FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Programa de Apoyo al Egreso de Profesionales en Actividad

TÍTULO: Modelo de Proceso para trabajo a gran escala – Caso de Aplicación

AUTOR: Saullo, Lisa

DIRECTOR ACADÉMICO: Mg. Esponda, Silvia

DIRECTOR PROFESIONAL: Lic. Arriola, Alejandra

CARRERA: Licenciatura en Sistemas

Resumen

En el trabajo se describen las metodologías Scrum, RUP, Kanban, y XP. Se definen características de los proyectos a gran escala, se presenta el proyecto y se explica como esta formado. Se expone el caso de aplicación, en base a la experiencia de trabajo en el proyecto, como pueden convivir distintas metodologías adaptándolas según la necesidad.
Se debe tener en cuenta que los proyectos requieren de distintas estrategias de implementación que respondan a sus objetivos, es por eso que no siempre una única metodología es la solución en todos los casos. Se puede combinar características de diferentes metodologías para lograr las metas deseadas.

Palabras Clave

Metodología ágil

Scrum

Rup

Kanban

XP

Requerimiento

Proyecto

Conclusiones

Dado el estudio realizado se pudo demostrar que no hay una única forma de llevar adelante un proyecto.

Es posible utilizar distintas metodologías, adaptando cada una según la necesidad, pudiendo convivir y obtener un buen resultado.

Trabajos Realizados

Se analizaron distintas Metodologías Agiles (SCRUM, KANBAN, XP), mostrando sus fortalezas, ventajas y desventajas de cada una.

Se presentó el nuevo modelo metodológico definido, con un caso de aplicación de manera de evidenciar la convivencia de las distintas metodologías

Trabajos Futuros

Planteo como posible trabajo futuro el análisis y aplicación de la metodología Less (Scrum a gran escala) en grandes proyectos.

Fecha de la presentación Agosto 2020

Modelo de proceso para trabajo a gran escala- Caso de Aplicación

Tesina de Licenciatura
en el marco del PAEPA

Alumno

Saullo, Lisa

Directora

Mg. Esponda, Silvia

Agosto 2020

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS Y FIGURAS

CAPÍTULO 1: Objetivo, Motivación y Desarrollo Propuesto

1.1 Introducción

Hoy en día existen diversas alternativas a la hora de elegir una metodología para desarrollar software. Años atrás imperaban las metodologías rígidas, también conocidas como las metodologías tradicionales, en las cuales preponderan la documentación, los modelados, las actividades, actores y/o roles. A partir del 2001, un punto de inflexión surge con el nacimiento de metodologías conocidas como *Metodologías Ágiles* para el desarrollo de software, creadas para ser aplicadas en proyectos pequeños/medianos y con requerimientos volátiles o que cambian con frecuencia; a diferencia de las metodologías tradicionales, que suelen ser más apropiadas para grandes proyectos y donde sus requerimientos son más resistentes a los cambios.

Tomando como base estos conceptos, surge el objetivo de esta tesis, ya que si bien las metodologías surgieron por los motivos expuestos, existen casos donde la aplicación de dichas metodologías, de manera pura, es compleja, y se opta por tomar las buenas prácticas de una u otra metodología, y decidir acerca de los métodos a utilizar.

A partir de una experiencia en un proyecto se intentará explicar cómo pueden convivir distintas metodologías tomando lo necesario de cada una y adaptándola al proyecto en cuestión.

1.2 Objetivos:

- Estudiar, analizar y comparar las metodologías: Rational Unified Process (RUP), SCRUM, KANBAN y eXtreme Programming (XP)
- Analizar ventajas y desventajas de su uso, en distintos proyectos, en

particular en aquellos de gran envergadura.

- Tomar los puntos que pueden ser aplicados al proyecto de los modelos anteriormente expuestos en un proyecto de gran escala.
- Presentar un caso de aplicación de la metodología diseñada para proyectos a gran escala a partir de adaptación diferentes metodologías ágiles.
- Realizar el análisis de los resultados obtenidos. Plantear nuevas líneas de investigación.

1.3 Motivación

EL Esidif es un proyecto de administración financiera, que consta de varios módulos y aplicaciones satélites.

Es utilizado por todos los organismos nacionales, entes autárquicos e instituciones de algunas provincias.

Dada la amplitud del proyecto, la complejidad de la temática y la cantidad de gente que trabaja en él, se comenzó a investigar distintas metodologías ágiles para mejorar los procesos de desarrollo, análisis y organización de las etapas.

Con la incorporación de estas metodologías, incorporación de herramientas y mejoras que se fueron incorporando, se concluye en un modelo metodológico utilizado el día de hoy.

Este nuevo modelo metodológico logró reducir los tiempos de entrega, mejorar la comunicación entre los equipos, y desarrollar de manera más eficiente.

De esta manera se logró corroborar que la combinación de metodologías mediante un proceso de análisis y depuración da como resultado una nueva metodología compuesta aplicada a un proyecto de alta envergadura con resultados óptimos.

1.4 Desarrollo Efectuado

El proyecto, desde sus comienzos sufrió una serie de transformaciones, basadas en la renovación tecnológica.

Inicialmente el proyecto se implementó con metodología RUP, basada en iteraciones, donde al terminar cada una de ellas se hace una entrega (etapa de transición) [9].

Estas iteraciones se hacían cada 6 meses. Al principio del proyecto esto no representaba mucho tiempo ya que era un proyecto nuevo. Pero luego, al ir realizando entregas, el cliente empezó a visibilizar la funcionalidad y a solicitar entregas en menor tiempo.

Para poder cumplir, se comenzó a evaluar cambios en el desarrollo para acelerar los tiempos.

Primero, por las dimensiones del proyecto, se modificó la forma de trabajo incorporando lo que se dio a llamar como buenas prácticas:

- Plantear objetivos y prioridades,
- Definir las herramientas que se van a utilizar.
- Estimar los costos, definir el alcance,
- Identificar habilidades del grupo que lo llevará a cabo y
- Organizar los recursos.

La incorporación de estas prácticas fue un proceso de crecimiento continuo que llevó a la organización a ahondar en el estudio de nuevas metodologías para lograr la madurez del proyecto

Dada la amplitud del proyecto, la complejidad de la temática y la cantidad de gente que trabajaba en él, se analizó la forma de trabajo de manera que se agilice el desarrollo, análisis y organización de las etapas.

Por este motivo se enfocó el estudio en las metodologías ágiles, que se centran en el equipo de desarrollo y sus interacciones, y la implementación del software por encima de su documentación.

El estudio comenzó en la metodología ágil más aplicada en la actualidad, como es Scrum, para iniciar la transformación del proyecto, entendiendo que es más adaptable.

Esta permite controlar la evolución del proyecto de forma incremental (por cada sprint hay un nuevo desarrollo), útil en un proyecto donde los requisitos son inestables. [2]

Con la incorporación de dicha MA, se modificó nuevamente la forma de trabajo:

- Se comenzaron a hacer entregas más frecuentes.
- Se planificaron sprints quincenales, lo que mejoraba notablemente la comunicación, realizando reuniones diarias de los equipos de trabajo (análisis, desarrollo y testing), reuniones semanales y quincenales de los niveles superiores para hacer las planificaciones siguientes y organizar las tareas de los equipos.

La participación del equipo en la toma de las decisiones fue más significativa, como por ejemplo, en la estimación de las tareas a desarrollar durante el sprint. [10]

- Se logró agilizar los despliegues, acortar los tiempos de implementación, una mayor productividad, dando así mejor visibilidad al usuario respecto de los avances e incorporaciones de funcionalidad.

- La metodología Rup se siguió utilizando para la parte de análisis y modelado de los requerimientos

Sin embargo, a pesar de todas las mejoras, aún había problemas sin resolver, como por ejemplo la necesidad de plasmar las planificaciones de corto y largo plazo en un artefacto. El equipo lo resolvió con la incorporación de un tablero, donde cada ítem a realizar o tarea planificada se escribía en una etiqueta y se agregaba al tablero.

De este y otros problemas, surgió la idea de incorporar el método Kanban [1] que se basa en la premisa de visualizar lo que se está haciendo, lo que se está terminando y lo que hay que hacer a continuación.
Otro problema a resolver eran los desarrollos urgentes, si bien no son frecuentes, era un problema a resolver.
Por eso surge la metodología XP, que permite que se enfoquen en el desarrollo extremo y poder cumplir con los tiempos solicitados por el usuario.
Como resultado, se concluyó en un modelo metodológico donde conviven distintas metodologías.
Este nuevo modelo metodológico logró reducir los tiempos de entrega, mejorar la comunicación entre los equipos y desarrollar de una manera más eficiente.

Capítulo 2: Proyectos a gran escala

2.1 Introducción

En este capítulo se presentan las características de los proyectos a gran escala, y aspectos a tener en cuenta a la hora de emprender uno de esta envergadura.

Un proyecto se puede definir como un *proceso con un conjunto de actividades relacionadas utilizando recursos disponibles para conseguir un objetivo en un plazo determinado, con un comienzo y un fin*. [16]

Cada proyecto nace de una necesidad y está orientado a la obtención de un resultado dentro de un tiempo limitado y con objetivos que determinen el alcance y los recursos a utilizar.

Cada proyecto es único, no es algo rutinario sino un conjunto de actividades y operaciones específicas con objetivos y medición de resultados.

Los proyectos grandes y complejos requieren de un proceso de elaboración con muchos elementos relacionados y un dedicado proceso de construcción.

Requiere de una evaluación de la complejidad en cuanto a tiempos, costos, comunicaciones, recursos y riesgos. Reducir el tamaño y la complejidad no es solo dividir el proyecto en procesos más pequeños y manejables sino también realizar un análisis de todas las áreas involucradas para poder tener una dimensión de tamaño y tareas así poder reducir luego haciendo uso de prácticas de gestión en cuanto a planificación y control del proyecto con áreas de trabajo.

2.2 Características

Todos los proyectos, sin importar de qué área sean, tienen en común un conjunto de características:

- Tienen un propósito

- Cuentan con un conjunto de actividades y objetivos
- Se ajustan a un tiempo determinado
- Cuentan, al menos, con una fase de planificación, ejecución y entrega
- Están orientados a resultados
- Utilizan recursos con distintos roles y responsabilidades
- Cada proyecto es diferente

Para proyectos de gran tamaño se debe considerar una serie de factores como ser: el *alto tráfico, lógica compleja*, muchas *interfaces de usuario*, gran *cantidad de datos*.

El *alto tráfico* se da por la gran cantidad de usuarios conectados en un mismo momento usando el sistema, y la operatoria que realizan.

La *lógica compleja* se da en proyectos con mucha operatoria y funcionalidad. Se generan gran cantidad de interfaces o ventanas donde el usuario opera interactuando con el sistema

2.3 Organización del proyecto

Cada proyecto, sea grande o pequeño, se forma con una estructura de vida:

Inicio, organización, ejecución y cierre.

En el inicio se define y se plantea el alcance y el equipo. El alcance define el tamaño del proyecto, el tiempo en que se va a desarrollar y la cantidad de recursos.

El equipo de trabajo debe concordar en tamaño y estructura con el tamaño del proyecto, las características y los objetivos de planificación, por ende un proyecto a gran escala es realizado por un equipo de trabajo con dimensiones acordes al mismo.

En los *equipos grandes* se debe prestar mucha atención a la comunicación y coordinación. A medida que crece en número se deberá tener más canales de comunicación.

Los proyectos con más de 50 personas no se organizan de manera que todos se comunican entre sí, se requiere de métodos de organización que simplifiquen las comunicaciones. Esto es un punto importante para tener éxito en proyectos a gran escala.

Una manera de organizarse es formar grupos más pequeños con responsables que interactúen entre sí y con los directivos del proyecto.

El equipo debe tener claro qué tareas realizar, cuándo realizarlas y que deben entregar.

La planificación, según la guía PMI, específica que se debe desarrollar un plan para dirigir el proyecto, relevar los requerimientos, definir el alcance, subdividir el proyecto en partes o componentes más pequeñas, identificar las actividades y sus relaciones, estimar los recursos (personas, equipos etc.), estimar costos y armar un presupuesto, etc.(Figura 2)

La etapa de planificación puede ser compleja si se tiene en cuenta el tamaño del proyecto, primero se deberá realizar una planificación estratégica, donde se considere la misión, el ambiente y los objetivos, luego realizar un análisis de requerimientos y la asignación de recursos. Una vez hecho esto puede desarrollarse el plan del proyecto.

El cierre del proyecto contempla los resultados alcanzados, cierre de las actividades y redacción del informe final.

Para la planificación de un proyecto grande y/o complejo, donde se debe ejecutar un número considerable de actividades, lo aconsejable es subdividir en partes más pequeñas (Figura 1). Primero separar las tareas más importantes y más grandes y luego dividirlas en tareas más pequeñas. Este proceso de subdivisión se conoce como ESP (Estructura de Subdivisión del Proyecto), se comienza en el nivel más alto con los elementos principales y se divide y subdivide hasta llegar a un nivel de detalle que sea manejable. Se obtienen paquetes de trabajo que permiten una administración y control.

