UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Titulo: PDSM, Proceso de Desarrollo de Software Mixto, combinando RUP y SCRUM

Autores: Mariani, María Florencia – Okabe, Evangelina
Director: Dra. Claudia Pons

Carrera: Licenciatura en Informática

Resumen

Existen distintas metodologías que intentan ayudar a las empresas de desarrollo de software a llevar a cabo proyectos exitosos, pero la realidad es que ninguna metodología es 100% ideal para obtener un producto rentable, en el tiempo estipulado y con la calidad esperada. A la hora de elegir una metodología se deben tener en cuenta varios factores, tales como: capacidades del equipo de trabajo, madurez de los requerimientos, tiempo con el que se cuenta para el desarrollo, interacción y relación con el cliente.

La experiencia indica que es mejor partir de una metodología sencilla, a la cual se le puede agregar lo necesario, antes que partir de una metodología muy completa, a la que se le deba quitar lo no necesario, ya que esto pocas veces se hace en la práctica. Basándonos en esto último, se tomaron los mejores conceptos de la metodología pesada rup y de la metodología ágil scrum para definir un nuevo proceso que ayude al desarrollo de software.

Palabras Claves

Las siguientes son las palabras claves en las cuales se encuadra el presente trabajo:
- Procesos
- Metodologías ágiles
- Metodologías pesadas
- Rup
- Scrum

Trabajos Realizados

El trabajo se basa en tomar los mejores conceptos de rup y scrum para definir un nuevo proceso que ayude al desarrollo de software. Para esto, destacamos las tareas más importantes realizadas:
- Estudio y análisis de los procesos rup y scrum.
- Definición de un modelo de proyecto típico.
- Análisis de los principios de rup y scrum.
- Análisis de los problemas típicos (sobre el modelo definido anteriormente).
- Definición de un proceso que resuelva los problemas detectados.
- Aplicación del nuevo proceso al proyecto definido.

Conclusiones

Nada asegura el éxito de la ejecución de proyectos, pero los procesos disminuyen la probabilidad de fracaso. Lo más importante, más allá de conocer técnicas y herramientas, es la “experiencia”.
No significa que el conocimiento, las habilidades y los procesos descriptos deban aplicarse siempre de manera uniforme en todos los proyectos, la definición de procesos no es estática, se debe evaluar en cada caso, qué grupo de procesos se aplicarán, quién y con qué grado de rigor.

Trabajos Futuros

Entre los trabajos futuros se destacan:

- Analizar la posibilidad de desarrollar una aplicación para apoyo del proceso definido.
- Continuar con la mejora de los procesos con el objetivo de aumentar la calidad del producto.
- Implementar herramientas de apoyo para los procesos definidos (herramientas de tracking, administración de casos de pruebas, etc.)
- Aplicar los procesos a otros tipos de proyectos y analizar los resultados.
- Tomar como modelo otros procesos tanto formales como agiles y plantear alternativas.

Fecha de la presentación: Noviembre 2010

PDSM: Desarrollo de Software Mixto,

combinando RUP y SCRUM

TESINA DE GRADO

RESUMEN

La experiencia de las empresas dedicadas al desarrollo de software indica que los proyectos exitosos son aquellos que siguen un proceso que permite administrar, organizar y controlar el proyecto. Por otro lado, en los últimos años, se ha destacado en la industria del software que los requerimientos de los clientes son cambiantes y que si no nos adaptamos rápidamente, se corre el riego de resolver el problema equivocado.

Ahora bien, existen distintas metodologías que intentan ayudar a las empresas de desarrollo de software a llevar a cabo proyectos exitosos, pero la realidad es que ninguna metodología es 100% ideal para obtener un producto rentable, en el tiempo estipulado y con la calidad esperada. A la hora de elegir una metodología se deben tener en cuenta varios factores, tales como:

- Capacidades del equipo de trabajo
- Madurez de los requerimientos
- Tiempo con el que se cuenta para el desarrollo
- Interacción y relación con el cliente

La experiencia indica que es mejor partir de una metodología sencilla, a la cual se le puede agregar lo necesario, antes que partir de una metodología muy completa, a la que se le deba quitar lo no necesario, ya que esto pocas veces se hace en la práctica. Basándonos en esto último, se tomaron los mejores conceptos de la metodología pesada RUP y de la metodología ágil SCRUM para definir un nuevo proceso que ayude al desarrollo de software.

El resultado de este trabajo es un nuevo proceso de desarrollo de software que se especificó teniendo en cuenta los principios de RUP y SCRUM y se adapta tanto a proyectos pequeños como medianos y proyectos nuevos y de mejoras y mantenimiento.

Dirección: Dra. Claudia Pons

Autores: Mariani, M. Florencia – Okabe, Evangelina

TABLA DE CONTENIDOS

1. OBJETIVOS Y ESTRUCTURA

1.1 OBJETIVOS Y ALCANCE

La mayoría de los proyectos de desarrollo de software en la Argentina son de tamaño pequeños y medianos, con una duración de entre 2 y 10 meses, en los cuales la definición de los mismos es pobre, el cliente nunca tiene en claro lo que realmente necesita y en la mayoría de los casos los tiempos de desarrollo son limitados.

Por otro lado existen distintas metodologías que intentan ayudar a las empresas de desarrollo a llevar a cabo proyectos exitosos, pero la realidad es que ninguna metodología es 100% ideal para obtener un producto rentable, en el tiempo estipulado y con la calidad deseada. A la hora de elegir una metodología se deben tener en cuenta varios factores tales como, las características del proyecto, el cliente, el equipo que llevará adelante el proyecto, etc. El mercado demanda alta calidad, diferenciación y bajo costo donde se requiere velocidad y flexibilidad. La clave está en la estrategia que se utiliza para gestionar el desarrollo.

Ante este escenario, el proyecto consiste en definir un proceso, tomando prácticas tanto de la metodología formal o tradicional como de la metodología ágil, particularmente nos basaremos en los procesos RUP y SCRUM respectivamente, para empresas desarrolladoras de software, que le permita llevar a cabo proyectos de manera organizada y efectiva logrando así rentabilidad en los proyectos y calidad en los productos.

El proyecto propuesto consta de las siguientes partes:

- Estudio y análisis de los procesos RUP y SCRUM. Los procesos fueron elegidos porque son los que se siguen en las empresas donde trabajamos actualmente y porque son procesos que han sido adoptados mayoritariamente en la industria del software en el mundo.
- Definición de un modelo de proyecto típico.
- Análisis de los principios de los procesos RUP y SCRUM.
- Análisis de los problemas típicos (sobre el modelo definido anteriormente) aplicando los procesos definidos en las empresas donde trabajamos. Nos basaremos en los procesos que siguen los principios, aclaramos esto porque existen procesos que no responden a ningún principio, son procesos que se definieron para certificar una norma tales como CMMI e ISO.
- Definición de un proceso que resuelva los problemas encontrados en el punto anterior. Consideramos que los procesos no son excluyentes, es decir, no porque decido usar RUP no puedo

usar SCRUM en un mismo proyecto y viceversa. La experiencia nos conduce a tomar lo mejor de cada uno.

- Aplicación del nuevo proceso al proyecto/modelo definido anteriormente.
- Análisis de los resultados obtenidos.
- Análisis de la posibilidad de extender algún framework para apoyo del proceso de desarrollo, por ejemplo el OpenUP.

1.2 ESTRUCTURA DE LA TESIS

El documento está organizado en 8 capítulos que abarcan la totalidad del trabajo de tesis.

• El capítulo **1. Objetivos y estructura** (el presente) sintetiza los objetivos de la tesis, y de qué manera se encuentra organizado el material de la misma.
• El capítulo **2. Motivación** presenta un panorama de la situación actual, mostrando el contexto que da origen al trabajo de tesis.
• El capítulo **3. Estudio Previo** contiene el estudio y análisis de los procesos RUP y SCRUM que dan soporte al nuevo proceso planteado en este trabajo.
• El capítulo **4. Proyectos típicos,** presenta una breve reseña de los proyectos de desarrollo de software típicos, para los cuales definiremos el nuevo proceso.
• El capítulo **5. Análisis de Problemas Típicos,** contiene el análisis de los principales problemas que se presentan en los proyectos de desarrollo de software.
• El capítulo **6. Definición del Proceso PDSM,** define un nuevo proceso basado en las mejores prácticas de RUP y SCRUM
• El capítulo **7. Aplicación del Proceso,** aplica el proceso definido en el capítulo anterior a un proyecto particular.
• El capítulo **8. Resultados Obtenidos** contiene una descripción de los resultados.
• El capítulo **9. Trabajo Futuro** contiene las futuras líneas de trabajo a seguir por aquellos interesados en el tema.
• El capítulo **10. Trabajos Relacionados** contiende una serie de trabajos relacionados con esta tesis.
• El capítulo **11. Conclusiones** contiene las conclusiones obtenidas luego de finalizado el trabajo de tesis.
• El capítulo **12. Apéndices**
• El capítulo **13. Glosario de Términos**
• El capítulo **14. Agradecimientos** menciona a aquellas personas que hicieron posible este trabajo.
• El capítulo **15. Bibliografía** contiene las referencias bibliográficas.

2. MOTIVACION

La aplicación de un proceso de desarrollo de software puede mejorar la efectividad y eficiencia de una organización. Decimos **puede** porque a veces una mala definición del proceso o una mala elección y aplicación del mismo resulta totalmente contraproducente. Toda empresa de desarrollo debería basarse en algún proceso para llevar a cabo un proyecto de manera exitosa, si no se sigue ninguno es muy difícil el éxito del mismo. Hay organizaciones que no se basan en ningún proceso, en el que cada uno hace lo que desea y le parece, hay otras que lo hacen inconscientemente, es decir, el proceso no está formalmente definido por la organización pero se siguen algunas prácticas de algún proceso ya definido para llevar a cabo una tarea determinada, finalmente hay otros que sí siguen un proceso definido.

Actualmente está pre-establecida o infundada la idea que en proyectos pequeños se suelen seguir procesos livianos, alguna metodología ágil. En estas metodologías no se documentan, hay poco de análisis y diseño y de manera informal y transitoria, y el código fuente es el centro ante cualquier otra actividad en el proyecto. Los proyectos grandes de docenas o cientos de personas, se basan en procesos más formales donde se documentan, las tareas de análisis y diseño involucran a personas no desarrolladoras, se tienen en cuenta reuniones, presentaciones, documentos y otros artefactos y en el que el código fuente es considerado un artefacto más.

Esto no significa que las metodologías formales y las metodologías ágiles sean opuestas o que una sea mejor que la otra. Todo problema y todo proyecto individual requiere de un proceso que se adapte a sus propias necesidades y sea acorde a su propio contexto.

En la actualidad existen aplicaciones en las que no hace falta aplicar RUP o SCRUM al 100%, las organizaciones deberían conocer ambas metodologías, procesos de cada metodología y de acuerdo a las características de sus proyectos decidir cuál proceso seguir, si la formal, liviana o mixta.

El objetivo de definir un proceso mixto es poder contar con un proceso cuyas prácticas sean realmente efectivas, dado que hay muchas prácticas de las metodologías formales que no suman en el proyecto sino que restan y hay muchas prácticas de las metodologías ágiles que no son suficientes. En nuestra empresa particularmente se siguen procesos de ambas metodologías en diferentes proyectos, algunos proyectos con sólo RUP, otros con sólo SCRUM y otros RUP + SCRUM. La idea es basarnos en la experiencia de cada uno, analizando las prácticas positivas y negativas y en base a ello plantear un proceso que sea realmente efectivo para la empresa. La idea es poder definir un proceso que realmente ayude a la empresa sin generar trabajo extra a los involucrados, dado que la mayoría de las veces los procesos fracasan porque generan trabajo extra o no deseado en las personas que deben seguirlo. Esto muchas veces se da también porque se delega en un grupo de personas la definición del proceso en el que la mayoría de las veces dichas personas

nunca participaron en lo que están definiendo, por ejemplo, al desarrollador le definen tareas sin haber ellos desarrollado antes. El proyecto de definición de un proceso es como un proyecto de desarrollo, deberían participar personas con diferentes perfiles para poder definir en conjunto las prácticas que ayudan a cada uno de los involucrados. Quien mejor que el que lo hace conoce lo que le ayuda y lo que no.

Si una organización sigue un proceso efectivo para la construcción de su producto, el resultado será sin lugar a dudas un producto de buena calidad incrementando así su nivel de competitividad en el mercado.

3. ESTUDIO PREVIO

3.1 RUP

3.1.1 INTRODUCCION

Muchas compañías de software todavía utilizan el proceso en cascada para el desarrollo de proyectos, donde se completa cada fase en una secuencia estricta: requerimientos, análisis, diseño, implementación y prueba.

En contraposición, RUP utiliza un enfoque iterativo, esto es, una secuencia de iteraciones incrementales. Cada iteración incluye algunas o muchas de las disciplinas de desarrollo (requerimientos, análisis, diseño, implementación, etc.). Cada iteración tiene un conjunto de objetivos bien definidos y produce una implementación parcial del sistema final. Cada sucesiva iteración se construye sobre las iteraciones anteriores para refinar el sistema hasta que el producto final este completo.

A través de este enfoque iterativo que generan entregables ejecutables, se logra detectar en forma temprana los desajustes e inconsistencias entre los requerimientos, el diseño, el desarrollo y la implementación del sistema, manteniendo al equipo de desarrollo focalizado en producir resultados. Además, este enfoque ayuda a atacar los riesgos mediante la producción de entregas parciales progresivas y frecuentes que permiten la opinión e involucramiento del usuario.

RUP es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable. Es un proceso práctico. RUP unifica todo el equipo de desarrollo de software y optimiza su comunicación proveyendo a cada miembro de una aproximación al desarrollo de software con una base de conocimiento on-line personalizable de acuerdo a las necesidades específicas del proyecto. La base de conocimiento unifica aun más al equipo identificando y asignando responsabilidades, artefactos y tareas de forma que cada miembro del equipo comprenda su contribución al proyecto. Unificando al equipo, se simplifica la comunicación, asegurando la asignación de recursos en forma eficiente, la entrega de artefactos correctos y el cumplimiento de los tiempos.

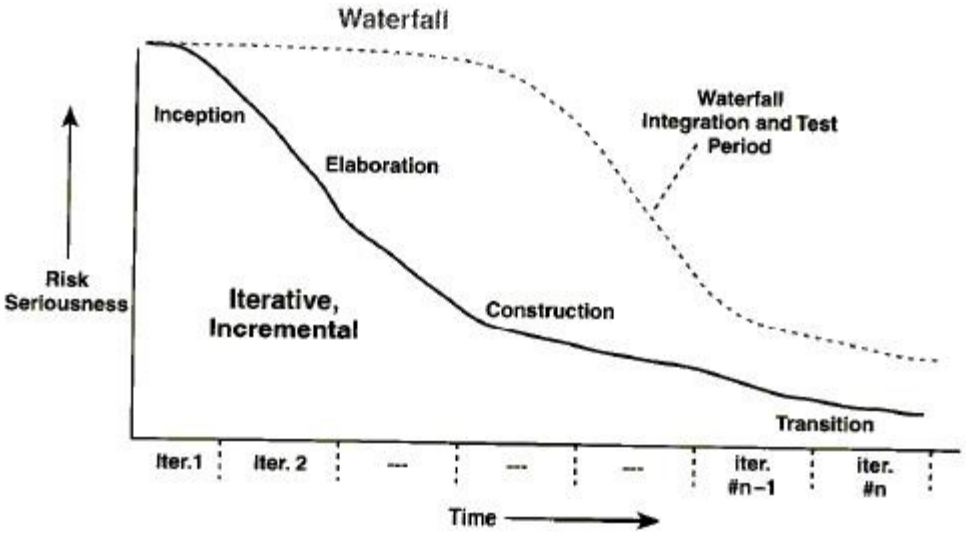
Además de todo esto, RUP mantiene al equipo enfocado en producir incrementalmente software operativo a tiempo, con las características y la calidad requerida.

3.1.2 PRINCIPIOS ESENCIALES

- *Atacar los principales riesgos de manera temprana y continua*

Identificar y atacar los principales riesgos tan pronto como sea posible, en lugar de hacerlo más tarde (riesgos de negocio, riesgos técnicos). RUP proporciona un enfoque estructurado para hacer frente a los principales riesgos primero, lo cual disminuye los costos generales y le permite hacer más temprana, más realistas y más precisas las estimaciones de cuánto tiempo se necesitará para completar el proyecto. La gestión de riesgos es un proceso dinámico y permanente.

Manejar los riesgos del proyecto implica: identificarlos, establecer la probabilidad de ocurrencia del mismo, el impacto que tendrán en el proyecto y que acción se deberá tomar para mitigar su ocurrencia. Una vez establecidos los riesgos iniciales del proyecto se debe realizar la tarea de monitoreo de los mismos, habrá riesgos que dejarán de serlo y otros nuevos aparecerán. En caso de que existan riesgos con alto impacto es posible tomar acciones que mitiguen su ocurrencia.



Perfil de Reducción de Riesgos para desarrollo en Cascada e Iterativo

Uno de los principales objetivos del desarrollo iterativo es reducir los riesgos desde el principio. Esto se hace mediante el análisis, la priorización y atacando los principales riesgos primero, en cada iteración.

Experiencia personal: Uno de los principales requisitos de los sistemas en los que trabajamos (Sistema de ruteo de órdenes para la Bolsa Mexicana de Valores) es que la aplicación tenga un tiempo de respuesta rápido y soporte volumen. En una de las últimas implementaciones se retrasaron las pruebas de stress test en los ambientes productivos y por cuestiones de tiempo, nunca se hicieron. Como consecuencia de esto, el día de la salida en producción la aplicación funcionaba muy lenta, esto siguió así por varios días, el cliente no permitía realizar el monitoreo de los equipos para poder detectar la causa, esto tuvo como final que el cliente quitara el sistema de producción a pocas semanas de su instalación.

Muchas veces las personas a testear necesitan ser capacitadas funcionalmente. Existen proyectos que son muy difíciles de testear dado que su lógica de negocio es muy compleja.

- *Asegurar el valor entregado al cliente.*

Documentar los requerimientos de forma tal que sean fáciles de ser entendidos por los clientes y trabajar sobre los mismos a través de todas las fases (diseño, implementación y prueba).

Los casos de uso hacen que sea fácil documentar los requerimientos funcionales de los usuarios; han demostrado ser una manera excelente de capturar los requerimientos funcionales y asegurarse que direccionan el diseño, la implementación y las pruebas del sistema, logrando así que este satisfaga las necesidades del usuario.

Experiencia personal: Una forma de reducir la probabilidad de que el producto a desarrollar no sea lo que el cliente está esperando, es realizar una especificación detallada que contenga maquetas de las pantallas y la interacción del usuario con el sistema de manera de demostrar su uso. Con esto, el usuario detecta los puntos débiles del sistema y mediante sucesivas iteraciones se refina el documento obteniendo una especificación aprobada por el/los usuarios.

- *Mantener el foco en el software ejecutable.*

Los documentos, diseños y planes son importantes, pero no reflejan el verdadero progreso del proyecto y su importancia pasa a ser secundaria comparado con el código fuente. Un código que compila y pasa las pruebas de manera exitosa es la mejor evidencia del progreso del proyecto.

- *Administrar los cambios continuamente en el proyecto.*

Las aplicaciones de hoy día son demasiado complejas para pretender tener los requerimientos, diseños e implementaciones correctas desde un primer momento o en las fases iniciales. Esto significa que debemos adaptarnos a los cambios constantemente.

Para optimizar la estrategia de manejo de cambios, se necesita entender el costo relativo de introducir diferentes tipos de cambios en los diferentes estados del ciclo de vida del proyecto.

Los cambios se pueden agrupar en cuatro categorías:

- Costo de cambio a una solución de negocio
- Costo de cambio en la arquitectura
- Costo de cambio de diseño e implementación
- Costo de cambio en el alcance

Tener en cuenta que el costo de cambios aumenta cuanto más avanzado está el proyecto y diferentes tipos de cambios tienen diferentes perfiles de costos. RUP tiene fases que se han establecido para reducir al mínimo el costo de cambio, mientras se reduce al mínimo la capacidad para permitir el cambio. Esta es la razón por la que RUP fuerza a tener una visión general al final de la fase de Inicio, una referencia de la arquitectura al final de la fase de Elaboración, terminar los desarrollos al final de la fase de Construcción. El uso de las herramientas adecuadas puede desempeñar un poderoso papel en la gestión del cambio y reducir al mínimo su costo.

Experiencia personal: Aun habiendo documentado correctamente los requerimientos, desarrollados los casos de uso, contando con la aprobación del cliente, habiendo realizado el diseñado acorde a todo lo especificado y teniendo validado el mismo; incluso habiendo construido en base a todo esto y teniendo las pruebas automáticas que cubra la mayoría de la funcionalidad, el cambio puede existir, y podrá ocurrir por diferentes razones, como por ejemplo: cambios en el negocio (reglas, regulaciones), cambios en el ambiente (cambio de servidor), cambios en la estructura del cliente; hasta incluso cambios que no se habían tenido en cuenta en el relevamiento inicial.

El haber realizado todos los documentos y contar con pruebas automáticas facilitan la introducción de cambios y los hace más flexibles.

- *Obtener una arquitectura funcional temprana*

Muchos de los riesgos de un proyecto pueden ser mitigados realizando un buen diseño, implementación y prueba de la arquitectura en las etapas iniciales del proyecto. Establecer una arquitectura estable temprana, también facilitará la comunicación y localización del impacto de los cambios.

La arquitectura es la estructura del esqueleto del sistema. Diseñando, implementando y probando la misma en una etapa temprana del proyecto, aborda los principales riesgos y hace más fácil ampliar el equipo de desarrollo e introducir miembros menos experimentados. Finalmente, ya que la arquitectura, define si el sistema es construido en bloques o componentes, le permite evaluar con mayor exactitud el esfuerzo necesario para completar el proyecto.

Experiencia personal: El uso de una arquitectura común dentro de la organización asegura una mínima curva de aprendizaje al cambiar los integrantes de un proyecto a otro. También permite probar la arquitectura en diferentes ambientes y entornos.

Una arquitectura que nos guíe en el desarrollo nos dará una base sólida para la construcción de cualquier proyecto de software, asegurándonos la homogeneidad y el control al momento de introducir nuevos componentes.

En nuestros proyectos siempre que se empieza un proyecto grande, antes, se realiza una prueba de concepto de los módulos importantes para con eso poder analizar la viabilidad y el esfuerzo del mismo. Un ejemplo de esto, son las interacciones con otros sistemas dado, que tienen sus propias interfaces de comunicación (módulo de facturación, módulo que se comunica con el teléfono de los usuarios de call center, etc.).

- *Construir el sistema orientado a componentes*

El proceso de software debe focalizarse en el desarrollo temprano de una arquitectura robusta ejecutable, antes de comprometer recursos para el desarrollo en gran escala. RUP describe como diseñar una arquitectura flexible, que se acomode a los cambios intuitivamente y promueve una efectiva reutilización de software. Soporta el desarrollo de software basado en componentes: módulos no triviales que completan una función clara. Además, provee un enfoque sistemático para definir una arquitectura utilizando componentes nuevos y preexistentes.

Las aplicaciones construidas con componentes son más resistentes al cambio y pueden reducir radicalmente el costo de mantenimiento del sistema. Los componentes facilitan el re-uso y permiten construir más rápido aplicaciones de alta calidad que usando descomposición funcional.

Uno de los aspectos funcionales de la descomposición es que se separan los datos de las funciones. Uno de los inconvenientes de esta separación es que mantener o modificar el sistema es más caro. Por ejemplo, un cambio en cómo se almacenan los datos puede afectar a varias funciones, y generalmente es difícil saber que función puede verse afectada a través de un sistema.

Experiencia personal: La separación en componentes nos ayuda a reducir el impacto del cambio y también a conocer ciertamente que funciones del sistema están involucradas si se encontrase un error. Al contar con una fachada para cada componente y conocer ciertamente quienes son los clientes, es muy sencillo encontrar y documentar quienes serán afectados por el cambio. Así mismo podemos documentar y planificar las pruebas necesarias, acotando el alcance al mínimo y disminuyendo los costos asociados a la inclusión de dicho cambio.

- *Trabajar juntos como un equipo.*

El desarrollo de software se ha convertido en un deporte de equipo y el enfoque iterativo enfatiza la importancia de la buena comunicación y el espíritu de equipo, donde cada miembro del mismo se siente responsable por el producto final terminado.

Tradicionalmente, las compañías tienen una organización funcional: Todos los analistas están en un grupo, los diseñadores en otro y los testers en otro diferente. Esta estructura organizacional construye centros de competencia y el inconveniente es que la comunicación entre los tres grupos se ve comprometida. Incluso se ha empeorado, las organizaciones funcionales son matrices funcionales, donde, por ejemplo un analista trabaja en varios proyectos diferentes al mismo tiempo y no es realmente parte integrante de ningún equipo. Este tipo de organizaciones pueden ser aceptables en proyectos de larga duración; pero si estamos en un desarrollo iterativo con un proyecto corto de nueve meses, por ejemplo, se necesita mayor comunicación entre equipos.

Conformar un equipo donde cada miembro es responsable por el todo y no por una parte ayuda al espíritu del mismo, donde todos los integrantes son responsables por el todo y no existen "dueños" de las partes del proyecto.

Cuando se trabaja con un equipo pequeño es más sencillo implementar esta política, en cambio cuando se tiene un equipo grande de desarrollo o el proyecto en si es muy grande se hace más difícil lograr que todo el conocimiento fluya de manera horizontal para que todos los integrantes del proyecto sean capaces de realizar cualquier tarea.

Recordemos esto:

- Organizar proyectos en torno a cruces de equipos funcionales conteniendo analistas, desarrolladores y testers.
- Proporcionar a los equipos las herramientas necesarias para una colaboración efectiva entre las funciones.
- Asegurar que los miembros de equipo tengan las actitudes de "Voy a hacer lo que se necesite para obtener un software de alta calidad" en lugar de "Voy a hacer mi pequeña parte del ciclo de vida del proyecto".

Experiencia personal: en uno de los proyectos de nuestras empresas se implementó este principio como prueba piloto y resultó todo un éxito, fue una experiencia increíble.

