Universidad de las Ciencias Informáticas

Facultad 3

Titulo: "Predictor: Sistema de descarga y procesamiento automatizado de patentes. Rol de Líder"

Trabajo de Diploma para optar por el Título de Ingeniería en Ciencias Informáticas

AUTOR

Vlamir Rodríguez Fernández

TUTOR

Ing. Rolando Sacher Camacho Pupo

Ciudad de La Habana, junio del 2007

"Año del 49 aniversario de la Revolución"

Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: *la voluntad*.

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Dirección de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Autor Tutor

DEDICATORIA

A mis padres por educarme y darlo todo por mí.
A mi hermano por siempre estar ahí para discutir.
A mi novia por apoyarme en estos años difíciles.
A mis amigos.
A todos las personas que han intervenido en mi preparación.

AGRADECIMIENTOS

Antes que nada aclarar que puede olvidárseme alguna persona a la hora de escribir este documento, pero lo más importante es que los llevo a todos en mi corazón, por haber intervenido en algún momento de mi formación como profesional y como persona.

A mi madre por su ternura, dedicación y años de preocupación por mi distancia, y además por creer en mi en todo momento.

A mi padre por ser mi ejemplo de tenacidad, honradez y esfuerzo, por ser mi Pepe Grillo en los momentos difíciles.

A mi hermano, por ser siempre mi motor impulsor para crear un ejemplo a seguir.

A mi novia Marieta por soportarme durante tres años, por ser mi amiga fiel e incondicional, y brindarme su amor.

A todos mis amigos del apartamento 2304.

A todos los amigos que me han acompañado y ayudado durante estos cinco años.

A mis compañeros del proyecto por sus gran ayuda y cooperación.

A todos los profesores que han aportado a mi preparación como profesional.

A mis enemigos, porque sin ellos no hubieran barreras que superar.

A todos y cada uno de los que me brindaron aunque sea un minuto de su tiempo.

Resumen

El presente trabajo pretende mostrar el desarrollo del líder del proyecto Delfos, para el desarrollo de un producto destinado a la Oficina con el mismo nombre. La cual se dedica, dentro de otras tareas, al estudio bibliográfico de patentes, para realizar ejercicios de inteligencia de mercado, perfiles estratégicos y análisis de tendencias; proceso realizado actualmente de forma manual.

El proceso de desarrollo de software requiere la mayor cohesión entre todas las partes interesadas, factor clave por lo cual es necesario estudiar las competencias y actividades realizadas por un líder dentro de un proyecto informático, para poder desarrollarlas dentro del mismo y aplicar los métodos y herramientas que permitan el funcionamiento correcto del equipo, como son los planes utilizados para regir el proceso y las herramientas utilizadas para el control de los mismos.

Se utilizo para el desarrollo del proceso de software una metodología cuyo hilo conductor es el Proceso Unificado de Software, además de utilizar las ideas de organización planteadas por los procesos: Proceso de Software Personal (PSP) y Proceso de Software en Equipo (TSP).

La aceptación por parte del cliente de los productos entregados certifican los resultados obtenidos durante todo el proceso de desarrollo el cual permitió que se planificara, controlara y remediara todas actividades, problemas y estrategias trazadas desde un comienzo.

Índice

Introducción

El gran desarrollo de la TICs a partir de los años 90, le ha proporcionado al hombre una de las herramientas más increíbles que ha podido tener durante toda la historia de la humanidad. Las relaciones humanas, comerciales y de negocios están soportadas hoy, en su mayoría, sobre la existencia de computadoras y redes de computadoras que nos permiten intercambiar una cantidad de Información increíble.

En nuestro país hemos visto la necesidad de avanzar en la informatización de la sociedad, paso definitivo para poder colocar a nuestro país dentro del espectro mundial, y establecer mayores relaciones de colaboración con el resto del mundo.

Por lo antes expuesto surge la necesidad de automatizar muchos sectores de la sociedad cubana, dentro de los que esta el sector de los servicios. La Consultoría Empresarial Delfos, oficina adjunta al Ministerio de Informática y Comunicaciones (MIC), forma parte de este sector y brinda asesoría a diferentes clientes en el área de lo negocios cumpliendo función de Observatorio Tecnológico. Una de sus tareas es el procesamiento y análisis de patentes. El uso de las patentes como indicador de innovación es decir como indicador de la creación de algo novedoso ha sido estudiado exhaustivamente y ha alcanzado un nivel de madurez y automatización muy alto. No a todas las invenciones es de carácter obligatorio patentarlas, pero haciéndolo se obtienen la posibilidad de protegerse contra imitaciones y además también sirve para obtener retribuciones por conceptos de licencias y acuerdos de transferencia.