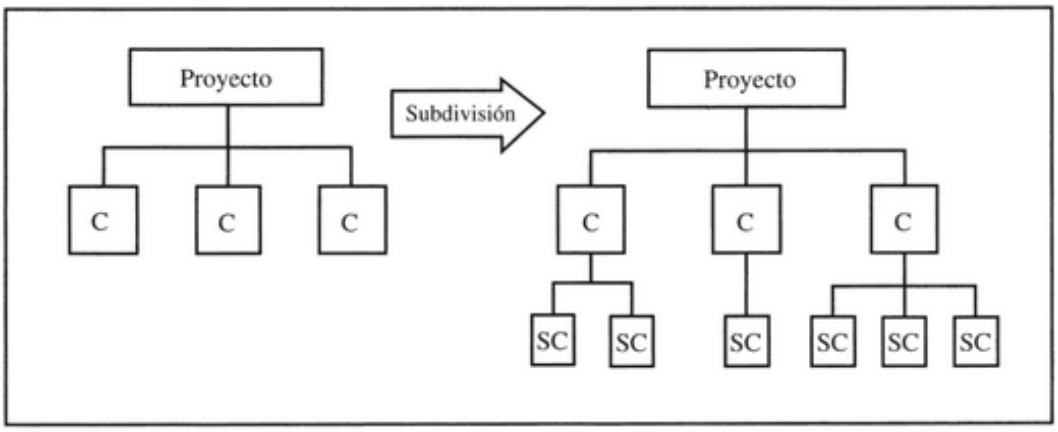

Figura 1 *Esquema de subdivisión del proyecto*

Los paquetes más pequeños son tareas que facilitan la planificación y control a lo largo del desarrollo del proyecto.

La ESP puede definirse según las necesidades del proyecto por área, por disciplina, nivel sistema, actividad, proceso etc. Sea cual sea la subdivisión del proyecto, se debe considerar que la tarea debe poder ser ejecutada por una persona o un pequeño grupo de trabajo. Esto ayuda a alcanzar los objetivos, coordinar los recursos y realizar una correcta dirección del proyecto.

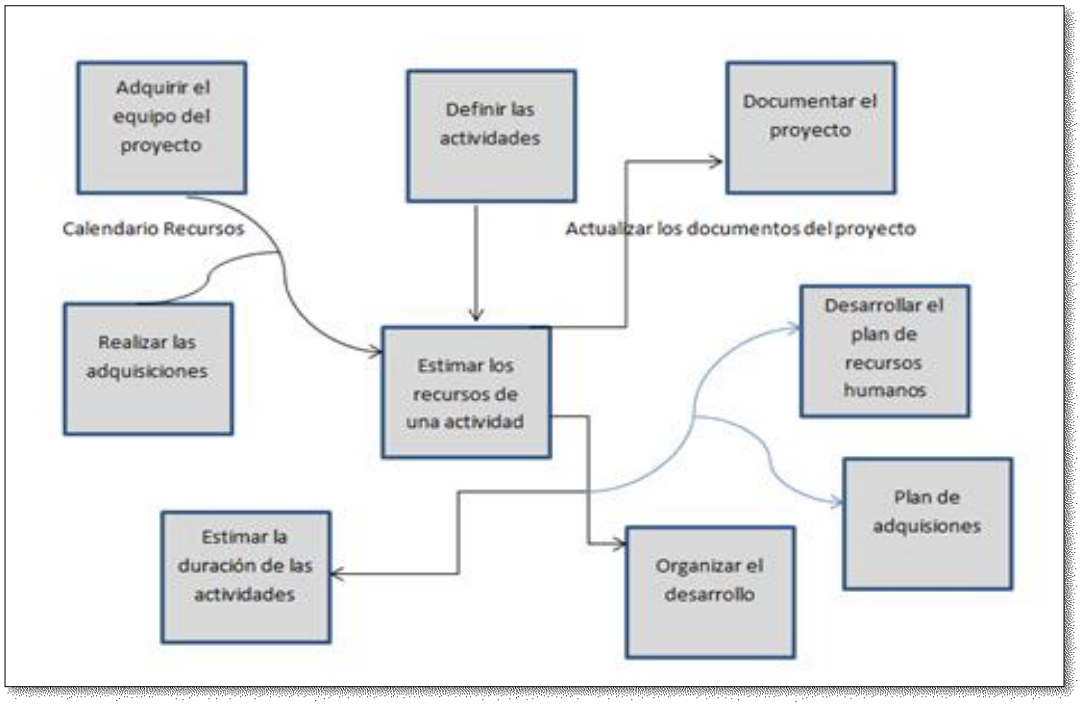

Figura 2 *Diagrama de estimación de recursos*

2.4 Conclusión

Abordar un proyecto complejo representa un gran reto y viene acompañado de muchas responsabilidades y metas a cumplir.

El resultado va a depender de la realización de las etapas con la colaboración del equipo y recursos materiales necesarios en cada momento. Las fases que se ejecuten, el ordenamiento, la estimación de recursos y costos a utilizar en cada etapa precisan de conocimiento y experiencia que permita prever y superar las dificultades que puedan aparecer durante su desarrollo.

Capítulo 3: Metodologías Agiles

3.1 Introducción

En el año 2001 después de una reunión en EEUU nace el término ágil. Se reúnen un grupo de personas, cansadas de los modos tradicionales de desarrollo de software. [4] [5] [12]

Su objetivo era tratar de permitir a los equipos, desarrollar software rápidamente y responder a los cambios que surjan a lo largo del proyecto.

El resultado fue un documento conocido como el *Manifiesto ágil*, que tiene 4 características comunes y 12 principios. [7]

Tras esta reunión se creó *The Agile Alliance* [3], una organización dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que los adopten.

La flexibilidad, la calidad y la necesidad de entregar proyectos y productos en cortos espacios de tiempo son una prioridad. Todo ello ha facilitado el nacimiento del manifiesto ágil, un documento que resume la filosofía ágil.

Según el manifiesto existen 4 características valorables:

1. Las interacciones entre las personas del equipo son *más* importantes que los procesos o las herramientas.
2. Implementar software que funcione es *más* importante que documentar en forma extensiva. Centrarse en lo necesario y de forma concreta.
3. La colaboración con el cliente es *más* importante que el contrato. Tener una comunicación diaria con el cliente hará que el proyecto avance de forma correcta.
4. El proceso debería responder ante el cambio, *en lugar* de seguir un plan. Poder responder ante los cambios de forma flexible.

Estas características inspiraron los 12 principios del manifiesto. Los primeros son generales y el resto tiene que ver más con los pasos a seguir, el equipo y la organización.

El desarrollo ágil de software posee 12 principios:

1. La prioridad es satisfacer al cliente mediante la entrega continua de software funcionando aunque sea rudimentario.
2. Siempre se acepta que los requisitos cambien, sin importar qué momento del proyecto sea. Esto le permite al cliente tener una mayor satisfacción.
3. Se entrega software funcional en pequeños periodos de tiempo
4. Tanto los desarrolladores como los responsables del negocio deben trabajar juntos diariamente durante todo el proyecto.
5. La información se transmite más fácilmente cara a cara dentro del equipo.
6. Hay que motivar a la gente mediante la creación de un ambiente de confianza y empoderamiento para poder llegar a finalizar el trabajo.
7. El software funcionando es la visibilidad principal de progreso.
8. El proceso ágil promueve un desarrollo sostenible. Los desarrolladores deben mantener un ritmo a lo largo del proyecto.
9. La atención continua en la calidad técnica y el buen diseño mejora la agilidad.
10. La simplicidad es una parte vital de una buena gestión ágil. Si el código es sencillo es más fácil adaptarlo a los cambios que vayan surgiendo.
11. Los equipos organizados por si mismos producen las mejores arquitecturas, requisitos y diseños.
12. Los equipos deben hacer revisión cada cierto intervalo de tiempo para poder ser más efectivos. Como el entorno cambia continuamente, estos deben ajustarse al nuevo escenario.

Las metodologías ágiles introducen nuevas prácticas al conjunto de desarrollo. Estas incluyen:

Product backlog, programación de a pares, cliente en el lugar, integración continua, refactorización, entre otras.

Beneficios de adoptar una metodología ágil:

La metodología ágil logra un compromiso entre el equipo y el cliente. Ya que éste participa activamente en el proyecto, generando una colaboración de ambas

partes. Esto ayuda a los desarrolladores a entender mejor la visión del cliente.[11]

Las entregas frecuentes (mediante iteraciones) permiten una visualización al cliente de los avances y entender mejor los costos estimados. Esto logra, a la hora de la toma de decisiones, mejorar la priorización de los puntos de cada entrega.

Si el objetivo es entregar un subconjunto acordado de funciones del producto, y surge algo inesperado o un cambio urgente, los procesos ágiles permiten cambiar las prioridades y refinar los requerimientos. Los cambios que quedaron pendientes se agregan a la siguiente iteración, de manera que pueden ser visibles en unas pocas semanas.

Cada iteración permite centrarse en partes del proyecto más manejables. Realizar pruebas, revisiones a lo largo de la iteración, permite encontrar errores y defectos y realizar una pronta solución, mejorando así la calidad del producto.

Esta forma de trabajo presenta al equipo metas a corto plazo, generando una mejor productividad al visualizar ellos mismos los avances, desafiándose a ser más rápidos y eficientes.

Hay diferentes métodos ágiles que fomentan los valores y principios del manifiesto. Scrum y XP son los más conocidos.

3.2 Scrum

Scrum nace a principios de los 90 y fue desarrollada por Ken Schwaber y Jeff Sutherland.

Es una metodología para la gestión de proyectos. No se basa en seguir un plan sino en adaptarse continuamente a la evolución del proyecto. [2] [3]

Schwaber afirma que el desarrollo de sistemas no es un proceso definido, sino que es empírico. Esto significa que no puede repetirse consistentemente, requieren monitoreo y adaptación constante. [11] [13]

Scrum está diseñado para adaptarse a los cambios de los requerimientos. Éstos se revisan y se ajustan durante el proyecto en intervalos cortos. Esto permite adaptarse en tiempo real y construir un proyecto que satisface al cliente resolviendo sus necesidades y aumentando su satisfacción.

Scrum está basado en un proceso constructivo iterativo e incremental. Sus características se resumen en dos principales: las iteraciones (es un ciclo corto repetitivo) denominadas sprints con una duración de dos a cuatro semanas y las reuniones diarias *Dailys* de corta duración.

Al final de cada sprint se obtiene una pieza de software ejecutable con un incremento de la funcionalidad.

Scrum se basa en los principios de inspección continua, adaptación, autogestión e innovación. El cliente se entusiasma con el proyecto al verlo crecer en cada iteración y el equipo de desarrollo ve un ambiente propicio para desarrollar sus capacidades profesionales y esto da una motivación a los integrantes.

3.2.1 Flujo de Scrum

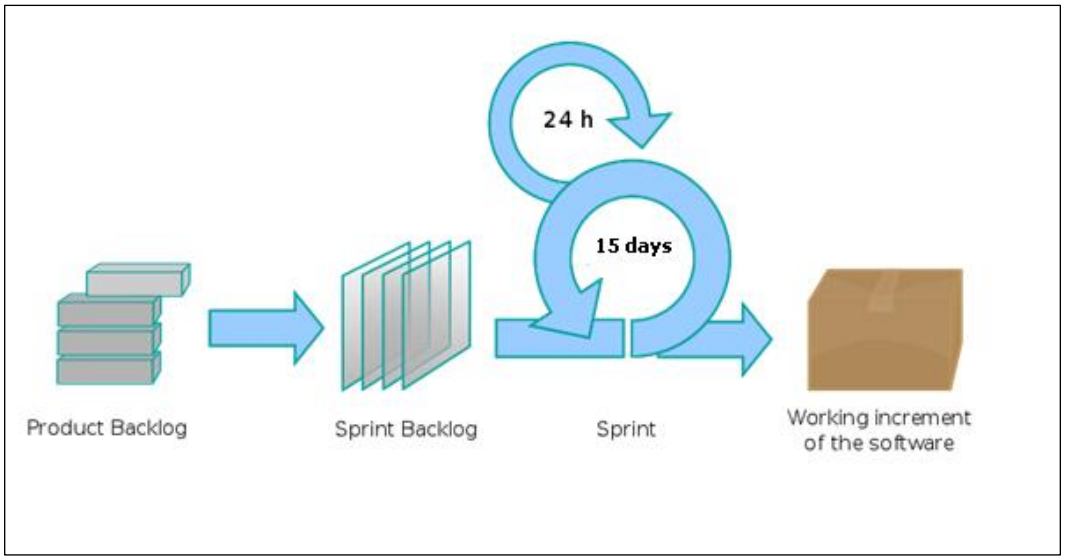

Figura 3 ***Flujo de trabajo de la metodología Scrum, desde el Product Backlog hasta el resultado***

El proceso inicia en la lista de requisitos, aquí el Product Owner prioriza los objetivos y se reparten en iteraciones (sprint) y entregas.

La iteración, normalmente de 2 semanas, concluye en un incremento del producto de software. (Figura 3)

El primer día de la iteración se realiza la planificación donde se seleccionan los requisitos y el equipo pregunta al cliente las dudas que puedan surgir y se

seleccionan los requisitos prioritarios. Luego con la lista finalizada, el equipo se organiza para resolver las tareas.

Cada día el equipo se reúne para revisar las tareas (15 minutos), donde cada miembro expone lo que está haciendo y lo que va a hacer para llegar al objetivo.

3.2.2 Roles de Scrum

Todas las responsabilidades del proyecto recaen en los roles:

- Product Owner
- Scrum Master
- Team
- Cliente

Product Owner: (El dueño del Producto) El Product Owner se asegura que el equipo trabaje en forma adecuada. Ayuda al usuario a elaborar los requisitos y agregarlos al Product Backlog con sus prioridades.

El dueño del producto es el nexo entre el equipo de desarrollo y los clientes, por lo tanto este rol requiere de buenas habilidades de comunicación.

Scrum Master (Lider de Proyecto) Su trabajo principal es eliminar los obstáculos que impidan que el equipo alcance los objetivos del sprint. Interactúa con el cliente y el equipo. Es responsable que el proyecto se lleve a cabo, y se cumplan las metas pautadas.