- *Hacer de la Calidad una forma de vida, no una ocurrencia tardía.*

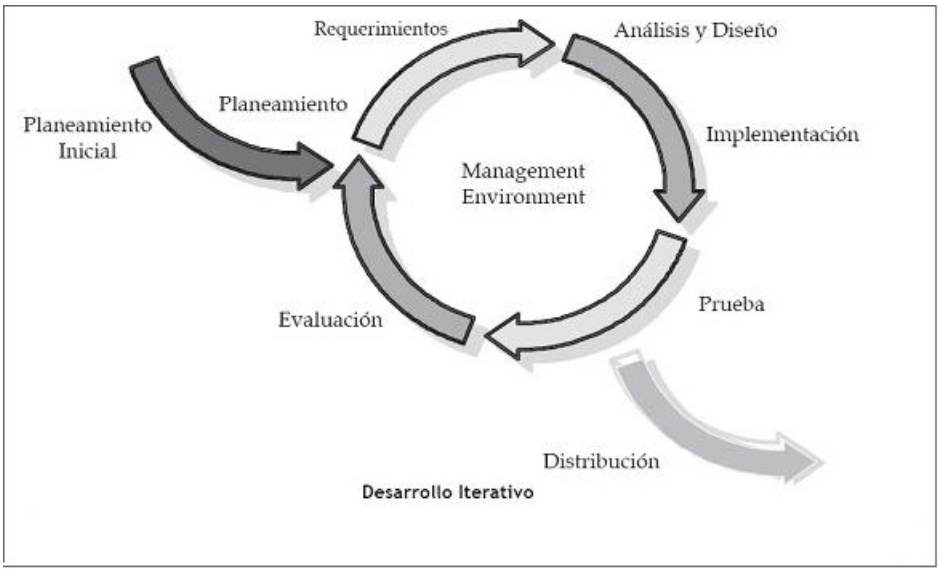
Garantizar una alta calidad implica más que sólo el equipo de pruebas. Se trata de todos los miembros del equipo y todas las partes del ciclo de vida. Un enfoque iterativo se centra en la primera prueba y pruebas de automatización para aumentar la eficacia de las pruebas de regresión, lo que reduce el número de defectos.

Uno de los mayores beneficios del desarrollo iterativo es que permite iniciar las pruebas mucho más temprano que los desarrollos en cascada. En la primera fase, *Inicio,* se pueden testear las funcionalidades capturadas en el prototipo, proveyendo además ida y vuelta de los casos de uso. En la segunda fase, *Elaboración*, el software ejecutable está arriba y corriendo, implementando la arquitectura. Esto significa que se puede comenzar a probar que la arquitectura realmente funciona.

Los desarrollos iterativos no solo permiten una prueba temprana, sino que también fuerza a hacerlo más frecuente. Por un lado, esto es bueno porque se puede ir probando el software existente para asegurar que

no se introdujeron nuevos errores. Por otro lado, el inconveniente es que la prueba de regresión puede ser más cara. Para minimizar los costos, intente automatizar lo más posible las pruebas. Esto frecuentemente reduce radicalmente los costos.

3.1.3 RUP Y EL DESARROLLO ITERATIVO



Desarrollo iterativo en RUP

RUP promueve un enfoque iterativo, en cada iteración se hace una pequeña parte de requerimientos, análisis, diseño, implementación y prueba. Cada iteración se basa en el trabajo de la iteración previa para producir un ejecutable que es un paso más hacia el proceso final.

El enfoque iterativo tiene algunas razones que hace que sea superior al enfoque en cascada:

- *Se ajusta mejor a los cambios de requerimientos*, los cambios pueden ser tecnológicos o derivados del usuario. Los cambios detectados sobre el final del proyecto inevitablemente provocan que el proyecto se retrase, que el cliente quede insatisfecho y disconformidad en el equipo de desarrollo. En cambio, un proceso iterativo como tiene como objetivo entregar una parte de la aplicación funcionando en cada iteración (pocas semanas) se centra en los requerimientos más importantes.

- *La integración no es un 'big bang' al final del proyecto*, integrar por partes es más fácil que integrar el total de la aplicación.

- *Los riesgos pueden ser detectados o tratados durante las primeras iteraciones*, la integración temprana ayuda a poder llevar a cabo esto.

- En ciertas ocasiones, se pueden utilizar los despliegues de las primeras iteraciones como un rápido entregable que permita competir con otras aplicaciones de la competencia, o bien pueden empezar a generar beneficios u otorgar valor agregado a la empresa.

- *Facilita la reutilización*, permite al arquitecto identificar las partes que tienen en común las iteraciones y así poder desarrollar algún componente o código en común que las siguientes iteraciones pueden utilizar.

- *Los defectos pueden ser encontrados y corregidos en cada una de las iteraciones*, permitiendo desarrollar una aplicación robusta y de alta calidad. No es lo mismo detectar errores al inicio del proyecto que al final, y un problema del tipo performance es mejor descubrirlo cuando aún se tiene tiempo de solucionarlo en lugar de generar pánico por tener que salir a producción.

- *Permite aprovechar más a los recursos*, en el modelo en cascada como es por etapas, la secuencia es la siguiente, los analistas le envían los requerimientos a los diseñadores, ellos envían sus diseños a los programadores, estos envían los componentes a los integradores y finalmente se envía el producto a los testers. De esta manera como no se paralelizan las tareas se retrasa mucho el proyecto y en cada etapa van perdiendo participación muchos de los recursos. En un proceso iterativo los recursos pueden cumplir varios roles y pueden trabajar todos los roles en una misma iteración.

- *El equipo de desarrollo aprende a lo largo del proyecto*, los miembros del equipo tienen la oportunidad en cada iteración de analizar los errores de la iteración anterior y corregir los mismos.

El proceso de desarrollo en sí mismo es mejorado y refinado a lo largo del proyecto, al final de cada iteración se analiza el estado del proyecto, del producto y qué cosas pueden ser mejoradas del proceso.

3.1.4 RUP – PROCESO DE INGENIERIA DE SOFTWARE

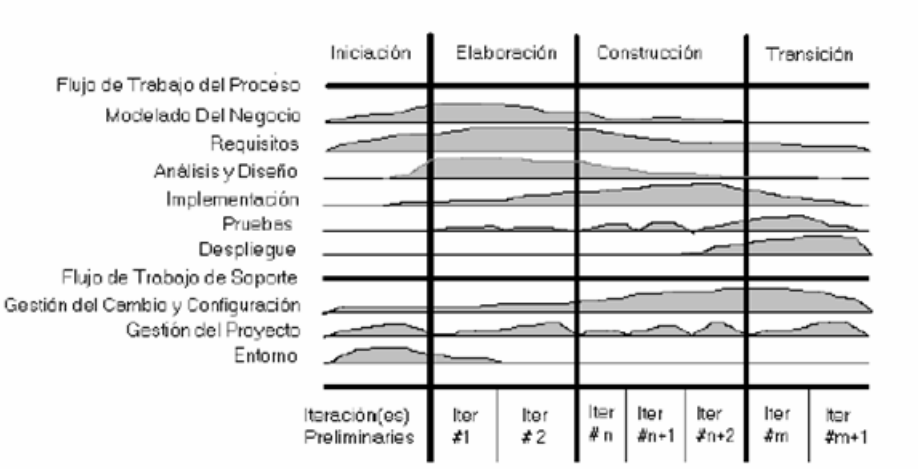
El proceso RUP tiene dos estructuras o dimensiones:

Estructura dinámica: Describe el proceso expresado en términos de ciclos, fases, iteraciones e hitos, desarrollados durante el ciclo de vida del proyecto.

El ciclo de vida RUP es una implementación del desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades.

Estructura estática: Describe el proceso como elementos - actividades, disciplinas, artefactos y roles- que lógicamente se agrupan en las disciplinas básicas del proceso.



Dos dimensiones de RUP

RUP es organizado a través de dos dimensiones. El aspecto dinámico (Horizontal) expresada en ciclos, fases, iteraciones y hitos; el aspecto estático (vertical) expresa actividades, disciplinas, artefactos y roles.

3.1.4.1 ESTRUCTURA DINAMICA

La estructura dinámica de RUP divide un proyecto en cuatro fases: *Inicio, Elaboración, Construcción* y *Transición*. Cada una de las fases de RUP tiene un conjunto definido de objetivos. Se deben utilizar dichos objetivos como guía para decidir qué actividades realizar y qué artefactos producir.

Cada fase contiene una o más iteraciones. Existen tantas versiones como sea necesario para hacer frente a los objetivos de esa etapa (lo suficiente, no más). Si el objetivo no puede ser cumplido dentro de esa fase, será agregado en otra fase. En cada iteración se debe poner foco solamente en el objetivo que se quiere alcanzar en cada fase. Por ejemplo, en la fase de incepción no hay que profundizar en los requerimientos, alcanza con entender el problema a alto nivel y definir el alcance.

Fase de Inicio

Las primeras iteraciones (en las fases de inicio y elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de riesgos críticos, y el establecimiento de una línea base.

Durante la fase de inicio las iteraciones hacen poner mayor énfasis en las actividades de modelado del negocio y de requerimientos.

Modelado del negocio

En esta fase el equipo se familiarizará más al funcionamiento de la empresa, sobre conocer sus procesos.

- Entender la estructura y la dinámica de la organización para la cual el sistema va a ser desarrollado.
- Entender el problema actual de la organización objetivo e identificar potenciales mejoras.
- Asegurar que clientes, usuarios finales y desarrolladores tengan un entendimiento común de la organización objetivo.

Requerimientos

En esta línea los requerimientos son el contrato que se debe cumplir, de modo que los usuarios finales deben comprender y aceptar los requerimientos que especificamos.

- Establecer y mantener un acuerdo entre clientes y otras partes involucradas sobre lo que el sistema podría hacer.
- Proveer a los desarrolladores un mejor entendimiento de los requerimientos del sistema.
- Definir el ámbito del sistema.
- Proveer una base para estimar costos y tiempo de desarrollo del sistema.
- Definir una interfaz de usuario para el sistema, enfocadas a las necesidades y metas del usuario.

Fase de Elaboración

En la fase de elaboración, las iteraciones se orientan al desarrollo de la línea base de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocio (refinamientos), análisis, diseño y una parte de implementación orientado a la línea base de la arquitectura.

Análisis y Diseño

En esta actividad se especifican los requerimientos y se describen sobre cómo se van implementar en el sistema.

- ✓Transformar los requerimientos al diseño del sistema.
- ✓Desarrollar una arquitectura para el sistema.
- ✓Adaptar el diseño para que sea consistente con el entorno de implementación.

Fase de Construcción

En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones.

Para cada iteración se seleccionan algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la nueva implementación de la nueva versión del producto.

Implementación

Se implementan las clases y objetos en archivos fuentes, binarios, ejecutables y demás. El resultado final es un sistema ejecutable.

- Planificar qué subsistemas deben ser y en qué orden deben ser integrados, formando un Plan de Integración.
- Cada implementador decide en qué orden implementa los elementos del subsistema.
- Si encuentra errores de diseño, los notifica.
- Se integra el sistema siguiendo el plan.

Pruebas

Este flujo de trabajo es el encargado de evaluar la calidad del producto que estamos desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

- Encontrar y documentar defectos del software.
- Generalmente asesora sobre la calidad del software percibida.
- Provee la validación de los supuestos realizados en el diseño y especificación de requerimientos por medio de demostraciones concretas.
- Verificar las funciones del producto de software según lo diseñado.
- Verificar que los requerimientos tengan su apropiada implementación.

Fase de Transición

Esta fase tiene como objetivo producir con éxito distribuciones de producto y distribuirlos a los usuarios.

En la fase de Transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

Las actividades incluyen:

- Probar el producto en su entorno de ejecución final.
- Empaquetar el software para su distribución.
- Distribuir el software.
- Instalar el software.
- Proveer asistencia y ayuda a los usuarios.
- Formar a los usuarios y al cuerpo de ventas.
- Migrar el software existente o convertir bases de datos.

3.1.4.2. ESTRUCTURA ESTATICA

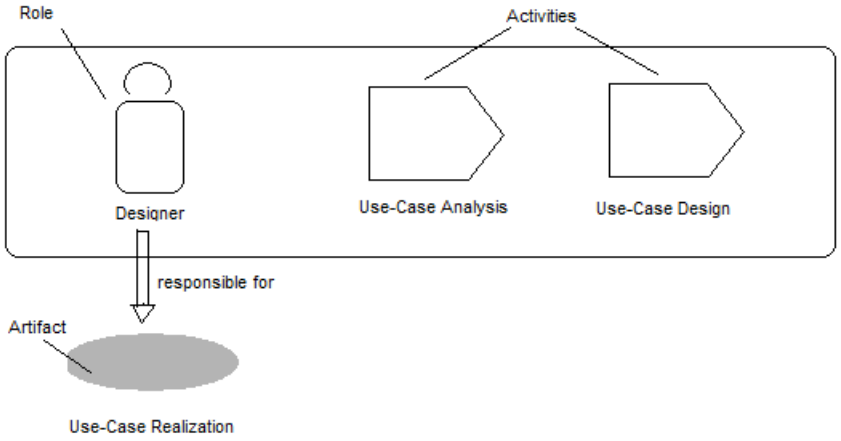
La estructura estática define como los elementos del proceso - actividades, disciplinas, artefactos y roles- están lógicamente agrupados en el proceso central. Un proceso describe **quién** (está haciendo) **qué, cómo** y **cuándo.**

Los cuatro elementos claves del modelado en RUP son:

- *Roles* – Quién
- *Actividades* – Cómo
- *Artefactos* – Qué
- *Workflows* – Cuándo

Un *rol* expresa quién (una persona individual ó un grupo) está realizando el trabajo, una *actividad* describe cómo lo está llevando a cabo y un *artefacto* es el resultado del trabajo. *Esto se resume a que una persona o un grupo con un rol determinado realiza determinadas actividades para producir un artefacto.*

Rol: cabe destacar que un rol no es una persona o un equipo, para RUP un rol es cómo una persona o grupo debe realizar una tarea, una persona generalmente cumple uno o más roles y diversas personas pueden ejecutar un rol. Esto se ve claro en el siguiente ejemplo, cuando la actividad consiste en modificar un artefacto, lo debe modificar el rol correspondiente, pudiendo ser otra persona y no la persona que creó el artefacto.



Actividad: una actividad tiene un claro propósito y generalmente se trata de la creación o actualización de un artefacto, tales como un componente, un modelo o un plan. Cada actividad es asignada a un rol específico.

Artefacto: un artefacto es una pieza de información que es producida, modificada o utilizada por el proceso. Son elementos que produce o utiliza el proyecto durante la construcción del producto final. Los artefactos son tomados como entradas por un rol para ejecutar una actividad y producir un resultado o un artefacto de salida.

Un artefacto puede tener varias formas:

- Un *Modelo*: un modelo, un modelo de caso de uso.
- Un *Elemento de modelo*: esto puede ser un elemento de un modelo, como ser una clase, un subsistema, etc.
- Un *Documento*: un Caso de Negocio
- *Código fuente*
- *Ejecutables*: como ser un prototipo.

Workflows: los roles, actividades y artefactos por sí solos no constituyen un proceso. Se necesita algo que defina la secuencia de actividades y la interacción entre los diferentes roles, a raíz de esto surge el concepto de workflow.

Los workflows vienen en diferentes formas y figuras, los dos workflows más comunes son las *Disciplinas*, las cuales son workflows a alto nivel y *Workflows Detail* detalles de workflows, los cuales son workflows dentro de una disciplina.

Disciplinas

Finalmente, todos los elementos del proceso – roles, actividades, artefactos y los conceptos asociados, guías y plantillas – son agrupados en un contenedor lógico llamado *Disciplinas.* Hay 9 disciplinas en el estándar de RUP pero la lista no es definitiva, se pueden agregar disciplinas. Dentro de las 9 disciplinas tenemos:

- Modelado de negocio
- Administración de requerimientos
- Análisis y diseño
- Implementación
- Despliegue
- Prueba
- Manejo de Proyecto
- Administración de cambios
- Ambiente

3.1.5 PRINCIPALES CARACTERISTICAS

- Forma disciplinada de asignar tareas y responsabilidades (quien hace que, cuando y como)
- Pretende implementar las mejores prácticas en Ingeniería de Software
- Desarrollo iterativo
- Administración de requerimientos
- Uso de arquitectura basada en componentes
- Control de cambios
- Modelado visual de software
- Verificación de la calidad del software

RUP se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye Artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de caso de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

En cuanto al alance de RUP, la metodología más apropiada para proyectos grandes, dado que requieren un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En proyectos pequeños, es posible que no se puedan cubrir los costos de dedicación del equipo de profesionales necesarios.

Por otro lado, en lo que se refiere a la metodología esta comprende tres fases claves:

- Dirigido por los casos de uso
- Centrado en la arquitectura
- Iterativo e incremental

3.1.6 BENEFICIOS QUE APORTA RUP

- Permite desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, el rendimiento, la reutilización, la seguridad y el mantenimiento del software mediante una gestión sistemática de los riesgos.

- Permite la producción de software que cumpla con las necesidades de los usuarios, a través de la especificación de los requerimientos, con una agenda y costo predecible.

- Enriquece la productividad en equipo y proporciona prácticas óptimas de software a todos los miembros.

- Permite llevar a cabo el proceso de desarrollo práctico, brindando amplias guías, plantillas y ejemplos para todas las actividades críticas.

- Proporciona guías explícitas para áreas tales como modelados de negocios, arquitectura web, pruebas y calidad.

- Se integra estrechamente con herramientas Rational, permitiendo a equipos de desarrollo aprovechar todas las ventajas de las características de los productos Rational, el lenguaje de Modelado Unificado (UML) y otras prácticas óptimas de la industria.

- Unifica todo el equipo de desarrollo de software y mejora la comunicación al brindar a cada miembro del mismo una base de conocimiento, un lenguaje de modelado y un punto de vista de cómo desarrollar software.

- Optimiza la productividad de cada miembro del equipo al poner al alcance la experiencia derivada de miles de proyectos y muchos líderes de la industria.

- No solo garantiza que los proyectos abordados serán ejecutados íntegramente sino que además evita desviaciones importantes respecto a los plazos.

- Permite una definición acertada del sistema en un inicio para hacer innecesarias las reconstrucciones parciales posteriores.

3.2 SCRUM

3.2.1 INTRODUCCION

Los objetivos de este proceso, (no es una metodología sino un PROCESO) son:

- Transmitir a los participantes las fases del proceso SCRUM con la que los equipos de desarrollo podrán gestionar proyectos donde la innovación y la inestabilidad de los requerimientos demandan una estrategia de gestión distinta a las tradicionales.

- Que los participantes experimenten trabajar en equipos auto-organizados y evalúen sus beneficios.

SCRUM es una metodología para la gestión de proyectos, en el que ponen de manifiesto que:

- El mercado competitivo de los productos tecnológicos, además de los conceptos básicos de calidad, coste y diferenciación, exige también rapidez y flexibilidad.
- Los nuevos productos representan cada vez un porcentaje más importante en el volumen de negocio de las empresas.
- El mercado exige ciclos de desarrollos más cortos.

*SCRUM es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software***.** El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad. Las iteraciones en general tienen una duración de dos a cuatro semanas. SCRUM se utiliza como marco para otras prácticas de ingeniería de software como RUP o Extreme Programming.

SCRUM se focaliza en *priorizar el trabajo en función del valor que tenga para el negocio*, maximizando la utilidad de lo que se construye y el retorno de inversión. Está diseñado especialmente para *adaptarse a los cambios en los requerimientos*, por ejemplo en un mercado de alta competitividad. Los requerimientos y prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares. De esta forma se

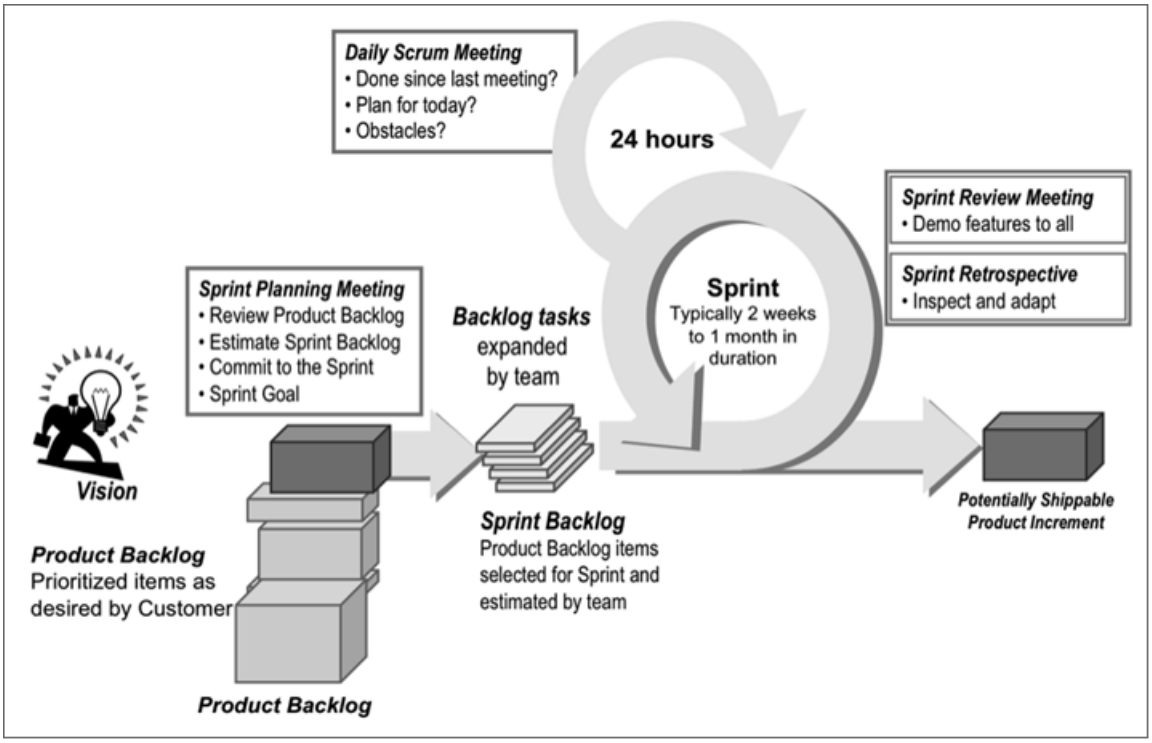
puede adaptar en tiempo real el producto que se está construyendo a las necesidades del cliente. Se busca entregar software que realmente resuelva las necesidades, aumentando la satisfacción del cliente.

En SCRUM, *el equipo se focaliza en una única cosa: construir software de calidad*. Por otro lado, la gestión de un proyecto SCRUM se focaliza en definir cuáles son las características que debe tener el producto a construir (qué construir, qué no y en qué orden) y en remover cualquier obstáculo que pudiera entorpecer la tarea del equipo de desarrollo. Se busca que los equipos sean lo más efectivos y productivos posible.

SCRUM tiene un conjunto de reglas muy pequeñas y muy simples y está *basado en los principios de inspección continua, adaptación, autogestión e innovación*. El cliente se entusiasma y se compromete con el proyecto, dado que ve crecer el producto iteración a iteración y encuentra las herramientas para alinear el desarrollo con los objetivos de negocio de su empresa. Por otro lado, los desarrolladores encuentran un ámbito propicio para desarrollar sus capacidades profesionales y esto resulta en la motivación de los integrantes del equipo.

3.2.2 EL ESQUELETO Y EL CORAZON DE SCRUM

SCRUM debe todas sus prácticas desde un proceso iterativo e incremental. El esqueleto de SCRUM se muestra en la siguiente figura. El círculo inferior representa una iteración del desarrollo de las actividades que ocurren una tras otra. El producto de cada iteración es un incremento en el producto. El círculo superior representa la reunión diaria que ocurre durante la iteración, en la cual los miembros individualmente del grupo conocen, inspeccionan las actividades y hacen los cambios apropiados. Como resultado de la iteración queda una lista de requerimientos. Este ciclo se repite durante todo el proyecto.



Esqueleto de SCRUM

Este esqueleto opera de esta manera:

I. Al comienzo de la iteración, el equipo revisa que es lo que debe hacer.

II. Luego, selecciona lo que cree que puede hacer para tener un incremento y un potencial prototipo funcional al término de la iteración.

III. El equipo se separa y hace su mejor esfuerzo por el resto de la iteración. Cuando ésta termina, el equipo presenta el incremento de la funcionalidad que construyó, de manera que los otros miembros del equipo puedan revisar las funcionalidades y hacer las modificaciones oportunamente al proyecto.

El corazón de SCRUM no es válido en la iteración. El equipo revisa los requerimientos, considerando la tecnología disponible, evaluando sus habilidades y capacidades. Luego, determina colectivamente como van a construir la funcionalidad, mientras que encuentran y discuten nuevas complejidades, dificultades y sorpresas. El equipo muestra cuales son las necesidades y cuál es la mejor forma de satisfacerlas. Este proceso de creatividad es el corazón de la productividad de SCRUM.

Este proceso es simple pero requiere mucha disciplina para ser exitoso y realizar cada uno de los siguientes de manera muy rigurosa:

- Administración de Requerimientos
- Administración de Riegos
- Planificación y Seguimiento

Como ya mencionamos anteriormente, este es un proceso y no una metodología y se basa en los siguientes principios ágiles:

- Colaboración estrecha con el cliente
- Predisposición y Respuesta a cambio
- Desarrollo incremental con entregas funcionales frecuentes
- Motivación y responsabilidad de los equipos por la auto-gestión, auto-organización y compromiso.

3.2.3 SCRUM APLICADO AL DESARROLLO DE SOFTWARE

En el desarrollo de software SCRUM está considerado como modelo ágil por la *Agile Alliance*.

3.2.3.1 ROLES DE SCRUM

Hay solo tres roles en SCRUM: el *Product Owner* (dueño del producto), *the Team (*el equipo*)*, y el *ScrumMaster (*Maestro Scrum*)*. Todas las responsabilidades de manejo de un proyecto se dividen entre estos tres papeles.

El Product Owner (dueño del producto)

- El Product Owner es el responsable de cuidar los intereses de cada uno de los participantes y del producto final.
- El *Product Owner* estima el financiamiento inicial y el requerido en el curso del proyecto mediante la creación de *the project's initial overall requirements* (los requisitos totales e iniciales del proyecto) y comparte con el equipo la visión del producto a construir.
- Decide la fecha en que se libera el producto.
- Acepta o rechaza el trabajo realizado.
- Es responsable por la rentabilidad del producto.
- Además, se debe preocupar por retornar los objetivos de inversión (ROI), y los planes de revisión.

El Scrum Master

El Scrum Master es responsable del proceso SCRUM, debe enseñar la metodología SCRUM a cada integrante implicado en el proyecto, preocupándose de poner la metodología en práctica de modo que se

encuentre dentro de la cultura de la organización y así entregue las ventajas previstas, asegurándose de que cada uno siga las reglas y prácticas de SCRUM.

