tratara de adoptar los siguientes valores:

- **Transparencia:** será necesario que se comparta toda la información que se tenga entre todos los miembros del equipo de forma clara para que haya un entendimiento de todos los miembros del equipo.
- **Colaboración:** se tratará de mejorar la forma en la que todos los miembros del equipo trabajan juntos y colaboran entre sí para mejorar la calidad del producto.
- **Comprensión:** será necesario que todos los miembros del equipo tengan un conocimiento claro de que es lo que se quiere lograr desde el principio para así mejorar el producto final.
- **Acuerdo:** se deberá llegar a un acuerdo entre todos los miembros del equipo de cuál será el objetivo de cada una de las funcionalidades que se irán agregando así como del resultado final.

Además se tratara de incorporar las siguientes prácticas:

- **Implementar circuitos de retroalimentación:** se intentará tener la retroalimentación del dueño del producto y del equipo de Testing siempre que se pueda para asegurarse que el equipo de desarrollo no está alejándose del objetivo deseado.
- **Mejorar colaborativamente, evolucionar experimentalmente:** se busca que este sea un proceso que pueda ir mejorando y que se vayan solventando las fallas que pueda tener. Será necesario que entre todos los miembros del equipo se sugieran, prueben y acepten o desechen mejoras para el proceso en función de los imprevistos que vayan surgiendo en el desarrollo del producto.

7.3.4 Crystal

Como se destacó cuando se vio Crystal, esta metodología potencia los puntos fuertes y cubre las debilidades de la organización, además, es apropiada para grupos que desean crear su propio proceso para desarrollo y entrega de software.

Se tomaran las siguientes propiedades de Crystal:

- **Entrega frecuente:** se intentará entregar funcionalidad tan frecuentemente como sea posible.
- **Mejora reflexiva (o Mejora Continua):** se buscará mejorar el proceso apelando a la experiencia que los miembros del equipo obtengan durante el proyecto siempre llegando a un acuerdo común.
- **Comunicación cercana:** se tratará de tener una comunicación diaria y fluida entre todos los miembros del equipo tanto para aclarar dudas como para evitar bloqueos.

También se intentara adoptar el siguiente valor de la metodología Crystal:

- Tanto los documentos de diseño como los requerimientos tendrán el nivel de detalle necesario para que no haya ninguna duda a nivel funcional o técnica de que es lo que debe hacer el producto o la funcionalidad que se está agregando por parte de ninguno de los integrantes del equipo

7.3.5 Lean

Partiendo de la idea central de Lean de maximizar el valor para el cliente y minimizar el desperdicio, se intentará llevar al proceso de desarrollo algunos de los principios Lean identificados por Mary y Tom Poppendieck:

- **Eliminar Residuos:** Primeramente intentar reducir y si es posible eliminar los defectos, este principio Lean está completamente alineado con el objetivo de esta investigación. Además es importante evitar el traspaso de tareas relacionadas entre distintos desarrolladores o grupos ya que en el traspaso siempre se pierde algo de información adquirida en el análisis de quien trabajó primero dicha tarea. Se evitarán reuniones extensas e innecesarias ya que las

interrupciones disminuyen la productividad. En el caso de que sea necesario tener una serie de reuniones se intentará disponer las mismas en horarios continuos para minimizar las interrupciones del trabajo de los desarrolladores. Además, los involucrados en las reuniones deberán asistir con una preparación previa del tema a tratar en la misma, así se evitarán pérdidas de tiempo por reuniones excesivamente extensas en las cuales no hay un hilo a seguir u objetivo.

- **Embeber la calidad:** Se incorporará la realización de test unitarios (pruebas) para cada funcionalidad que se agregue. De esta manera, si luego se modifica el código afectado por las pruebas, los errores saltarán a la vista al correrlas haciendo visibles a su vez los puntos que deben corregirse.

- **Crear conocimiento:** Se documentarán los cambios, esto permitirá tener un registro de lo que se hizo y porqué se hizo dando un contexto del pedido del cliente y del análisis de los desarrolladores que lo trabajaron.

- **Aplazar el compromiso:** En línea con este principio, las historias se agregaran al sprint cuando tengan toda la información necesaria para trabajarla. Sino urge entregar esa historia lo mejor es postergar su inclusión en el sprint, sin esperar demasiado ya que el no completarla no debería retrasar otras historias o aspectos del proyecto.

- **Respeto a las personas:** Todo el equipo estará involucrado en mejorar el proceso. Ante la detección de una falencia en el mismo todos tienen la posibilidad de hacerlo notar para corregirlo y sumar calidad al proceso y por lo tanto al producto.

7.4 Tailoring de la solución

Si bien la aplicación de las mejores prácticas que se vieron en los puntos anteriores en la teoría sería lo ideal, en la práctica deberán ajustarse a ciertas restricciones impuestas tanto por el cliente, como por la propia compañía que realiza el sistema para este. Es aquí donde jugará un papel fundamental el concepto de tailoring, ya que se deberán ajustar las definiciones de la teoría no solo al entorno del cliente, sino también se deberán adaptar las mismas para que puedan coexistir entre las distintas metodologías que se integraran. Hay que recordar que no solo se están usando las mejores prácticas de una metodología o proceso sino que se busca integrar las mejores prácticas de varios de estos.

Durante el siguiente punto se desarrollará la solución y se adaptaran entonces, estas definiciones para llegar a un proceso que pueda cumplir con las necesidades y restricciones particulares tanto de la compañía que desarrolla el sistema como del cliente final permitiendo, además, la integración de las mejores prácticas de las distintas metodologías y procesos.

CAPÍTULO 8. DESARROLLO DE LA SOLUCIÓN

En los capítulos anteriores se describió en detalle las mejores prácticas de las metodologías tradicionales y ágiles, y cómo estas benefician a los entornos donde son aplicadas. También se seleccionaron cuáles de estas mejores prácticas se utilizaran para elaborar nuestro proceso de desarrollo.

En este capítulo se detallaran cada una de las partes del proceso de desarrollo, su estructura, sus actores y las herramientas de las cuales se vale para darle solución a los distintos inconvenientes a los que se enfrentaban con el anterior proceso de desarrollo.

8.1 Características del proceso de desarrollo

Para comenzar con una idea general del proceso de desarrollo se enumeraran sus elementos principales a continuación.

El nuevo proceso de desarrollo cuenta con 2 fases principales en donde se realizará el desarrollo del producto, la fase de diseño y la fase de desarrollo. Cada una de ellas cuenta con diversas actividades realizadas por los diferentes actores.

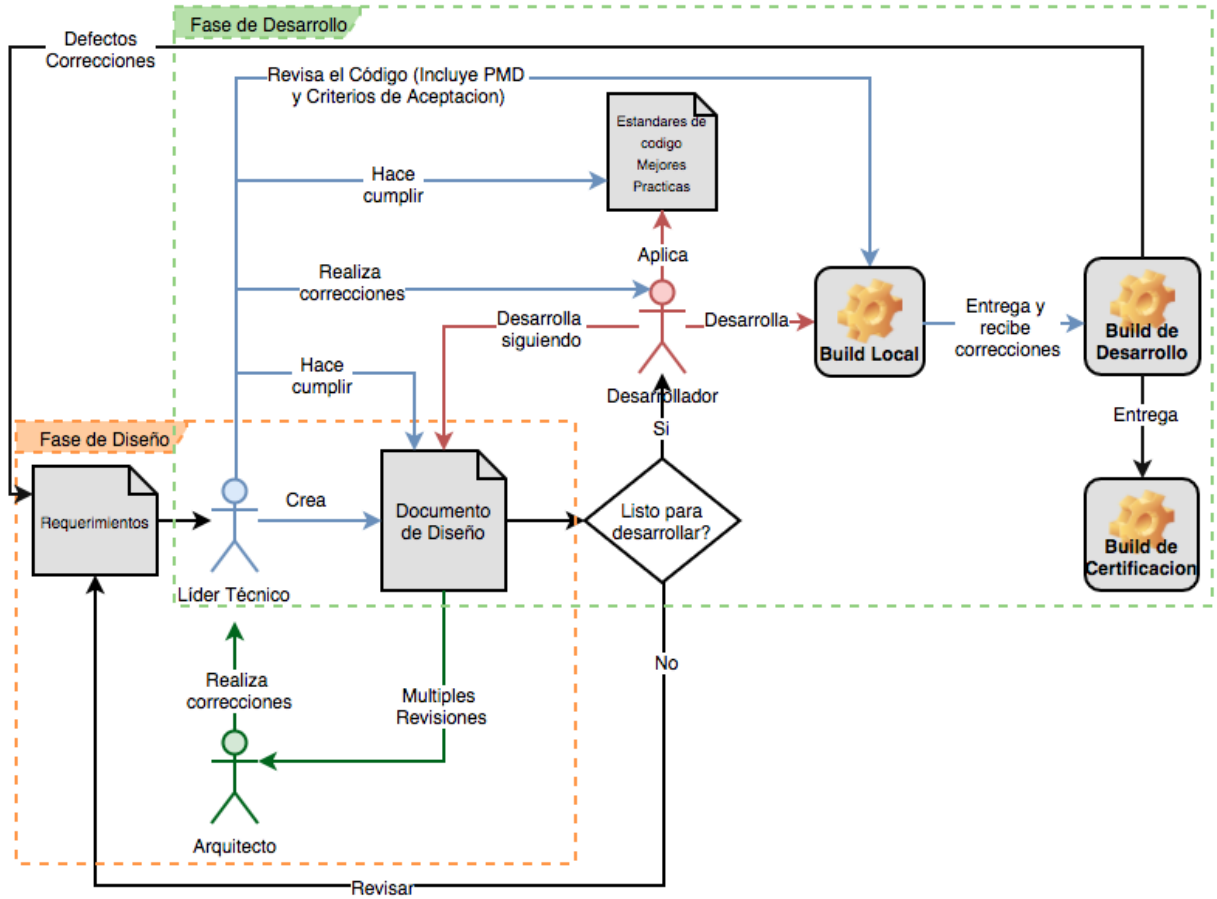

Figura 25. Diagrama de flujo del nuevo proceso de desarrollo, elaboración propia basada en la investigación y el desarrollo del nuevo proceso.