Team (Equipo): Tienen la responsabilidad de la entrega del producto al finalizar el sprint. Son autónomos y organizados. El equipo se forma entre 5 y 8 personas.

Cliente: El cliente es quien proporciona los requerimientos al Product Owner.

3.2.3 Elementos que conforman el desarrollo de Scrum

Product Backlog

El Product Backlog es el listado de tareas que engloba todo el proyecto. Debe incluir toda funcionalidad o tareas a realizar en el proyecto, en un tiempo estimado por el equipo de desarrollo.

La responsabilidad exclusiva de ordenar el Product Backlog es del Product Owner, que se encuentra en constante comunicación con el cliente para asegurarse que las prioridades están bien establecidas.

Sprint Backlog

El Sprint Backlog es una lista de tareas, extraídas del Product Backlog, que el equipo desarrollará durante el sprint. Esta lista se define y estima en la reunión de planificación del sprint, antes de comenzar la iteración. Sirve para visualizar el trabajo a realizar en cada sprint. El Sprint Backlog permite analizar qué objetivos se cumplieron y cuáles se podrían eliminar.

Incremento

Se llama incremento al resultado de un sprint. Es la sumatoria de tareas, casos de uso, correcciones y cualquier otro elemento que se haya desarrollado en la duración del sprint y que se pondrá a disposición del usuario final.

Construir el sistema de manera ágil mediante iteraciones y de forma incremental asegura que todas las etapas del ciclo de vida de software ocurren en las semanas que dura el sprint.

No es posible construir todo el sistema en la iteración pero sí se puede ir efectuando entregas donde se visualice el incremento.

3.2.4 El proceso de *Scrum*

Sprint

El sprint es una iteración de una cantidad variable de días (pudiendo variar de una semana a un mes) donde el equipo desarrolla las tareas elegidas en el sprint Planning Meeting (reunión de planeamiento del sprint) Esto se transformará en un incremento de funcionalidad al sistema. Es el núcleo de Scrum que divide el desarrollo de un proyecto en pequeñas partes.

Durante el sprint no se puede modificar la lista de tareas del Sprint Backlog. Solo es posible cambiar el sprint si por alguna circunstancia cambiaron las prioridades del negocio. En este caso el Product Owner puede terminar el sprint, lo cual es costoso, para todo el equipo y el producto, por lo que el Product Owner lo hará salvo circunstancias extremas.

Sprint Planning Meeting

Antes que comience la reunión de planificación del sprint el Product Owner ya tiene que haber seleccionado de la lista Product Backlog las tareas prioritarias para el sprint.

El Product Owner y el Team revisan el proceso hasta que finalizan la lista acorde a los recursos disponibles para el sprint, quedando definido el Sprint Backlog.

Es conveniente establecer el objetivo del sprint, un propósito del negocio que ayude al equipo a mantener el foco de porqué se están realizando las tareas elegidas. Esto además ayuda al equipo a adaptarse a situaciones que pudieran surgir en el transcurso del sprint.

Scrum Diario

- Diariamente el equipo se reúne para una reunión llamada Daily Scrum Meeting.
- La duración aproximada es de 15 minutos.
- Se realiza todos los días a la misma hora.
- Participa el equipo y el Scrum Master.
- Estas reuniones se usan para detectar si surgen impedimentos y poder solucionarlos de manera de poder seguir con las tareas acordadas.
- Cada participante comenta lo que hizo y lo que hará y si tuvo algún obstáculo en el camino durante su trabajo. De esta forma todos conocen el estado del proyecto.

Revisión del Sprint

Al finalizar el sprint se realiza una reunión de revisión del sprint (Sprint Review Meeting)

En esta reunión se le muestra al cliente, al Product Owner y los demás involucrados en el proyecto el incremento alcanzado durante el sprint.

El Team es quien muestra los avances realizados y el cliente nos da su feedback de los cambios involucrados.

Aquí se define la fecha de la próxima reunión.

Retrospectiva

Las reuniones de retrospección (Sprint Review Meeting) se realizan para plantear las dificultades durante el sprint y las mejoras para el próximo. En este ámbito el equipo puede consensuar las mejoras y los cambios, además los miembros pueden exponer lo bueno y que cosas quisieran mejorar para el próximo sprint.

Después de la revisión del sprint el Product Owner puede actualizar la lista Product Backlog, reordenando, eliminando o actualizando con nuevas ideas.

En ese momento se puede comenzar otro sprint. No hay brechas entre un sprint y otro. Esta es la clave de Scrum, sprint cortos, entregas frecuentes, visibiliza el incremento continuo del sistema y la interacción entre clientes y el equipo.

3.3 RUP

El Proceso Unificado Racional (Rational Unified Process, RUP) es un proceso de desarrollo de software que trabaja en conjunción con UML, y juntos constituyen la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. [8][9]

RUP sigue el modelo incremental, trabaja con fases y flujo de trabajo. Cada fase se puede realizar en forma iterativa realizando un incremento y todo el trabajo que componen las fases puede implementarse de manera incremental. [6][7]

RUP plantea tres formas de pensar el modelo: estático, dinámico y en buenas prácticas de uso.

.

3.3.1 Modelo Dinámico

La estructura dinámica de RUP es la que permite que este sea un proceso principalmente iterativo.

Ciclo de vida – Fases e iteraciones

RUP se repite en ciclos, cada ciclo culmina en una versión del producto. Cada versión es un producto entregable, que incluye los requisitos, casos de uso, casos de prueba, componentes de arquitectura y modelo visual modelados con artefactos de UML.

Cada ciclo se compone de 4 fases. Cada fase se subdivide en iteraciones.

Fases *:*

Las 4 fases son:

1. *Inicio,*
2. *Elaboración,*
3. *Construcción*
4. *Transición*

Fase de Inicio

En esta fase se define el alcance del proyecto con los clientes, se identifican los riesgos asociados al proyecto, se elabora el plan, y se detalla de manera general la arquitectura del software. Se concreta la idea y la visión del producto.

Fase de Elaboración

En esta fase se completa el modelo de requerimientos, se especifican los casos de uso, los actores y se diseña la arquitectura del sistema. Se realiza el primer análisis del dominio del problema y se diseña la solución preliminar.

Fase de Construcción

En esta fase es donde se crea el producto. El producto crece hasta convertirse en un software preparado para ser entregado a los usuarios. Al final de esta fase el producto contiene toda documentación del sistema.

Es posible que no esté libre de defectos, muchos de estos se descubrirán y solucionarán en la fase de transición.

Fase de Transición

Esta fase sucede cuando el producto está lo suficientemente maduro. EL objetivo es llegar a una versión entregable.

En esta etapa se realizan las pruebas y se informan defectos y errores. El grupo de desarrollo corrige los problemas e incorpora algunas de las mejoras sugeridas.

Los errores los dividen en los que tienen impacto para el funcionamiento y los que pueden corregirse en la siguiente entrega.

Al final de esta fase se debe llegar a un producto de software que funcione correctamente.

3.3.2 Modelo estático

El modelo estático se enfoca en términos de artefactos, trabajadores y flujos de trabajo. *RUP* define 4 elementos:

- *Roles*: “Quien” hace
- *Actividades*: “Como” hace
- *Artefactos*: “Que” hace

- *Flujos*: “Cuando” hace

***Rol*:** El rol determina el comportamiento y las responsabilidades de una persona o un equipo de personas que trabajan como un equipo. Una persona puede tener más de un rol y un rol puede ejecutarse por más de una persona. El rol tiene como responsabilidad llevar a cabo un conjunto de actividades para producir un artefacto.

***Actividades*:** Una actividad realiza la actualización o creación de un artefacto. La realiza un determinado rol. Un conjunto de actividades convierten los requisitos del usuario en un conjunto de artefactos que conforman un producto o parte de él.

***Artefactos*:** Un artefacto es una porción de información que es utilizada, modifica o producida por el proceso para obtener el producto final.

Un artefacto puede ser:

- Un documento, puede ser el que realiza arquitectura. Por ej: diagrama de secuencia o diagrama de estados. (Este último se usa mucho para saber las transiciones de un comprobante desde que se inicia hasta que termina su circuito)

- Un modelo, como casos de uso o modelo de diseño. Es una abstracción del sistema que construyen los arquitectos o desarrolladores. Los que modelan los requisitos funcionales no se centran en cómo es el sistema por dentro, sino que lo piensan en que necesitan los usuarios. Esto lo plasman en los casos de uso y sus actores. Los actores pueden ser personas o subsistemas externos que interactúan con el sistema.

- *Flujo de trabajo.* Un flujo de trabajo es una relación de actividades que producen resultados y muestran la interacción entre los roles. Los roles, actividades y artefactos por si mismos no definen el proceso.

Figura 4 *Relación entre roles, actividades y artefactos*

Hay nueve procesos en el flujo de trabajo de RUP (Figura 5):

Flujo de Trabajo*:*

- Modelado de negocio: se modelan los procesos que utiliza el negocio
- Requerimientos: se relevan los requerimientos, se identifican actores y se genera la documentación del requerimiento.
- Análisis y diseño: se analiza el requerimiento y se generan los casos de uso, diagrama de secuencia, de estados y se hace un modelo de diseño.
- Implementación: es el desarrollo del sistema.
- Despliegue: se genera la versión del sistema y se instala en el cliente.
- Prueba: las pruebas se realizan y se trabaja en conjunto con desarrollo.

Flujo de Soporte:

- Entorno: herramientas necesarias para el sistema.
- Administración de cambios: gestiona los cambios del sistema.

- Gestión de Proyecto: gestiona el desarrollo del sistema.

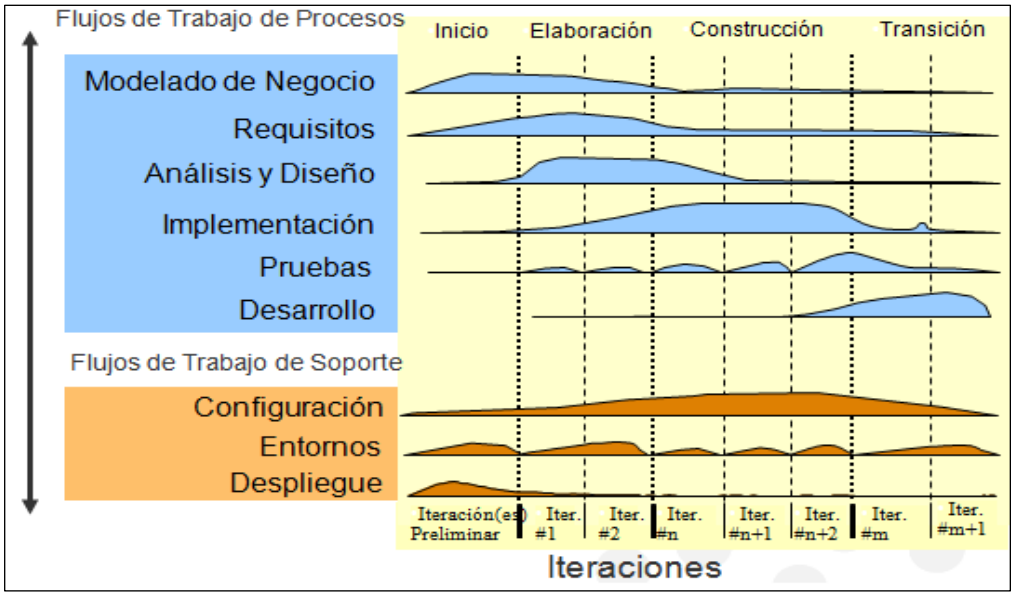

Figura 5 *Fases y flujo de trabajo en RUP*

3.3.3 Buenas Prácticas

RUP recomienda 6 buenas prácticas para el desarrollo de un sistema:

- Basado en iteraciones: Se tiene que priorizar el desarrollo de requisitos de mayor prioridad generando un incremento en el sistema. Al final de cada iteración se cuenta con una versión ejecutable. Esto aumenta la respuesta al usuario.

- Orientado a componentes: La arquitectura basada en componentes permite crear un sistema fácilmente extensible y donde se fomenta la reutilización del software y permiten construir más rápidamente aplicaciones de calidad. Los sistemas desarrollados con componentes son más resistentes al cambio y pueden reducir el mantenimiento del mismo.

- Verificar la calidad: El control de calidad debe realizarse en todo el proceso y no al final de cada iteración. Asegurar la calidad del sistema forma parte del proceso de desarrollo, de todos los miembros del equipo y de todas las partes del ciclo de vida.

- Control de cambios: RUP tiene fases que ayudan a reducir el costo del cambio. El control de cambio es una ayuda fundamental para el éxito del proyecto y el uso de las herramientas adecuadas puede desempeñar un poderoso papel en la gestión del cambio y reducir al mínimo su costo. Los cambios pueden surgir por diferentes razones, por cambios en el negocio, por puntos que no se tuvieron en cuenta a la hora de definir el requerimiento. Las razones pueden variar. Tener toda la documentación y pruebas ayuda a la introducción de un cambio.

- Control de requerimientos: Se debe documentar todos los requerimientos del cliente y analizar su impacto antes de dar una respuesta.

- Mantener el nivel de abstracción: Este principio motiva el uso de conceptos reutilizables como patrones de diseño o esquemas acompañados por representaciones visuales de la arquitectura por ejemplo con UML. Este último se ha convertido en un estándar para representar proyectos y es ampliamente utilizado por RUP. El uso de modelos visuales permite a los miembros tener una mejor comprensión del problema y así involucrarse más en el proyecto.

3.4 KANBAN

La metodología Kanban fue creada por un ingeniero japonés que trabajaba en Toyota a principios del siglo XX. Surgió para optimizar la producción de automóviles.