Entre otras de sus tareas:

- Remueve impedimentos
- Protege al equipo de la interferencias externas
- Invita a las personas que deben asistir a las reuniones
- Actúa como coach y árbitro

La gente que sigue estos roles son las personas que confían en el éxito del proyecto. Otros pudieron estar interesados en el proyecto, pero no están comprometidos. SCRUM hace una distinción clara entre estos dos grupos y asegura de que los que son responsables del proyecto tengan la autoridad suficiente para hacer lo que consideren necesario para el éxito del mismo y de que los que no sean responsables no interfieran innecesariamente.

Esto se ejemplifica a continuación, refiero a esta gente como "cerdos" y "pollos," respectivamente. Estos nombres vienen de una vieja historia: El pollo y el cerdo

Están caminando en un sendero. El pollo dice al cerdo, "¿Deseas abrir un restaurante conmigo?" El cerdo considera la pregunta y responde, "Esta bien, ¿Cómo quieres tú que se llame el restaurante?" La gallina responde, "El jamón de Cerdo y los huevos de pollo!" El cerdo se detiene repentinamente y dice, "pensándolo bien, creo que no voy a abrir un restaurante contigo. Porque yo estaría realmente comprometido, pero tu solamente estarías implicado."

Esta distinción es importante en SCRUM. Debe siempre estar clara quién está comprometido y quién está solo implicado. Quién es ¿responsable del ROI, y quién tiene un intereses marcado en el ROI pero no es responsable? ¿Quién tiene que convertir la tecnología difícil en funcionalidad, y quién es un "abogado de diablo molesto"?, las reglas de SCRUM distinguen entre los pollos y los cerdos para aumentar la productividad, crean ímpetu, y poner fin al forcejeo.

El Team

Los equipos auto-suficientes, auto-organizados y funcionales, tienen la responsabilidad, en cada iteración, de generar el valor del negocio.

- Operan con alto grado de *confianza y autonomía*
- Debe existir entre los miembros un alto grado de *colaboración y comunicación*
- Producen *resultados medibles*

Entre las tareas que realizan están:

- Transformar los requerimientos en funcionalidad en cada incremento.
- El equipo debe estar conformado entre 5 y 8 personas.
- Es responsable de estimar el tamaño y esfuerzo del producto a construir.

- Cross-funcional (programadores, testers, diseñadores, etc.)
- Dedicación full time al proyecto
- Auto-organizados, autónomos
- Realizan seguimiento diario del proyecto.

3.2.3.2 EL FLUJO DE SCRUM

Un proyecto de SCRUM comienza con una visión del sistema que se irá desarrollando a medida que este avance. La visión pudo ser vaga al principio, quizás pensada en términos del mercado más que en términos del sistema en sí, pero quedará más claro a medida que el proyecto avance. El Product owner es responsable de financiar el proyecto y además debe entregar sus ideas de manera que se maximice su ROI. El dueño del producto formula un plan de modo que se incluya el Product Backlog. El Product Backlog es una lista de los requerimientos funcionales y no funcionales que, cuando se esté pensando en la funcionalidad, debe entregar el camino a seguir. Se da la prioridad al Product Backlog de modo que los artículos que probablemente sirvan para generar valor sean prioridad superior. El Product Backlog es un punto de partida, el contenido, las prioridades, y el agrupar el Product Backlog en diferentes lanzamientos usualmente cambia en el momento en que se comienza a deducir más exhaustivamente que espero del proyecto. Los cambios en el Product Backlog reflejan requerimientos del negocio que cambian, y cómo el equipo puede transformar rápidamente o lentamente el Product Backlog en funcionalidad.

Sprint

Todo el trabajo se hace en Sprint. Cada Sprint es una iteración de 30 días de calendario consecutivos. Cada Sprint se inicia con un Sprint Planning Meeting (reunión de planeamiento del Sprint), donde el Product Owner y el Team idean juntos lo que se espera para el siguiente Sprint. Seleccionando desde el Product Backlog la prioridad más alta, el Product Owner le dice al Team que desea, y el Team le dice al Product Owner cuánto de lo que quiere se puede transformar en funcionalidad durante el Sprint. Las reuniones de planeamiento del Sprint no pueden durar más de ocho horas.

Goal	Agree on the Scope of Sprint
Participants	Product-Owner, Team, Scrum-Master
Moderation	Scrum-Master
Duration	1 hour / Sprint-Week
Deliverables	Sprint Contract: Basic Parameters and the Sprint Backlog

What	Who	Duration
1. Basic Parameters • Start & End Date of Sprint • Review & confirm definition of done • Availability of team members. • (Expected velocity)	Scrum Master	5 to 10 minutes
2. Desired Scope • Sprint Goal • Present & discuss individual stories to be implemented, including acceptance criteria	Product Owner and Team	3 to 5 minutes per story.
2a. Reserve for difficult stories		10 to 20% of time allotted for Desired Scope
3. Commitment • Accept each story one at a time until the teams cannot accept any more stories	Scrum Master and Team	One to two minutes per story.
4. Agreement • Confirm the list of committed stories with the Product Owner	Scrum Master	One or Two Minutes
Follow-Up	Scrum Master	After the meeting, confirm the Sprint Contract with the Product Owner

Sprint Planning Meeting

La reunión de planeamiento del Sprint se divide en dos partes. Las primeras cuatro horas se dedican al *Product Owner* que presenta la prioridad más alta del Product Backlog al equipo. *El Team* le pregunta a él sobre el contenido, el propósito, el significado, y las intenciones del *Product Backlog*. Cuando el equipo sabe bastante, pero antes de que las primeras cuatro horas pasen, el equipo selecciona del *Product Backlog* lo que cree poder transformar en un incremento funcionalidad para el final del Sprint. El *Product Owner* confía que el equipo hará su mejor esfuerzo. Durante las segundas cuatro horas de la *reunión de planeamiento del Sprint*, el equipo planea su propio *Sprint*. Porque el equipo es responsable de manejar su propio trabajo, necesita un plan para comenzar con el *Sprint*. Las tareas que componen este plan se ponen

en un *Sprint Backlog*; las tareas en el *Sprint Backlog* emergen mientras que *el Sprint* se desarrolla. Al comienzo del segundo período de cuatro horas en la reunión de planeamiento del Sprint, comienza a correr el tiempo, y el reloj avanza rápidamente hacia el límite del Sprint (30 días).

Diariamente, el equipo se reúne para una reunión minuciosa llamada ***Daily Scrum Meeting***. Esta reunión tiene las siguientes características:

- Duración aproximada: 15 minutos
- Todos los días a la misma hora y en el mismo lugar
- Participa el equipo y el Scrum Master
- No hay conversación entre los miembros del equipo
- No se divaga
- El principal objetivo es eliminar impedimentos.
- Las tres preguntas básicas en esta reunión son:

  1. ¿Que haz terminado desde la última Daily Scrum?

  2. ¿Que planeas hacer a favor del proyecto entre esta y la próxima Daily Scrum?

  3. ¿Hay impedimentos que te detienen con cumplir con el Sprint y con este proyecto?

Los beneficios de estas reuniones *Daily Scrum Meeting* son:

- Mejor entendimiento de las interdependencias entre los miembros del equipo (sincronización).
- Mejora la comunicación.
- Todos conocen el estado del proyecto.
- Elimina la necesidad de otras reuniones.
- Identifica y remueve impedimentos.
- Promueve decisiones rápidas.

Al final del Sprint, se realiza una reunión de revisión de Sprint (**Sprint Review Meeting**).

Como *objetivo* esta reunión tiene presentar al Product Owner y demás involucrados del proyecto el trabajo realizado (incremento del producto) durante el Sprint o arquitectura subyacente.

De esta reunión *participan*: equipo, Scrum Master, Product Owner, todas las personas involucradas en el proyecto.

Reglas a seguir:

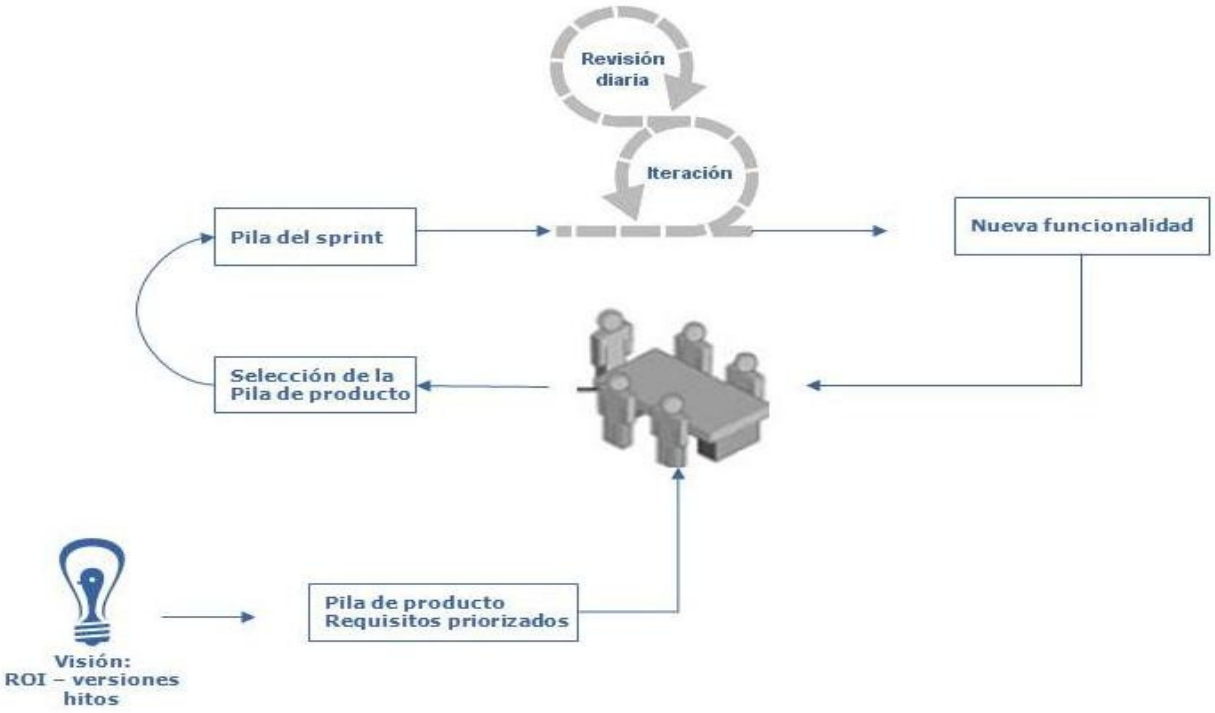
- El Team invierte el tiempo mínimo y necesario para preparar el Sprint Review.
- Las funcionalidades no finalizadas completamente no se presentan.
- Los miembros del equipo presentan las funcionalidades.
- Las demostraciones se realizan en las workstations de los miembros del equipo.
- Al finalizar la reunión se pide opiniones a los participantes, los cuales pueden sugerir cambio y mejoras.
- Al finalizar, se actualiza y se vuelve a priorizar el Product Backlog y el Scrum Master anuncia el lugar y la fecha de la próxima reunión de Sprint.

Después del Sprint Review Meeting y antes de la *reunión de planeamiento del Sprint*, el *Scrum Master* convoca a una *Sprint Retrospective* del Sprint con el equipo. En esta reunión el *Scrum Master* hace que el *Team* revise, dentro del marco y de las prácticas de proceso, su proceso de desarrollo SCRUM, para hacerlo más agradable, productivo y eficaz en las próxima iteraciones.

Cada miembro del equipo debe tener la chance de poder decir *que fue lo bueno, que cosas se pueden mejorar y que haría diferente en el próximo Sprint.*

- Todo aquello que afecte a como el equipo está construyendo el software se debe debatir.
- Siempre evaluar velocidad actual vs. planeado.
- Permitir al equipo evolucionar continuamente mejorando durante el proyecto.

En conjunto, *Sprint Planning Meeting*, *Daily Scrum Meeting*, *Sprint review Meeting*, y el *Sprint Retrospective*, constituyen la inspección empírica y prácticas de la adaptación del SCRUM.



3.2.3.3 ARTEFACTOS DE SCRUM

Product Backlog

El Product Owner es responsable del contenido, priorización, y disponibilidad del *Product Backlog*. *El Product Backlog* nunca se acaba, y es usado en la planificación del proyecto, es simplemente una estimación inicial de los requerimientos.

El Product Backlog se desarrolla paralelamente a medida que el producto y el ambiente en el cual se trabaja evoluciona. *Este* es dinámico; maneja constantemente los cambios para identificar que necesita el producto para ser:

- Apropiado
- Competitivo
- Útil

Mientras exista un producto, el Product Backlog también existe.

Datos relevantes del Product Backlog:

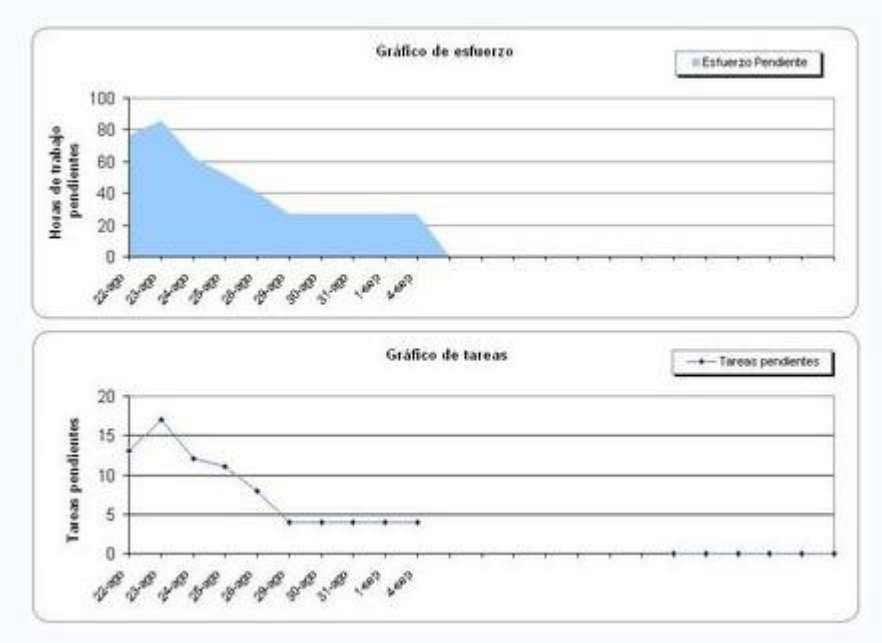
- Repriorizada al comienzo de cada sprint.
- Cualquiera puede aportar nuevos requerimientos.
- Es visible a todos.

Id	Módulo	Descripción	Est.	Por
Crítico				
1		Plataforma tecnológica	30	AR
2	Cliente	Interfaz de usuario	40	LR
3	Cliente	Un usuario se registra en el sistema	40	LR
4	Trastienda	El operador define el flujo y textos de un expediente	60	AR
5	Trastienda	Etc...	999.	XX
Necesario				
6	Cliente	El usuario modifica su ficha personal	30	AR
7	Cliente	El usuario consulta los expedientes asignados	15	LR
8	Cliente	El usuario tramita un expediente	35	LR

Nota: Los requerimientos internos como, refactoring, por ejemplo, no están en el backlog.

Carta Burndown

Una carta del *burndown* demuestra la cantidad de trabajo restante a través del tiempo. *La carta burndown* es una manera excelente de visualizar la correlación entre la cantidad de trabajo restante en cualquier punto y el progreso de los equipos de proyecto en la reducción de este trabajo. La intersección de una línea de la tendencia para el trabajo restante y el eje horizontal indica el punto más probable en el que terminen las actividades. Esto me permite ver el "Que pasa si…" al proyecto le voy agregando o quitando funcionalidad de manera de manejar el tiempo y si es necesario aplazar en función de ganar funcionalidad y vice -versa.



Sprint Backlog

El *Sprint Backlog* define el trabajo, o las tareas, que *el Team* desarrollará, en un incremento potencialmente funcional del producto. El equipo crea una lista inicial de estas tareas en la segunda parte del *Sprint planning meeting.* Las tareas deben ser divididas de modo que cada una demore entre 4 a 16 horas para finalizarlas.

3.2.4 BENEFICIOS QUE APORTA SCRUM

Una metodología de trabajo permitirá que la empresa cuente con patrones y alcance a disponer una gestión ágil de proyectos. Sin embargo, desarrollar suele ser una preocupación y encierra temor a lo desconocido.

Oh! Hay que introducir un cambio cuando ya estamos con el proyecto casi terminado…. ¿Que hacemos, tiramos todo por la borda? ¿Llegaremos con los plazos de finalización previstos?

La respuesta es "No es para desesperar". Solo se trata de instrumentar una metodología de trabajo al que permita "desviarse del cambio que se viene transitando" y elegir otro alternativo en función de las nuevas necesidades.

3.2.5 CLAVES DE SCRUM

- *Mejor con equipos pequeños y auto organizados:* es fundamental que los integrantes sean de distintas disciplinas para obtener mejores resultados. Además, la auto organización y transparencia hace que se encuentren todos comprometidos y motivados con la tarea. De hecho, la palabra SCRUM procede del vocabulario del rugby y es esa "figura" en la que los compañeros del equipo se amontonan, forman una piña y empujan todos en la misma dirección.

- *Punto de vista del usuario:* Se utiliza un lenguaje claro. "QUE es lo que queremos hacer". Estas sirven para confeccionar las listas de requerimientos del producto. A cada ítem de la lista se le asigna una prioridad. Además el equipo de trabajo estima el tiempo que demandará cada una de las tareas.

- *Sprints cortos, entregas frecuentes:* El mercado requiere ciclos de desarrollos cortos para esto se utiliza el sprint de requerimientos, una lista en la que se detalla "COMO" se van a construir los diferentes requerimientos del producto. Estos se dividen en actividades de no más de 16 horas y a su vez, cada sprint suele realizarse en un plazo de entre 2 y 4 semanas.

- *Roles dentro de SCRUM.* Cerdos y Pollos.

- *Reunión diaria. Transparencia diaria:*

Figura fundamental de la metodología SCRUM, algunas de las cosas a tener en cuenta:

I. La reunión es *diaria* y se hace siempre a una hora predefinida normalmente por la mañana.

II. Debe durar *alrededor de 15 minutos y se realiza de pie*, para mantener al máximo el máximo de concentración y atención.

III. Todos los roles son bienvenidos, pero sólo los "cerdos" pueden hablar.

IV. En la reunión se realizan las siguientes tres preguntas claves: ¿Que has hecho desde ayer?, ¿Que tienes planeado hacer mañana?, ¿Has encontrado algún problema para conseguir tu objetivo?

V. Uno de los puntos más importantes de la transparencia: todos los miembros saben que están haciendo los demás, y los problemas deben ser sacados a la luz en cuanto se detectan.

4. PROYECTOS TIPICOS

Si leemos la teoría de proyectos hay una regla básica que habla de una triple restricción. Es muy simple, hay tres componentes que no se pueden modificar sin que impacte en alguno de los otros dos (o en los dos). Los tres componentes son: *alcance* (todo lo que hay que hacer), *tiempo* (plazos para realizar el proyecto), y el *costo* (es el dinero que lleva hacer el proyecto. En la informática esta normalmente relacionado casi directamente al esfuerzo dado por los recursos humanos involucrados). No puedo pretender hacer más cosas con el mismo dinero y en el mismo tiempo. Tampoco puedo tardar menos tiempo con la misma gente (costo) y manteniendo el mismo alcance. Así podemos seguir sucesivamente. Hay una cuarta variable que muchas veces está relacionada a estas y en las bibliografías se la conoce como la cuarta restricción, que es la *calidad*. Es simple, comprensible, está en todos los libros, ahora por qué hay muchos que quieren todas las cosas rápidas, buenas, y baratas.

4.1 CARACTERISTICAS GENERALES DE LOS PROYECTOS

La mayoría de los proyectos de desarrollo de software en la Argentina son de tamaño pequeños y medianos, con una duración de entre 2 y 10 meses y muy cambiantes, no por indefiniciones sino por las reglas impuestas por factores externos como por ejemplo el propio negocio, leyes o políticas de la empresa, en los cuales la definición de los mismos es pobre, el cliente nunca tiene en claro lo que realmente necesita y en la mayoría de los casos los tiempos de desarrollo son limitados.

Para realizar nuestro análisis, vamos a separar los proyectos de desarrollo de software en dos:

- Proyectos nuevos
- Proyectos de mejoras y/o mantenimiento

Se consideran proyectos nuevos, aquellos que comienzan desde su preventa hasta su implementación final y proyectos de mejora y/o mantenimiento, aquellos a los cuales se les realizan cambios menores y agregados de nuevos módulos.

En general, se podría decir que los proyectos de mejoras y/o mantenimiento están más organizados, tiene menos presión y son más tranquilos para trabajar, no siendo así los proyectos nuevos; aquí los requerimientos suelen estar poco definidos o ser cambiantes, al acercase la fecha de implementación todo se transforma en caos debido a la presión de que todo debe estar terminado. Dependiendo de la cantidad de defectos que se detecten en la aplicación los miembros del equipo en la mayoría de los casos deben trabajar más horas y bajo presión para poder cumplir con los tiempos estipulados.

4.2 CARACTERISTICAS GENERALES DE LOS CLIENTES

Es muy importante la disponibilidad del cliente dado que debe proveer información, tomar decisiones, revisar y aprobar documentos, prototipos, etc. Esto no siempre es factible.

Muchas veces el cliente no tiene bien definido lo que realmente quiere, eso provoca muchos cambios posteriores.

La capacidad de respuesta al cambio se fundamenta en la colaboración con el cliente. Intercambiar información continuamente permite obtener un producto final que satisface los requisitos iniciales. El cliente debe confiar en el criterio de los desarrolladores, pero debería estar informado de lo que se está haciendo, pudiendo así ayudar al equipo de desarrollo a refinar los requerimientos.
La estructura de la organización cliente debe estar preparada para mantener este canal de comunicación continua, lo cual no siempre es así. Esto se complica aun mas cuando es el caso de clientes extranjeros, con grandes corporaciones, donde cada mínimo cambio debe ser aprobado por todas las áreas de la corporación, se suma también la coordinación que se debe hacer debido a la diferencia horaria que pueda llegar a existir.

También existen casos en que los clientes se resisten a pagar el costo que requieren los cambios dado que consideran que la aplicación no realiza lo pedido, no satisface sus necesidades.

Muchas veces el cliente solicita cosas que implican una gran complejidad en su desarrollo y luego no la utilizan

4.3 CARACTERISTICAS GENERALES DE LOS EQUIPOS

En cuanto al equipo de desarrollo podemos decir que existen varios tipos: equipos colaborativos y no colaborativos, organizados y desorganizados, comprometidos y no comprometidos. Así mismo, dentro de cada uno de estos tipos de equipos tenemos diferentes características, como ser:

- Categoría de cada miembro (esto es a nivel de conocimiento, pasantes, Junior, Semi-Senior y Senior)
- Horarios de trabajo, hay personas dentro de un mismo equipo que trabajan part-time y otras full-time
- Equipos que no están todos en el mismo lugar físico

Se pueden presentar también casos particulares dentro del equipo de desarrollo como ser:

- épocas en las que se presentan muchas rotaciones de personas
- personas disconformes con la empresa o que consideran una mayor posibilidad de crecimiento en algún otro lugar se van e ingresan o no otras nuevas.

Dadas las características generales ahora ejemplificaremos cada uno de los casos.

4.4 PROYECTO NUEVO

El proyecto se trata de un sistema de Ruteo y Gestión cuyo objetivo es soportar el negocio de Mercado de Capitales: cubriendo todos los aspectos de la operación de Trading. Este sistema recibe órdenes de los clientes y las envía al Mercado. A su vez, recibe de este la información de las ejecuciones y las asigna a las contrapartes correspondientes.

¿Para qué sirve este sistema?

- Gestión de órdenes e instrucciones.
- Envío y recepción de transacciones e información de Mercado en tiempo real.
- Procesamiento de alto volumen de órdenes.
- Cálculo de posición y ganancias/pérdidas intradía (P&L) por cuenta.

¿Quién lo utiliza?
Este sistema es utilizado por los distintos tipos de usuarios de las Casas de Bolsa líderes del mercado: Traders, Sales Traders, usuarios de middle-office y administradores de seguridad.

¿Por qué utilizarlo?

- Permite procesar altos volúmenes de transacciones, indispensables para negocios basados en trading electrónico.
- Gestiona todo tipo de órdenes del mercado.
- Soporta en forma integral la operación de trading.
- Visualización en tiempo real del estado de las operaciones.

Este sistema fue construido desde su etapa inicial por la empresa para un banco de México; con el tiempo fue tomando fuerza en el mercado Mexicano y así fue como gran parte de los bancos de ese lugar hoy cuentan con él.

Si bien el sistema está considerado dentro de la compañía como un producto, cuando un nuevo cliente lo compra, se trata como un nuevo proyecto.

El sistema nació en el año 2006 para un banco muy grande e importante, contó con tiempos de desarrollo muy acotados y un equipo de trabajo muy reducido (sólo 4 personas). Fue un proyecto muy complicado pero exitoso, ya que entre el año 2008 y 2009 el sistema fue vendido a 6 nuevos clientes. Esto provocó varios cambios en muchos sentidos.

4.4.1 CARACTERISTICAS GENERALES

Como hemos mencionado, estamos hablando de un sistema que se vende a varios clientes. Este sistema se adapta a los diferentes mecanismos que se pueden utilizar para operar con la Bolsa Mexicana de Valores, todo esto es configurable. Cada cliente configura el sistema a su modo de operación y de ser necesario solicita cambios e incorpora nuevas funcionalidades, para esto último no debemos perder de vista que se deben introducir de manera tal que no afecta el funcionamiento del resto de los clientes, nunca podemos perder de vista la visión de producto.

Como se dijo, el sistema se vendió a varios clientes en los últimos años, en particular durante el 2008, tres nuevos clientes salieron a producción y se mantuvieron los otros tres que existían.

Algunos de los problemas que se presentaron en el transcurso de estos tres nuevos proyectos fueron los siguientes:

El equipo de trabajo (cantidad de personas) no se modificó, al introducir estos 3 nuevos proyectos. Al realizarse las estimaciones de los mismos se tuvo en cuenta que se desarrollarían los 3 proyectos en paralelo (por los pedidos de los clientes no todos los proyectos tenían la misma duración), pero no que pudiera existir algún problema en los otros 3 clientes que ya estaban en producción.