8.1.1 Fase de diseño

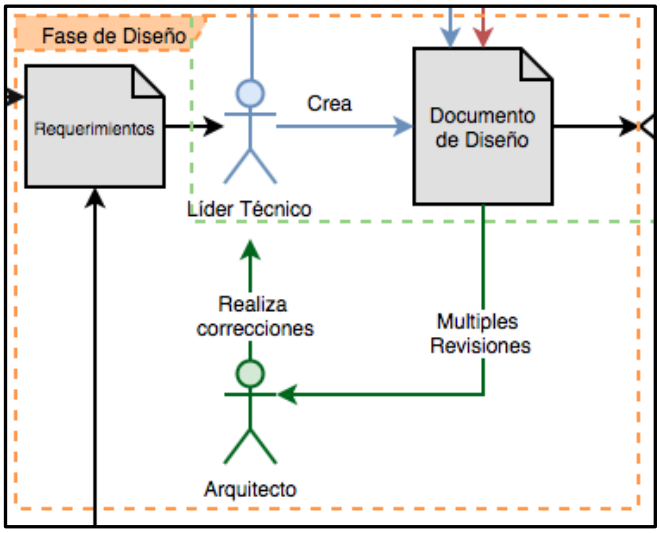

Figura 26. Diagrama de flujo con foco en la fase de diseño, elaboración propia basada en la investigación y el desarrollo del nuevo proceso.

Las actividades desarrolladas en esta fase tienen que ver con todo lo relacionado a definir, clarificar y documentar toda la información necesaria para que los desarrolladores puedan realizar su trabajo. La Fase de diseño contempla las actividades que se realizan dentro de los sprints que tiene el desarrollo del producto. Estas actividades se repetirán y serán las mismas en cada uno de los Sprints.

La **entrada** para esta fase son los requerimientos del cliente y la **salida** es el documento de diseño aprobado por el cliente.

Sprint

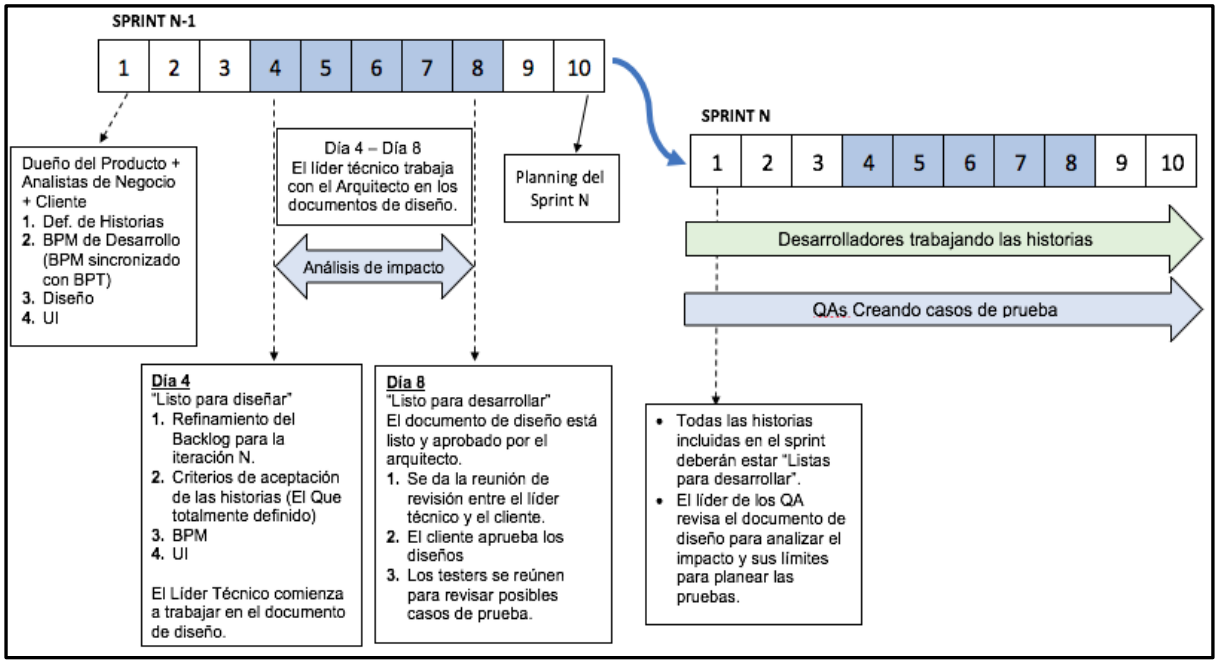

Figura 27. Diagrama donde se detalla las actividades dentro de Sprint, elaboración propia basada en la investigación y el desarrollo del nuevo proceso.

Los Sprints tendrán una duración de 2 semanas (10 días hábiles) y estarán divididos en 5 secciones o etapas. Existirán 2 tipos distintos de Sprints, durante el primer Sprint no se realizará ningún desarrollo hasta el día 8 donde el documento está aprobado y todo está listo para empezar a desarrollar. Desde el Sprint 2 hasta el Sprint N (último Sprint del proyecto), los desarrolladores estarán desarrollando código basado en el documento de diseño aprobado por el cliente y el arquitecto durante el sprint anterior.

A. FASE DE DISEÑO: Día 1 al 3

Del día 1 al 3 se realizarán tareas que serán llevadas a cabo por parte del cliente.

A.1. Actividades del día 1 al 3

El Dueño del Producto tendrá reuniones con el área del cliente que tiene más conocimiento del negocio quienes le comunicarán sus requerimientos para el producto.

A.1.1 Definición de User Stories

El Dueño del Producto deberá identificar los requisitos del producto y escribirlos en lenguaje técnico así como también los criterios de aceptación para los desarrolladores y testers.

A.1.2 Creación del Modelo del Proceso de negocio

El Dueño del producto también deberá tomar los requerimientos identificados en las reuniones con el área de negocios del cliente y transformarlos en casos de Uso de negocio para luego realizar el modelo de negocio del producto.

A.1.3 Creación de Interfaces de Usuario

A su vez deberá diseñar las interfaces de usuario (en los casos en que fueran necesarias) basándose a los indicaciones del cliente.

A.2. Actores que intervienen en los días 1 al 3

Si bien no se tiene influencia sobre la organización interna del cliente, se busca que se comprometa a seguir el proceso de desarrollo al pie de la letra, asignar los roles requeridos a su personal interno y entregar la información necesaria en tiempo y forma. Los actores de esta primera etapa se mencionan a continuación:

A.2.1 Dueño del Producto

Es la persona dentro del proyecto que tiene más conocimiento sobre el producto y será el contacto directo entre el los desarrolladores y el equipo del cliente que maneja el negocio. Será también el encargado de realizar los diagramas de modelo de negocio traduciendo los requerimientos del cliente en casos de uso.

A.2.2 Cliente

Será la o las personas de lado del cliente que entregarán los requerimientos del producto al dueño del producto y le darán toda la información acerca de que es lo que se espera del mismo. Normalmente es gente sin conocimientos técnicos, por lo cual delegaran los detalles técnicos en el Dueño del Producto y se enfocaran solo en las necesidades del negocio.

B. FASE DE DISEÑO: Día 4

Durante el día 4 se llevarán a cabo reuniones entre el equipo. En estas reuniones se realizará el Refinamiento del Backlog y el cliente presentará las Historias (con sus respectivas descripciones, criterios de aceptación e Interfaces de usuario) y el BPM de las mismas, en las que se estuvo trabajando en los 3 primeros días del Sprint.

B.1. Actividades del día 4

Se revisará que tengan toda la información necesaria para que el equipo de desarrollo realice sus actividades. Si llegase a faltar información, se encontrará ambigüedad en alguna parte de los documentos o fuese necesario aclarar algo se hará durante estas reuniones. Si se determina que todo está correcto, el equipo de desarrollo aceptara los documentos presentados.

B.1.1 Refinamiento del Backlog

En esta reunión se hará una revisión de las historias que se encuentran en el backlog y se determinara si la información que estas contienen es precisa, completa y no existen dudas al respecto de la misma. Si hubiese dudas el Dueño del producto se llevará la tarea de reunirse con el equipo del negocio y aclarar las mismas para agregar esta información en las historias. Las historias con la información actualizada serán revisadas en una próxima reunión de refinamiento del backlog (para ser incluidas en un Sprint futuro).

B.1.2 Revisión de las Historias

Se hará una revisión de las historias, las mismas deberán tener al menos una descripción, uno o más criterios de aceptación y, si hicieran falta, uno o más diseños de las interfaces de usuario.

B.1.3 Revisión de las Descripciones

Contendrá una descripción de la funcionalidad en base a los requisitos que el cliente quiere darle al sistema. La misma estará escrita en el lenguaje del usuario y será completa, clara y precisa.

B.1.4 Revisión del Criterio de aceptación de la Historia

Todas las historias deberán tener al menos 1 o más criterios de aceptación, los mismos se entienden como los mínimos requisitos que debe cumplir la nueva funcionalidad. De no cumplir con estos requisitos la historia no será aceptada como completa.

B.1.5 Revisión de las Interfaces de Usuario

En caso de que la Historia Requiera una modificación en una o más Interfaces de Usuario, el Dueño del producto debe adjuntar imágenes del resultado esperado. El desarrollador usará la imagen como guía para realizar los cambios en la Interfaz de Usuario y la misma debe quedar igual que la imagen provista, en caso contrario la historia no será aceptada como completa.

B.1.6 Revisión del BPM

El dueño del producto realizará la presentación del Modelo de Negocio en el cual se representará la funcionalidad de la Historia en forma de UML. El Modelo de Negocio mostrará paso a paso la lógica que debe seguir la funcionalidad desde la visión del negocio.

B.2. Actores que intervienen en el día 4

En esta etapa interactúan tres actores principales, si bien pueden sumarse otros actores e interesados, normalmente solo será por cuestiones informativas y no tendrán influencia en las decisiones. Los 3 actores principales son:

B.2.1 Dueño del producto

En esta etapa el dueño del producto realizará la presentación de los documentos y responderá dudas sobre los mismos. También funciona como vínculo entre el equipo de desarrollo y el cliente en caso de que se necesite más información.

B.2.2 Líder Técnico

Es quien lidera el grupo de desarrollo, normalmente tiene un mayor conocimiento técnico y más experiencia trabajando en la aplicación. Trabajará activamente en el refinamiento del backlog, en la revisión de los criterios de aceptación, del BPM y de

las Interfaces de Usuario. Asimismo empezara a trabajar en las áreas técnicas del documento de diseño.