El término Kanban se compone de la palabra 'Kan' que hace referencia a algo visual y 'Ban' que significa Tablero. [1][15]

La metodología Kanban muestra cómo administrar el flujo de trabajo a través de la visualización de las tareas en un tablero (Figura 6). Este puede ser una pizarra dividida en columnas (es lo que se usaba originalmente).

Cada columna representa una fase: Tareas por hacer, en proceso y hechas. Se pueden agregar las columnas que se consideren necesarias para llevar adelante el proceso de trabajo. Cada tarea que ingresa se representa con una tarjeta y entra en la columna de las tareas por hacer (Requested).

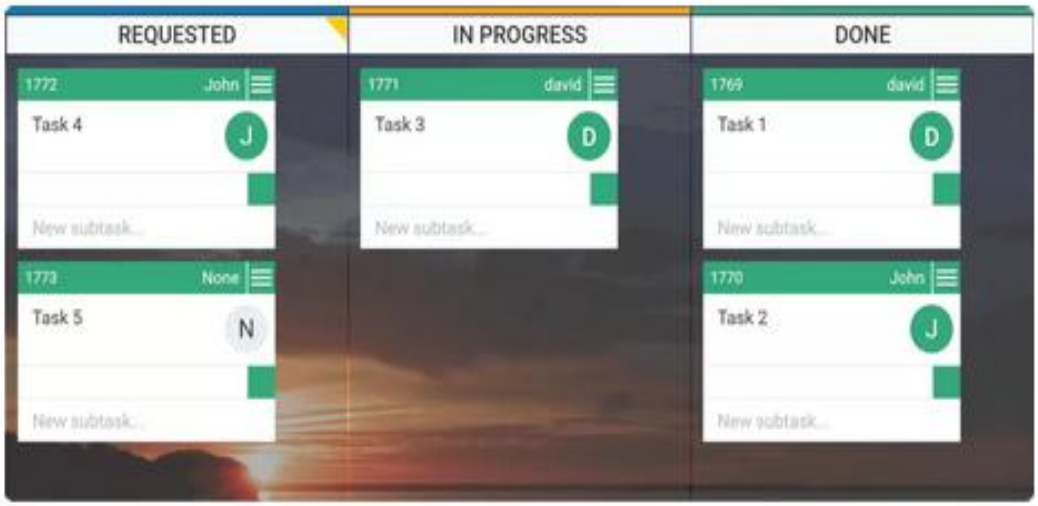

Figura 6 *Tablero de trabajo Kanban*

Si en una columna entran más tareas de las que salen se está frente a un problema. Se hará un cuello de botella.

La idea de Kanban es limitar el número de tareas que pueden realizarse en cada fase. Por ejemplo: 4 tareas en progreso, 1 en prueba, etc . Limitar el número de tareas se lo llama 'Work in progress' (WIP).

El ideal es centrarse en cerrar tareas y no en abrir nuevas. Limitar el wip impide abrir tareas nuevas hasta que se hayan cerrado en las que se está trabajando.

3.4.1 Principios de la metodología Kanban

Visualización

Tener todas las tareas en el tablero facilita la visualización de lo que cada uno tiene que hacer. Tener el flujo de trabajo visible ayuda a ver como avanza el trabajo.

Proceso

Se debe trabajar con una tarjeta a la vez; luego de finalizada se pasa a las columnas restantes. Las tareas deben ser pequeñas y de corta duración. Si durante su desarrollo surgen dudas o modificaciones estas pueden modificarse.

Priorización

La columna de tareas pendientes determina cual es la siguiente tarea a realizar. De esta forma se gestiona mejor los tiempos.

Acomodar las tareas en un orden de prioridades facilita el trabajo de la persona y del equipo.

Medir el Tiempo

Las tareas se van tomando de las pendientes y se van pasando a las siguientes. Esto nos permite medir el tiempo en que trabajamos en ellas. Pero es posible priorizar ciertas tareas que entren según las necesidades del proyecto y dar una respuesta ante un imprevisto.

3.4.2 Ventajas

Las ventajas que aporta Kanban son:

- Optimización de los tiempos de uso de los recursos dado por la organización de las tareas en columnas y el flujo de las mismas. Todos trabajan en el mismo tablero, comparten tareas y colaboran entre sí.

- Una mejor planificación, se limita el trabajo en curso y se achican las tareas. Menos tiempo de producción.

- Flexibilización de las respuestas, ya que las prioridades podrían cambiar. Es posible modificar las tareas entrantes para obtener mejores resultados.

- Es de fácil uso y se puede ver fácilmente el estado del proyecto.

En conclusión Kanban es aplicable para proyectos cortos que necesitan flexibilidad en el manejo de los requerimientos, tiempos cortos y un mejor trabajo en equipo.

3.5 Extreme Programing (XP)

La Programación Extrema (XP) es una metodología ágil que se basa en un conjunto de características y de buenas prácticas para aumentar la productividad a la hora de desarrollar un sistema. Busca que los desarrollos sean más sencillos. [10][14]

La metodología XP se desarrolla en 4 etapas: planificación, diseño, codificación y prueba

Planificación:

Comienza escuchando al usuario para recopilar los requerimientos, entender que necesita el cliente, creando 'historias' (historias de usuarios). El cliente escribe sus historias y luego el equipo de desarrollo evalúa cada una y estima su costo. Si este es muy elevado, entonces pide al usuario que la divida en historias más chicas. El cliente y el equipo trabajan en conjunto para decidir qué historia entra en cada entrega. La metodología XP hace énfasis en escuchar al cliente, saber si lo producido es lo que desea, los problemas de su negocio, explicar que es más sencillo de realizar y que no. La realimentación entre ambos ayuda al entendimiento de los problemas.

Diseño:

Los diseños para XP deben ser sencillos. Si alguna parte es compleja se debe dividir en partes. Si se detecta algo mal diseñado se debe solucionar lo antes posible. La metodología XP favorece el rediseño, esto es porque siempre puede mejorarse la estructura interna sin modificar la estructura externa. Esto limpia el código y así se minimiza la posibilidad de introducir errores. Para XP el diseño ocurre antes y durante se codifica el sistema.

Codificación:

XP plantea la codificación en parejas. La metodología XP propone que dos personas programen en una estación de trabajo. Cuando una pareja termina su parte, se integra al resto del sistema, esto lo hace el equipo de integración o la misma pareja de desarrolladores. Esta integración continua facilita la detección de errores a tiempo y la compatibilidad entre los desarrollos.

Prueba:

En esta fase se aplica la prueba unitaria, de esta forma se comprueba el funcionamiento de cada historia desarrollada. XP apunta a que estas pruebas están automatizadas de manera que puedan aplicarse cuando se modifica el código (esto puede ocurrir en forma seguida por el rediseño que plantea XP).

Para comprobar el funcionamiento final se crean las pruebas de aceptación, estas son usadas por los usuarios para comprobar que sus historias funcionan correctamente. Estas pruebas se basan en las características y funcionalidades generales del sistema.

3.5.1 Características de XP

Prácticas en las que se basa la metodología XP:

- *Desarrollo iterativo e incremental*: trabajar en forma continua, con pequeños incrementos que pueda visualizar el cliente.
- *Programación en parejas*: las tareas de desarrollo se realizan de a dos programadores por puesto.
- *Pruebas unitarias continuas*: estas pruebas se realizan antes del desarrollo. Deben ser automatizadas y deben aplicarse frecuentemente. Cada historia desarrollada debe pasar las pruebas unitarias.
- *Integración continua*: cuando una historia esta lista se integra al sistema.
- *Simplicidad*: mantener el código lo más simple posible. Es mejor desarrollarlo en forma simple y ante cualquier cambio trabajar un poco más si lo requiere.
- *Refactorización*: reescribir código para que sea más legible pero sin modificar el comportamiento. Las pruebas se encargan de verificar que no se ha introducido algún error.
- *Integración del equipo con el cliente*: algún representante trabaja con el equipo y debe estar en comunicación continua.

- *Normas de desarrollo*: deben seguirse una normativa de desarrollo para escribir código legible. De esta forma todo el código parece haberse escrito por un solo desarrollador.
- *Código compartido*: cualquier desarrollador puede modificar o extender cualquier parte del sistema. La idea es que todos conozcan el código y no solo el modulo en el que trabajan.
- *Corregir errores*: se deben corregir todos los errores antes de agregar nueva funcionalidad.

3.6 Conclusión

La metodología Scrum se centra en proyectos que necesitan mejoras rápidas, ser ágil en cuanto a los cambios, mantener las entregas frecuentes y soportar la flexibilidad de los requerimientos.

RUP es una metodología que se caracteriza por estar centrada en la arquitectura, ser iterativo e incremental y basarse en casos de uso. Incluye artefactos, que son los productos tangibles, y roles, que es el papel que desempeña una persona en un determinado momento.

RUP es más apropiado para proyectos grandes pues se requiere de un gran equipo de trabajo para un proceso complejo. Pretende obtener productos de buena calidad al estar formado por varias fases, iteraciones etc.

No es recomendado para proyectos pequeños, ya que es posible que no puedan cubrirse los costos de dedicación. Genera trabajo adicional de documentación y comunicación por lo que puede no resultar práctico.

La metodología XP es adaptable, organizada, tiene baja probabilidad de error y una buena comunicación con el cliente, sin embargo funciona mejor en proyectos cortos y en caso de falla los costos son altos. La programación en parejas no siempre es algo positivo, a algunos programadores no les gusta que otro programador modifique su código.

Para concluir la metodología Kanban ayuda a aumentar la eficiencia en los procesos, no desaprovechar recursos, reduce los tiempos muertos y evita la sobreproducción.

Capitulo 4: Proyecto Esidif

4.1 Introducción

A partir de la sanción de la Ley de Administración Financiera la Dirección General de Sistemas de Administración Financiera comenzó su camino ofreciendo más y mejores servicios, hasta llegar a lo que es hoy en la actualidad.

En un principio se desarrolló el Sistema Integrado de Información Financiera (SIDIF) era un sistema compuesto por varios subsistemas con una base de datos central y tantas bases como servicios de administración existieran. (Figura 7)

Había una diversidad de sistemas operando con el SIDIF central. Los sistemas se comunicaban con la base de datos central.

El SIDIF se siguió expandiendo a más organismos. Con el crecimiento y la cantidad de sistemas satélites comenzó una transformación que derivó en un único sistema central SLU (SIDIF Local Unificado). Este sistema amplió los requerimientos funcionales e incorporó equipamiento para soportar el crecimiento en cantidad de usuarios y operaciones.

En ese momento, se realiza un proceso de réplica del sistema SLU en Organismos de la Administración con el objeto de reemplazar los sistemas existentes. En pocos años el SLU ya era usado por 63 organismos.

Luego de este crecimiento, con los avances tecnológicos y el crecimiento profesional de las personas involucradas se dio lugar al desarrollo de una herramienta con plataforma web denominada Esidif basada en el alcance funcional y la gestión orientada a resultados.

Desde el principio se pensó como un producto adaptable al entorno y diferentes tamaños de organizaciones, navegable y con una única base de datos centralizada donde se almacenaría toda la gestión de los organismos del estado y con una arquitectura en capas que facilitaría el mantenimiento y la evolución del sistema.

A partir de su concepción, Esidif interopera con sistemas de otros organismos del estado para mantener los datos en línea, consistentes y actualizados a través de servicios web.

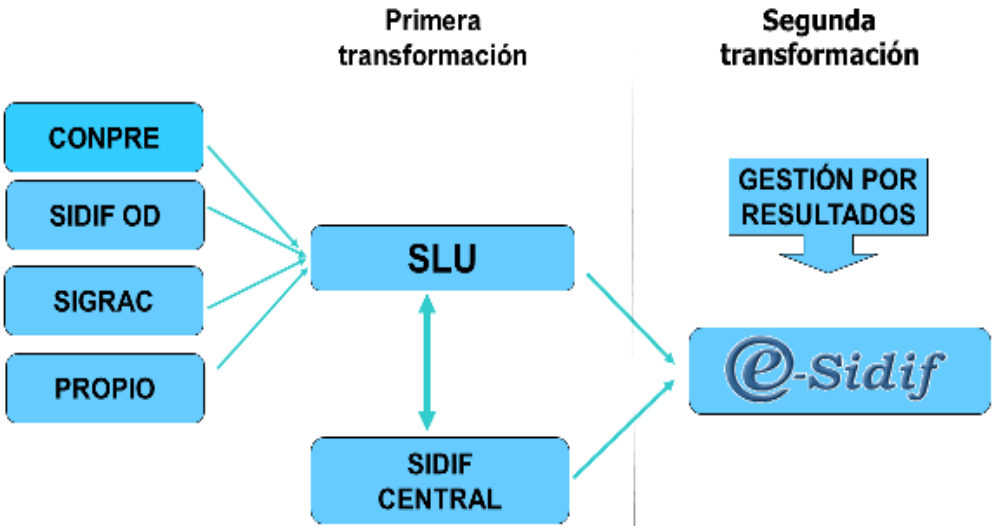

Figura 7 *Transformaciones de los sistemas*

4.1.2 Esidif: beneficios

Al llegar a la segunda transformación (nacimiento de Esidif) se logró:

Tener una base de datos única, gestión, registro y control en simultáneo, conseguir mejoras en las buenas prácticas de administración financiera, conseguir gestión por resultados y ser el único sistema de administración financiera. (Figura 8)

Características del Sistema

- Autonomía del usuario
- Obtención de distintas visiones de la información.
- Mayor seguridad
- Auditoría
- Fácil uso
- Descentralización operativa
- Facilidad en la toma de decisiones
- Despliegues puestos en producción de manera incremental

Subproyectos asociados a la transformación de Esidif:

SubProy Convivencia

Este subproyecto hace referencia a un conjunto de programas que se utilizan para mantener espejadas y sincronizadas las bases de datos de los sistemas a reemplazar.