En el transcurso de un mes, surgieron varios problemas de estabilidad en la aplicación de uno de los clientes que ya estaba en producción, esto provocó que se retrasaran los desarrollos de los nuevos proyectos y por consecuencia que se acortaran los tiempos de prueba y generó sobrecarga de trabajo para muchos integrantes del equipo.

Cuando ocurrió esto, todo se manejaba con un único equipo dentro de la empresa (no existía la tercerización ni el equipo de desarrollo externo). Pasada esta etapa crítica, el sistema se siguió vendiendo a otros clientes, fue este el momento en donde el equipo no dio abasto, se comenzó a tercerizar y se creó el nuevo equipo de desarrollo externo. Este cambio hizo que se tengan que reforzar varias tareas y fases del proyecto que antes estaban descuidadas:

- Por falta de tiempo, no se realizaba la documentación necesaria. Esto tuvo que cambiar rotundamente ya que se debía enviar la documentación funcional al equipo externo.

- Se comenzó a utilizar una herramienta de diseño. Si bien no tenemos documentado todo el sistema, cualquier modificación que impacte en el diseño ó un módulo nuevo, debe quedar asentado. La idea es que progresivamente todo el diseño del sistema quede documentado.

- Se comenzaron a cargar casos de prueba para cada uno de los desarrollos, mucho más detallado y tratando de cubrir el impacto del cambio en toda la aplicación.

Junto con todos estos cambios se implementó un nuevo proceso de Testing, si bien el sistema es uno solo, el hecho de que sea tan configurable, hace que cada cambio tenga que testearse en varios ambientes (diferentes clientes). Se sumó gente destinada sólo a testing (2 testers).

Actualmente, la carencia más importante que tenemos es en la etapa de pruebas, si bien el sistema lleva varios años en producción, está en constante modificación porque se siguen sumando clientes. Esto hace que se tengan que organizar entregas todos los meses, y el testing se hace pesado, ya que no tenemos todavía el testing automático.

4.4.2 CARACTERISTICAS GENERALES DEL CLIENTE

Como mencionamos anteriormente, trabajamos para clientes de México, el hecho de que el cliente esté en el exterior, agrega cierta complejidad al proyecto.

En cuanto a la relación con el cliente, no es sencilla, debido a la distancia y la diferencia horaria; más aún si son grandes corporaciones como son los bancos para los que trabajamos, muchas veces se dificulta obtener respuestas rápidas, además es complejo llegar al usuario final, ya que existen diferentes áreas que impiden llegar hasta ellos; esto tiene como consecuencia que se definen cosas con áreas intermedias, que no son usuarios finales, que luego no sirven, ya que no es lo que realmente solicitó el usuario final, es tanta la gente involucrada que se termina transgiversando el pedido inicial.

Para resolver este tipo de problemas, cuando se trata de un cliente nuevo (compra del producto) debe viajar gente de Argentina a México para realizar el relevamiento de los requerimientos, luego solemos estar en contacto telefónico para terminar de definir el alcance del proyecto, esto no suele ser una tarea simple.

Cuando se trata de cambios ó nuevos módulos para clientes existentes, que no afectan en gran medida al sistema se resuelve con conferencias telefónicas, reuniones vía web, demos, etc.

Por último, también agrega complejidad el hecho de que algunos de estos clientes, tienen áreas distribuidas en otros lugares físicos, como ser New York, China, etc., esto hace aún más compleja la comunicación.

4.4.3 CARACTERISTICAS GENERALES DEL EQUIPO

Como se mencionó anteriormente, inicialmente el equipo constaba de 4 personas para desarrollo, una de las cuales tenía también el rol de Analista, más el Manager del proyecto. A medida que el producto se fue vendiendo a otros clientes esto se fue modificando.

Hoy en día, con 8 clientes, el equipo consta de 18 personas. Estas 18 personas están organizadas de la siguiente manera:

*Equipo de desarrollo***:** Desarrolladores Juniors y Semi-Seniors, más un desarrollador Senior que es quién dirige a este equipo.

*Equipo de implementaciones y Clientes***:** Consta de personas que coordinan clientes, estas están encargadas de mantener la comunicación diaria con el cliente y estar al tanto de los pedidos que estos realizan.

*Equipo de Implementaciones y Soporte***:** Equipo de personas que particularmente realiza tareas de soporte e implementación.

*Manager***:** El manager se encarga de mantener las cuentas de los clientes y hacer principalmente la parte comercial del proyecto.

Existen varias particularidades en este equipo de trabajo, por un lado, el equipo de desarrollo se encuentra en distintos lugares físicos, una parte en las oficinas de la empresa y otra en oficinas externas; además, se terceriza parte del desarrollo a diferentes consultoras. Esto hace que las reuniones de estado semanales sean fundamentales, para comunicar las novedades y analizar el estado del proyecto.

Por otro lado existen personas, que tienen el rol de coordinar al cliente (debido que son varios clientes, existe una persona encargada de llevar el día a día de cada uno de ellos), estas mismas personas se encargan de dar soporte y de coordinar otras fases del proyecto, si bien todas las personas realizan todas las tareas existe un referente para cada fase del proyecto, está el leader funcional, leader de testing, etc.

Por último, existe un equipo de soporte e implementación en oficinas de la empresa localizadas en México, estos últimos son quienes asisten a los diferentes clientes ante algún problema (soporte), además de estar presentes en cada una de las implementaciones. Algunos de ellos, también realizan parte del análisis funcional de los requerimientos que solicitan los usuarios, de esta forma la gente que está establecida en Argentina viaja con menos frecuencia.

4.5 PROYECTO DE MEJORAS Y MANTENIMIENTO

4.5.1 CARACTERISTICAS GENERALES

El proyecto al que haremos referencia se trata de un sistema para una empresa de Telecomunicaciones en la Argentina, en el que existen varias áreas, Call center, Atención al público, Telecobranzas, Prelegales, BackOffice, Precontención y muchas más. La razón de ser del sistema era facilitar y agilizar la atención al cliente dado que dicha empresa se caracteriza por brindar calidad en la atención del cliente y en el servicio prestado. Hasta ese momento el usuario de la aplicación para atender a un cliente debía interactuar con varios sistemas, 7 o más, a los usuarios los denominaban expertos del uso de las teclas control + tab. El objetivo del proyecto era desarrollar un portal que integre todos esos sistemas, de esta manera el usuario interactuaría sólo con un sistema.

El proyecto comenzó a principios del año 2005, el mismo cliente realizó la incepción y la elaboración del proyecto, nuestra empresa recién se involucró en la etapa de la construcción y transición. Fue un proyecto muy duro pero exitoso.

Como ocurre en la gran mayoría de los proyectos este fue un proyecto más en el que las estimaciones de los tiempos eran muy optimistas, 5 meses cuando en realidad se necesitaban unos 3 meses más. Pero por suerte gracias al esfuerzo de todos en 5 meses tuvimos una exitosa puesta en producción. Luego de esto comenzó lo que denominamos la etapa de mejoras y mantenimiento de la aplicación, digo aplicación porque surgieron también proyectos desde 0 que se anexarían a la misma, son módulos grandes totalmente independientes que tienen su entrada a través del Portal. Ellos son sistemas de gestiones, Servicio Técnico y Siniestros, Colas, Gestión de Mora y Cobranzas y muchos más. Hay requerimientos o necesidades chicos que entran dentro del circuito de mejoras y mantenimiento o muy grandes que entran dentro del circuito de proyectos nuevos.

A los proyectos de este cliente los podemos caracterizar como medianos, con una duración de entre 2 y 10 meses y muy cambiantes, no por indefiniciones sino por las reglas impuestas por factores externos como por ejemplo el propio negocio, leyes o políticas de la empresa, en los cuales la definición de los requerimientos muchas veces es pobre y en la mayoría de los casos los tiempos de desarrollo son limitados.

Ejemplos: muchas veces el tiempo depende de la urgencia por diferentes motivos que tiene el cliente de poner en producción algo. En este cliente se da que estamos ajustados en tiempo por decisiones políticas de la empresa o por definiciones de leyes. Ejemplo del primer caso, un proyecto de Garantías Especiales y del segundo un proyecto de Factura Electrónica.

4.5.2 CARACTERISTICAS GENERALES DEL CLIENTE

En cuanto al cliente nos ha tocado un cliente que nos provee información siempre que la necesitamos, toma decisiones rápidamente y realiza las revisiones y aprobaciones correspondientes. La empresa asigna a cada proyecto un key user al 100%.

No es muy común que los clientes asignen al proyecto al usuario más capaz y con más experiencia, siendo este un punto clave en todo proyecto.

El cliente no se encuentra en el mismo lugar físico que el equipo de desarrollo. Los canales de comunicación son el chat, el mail, el teléfono, y las reuniones de avance que se realizan en forma semanal.

Cuando se realizan revisiones y aprobaciones de documentos que definen las necesidades de los usuarios como por ejemplo, definición de requerimientos o casos de uso NO es recomendable que los mismos sean enviados por mail y quedarse a la espera de una respuesta porque si a futuro llegara a surgir algún cambio muchas veces las respuestas ante la frase ‘en el caso de uso dice tal cosa y fue aprobado por ustedes’ ellos responden, pero cuando yo lo leí interpreté otra cosa. Lo ideal es revisar y aprobar los documentos en una reunión y luego dejar asentada una minuta de la reunión

4.5.3 CARACTERISTICAS GENERALES DEL EQUIPO

El equipo se compone de la siguiente manera:

- Líder de proyecto
- Analistas programadores
- Líderes técnicos
- Desarrolladores

No todos los miembros del equipo trabajan el mismo tiempo, hay personas que trabajan full-time y part-time, y con diferentes señorities (pasantes, junior, semi-senior y senior).

El equipo de desarrollo se encuentra en dos lugares diferentes físicamente, uno en las oficinas de Capital y otro en las oficinas de La Plata. Los canales de comunicación entre ambos son el chat, los mails, el teléfono y visitas formales de una vez por semana o cuando una determinada situación lo requiere.

A lo largo de todos estos años han rotado mucho los miembros del equipo. Se podría decir que de las 18 personas sólo dos son personas que están desde el inicio.

En cuanto a ser un equipo colaborativo depende del tiempo, hay momentos en los que todos están contentos y hay un equipo colaborativo pero hay momentos en los que eso no ocurre.

A lo largo de estos 4 años en la vida del Proyecto se fueron aplicando distintas prácticas de distintos procesos en los que algunos sirvieron y otros no, todos los miembros del equipo creo fuimos aprendiendo de los errores y aciertos que fuimos teniendo. Basándonos en esas experiencias queremos plantear prácticas o procesos que realmente aporten a un proyecto de estas características.

5. ANALISIS DE LOS PRINCIPIOS DE RUP Y SCRUM

5.1 RUP

5.1.1 PRINCIPIOS DE RUP

Uno de los grandes defectos de los usuarios de RUP es que producen artefactos sólo porque RUP describe como hacerlos.

Se ha comparado a RUP con un gran desayuno tipo Buffet. Si bien el desayuno presenta varias opciones para consumir, uno elige lo que le gusta o lo que necesita para que le haga bien, RUP es lo mismo, ofrece o define un conjunto de buenas prácticas, y el proyecto de acuerdo a sus propias características debe tomar las prácticas que lo benefician.

Experiencia personal:
Esto último es cierto, en nuestra empresa no todos los artefactos son útiles, la mayoría de ellos se hacen para la certificación de CMMI. Hacer un artefacto lleva tiempo y si nadie lo consume resulta una pérdida de tiempo. Muchas veces nadie consume el artefacto porque no tiene un formato amigable, es importante pensar en que el artefacto producido debe ser amigable y fácil de realizar. Nos pasa con la matriz de traza por ejemplo, se considera muy útil, pero no resulta provechoso la manera en que se implementa en la empresa, la misma produce sobrecarga de trabajo para realizarlo y es difícil su interpretación a la hora de necesitar la información.

Tener un proceso es importante, pero las herramientas utilizadas para llevarlo a cabo deben ser herramientas que realmente ayudan y que no generen doble trabajo. Los analistas en nuestra empresa en algunos casos NO contamos con buenas herramientas para cumplir con las tareas definidas en los procesos.

Una opinión personal es que cuando una empresa decide establecer un proceso, la mejor manera es adoptar una metodología ágil y luego paulatinamente ir migrándolo hacia una metodología pesada si así lo desea. Las metodologías pesadas suelen quedar en la nada por todo el cambio que implica, es mucho trabajo y requiere de mucho esfuerzo.

En la mayoría de los proyectos uno se debe focalizar o debe tratar de minimizar el número de artefactos producidos para reducir la sobrecarga de trabajo. Lo que se aconseja en el libro de RUP, es que si uno está en la duda de producir o no un artefacto, mejor NO lo produzca. PERO tampoco se debe usar este consejo

como excusa para no hacer otros artefactos que SI son importantes y es mejor hacerlos, como por ejemplo, definir el documento de visión, documentar los requerimientos, tener un diseño, planificar el esfuerzo de la etapa de pruebas.

A continuación se realizará un análisis de cada uno de los principios de RUP y SCRUM aplicados a los proyectos planteados en la sección anterior, Proyecto Nuevo (de aquí en adelante **PN**) y Proyecto de Mejoras y Mantenimiento (de aquí en adelante **PMM**). Para cada uno de ellos indicaremos si resulta positivo o negativo, en función de si se considera importante o no para el proyecto.

1. Tanto RUP como SCRUM utilizan un enfoque iterativo, esto es, una secuencia de iteraciones incrementales.

Cada iteración incluye algunas o muchas de las disciplinas de desarrollo (requerimientos, análisis, diseño, implementación, etc.). Cada iteración tiene un conjunto de objetivos bien definidos y produce una implementación parcial del sistema final. Cada sucesiva iteración se construye sobre las iteraciones anteriores para refinar el sistema hasta que el producto final este completo.

A través de este enfoque iterativo que generan entregables ejecutables, se logra detectar en forma temprana los desajustes e inconsistencias entre los requerimientos, el diseño, el desarrollo y la implementación del sistema, manteniendo al equipo de desarrollo focalizado en producir resultados. Además, este enfoque ayuda a atacar los riesgos mediante las entregas parciales progresivas y frecuentes que permiten la opinión e involucramiento del usuario.

El proceso de desarrollo en sí mismo es mejorado y refinado a lo largo del proyecto, al final de cada iteración se analiza el estado del proyecto, del producto y qué cosas pueden ser mejoradas del proceso.

<table>
<tr><th colspan="2">PRINCIPIO POSITIVO</th></tr>
<tr><td colspan="2">Se considera positivo porque:

a. Permite paralelizar las tareas de análisis, diseño, desarrollo, implementación y prueba.

b. La definición y cumplimiento de objetivos en el corto plazo motiva tanto al equipo de trabajo como al cliente. Las personas por naturaleza trabajan por objetivos.

c. El resto de los beneficios están explicados en el segmento RUP Y EL DESARROLLO ITERATIVO.

Problema: Se debe tener cuidado cuando se establecen los tiempos de las iteraciones, muchas veces se determinan tiempos tan largos que se pierde el sentido del enfoque iterativo.</td></tr>
<tr><th>PMM</th><th>PN</th></tr>
<tr><td>En los proyectos utilizamos el enfoque iterativo

Se trata de planificar iteraciones pequeñas, es decir, si tenemos una lista de 10 requerimientos, se priorizan, de manera tal de seleccionar 3 o 4 para salir a producción, luego se seleccionan otros 3 o 4</td><td>Se utiliza el enfoque iterativo en varias circunstancias.

En primer lugar, cuando se trata de un cliente nuevo, si se solicitaron muchos requerimientos, se planifica desde la etapa inicial cuáles van a</td></tr>
</table>

y se hace lo mismo. La idea de esto, es realizar varias entregas y que vayan saliendo a producción a medida que se van terminando los desarrollos, de esta forma se van obteniendo ganancias antes de tiempo, es decir, se focaliza en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de inversión	entregarse en qué fechas, es decir, se realiza la entrega de requerimientos por fases, de esta forma el usuario puede ir realizando pruebas (UAT – User Acceptance Testing) antes de tener el sistema completo. Por otro lado, dado que hay varios clientes y todos solicitan modificaciones y nuevas funcionalidades constantemente, se planifican entregas parciales, las cuales involucran requerimientos de varios clientes. Estas entregas se realizan bimestralmente, empezamos a seguir este proceso porque el hecho de tener tantos clientes, hacía que se realizaran varias entregas todas las semanas, esto tuvo sus ventajas y desventajas que se explicarán más adelante.

2. Atacar los principales riesgos de manera temprana y continua. La gestión de riesgos es un proceso dinámico y permanente.

Manejar los riesgos del proyecto implica identificarlos, establecer la probabilidad de ocurrencia de los mismos, el impacto que tendrán en el proyecto y que acción se deberá tomar para mitigar su ocurrencia.

PRINCIPIO **POSITIVO**	
Este principio es positivo dado que los improvistos o problemas en los proyectos nunca son buenos, siempre impactan de manera negativa. Es mejor detectarlos y manejarlos en etapas tempranas.	
PMM	**PN**
En la empresa existe el *proceso de gestión de Riesgos*, pero no resulta práctico. La definición del mismo es correcta, lo que no es útil es el artefacto que produce. El artefacto es una planilla Excel (Risk Log.xls) que permite al Líder de Proyecto gestionar ordenadamente los riesgos del proyecto. A nuestro entender, el artefacto debería ser una aplicación tipo tracking de riesgos para facilitar la carga, administración y consulta de la información. Con una planilla en Excel es fácil manejar la información en el momento en el que se está realizando la actividad, pero luego de varios años se dificulta interpretar la información; el primer problema sería encontrar el Excel. En los proyectos de tipo mejoras y mantenimiento	En nuestra empresa, la Gestión de Riesgos es una política, que establece que dentro de la etapa de planificación, todos los proyectos deben documentar y validar con el cliente el plan de proyecto, que se desarrolla en las fases tempranas del mismo, uno de los ítems de este documento es la identificación de riesgo. Dentro de lo que llamamos 'identificación de riesgos' lo que se hace, es listar todos los riesgos del proyecto, categorizarlos y clasificarlos para luego poder administrarlos. El propósito de esto, es establecer un plan de contingencia y/o mitigación. Si bien existe un proceso, no lo seguimos en forma estricta, se hace mucho menos de lo que se debería. Creo que es bueno el hecho de llevar esto en la

no se está aplicando actualmente según el proceso definido, se realiza de manera informal. Lo cierto es que muchas de las actividades se realizan inconscientemente, es lógico que una persona en el momento de planificar una iteración piense naturalmente en los riesgos que existen y la manera en el que los mismos se pueden mitigar. A veces no es necesario formalizar eso en un artefacto sino tratar de mitigarlo.	herramienta de tracking ya que se lo tiene presente en todo momento del proyecto. Debo reconocer que si bien como política se dice que se debe monitorear y actualizar todos los meses lo que sucede, se cargan los riesgos iniciales al comienzo del proyecto y luego se monitorean esos mismos durante la vida del proyecto, no se cargan nuevos cuando sí surgen, salvo casos excepcionales. La empresa considera que se hace muy poco respecto a este tema, pero actualmente esta focalizada en los requerimientos y las pruebas.

3. Documentar los requerimientos de forma tal que sean fáciles de ser entendidos.

El mismo puede ser a través de la especificación de Casos de uso y prototipos de pantallas. Es importante que tanto el cliente, los desarrolladores y los testers hablen el mismo idioma, esto es, que se esté de acuerdo en lo que el cliente necesita, lo que el desarrollador construye y lo que el tester debe probar para confirmar que la aplicación responde a las necesidades del usuario.

Los prototipos de pantallas son la manera más efectiva de documentación dado que es lo más cercano a la aplicación final donde el cliente puede ver el diseño de las pantallas, la interacción con la misma y la información que se brinda, pudiendo analizar en ese momento (antes de comenzar la construcción) lo que no le gusta, no le sirve o lo que se podría mejorar.

PRINCIPIO **POSITIVO**	
Es importante que tanto el cliente como el equipo de desarrollo y de pruebas interpreten lo mismo de un requerimiento. Una mala interpretación se paga muy caro dado que se termina construyendo algo que el cliente no pidió o el tester puede dar como válido algo que no lo es. Los prototipos al ser el modelo más cercano al producto final ayudan a tener una mejor interpretación de los requerimientos.	
PMM	**PN**
En la empresa existe el proceso de Administración de Requerimientos, pero el mismo genera sobrecarga de trabajo. El proceso define que se deben Administrar los cambios a los requerimientos conforme estos evolucionan durante el proyecto, la definición está bien, lo que no se comparte es la manera en la que se debe llevar a cabo, mucha burocracia, la confirmación de un cambio puede llevar varios	Como política de la empresa tenemos que todos los proyectos deben documentar, validar y obtener la aprobación del cliente, de acuerdo al procedimiento establecido. Se debe mantener en todo momento los planes, tareas y productos de trabajo consistentes con los requerimientos definidos. Como crítica se podría decir que no estoy de acuerdo con que el análisis de los requerimientos se haga con grupos de personas (en general el analista, uno de los lideres y la persona que lo va a

meses. El proceso define también que se debe mantener una trazabilidad bidireccional entre los requerimientos, los planes y productos de trabajo. Esto se hace relacionando en el EA los casos de uso con las clases que lo implementan, se considera que el costo es mayor al beneficio. Por otra parte, el documento de especificación de requerimientos de software (ERS) es muy importante pero es manejado en un archivo de texto (Word). Sería mejor si se contara con una herramienta. Hay proyectos que siguen el proceso ágil SCRUM y entre las prácticas no está el escribir casos de uso. Se considera que los casos de uso son importantes. Cuando se escriben casos de uso, se presentan dudas que lo van ayudando a uno a acercarse a lo que realmente quiere el usuario. Ayuda a ir cerrando cada uno de los circuitos que se presentan; flujos básicos, flujos alternativos, validaciones, etc. Se realizan revisiones de la adherencia del producto, de acuerdo a lo definido en el plan de QA (Quality Assurance) del proyecto a partir de un checklist correspondiente. Muchas veces no se cuenta con el tiempo disponible para ajustar los documentos de acuerdo con la revisión realizada. No se considera tan importante el tema de que los documentos estén perfectos.	desarrollar), ya que el resto queda afuera de la discusión. En mi caso particular (Líder de Testing) esas funcionalidades nuevas después llegan a la etapa de Pruebas y muchas veces no sé ni de que se tratan. También es cierto que hay muchos puntos por analizar y de llevar a todo el equipo a este tipo de reuniones sería una pérdida de tiempo. En cuanto a la herramienta que utilizamos, no sé si es la mejor, pero al menos tenemos todo centralizado en un solo lugar, antes de utilizar esta herramienta, no se llevaban registros de requerimientos como tales, sino que se cargaban issues dentro de la herramienta de tracking y de ser necesario teníamos algún documento relacionado (funcional y diseño) en el cvs; esto era más complejo de administrar, nunca se encontraban las cosas y se hacía muy difícil encontrar la funcionalidad propia de un cliente. Por otra parte, es útil el hecho de tener una BFS por cada requerimiento, esto hizo que se tengan algunas partes del sistema documentadas; no es la mejor documentación pero al menos vamos teniendo algo, lo malo de esto, es que esta documentación es por requerimiento y no por módulo funcional. Por último, es bastante costoso el hecho de mantener todo en el EA, requerimientos, diseños y casos de prueba, esto hace que el trabajo del analista funcional sea mayor, pero es la única manera que encontramos por ahora de organizar todos los clientes, todas las versiones y los grupos de desarrollos (tercerización).

4. Mantener el foco en el software ejecutable.

Los documentos, diseños y planes son importantes, pero no reflejan el verdadero progreso del proyecto y su importancia pasa a ser secundario comparado con el código fuente. Un código que compila y pasa las pruebas de manera exitosa es la mejor evidencia del progreso del proyecto.

PRINCIPIO **NEGATIVO**
Es cierto que un código que compila y pasa las pruebas de manera exitosa es una buena evidencia del

<table>
<tr><td colspan="2">progreso del proyecto, pero no se puede dejar de tener una mínima documentación. Los planes y diseños son importantes.
Con la realidad actual no se debe dejar de lado el tema de la transferencia de conocimiento, el mercado actual lamentablemente facilita o incentiva a que las empresas sufran la rotación de personal. De alguna manera la información del proyecto debería quedar formalizada en algún documento. Los documentos pertenecen a la empresa en cambio las personas no.</td></tr>
<tr><th>PMM</th><th>PN</th></tr>
<tr><td>No hay un proceso definido en la empresa para este principio.

Este punto es discutible, dado que se puede pagar caro luego con el correr del tiempo. Es cierto que un código que compila y pasa las pruebas de manera exitosa es la mejor evidencia del progreso del proyecto pero esto sirve sólo en el momento que se está construyendo y antes de su salida a producción. Sin tener un análisis previo o un diseño, ¿quién garantiza que las pruebas que se corren sobre ese código son correctas?.

Contar sólo con código fuente puede resultar costoso en el momento en el que se deba realizar una corrección o mejora en el futuro. No tener documentación sobre la funcionalidad o diseños puede hacer más compleja la implementación. Es muy común que en el momento en el que se le asigna un requerimiento a un desarrollador pregunte, ¿hay documentación acerca del mismo?, ¿hay algo que se pueda leer en algún lado?

Los diseños también son importantes, una representación gráfica de todo el circuito de un proceso (*) por ejemplo, ayuda mucho más que miles de líneas de código o un documento con muchas hojas.

Si un desarrollador está construyendo un requerimiento que no está documentado y además no documenta lo que va construyendo, luego si se enferma (se ausenta algunos días) y esto se da antes de subir un release a test por ejemplo, es un problema. Los documentos o anotaciones de alguna manera ayudan a evitar estas situaciones. Es muy peligroso que las informaciones queden sólo en la cabeza de las personas. Ante la ausencia de una persona, otra debería poder seguir con el</td><td>No hay un proceso definido para esto ya que se pide que todo se documente. Si bien quizás para ciertas cosas no hace falta generar documentos, lo dejamos asentado en la herramienta de tracking.</td></tr>
</table>

desarrollo. (*) proceso: es un programa o aplicación que recupera información de una fuente de datos y los procesa de manera exitosa o con fallas.	

5. Administrar los cambios continuamente en el proyecto.

Cambios pueden surgir en cualquier momento y no sólo cambios funcionales, sino cambios ajenos a las necesidades del cliente como lo son los cambios tecnológicos, cambios en las regulaciones, etc.

RUP tiene fases que se han establecido para reducir al mínimo el costo de cambio, mientras se reduce al mínimo la capacidad para permitir el cambio. Esta es la razón por la que RUP fuerza a tener una visión general al final de la fase de Inicio, una referencia de la arquitectura al final de la fase de Elaboración, terminar los desarrollos al final de la fase de Construcción.