Las patentes no se otorgan tan pronto como se hace la solicitud. Pero como regla general siempre se otorgan antes de que salgan los productos al mercado, por esta razón se consideran que las estadísticas de patentes siempre están más actualizadas que las de productos o mercado. Estas son documentos muy valiosos ya que poseen mucha información desde el punto de vista legal, técnico y comercial.

Las patentes son de gran importancia para establecer las estrategias de negocios de una compañía o políticas de desarrollo de los países menos desarrollados. Se considera que más del 80% de la información aparecida en las patentes no aparecen publicadas en otro tipo de documentos, por lo que constituyen una fuente ideal para la vigilancia tecnológica.

El volumen de trabajo asumido por la oficina es muy grande y sus trabajadores realizan la búsqueda y descarga de patentes en bases de datos de Internet de forma manual, lo cual retrasa mucho su trabajo.

Para darle solución a este problema los directivos de la oficina se acercaron a la dirección de la universidad, primero para utilizar estudiantes en la búsqueda y descarga de patentes de forma manual pero con mayor personal, luego después de un estudio se le presenta a ellos la posibilidad de realizar un software que automatizara la descarga y procesamiento de las patentes, surge “Sistema de descarga y procesamiento automatizado de patentes (Predictor)”; el cual les permitiría ganar en velocidad de respuesta a los pedidos de sus clientes y una forma más eficiente y organizada de procesar toda la información que brindan la patentes, teniendo en cuenta que existen diferencias entre las bases de datos de donde se obtienen, Predictor permite una organización estándar teniendo en cuenta parámetros de obtención que generalizan los criterios de todas las bases de datos.

Debido esta situación se conformo un equipo para el desarrollo de la aplicación. En la formación de este proyecto se tuvieron en cuenta los roles propuesto por Proceso Unificado de Desarrollo (RUP), el cual enuncia algunos como: arquitecto, diseñador, analista, ingeniero de prueba, planificador, líder entre otros; este ultimo juega dentro del equipo un papel fundamental para lograr un desarrollo satisfactorio del producto. Él debe ser una profesional con pasión por la excelencia, con el constante deseo de cumplir y exceder las expectativas del proyecto. El líder evita tomar decisiones por si solo, sino que facilita a su equipo la toma de la mejor decisión en conjunto, proporcionando la información necesaria. Es el principal responsable de la experiencia global (buena o mala) que los actores del proyecto viven durante el desarrollo del mismo.

Un líder de proyecto es una persona con carácter, lo que significa, un conjunto de cualidades internas entre las que se encuentran: Atención, Autoconocimiento, Integridad, Respeto, Veracidad, Iniciativa, Tenacidad, Lealtad, Obediencia. Responsabilidad, Puntualidad, Optimismo, Autodisciplina, Seguridad, Humildad, Sinceridad, Confianza, Serenidad y otras cualidades adicionales**.** (JODOCHA 2005)

El líder de proyecto debe de garantizar, desde la reunión de inicio del proyecto hasta su término, que todos los actores sepan:

- Qué se decide

- Qué se acuerda
- Qué se debe realizar
- Qué y cuándo se entrega

El seguimiento de un proyecto no se debe descuidar ni un sólo día. El líder de proyecto se puede apoyar de bitácoras, minutas, etc. de acciones que le permitan trazar el avance del proyecto: ¿Qué se ha hecho?, ¿Qué falta por hacer?, etc. Así se podrá identificar claramente qué hay que hacer, por qué, cómo, quién lo hará y cuál es la fecha límite de cada una de las actividades.

El problema a resolver dentro del proyecto, para que se obtenga el producto que necesita la Oficina es el siguiente:

- El mal desempeño del rol de líder es uno de los factores que conlleva a que el equipo de trabajo no logre elaborar el software con la calidad y el tiempo requerido por el cliente.

El desarrollo del trabajo parte de la hipótesis:

- Si no se realiza un a buena labor de selección del capital humano, una estimulación al trabajo en equipo y control de la tareas asignadas a los integrantes del proyecto por parte del Líder, entonces el proyecto no podrá desarrollarse con la máxima calidad, ni cumplir con las metas propuestas dentro de los planes establecidos.

El *objeto de estudio* done se enmarca este trabajo así como el *campo de acción* son en este mismo orden:

- Las acciones y competencias del Rol de Líder dentro de un proyecto
- Las acciones y competencias del Rol de Líder dentro de un proyecto informático.