SubProy Migración

Este subproyecto se refiere a la manera de extraer los datos del sistema actual e importarlos en el nuevo sistema con las modificaciones que fueran necesarias.

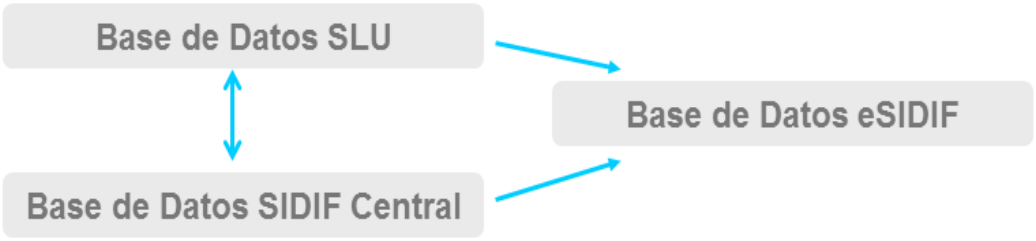

Figura 8 *Migración de base de datos*

SubProy Desarrollo

Se realiza la implementación del sistema Esidif por modulos de negocio. Por ej : módulo Presupuesto, Gastos.

SubProy Mantenimiento de los módulos implementados

Este subproyecto lleva a cabo el mantenimiento a los módulos que se van implementando

4.1.3 Arquitectura tecnológica

La arquitectura elegida fue JEE (Java Enterprise Edition). Se comenzó con una versión anterior pero a medida que las versiones fueron mejorando, se fue migrando hasta llegar a la actual (JEE).

El lenguaje utilizado para la implementación de Esidif fue Java y el entorno Eclipse.

La metodología utilizada es RUP con adaptaciones ágiles (Scrum, Kanban, XP) según las necesidades del proyecto.

En el siguiente capítulo se explicará con mayor detalle.

4.1.4 Formación de los equipos del proyecto Esidif

Esidif es un proyecto, en la actualidad, con un gran número de personas. Al momento cuenta con 165 personas.

A continuación se presenta su formación y organización de las áreas y grupos de trabajo.(Figura 9)

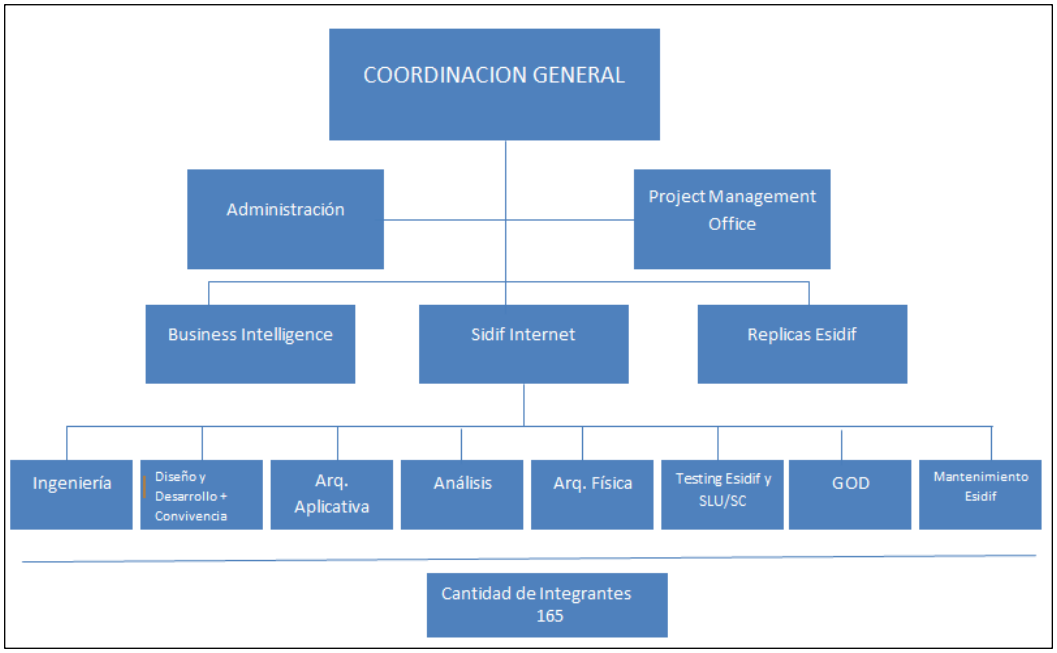

Figura 9 *Organigrama general del Esidif*

El proyecto se encuentra dividido en áreas donde cada una tiene distintas tareas asignadas.

A continuación se realiza una breve explicación de las funciones de cada área, descripta por niveles:

1. El primero está formado por la coordinación general (1 persona) la cual lleva adelante la dirección del proyecto cumpliendo los objetivos que le solicita la Subsecretaria de Presupuesto.
2. El segundo nivel está formado por:

   (a) *Area de Administración* : cuenta con 3 integrantes y es quien lleva adelante la administración del personal, generación de documentación y la gestión de expedientes.

   (b) *Area de Proyect Managment Office* (PMO) que realiza tareas de planificación y gestión del proyecto y de los asociados, contratación de recursos y gestión de despliegue. Está compuesta por 3 integrantes.

   .

3. En el tercer nivel, está **Sidif Internet** que se dividen en las siguientes áreas:

   a) *Ingeniería:* Esta área tiene como objetivo principal el de definir procesos para llevar adelante el proyecto en las distintas áreas y proveer un estándar de herramientas que soporten el proceso definido. Las herramientas definidas las especificaremos más adelante.

   Además llevan adelante distintas funciones:

   - Definir procesos y guías de trabajo
   - Investigar las disciplinas del proceso RUP adaptándolas al proyecto y definir roles y artefactos.
   - Supervisar el cumplimiento de las pautas definidas para el control de calidad de los artefactos generados.
   - Estandarizar los artefactos utilizados: Templetes, Guías de buenas prácticas, Material de Capacitación, etc.
   - Desarrollar herramientas para automatizar tareas repetitivas
   - Brindar capacitación y soporte al equipo durante el proceso de desarrollo
   - Elaborar las guías de instalación y uso de herramientas.
   - Investigar, seleccionar y desplegar herramientas de soporte que ayuden al proceso de desarrollo.

- Investigar y adaptar prácticas de las metodologías ágiles al proyecto.

Esta área está formada por 5 integrantes y se encuentra dividida según las funciones que realizan (Figura 10):

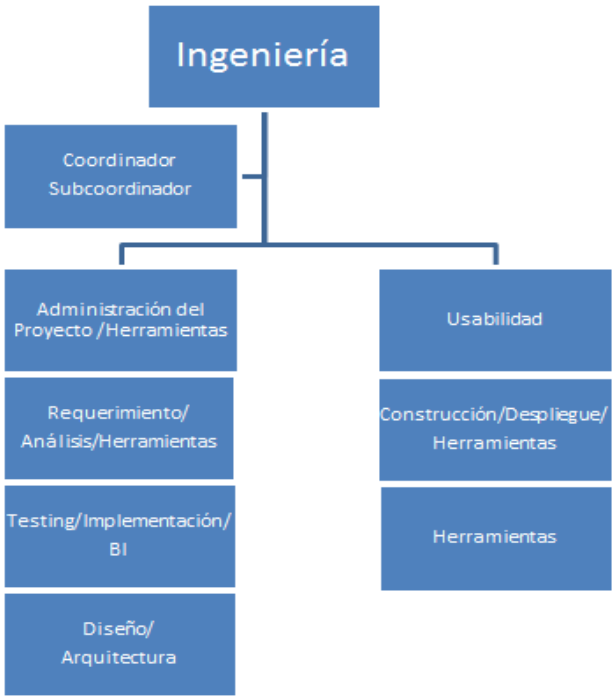

Figura 10 ***Organigrama área de Ingeniería***

(b) *Diseño y Desarrollo + Convivencia* Esta área tiene como función principal la de implementar los requerimientos del usuario en artefactos visibles y ejecutables.

Entre las tareas que realizan mencionamos:

- Modelar y construir componentes que permitan llevar adelante la funcionalidad del sistema, utilizando componentes genéricas provistas por arquitectura.
- Construir algoritmos de convivencia entre sistemas legados desarrollados por el equipo de convivencia.

Esta área está formada por 57 integrantes y se divide como muestra la figura siguiente (Figura 11).

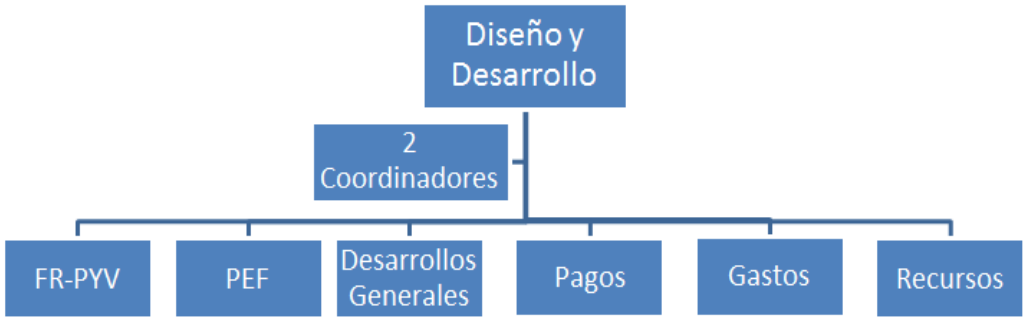

Figura 11 *Organigrama área desarrollo*

.

(c) *Convivencia (Desarrollo y mantenimiento).* Esta área tiene como función principal la construcción de algoritmos en PL/SQL que mantengan la actualización y control de datos entre todos los sistemas que componen el Esidif.

Esta área está formada por 4 integrantes.(Figura 12)

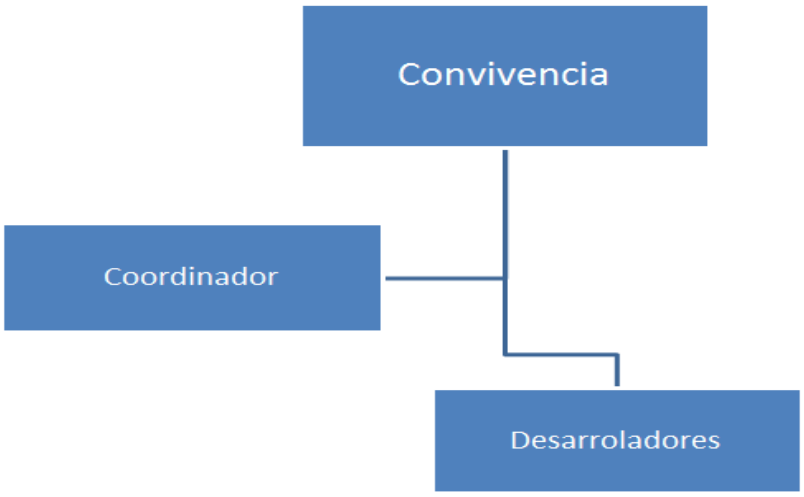

Figura 12 *Organigrama área convivencia*

(d) *Arquitectura Aplicativa* Esta área tiene como objetivo principal definir la arquitectura del proyecto Esidif y la plataforma de desarrollo sobre la que se construye el mismo.

Deben proveer la actualización tecnológica de la plataforma, brindar soporte a los equipos para su uso y tener una visión de mejora continua.

Entre sus funciones podemos mencionar:

- Definir la arquitectura del software
- Diseñar, implementar y mantener la plataforma de desarrollo
- Evaluar nuevas tecnologías que puedan brindar mejoras a la plataforma.
- Capacitar a los equipos
- Realizar manuales de uso de cada uno de los componentes de la plataforma.

El área está formada por 8 integrantes y se compone como muestra la figura siguiente (Figura 13).

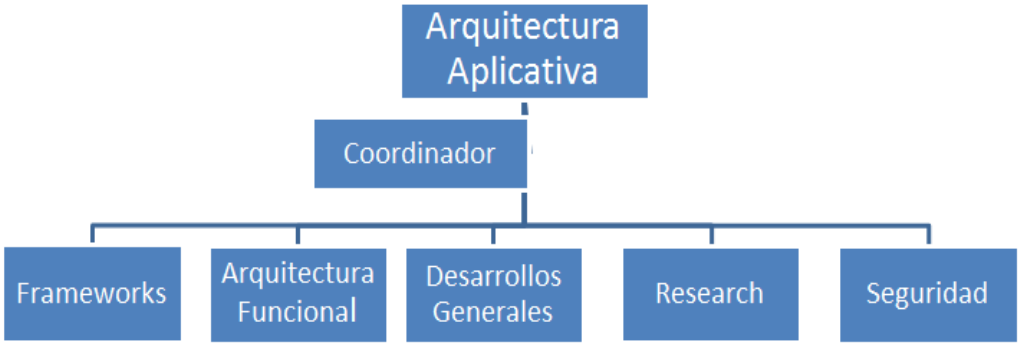

Figura 13 *Organigrama área Arquitectura Aplicativa*

(e) *Análisis. Esta* área tiene como misión principal interpretar los requerimientos del usuario, elaborar la solución y materializarla en los modelos definidos en el proceso utilizando el lenguaje de modelado UML.

Algunas de sus funciones son:

- Realizar los talleres de relevamiento de requerimientos con los usuarios.
- Modelar la solución a los requerimientos relevados. Estimar el tiempo y esfuerzo de cada negocio.
- Implementar los modelos UML definidos en la metodología del proyecto (ej: MCU, MRN, MCA)
- Generar la documentación de despliegue
- Atender las consultas funcionales
- Revisar y controlar casos de prueba y modelos de base de datos de otros equipos

El *Área de Análisis* está compuesta por 23 personas incluyendo a los coordinadores y se divide como muestra la siguiente figura (Figura 14).