El uso de las herramientas adecuadas puede desempeñar un poderoso papel en la gestión del cambio y reducir al mínimo su costo.

PRINCIPIO **POSTIVO**
Todos los proyectos tienen cambios. La gente no se da cuenta de todo al comienzo, o con el tiempo ve que sería mejor hacerlo de otra forma o algún caso no estaba contemplado. Todo esto es normal, es humano, es esperable (claramente esto nunca está incluido en el precio de un producto "llave en mano"). De esto, se deduce claramente que no hay forma que un proyecto termine con el costo, el tiempo y el alcance establecido en su inicio. Considerando que los cambios son inevitables es importante administrarlos de manera eficiente. En el contexto actual los cambios deben ser bienvenidos. Para este principio es más acorde la visión de SCRUM que la de RUP. Para SCRUM un cambio es siempre bienvenido, no importa en qué fase del proyecto ocurre, en cambio RUP tiende a restringir el cambio sobre las fases finales. Una heurística personal, que varia según el cliente y el contexto, es que todo proyecto al menos se corre un 20% de la apreciación inicial. A pesar de esto, muchas veces se vive en un mundo de fantasías generando expectativas no ciertas, son las reglas de juego. No es el problema que haya cambios, de hecho, debería haber un presupuesto de tiempo y costo para cambios pre-acordado, lo importante es acordarlos y que la persona pueda reconocerlo (para esto es esencial que el usuario clave sea una persona con confianza, segura en la empresa, respaldada, y con autoridad para consumir presupuesto). El no tenerlo de esta manera lleva a una situación de stress desgastante para todas las partes. Hay que aclarar que el solo hecho de evaluar cambios lleva esfuerzo (lo que implica costo y tiempo), aunque se determine no hacerlo (¿alguien reconoce esto?). No solo hay que tener cuidado con los cambios importantes, también esta el caso de "las cositas chicas" (si es simple, no me vas a cobrar por eso). Si dejo un balde en una canilla que esta goteando, al otro día el balde esta lleno de agua. El hecho de tener que pedir presupuesto por cada cambio y conseguir autorizaciones dando explicaciones, hace que el proceso sea complejo y que por todos los medios trate de

no reconocerse cambios.	
PMM	**PN**
El tema de Cambios, se considera que es uno de los puntos más importantes, este proceso debería ser sencillo y práctico para que se lleve a cabo en toda la empresa. El proceso definido en la empresa es el *proceso de Administración de Cambios de Requerimientos*. El proceso está definido para proyectos grandes y cambios importantes. El costo de seguir el proceso para los cambios chicos es más grande que el beneficio que se obtiene.	Como política de la empresa todos los proyectos deben realizar y documentar el análisis de los pedidos de cambio, existe un proceso para esto. Considero que es una forma sencilla de mantener los cambios y las críticas son las mismas que se listaron para el principio de requerimientos ya que se utilizan casi los mismos pasos, las mismas herramientas y los mismos documentos.

6. Obtener una arquitectura funcional tempranamente

La arquitectura es la estructura del esqueleto del sistema. Diseñando, implementando y testeando la misma en una etapa temprana del proyecto, aborda los principales riesgos y hace más fácil ampliar el equipo de desarrollo e introducir miembros menos experimentados. Finalmente, ya que la arquitectura, define si el sistema es construido en bloques o componentes, le permite evaluar con mayor exactitud el esfuerzo necesario para completar el proyecto.

PRINCIPIO **POSITIVO**	
La definición de la arquitectura es importante dado que es la base de todo proyecto; es como construir una casa, antes de comenzar a construirla se debe definir la arquitectura de la misma. El desarrollo sobre una arquitectura definida conduce a obtener un producto de mejor calidad. Se supone que la arquitectura la definen personas con más experiencia. El uso de una arquitectura común dentro de la organización asegura una mínima curva de aprendizaje al cambiar los integrantes de un proyecto a otro. También permite probar la arquitectura en diferentes ambientes y entornos. Una arquitectura que nos guíe en el desarrollo nos dará una base sólida para la construcción de cualquier proyecto de software, asegurándonos la homogeneidad y el control al momento de introducir nuevos componentes.	
PMM	**PN**
No hay un proceso definido en la empresa para este principio. En nuestra empresa, nos fue mal con un sistema de Colas (proyecto nuevo), se asignó al proyecto una persona no capacitada como arquitecto, lo que ocasionó que se defina una arquitectura totalmente errónea, consecuencia de eso se tuvo que sacar de	En cuanto a la arquitectura, la organización mantiene ciertas políticas, como ser, todos los proyectos deben tender a la reutilización de arquitecturas existentes en la organización, deben establecer requerimientos de arquitectura, luego deben desarrollar y documentar la misma en forma completa y consistente con dichos requerimientos.

producción la aplicación. Como aprendizaje de esto, pudimos sacar que se paga muy caro no definir correctamente los sistemas en cuanto a la arquitectura en las etapas tempranas del proyecto. Siempre es más fácil cuando se tiene un buen diseño y una buena arquitectura.	Existe un grupo de personas dentro de la empresa que forma el grupo de arquitectura, estas personas tienen diferentes roles en distintos proyectos, cuando sale un nuevo proyecto se juntan unas semanas para determinar la arquitectura del mismo. En la empresa la mayoría de los proyectos son de finanzas (mismo fin) por lo que se trata que todos los proyectos reutilicen la arquitectura existente. Es muy bueno tener un equipo destinado a arquitectura, constantemente se están investigando cosas nuevas, además van haciendo prototipos que luego se pueden utilizar para futuros proyectos. Actualmente el 60% de los proyectos de la empresa utilizan la misma arquitectura (todos los proyectos relacionados a finanzas).

7. Construir el sistema orientado a componentes

La separación en componentes nos ayuda a reducir el impacto del cambio y también a conocer ciertamente que funciones del sistema están involucradas si se encontrase un error. Al contar con una fachada para cada componente y conocer ciertamente quienes son los clientes, es muy sencillo encontrar y documentar quienes serán afectados por el cambio. Así mismo podemos documentar y planificar las pruebas necesarias, acotando el alcance al mínimo y disminuyendo los costos asociados a la inclusión de dicho cambio.

PRINCIPIO **POSITIVO**	
Si no se desarrolla orientado a componentes los cambios pueden descontrolarse. En general en una aplicación Web se utilizan la capa de Action, Facade, otras clases y DAO. Muchas veces se confunden y se invocan métodos del DAO desde un Action. Esto es un problema dado que ante una modificación en el DAO la misma no debería afectar al Action sino que debería estar centralizado en el Facade.	
PMM	**PN**
No hay un proceso definido en la empresa para este principio. Las soluciones técnicas muchas veces tienen su rol central y suelen ser complejas por demás. Hay quienes arman Frameworks con la versión que es la única forma de reusar. Un framework debería construirse cuando uno implementa un negocio una y otra vez, y entonces conociendo el domino arma una solución genérica que soporta las soluciones conocidas. El problema ocurre cuando se tratan de	Este principio resulta positivo tanto para proyectos nuevos como de mantenimientos y mejoras, es independiente al tipo de proyecto. Lo positivo de este principio, es que construir un sistema orientado a componentes hace que sea más sencillo el cambio, lo negativo de esto es que muchas veces para poder construir un sistema orientado a componentes se buscan soluciones muy complejas y esto tampoco es bueno, pensando en la mantenibilidad del mismo.

hacer generalizaciones de "primera mano" sin mucha experiencia en el dominio, suponiendo como debería ser, haciendo configurable y extensible cosas que no lo necesitaban, que nunca se usaran y donde es probable que lo necesario no se haya pensado. Las soluciones muy configurables y genéricas, suelen ser muy abstractas y complejas, difíciles de mantener, solo valen la pena si agregan un valor suficiente en proporción a la complejidad que representa. Muchas veces se valora que se pueden cambiar el comportamiento sin tener que programar. Ocasionalmente, realizar esas configuraciones es más complejo que programar. Siempre la mejor solución es la más simple que cumple con lo requerimientos actuales, para esto hay que pensar bien antes de trabajar, siempre es mejor hacer las cosas bien de entrada que tener que rehacer.	

8. Trabajar juntos como un equipo

El desarrollo de software se ha convertido en un deporte de equipo y el enfoque iterativo enfatiza la importancia de la buena comunicación y el espíritu del mismo, donde cada miembro del equipo se siente responsable por el producto final terminado.

RUP sostiene que hay que tratar de minimizar la documentación, aconseja la comunicación personalmente, tener más reuniones, tener una comunicación directa en lugar de intercambios de e-mails y los miembros del equipo deben estar lo más cerca posible. El grupo debe tener una actitud de equipo y no individual, todos somos responsables de todo. No existen las frases del tipo: tu requerimiento está incompleto o mi código no tiene defectos.

PRINCIPIO **POSITIVO**
El trabajo en equipo hace que se trabaje en un ambiente distendido; es difícil trabajar en un ambiente competitivo o donde hay roces. Es importante que los miembros del equipo independientemente del rol que cumplan trabajen todos en pro del proyecto, tratando entre todos de llevar adelante el mismo enfrentando los problemas que puedan surgir. Este punto es muy importante, no sólo para el ámbito informático sino para todos los ámbitos laborales. Trabajar en equipo por un objetivo común genera fuerzas en todos los miembros para poner todo su empeño en lograr alcanzar el objetivo propuesto. Es importante NO hacer distinción de funciones o cargos jerárquicos, todos hacemos todo. El equipo es esencial, la base de todo. La confianza entre las partes, el estar seguro que el otro va a hacer

lo mejor posible, el que ninguno salga perjudicado (tiene que ser una relación ganador-ganador) son piezas fundamentales para generar proyectos exitosos. Los equipos son fundamentales para el éxito.

Hay que encontrar formas de contratación y trabajo que permitan al equipo completo trabajar con tranquilidad, ayudando todos al objetivo común sin tener que estar en negociaciones o fricciones permanentes, que a la larga llevan tanto esfuerzo de las partes que termina siendo todo más lento y se desvía esfuerzo de generar valor agregado a negociaciones (generando el retraso de retorno a la inversión y en el fondo termina siendo más caro por lo que se pierde en el negocio).

PMM	**PM**
No hay un proceso definido en la empresa para este principio. En mi experiencia personal los equipos que funcionaron de esta manera fueron siempre exitosos no siendo así los casos contrarios. Es muy común que existan roces entre desarrolladores y testers y es normal dado que a nadie le gusta que le marquen los errores o que le digan que algo que hizo no está bien. También, suele ser común que el desarrollador le diga al analista 'eso no se hizo porque no está especificado'. Por otra parte, también ocurre que el analista le dice al cliente 'eso no está en el caso de uso porque no me lo especificaste'. Todo esto ocurre, es real y pasa con gran frecuencia generando irritación o disconformidad en los miembros del equipo. Cada uno quiere tener la razón. El equipo esta compuesto por todas las personas involucradas en el mismo, los desarrolladores, los usuarios, los analistas, los líderes de proyecto, los gerentes, la gente de infraestructura de las compañías, el grupo de gente que se encarga de la calidad, la persona que solicitó y financió el proyecto. Un equipo es más que un conjunto de personas que trabajan juntas, es gente con diferentes roles y habilidades que se complementan y cooperan en función del objetivo. A diferencia de otras ciencias, los obreros son muy calificados y cada persona en cada decisión, hace al negocio. Las personas que trabajan "en el frente de batalla" con las sumas de sus decisiones son las que más influyen en el resultado, por lo que todos deben compartir la visión y tenerla siempre presente.	No existe un proceso definido en la empresa, pero se intenta trabajar en equipo; si bien la mayoría de las veces se logra, no siempre es así. Ya se mencionaron anteriormente las características de este equipo, en particular formo parte del equipo que está en la oficinas de la empresa, allí la mayoría de las personas son líderes, en este caso no son lideres por coordinar personas sino por llevar clientes, si bien dentro de los lideres hay referentes (funcional, técnico, etc.), todos hacemos todo, esto hace que en determinados momentos del proyecto algunas personas estén más sobrecargadas que otras. En mi caso particular (Líder de Pruebas), el hecho de tener ocho clientes y entregas bimestrales, esto quiere decir que se entregan ocho versiones (por más que sea la misma versión las configuraciones del sistema son completamente diferentes) cada dos meses, hace que esté constantemente sobrecargada de trabajo, además de probar, coordinar a los testers (asignarles tareas), registrar los defectos que se van levantando por cliente y desplegar las versiones a medida que se van solucionando los problemas detectados, realizo otras tareas (tareas de soporte, hablar con los clientes) se torna un poco desgastante por momentos. Relacionado con el punto anterior, a mi modo de ver, cuando un manager (jefe), ve que las personas le responden incondicionalmente (como es el caso de la mayoría de los líderes de este equipo), siguen asignando tareas en cambio de incorporar gente al equipo. Por otra parte el hecho de que haya tantos líderes y no existan personas que liderar también trae

Muchas veces existen diferentes empresas involucradas en un desarrollo, donde en lugar de trabajar en equipo tratando de resolver los problemas entre todos según las necesidades, cada uno destaca su trabajo y tratan de dejar en evidencia los problemas de los otros y como impacta en su trabajo. La experiencia me podría permitir asegurar que todos tienen errores, y hay que saber convivir y materializarlo como oportunidad de mejora del equipo, no hay que acusar sino ayudar a mejorar la situación. Es interesante no ser solo parte del problema, sino sumarse a formar parte de la solución.	algunos problemas aparejados, a veces, cuando no están claras las tareas que debe realizar cada una de las personas, existen roces.

9. Hacer de la calidad una forma de vida, no una ocurrencia tardía

Garantizar una alta calidad implica más que sólo el equipo de pruebas. Se trata de todos los miembros del equipo y todas las partes del ciclo de vida.

El desarrollo iterativo obliga a probar en etapas tempranas y con más frecuencia. RUP promueve las pruebas lo antes posible.

PRINCIPIO **POSITIVO**	
El tema de la calidad es algo que concierne a todos los miembros del equipo, no sólo a los testers, y se debe aplicar en todo el ciclo de vida de un proyecto.	
PMM	**PN**
No hay un proceso definido en la empresa para este principio. Es importante pero muy difícil llevarlo a cabo. Depende mucho de la actitud de cada miembro del equipo. La mayoría de las veces no se le da mucha importancia a esto. Los desarrolladores se centran en tratar de entregar algo lo más rápido posible y a nuestro criterio ahí está el error; muchos prefieren entregar algo aunque funcione mal y no esperar unos días y entregar algo de mejor calidad. Esto es porque a la mayoría no le gusta testear. A veces es mejor invertir tiempo testeando y no esperar a que el usuario encuentre y reporte los errores. Cuanto más tarde se detecta un incidente, más caro es resolverlo.	Como política de la empresa todos los proyectos deben tener un documento de Plan de Pruebas, deben ejecutar por lo menos un ciclo de pruebas que incluya la funcionalidad identificada como crítica antes de pasar a UAT (pruebas de usuario); además para cada prueba planificada se deben tener los casos de prueba con sus datos de entrada y el resultado esperado. La complejidad de nuestro proyecto está en que generalmente tenemos al menos tres versiones (una por cliente) en la etapa de testing al mismo tiempo, por este motivo tanto los casos de prueba como los ítems de construcción tienen el ambiente en el que deben ser testeados. La coordinación de esta tarea no es nada sencilla. Por otra parte, tenemos implementada una parte de

	la funcionalidad (muy poco) con testing automático que corre en todos los ambientes de integración (clientes) todas las noches. Ayuda también a la calidad el hecho de realizar revisión de a pares antes de entregar el código, lamentablemente por falta de tiempo no cumplimos con esta práctica. Por último, también utilizamos un programa que utiliza el análisis estático en busca de fallos en el código Java (Find Bugs). Como dije anteriormente este proceso de Prueba se implementó hace un atrás (diciembre 2009), las dos experiencias que tuve en las entregas bimestrales fueron desgastantes, resultó casi imposible llegar a tiempo, y si bien entregamos en fecha, los resultados no fueron buenos, tuvimos varios defectos encontrados en producción. Se hace casi imposible entregar tantas versiones el mismo día de un producto que se modifica constantemente con requerimientos de ocho clientes, la realidad es que quién desarrollo un requerimiento no testea en los ocho clientes, tampoco se estima el punto con este tiempo incluido. Actualmente (junio 2010) estamos construyendo una nueva versión muy grande del producto (cambió la regulación de las casas de bolsa mexicanas) y se implementó este proceso de manera exitosa. Se realizaron las entregas de las ocho versiones en el término de una semana (entre el lunes y el viernes, se entregaron las versiones a los ocho clientes). Esto se hizo varias veces, dado que se pactaron entregas parciales.

5.1.2 CONCLUSION DE RUP

Hay que tener presente el espíritu de RUP, los principios. **No** hay que focalizarse en producir muchos artefactos o llevar a cabo las actividades definidas en RUP.

Basándonos en el espíritu de RUP debemos adoptar solo las actividades y artefactos que nos ayudan realmente a aplicar los principios. Definiremos nuestros procesos basándonos en esto.

5.2 SCRUM

5.2.1 PRINCIPIOS DE SRCUM

En SCRUM, el equipo se focaliza en una única cosa: construir software de calidad. Se busca que los equipos sean lo más efectivos y productivos que sea posible.

Opinión personal:

Esto a veces se torna difícil dado que existen casos en los que las personas no se comprometen como lo deberían hacer o como uno esperaría que lo hagan. Todos sabemos que una vez terminado el desarrollo de un requerimiento luego hay que probarlo. Para las pruebas funcionales se deben armar varios casos de pruebas y si todos se ejecutan exitosamente, se libera el desarrollo para que pase a las pruebas de Integración.

Experiencia personal:

Se le asignó un requerimiento a uno de los desarrolladores y se le entregó también una especificación de todos los casos de pruebas que se debían ejecutar para confirmar que el mismo funciona correctamente. El desarrollador terminó el requerimiento y lo reportó al analista, luego se le preguntó si había ejecutado todos los casos y respondió que sí. Luego el analista realizó las pruebas por una cuestión de inseguridad y resultó que el segundo caso de prueba falló. Conclusión, el desarrollador sólo había probado el primer caso.

En la actualidad, donde el mercado demanda muchos recursos y como consecuencia de eso se terminan contratando estudiantes que no están avanzados en sus carreras, se hace difícil pensar en que todos deben focalizarse en construir software de calidad; con un equipo en el que todos están comprometidos es fácil y viable obtener un buen resultado, pero en la actualidad lamentablemente es muy difícil contar con un equipo en el que todos estén comprometidos y capacitados.

Por otro lado, cuando se habla de calidad, se hace referencia a calidad de software en cuanto a funcionalidad o código fuente. Lo que pasa actualmente, es que muchos no conocen el paradigma en el que programan, por ejemplo: en un paradigma orientado a objetos programan como si fuera un paradigma procedural, re-implementan funciones que ya están provistas en librerías, etc. (Como anécdota, me comentaron que para saber si un número es negativo, lo que se hizo fue transformar el número a una cadena de caracteres y verificar si el mismo empezaba con el carácter ‘-’).

SCRUM es un proceso simple pero requiere mucha disciplina para ser exitoso y realizar cada uno de los siguientes de manera muy rigurosa:

- Administración de Requerimientos
- Administración de Riegos
- Planificación y Seguimiento

Como ya mencionamos anteriormente, este es un proceso y no una metodología y se basa en los siguientes principios ágiles:

1. ***Colaboración estrecha con el cliente***

PRINCIPIO **POSITIVO**	
Independientemente de si el proyecto es Nuevo o de Mejoras y Mantenimiento, es muy importante mantener una buena relación con el cliente, de esta manera se pueden obtener mejores resultados para el proyecto. Es fundamental para las tareas de Relevamiento y Análisis de Requerimientos tener la colaboración del cliente. Este principio se asemeja al principio *Trabajar juntos como equipo* de RUP.	
PMM	**PN**
No hay un proceso definido en la empresa para este principio. La experiencia nos dice que los proyectos en los que la relación con el cliente es buena el proyecto sigue su curso normal y tiende a ser exitoso, a diferencia de los proyectos en el que la relación con el cliente es difícil, tienden a demorarse, los miembros del equipo se desgastan y no terminan bien. Tenemos un proyecto en el que se fueron alrededor de 10 personas.	

2. ***Predisposición y Respuesta a cambio***

PRINCIPIO **POSITIVO**	
Se corresponde con el principio *Administrar los cambios continuamente en el proyecto* de RUP**.** La diferencia con RUP es que para SCRUM los cambios son siempre bienvenidos, independientemente de la fase o instancia del proyecto.	
PMM	**PN**
	En contraposición se podría decir que si bien es bueno que haya predisposición y respuesta al cambio, la mayoría de las veces en los proyectos nuevos podría jugar en contra, ya que hace que se dilate la fecha de salida a producción, siempre existe algún cambio para realizar ya sea por pedido del cliente o pedidos internos, pero tampoco hay que abusar de esto.

3. ***Desarrollo incremental con entregas funcionales frecuentes***

PRINCIPIO **POSITIVO**
Se corresponde con el primer principio de RUP 'seguir un enfoque iterativo'. La diferencia con RUP es que en SCRUM cada iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad. En RUP en las fases iniciales como lo son la fase de Incepción y de Elaboración no hay software

ejecutable.	
PMM	**PN**
Independientemente del tipo de proyecto, nuevo o de mejoras y mantenimiento, se puede seguir este principio; en particular en los proyectos de mejoras y mantenimiento no se requieren las fases de Incepción y Elaboración.	

4. ***Motivación y responsabilidad de los equipos por la auto-gestión, auto-organización y compromiso***

PRINCIPIO **POSITIVO**	
Contar con un equipo motivado y responsable es bueno para todo proyecto, pero es difícil contar con ello.	
PMM	**PN**
Independientemente del tipo de proyecto, es difícil mantener a todos los miembros del equipo motivados. No todos son responsables o autosuficientes.	

5.2.2 CONCLUSION DE SCRUM

PMM (Positivo)	**PN (Negativo)**
Se está de acuerdo con lo especificado en el proceso, pero lo que lo hace difícil de implementar, en algunos casos, es que se depende mucho de las personas, de sus principios y valores.	Basándonos en la experiencias de la empresa, se hace difícil pensar en que no exista documentación o que la misma sea escasa; SCRUM no lo exige, eso hace que todo lo planteado por SCRUM parezca negativo. Además, SCRUM dependen mucho del equipo de trabajo y esto hace pensar que el proceso no se seguiría.

5.3 CONCLUSIONES GENERALES

Un proceso ágil se basa en las personas y un proceso formal en los documentos. No hay que ser extremistas, la mejor opción sería tomar lo mejor de cada uno de ellos y definir un proceso que se ajuste a las necesidades de cada proyecto según el contexto en el que será llevado a cabo.

Se considera que las prácticas planteadas por SCRUM son factibles y algunos artefactos planteados por RUP también son importantes, por lo que apuntaremos a eso en la definición del nuevo proceso.

6. DEFINICION DEL PROCESO PDSM: PROCESO DE DESARROLLO DE SOFTWARE MIXTO, COMBINANDO RUP Y SCRUM

Como objetivo del proyecto se había planteado lo siguiente:

- El proyecto consiste en definir un proceso para empresas desarrolladoras de software, que le permita llevar a cabo proyectos de manera organizada y efectiva logrando así rentabilidad en los proyectos y calidad en los productos.

- La idea es poder definir un proceso que realmente ayude a la empresa sin generar trabajo extra a los involucrados, dado que la mayoría de las veces los procesos fracasan porque generan trabajo extra o no deseado en las personas que deben seguir el proceso.

Comenzaremos por definir los siguientes términos: organizar, efectividad, rentabilidad y calidad.

Organizar [1]: Planificar o estructurar la realización de algo, distribuyendo convenientemente los medios materiales y personales con los que se cuenta y asignándoles funciones determinadas.

Efectividad [2]: Hacer lo correcto.

Rentabilidad [3]: En economía, el concepto de rentabilidad se refiere, a obtener más ganancias que pérdidas en un campo determinado.

Calidad [4]: Es la aptitud de un producto o servicio para satisfacer las necesidades del usuario.

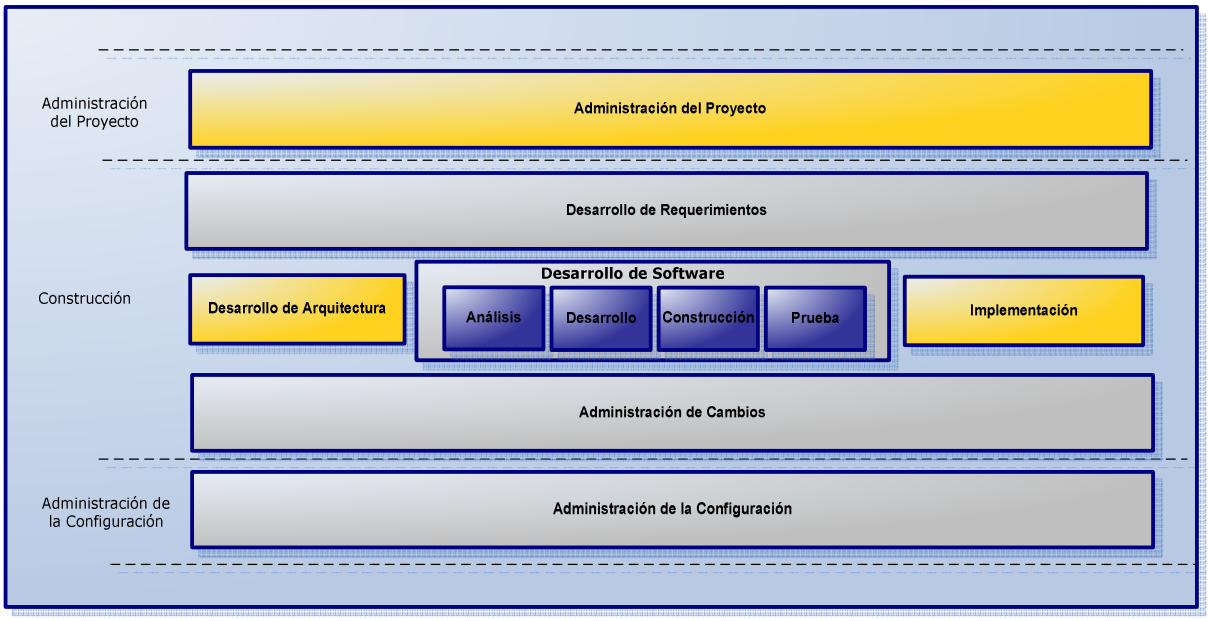
También, se puede definir como que la calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad.