Por ello este trabajo tiene como objetivo:

- Establecer y aplicar métodos y herramientas para incentivar y controlar el desempeño correcto de los implicados dentro del proyecto Delfos, logrando la construcción de una aplicación final con calidad y que satisfaga al cliente.

De este objetivo, podemos obtener algunos más específicos como son:

1. Desarrollar las competencias y actividades dentro del proyecto.
2. Aplicar métodos y herramientas que permitan el funcionamiento del equipo.
3. Obtener la aplicación final con calidad debido a los objetivos anteriores.

Para podar llevar a cabo lo antes expuesto hay que cumplir una serie de tareas:

1. Estudiar las competencias y actividades que realiza un líder de proyecto.
2. Estudiar técnicas para la compenetración, control y evaluación para el trabajo de grupo.
3. Estudio de planes estándares para la evolución del desempeño del proyecto.
4. Estudiar los métodos de descargas de las patentes.
5. Estudiar de la estructura de las base de datos de patentes en Internet.
6. Analizar el sistema de trabajo de la oficina Delfos.
7. Implementación de un producto que satisfaga las necesidades del cliente.

Este documento esta estructurado en: introducción, 2 capítulos, conclusiones, recomendaciones, bibliografía y anexos. Los capítulos están divididos de la siguiente manera:

- Capítulo I: Fundamentación Teórica. Aborda temas fundamentales para el entendimiento de este trabajo de diploma. Se realiza una descripción del estado actual del proceso. Se realiza igualmente una descripción del estado actual del desempeño de los líderes de proyectos a nivel global. Además de la fundamentación de los objetivos, metodologías y herramientas utilizadas en el desarrollo del software.
- Capítulo II: Este capitulo presenta muchas de las tareas realizadas por el líder del proyecto Delfos, las técnicas, artefactos y herramientas utilizadas para el control, seguimiento y desempeño dentro del proyecto, además de la evaluación de los planes desarrollados dentro del proceso de construcción del software y la valoración del estado actual del proyecto.

Capítulo I: Fundamentación Teórica.

1.1 Introducción

Debido a la gran importancia que tienen las patentes para establecer las estrategias de negocios de una compañía o políticas de desarrollo de los países menos desarrollados y a la consideración de que más del 80% de información aparecida en las patentes no aparecen en otra publicación, por lo que constituyen una fuente ideal para la vigilancia tecnológica. Además de permitir conocer que patentes están vigentes en el mercado es fundamental antes de desarrollar nuevos productos que luego pudieran ser bloqueados y perderse mucho dinero. Esto hace indispensable para la Oficina Delfos la utilización de una herramienta que le permita hacer mucho más efectiva y veloz la descarga y procesamiento de la información aparecida en las patentes. La creación de una herramienta que les permita a ellos realizar esta tarea, tiene como primicia la existencia en el mundo de otras herramientas con estas funciones pero que presentan sus diferencias y especificaciones lo cual limitan o imposibilitan su uso.

Para el desarrollo de este producto se conformo un equipo en la Universidad de las Ciencias Informáticas, siguiendo las diferentes metodologías y estándares a nivel mundial. Donde el líder, uno de los factores claves para el avance del proyecto, tuvo que adaptar a su proyecto las metodologías, procesos y herramientas útiles para el desarrollo del mismo.

1.2 Objetivos de la Organización

La oficia Delfos es una Consultoría Empresarial, que tiene dentro de sus tareas, el análisis de la patentes de productos, la cual aportan gran información a la hora de una decisión empresarial y a nivel de país sobre el desarrollo o comprar de un producto determinado. Los clientes de la oficina se acercan a ellos con la necesidad de realizar estudios sobre determinadas tecnología o producto, las patentes pueden analizarse de varias maneras, ejemplos:

- Patente por tipo de inventor.
- Patente por tipo de empresa o grupos de empresas.

- Patentes por registro por uno o más campos de la tecnología.

Para mayor profundidad ver el artículo “La información de marcas como indicador de innovación tecnológica“, de Rolando González Hernández, licenciado en Química, Responsable del Grupo de Vigilancia Tecnológica de la Consultoría del Ministerio de la Informática y las Comunicaciones, Delfos.

1.2.1 Flujo Actual de los Procesos

Específicamente el estudio de patentes es una de las áreas, como se había explicado anteriormente, que realizan en esta oficina, los clientes de la oficina llegan haciendo una solicitud sobre un producto en específico (para el software criterios de búsquedas), en la oficina se hace la descarga y procesamiento de la información que pueden brindar las patentes existentes sobre ese producto. Las patentes son descargadas manualmente y luego se clasifica toda la información llevándola a un formato estándar, para con posterioridad utilizando herramientas como el Procite o el Microsoft Excel, útiles para hacer estudios estadísticos y económicos para poder entregarle al cliente un resultado que le provea de datos para saber si producir o comprar dichos productos.