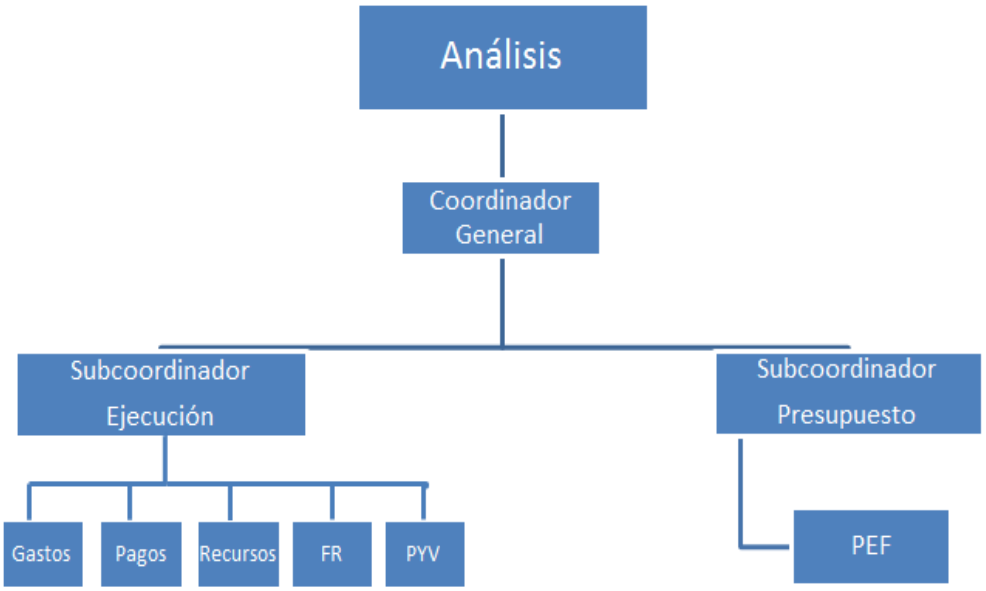

Figura 14 ***Organigrama área análisis***

(f) *Arquitectura Física.* El objetivo de esta área es monitorear el ambiente de producción para detectar potenciales problemas y proponer alternativas de solución.

Algunas de sus funciones son:

- Intervenir en el armado de los entornos sobre los que corre la aplicación para lograr el mejor rendimiento de la misma.
- Monitorear los ambientes de testing y producción para detectar comportamientos no deseados. Se investiga la causa y luego se reporta para su resolución.
- Pruebas antes de los cambios de versión o configuración

Esta área está compuesta por un coordinador y 10 integrantes que realizan las tareas.

(g) *Testing:* El objetivo principal de esta área es controlar la calidad de la aplicación, encontrando y documentando defectos y validar que los requerimientos se hayan desarrollado correctamente.

Algunas de sus funciones son:

- Diseñar los casos de prueba.
- Ejecutar las pruebas de integración y regresión.
- Reportar los defectos encontrados durante la prueba.
- Pasaje a producción de las entregas.
-

Esta área está compuesta por alrededor de 30 personas incluyendo al coordinador y dos subcoordinadores. Se divide como muestra la figura siguiente (Figura 15)

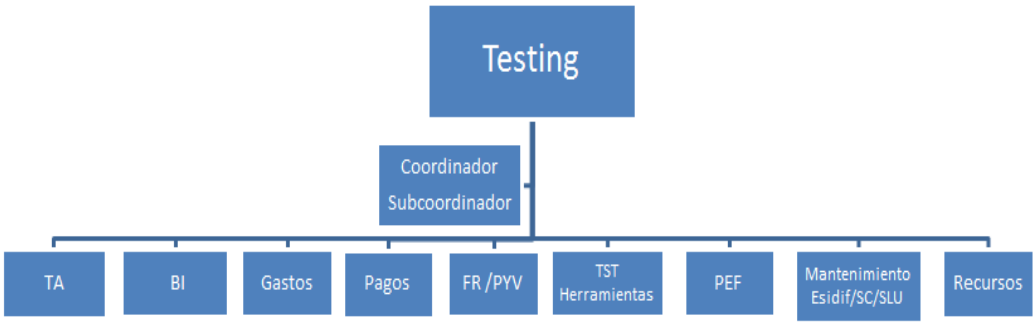

Figura 15 ***Organigrama área Testing***

(h) GOD (Grupo de Orientación del Diseño)

Esta área tiene la misión de optimizar los productos intermedios del ciclo de desarrollo para mejorar la performance y ver oportunidades de reuso. Está compuesta por 5 personas.

Algunas de sus responsabilidades principales son:

- Fomentar la reutilización de código normalizado y pantallas comunes
- Asegurar los procesos de calidad en el modelado de análisis y diseño
- Colaborar con arquitectura en la implementación de componentes
- Colaborar con el equipo de ingeniería en la definición de pautas en los procesos de desarrollo.

(i) Mantenimiento

Por último en este nivel se encuentra el área de mantenimiento. Esta se encarga de tomar los requerimientos de los usuarios de las aplicaciones de producción, los cuales pueden ser nueva funcionalidad o alguna corrección

Entre sus funciones encontramos:

- Interpretar los requerimientos de los usuarios
- Asistir a las reuniones para priorizar los requerimientos
- Participar en la mejora de procesos

Esta área está formada por 14 integrantes pero están divididos según las tareas del sistema al que le realicen el mantenimiento.(Figura16)

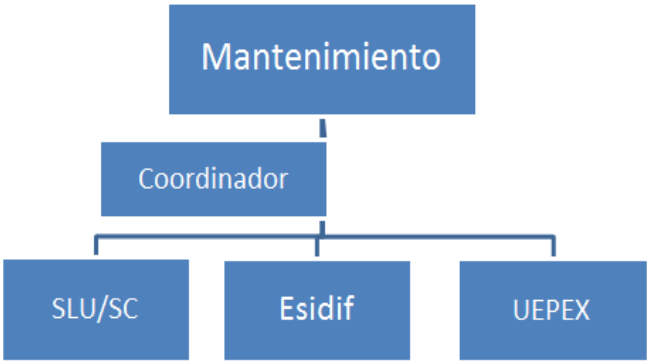

Figura 16 ***Organigrama área de mantenimiento***

En el tercer nivel encontramos también al área de Business Intelligence (BI) y Réplicas Esidif.

La primera se encarga de interpretar requerimientos de niveles ejecutivos y desarrollarlos con herramientas gerenciales. Está compuesta por 4 integrantes. La siguiente área es Replicas su misión principal es la capacitación de los usuarios hasta que alcanzan su autonomía. Está compuesta por 6 integrantes.

Sus funciones están vinculadas a la etapa del proyecto:

- Si el proyecto está en etapa de pasaje a producción realizan demos y capacitaciones. Dan asistencia y soporte a usuarios finales.
- Si el proyecto está en etapa de desarrollo participan en talleres funcionales, reuniones de avance y evaluación de negocios.

4.1.5Herramientas utilizadas en el proyecto

Las herramientas que se utilizan (aún hoy) en el proyecto son provistas por el ingeniería. Las mismas las nombramos a continuación (Figura 17)

Herramienta	Descripción
Enterprise Architect (EA)	Herramienta para modelado UML. La utiliza análisis para modelar MCU,MCA, DTE, MRN, etc
Eclipse	Entorno integrado de desarrollo de software.
CVS	Sistema para control de versiones. Se utiliza integrado con eclipse para el versionado del proyecto.
Item	Sistema de administración de requerimientos.
QuickItem	Sistema que permite acceder más rápidamente a la información administrada por ítem.
BAMBOO	Integración continúa.
Sitio BWA	Reportes de la construcción de la aplicación
PSI	Software interno de mensajería instantánea entre los integrantes del proyecto.
GLO	Glosario funcional.
Windows Tester	Automatización de pruebas
TOOLKIT	Automatiza tareas recurrentes de los desarrolladores
MOIN MOIN	Wiki, portal colaborativo
NEA	Notas de entrega automáticas.
SGR	Software para administrar los requerimientos de todo tipo. Principalmente para defectos y fallas en el sistema.
Agilefant	Herramienta de seguimiento.
Trello	Software de administración del proyecto
Wiki	Repositorio de información accesible por todo el equipo.
Website	Herramienta para la integración de los desarrollos al proyecto.

Figura 17 *Herramientas utilizadas*

4.1.6 Herramienta Trello

Trello es un software de administración de proyectos basado en el método Kanban, que permite gestionar tareas mediante tableros virtuales.

Es conocido por su fácil uso y posee herramientas y funciones fáciles de entender.

Los tableros permiten que todos los miembros del equipo tengan una visión compartida del trabajo en curso y de lo que queda por hacer.

Trello se compone de tarjetas y listas. Las listas representan un proceso o flujo de trabajo y las tarjetas, representan tareas, las cuales avanzan en las listas a medida que se completan.

En el caso del proyecto Esidif, se utiliza para organizar la planificación anual del proyecto. En este se visualizan las tareas pendientes agrupadas por trimestre.

4.1.7 Herramienta Agilefant

El Agilefant es una herramienta para el manejo de proyectos en cuanto a gestión de tareas y organización. Facilita la organización dentro del equipo y a su vez cada integrante es independiente durante el sprint. Éste va volcando el esfuerzo diario de la tarea realizada y de esta manera el coordinador puede tener un seguimiento detallado del avance de las tareas en el sprint.

Cada integrante accede a su sesión, donde puede ver en la parte izquierda los Product Backlog en los que se encuentra incluido.

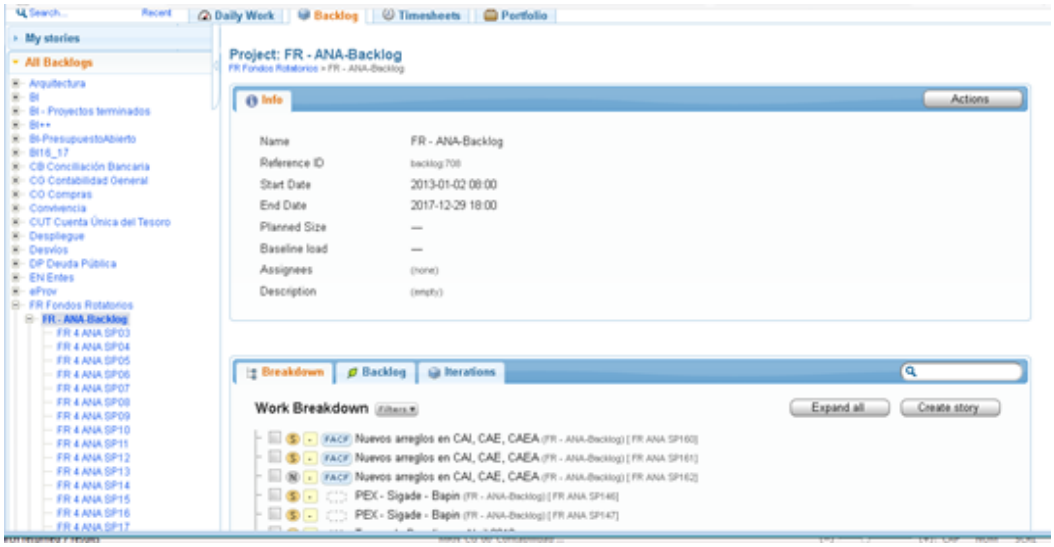

Figura 18 *Product Backlog en Agilefant*

En la imagen, a la izquierda en la parte superior My Stories es una lista de todas las stories y debajo se muestra All Backlogs es la lista de Product Backlog creados. (Figura 18)

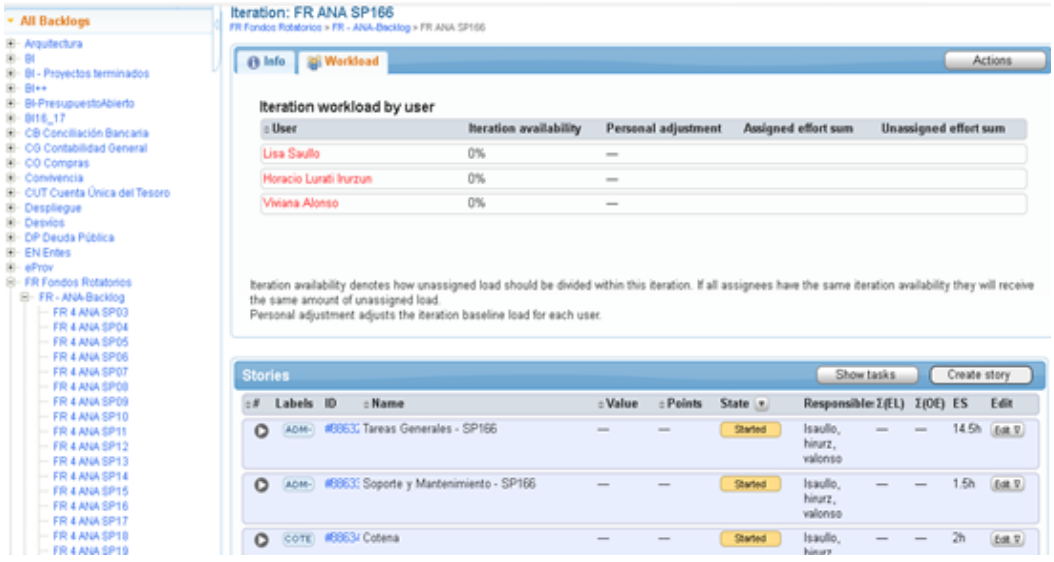

Figura 19 *Ejemplo de sprint*

En la Figura 19 se muestra con más detalle cómo se forma el sprint. En la parte izquierda de la imagen se encuentra en menú FR- Fondos Rotatorios con la lista de sprint de ese negocio. En la parte superior se presentan los integrantes del

negocio y en la parte inferior, la lista de tareas que forman el sprint agrupadas por stories.