B.2.3 Desarrolladores

Son el resto de los integrantes del equipo de desarrollo, además del Líder Técnico. Son quienes harán el desarrollo del código que brindara la solución en base al documento de diseño. El equipo de desarrollo trabajará junto con el Líder Técnico en el refinamiento del backlog, en la revisión de los criterios de aceptación, del BPM y de las Interfaces de Usuario.

C. FASE DE DISEÑO: Días 5 al 7

Una vez que las historias que se trabajaran en el Sprint fueron catalogadas como "Listas para ser trabajadas" y fueron ingresadas al Sprint Backlog, El líder técnico invertirá estos 3 días para hacer la investigación necesaria para determinar la mejor solución técnica para cada historia.

C.1. Actividades de los días 5 al 7

C.1.1 Análisis y documentación

El Líder Técnico hará un análisis detallado del BPM y lo usará como base y guía para el desarrollo de la solución. Realizará una investigación detallando el impacto que generarán en el código los cambios requeridos para esta solución. Además del análisis del impacto creará un modelo de lógica que traducirá el BPM realizado con una visión de negocio en un modelo técnico. El mismo tendrá una vista de los componentes de la aplicación impactados en este cambio. También invesntigará y dejará asentado en el documento de diseño si la Historia requiere de cambios en la Experiencia de Usuario, cambios en la Base de Datos o si para completar la Historia se requiere consumir algún Servicio Web específico.

Luego hará una descripción detallada de cuáles serán las clases y los métodos de las mismas impactados por los cambios que se realizarán para esta solución así como también se detallará si se crearan nuevos métodos/clases y una descripción de que función cumplirán las mismas.

C.1.2 Revisión del documento con el Arquitecto de la aplicación

Una vez que el líder técnico tenga toda la información necesaria y haya completado el documento de diseño, se reunirá con el arquitecto de la aplicación y revisarán uno a uno los segmentos del documento de cada una de la Historias que serán trabajadas en el Sprint. El Arquitecto realizará las correcciones apropiadas si las hubiere, o en caso de que no haya correcciones/mejoras para realizar, aprobará los cambios detallados en el documento de diseño y el mismo estará listo para ser presentado al resto del equipo.

C.2. Actores que Intervienen en los días 5 al 7

C.2.1 Líder Técnico

En esta etapa el Líder Técnico completara el documento de diseño con todos los datos necesarios y se los presentara al Arquitecto de la aplicación.

C.2.2 Arquitecto de la aplicación

El Arquitecto de la aplicación es quien tiene un mayor conocimiento de la aplicación y de los cambios que están llevando a cabo los distintos equipos que trabajan sobre la aplicación. Será el encargado de tomar las decisiones a nivel arquitectónico y, además, será el referente técnico de los Lideres Técnicos. Es el encargado de revisar el documento de diseño junto con el líder técnico y además es quien tiene la última palabra a la hora de aprobar todos los cambios que se realizarán en la aplicación. En esta etapa, el Arquitecto de la Aplicación proveerá sus recomendaciones y finalmente aprobará el documento de diseño.

D. FASE DE DISEÑO: Día 8

D.1. Actividades del día 8

Como se dijo antes, el documento de diseño estará ya revisado y aprobado por el arquitecto, durante el día 8 se realizará una reunión con el Dueño del Producto en la cual participará el equipo de desarrollo así como también el equipo de Testing y el Scrum Master.

En la reunión se hará una presentación de cada una de la Historias y se revisaran todos los puntos que fueron agregados al documento por el Líder Técnico. El Dueño

del producto podrá realizar preguntas para aclarar dudas que pudiera tener y si está de acuerdo con el documento de diseño dará su aprobación.

Una vez que se hayan revisado y aprobado todas las Historias y todos los miembros del equipo tengan un mismo entendimiento de que es lo que se va a realizar, se dará por concluida la reunión y los desarrolladores podrán empezar con el desarrollo del código de la solución.

Para el desarrollo del código los desarrolladores utilizarán como única guía el documento de diseño aprobado. Si surgen dudas al respecto podrán consultar al Líder Técnico que brindará su soporte y guía pero nunca deberán cambiar lo que fue acordado en el documento.

D.2. Actores que intervienen en el día 8

D.2.1 Dueño del producto

En esta etapa el Dueño del producto asistirá a la reunión de revisión del documento de diseño, podrá hacer todas las preguntas que necesite para clarificar dudas y finalmente es quien tiene el deber de aprobar o no el documento de diseño.

D.2.2 Líder Técnico

En esta etapa el Líder Técnico hará la presentación del documento de diseño, explicara cada uno de los puntos que se detallan en la Historias que serán trabajadas en el Sprint y contestara las dudas que pudieran surgir durante la reunión por parte del Dueño del Producto.

D.2.3 Desarrolladores

Los desarrolladores deberán asistir a las reuniones pero no tendrán interacción en las mismas a menos que el Líder Técnico lo pida por alguna cuestión particular.

D.2.4 Testers

No se tiene influencia en el equipo de Testing pero el cliente se compromete a que su equipo de Testing se reúna y genere los casos de prueba para el momento de testear las Historias.

E. FASE DE DISEÑO: Día 10

Se realizarán 2 reuniones en las cuales participaran todo el equipo y los interesados.

E.1. Actividades del día 10

E.1.1 Reunión de Planeamiento del Próximo Sprint

La reunión de planeamiento del próximo Sprint consiste en hacer un relevamiento de las Historias del backlog previamente ordenadas por prioridad de importancia para el producto y realizar una estimación y puntuación de las mismas. Una vez que fueron puntuadas, se podrán elegir las Historias serán trabajadas en el próximo Sprint por el equipo de desarrollo.

E.1.2 Reunión de Retrospectiva del Sprint

La Reunión de Retrospectiva del Sprint se realizará justo al finalizar la Reunión de Planeamiento del Próximo Sprint. En la misma participaran todos los miembros del equipo de desarrollo, el Scrum Master, el Dueño del Producto, el Equipo de Testing y pueden ser invitados otros interesados.

E.2. Actores que intervienen en el día 10

E.2.1 Scrum Master

Asistirá a las reuniones, actuará como moderador y facilitará la adquisición de recursos externos cuando sean necesarios para sortear bloqueos o mejorar el proceso. Además deberá dar su opinión sobre que salió bien y que salió mal durante el Sprint y por último debe por lo menos proponer una mejora para que sea trabajada en el próximo Sprint en la reunión de la Retrospectiva.

E.2.2 Dueño del Producto

El Dueño del Producto debe asegurarse de que el backlog este correctamente priorizado y además que las Historias tengan la información necesaria para poder ser trabajadas en la reunión de planeamiento del próximo Sprint.

Por otro lado deberá dar su opinión sobre que salió bien y que salió mal durante el Sprint y por último debe por lo menos proponer una mejora para que sea trabajada en el próximo Sprint en la reunión de Retrospectiva.

E.2.3 Desarrolladores

Los Desarrolladores participaran activamente en la reunión de planeamiento del próximo Sprint asegurándose que las Historias tengan la información necesaria para ser trabajadas. Además deberán dar su opinión sobre que salió bien y que salió mal durante el Sprint y por último deben proponer por lo menos una mejora para que sea trabajada en el próximo Sprint en la reunión de Retrospectiva.

E.2.4 Líder Técnico

El Líder Técnico como los desarrolladores participara activamente en la reunión de planeamiento del próximo Sprint asegurándose que las Historias tengan la información necesaria para ser trabajadas. También deberá dar su opinión sobre que salió bien y que salió mal durante el Sprint y por último debe proponer por lo menos proponer una mejora para que sea trabajada en el próximo Sprint en la reunión de Retrospectiva.

E.2.5 Testers

Los Testers no tendrán mucha participación en la reunión de planeamiento del próximo Sprint pero si deberán dar su opinión sobre que salió bien y que salió mal durante el Sprint y por último deben proponer una mejora para que sea trabajada en el próximo Sprint en la reunión de Retrospectiva.

E.2.6 Otros Interesados

Pueden ser miembros del equipo del cliente u otro miembro de algún equipo técnico que pueda aportar alguna mejora al proceso o ayudar con algún bloqueo tanto de negocio como técnico o de procedimiento.

8.1.2 Fase de desarrollo

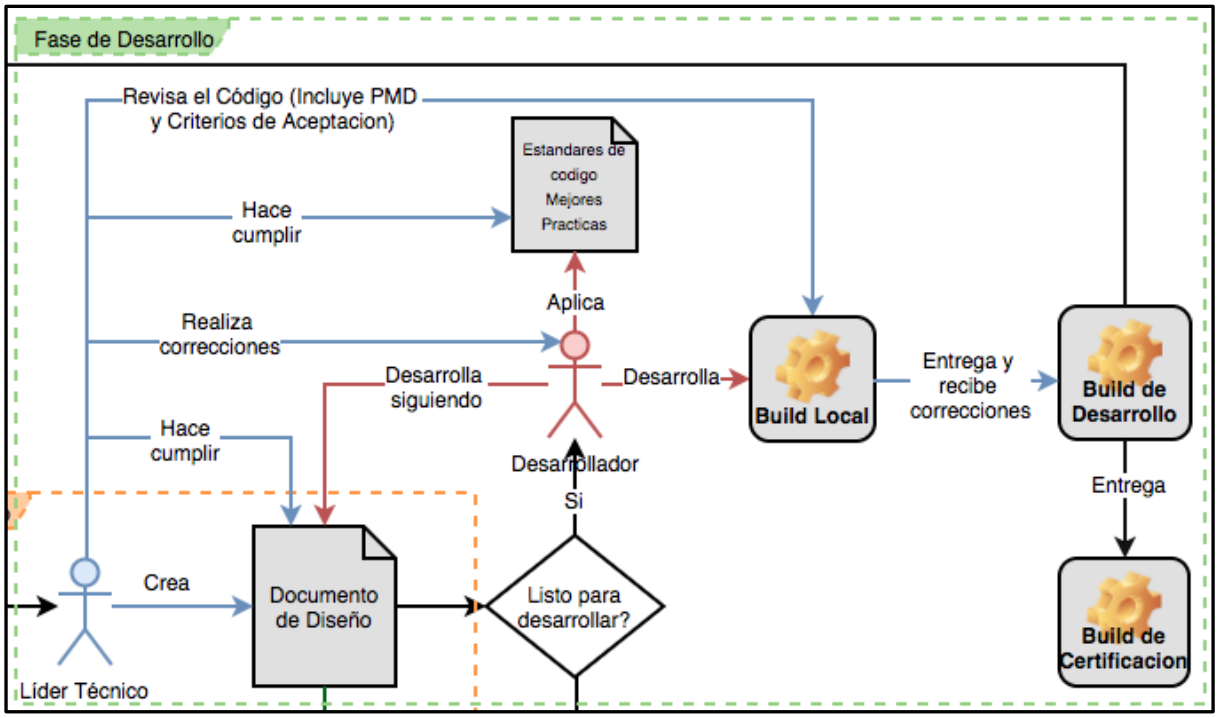

Figura 28. Diagrama flujo con foco en la fase de diseño, elaboración propia basada en la investigación y el desarrollo del nuevo proceso.