El área critica del proceso es las descarga y procesamiento de la información de las patentes par dejarlas listas para hacer el estudio estadístico y económico, ya que es un proceso muy lento donde deben intervenir muchos especialistas para obtener muy poca información.

1.2.2 Sistemas Automatizados existentes vinculados al Campo de Acción

En concordancia con esto se han desarrollado varias herramientas informáticas que permiten la descarga y análisis de las patentes.

La existencia de herramientas como:

Matheo Patent (Francia): software que presenta una primera licencia €600/año, este realiza una búsqueda, recuperación y análisis de las bases de datos USPTO y Esp@cenet.

PatentHunter (EUA): software que presenta una suscripción anual USPTO 69 USD y EPO 99 USD. Además un paquete profesional con una precio de 349 USD. Busca, descarga y maneja patentes de USPTO y EPO.

PM Manager (Corea del Sur): software gratuito, pero tiene una suscripción mensual de 200 USD y anual de 2000 USD a WIPSGLOBAL v.4. Busca, descarga y analiza patentes de USPTO, EPO, PCT, Japón y Corea.

Predictor presenta soluciones semejante, en lo que a búsqueda y descarga se refiere, a todos los sistemas mencionados anteriormente, pero aplicadas a las necesidades cubanas, ya que en el país no existe ningún sistema que realice esta función. Como se observa la adquisición y/o utilización de estos software es muy costoso. Además se logra la unión en un solo sistema de una mayor cantidad de bases de datos de patentes libres.

1.3 Tendencias y Tecnologías Actuales

1.3.1 Equipo de Desarrollo de Software

1.3.1.1 ¿Qué es el trabajo en equipo?

Según la literatura el trabajo en equipo implica que todas las personas involucradas estén orientadas hacia una meta común, logrando la sinergia que les permitirá llegar más rápido y mejor que si cada uno se reparte un segmento del trabajo. Es decir, el compromiso y la identificación son requisitos fundamentales para el trabajo en equipo.

Además se define que el Trabajo en Equipo se contrasta con el trabajo grupal, mientras en el primero todos se sienten responsables por la meta común; en el trabajo grupal la tendencia es a fragmentar la responsabilidad, y cada persona se hace responsable de su comportamiento.

Pero la conclusión fundamental a la que se llego es que cuando se tiene la actitud para trabajar en equipo, es cuando verdaderamente se tiene la capacidad de crear valor ya sea para la organización, para la sociedad o para algún proyecto que tenga como meta común.

El trabajo en equipo brinda muchos beneficios entre ellos:

- Disminuye la carga de trabajo, ya que varias personas colaboran.
- Se obtienen mejores resultados.
- Se desarrolla el respeto y la escucha.
- Permite organizarse de una mejor manera.
- Mejora la calidad del comercio interno y externo.

1.3.2 Líder de Proyecto o de Equipo.

1.3.2.1 Competencias del Líder

El líder debe ser una profesional con pasión por excelencia, con el constante deseo de cumplir y exceder las expectativas del proyecto. Un líder de proyecto es una persona con carácter, lo que significa, un conjunto de cualidades internas entre las que se encuentran:

Comunicación: Se podría denominar como intercambio de información entre un emisor y un receptor.

Esta información puede tener distintas dimensiones:

- Escrita y oral, oída y hablada.
- Interna y externa.
- Formal e informal.
- Verticales (hacia arriba y hacia abajo en la organización) y horizontales (con nuestros pares en la organización).

Un líder en cualquier posición dentro del proyecto tiene que dominar toda forma de comunicación que le fuese posible, de esta forma puede comunicar de forma correcta los objetivos planteados.

Negociación: La negociación implica conferenciar con algún otra persona para llegar a un acuerdo y obtener algún beneficio del mismo.

Un líder debe saber negociar para poder realizar de forma eficiente la visión y los objetivos planteados. Los objetivos de negociación pueden ser muchos, éstos son los más frecuentes.

- Alcance, costo y objetivos del programa.
- Cambios de alcance, costo o programa.
- Términos contractuales y condiciones.
- Designaciones.
- Recursos.

Solución de problemas: Los problemas son figurita repetida en todo programa, proyecto y organización. Un líder tiene que saber enfrentar los problemas y guiar al grupo a través de ellos hasta lograr una solución integral.

Un líder, tiene que detectar los síntomas de un problema antes que este ocurra y de esta forma estar preparado para cuando este se presente.