En la Figura 20 se visualiza el Sprint Backlog. El estado de la story depende del estado de la tarea. Si la tarea ya fue comenzada, está en started y la story también lo estará. Cuando la tarea sea finalizada y cambie a estado Done, la story quedará en estado finalizada. En esta figura se visualiza además cada tarea que está dividida en un conjunto de subtareas. Cuando una tarea es tomada se cambia el estado a Started, al finalizar el dia se vuelca el esfuerzo (ES) en horas trabajadas, dedicadas a esa tarea. Al finalizar la tarea cambia a estado Done.

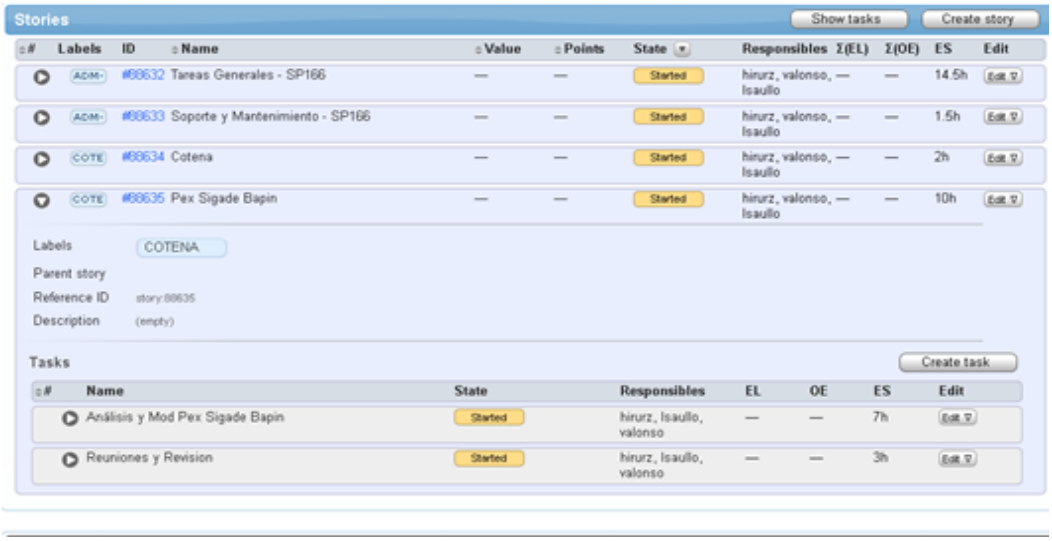

Figura 20 *Sprint Backlog*

Capítulo 5: Caso de Aplicación

5.1 Introducción

En este capítulo se explicará cómo se trabaja en el proyecto Esidif para evidenciar la convivencia de metodologías.

El proceso que se muestra en la Figura 21 representa la secuencia de pasos que se llevan a cabo cuando surge un nuevo requerimiento o funcionalidad al proyecto Esidif.

Dicho proyecto ya se encuentra en funcionamiento por lo que el Product Backlog ya está creado y se va actualizando con cada nuevo requerimiento.

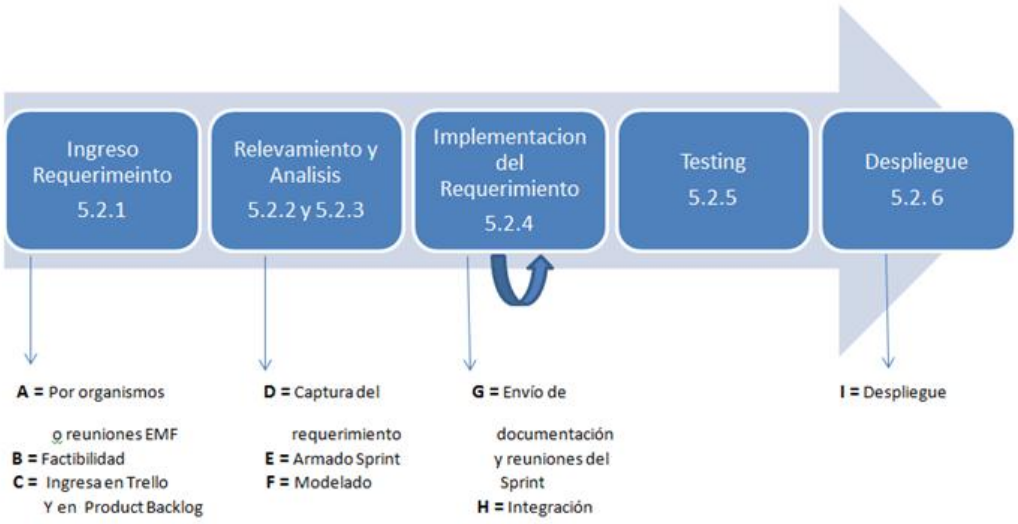

Figura 21 *Mapa de Proceso- Nuevo Requerimiento*

Descripción de ítems del mapa de Proceso

A. Ingresa un requerimiento

B. EL requerimiento entra al área de análisis para su evaluación de factibilidad

C. El requerimiento es agregado en el tablero Trello con una nueva tarjeta y en el product Backlog

D. El equipo de análisis se reúne con el usuario a relevar el requerimiento

E. El requerimiento es tomado en el sprint que comienza

F. El equipo de análisis realiza el modelado del requerimiento

G. Se envía la documentación al equipo de desarrollo para su implementación y al equipo de testing para que realice el caso de prueba.

H. Integración del requerimiento en el sistema y testing del mismo.

I. Despliegue.

5.2 Proceso

5.2.1 Ingreso del requerimiento

En el momento en que se solicita una nueva funcionalidad o requerimiento(A), el negocio(1) del *Área de Análisis* a quien le corresponde el requerimiento, evalúa la factibilidad, alcance y revisa la planificación para estimar cuándo podría iniciarse(B).

Un nuevo requerimiento puede surgir por un organismo que solicita una nueva funcionalidad, o puede surgir en las reuniones EMF (Equipo Multidisciplinario Funcional). Estas reuniones se llevan a cabo cada 15 días. En estas se reúnen personas de distintos sectores, clientes y los referentes funcionales del proyecto. Se discute y planifican ajustes a las funcionalidades y además puede surgir algún nuevo taller o relevamiento específico para el sistema.

El requerimiento se agrega en una tarjeta en Trello (C) y además en el Product Backlog.(Figura 22).

El tablero se utiliza para poder gestionar la planificación del proyecto y tener las principales tareas visualizadas, ya sean de los negocios que lo componen, de los sistemas satélites, o cualquier tarea que necesite planificarse. Esto mejora la organización, ya que cuando llega un nuevo requerimiento se revisan las tarjetas trello para poder estimar cuando es posible tomar ese requerimiento.

(1) Se llama a un módulo funcional del proyecto Ej: Gastos, Pagos a un módulo funcional del proyecto Ej: Gastos, Pagos

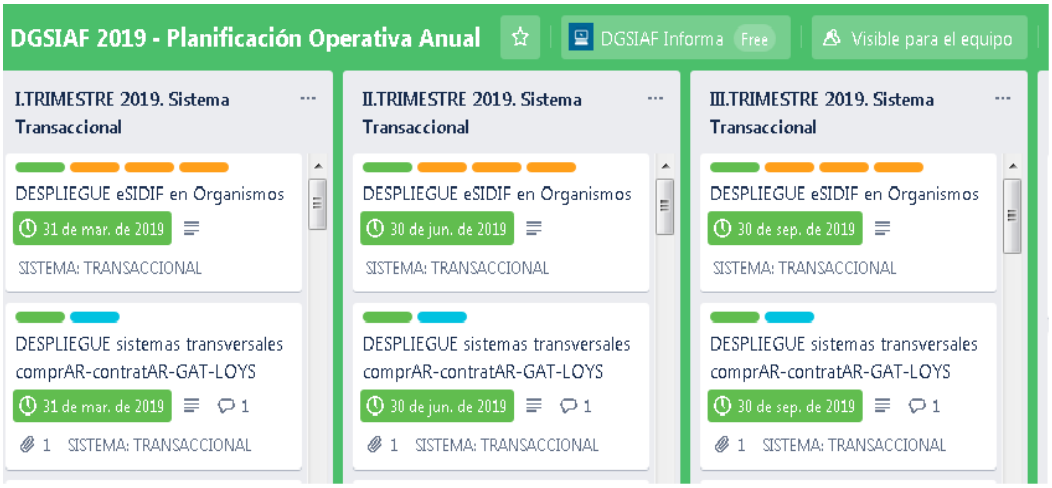

Figura 22 *Tablero de planificación del proyecto (Trello)*

5.2.2 Relevamiento del Requerimiento

Para poder lograr el detalle que el *Área de Diseño y Desarrollo* requiere para la implementación del requerimiento, el *Área de Análisis* se reúne con los usuarios (D) en reuniones denominadas Talleres, donde se detalla el requerimiento solicitado.

El objetivo de los talleres funcionales es lograr llegar a definir el requerimiento comprendiendo la funcionalidad y lo que el usuario necesita.

A continuación los analistas diseñan prototipos de pantallas (PIUs Prototipos de Interfaz de Usuario), ejemplos de diagramas de casos de uso y una minuta del requerimiento en base a lo descrito por el usuario.

A medida que se avanza con los talleres, donde se presenta todo lo realizado por los analistas, haciendo el usuario algún ajuste si es necesario, se refinan los detalles y los analistas pueden pasar a la etapa de *Análisis del requerimiento.*

En base a las tareas que se encuentran en el Product Backlog, se establecen, en conjunto con el resto del equipo, las tareas que va a incluir el nuevo sprint (E).

Se listan las tareas que implican la implementación del requerimiento y se hace la estimación de tiempos de las tareas y se arma el Sprint backlog.

Cada Área (análisis, desarrollo y testing) lleva su propia organización de las tareas.

5.2.3 Análisis del Requerimiento

Para esta etapa se aplicaron los principios de la metodología RUP adaptados al proyecto. Se realizan fases y se generan artefactos que luego serán consumidos por las *Áreas de Diseño y Desarrollo y Testing*.

Las fases en esta etapa son: Captura del requerimiento y Análisis del requerimiento. En estas fases participa además el *Área de Réplicas* que cumple con sus roles según la fase. Tiene la responsabilidad de aportar la visión del usuario.

Los analistas tienen la responsabilidad de relevar el requerimiento y realizar el modelado de los procesos de negocio.

Los analistas realizan el análisis y modelado del requerimiento (F) volcando en artefactos la funcionalidad del requerimiento, que luego será consumida por el *Área de Diseño y Desarrollo* y por el *Área de Testing*, logrando con el modelado un conocimiento detallado de la funcionalidad del requerimiento.

En el análisis del requerimiento los puntos principales son:

- Modelado de los casos de uso, descubriendo las clases y atributos necesarios.
- Modelado de los diagramas de clases y las relaciones entre ellas.
- Los diagramas de transición de estados para cada objeto.

Los productos generados como consecuencia en la etapa de análisis son volcados en el EA (Enterprise Architecture).

En esta se generan los artefactos (Figura 23):

- MDR: Minuta de Requerimientos
- LBR: Línea Base de Requerimientos
- MCU : Modelo de Casos de Uso
- MCA: Modelo de Clases de Análisis.
- MRN: Modelo de Reglas de Negocio
- DTE: Diagrama de Transición de Estados
- PIU: Prototipo de Interfaz de Usuarios
- MSJ: Mensajes al Usuario.

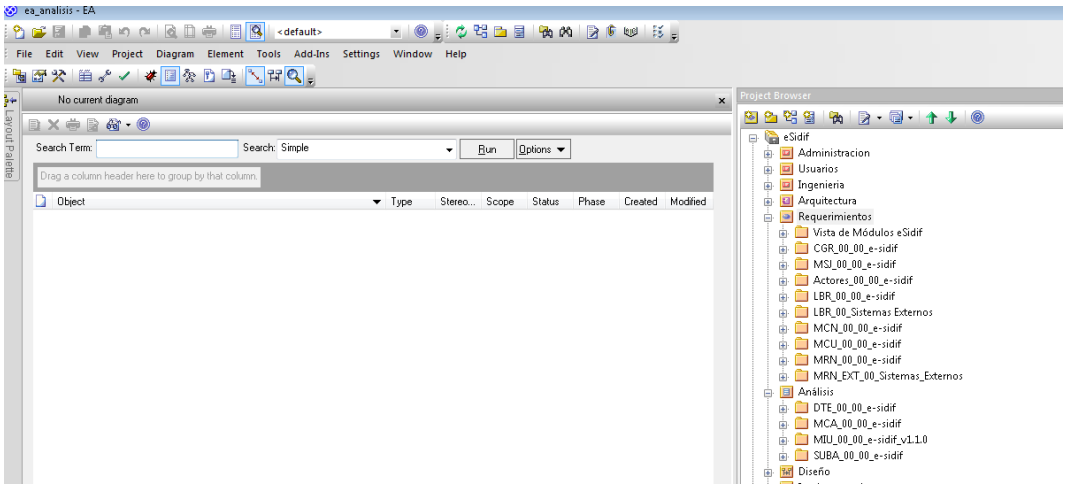

Figura 23 **Listado de Artefactos en EA Enterprise Architect**

Artefactos Generados

- MDR (Minuta de Requerimientos) : Es la documentación donde se registran las conclusiones de los talleres funcionales. Incluye la descripción funcional, los problemas encontrados, los requerimientos propuestos, el glosario utilizado en el taller y las personas que participan.
- LBR (Línea Base de Requerimiento) : es el repositorio de la documentación de los requerimientos. Es la base para generar los casos de uso.
- MCU (Modelo de Casos de Uso) : El caso de uso es un diagrama que describe la funcionalidad y la interacción entre el actor y el sistema. Debe especificar las condiciones, funciones y reglas (MRN) que sean necesarias para describir la funcionalidad.
- El caso de uso es la base para las áreas de desarrollo y testing, es la guia para entender la funcionalidad y poder implementar o diseñar el caso de prueba.
- MCA (Modelo de Clases de Análisis) : Las clases de análisis especifican los elementos del modelo conceptual para el sistema. Representan las clases sugeridas y son el primer paso de la abstracción principal que tendrá el sistema.
- MRN (Modelo de Reglas del Negocio) : Las reglas de negocio especifican características y comportamiento definido en el requerimiento.