En la fase de desarrollo se hará foco en detallar todas las actividades técnicas relacionadas al desarrollo del código así como también las actividades de revisión del mismo y trabajos adicionales que se realizan en este. No se detallaran las actividades de Testing porque son realizadas por personal del cliente pero serán nombradas porque las mismas pueden generar un retrabajo del equipo de desarrollo.

Esta fase requiere como **entrada** el documento de diseño aprobado por el cliente y listo para trabajar y sus **salidas** son el Build de Desarrollo y el Build de Certificación.

Sprint

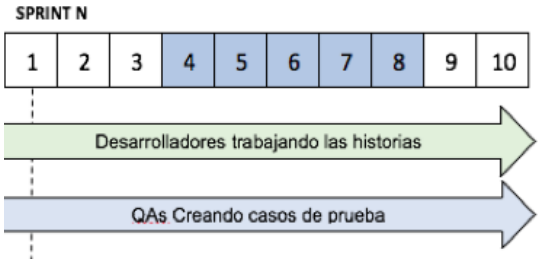

Figura 29. Detalle las actividades realizadas en los Sprints que siguen al primer Sprint, elaboración propia basada en la investigación y el desarrollo del nuevo proceso.

Como se vio anteriormente, durante el primer Sprint prácticamente no se realizarán tareas de desarrollo. Recién a partir del segundo Sprint comenzarán las tareas de desarrollo y las mismas continuarán hasta el último Sprint.

Las tareas de desarrollo que se realizarán durante el Sprint, a diferencia de las tareas de diseño, no serán divididas en etapas, las mismas serán continuas (no habrá tiempos muertos ni momentos en los que los desarrolladores no tengan historias listas para desarrollar en el documento de diseño) y se basaran en el documento de diseño que fue completado, revisado y aprobado en el Sprint anterior.

A.1. Actividades de la fase de desarrollo

Se detallaran a continuación todas las actividades que se realizan durante la fase de desarrollo.

A.1.1 Desarrollo de Código

Los desarrolladores utilizaran el documento de diseño aprobado por el cliente como base para realizar el desarrollo de código que cumplirá con los requisitos pedidos por el cliente. Utilizan además para el desarrollo del código las mejores prácticas del lenguaje provistas por Oracle y los estándares de código.

Una vez que finalizan el desarrollo del código realizarán el pruebas unitarias en los entornos locales que cada desarrollador posee en su computadora personal. El desarrollador hará la validación de su código contra cada uno de los criterios de aceptación provistos en las historias y considerando el documento de diseño.

Si el código pasa satisfactoriamente todos los criterios de aceptación en el entorno local entonces el desarrollador está listo para promover el código al entorno de desarrollo utilizando un versionador de código.

Una vez que el código ha sido promovido, el desarrollador genera un Build de Desarrollo utilizando una herramienta de integración continua y notifica a todo el equipo, con lo cual el equipo de Testing comienza a realizar sus pruebas.

A.1.2 Generación, Validación y Entrega de Builds de Desarrollo

Como se indicó en el punto anterior, es el Desarrollador quien se encargará de generar el Build de Desarrollo. Por lo tanto también le corresponde validar que el Build se haya generado correctamente y realizar las mismas pruebas de la funcionalidad agregada que realizó en su entorno local.

Si el Desarrollador encontrara alguna irregularidad con el Build de Desarrollo (ya sea por la nueva funcionalidad o por otro problema no determinado) procederá a realizar la investigación correspondiente y el análisis para determinar cuál es el problema y como resolverlo o, en el caso que sea un problema externo, involucrar al equipo que debe darle resolución.

Si el Build de Desarrollo pasa todas las pruebas y validaciones realizadas por el Desarrollador correctamente, este pasara a notificar al equipo de Testing que hay un nuevo Build de Desarrollo disponible para realizarle las pruebas correspondientes a la historia trabajada. Además el Desarrollador notificará a todo el equipo y otros interesados en este mail para mantener a todos informados.

En este punto el equipo de Testing realizará todas las pruebas y validaciones necesarias sobre el Build de Desarrollo. Si hay alguna validación que no supera los tests el equipo de Testing creara una tarea del tipo Defecto para el equipo de desarrollo.

A.1.3 Revisión de Código (Code Review)

Una vez que los Desarrolladores hayan finalizado sus tareas de desarrollo y entreguen el Build de Desarrollo y mientras los Testers realizan las pruebas y validaciones, el Líder técnico realizará la revisión de código.

La revisión de código tendrá 3 puntos principales sobre los que trabajará el Líder Técnico:

A.1.4 Revisión de errores de código

El Líder Técnico revisará las líneas de código que el Desarrollador promovió al ambiente de Desarrollo y se asegurará que estas líneas no contengan errores de lógica o de sintaxis. El Líder Técnico buscará principalmente salvar errores de nullpointer, flujo del código mal definido, condicionales innecesarios, etc.

A.1.5 Revisión de mejores prácticas

El Líder Técnico hará una revisión del código promovido y comprobará que el mismo haya sido desarrollado teniendo en cuenta las mejores prácticas definidas para el lenguaje.

A.1.6 Revisión de optimización

El Líder Técnico revisara el código que ha sido promovido y buscará optimizar el tiempo de procesamiento del mismo así como también reducir la cantidad de líneas de código y hacer un código más modular.

A.1.7 Ejecución de herramienta para revisión de código estático

Además de la revisión manual del código, el Líder Técnico ejecutara un software de revisión de código estático. Este analizador encuentra fallas comunes de programación como variables no utilizadas, bloques de catch vacíos, creación de objetos innecesarios, código duplicado, etc.

A.1.8 Soporte al desarrollador para realizar la mejora

Una vez finalizados los trabajos de revisión, el Líder Técnico informará los resultados de su análisis al Desarrollador que trabajo la historia y hará saber a este si hacen falta realizar trabajos extras sobre el código para mejorar o corregir errores en el mismo. También realizará tareas de soporte y capacitación al Desarrollador asegurándose que este entienda y aprenda de los errores para no volver a cometerlos en futuros desarrollos.

A.1.9 Retrabajo (si existiese)

Las tareas de retrabajo serán realizadas por el Desarrollador y las mismas pueden tener su origen en 4 puntos distintos.

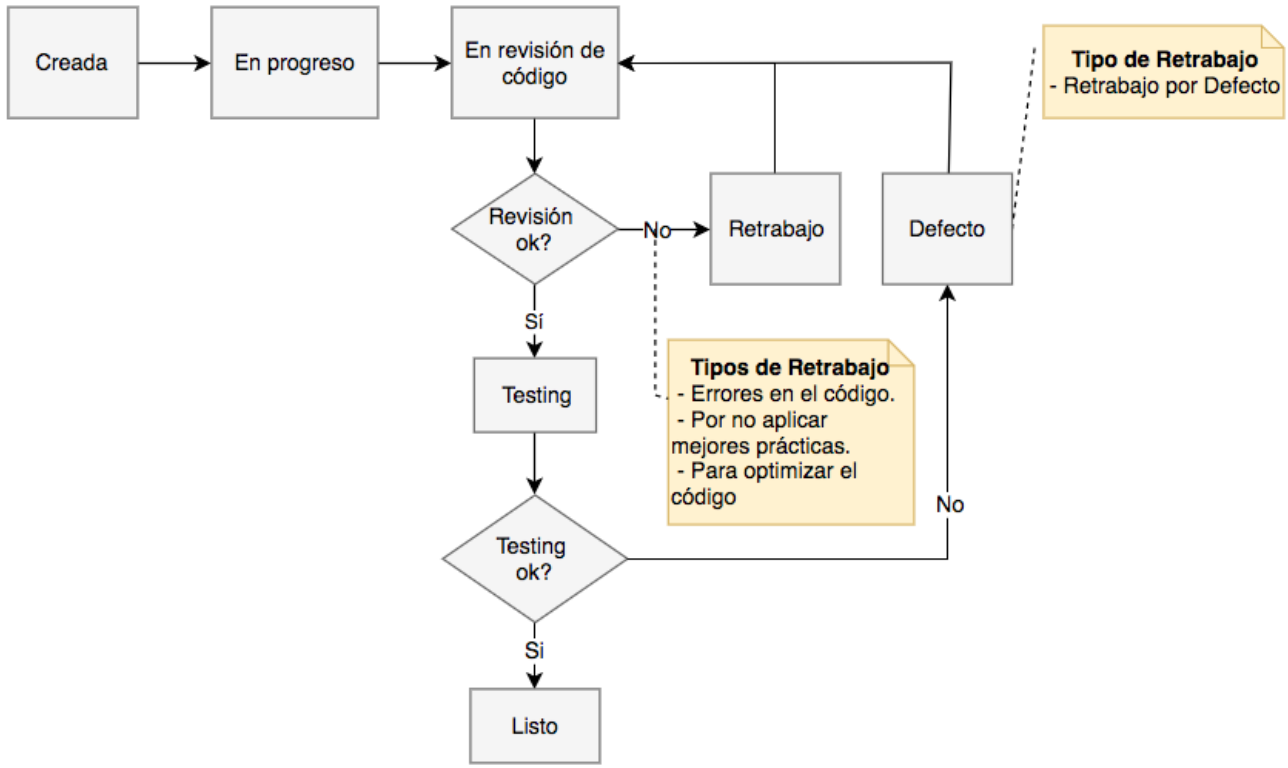

Figura 30. Diagrama de flujo de retrabajo, elaboración propia basada en la investigación y el desarrollo del nuevo proceso.