La calidad del software es medible y varía de un sistema a otro o de un programa a otro. La calidad del software puede medirse después de elaborado el producto. Pero esto puede resultar muy costoso si se detectan problemas derivados de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software [5].

Habiendo comprendido las definiciones anteriores y basándonos en nuestras experiencias personales, como conclusión podríamos decir que si se trabaja de manera organizada y efectiva se tiende a aumentar la calidad del producto y tanto la organización como la efectividad son mejoradas (decimos mejoradas porque no se resuelven) con la definición de procesos. El tema de la Rentabilidad es más complejo; es difícil definir procesos que aseguren la rentabilidad del proyecto. En general los conceptos de calidad y rentabilidad van de la mano, a mayor calidad más costoso el producto. Y desde el punto de vista del

proveedor puede pasar que el producto producido sea de muy buena calidad y el cliente está muy contento pero la empresa perdió dinero.

Entonces, si la calidad es la conformidad con las especificaciones y la especificación es la relación entre requerimientos y características del producto (en nuestro caso el software) el proceso se define de la siguiente manera:



Comienza con el proceso de Administración de Requerimientos que tiene como entrada los pedidos del cliente (requerimientos) y finaliza con el proceso de Pruebas, que sería lo que indica si el producto (software) cumple con los requerimientos solicitados.

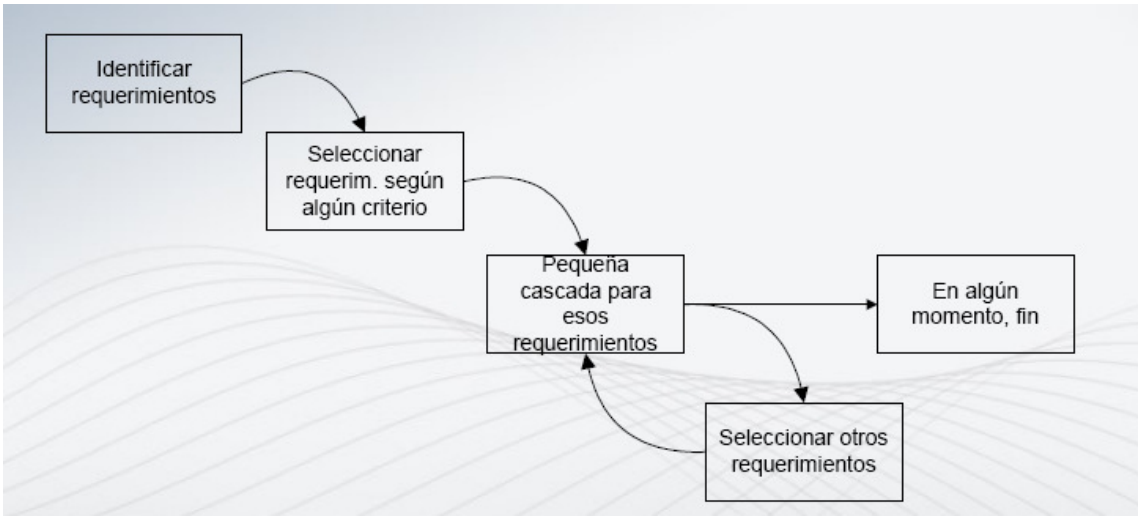
6.1 DEFINICION DEL PROCESO

Antes de comenzar a definir cada uno de los procesos vamos a plantear algunos principios o valores que consideramos importantes. Es decir, la empresa debería tener formalizado de alguna manera cuáles son sus principios o valores para que toda persona que trabaje en ella lo tenga presente en todo momento. Los principios y valores no son tareas o artefactos que puedan ser definidos en procesos, es por eso que serán planteados como principios de la empresa.

6.1.1 PRINCIPIOS DE LA EMPRESA

- *Trabajar en equipo*: trabajar juntos como un equipo . (Ver apéndice A)
- *Ser proactivo*: detectar oportunidad de mejora.
- *Tener foco en el cliente*: colaboración estrecha con el cliente.
- *Compromiso con la calidad*: motivación y responsabilidad de los equipos por la auto-gestión, auto-organización y compromiso.

A continuación definiremos nuestro proceso, que será ***iterativo*** e ***incremental***:



El modelo incremental se perfila como el "ganador de las discusiones". Las ventajas que tiene son [6]:

- El usuario ve algo relativamente rápido. Esto hace que se obtenga el feedback del usuario lo antes posible, para orientar el desarrollo al cumplimiento de sus necesidades y realizar todas las adaptaciones identificadas para cumplir con los objetivos planteados.
- Se admite que lo que se está construyendo es el sistema, y por lo tanto se piensa en su calidad desde el principio.
- Se pueden atacar los principales riesgos.
- Los ciclos van mejorando con las experiencias de los anteriores.
- El aprendizaje y experiencia del equipo iteración tras iteración, mejora exponencialmente el trabajo, aumenta la productividad y permite optimizar el proceso en el corto plazo.
- El trabajo iterativo deja una experiencia en el equipo que permite ir ajustando y mejorando las planificaciones, logrando menores desvíos en la duración total del proyecto.

El ciclo de vida organizará las tareas en **4 fases**, inicio, elaboración, construcción y transición.

Se definirá el **documento de visión**: el documento permite tener una visión general de las necesidades del cliente, del nuevo producto, servicio o resultado que se pretende satisfacer. Se plantearán los siguientes puntos.

- Objetivo del proyecto o la justificación.
- Mesurables criterios de éxito relacionados.
- Requisitos de alto nivel.
- Descripción de alto nivel del proyecto.
- Resumen del presupuesto.
- Gerente de proyectos afectados, la responsabilidad y el nivel de autoridad.

El documento de visión permite establecer prioridades cuando se tiene poco tiempo de desarrollo o cuando se deben dejar requerimientos para una segunda fase, por ejemplo. Dicho documento es definido en el proceso de Administración de Proyecto y utilizado como entrada de otros procesos, como por ejemplo, Administración de Requerimientos.

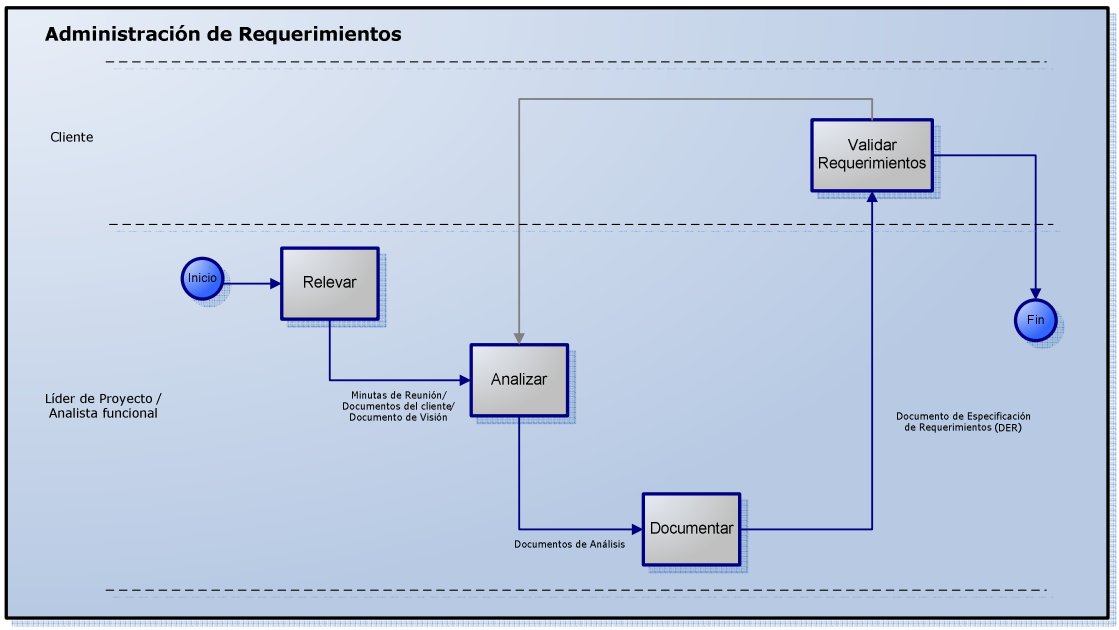
6.1.2 PROCESOS

<table>
<tr><th colspan="2">Proceso de Administración de requerimientos</th></tr>
<tr><td colspan="2">La administración de requerimientos comprende las actividades relacionadas con la definición, clasificación, asignación, seguimiento y control de los requerimientos durante todo el ciclo de vida de desarrollo de software. Es una metodología indispensable para el aseguramiento de la calidad de los productos, así como para el control y seguimiento de los proyectos [7].</td></tr>
<tr><td>Objetivo</td><td>Administrar (documentar, validar y aprobar) de manera organizada los requerimientos de los clientes, analizando si son factibles de llevarse a cabo y definir prioridades ante otros requerimientos.</td></tr>
<tr><td>Entrada</td><td>• Requerimientos del Cliente (minutas de reunión, mails, conferencias, conferencias telefónicas, documentos del cliente),
• Documento de visión,
• Documento de Especificación de Requerimientos (DER).</td></tr>
<tr><td>Salida</td><td>• Documento de Especificación de Requerimientos (DER).
• Lista de requerimientos de alto nivel.</td></tr>
<tr><td>Rol Primario</td><td>• Cliente
• Analista Funcional</td></tr>
<tr><td>Rol Secundario</td><td>Líder del Proyecto</td></tr>
<tr><td colspan="2">Tarea 1: Relevar
Descripción: Relevar las necesidades del cliente/usuario (se obtienen minutas de reunión).
El cliente plantea una necesidad al analista funcional, mediante algún medio (mail, reunión, etc.)

Tarea 2: Analizar
Descripción: Analizar la información obtenida por parte del cliente/usuario.
El analista funcional debe analizar la información obtenida y entender el problema.

Tarea 3: Documentar (DER)
Descripción: Documentar los requerimientos obtenidos de haber entendido el problema planteado por el cliente/usuario.
Se elabora el documento de especificación de requerimientos (DER). A partir del análisis realizado previamente se determina el tipo de requerimiento (funcional y no funcional).

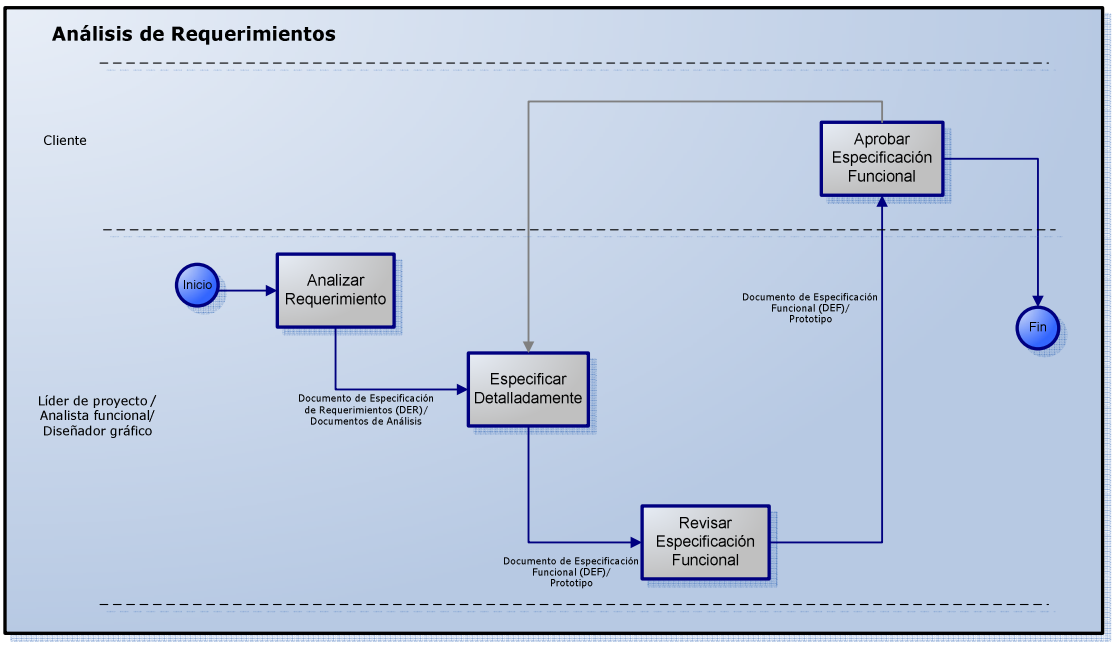
Tarea 4: Validar requerimientos
Descripción: Validar con el cliente la especificación de requerimientos para obtener su aprobación.
El analista funcional presenta (vía mail, en una reunión) la especificación de requerimientos al cliente.</td></tr>
<tr><td>Herramientas</td><td>Las herramientas utilizadas en este proceso podrían ser: Word, Excel, EA (Enterprise Architect), Mantis y el JIRA</td></tr>
</table>


Administración de Requerimientos
Cliente
Validar Requerimientos
Inicio
Relevar
Fin
Analizar
Líder de Proyecto / Analista funcional
Minutas de Reunión/ Documentos del cliente/ Documento de Visión
Documento de Especificación de Requerimientos (DER)
Documentar
Documentos de Análisis

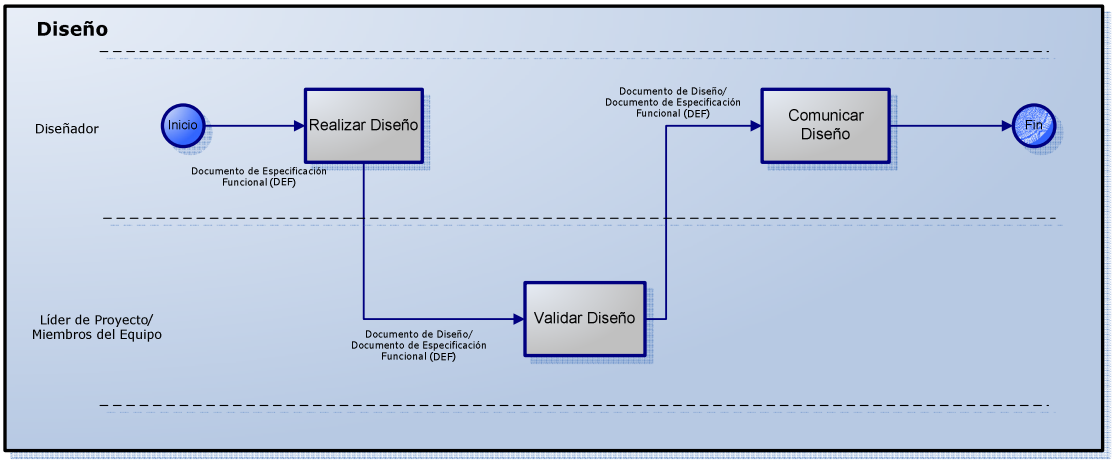
<table>
<tr><th colspan="2">Proceso de Análisis de Requerimientos</th></tr>
<tr><td colspan="2">Esta etapa se enfoca en entender los Requerimientos del cliente e intenta solucionar las deficiencias que estos puedan tener.</td></tr>
<tr><td>Objetivo</td><td>Obtener una Especificación Funcional del requerimiento.</td></tr>
<tr><td>Entrada</td><td>- Documento de Especificación de Requerimientos (DER)</td></tr>
<tr><td>Salida</td><td>• Especificación funcional (Documento de Especificación Funcional, Casos de Uso, especificación de interfaces, prototipos)
• Casos de Prueba</td></tr>
<tr><td>Rol Primario</td><td>• Analista Funcional</td></tr>
<tr><td>Rol Secundario</td><td>• Diseñador Gráfico</td></tr>
<tr><td colspan="2">Tarea 1: Analizar Requerimiento
Descripción: Analizar y Detallar la información obtenida y plasmada en el Documento de Especificación de Requerimientos (DER).
El Analista Funcional revisa toda la documentación obtenida en el proceso anterior y entiende el problema.
Tarea 2: Especificar Detalladamente
Descripción: Elaborar los documentos de especificaciones funcionales, Casos de Uso, prototipos y Casos de Prueba del requerimiento. Es aquí, donde se priorizan, agrupan, tipifican y se estima la complejidad de los mismos (previo a su especificación).
El Analista Funcional, escribe el Documento de Especificación Funcional (DEF, este puede ser un documento, prototipo, caso de Uso, etc.) además de diseñar y especificar los Casos de Prueba del requerimientos. Luego lo registra para mantener la trazabilidad con el mismo.

Tarea 3: Revisar Especificación Funcional
Descripción: Revisar la Especificación Funcional interna.
El Analista Funcional revisa la Especificación Funcional con algún Revisor, este es algún integrante calificado del equipo. De haber puntos a modificar, el Analista Funcional los realiza y vuelve a hacer la revisión.

Tarea 4: Aprobar Especificación Funcional
Descripción: Aprobar la Especificación Funcional (DEF)
Se entrega la Especificación Funcional al Cliente y se espera su aprobación.</td></tr>
<tr><td>Herramientas</td><td>Las herramientas que se pueden utilizar en este proceso son las siguientes: herramienta de edición de texto, Visio, EA, Mantis y el JIRA.</td></tr>
</table>


Análisis de Requerimientos
Cliente
Líder de proyecto / Analista funcional/ Diseñador gráfico
Inicio
Analizar Requerimiento
Documento de Especificación de Requerimientos (DER)/ Documentos de Análisis
Especificar Detalladamente
Documento de Especificación Funcional (DEF)/ Prototipo
Revisar Especificación Funcional
Aprobar Especificación Funcional
Documento de Especificación Funcional (DEF)/ Prototipo
Fin

Proceso de Diseño	
Es la etapa donde se deriva una solución partir de un problema.	
Objetivo	Encontrar la solución a un problema. Muchas veces se desprenden varias soluciones de un mismo problema y luego se debe seleccionar una.
Entrada:	• Documento de Especificación funcional (DEF)
Salida	• Documento de Diseño
Rol Primario	• Diseñador de software
Rol Secundario	• Miembros del Equipo
Tarea 1: Realizar diseño Descripción: Realizar un diseño detallado de la solución a implementar, a partir de la Especificación Funcional. El diseñador agrega el diseño al modelo aplicativo existente, verificando el acoplamiento con el resto de los componentes del sistema. Además de documentar y mantener la trazabilidad de los requerimientos. *Tarea 2: Validar diseño* Descripción: Realizar la validación del diseño. El Diseñador solicita la revisión del diseño a algún integrante del equipo (un par o el leader del proyecto). *Tarea 3: Comunicar diseño* Descripción: Comunicar el diseño propuesto al resto del equipo. El diseñador presenta el diseño al resto del equipo, esto se podría hacer por mail o mediante una presentación dependiendo la complejidad del mismo.	
Herramientas	Se puede utilizar EA, Visio para realizar el diseño y mantener la trazabilidad con los requerimientos.



Proceso de Construcción	
Objetivo	Construir el código necesario teniendo en cuenta los estándares de programación definidos en la organización o el proyecto.
Entrada	• Documento de Especificación funcional (DEF), Documento de Diseño, Casos de Prueba
Salida	• Código fuente
Rol Primario	• Desarrollador
Rol Secundario	• Miembros del Equipo

Tarea 1: Codificar

Descripción: Construir el código necesario para llevar a cabo el requerimiento.

El Desarrollador construye el código necesario a partir de la documentación brindada (Documento de Especificación Funcional, Documento de Diseño y Casos de Prueba) para solucionar el problema.

Tarea 2: Ejecutar Pruebas Unitarias

Descripción: Ejecutar pruebas unitarias para testear la funcionalidad construida.

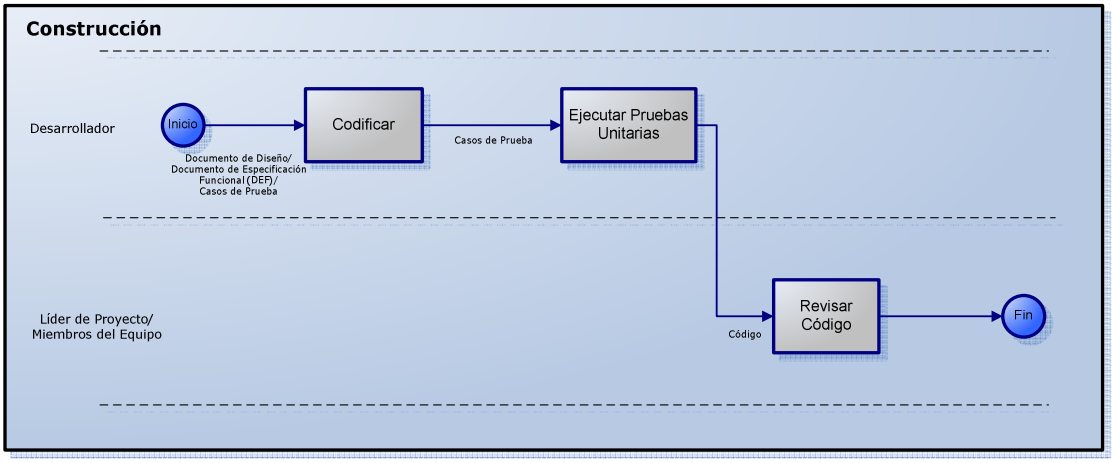
El desarrollador debe ejecutar un set de pruebas unitarias y muchas veces Casos de Prueba que se le brindan al momento de comenzar a desarrollar.

Tarea 3: Revisar código

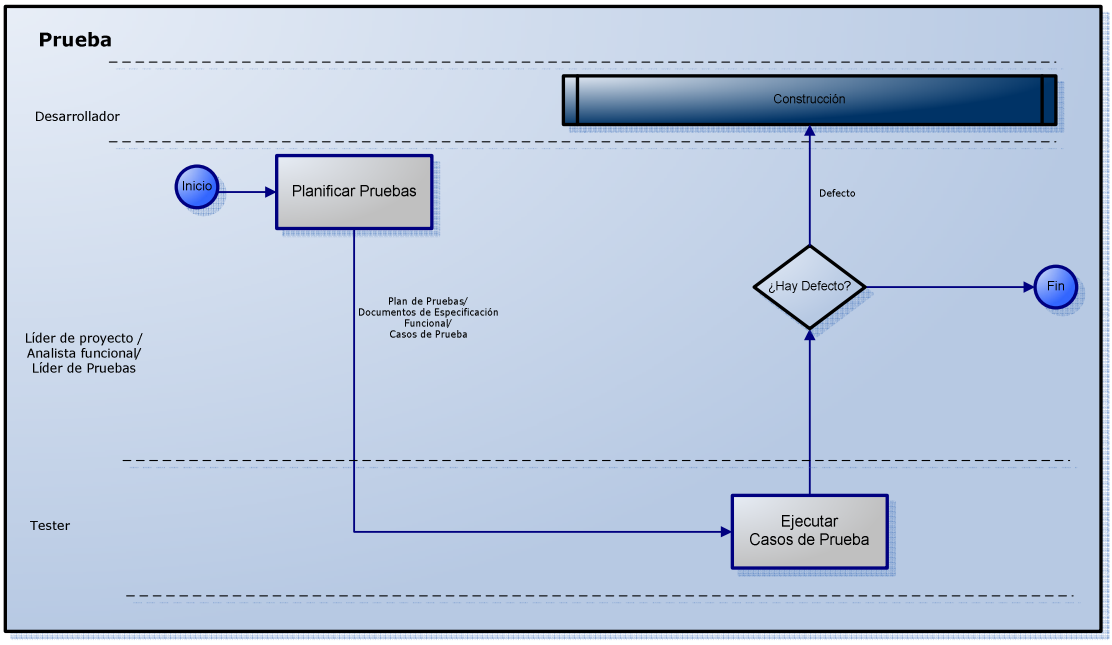
Descripción: Realizar la revisión del código.

El desarrollador una vez finalizado el desarrollo, solicita a algún par o superior la revisión del código. De surgir cosa a solucionar, se vuelve a codificar, ejecutar pruebas unitarias y nuevamente se realiza la revisión con el mismo revisor.

Herramientas	Son varias las herramientas utilizadas para desarrollar, Eclipse, Base de Datos, CVS, SVN, Herramienta de Testing (de existir casos de Prueba cargados), Herramientas de Traking (JIRA, Mantis, Track), Google.