Para esto tiene que tomar decisiones, estas decisiones tienen que ser adoptadas por todo el grupo para optimizar la acción a realizar.

Influenciar a la organización: Poder influenciar en el grupo significa que este haga las cosas que se tengan que hacer. Para lograr esto es necesario conocer las estructuras formales e informales de la organización.

Además en el libro Ingeniería de Software, un enfoque práctico Pressman plantea que un líder debe tener capacidad de motivación, organización de ideas e innovación, pero además enuncia también otros puntos clave que debe tener un buen líder de proyecto como son:

- **Resolución del problema**: tener la capacidad de poder diagnosticar los aspectos técnicos y de organización más relevantes y ser lo suficientemente flexible a la hora de tener que cambiar de solución si los primeros intentos son fallidos.
- **Dotes de gestión**: un líder debe tener confianza para asumir el control de cualquier problema que surja dentro de su equipo.
- **Incentivos por logros**: un líder debe saber estimular los esfuerzos e iniciativas y demostrar que no se penalizar a ningún miembro por cometer errores controlados.
- **Influencia y construcción de espíritu de equipo**: el líder debe motivar a todos los miembros a que son parte de una gran familia donde cada uno contribuye al desarrollo y el trabajo del otro.

Lo planteado por Pressman es casi una guía sobre las competencias que debe poseer o formarse en un líder que dirija un proyecto informático cualidades a las cuales hay que añadir muchas otras de índole personal fundamentalmente que se deben poseer, como: respeto, iniciativa, tenacidad, responsabilidad, puntualidad, autodisciplina, seguridad, humildad, sinceridad, confianza, serenidad entre otras.

Todas estas competencias que debe poseer una persona para poder desempeñar dicho rol son el resultado de todos los objetivos que deben cumplir a la hora de enfrentarse a un equipo.

1.3.2.3 Objetivos de un líder dentro del equipo de desarrollo.

Un líder dentro de un proyecto debe tener en cuenta todos estos objetos aquí enumerados para la buena conformación del mismo:

O 1. Construir un equipo efectivo y funcional.
O 2. Motivar a los miembros del equipo y mantener un avance agresivo en el proyecto.
O 3. Ayudar a resolver alguna dificultad que pueda existir entre los miembros del equipo y funcionar como facilitador.

O 4. Mantener informado al cliente del progreso del equipo, además de ser el que logre una efectiva comunicación de los clientes con los miembros del equipo.

O 5. Tramitar todas las dudas o inconformidades que tengan los miembros del equipo con la institución u organización a la cual pertenecen.

1.2.2.4 Dificultades a las que se enfrenta un Líder.

Generalmente la causa de fracaso de un proyecto de software esta asociada a dificultades con el equipo de desarrollo y mala organización. Los mayores fracasos están ligados a la incapacidad del equipo de software de manejar situaciones de presión durante el desarrollo del producto (HUMPHREY 1999).

El exceso de presión y la mala organización en el desarrollo de un producto de software puede ser fatal.

- Que exista un error a la hora del resultado por el equipo de desarrollo y que las relaciones entre los miembros del equipo se debiliten.
- Que se piense que los problemas y dificultades son insolubles incrementando la tensión del proceso.
- Insuficiente estimación de costo y tiempo que conlleva al incumplimiento de compromisos.
- Que el equipo asuma soluciones técnicas pobres y atajos que en ocasiones afectan significativamente la calidad del producto final. Que con frecuencia llevan al software a soluciones parciales que deben ser corregidas transformándose en atrasos.
- Deficiencia en el levantamiento de requisitos, generándose numerosos y grandes cambios en momentos cercanos al despliegue aumentando los riesgos del proyecto.

A todos estos problemas se enfrenta un líder que no haya logrado jugar su rol de manera efectiva dentro del equipo. Humphrey plantea en su libro Introduction to the Team Software Process, una estrategia para el manejo de la presión personal es la aplicación de técnicas de gestión de compromisos y la planificación del tiempo.(HUMPHREY 1999)

Para el manejo de la presión en la dimensión de equipo se sugiere:

- Las tareas a hacer por cada miembro sean claras y distintivas que el trabajo del equipo y sus objetivos estén explícitamente definidos.
- El equipo es limpiamente identificable, con esto se refiere a que cada uno de los miembros del equipo conoce a todos los demás miembros y sus respectivos roles.
- El equipo tiene control sobre sus tareas, con esto se refiere a que los miembros del equipo conocen el alcance del trabajo, como lo hacen, cuando lo hacen y cuando terminan. Los miembros conocen que ellos son los responsables del trabajo y controlan los procesos que rigen su trabajo, participa en la toma de decisiones.