a) Retrabajo por defecto

El retrabajo por defecto se da cuando durante la fase de Testeo el equipo de Testing confirma que alguno de los criterios de aceptación que se encuentran detallados en el documento de diseño no se cumple. En este caso abren una tarea del tipo Defecto en la Historia y notifican al Desarrollador que trabajó la misma y a todo el equipo. En esta tarea se deberán detallar los pasos realizados para reproducir el error, se deben proporcionar datos de prueba si hiciera falta, y además, el resultado obtenido así como también el resultado esperado.

b) Retrabajo por errores en el código

Este tipo de retrabajo ocurrirá cuando, durante la Revisión de errores de código, el Líder Técnico detecte en su análisis que se deben realizar cambios en el código

promovido. Hay distintos tipos de errores que pueden generar este tipo de retrabajo, el Líder Técnico notificará únicamente al Desarrollador informando su análisis y las correcciones que deben ser realizadas. El Desarrollador realizará dichas correcciones (con soporte del Líder Técnico de ser necesario), hará las pruebas para validar la funcionalidad y volverá a subir el código generando un nuevo Build de Desarrollo.

c) Retrabajo por optimización del código

El retrabajo por optimización del código provendrá del Líder Técnico al hacer la revisión del código promovido. Si el líder técnico detecta que pueden realizarse cambios que harán que el código sea más performante o puede escribirse un código de manera más limpia y legible, este notificará únicamente al Desarrollador que trabajó la Historia informando cuales son los cambios a realizar. El Desarrollador realizará los cambios (pidiendo soporte al Líder Técnico de ser necesario) y hará las validaciones necesarias para la Historia en cuestión. Una vez validado el código, éste será promovido por el Desarrollador y se generará un nuevo Build de Desarrollo.

d) Retrabajo por no aplicar las mejores prácticas

Por último, este tipo de retrabajo también será resultante del análisis de la revisión de código del Líder Técnico. Si el Líder Técnico detecta que el código promovido no fue desarrollado siguiendo las mejores prácticas del lenguaje, el mismo notificara, como en los casos anteriores, únicamente al Desarrollador que trabajo la historia. En la notificación incluirá las mejoras a realizar y el detalle de las mejores prácticas que deben aplicarse al momento de hacer el desarrollo y cuales no fueron aplicadas a modo de entrenamiento para el desarrollador. El desarrollador hará las correcciones y validaciones necesarias y, finalmente, promoverá el código y generará un nuevo build.

A.1.10 Generación, Validación y Entrega de Builds de Certificación

Una vez que el Build de desarrollo fue validado por el equipo de Testing pasando exitosamente todas las pruebas y además el Líder Técnico realizo todas las revisiones de código sin encontrar la necesidad de realizar un retrabajo, el Líder Técnico promoverá el código al ambiente de Certificación. Una vez promovido el código el Líder Técnico será el encargado de generar el Build de Certificación utilizando una herramienta de integración continua, este build se generará a solicitud de los testers

cuando consideren que hayan realizado pruebas suficientes en el ambiente de desarrollo. Además el Líder Técnico deberá validar que el mismo se genere correctamente así como también será su responsabilidad realizar pruebas básicas de los criterios de aceptación de las historias entregadas en el mismo. Si las pruebas básicas se completan con éxito se notificará que hay un nuevo build de Certificación disponible a todo el equipo (incluyendo el equipo de Testing y otros interesados). En el caso de que las pruebas fallen, el Líder Técnico realizara un análisis del error encontrado y trabajará en la solución del mismo involucrando a los desarrolladores o equipos externos si correspondiese.

A.2. Actores de la fase de desarrollo

A.2.1 Desarrollador

En la fase de desarrollo, el desarrollador trabajará en el código siguiendo el documento de diseño aprobado en la Fase de Diseño y aplicará los estándares de código y mejores prácticas del lenguaje. Será además, el encargado de promover el código al ambiente de desarrollo (en el versionador de código) y de generar los builds de Desarrollo que se necesiten, validar los mismos y notificarlos. También en el caso de que corresponda será responsable de trabajar en las tareas que surjan como Defecto (generado por el equipo de Testing) o Retrabajo (generado por el Líder Técnico).

A.2.2 Líder Técnico

En esta fase realizará las revisiones del código generado por los desarrolladores haciendo cumplir tanto el Documento de Diseño como los estándares de código y mejores prácticas del lenguaje. Además, será el encargado de ejecutar la herramienta que analiza el código estático. Una vez finalizadas estas actividades notificará a los desarrolladores acerca de los retrabajos necesarios cuando se haya encontrado errores u oportunidades de mejora en el código.

Por otro lado, será quien genere los Build de Certificación y realizará sus respectivas validaciones notificando a todo el equipo en caso de que el mismo esté listo para ser entregado. Si encontrase errores, le corresponderá realizar un análisis para buscar e implementar una solución.

A.2.3 Equipo de Testers

Ejecutarán los casos de pruebas asociados a cada historia utilizando como guía el documento de diseño. Deberán validar tanto los Build de Desarrollo como los de Certificación. Además en el caso de encontrar casos de prueba fallidos crearán las tareas del tipo Defecto para que el equipo de desarrollo pueda trabajar en la corrección. Será su responsabilidad incluir en estas tareas: pasos para reproducir el error, datos de prueba, resultado obtenido y resultado esperado.

8.1.3 Artefactos comunes a todas las fases

A continuación se detallaran los artefactos que son comunes a la fase de diseño y a la de desarrollo.

A. Documento de Diseño

Es el documento que incluirá toda la información relacionada con las historias trabajadas en un proyecto particular. Iniciará con determinada información del proyecto y se irá completando en cada Sprint con los detalles de las historias que se vayan trabajando. En este documento se podrá encontrar desde los requerimientos, pasando por el modelo de proceso de negocio y el diseño lógico hasta el diseño de interfaces específico de cada historia. Además se podrán encontrar detalles de cambios a nivel base de datos o consumo de servicios web.

Este documento estará aprobado por el equipo de desarrollo (incluyendo al Arquitecto) y por el Dueño de Producto en representación del cliente. El mismo además de servir como guía para los desarrolladores y el equipo de Testing funcionará como un contrato de lo que se espera y lo que no en el producto.

En el anexo se incluye un modelo de este documento (de desarrollo propio de los autores de este proyecto de grado).

B. Historias (User Stories)

Se entiende por "Historia" como una funcionalidad que se quiere agregar al sistema, la misma estará escrita en el lenguaje del usuario y contendrá al menos una descripción, uno o más criterios de aceptación y, si hicieran falta, uno o más diseños de las interfaces de usuario.

C. Estándares de código y mejores prácticas

Se hará referencia a los estándares y mejores prácticas de cada lenguaje en particular.

D. Defecto

Se entiende por defecto a la tarea creada por el equipo de Testing ante la detección de una falla en las pruebas de validación de las historias.

8.1.4 Comparación entre el proceso viejo y el proceso nuevo

En el punto 8.1 se vio un diagrama de flujo del nuevo proceso de desarrollo (**Figura 25**), ahora se volverá a presentar ese diagrama para compáralo con el anterior proceso de desarrollo y ver claramente las incorporaciones y modificaciones que se realizaron en el mismo.

Proceso de desarrollo anterior:

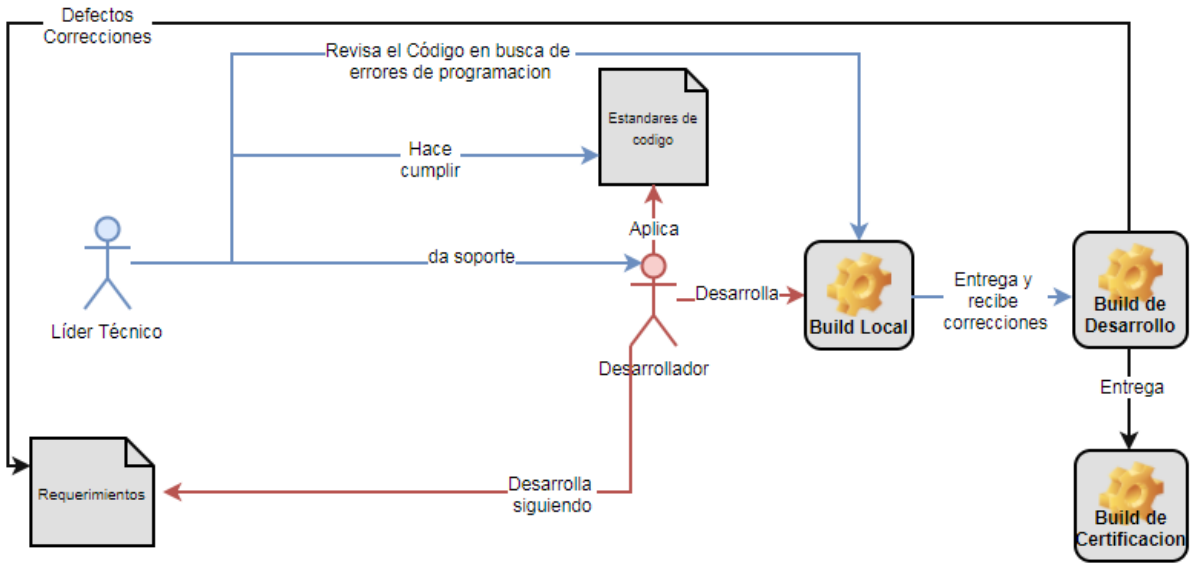

Figura 31. Diagrama de flujo de proceso de desarrollo anterior, elaboración propia basada en la investigación para el desarrollo de este trabajo.