Proceso de Prueba	
Las pruebas son los procesos que permiten verificar y revelar la calidad del producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad. Básicamente es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas.	
Objetivo	El proceso de pruebas consiste, básicamente, en la realización de una serie de pruebas al código construido para encontrar errores y solucionarlos.
Entrada	• Especificaciones Funcionales y Casos de Prueba
Salida	• Resultado de la ejecución de las pruebas
Rol Primario	• Líder de Testing • Testers
Rol Secundario	• Desarrollador
Tarea 1: Planificar las Pruebas Descripción: Define las actividades que se llevan a cabo para realizar las pruebas del sistema. El Líder del Proyecto junto con el Líder de Testing planifican las pruebas y definen el ambiente de pruebas. Luego el líder de Testing asigna las tareas y responsabilidades. *Tarea 2: Ejecutar Casos de Prueba* Descripción: Se ejecutan las pruebas de acuerdo a lo planificado. El Tester ejecuta los Casos de Prueba que le fueron asignados y registra los defectos en la herramienta de tracking, en caso de encontrarlos.	
Herramientas	Se pueden utilizar herramientas como EA (Enterprise Architect) o TestLink, para registrar los casos de prueba

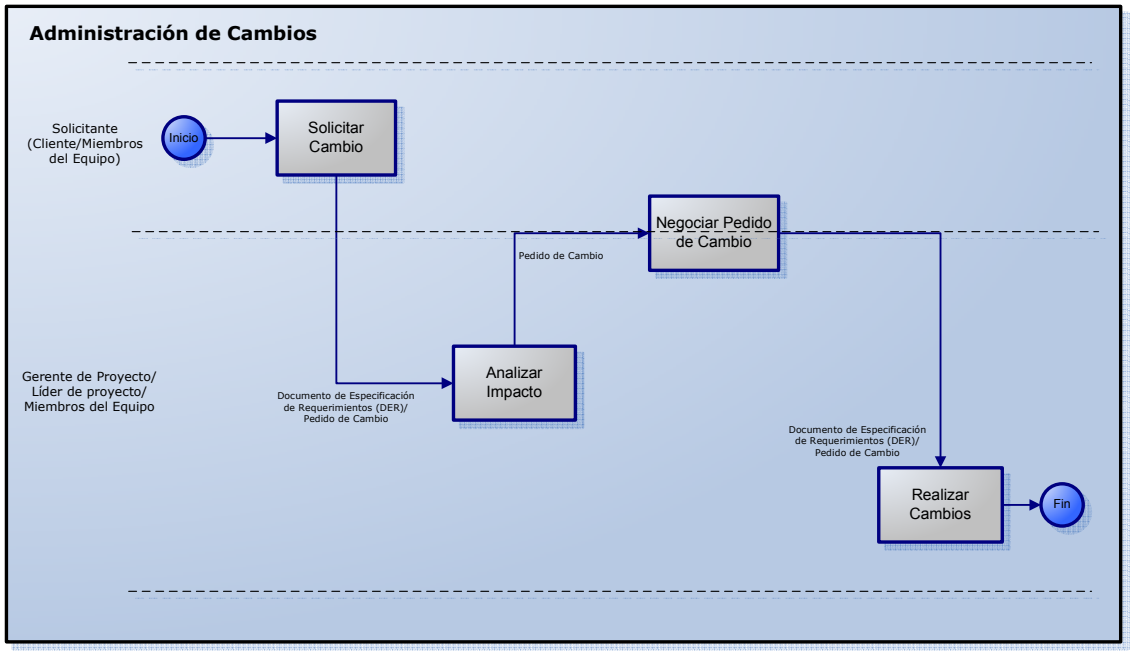


<table>
<tr><th colspan="2">Proceso de Administración de Cambios</th></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Objetivo</td><td>El Objetivo de este proceso es administrar todo cambio que surja dentro del proyecto luego de que las Baseline fueron generadas. En particular pueden ser cambios sobre los requerimientos, arquitectura, diseño, documentos formales del proyecto, código fuente, etc.</td></tr>
<tr><td>Entrada</td><td>• Pedido de cambio (minutas de reunión, mails)
• Documento de Especificación de Requerimientos (DER)</td></tr>
<tr><td>Salida</td><td>• Pedido de Cambio aprobado</td></tr>
<tr><td>Rol Primario</td><td>• Solicitante (Cliente, Miembro del Equipo)
• Equipo de análisis</td></tr>
<tr><td>Rol Secundario</td><td>• Líder del Proyecto</td></tr>
<tr><td colspan="2">Tarea 1: Solicitar cambio
Descripción: Las solicitudes de cambio a los requerimientos se registran formalmente en la DER.
Cualquier persona relacionada al proyecto, sea interna o del cliente, puede detectar la solicitud de cambio a un requerimiento; esto es registrado por cualquier integrante del equipo en la DER indicando si es un Cambio al Requerimiento o un Nuevo Requerimiento.

Tarea 2: Analizar impacto del cambio
Descripción: El pedido de cambio es analizado para evaluar el impacto y el esfuerzo.
El líder del proyecto junto con miembros del equipo analiza el impacto del cambio y las fechas, luego de esto registra el análisis realizado en el requerimiento previamente cargado DER.

Tarea 3: Negociar Pedido de Cambio
Descripción: Se negocia con el cliente los pasos a seguir.
Si el pedido de cambio no es aprobado por el cliente, se deja asentado que el cambio fue rechazado. En caso de ser aprobado se realizan los cambios necesarios.

Tarea 4: Realizar Cambios
Descripción: Se realiza el cambio-
El desarrollador en base al análisis de impacto realizado modifica todas las partes afectadas del requerimiento.</td></tr>
<tr><td>Herramientas</td><td>Las herramientas utilizadas en este proceso son similares a las usadas en el proceso de Administración de requerimientos, es decir: Word, Excel, EA (Enterprise Architect), Mantis y el JIRA.</td></tr>
</table>


Administración de Cambios
Solicitante (Cliente/Miembros del Equipo)
Inicio
Solicitar Cambio
Negociar Pedido de Cambio
Pedido de Cambio
Gerente de Proyecto/ Líder de proyecto/ Miembros del Equipo
Documento de Especificación de Requerimientos (DER)/ Pedido de Cambio
Analizar Impacto
Documento de Especificación de Requerimientos (DER)/ Pedido de Cambio
Realizar Cambios
Fin

<table>
<tr><th colspan="2">Proceso de Administración de la Configuración</th></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Objetivo</td><td>El objetivo del proceso es establecer y mantener la integridad de los Productos de Trabajo de software durante todo el Ciclo de Vida del Software.</td></tr>
<tr><td>Entrada</td><td>• Guía de Instalación
• Notas de Release del Producto
• Plan de Implementación</td></tr>
<tr><td>Salida</td><td>• Plan de Administración de la Configuración</td></tr>
<tr><td>Rol Primario</td><td>• Administrador de la Configuración</td></tr>
<tr><td>Rol Secundario</td><td>• Líder del proyecto</td></tr>
<tr><td colspan="2">Tarea 1: Planificar la Administración de la Configuración
El Líder de Proyecto junto con el Administrador de la Configuración, planifican las tareas de Administración de la Configuración.

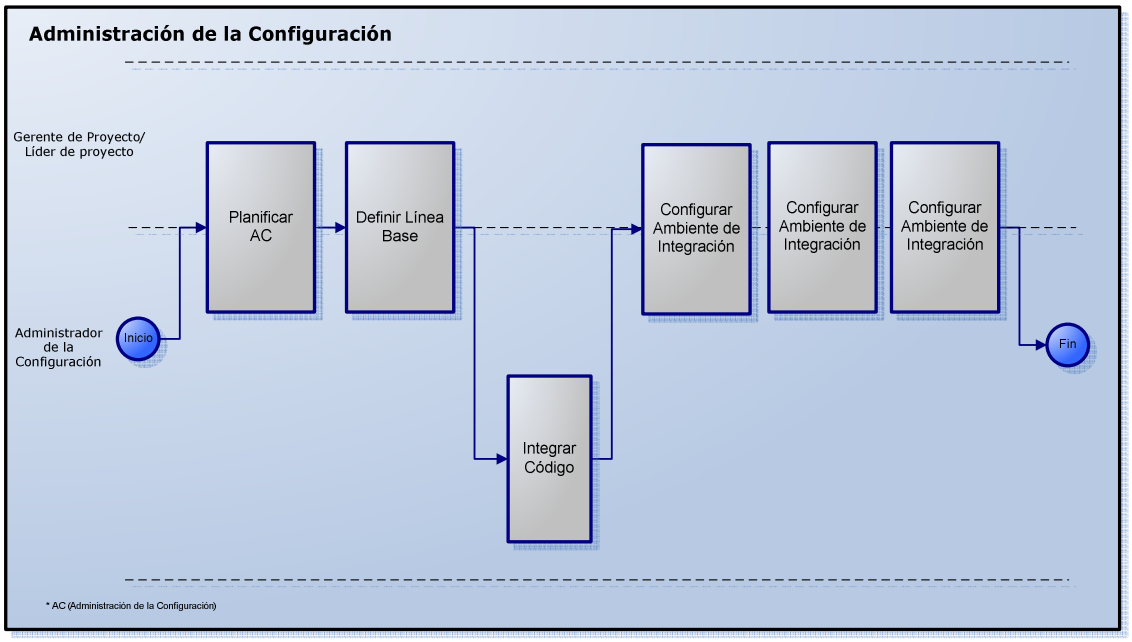
Tarea 2: Definir línea base de Planificación
Se establecen las líneas base de acuerdo al Plan de Administración de la Configuración.

Tarea 3: Integrar el código
Descripción: Se consolida el código fuente de la versión a probar.
El Administrador de la Configuración recolecta la última versión del código fuente, consolida y arma el release.

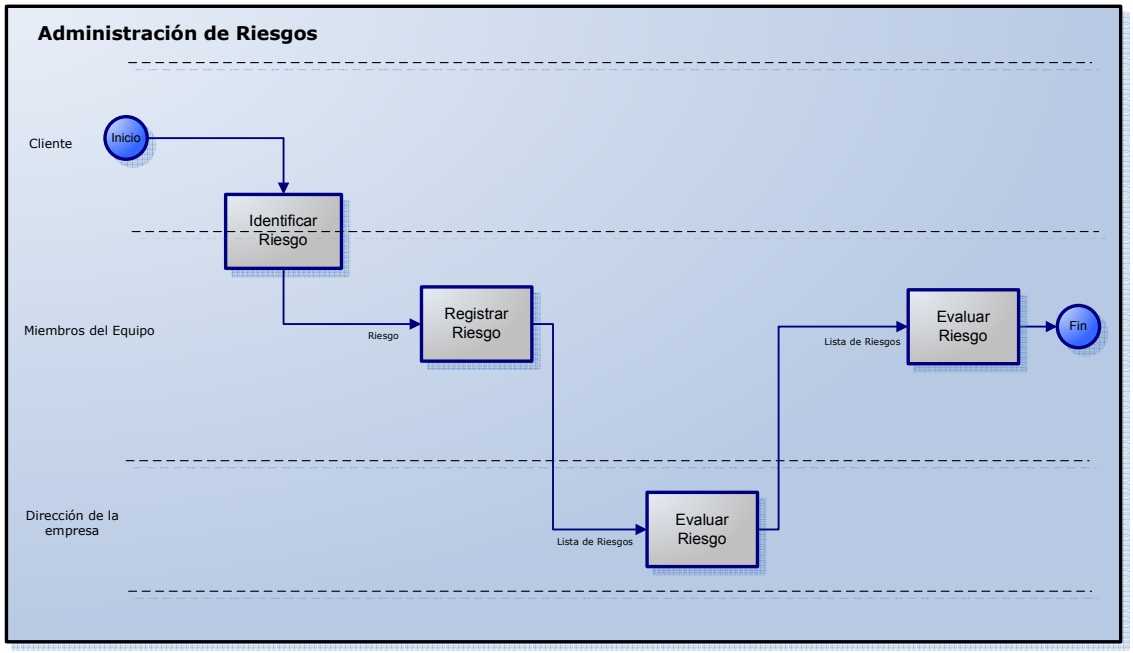
Tarea 4: Configurar ambiente de Integración
Se prepara y se genera el ambiente de integración.

Tarea 5: Configurar ambiente de UAT
Se prepara y se genera el entorno de pruebas del usuario.

Tarea 6: Configurar ambiente de Producción
Se instala el producto en el ambiente de producción.</td></tr>
<tr><td>Herramientas</td><td>Eclipse para integrar el código fuente, herramienta de tracking para actualizar los items de construcción, herramienta de repositorio de CM, como SNV o CVS.</td></tr>
</table>


Administración de la Configuración
Gerente de Proyecto/ Líder de proyecto
Planificar AC
Definir Línea Base
Configurar Ambiente de Integración
Configurar Ambiente de Integración
Configurar Ambiente de Integración
Administrador de la Configuración
Inicio
Integrar Código
Fin
* AC (Administración de la Configuración)

Proceso de Administración de Riesgos	
La gestión de riesgos en proyectos de software pretende identificar, estudiar y eliminar las fuentes de riesgo antes de que comiencen a amenazar el éxito o la finalización exitosa de un proyecto.	
Objetivo	Gestionar la exposición del proyecto a los riesgos, considerando la probabilidad de ocurrencia de los mismos y el impacto que tienen en caso de manifestarse.
Entrada	• Lista de Riesgos
Salida	• Lista de Riesgos
Rol Primario	• Miembros del Equipo • Cliente
Rol Secundario:	• Dirección de la empresa
Tarea 1: Identificar el riesgo *Tarea 2:* Registrar el riesgo *Tarea 3:* Evaluar el riesgo *Tarea 4:* Determinar la acción para mitigar la ocurrencia del riego	
Herramientas	Herramienta de tracking.



7. APLICACIÓN DEL PROCESO

Seleccionamos uno de los proyectos de una empresa de Telecomunicaciones para aplicar el proceso definido.

En toda empresa de telecomunicaciones existen diferentes áreas: Call Center, Atención al público, Precontención, etc. Nos focalizaremos en una de las problemáticas que se presentó en el área de Precontención. Esta área, es la encargada de retener a los clientes que deciden dar de baja el servicio.

Todo cliente puede tener uno o varios teléfonos y cada teléfono tiene el servicio de telefonía. Por otra parte, los teléfonos tienen dos modalidades de contratación:

- alquilados
- propios

Los teléfonos *alquilados*, son propiedad de la empresa y el cliente abona un monto de alquiler por mes, en cambio, los p*ropios* son propiedad del cliente.

La empresa cuenta con un sistema que permite dar de baja el servicio, esto se puede hacer en forma inmediata o programada para x cantidad de días más adelante.

Si el teléfono es alquilado y la baja es inmediata, el cliente debe devolver el equipo en ese mismo momento.

Si el teléfono es alquilado pero la baja del servicio es programada para más adelante, el cliente deberá devolver el equipo días mas tarde.

Uno de los problemas que tenía la empresa frente a este esquema de baja del servicio, era que muchos de los teléfonos con suspensiones programadas para x cantidad de días más adelante no eran devueltos.

Para solucionar esto, lo que se hacía era obtener un listado de dichos teléfonos y emitir una factura con el costo del mismo al cliente, este proceso se realizaba en forma manual; había una persona que diariamente se encargaba de obtener dicho listado y se lo presentaba al área de facturación para emitir las facturas.

Luego del gran problema económico que se sufrió a nivel mundial, muchos clientes no pudieron hacer frente a los gastos del servicio y decidieron dar de baja el mismo, incrementándose así significativamente el número de bajas. Una de las estrategias aplicadas por la empresa fue no suspender el servicio en forma inmediata sino programarlas para dentro de un mes más adelante.

Ante este escenario se presentó el siguiente problema, la cantidad de bajas aumentaban y como las suspensiones eran en su mayoría programadas para un mes hacia adelante, los teléfonos en alquiler y no devueltos aumentaron también. Ante esa situación, la empresa decidió desarrollar un sistema que en forma automática detecte los teléfonos en este estado y les realice la factura correspondiente.

7.1 CARACTERISTICAS GENERALES DEL PROYECTO

Fue un proyecto de tamaño pequeño, tuvo una duración de aproximadamente dos meses. Fue un proyecto que surge como consecuencia de una determinada situación económica en el país. Encuadra en el modelo de proyectos del tipo mejoras y/o mantenimiento.

7.2 CARACTERISTICAS GENERALES DEL CLIENTE

En cuanto a la disponibilidad del cliente, no hubo problemas. Se asignó una persona con perfil analista quién nos proporcionó toda la información necesaria, tomó las decisiones adecuadas, realizó revisiones sobre temas o dudas propuestos, etc. También fueron asignados usuarios claves para realizar las pruebas sobre el desarrollo y aprobaciones de prototipos.

Como en todo proyecto, hubo cambios e indefiniciones y obviamente afectó en el plan del proyecto, pero no resultó ser un problema dado que en todo momento se lo involucró al cliente para que se sienta parte del mismo. Esto ocasionó que todos estemos comprometidos y seamos consientes de que el retraso de un plan es por todos y no sólo por el equipo de desarrollo (que en general suele ocurrir).

7.3 CARACTERISTICAS GENERALES DEL EQUIPO

En cuanto a los miembros del equipo de desarrollo, el mismo se conformó de la siguiente manera:

- Un líder de proyecto
- Un analista
- Dos desarrolladores
- Un líder técnico
- Dos testers

Los miembros del equipo trabajaban diferente cantidad de horas (part-time o full-time) y no todos en la misma franja horaria. Había desfasaje de horarios, las personas que trabajan part-time lo hacían en el rango de 13 hs. a 19 hs. mientras que los que trabajan full-time lo hacían de 9 hs. a 18 hs. No todos tenían el mismo nivel de conocimiento, había pasantes, junior, semisenior y senior.

Por suerte contamos con un equipo colaborativo, organizado y muy comprometido.

En los dos meses que duró el proyecto no hubo rotaciones de personas.

Durante la aplicación del proceso se irán mencionando diferentes características que se podrían haber presentado en el proyecto planteado y que podrían afectar a las tareas del proceso aplicadas, es decir, seguramente no aplicaríamos las mismas tareas del proceso si tenemos un equipo de desarrollo pequeño

dentro de la empresa, que varios equipos de desarrollo que trabajen para el proyecto en diferentes consultoras (desarrolladores externos o desarrollo tercerizado).

7.4 APLICACIÓN DEL PROCESO DEFINIDO

Habiendo realizado una pequeña introducción sobre el proyecto a ejecutar y una breve descripción del contexto, vamos a comenzar con el desarrollo del mismo, o sea, aplicar el proceso definido PDSM.

Nombre del proyecto: Facturación de Teléfonos suspendidos y No Devueltos.

División en fases: Si bien intuitivamente y en forma secuencial se realizan los pasos definidos en cada una de las fases, no se va a formalizar esta tarea. Cuando decimos formalizar significa que no se va a encontrar en ningún documento alguna referencia a los mismos.

Definición del documento de visión: Se especificó el documento de visión en el proceso de Administración de Proyecto, que no está contemplado en la definición de este nuevo proceso. Ver el documento "*Doc de Visión Proceso Facturacion.doc*".

Nos basaremos en el siguiente principio:

El director del proyecto y el equipo del proyecto son responsables de determinar qué procesos de los Grupos de Procesos serán utilizados, quién los usará, y el grado de rigor de ejecución de esos procesos para alcanzar el objetivo deseado del proyecto.

<table>
<tr><th colspan="2">Proceso de Administración de requerimientos</th></tr>
<tr><td colspan="2">Aplicación del Proceso

Tarea 1: Relevar
Se realizó una reunión donde el cliente dio una breve descripción de la problemática a resolver. La evidencia de la misma es la minuta de la reunión. Ver el documento Minuta con el requerimiento.doc

Tarea 2: Analizar
El analista funcional analizó la información obtenida y entendió el problema.

Tarea 3: Documentar (DER)
Se agregaron los requerimientos correspondientes en la DER. Ver el documento DER Facturación Automatica.xls

Tarea 4: Validar requerimientos
Se realizó una reunión con el cliente para validar y aprobar los requerimientos de la DER.</td></tr>
<tr><td>Herramientas</td><td>La herramienta utilizada es el Mantis.</td></tr>
<tr><td colspan="2">Variante 1: Tener el cliente a distancia (en el exterior)

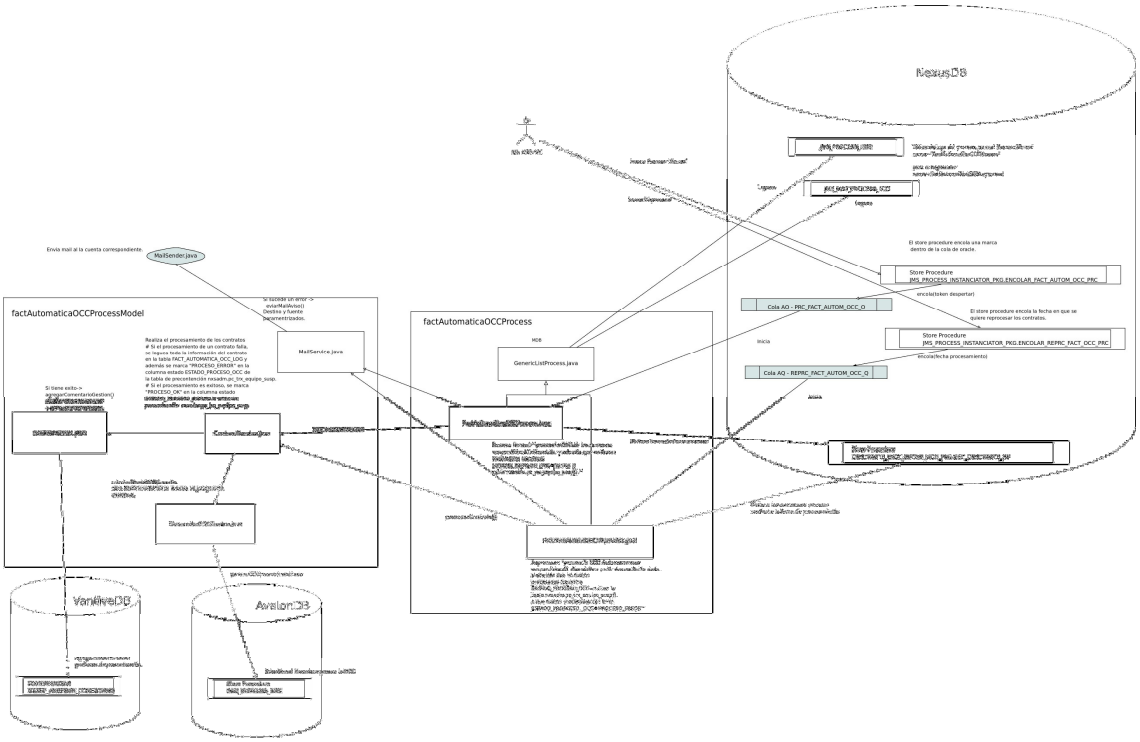
Si bien las reuniones para Relevar se pueden realizar a distancia, ya no serian de la misma forma, es decir, podrían hacerse a través de una conferencia telefónica, donde si bien se va a llegar a un acuerdo no es tan sencillo como estar frente a frente, se podría viajar al lugar donde se encuentra el cliente, esto implicaría tiempo y un costo más alto.</td></tr>
<tr><td colspan="2">Variante 2: Que el cliente sea una empresa con una corporación muy grande

En estos casos, suelen dilatarse mucho los tiempos al momento de aprobaciones, ya que estas corporaciones tienen varias áreas, donde cada una de ellas debe aprobar el documento.</td></tr>
</table>

Con estas variantes podemos ver, que si bien se podría cumplir con el proceso de Administración de Requerimientos y obtener las mismas evidencias, ya no sería tan sencillo.

Proceso de Análisis de Requerimientos	
Aplicación del Proceso *Tarea 1: Analizar el Requerimiento* El Analista Funcional revisó e interpretó todos los requerimientos de la DER. *Tarea 2: Especificar Detalladamente* El Analista Funcional escribió un documento con la especificación funcional para el ABM de los costos de los teléfonos y otro documento con el detalle técnico del proceso a desarrollar y los casos de prueba correspondientes a dicha funcionalidad. *Ver los documentos ABM costo Telefonos.doc, Especificación Técnica del proceso de facturación.doc* *Tarea 3: Revisar la Especificación Funcional* La solución fue revisada y aprobada por el líder del proyecto. *Tarea 4: Aprobar la Especificación Funcional* La aprobación del documento de especificación funcional fue realizada en forma conjunta en una reunión con el cliente. *Ver el documento Minuta aprobación de requerimiento.doc*.	
Herramientas	Word, para escribir la especificación funcional y EA para los casos de prueba.
***Variante 1**: Varios clientes* Una variante que podría presentarse, sería que el sistema sea utilizados por varios clientes, en este caso el equipo de desarrollo podría tener varios líderes de proyecto (uno por cliente) y esto haría más larga la toma de decisiones (para obtener la solución al problema) y la aprobación de la especificación funcional que presente el Analista Funcional ya que la deberían revisar todos los lideres de proyecto. Además de esto, el hecho de tener varios clientes haría que los tiempos sean más largos.	

Al igual que en el proceso anterior, si bien se podría cumplir con el proceso y obtener las mismas evidencias, existen variantes que pueden hacer este mismo proceso más complejo.

Proceso de Diseño	
Aplicación del Proceso *Tarea 1: Realizar el diseño* Se realizó un diseño detallado de la solución a implementar. Se definió un bosquejo de la solución técnica.  *Tarea 2: Validar el diseño* El diseño fue revisado y validado con el líder técnico del equipo y con el analista funcional del cliente. *Tarea 3: Comunicar el diseño* El diseño fue comunicado al responsable del desarrollo en una reunión. En dicha reunión el analista le asigna la problemática al desarrollador, es decir, le cuenta lo que se debe implementar.	
Herramientas	EA/Mantis
Variante 1: *El cliente no formara parte del equipo* En el caso detallado, el analista funcional es el cliente y se encuentra físicamente en la empresa, si el cliente se encontrara en el exterior, no podría formar parte de este tipo de reuniones, en tal caso el diseño se podría revisar y comunicar al resto del equipo.	
Variante 2: *Que el proyecto sea de un producto*	

En este caso el cliente podría no formar parte de las decisiones, sino que solicita un requerimiento y nosotros como proveedores le damos una solución, podría no enterarse de cómo fue resuelto.

Tarea 1: Realizar el diseño

Se realiza el diseño detallado de la solución al problema planteado.

Tarea 2: Validar el diseño

Se revisa el diseño con los líderes del proyecto y las personas que se consideren necesarias.

Tarea 3: Comunicar el diseño

El diseño se comunica al resto del equipo. Dependiendo de la complejidad y el impacto que tiene el diseño en la aplicación, podría presentarse a todo el equipo o sólo a la gente que se ve involucrada en la solución y muchas veces en las pruebas (para que conozcan las nuevas funcionalidades).

Proceso de Construcción	
Aplicación del Proceso *Tarea 1: Codificar* El Desarrollador construyó el código necesario a partir de la documentación brindada (Documento de Especificación Funcional, Documento de Diseño y Casos de Prueba) para solucionar el problema. *Tarea 2: Ejecutar Pruebas Unitarias* El desarrollador ejecutó un set de pruebas unitarias y algunos Casos de Prueba que se le brindaron al momento de comenzar a desarrollar. *Tarea 3: Revisar el código* No se realizó revisión de código.	
Herramientas	Eclipse

Variante 1: *Grupo de desarrollo en diferente lugar físico que el resto del equipo*

Esto podría ser: desarrollos tercerizados a consultoras o equipo de desarrollo en diferentes oficinas (por cuestiones de espacio). Bajo estas condiciones, se dificulta mucho más la tarea de codificación, en donde generalmente, por más que exista la documentación, surgen preguntas, no es lo mismo hacerlas personalmente que por mail o teléfono; muchas veces también se requieren reuniones adicionales con el equipo de desarrollo externo, que implican tiempo.

Otra problema que se podría presentar, es que no lleguen con los tiempo, entonces el desarrollador deja de ejecutar los casos de prueba en la instancia de desarrollo, lo que provoca que en el 99% de los casos se detecten errores y el desarrollo tenga que volver, esto es muy costoso en cuanto a tiempos.
Este problema también puede ser provocado por el Analista Funcional, donde muchas veces por cuestiones de tiempo, escribe los casos de prueba después de haber asignado el desarrollo para que se comience a hacer, esto hace que se empiece a codificar sin tener el panorama completo, lo que provoca re-trabajo y errores detectados en etapas tardías.

En el escenario planteado, es muy importante la revisión de código, más que nada, porque al ser tercerizado el desarrollo, la gente generalmente, no tiene el contexto completo del cliente y suelen introducirse errores de lógica de negocio.
En particular se detectaron mucha menor cantidad de errores las veces que se realizó la revisión de código. Para esto contamos con un plug-in que nos permite obtener el código commiteado por requerimiento, y se puede hacer en forma remota, el desarrollador commitea y desde las oficinas locales se realiza la revisión de código, si se detectan errores permite volver para atrás el desarrollo y sino pasa a la etapa de pruebas de integración.