Estas ideas planteadas en el TSP ofrecen una idea clara de las dificultades que puede acarrear la presión, sobre el equipo de desarrollo, para el líder de proyecto. Para todos los líderes de proyectos es muy importante tener en cuenta la estrategia para el manejo de dicha presión, más aun si los líderes son personas jóvenes con muy poca experiencia y necesitadas de mantener un orden en el proyecto para que este no avance sin mantener un control.

1.2.2.5 El Líder en grupos de desarrollos mundiales.

En el mundo empresas e instituciones emplean diversas formas para la selección y preparación de los líderes de proyectos informáticos los cuales deben tener todas las competencias antes mencionadas.

Por ejemplo en el proyecto Debian seleccionan un Líder, de forma anual el cual es seleccionado entre todos los desarrolladores que integran el grupo, este líder seleccionado cumple ciertas tareas dentro del proyecto:

- El líder del proyecto puede definir un área específica de responsabilidad y delegarla en un desarrollador de Debian.
- Dotar de autoridad a otros desarrolladores.

- El líder del proyecto puede hacer declaraciones de apoyo de puntos de vista o de otros miembros del proyecto.
- Tomar cualquier decisión que requiera acción urgente.
- Tomar cualquier decisión sobre la cual nadie más tenga responsabilidad.
- Junto con Software in the Public Interest (SPI), tomar decisiones afectando a propiedades que estén en custodia de proyectos relacionados con Debian.
- El líder del proyecto puede tomar decisiones acerca de cómo se utiliza el dinero que tiene Debian.

Esta manera empleada por el proyecto Debian, es muy democrática ya que es muy fácil poder seleccionar a los mejores desarrolladores para cumplir dicha función. Pero esta tiene una falla ya que un líder debe cumplir ciertas competencias, que algunas son innatas, pero otras se entrenan con el tiempo y mucha experiencia dirigiendo proyectos. En las muchas otras empresas a nivel mundial que reconocen la necesidad de especialización y colocan Gerentes de Proyecto profesionales al frente de sus proyectos. Este proceso de "profesionalización" en el manejo de los proyectos no es ni rápido ni fácil. No basta con la capacitación sino que requiere también de un cambio cultural en las organizaciones.

En la actualidad como se exponía anteriormente la mayoría de las instituciones u organizaciones coinciden en las competencias, objetivos y tareas que debe desarrollar un Líder dentro de un proyecto informático, la diferencia radica en la forma de su selección a la hora de cumplir dicha tarea.

Existen muchos factores que provocan el bajo desempeño de un líder:

- Poca experiencia en el desarrollo de proyectos de software.
- Poca preparación y capacitación sobre técnicas de dirección.

1.3.3 Gestión de Proyectos

La gestión de proyectos implica directamente, para los equipos que desarrollan productos informáticos, la planificación, supervisión y control del personal, del proceso y de los eventos que ocurren mientras evoluciona el software. Esto lo expresa Pressman en su libro “Ingeniería de Software, un enfoque práctico”, lo cual es un elemento de partida muy claro del área que atiende la gestión de proyecto. Tiene una gran importancia la gestión, ya que la elaboración software se compara con el trabajo realizado en una empresa muy compleja y más si implica gran cantidad de gente durante mucho tiempo.

Ideas importantes a tener en cuenta para gestionar un proyecto de software:

- *En cuanto a la duración del proyecto*, que no sea mayor a 6 o 7 meses. Los clientes no pueden esperar más de 2 o 3 meses para ver resultados, si un proyecto es muy grande hay que dividirlo por módulos y establecer entregables cada 2 o 3 meses.
- *Los requerimientos del cliente deben quedar claros*. Él puede pedir un Sistema de Contabilidad, y lo desarrolladores hacer un sistema de Contabilidad, pero lo que quería era un Sistema Integrado, es decir ventas, compras, inventario, etc.
- *Realizar un documento de requerimientos*, y hacer que los usuarios lo validen y aprueben. Esto principalmente servirá para dos cosas: la primera será saber que hacer, que tareas tienes que realizar, y en general para planificar la solución. La segunda cosa importante para la cual sirve, es que si estas en la etapa de pruebas y el usuario informa que se ha dado cuenta que un requerimiento no es lo que quiso decir, y hay que hacer cambios, que pueden tomar unas semanas más, es ahí donde se saca el documento de requerimientos, para mostrar que esa petición no estaba, y de esa manera si el usuario acepta, el proyecto puede durar unas semanas más.
- *Los roles*. La distribución correcta de los roles dentro del equipo. No es necesario tener un usuario por cada rol.
- *Asignar tiempo exclusivo a los proyectos*. Esto es importante sobre todo para los desarrolladores que aún son estudiantes y ya están dentro de un proyecto este caso es muy importante tenerlo en cuenta en la UCI. Si el horario de los estudios es el ideal, o sea, se estudia o bien en la mañana o

bien en la tarde, o bien en la noche. Esto permite darle un horario aceptable al proyecto. Pero qué pasa si se tiene clases en la mañana y en la tarde, y muy variado. Para estos casos solo le dedicamos el tiempo que podemos al proyecto, y muchas veces esto hace que sumemos a las estadísticas de porqué fallan los proyectos.