Nuevo proceso de desarrollo:

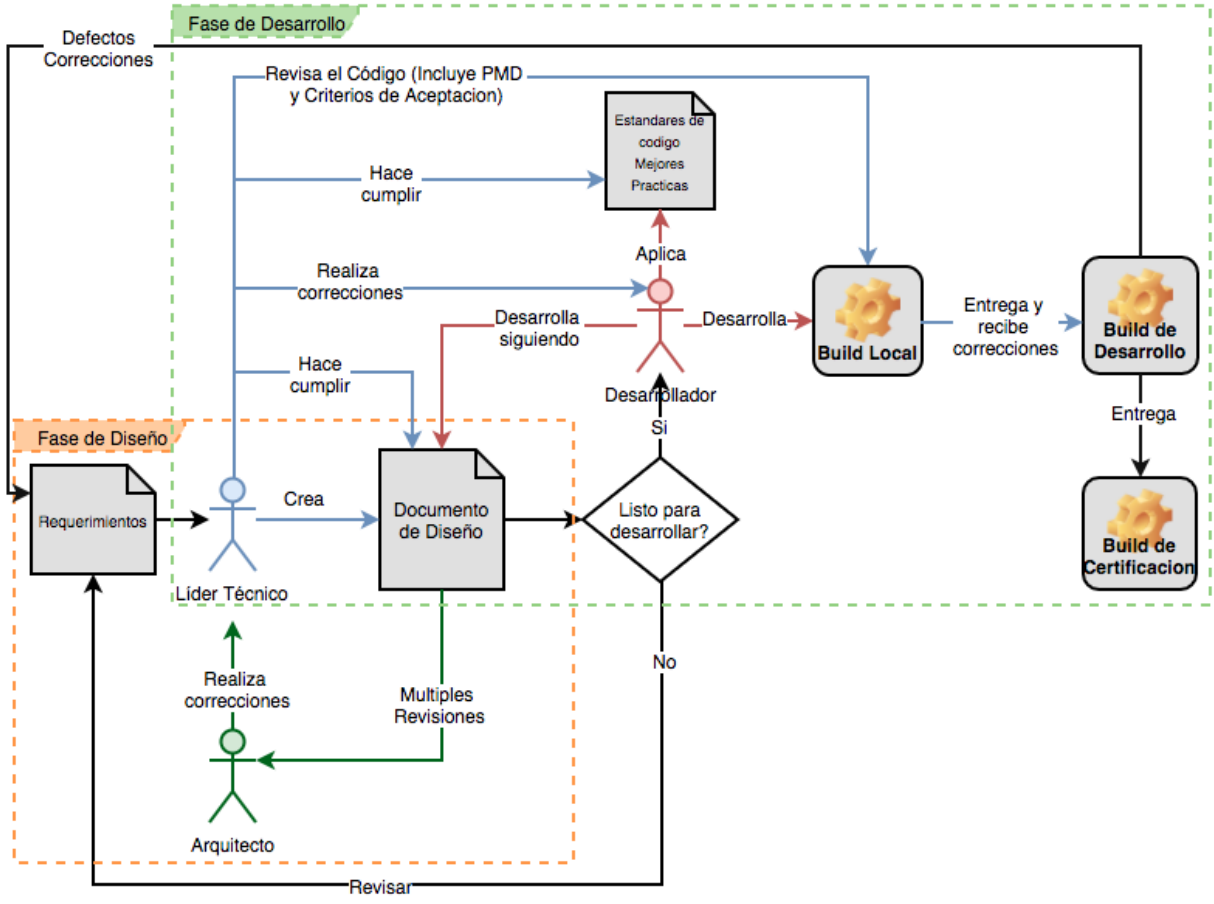

Figura 25. Diagrama de flujo del nuevo proceso de desarrollo, elaboración propia basada en la investigación y el desarrollo del nuevo proceso.

Como se puede ver claramente al comparar los 2 diagramas, que se dividió el proceso de desarrollo en 2 fases.

Por un lado, una fase de diseño en la que se agregó el documento de diseño agregando una tarea al Líder Técnico que es la creación del mismo. Se agrega además al Arquitecto y las tareas de revisión y corrección que este le realizara al documento de diseño. Esto significa que los requerimientos ya no son simplemente tomados por el desarrollador, sino que son revisados y procesados por el Líder Técnico y luego revisados y aprobados por el arquitecto.

Por otro lado está la fase de desarrollo, la cual solo empieza cuando el documento de diseño está listo para ser trabajado ya que el desarrollador se guiara con este para

realizar su trabajo en lugar de utilizar los requerimientos que vienen del cliente. Además se le agrega al Líder Técnico la tarea de hacer cumplir lo estipulado en el documento de diseño. También se actualizaron un poco las tareas que realizaba anteriormente como puede verse al comparar los gráficos, si bien sigue dando soporte no es algo que le ocupe tanto tiempo ya que el Desarrollador tiene el documento de diseño para guiarse, con lo cual el Líder Técnico puede dedicarse más a realizar correcciones, además se agrega a la revisión de código la ejecución de la herramienta de revisión de código estático.

8.2 Implementación del Proceso de Desarrollo en un caso práctico

A continuación se mostrara como las prácticas desarrolladas en los capítulos anteriores se han implementado experimentalmente en un caso práctico.

Se presentara la implementación del nuevo proceso de desarrollo en la segunda fase del proyecto que se utilizó como ejemplo en el diagnóstico de la situación actual. Se eligió este proyecto como modelo de experimentación dado que la segunda fase del mismo tendría características similares a la primera y por lo tanto se entendió que era un buen caso para el estudio de los resultados.

8.2.1 El proyecto

El proyecto en cuestión contó con una carga de trabajo de 70 historias durante un periodo de 6 meses (12 Sprints de 2 semanas cada uno). El equipo de desarrollo estuvo formado por un Líder Técnico y 2 desarrolladores, además contó con un Project manager, un Dueño del producto y un equipo de Testing. Todos los actores siguieron al pie de la letra los pasos detallados en el nuevo proceso de desarrollo.

Se trabajaron un promedio entre 5 y 6 Historias por Sprint (dependiendo de la complejidad de las mismas), para esto, el Dueño del producto completo y entregó toda la información y los detalles de las Historias en tiempo y forma y el Líder Técnico se ocupó de realizar el análisis y el diseño de las mismas en el Documento de Diseño. Los Desarrolladores realizaron todo el trabajo de desarrollo del código basándose en el documento de diseño aprobado por el Arquitecto y el Cliente. A su vez el equipo de Testing desarrollo todos sus casos de prueba también basándose en el documento de diseño aprobado.

Durante el tiempo que duró el proyecto se notó que hubo un gran trabajo previo al desarrollo pero a su vez disminuyó mucho el trabajo posterior a la entrega de las historias. A diferencia del proyecto anterior en donde las Historias se empezaban a desarrollar y durante el desarrollo de la historia los requerimientos se redefinían una y otra vez, en este proyecto toda la información detallada de que se realizaría en el código y al momento de Testearlo estaba disponible al momento de empezar con el desarrollo del mismo.

Se notó también un mayor orden en la distribución del trabajo, cada actor sabia cuáles eran sus responsabilidades, su función y los tiempos que tenía para hacer sus entregas y el nivel de detalle que tenían que tener las mismas disminuyendo así la entropía dentro del equipo.

Además disminuyo la cantidad de reuniones adicionales que debían llevarse a cabo para aclarar dudas ya que todo estaba documentado en el documento de diseño. Esto hizo que se optimizará el tiempo de trabajo durante el proyecto.

8.2.2 Resultados de la implementación

Habiendo finalizado el proyecto que se tomó como ejemplo, se volvieron a examinar las métricas de los defectos que se abrieron durante el mismo para observar las variaciones en los gráficos.

Primero, es importante destacar que en este proyecto se observaron solo 35 defectos en las 70 historias que se trabajaron, a comparación con los 106 defectos reportados en las 70 historias trabajadas durante la primera fase del mismo (sin utilizar el nuevo proceso de desarrollo). Esto implica que se pasó de tener 1,5 defectos por historia a 0,5 defectos por historia, es decir una disminución del 60% en los defectos de un proyecto.

Además, los defectos se volvieron a analizar desde los mismos 4 patrones con los que se los analizó al comenzar este trabajo de investigación.

A. Tipo de Problema

Tipo de Problema	Cantidad	Porcentaje	Porcentaje anterior
New Development	17	48,57%	81,13%
Defect	13	37,14%	10,38%
Merge	5	14,29%	8,49%
Total	35	100%	100%

Figura 32. Tabla de tipo de problema, elaboración propia basada en los resultados obtenidos de la aplicación del nuevo proceso de desarrollo.

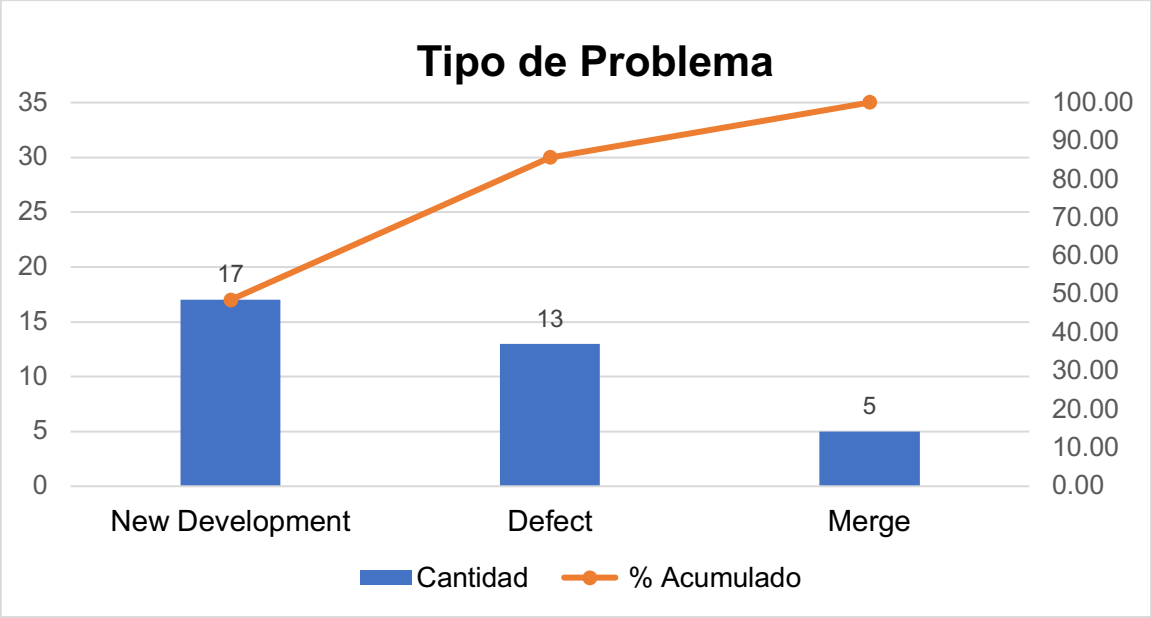

Figura 33. Histograma de tipo de problema, elaboración propia basada en los resultados obtenidos de la aplicación del nuevo proceso de desarrollo.