Proceso de Prueba

Aplicación del Proceso

Tarea 1: Planificar las Pruebas

Se definió el ambiente de test como ambiente de prueba. Se asignaron tres personas para realizar esta tarea. Dos personas para que prueben el ABM de costos y una para que pruebe el proceso de facturación.
La actividad consistió en ejecutar los casos de prueba y documentar los resultados.

Tarea 2: Ejecutar Casos de Prueba

Se ejecutaron las pruebas de acuerdo a lo planificado. Cada Tester ejecutó los Casos de Prueba que le fueron asignados y los defectos fueron registrados en el Mantis. *Ver los documentos Casos de pruebas ABM costo Telefonos.doc, Casos de pruebas Proceso Facturacion Automat.doc, ejecucionCasoPruebaProcesoFactAutomat20100224.doc y DER Facturacion Automatica.xls*

Herramientas	Mantis

Variante 1: *Varios clientes del sistema, con diferentes configuraciones y que esto impacte de diferente manera en la funcionalidad incluida.*

En ese caso, el proceso de pruebas se dificulta ampliamente, ya que no tendríamos un único ambiente de test, sino uno por cliente (para poder realizar las pruebas de la nueva funcionalidad con las diferentes configuraciones del sistema).

Ante este escenario, se podría tener una persona encargada de la fase de pruebas del sistema, la cual planificaría las misma para cada uno de los ambientes (diferentes clientes), luego una persona (podría ser la misma) que sea encargada de armar los distintos ambientes de test, la cual recolectaría la última versión del código y armaría el ear para cada uno de los clientes y por último los testers (ya no podría ser uno solo) que ejecutarían los casos de prueba.

Como se puede notar, las tareas realizadas en este proceso serían las mismas (Planificar, Integrar, Configurar y Ejecutar), pero sería mucho más compleja la organización:

- Se requiere más gente focalizada en tareas específicas
- Planificar mucho más la fase de pruebas
- Mantener diferentes ambientes de prueba al mismo tiempo

Tener mayor control en las pruebas, porque pueden surgir errores en algunos ambientes y en otros no, esto hace que se deba analizar mucho más los cambios a realizar para solucionar el error detectado.

Proceso de Administración de Cambios	
Aplicación del Proceso *Tarea 1: Solicitar el cambio* Los cambios fueron solicitados de manera informal a través de chat y mails. Los mismos fueron registrados en el mantis y en minutas. *Ver el documento DER Facturacion Automatica.xls, Minuta solicitud de cambio 20100308.doc, Minuta solicitud de cambio 20100309.doc y Minuta solicitud de cambio 20100323.doc* *Tarea 2: Analizar el impacto del cambio* El pedido de cambio fue analizado para evaluar el impacto y el esfuerzo. Si bien el impacto no fue importante en cuento a esfuerzo sí afectó en la fecha. El cambio fue detectado el mismo día en el que estaba planificado salir a preproducción. Se retrasó el pasaje a preproducción. *Tarea 3: Negociar el Pedido de Cambio* El análisis del cambio fue aprobado por el cliente. *Tarea 4: Realizar los cambios* Se realizó el cambio.	
Herramientas	Mantis
Variante 1: *Varios Clientes* Si se tuviesen varios clientes, el cambio solicitado por uno de ellos, debería analizarse en cada una de las diferentes configuraciones. El proceso de Administración de Cambios sería el mismo, sólo que se requiere un mayor esfuerzo en la tarea de Analizar el Impacto.	

Proceso de Administración de la Configuración

Aplicación del Proceso

Tarea 1: Planificar la Administración de la Configuración

Las tareas de Administración de la Configuración son planificadas por el Analista Funcional del cliente.

Tarea 2: Definir línea base e Planificación

Las líneas base de acuerdo al Plan de Administración de la Configuración las establece el Analista Funcional del cliente. Nosotros establecemos la línea base de nuestros documentos y el código fuente.

Tarea 3: Integrar el código

Los encargados del desarrollo son quienes recolectan la última versión del código fuente y arman el release en test (es el commit del ear en el cvs). Recolectan los scripts de base de datos y la información de todas las configuraciones necesarias para dejar lista la aplicación en test.

Tarea 4: Configurar ambiente de Integración

El código del producto lo integramos nosotros y se lo entregamos al cliente. El ambiente de integración está a cargo del cliente.

Tarea 5: Configurar ambiente de UAT

El cliente prepara y genera el entorno de pruebas del usuario.

Tarea 6: Configurar ambiente de Producción

El cliente instala el producto en el ambiente de producción.

Herramientas	

Variante 1: *Las tareas se realizan en el Proveedor y no en el Cliente*

Dependiendo de cómo se hayan pactado las entregas con el cliente, estas tareas podrían realizarse en el proveedor; a continuación se detalla cada una de ellas:

Tarea 1: Planificar la Administración de la Configuración

Las tareas de Administración de la Configuración son planificadas por el encargado de administrar la configuración.

Tarea 2: Definir línea base e Planificación

Las líneas base de acuerdo al Plan de Administración de la Configuración las establece el Administrador de la Configuración (integrante del equipo) y el resto del equipo adapta todos los documentos relacionados a la versión con esta línea base.

Tarea 3: Integrar el código

El encargado de integrar el código, recolecta la última versión del código fuente y arma el release. Luego recolecta los scripts de base de datos correspondientes a la versión t los ejecuta en la base de datos te integración.

Tarea 4: Configurar ambiente de Integración

Una vez finalizados los desarrollos, se integra el código, se configura el ambiente de integración (base de datos, número de versión, etc.) y se despliega para realizar las pruebas de integración.

Tarea 5: Configurar ambiente de UAT

Se configura el entregable según lo detalla cada cliente y se entrega para que se realicen las pruebas de usuario

Tarea 6: Configurar ambiente de Producción

Se configura el entregable según lo detalla cada cliente para el ambiente de producción y se definimos una fecha de instalación en conjunto con el cliente.

Proceso de Administración de Riesgos
Aplicación del Proceso En nuestro proyecto los riesgos se registran en el informe de avance y son notificados al cliente en las reuniones de avance. *Tarea 1:* Identificar el riesgo *Tarea 2:* Registrar el riesgo *Tarea 3:* Evaluar el riego *Tarea 4:* Determinar la acción para mitigar la ocurrencia del riego

8. RESULTADOS OBTENIDOS

Hemos aplicado el proceso en varios proyectos, lo que ha permitido adquirir experiencias y realizar mejoras continuas al mismo. En líneas generales, los resultados obtenidos fueron los siguientes:

- **La reducción de los artefactos producidos disminuyó la sobrecarga de trabajo**. Se focalizó solamente en producir los artefactos relevantes para el proyecto, es decir, aquellos que son entradas para otros procesos. La no obligación de generar todos los artefactos definidos alivianó la tarea cotidiana. Muchas veces no hizo falta definir un caso de uso porque alcanzó con un diseño.

- **Las herramientas utilizadas agilizaron las tareas de gestión**. Las herramientas de Tracking utilizadas para la administración de requerimientos simplificaron las tareas de creación, búsqueda, asignación, seguimiento, control y cierre, además de permitir manejar con facilidad todo el ciclo de vida de un requerimiento. Estas herramientas fueron mucha utilidad en los momentos donde surgían dudas o inconsistencias con el cliente, lo que está definido en la herramienta de Tracking es lo que en principio vale. Ayudaron también para organizar o hacer un seguimiento de las distintas iteraciones; la utilización de las herramientas disminuyeron considerablemente el tiempo de gestión en los proyectos.

- **El proceso mixto logró ser incorporado en varios proyectos en poco tiempo**. Como el proceso definido es un proceso que no requiere trabajo extra sino que cuenta con lo mínimo e indispensable para que un proyecto sea organizado no costó convencer a la gente de su utilización. Luego de aplicar el proceso, se observó que muchos de los proyectos desorganizados lograron organizarse; eso significa que disminuyeron las horas extras, aumentó la comunicación entre los miembros del equipo, disminuyeron los tiempos de desvío, algunas relaciones complicadas con los clientes mejoraron y el software entregado mejoró en general su calidad.

- **Mejoró la administración de los tiempos.** Al definirse procesos para las etapas de definición, desarrollo, pruebas y transición, hizo que mejoraran los tiempos dado que se aplicaron al momento de hacer las estimaciones.

- **Aumentó la motivación del equipo**. Al reducirse las horas extras y el trabajo bajo presión por falta de tiempo hizo que las personas trabajen más relajadas. La administración de los requerimientos hizo que su asignación y seguimiento sea más organizada.
  El seguir un proceso, muchas veces hace que las personas tengan responsabilidades asignadas, se sientan más comprometidas con el resultado y esto los motiva.

- **Mejoró la Calidad**: Contar con un proceso de Pruebas, hizo que disminuyeran considerablemente la cantidad de defectos que llegan al cliente, con esto mejoramos la imagen al cliente, le damos confianza y un software de mayor calidad. Por otra parte, realizar entregas en forma temprana ayudó a que el cliente pueda ir probando la aplicación y detecte los problemas e inconsistencias tempranamente. Cuanto antes se detectan los problemas en el ciclo de vida de un proyecto es mejor; el enfoque iterativo permite ir mejorando a lo largo del proyecto.

- **Mejoró la Planificación**: El hecho de planificar las etapas y seguir un proceso, hizo que se organizaran las entregas y en el caso de tener que entregar el producto a más de un cliente no fue un caos. Además, la planificación de iteraciones pequeñas permitió disminuir los tiempos de desarrollo.

- **Reducción de tiempo de Consultoría**: Requerimientos: Si bien el hecho de especificar detalladamente los requerimientos provocó mayor trabajo en los analistas funcionales, fue de gran

utilidad; es notable como se redujeron los errores por parte de los desarrolladores. Además de esto se redujeron los tiempos de consultoría, es decir, anteriormente llamaban muy seguido para consultar cosas, que hoy están detalladas en las especificaciones, diseños y casos de prueba.

9. TRABAJO FUTURO

Dentro de las futuras líneas de este trabajo sobre el proceso definido se pueden destacar las siguientes:

- Analizar la posibilidad de desarrollar una aplicación para apoyo del proceso definido.
- Continuar con la mejora de los procesos con el objetivo de aumentar la calidad del producto.
- Implementar herramientas de apoyo para los procesos definidos (herramientas de tracking, administración de casos de pruebas, etc.)
- Aplicar los procesos a otros tipos de proyectos y analizar los resultados.
- Tomar como modelo otros procesos tanto formales como agiles y plantear alternativas.

10. TRABAJOS RELACIONADOS

En una búsqueda bibliográfica, se han encontrado algunos trabajos que comparan los procesos RUP y SCRUM en distintas circunstancias y para diferentes fines.

BIKO, Information Technology Advisors [8] en *Buenas prácticas de gestión en empresas de servicios avanzados*, muestra el resultado de aplicar técnicas de metodologías ágiles de desarrollo en una empresa con certificación CMMI nivel 2 en un entorno corporativo de Banca. Dado que CMMI en su nivel 2 no especifica nada sobre las metodologías de desarrollo y gestión de equipo, ni del proceso concreto de creación de software, se decide ir en busca de técnicas para el mejor control del desarrollo para abordar dichos temas.

Por otra parte, Juan Palacio [9] en *Gestión de Proyectos: formal o ágil?*, plantea las características relevantes, no sólo del proyecto sino también de la organización que las va aplicar, que se deben tener en cuenta al momento de elegir un proceso formal o ágil para obtener mayores beneficios en la gestión.

CoMakeIT, [10] en *White paper on the Agile process: RUP and Scrum*, describe brevemente el proceso RUP y cómo puede integrarse con SCRUM. Básicamente demuestra cómo pueden convivir ambos procesos, RUP y SCRUM, en un mismo proyecto.

Remi Armand Collaris y Eef Dekker [11] en *Scrum and RUP - A Comparison Doesn't Go on All Fours*, presenta según la experiencia del autor, como RUP y SCRUM no se enfrentan sino que se complementan entre sí. Esto lo hacen mostrando en primer lugar Qué malas interpretaciones de los procesos RUP y

SCRUM están detrás del punto de vista que ambos procesos se enfrentan y en segundo lugar, Qué evidencia hay para sustentar que ambos procesos tienen vistas complementarias.

Por último, Santiago D'Andre, Miguel Martinez Soler y Gabriela Robiolo [12] en *RUP versus Scrum, una comparación empírica en un ámbito académico*, realizan una comparación de ambos procesos, RUP y SCRUM, mediante un experimentó formal comparando el desarrollo de un mismo producto –juego de estrategias por turno- aplicando en forma paralela estos dos métodos en un contexto académico. Los resultados obtenidos no permiten afirmar taxativamente que un método es mejor que otro, sino que resaltan las características particulares de cada uno.

11. CONCLUSIONES

En un mundo de cambios constantes y competencia global, las organizaciones de desarrollo de software son presionadas a alcanzar mayor eficiencia con menores costos. Para poder lograr este objetivo, es necesario adoptar una forma de trabajo que permita entender, controlar, comunicar, mejorar, predecir y certificar el trabajo realizado.

Actualmente existe una gran diversidad de opciones relacionadas con procesos de desarrollo. Constantemente se escuchan diferentes acrónimos como CMM, CMMI, RUP, ISO, PSP, TSP, etc., que causan confusión, principalmente debido a la mala interpretación de los mismos.

Las organizaciones de desarrollo de software, además de entender los principales modelos y procesos existentes, deben considerar varios factores para poder decidir qué camino seguir, como: tamaño de la organización, recursos, enfoque en mercado global o local, habilidades, etc.

Nada asegura el éxito de la ejecución de proyectos, pero disminuye la probabilidad de fracaso. Lo más importante, más allá de conocer técnicas y herramientas, es la "Experiencia".

No significa que el conocimiento, las habilidades y los procesos descriptos deban aplicarse siempre de manera uniforme en todos los proyectos. El director del proyecto y el equipo son responsables de determinar qué procesos de los Grupos de Procesos existentes serán utilizados, quién los utilizará, y el grado de rigor de ejecución de los mismos para alcanzar el objetivo deseado.

La definición de procesos no es estática, es decir, va variando en el tiempo. Existe el proceso de mejora continua de los procesos; con la experiencia uno va aprendiendo y puede determinar qué cosas se seguirán aplicando de la misma manera, porque se obtuvieron buenos resultados y cuáles se van a mejorar. Los procesos son mejorados proyecto tras proyecto.

Los artefactos son importantes a futuro, en el momento actual, donde la rotación de los recursos es frecuente, es importante tener documentación para realizar más rápidamente la transferencia de conocimiento y las capacitaciones. Si se da la situación, que en un mismo momento ingresan más de dos personas al equipo, el hecho de que no exista documentación tiene un costo alto en tiempo en capacitación. Por otro lado, cuando surge algún problema es fácil revisar el histórico para entender qué paso.

12. APENDICES

12.1 APENDICE A

Grupo: personas que trabajan juntas.

Equipo personas: personas relacionadas significativamente en orden para cumplir objetivos compartidos.

Característica de un equipo exitoso: una de las claves para el éxito de un equipo es que cada integrante esté *motivado*.

12.2 APENDICE B

- **¿Qué es la motivación?**

"Son las fuerzas internas dentro de una persona, que causan que ella, voluntariamente, realice un esfuerzo extra, en una manera directa y específica para obtener un objetivo"

Técnicas de motivación

M *Manifest*: manifieste confianza al delegar.
O *Open*: mantenga comunicación siempre abierta.
T *Tolerance*: tolere con paciencia los errores.
I *Involve*: involucre a todos los participantes.
V *Value*: valore esfuerzos, reconozca rendimiento.
A *Align*: alinee objetivos proyectuales e individuales.
T *Trust*: confíe en su equipo y sea confiable.
E *Empower*: habilite a los miembros del equipo.

- **Gestionar el equipo de proyecto**

Objetivo

Controlar el *rendimiento* de los *miembros del equipo* del *proyecto* con el objetivo de mejorarlo.

¿Cómo lo logra?

- Haciendo un seguimiento del rendimiento de los miembros del equipo.
- Proporcionando retroalimentación.
- Resolviendo polémicas.
- Coordinando cambios.

Técnicas y Herramientas

Observación y Conversación.

Evaluaciones de rendimiento del Proyecto.

- Objetivos:
  - Verificar roles y responsabilidades.
  - Retroalimentación positiva hacia los integrantes del trabajo.
  - Descubrir problemas ocultos o latentes.
  - Desarrollar planes de capacitación complementarios.
  - Establecer objetivos tangibles y mensurables para el trabajo a realizar.

- **¿Qué es un conflicto?**

Se refiere a cualquier situación en la cual existen objetivos, pensamientos o emociones incompatibles, dentro o entre los individuos o grupos, que conducen a desacuerdos u oposiciones.

¿Por qué son inevitables en un proyecto?

Porque existen metas personales, objetivos que compiten entre sí en la empresa, requerimientos de recursos y diferentes visiones, que deben ser integradas para satisfacer los objetivos generales del proyecto.

Para la **visión contemporánea:**

- El conflicto es natural e inevitable, pudiendo tener un efecto negativo o positivo, por lo tanto lo acepta.
- Puede ser beneficioso para el proyecto, el Gerente del Proyecto debe gerenciarlo efectivamente, antes que suprimirlo o evitarlo.
- Lo alienta para lograr superar los desafíos del cambio, y promover la innovación.
- Es necesario para mejorar el desempeño.

- **¿Definir un proceso de Software?**

Definir un proceso de software implica precisar los objetivos, las personas (roles) involucrados, las entradas y salidas del mismo, los criterios de entrada y salida, las actividades, los métodos y las herramientas que se utilizarán, la manera como se medirán elementos dentro del proceso que permitan verificar resultados, etc.

13. GLOSARIO DE TERMINOS

Ambiente de integración: Son las herramientas de soporte necesarias para acoplar las diferentes componentes de un producto, puede incluir software y equipos necesarios.

Artefacto: Resultado útil de un proceso. Puede incluir archivos, documentos, productos, partes de un producto, servicios, descripciones de proceso, especificaciones y facturas [Chr06].

Casos de prueba: Conjunto de condiciones, las cuales forman la base informativa para que el analista determine si el requerimiento está satisfecho por el Sistema.

Calidad: La capacidad de un conjunto de características inherentes de un producto, de los componentes del producto o de un proceso para satisfacer los requerimientos impuestos por los clientes [Chr06].

Proceso:

Definición 1: Un conjunto de pasos que ayudan a resolver un problema. Estos pasos deben estar definidos de manera que no sean ambiguos, esto implica que puedan ser fácilmente entendidos y capaces de ser seguidos de manera consistente por cualquier persona que lo utilice. Además un proceso debe tener sus entradas y salidas claras y bien definidas.

Definición 2: Es un conjunto de tareas interrelacionadas, coordinadas e integradas para lograr uno o varios resultados.

Definición 3: Los procesos de software comprenden un conjunto de actividades, tanto técnicas como administrativas, que son necesarias para la fabricación de un sistema de software.

Definición 4: Se define como Proceso al conjunto ordenado de pasos a seguir para llegar a la solución de un problema u obtención de un producto, en este caso particular, para lograr la obtención de un producto software que resuelva un problema.

Definición 5: Los procesos de software comprenden un conjunto de actividades que interactúan, tanto técnicas como administrativas, que son necesarias para la fabricación de un sistema de software.

Proyecto: Es un conjunto de actividades interrelacionadas y coordinadas; la razón de un proyecto es alcanzar un objetivo dentro de los límites que impone un presupuesto y un lapso de tiempo previamente definidos.

Pruebas de Regresión: Las pruebas de regresión consisten en a realizar pruebas ejecutadas previamente, con el fin de asegurar de que los cambios hecho en el códigono ha provocado efectos laterales no deseados.

Roles: Roles de un proceso se refiere al conjunto de responsabilidades y funciones que deben ser realizadas por los actores del proceso.

RUP (Rational Unified Process): Proceso de desarrollo de software unificado. Provee un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización. El objetivo es asegurar productos software de alta calidad que satisfagan las necesidades de los usuarios finales, dentro de los tiempos y presupuestos previstos [Rat03].

Trazabilidad: Una asociación distinguible entre dos o más entidades lógicas, como requerimientos, elementos del sistema, verificaciones, tareas.

14. AGRADECIMIENTOS

Con este trabajo se cierra una etapa muy importante para nosotras, que quizás se haya dilatado demasiado. Por eso queremos agradecer especialmente a todos aquellos que nos ayudaron y nos dieron su apoyo para llegar hasta acá.

A Claudia, nuestra directora, por guiarnos, ayudarnos y motivarnos en todo momento brindándonos lo necesario para concluir el trabajo. Gracias por la paciencia y la dedicación.

EVA

A Flor, que a pesar de las situaciones difíciles supo sobreponerse y me acompañó en todo momento de principio a fin. Mates, bizcochitos y facturas de por medio hizo siempre alegres las reuniones de trabajo.

A Rodrigo, por contribuir aportando su experiencia laboral.

A mi familia y amigos, que gracias a sus perseverantes insistencias hemos logrado terminar la TESIS.

Gracias a todos.

FLOR

Gracias a Eva, sin su ayuda, su paciencia y su perseverancia, no hubiéramos logrado llegar hasta acá. Gracias por esperar y no darse por vencida.
Gracias Evita ☺

A Rodrigo, gracias por sus aportes y su colaboración, y en lo personal, gracias por acompañarme.

A mi mamá y a mi papá, a quienes les estoy eternamente agradecida por haberme guiado, acompañado y ayudado en este camino, sin ellos no hubiera sido posible.

Al resto de mi familia y amigos que de alguna u otra forma colaboraron e hicieron posible que logremos concluir ésta etapa.

Gracias a todos**,** por haber insistido ☺

15. BIBLIOGRAFIA

Libros:

- The Rational Unified Process Made Easy. A Practittioner's Guide to the RUP. Per Kroll, Philippe Kruchten. Addison Wesley. ISBN 0-321-16609-4
- Agile Software Development with Scrum, Ken Schwaber and Mike Beedle. Prentice hall, upper Saddle River, New Jersey 07458.
- Una historia de guerra Ágil. Scrum y XP desde las trincheras. Cómo hacemos SCRUM. Escrito por Henrik Kniberg. Prólogos de Jeff Sutherland y Mike Cohn. InfoQ Enterprise Software Development Series. Versión online gratuita. http://infoq.com/minibooks/ scrum-xp-from-the-trenches. 2007 C4Media Inc. Todos los derechos reservados. C4Media, editor de InfoQ.com. Editor Jefe: Diana Plesa, Portada: Dixie Press, Composición: Dixie Press, Traducción al castellano: Ángel Medinilla, Datos de catalogación de la Biblioteca del Congreso: ISBN:978-1-4303-2264-1, Impreso en los Estados Unidos de América

Papers

- Ochoa, Sergio. SSP: A Simple Software Process for Small-Size Software Development Projects. Publicado en: IFIP International Federation for Information Processing series. ISSN: 1571-5736. Volume 219. Special issue on Advanced Software Engineering: Expanding the Frontiers of Software Technology, (Boston:Springer) (2006).
- Richard F. Paige, Priyanka Agarwal, Phillip J. Brooke: Combining Agile Practices with UML and EJB: A Case Study in Agile Development. Genova, Italy, May 25-29, 2003 Proceedings. Lecture Notes in Computer Science 2675 Springer 2003, ISBN 3-540-40215-2. pgs. 351-353
- Patricio Letelier, José Hilario Canós Cerdá, Emilio A. Sánchez:
  An Experiment Working with RUP and XP..
  Extreme Programming and Agile Processes in Software Engineering, 4th International Conference, XP 2003, Genova, Italy, May 25-29, 2003 Proceedings. Lecture Notes in Computer Science 2675 Springer 2003, ISBN 3-540-40215-2. pgs. 41-46
- René Noël, Gonzalo Valdes, Marcello Visconti, Hernán Astudillo. Deconstructing Agile Processes: Would Planned Design Be Helpful in XP Projects? In IEEE International Conference SCCC 2008. Chile. (2008)

Material de Interes:

- dotNet Team Agil, distendido y eficiente, Snoop Update 08. www.update08.org

- Universidad de Palermo, mayo 2008. Presentación de un caso ágil. Nicolás Paez. Snoop Consulting.
- Seminario Taller. SCRUM: Gestión ágil de proyectos de software. Laboratorio de Calidad en Tecnología de la Información Polo Tecnológico de Rosario – INTI. Buenos Aires, noviembre de 2007, Fabián Longhitano.
- http://oeguzman.googlepages.com/
- http://es.wikipedia.org/wiki/Scrum
- http://www.baufest.com/spanish/scrum/scrumconference2006/Que_es_scrum.pdf
- http://www.microsoft.com/argentina/pymes/documentos/scrum.mspx
- http://140.99.29.241/images/uploads/IntroToScrum.pdf
- http://www.rational.com.ar/herramientas/rup.html
- http://es.wikipedia.org/wiki/RUP
- http://hanzcocchi.net/el-rol-del-analista-en-rup/
- http://www.concytec.gob.pe/clubciencias/index.php?option=com_content&task=view&id=677&Itemid=375
- http://infogdssistemas.spaces.live.com/blog/cns!5B9ED81A701B41D2!139.entry
- http://codeticainge.googlepages.com/guiaing.pdf
- http://codeticainge.googlepages.com/guiaing.pdf
- http://snoopdotnet.wordpress.com
- [1] http://www.wordreference.com/definicion/organizar
- [2] http://www.monografias.com/trabajos74/conceptos-terminos-admin-empresas/conceptos-terminos-admin-empresas2.shtml
- [3] http://es.wikipedia.org/wiki/Rentabilidad
- [4] http://es.wikipedia.org/wiki/Calidad_de_software
- [5] http://www.bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.htm
- [6] http://scruz334.blogspot.es/1193793960/

- [7] http://www.monografias.com/trabajos57/modelo-calidad-cmmi/modelo-calidad-cmmi2.shtml
- http://www.sg.com.mx/content/view/4/11/
- [8] http://najaraba.com/agil/porpropiaexperiencia-Biko.pdf
- [9] http://www.navegapolis.net/files/s/NST-004_01.pdf
- [10] http://www.comakeit.com/download/agile_process_rup_scrum.pdf
- [11] http://www.scrumup.eu/AgileRecord_01_Scrum_and_RUP.pdf
- [12] http://www.39jaiio.org.ar/node/85