- En cuanto a la ***metodología a usar***, la revisión de todas las metodologías que existen pueden dar una visión de cual ajustar al el proyecto que se va a ejecutar, como las metodologías ágiles por ejemplo SCRUM. La cual es una forma de gestionar proyectos de software, y una metodología para la gestión del trabajo. También existe XP como metodología ágil. Otro tipo de proceso para el desarrollo de software es el propuesto por RUP.

  Este el proceso de desarrollo que se utilizará para la conformación del equipo de Delfos y la realización del producto Predictor, a lo largo del trabajo se va exponiendo por que es utilizado en las diferentes fases del desarrollo del producto el Proceso Unificado de Desarrollo (RUP). Pero es importante aclarar que la adaptación de este al proyecto tiene como objetivo fortalecer todo el proceso y no entorpecerlo, por lo cual no hacer por ejemplo: todos los diagramas UML que propone, si no hacer los realmente necesarios y que ayudan a entender mejor el modelo.

- ***Las herramientas no son el fin, son los medios***. En un proyecto no debe ser lo más importante la idea de usar la herramienta o entorno X, ya que las herramientas con el tiempo cambian, lo importante es el desarrollo del modelo del negocio, ya que este cambiará muy poco con el tiempo, y entre las diversas organizaciones. Esto no quiere decir que no sea importante escoger una buena herramienta de desarrollo.

La gestión esta enfocada hacia el personal, el producto, el proceso y el proyecto. Un personal altamente calificado es uno de los factores fundamentales de la elaboración de un software de calidad por lo tanto la gestión de proyecto tiene como uno de sus pilares la gestión para la superación y el control del personal. Ejemplo de esto es el modelo de madurez de capacidad del personal que propone SEI (Instituto de Ingeniería del Software de la Universidad Carnegie Mellon), el llamado P-CMM, por sus siglas en ingles. El modelo de madurez de gestión de personal define las siguientes áreas clave prácticas para el personal que desarrolla software: reclutamiento, selección, gestión de rendimiento,

entrenamiento, retribución, desarrollo de la carrera, desafió de la organización y del trabajo y desarrollo cultural y de espíritu de equipo.

Junto con esto SEI propone muchos modelos de madurez que impulsan a una gestión de proyectos, para la mejora y medición de la madurez específicos para varias áreas. Todos propuestos en la década de los 90:

- CMM-SW: CMM for software
- P-CMM: People CMM.
- SA-CMM: Software Acquisition CMM.
- SSE-CMM: Security Systems Engineering CMM.
- T-CMM: Trusted CMM
- SE-CMM: Systems Engineering CMM.
- IPD-CMM: Integrated Product Development CMM.

Luego CMMI se desarrolló para facilitar y simplificar la adopción de varios modelos de forma simultánea, y su contenido integra y da relevo a la evolución de sus predecesores:

- CMM-SW (CMM for Software).
- SE-CMM (Systems Engineering Capability Maturity Model).
- IPD-CMM (Integrated Product Development).

CMMI propone 5 niveles de madurez para clasificar a las organizaciones, en función de qué áreas de procesos consiguen sus objetivos y se gestionan con principios de ingeniería. Es lo que se denomina un modelo escalonado, o centrado en la madurez de la organización. CMMI plantea que en el nivel 2 de madurez es donde se comienza verdaderamente a gestionar el proyecto ya que en este nivel el incluye diferentes áreas de proceso como:

- Gestión y acuerdo con proveedores
- Planificación de proyecto
- Monitorización y control de proyecto

Además plantea otras áreas de proceso que entrar dentro de la gestión de proyectos pero que pertenecen a otros niveles de madures por ejemplo:

Nivel 3

- Gestión de riesgos
- Formación
- Procesos orientados a la organización
- Definición de procesos
- Gestión de equipos
- Gestión integral de proveedores
- Gestión integral de proyecto

Nivel 4

- Rendimiento de los procesos de la organización
- Gestión cuantitativa de proyectos

Nivel 5

- Innovación y desarrollo

Pero esta distribución propuesta por CMMI es muy ambiciosa par un proyecto pequeño debido a esto se realizan muchos de los procesos planteados en el nivel 2 y 3 pero sin atarse a dicha estructura, la mayoría de los procesos propuestos por CMMI también lo propone RUP lo que con una organización diferente, proponiendo mas o menos el desarrollo de ellos lo que en diferentes fases a lo largo de un mismo proyecto y esta prepuesta es la que aplicada en el desarrollo de este proyecto.