En este caso se observa que la mayoría de los defectos siguen siendo los introducidos por nuevos desarrollos, seguidos por aquellos provenientes de código heredado y finalmente los introducidos al fusionar el código del proyecto en cuestión con otros proyectos.

Si bien la mayoría de los defectos siguen teniendo su origen en nuevos desarrollos es importante destacar la disminución del porcentaje de este tipo de defectos que pasaron de representar un 81,13% del total a un 48,57% del total.

B. Tipo de Resolución de los defectos.

Nuevamente para considerar los defectos según su resolución solo se tendrán en cuenta aquellos introducidos por fusiones de código (**merge**) y por nuevos desarrollos.

Tipo de Resolución	Cantidad	Porcentaje	Porcentaje anterior
Funcionando como fue diseñado	11	50%	25,26%
Reparado (Código Actualizado)	7	31,82%	66,32%
No es posible reproducir el error	4	18,18%	8,42%
Total	22	100%	100%

Figura 34. Tabla de tipo de resolución, elaboración propia basada en los resultados obtenidos de la aplicación del nuevo proceso de desarrollo.

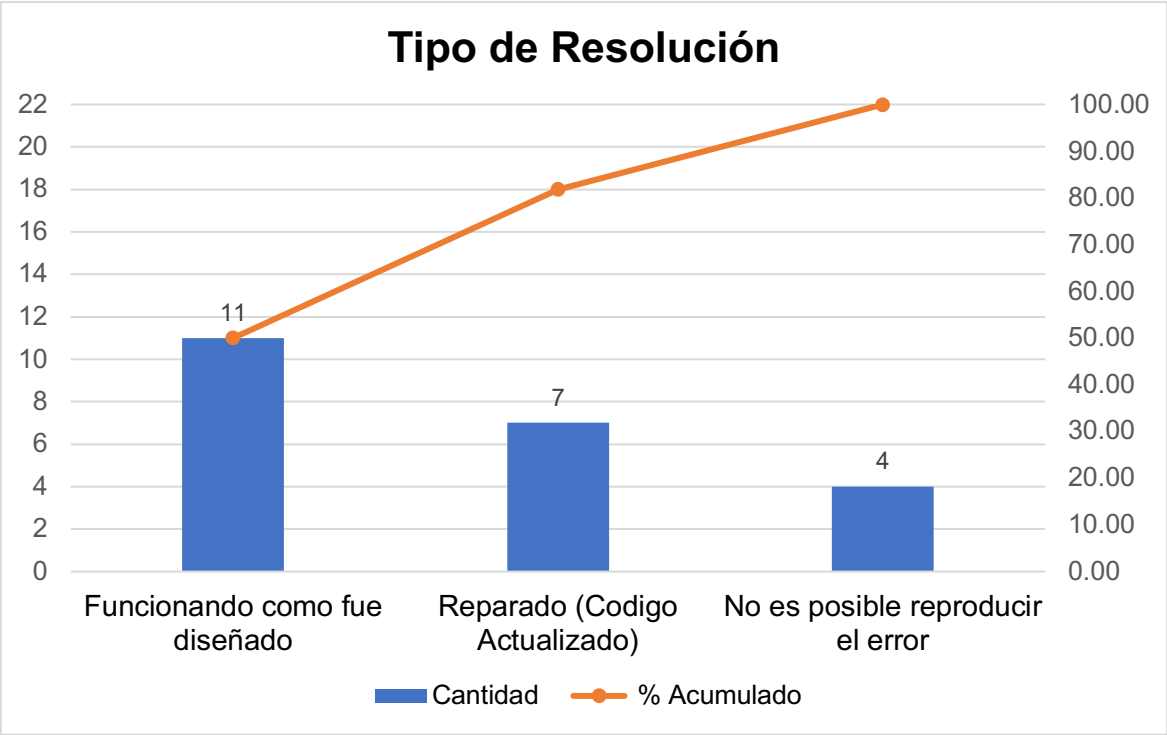

Figura 35. Histograma de tipo de resolución, elaboración propia basada en los resultados obtenidos de la aplicación del nuevo proceso de desarrollo.

Respecto a los defectos analizados según el tipo de resolución, se puede observar que aumentó el porcentaje de defectos resueltos de la manera "Funcionando como fue diseñado" y que disminuyo el porcentaje de defectos "Reparados", aumentando por otro lado, el porcentaje de errores temporales ("No es posible reproducir el error").

Nuevamente los errores temporales no son de relevancia para este análisis, sin embargo, si se puede decir que el aumento en los defectos del tipo "Funcionando como fue diseñado" indica que se ha logrado diseñar correctamente según los requerimientos del cliente y que lo entregado es lo solicitado. El problema en estos casos fue que el cliente o el dueño de producto no pudieron entender o expresar con claridad los requerimientos por lo que finalmente se solicitó una funcionalidad de manera incorrecta o bien que el equipo de Testing interpretó incorrectamente los casos de prueba creando un defecto cuando no existía uno.

Por otro lado, la disminución en los defectos "Reparados" indica que ha disminuido la cantidad de retrabajo, siendo este uno de los objetivos planteados al implementar el nuevo proceso de desarrollo.

C. Causa raíz de los defectos

Causa del defecto	Cantidad	Porcentaje	Porcentaje Anterior
Desarrollo	7	38,89%	59,77%
Requerimientos	4	22,22%	19,54%
Testing	5	27,78%	9,20%
Dependencias Externas	2	11,11%	11,49%
Total	18	100%	100%

Figura 36. Tabla de causa raíz de los defectos, elaboración propia basada en los resultados obtenidos de la aplicación del nuevo proceso de desarrollo.

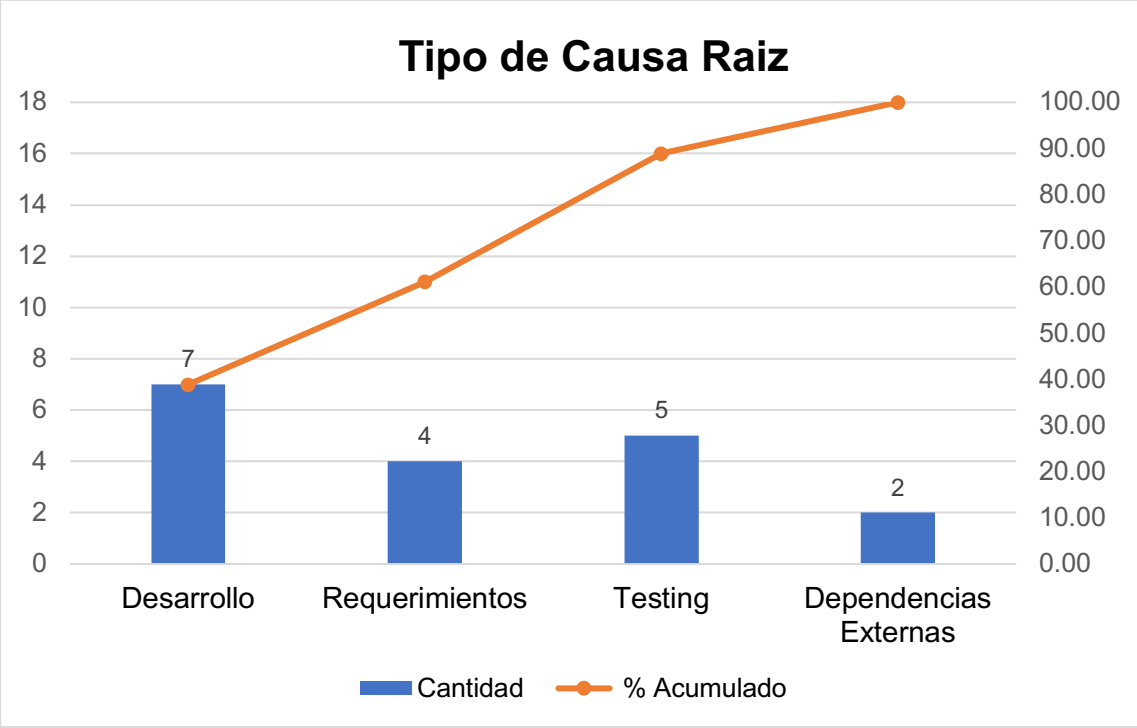

Figura 37. Histograma de causa raíz de los defectos, elaboración propia basada en los resultados obtenidos de la aplicación del nuevo proceso de desarrollo.

Analizando las distintas causas de los defectos se puede determinar que disminuyó el porcentaje originado en el "Desarrollo", esto probablemente se deba a que la

documentación ayudó guiando a los desarrolladores en su trabajo. Por otro lado, aumentó ligeramente el porcentaje de defectos de causa "Requerimientos" lo cual simplemente puede deberse a la disminución general de defectos y a la disminución de otras causas raíces. Además se observa un aumento importante en el porcentaje de defectos causados por "Testing", tal vez esto se deba a que es la parte del proceso que se vio menos involucrada en el nuevo proceso de desarrollo por estar del lado del cliente. En cuanto a los defectos causados por "Dependencias Externas" el porcentaje se mantuvo en niveles similares.

En resumen, a raíz de estos resultados se pudo determinar que el nuevo proceso de desarrollo tuvo un efecto altamente positivo, no solo en la disminución de la cantidad total de defectos (y porcentualmente la cantidad de defectos por parte del equipo de desarrollo) sino que además tuvo un impacto positivo en la sinergia del equipo completo, mejorando la dinámica de la interacción de sus distintos integrantes así como también el trabajo individual de los mismos. Se logró además una mayor satisfacción del cliente, ya que además de lo anteriormente mencionado hubo un mayor control de todo el proceso en general y se hizo mucho más fácil identificar en qué punto se producen las fallas y que es lo que hay que ajustar para resolverlas.

CAPÍTULO 9. CONCLUSIONES

Cuando inició este trabajo de investigación se tenía en claro que debían aplicarse las mejores prácticas de las diversas metodologías disponibles con el fin de mejorar el proceso de desarrollo para dar solución a los problemas que se enfrentaba en el mismo, y finalmente crear un mejor producto final.

A medida que fue avanzando el desarrollo de este proyecto de grado se fue haciendo evidente que, si bien en la teoría, la aplicación de las mejores prácticas de las distintas metodologías llevarían a los mejores resultados, en la práctica esto era irrealizable. Simplemente porque no había una única metodología ideal completamente aplicable que pudiera pueda resolver exitosamente todos los problemas.