- Describen las restricciones, funciones, validaciones, valores permitidos, impactos, obligatoriedad, visualización. Son las que completan, junto con la especificación del cu , las interacciones y relaciones entre objetos.
- DTE (Diagrama de Transición de Estados) : El diagrama de transición de estados muestra los estados que puede pasar un objeto (por ej un comprobante del sistema) durante su ciclo de vida y los eventos que provocan la transición de un estado a otro.
- PIU (Prototipo de Interfaz de Usuario) : Especifica la interfaz que verá el usuario para el caso de uso. En dicha ventana se detalla los componentes, la información de las entidades a mostrar y la navegación.
- MSJ (Mensajes al Usuario) : Los mensajes al usuario ocurren en cualquier momento del caso de uso, lo orientan a cómo seguir operando , los informan ante una acción, o les muestran el resultado de una operación. Son invocados desde las reglas de negocio.

El objetivo del modelado de análisis es que los desarrolladores y testers puedan entender el requerimiento especificado.

El modelado no tiene relación con la tecnología en la cual se desarrolla el sistema. Es independiente de la tecnología utilizada.

Con el modelado de los distintos artefactos el *Área de Diseño y Desarrollo* analiza la documentación y puede tomar las decisiones de diseño e implementación que considere para llevar adelante la funcionalidad requerida.

El *Área de Testing* por su parte, consume la documentación generada por los analistas y genera los casos de prueba que considere necesarios para realizar el testeo del nuevo requerimiento.

5.2.4 Implementación del requerimiento

Cuando el *Área de Análisis* finaliza la especificación del nuevo requerimiento, completando las tareas de especificación del negocio, registrando los términos en el glosario, habiendo validado, en conjunto con el equipo de EMF, el alcance e impacto de la funcionalidad e identificando posibles problemas, el requerimiento está listo para que sea consumido por el *Área de Diseño y Desarrollo*(G).

Este equipo lleva adelante la implementación del requerimiento. El desarrollo se realiza en Java con Eclipse, integrado con Toolkit para la sincronización con el repositorio. De esta forma se pueden manejar versiones del proyecto.

La interacción con el *Área de Análisis* es realizada a través del EA donde consumen los artefactos generados para comprender la funcionalidad y poder desarrollar el requerimiento.

Integración (H)

Al finalizar el sprint se realiza la integración. El *Área de Diseño y Desarrollo* genera su versión que luego son integradas en la versión final del producto. La integración es realizada con la herramienta Website provista por el *Área de ingeniería.*

Esta tarea representa un nuevo incremento a la funcionalidad y genera un nuevo release del sistema.

La persona encargada de realizar esta tarea se denomina Integrador. Es un miembro del *Área de Diseño y Desarrollo*. El rol se va asignando a distintos miembros del equipo (rotando) con los sprints. Al finalizar la integración, la nueva versión es instalada en un entorno de prueba y se comunica a todas las áreas de testing para que realicen las pruebas necesarias. Si hubo algún error es informado para su corrección y, en consecuencia, se genera una nueva integración.

Reuniones dentro del sprint

Planning

La reunión de planning se realiza el primer día del sprint, el equipo se reúne para realizar la planificación de las tareas que lo conforman.

Cada integrante estima el tiempo que le puede llevar la realización de la tarea que se le asigna. De esta forma se arma el Sprint Backlog con la lista de tareas y el tiempo estimado de cada una.

Reunión Diaria (Daily)

Las daily son reuniones diarias que se realizan una vez comenzado el sprint. La realizan todos los negocios y se hacen todos los días en el mismo horario, participan las áreas de desarrollo, testing y análisis. No dura más de 15 minutos y el objetivo es intercambiar información entre los miembros para facilitar el avance de las tareas y evitar posibles imprevistos.

Las reuniones de daily se realizaban todos los días en un principio, luego cada grupo fue acomodando la cantidad según la necesidad. Hoy en líneas generales se realiza 3 veces por semana.

Retrospectiva

La reunión retrospectiva se realiza al finalizar el sprint. Se revisan los objetivos y porque alguno de ellos tuvo dificultad para ser alcanzado.

El equipo comenta el desempeño y revisa en qué áreas es posible mejorar para llegar a una mejor eficiencia en el siguiente sprint.

5.2.5 Testing

El equipo de testing lleva adelante el testeo de las funcionalidades, utilizando los casos de prueba desarrollados, como mencionamos anteriormente. Si detecta un error o mejora cargan en la herramienta un nuevo ítem (utilizan la herramienta de Ítem mencionada en la lista de herramientas utilizadas en el proyecto) con lo encontrado y una breve explicación.

Al cargar un ítem, se selecciona el negocio para el cual se está realizando el testeo, le llega un aviso al área de desarrollo para que pueda realizar las correcciones necesarias. Cuando el *Área de Testing* vuelva a probar la nueva integración y chequee que el error fue corregido, pasa a estado cerrado el ítem levantado. De esta forma se llega a la versión final de la nueva funcionalidad.

5.2.6 Despliegue

En esta instancia, se instala una nueva versión del sistema en el entorno de producción. Es cuando el usuario puede visualizar la funcionalidad en el sistema y comienza a utilizarla. Esta etapa la realiza el *Área de Soporte*.

En esta etapa el *Área de Réplicas* interviene realizando capacitaciones, manuales de usuario o asistiendo al usuario cuando este lo requiere.

5.3 Reuniones

Reuniones Organizativas (N1 a N5)

Estas reuniones son parte de una metodología del proyecto adaptada a los enfoques del PMI y prácticas del PMBox. (PMBox es la guía de referencia para la gestión de proyectos en las organizaciones).

Para la organización cada tipo de reunión tiene un foco particular y los participantes son específicos.

Las reuniones N1 son las más internas y privadas de cada sector.

Tanto las reuniones N1 como las N2 son para aclarar dudas que pueden surgir en el sprint. Se revisa la lista de requerimientos para el próximo sprint y el reporte de bugs.

En la N3 se conversan temas de los equipos, cambios futuros, recursos, problemas del negocio, solicitudes de los órganos rectores.

Las reuniones N4 y N5 son a nivel gerencial y se realizan por fuera de la organización.

Las N4 suelen hacerse una vez al mes.

Las N5 son las más diplomáticas. Participan funcionarios, por ej.: secretario de hacienda Estas reuniones son bimestrales, pero también pueden surgir porque alguna autoridad la solicite.

Reunión de PMO

Esta reunión se realiza una vez por mes a lo largo de todo el proyecto. En esta reunión participan los PMO de cada área.

En la reunión se tratan temas de avance de los distintos negocios, temas relacionados entre los mismos, sincronización de tareas comunes, posibles alertas. Al tener presencia de los distintos sectores de la organización, la gerencia se apoya en estas para monitorear el conjunto de proyectos.

5.4 Desarrollo extremo

EL desarrollo extremo se da ante ciertos eventos o urgencias donde la organización se flexibiliza para atender la situación y poder resolverla en poco tiempo.

Cuando ingresa un pedido urgente, se deja de lado el modelado, estimaciones, planificaciones, priorizaciones. Es decir todo lo que se hace ante un requerimiento como se explicó en el punto anterior, y se pone foco en el desarrollo extremo. (Basándose en la Metodología XP).

Esto se da en la organización como un plan alternativo ante urgencias de desarrollo que tienen que estar operativas en un tiempo determinado.

Capítulo 6: Conclusiones

6.1 introducción

A lo largo del desarrollo de la tesina, se logró evidenciar, mediante un caso de aplicación, que no hay una única metodología utilizada para la implementación de un proyecto.

En el caso presentado, la organización fue tomando lo que consideró que podía adaptarse a su proyecto, teniendo en cuenta el tamaño de la organización, tiempos, costos y la meta a lograr, por lo que fue diseñando su propio modelo de desarrollo.

Dicho modelo fue cambiando, manteniendo aquellas prácticas con las cuales se obtuvieron buenos resultados y se incorporaron nuevas, reemplazando aquellas que no, y debían mejorar.

En proyecto de grandes dimensiones como este, donde la rotación de personal se realiza de forma permanente, es muy importante contar con documentación (minutas de talleres, casos de uso, reglas, etc.) que transfieren el conocimiento y poder consultar información de todo el sistema.

Por último se puede concluir que no hay una única forma de llevar adelante un proyecto, pero sí disminuir la probabilidad de error.

6.2 Convivencia de metodologías

El modelo que finalmente se utiliza en Esidif trabaja con un t*ablero de tareas* para ejecutar el proyecto, esto surge por la cantidad de requerimientos presentes en el mismo, la necesidad de tener centralizado en un solo lugar y, poder visualizar la planificación, al llegar una nueva funcionalidad. Cada tarjeta contiene un estado que puede ser, en proceso, finalizada, pendiente o suspendida. Esto se basa en la metodología ***Kanban***. Si bien la metodología plantea la organización de las tarjetas en columnas con diferentes estados, el proyecto lo adaptó según su necesidad, generando columnas por trimestre y poniendo el estado en cada tarjeta que representa una funcionalidad nueva.

Como segundo punto, el área de análisis trabaja siguiendo los lineamientos de la *metodología* ***RUP***. Relevan los requerimientos en los encuentros llamados Talleres, plasman la documentación del mismo en *artefactos* (documento de requerimientos, casos de uso, diagramas, etc.) que luego son consumidos por las Äreas de Desarrollo y Testing.

Para la gestión de los procesos se utiliza ***SCRUM.*** Se organiza el trabajo en equipo, los cuales participan en la toma de decisiones, como por ejemplo estimación de tiempos. Los requerimientos se agrupan por sprint, y al final se genera un incremento de funcionalidad del sistema. Durante dicho sprint se realizan las reuniones diarias que ayudan a resolver dudas funcionales y posibles complicaciones.

Si bien al principio del proyecto se utilizó la metodología ***RUP*** como única metodología de implementación del proyecto, por el crecimiento del sistema y la demanda de los usuarios, en cuanto a nuevas funcionalidades y tiempos de entrega, se incorporó ***Scrum***. Esto permitió poder adaptarse a cambios frecuentes sin tener un costo alto.

Por el tamaño del sistema, surgen muchos requerimientos por parte de los usuarios, nuevas funcionalidades, tareas técnicas (por ej: actualizaciones de la versión de la BBDD) y nuevos proyectos satélites. Un sinfín de cosas que requieren ser centralizadas para poder lograr una estimación efectiva de tiempo. Por todo esto, se incorporó el tablero Trello basado en la metodología ***Kanban***. De esta manera se logra tener todas las tareas en un solo lugar consiguiendo una mejor organización y estimación de los tiempos.

Para las situaciones en que ingresa una solicitud con cierta urgencia, la organización incorporó la metodología ***XP***, como una salida a la atención de estos requerimientos. Esta forma de trabajo permite enfocarse en el requerimiento y responder en los tiempos solicitados

Como se demostró a lo largo de la tesina, es posible la convivencia de metodologías, cada una con sus ventajas y desventajas, tomando lo que el proyecto necesita de cada una, adaptarlo y obtener resultados óptimos.

6.3 Trabajos futuros

A partir del trabajo realizado, visualizando la aplicación de las distintas metodologías se analizará la aplicación de la metodología ***Less*** (Scrum a gran escala - Large Scale Scrum) encontrando el desafío de aplicarla a una organización con varios equipos.

Capítulo 7: Referencias bibliográficas

[1] Kanban&Scrum , Henrik Kniberg & Mattias Skarin, Managing Editor: Diana Plesa 2010

[2] Juan Palacio, Flexibilidad con Scrum., Edición Octubre - Noviembre 2007.

[3] Scrum y Xp desde las trincheras, Henrik Kniberg

[4] AgileEstimatingAndPlanning, Mike Cohn

[5] Agile software requirements agile Denver, Dean Leffingwell

[6] https://metodoss.com/metodologia-rup/

[7] http://ingenieriadesoftware.mex.tl/63758_aup.html

[8] El proceso unificado de desarrollo de software, Ivar Jacobson – Grady Booch –James Rumbaugh, Addison Wesley , 2000

[9] Ingeniería de software, Ian Sommerville, Novena Edición, Addison Wesley, México 2011

[10] José H. Canós, Patricio Letelier y Mª Carmen Penadés, Metodologías Ágiles en el Desarrollo de Software DSIC. Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia.

[11] https://proyectosagiles.org

[12] Metodologías ágiles y desarrollo basado en conocimiento

[13] http://www.mountaingoatsoftware.com/scrum

[14] https://ronjeffries.com/xprog/what-is-extreme-programming/

[15] https://blog.trello.com/es/metodologia-kanban

[16] Guía de los Fundamentos para la Dirección de Proyectos (PMBOK) Cuarta edición, Junio 2009. LisaC