A través del estudio e investigación de las metodologías se descubrió que mediante el "Tailoring" y la combinación de algunas de las mejores prácticas de estas, se podía desarrollar un proceso adecuado para las necesidades de este proyecto en particular.

Para la selección de las mejores prácticas se puso el foco en el producto, definiendo como se realizaba el desarrollo del mismo y de qué manera podría mejorarse este proceso, en las personas que lo desarrollaban, estudiando cada uno de los roles y como mejorar su actividades, responsabilidades y tiempos de entrega, y por último, en las limitaciones preexistentes del cliente y la compañía.

En resumen, al finalizar este trabajo se puede concluir que no hay una sola metodología que pueda ser aplicada para resolver todos los problemas que enfrenta un equipo de desarrollo, que es necesario realizar un estudio a fondo de cuáles son las causas raíces de los problemas para poder dar con la solución adecuada a los mismos, que muchas veces será necesario investigar diversas metodologías para dar con esta solución y, cuando no exista una que se adecue, se deberá hacer uso de lo aprendido para generar una solución propia que resuelva el problema puntual. Se concluye además que para generar esta solución se deberá adicionar a lo mencionado anteriormente, un conocimiento profundo del producto, de las personas y de las limitaciones que existen en el entorno, pero por sobre todo, se puede concluir que si todo lo anterior se realiza de manera adecuada, mediante la creación de un

proceso a medida pueden obtenerse no solo resultados que superen los que se obtendrían con una metodología genérica sino que además se tiene la flexibilidad de ir adaptando el proceso para alcanzar e incluso superar las propias expectativas fijadas.

REFERENCIAS BIBLIOGRÁFICAS

1. cmmiinstitute.com [Internet] CMMI® Institute LLC, 2017 [citado 13 de Noviembre de 2017] Disponible en: http://cmmiinstitute.com/capability-maturity-model-integration

2. cmmiinstitute.com [Internet] CMMI® Institute LLC, 2017 [citado 13 de Noviembre de 2017] Disponible en: http://cmmiinstitute.com/sites/default/files/resource_asset/Which_Model_2017.pdf

3. CMMI Product Team. CMMI® for Development. Versión 1.3. Estados Unidos: Carnegie Mellon; 2010.

4. Ivar Jacobson, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Primera Edición. Addison-Wesley Professional. 2000.

5. agilealliance.org [Internet] [citado 15 de Noviembre de 2017] Disponible en: https://www.agilealliance.org/agile101/

6. Paul E. McMahon. 1.2 Agile Primer. Integrating CMMI and agile development: case studies and proven techniques for faster performance improvement. Primera Edición. Estados Unidos; Addison-Wesley Professional;2010.

7. agilemanifesto.org [Internet] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thoma: Utah, 2001 [actualizado en 2001; citado en Noviembre de 2017] Disponible en: http://agilemanifesto.org/iso/es/principles.html

8. agilemanifesto.org [Internet] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thoma: Utah, 2001 [actualizado 2001; citado en Noviembre de 2017] Disponible en: http://agilemanifesto.org/iso/es/manifesto.html

9. scrumguides.org [Internet] Estados Unidos: Ken Schwaber y Jeff Sutherland; [actualizado en Julio de 2016; citado Noviembre de 2017] Disponible en: http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf

10. agilealliance.org [Internet] [citado 15 de Noviembre de 2017] Disponible en: https://www.agilealliance.org/glossary/xp

11. extremeprogramming.org [Internet] Estados Unidos: Don Wells; 1999 [actualizado el 8 de Octubre de 2013; citado en Noviembre de 2017]. Disponible en: http://www.extremeprogramming.org/

12. extremeprogramming.org [Internet] Estados Unidos: Don Wells; 1999 [actualizado el 8 de Octubre de 2013; citado en Noviembre de 2017]. Disponible en: http://www.extremeprogramming.org/values.html

13. Cynthia Andres, Kent Beck. Extreme Programming Explained: Embrace Change. Segunda Edición. Addison-Wesley Professional. Noviembre 2004

14. agilealliance.org [Internet] [citado 15 de Noviembre de 2017] Disponible en: https://www.agilealliance.org/glossary/kanban/

15. Alistair Cockburn. Crystal Clear A Human-Powered Methodology for Small Teams. Primera Edición. Addison-Wesley Professional.Octubre 2004

16. Alistair Cockburn. Agile Software Development. Addison-Wesley, Boston, MA, 2002.

17. lean.org [Internet] Estados Unidos: Lean Enterprise Institute, Inc; 2000 [actualizado en 2017; citado en Noviembre de 2017]. Disponible en: https://www.lean.org/WhatsLean/

18. Mike Sullivan , Steve Jewett , Curt Hibbs. The Art of Lean Software Development. O'Reilly Media, Inc. Enero 2009

19. Mark P. Ginsberg, Lauren H. Quinn. Process Tailoring and the Software Capability Maturity Model. [Internet] Carnegie Mellon University Pittsburgh. 1994 [actualizado en Noviembre de 1995; citado en Noviembre de 2017] Disponible en: http://repository.cmu.edu/cgi/viewcontent.cgi?article=1241&context=sei

20. wikipedia.org [Internet] Fundación Wikimedia, Inc.;[actualizado en Noviembre de 2017; citado en Enero de 2018]. Disponible en: https://es.wikipedia.org/wiki/Proceso_de_mejora_continua

ANEXO

A continuación, se presenta una plantilla del documento de diseño utilizado en el nuevo proceso de desarrollo. Este documento fue elaborado por los autores de este proyecto de grado.

[NOMBRE DEL PROYECTO]

DOCUMENTO DE DISEÑO

VERSIÓN: [VERSIÓN ACTUAL]

Historial de Revisión

Preparado por	Fecha	Versión
[Nombre]	[Fecha]	[Versión]

Revisado por	Fecha	Razón de la Revisión	Versión
[Nombre]	[Fecha]		

ÍNDICE DE CONTENIDOS

1 INTRODUCCIÓN

1.1 PROPÓSITO

El documento de diseño contiene el diseño de los componentes necesarios para brindar la funcionalidad solicitada en los "Detalles de Requerimientos". Los componentes adicionales identificados en el documento de arquitectura pueden requerir más detalles tales como interfaces, estructuras, comportamiento del componente, y la motivación para las decisiones de diseño.

Este documento debe evolucionar en conjunto con el proyecto incorporado cambios y decisiones que se lleven a cabo en el proyecto.

[Definir el rol o propósito del (de los) componente(s) relativos al proyecto y listar los requerimientos que se alcanzaran]

1.2 ALCANCE DEL DISEÑO

[Describir brevemente el alcance del componente(s) incluido en el documento. Identificar cualquier restricción, suposición, riesgo e interfaz que tenga impacto en el diseño.]

1.3 RESTRICCIONES DE DISEÑO

[Documentar las restricciones son aplicables al enfoque de diseño.]

1.4 SUPOSICIONES DE DISEÑO

[Enumerar las suposiciones que hace el equipo de diseño sobre el diseño. Todo el equipo del proyecto, tanto el equipo de desarrollo como los clientes, deben conocer estos supuestos.]

1.5 RIESGOS

[Documentar cualquier riesgo que pueda introducirse basado en el diseño del producto o el enfoque de diseño que se utiliza.]

1.6 INTERFACES EXTERNAS

[Describir las interfaces externas que provee el componente, incluyendo entradas, salidas y condiciones de error. El tipo de interface depende del nivel del componente y el tipo de Sistema. Por ejemplo, si el nivel de un componente o el tipo de sistema es un servicio web, la interfaz puede ser una hoja de estilo de Lenguaje de marcado extensible (XML). Si el nivel de un componente o el tipo de sistema es Enterprise Java Bean (EJB), la interfaz puede ser una interfaz remota.]

2 Diseño Lógico/Técnico

2.1 Referencia de caso de uso/Historia

Referencias de Historias según el Backlog

A [Identificador de Historia/Descripción]

[Descripción de la historia como se encuentra cargada en la herramienta.]

Link a la Historia

http://xxxxx.....

Modelo de Proceso de Negocio (BPM)

[Insertar archivo con el BPM.]

Análisis De Impacto

[Definir las áreas que pueden verse afectadas por este cambio. Definir consideraciones de prueba que deben tener en cuenta los testers. Considerar las áreas de impacto directo (donde se planea hacer cambios) y las áreas de impacto indirecto (que pueden verse afectadas debido a los efectos dominantes de sus cambios. Considere también el impacto en sistemas externos. Si es necesario, proporcionar detalles específicos para la creación de datos.]

Modelo Lógico

[Insertar archivo Visio con el diagrama.]

Cambios de UX (Experiencia de Usuario)

[Insertar cambios en maquetados de UX o agregar un enlace a la documentación de UX. Asegurarse de que cada cambio de UX incluya los Detalles / Validaciones de controles]

- Detalles de Controles

Nombre del Control	Tipo	Descripción / Acción	Formato	Por defecto

- Validaciones

Condición de Negocio o de datos	Acción Correspondiente	Salida o Nuevo estado

- Funcionalidad Especial
  [Agregar todos los comportamientos especiales incluidos en el cambio de UX como navegaciones, selecciones por defecto y comportamientos esperados.]

Cambios de Base de Datos

[Incluir detalles de cambios en la Base de Datos requeridos para la solución. Esto incluye tanto cambios en la estructura como cambios en los datos. En caso de nuevas tablas o columnas, es necesario incluir un Diagrama Entidad Relación(DER) o una descripción detallada de los cambios aplicados.]

Consumo de Servicios

[Documentar requerimientos para consumir el servicio tales como: certificados, endpoints para cada ambiente, códigos de error, ejemplos de solicitudes/respuestas para escenarios de éxito y falla, documentación del proveedor. Identificar además cuales son las pantallas afectadas por cada servicio.]

Especificaciones Técnicas

[Incluir todas las especificaciones relacionadas con aspectos existentes de la aplicación que vayan a ser modificados o agregados: clases, métodos, constantes, etc.]

[Repetir la sección numerada con letras del abecedario una vez por cada Historia que se trabaje en el proyecto]:

B [IDENTIFICADOR DE HISTORIA/DESCRIPCIÓN]

3 GLOSARIO TÉCNICO

3.1 GLOSARIO DEL DOCUMENTO DE DISEÑO

Término	Definición