1.3.4 ¿Cómo Formar un Equipo?

Según el libro de TSP la buena conformación de un equipo de trabajo para la construcción de un software asegura la obtención de un producto de alta calidad. Existen algunos pasos importantes en la formación de los equipos:

P 1. “Se define el objetivo del trabajo que enfrentara el equipo, y cada uno de los miembros del equipo debe estar de acuerdo con este objetivo global.”(HUMPHREY 1999)

P 2. “Se definen las responsabilidades dentro del equipo. TSP propone los siguientes roles: líder del proyecto, gerente de desarrollo, gerente de planificación, gerente de calidad y gerente de soporte.”(HUMPHREY 1999)

P 3. “Se realiza un análisis de la estrategia para conseguir el objetivo propuesto. Se propone dividir el producto en módulos o subsistemas y se prevé la integración de todas las partes.”(HUMPHREY 1999)

P 4. “Se establecen protocolos y mecanismos claros de comunicación interna que garanticen la coordinación entre los diferentes miembros y módulos separados.”(HUMPHREY 1999)

P 5. “Se establecen protocolos de comunicación externa que definen los mecanismos de comunicación con los clientes, los instructores y otras entidades externas al equipo de desarrollo.”(HUMPHREY 1999)

Además de estos pasos importantes antes mencionados son necesarios otros para fortalecer la conformación del equipo.

P 6. Para la conformación, es necesario hacer una comprobación sobre el personal disponible para integrar el equipo, para detectar su nivel de preparación y el rol a desempeñar según sus aptitudes.

P 7. Establecer sistemas de superación y preparación del personal.

Como bien plantea el paso 2 enunciado anteriormente, el TSP propone varios roles para el desempeño dentro de un equipo de desarrollo de software, los cuales son explicados en dicho libro.

Además RUP propone los roles que deben desarrollar en cada flujo de trabajo durante todo el ciclo de vida del proyecto, roles que tienen un papel importante a la hora del trabajo.

Atendiendo a lo que propone RUP y teniendo en cuenta los roles propuestos por TSP para el desarrollo en equipo, se ajusta al entorno de desarrollo y a las necesidades del proyecto, los roles asignados principalmente son:

1. Líder de proyecto.
2. Analista de sistema.
3. Arquitecto.
4. Diseñador.
5. Programador.
6. Planificador.
7. Ingeniero de prueba.

Estos roles enunciados anteriormente son en las principales actividades que desarrollaron el personal del equipo de trabajo, pero muchos se desempeñaron en otros roles de los propuestos por RUP, dada necesidad, en el desarrollo del producto.

1.3.5 Principios para la selección de los integrantes de un equipo.

La conformación de un equipo de desarrollo de software es un paso primordial a la hora de enfrentar un proyecto determinado, para ello es necesario determinar cuales son realmente los personas que se deben asignar a los roles que se definieron que deben intervenir en el desarrollo del producto. Para realizar esto es muy importante tener en cuenta que para que una persona sea competente para un cargo debe cumplir no solamente con competencias de tipo cognitivo, sino también emocional, aquí enunciamos algunas ideas que se deben tener en cuenta a la hora de asignar personas a un puesto de trabajo dentro de un equipo.

1. Las personas no son solamente competentes desde el punto de vista cognitivo (coeficiente de inteligencia), sino también desde el punto de vista emocional y como un todo para determinar si es competente o no en determinadas actividades o desempeños laborales.
2. Las competencias son características subyacentes a las personas, que como tendencias están relacionada al desempeño exitoso en un puesto de trabajo.
3. Existen diferentes tipos de competencias: básicas o primarias, asentadas en aptitudes (razonamiento, expresiones verbales, etc.), rasgos de personalidad (ascendencia, auto confianza, etc.) y actitudes (predisposiciones al riesgo, al buen sentido del humor, etc.). Y hay

competencias secundarias o complejas, basadas en la unión de varias competencias simples (capacidad de negociación, liderazgo, planificaron, etc.).

4. Los perfiles de competencias definidos para un rol son un conjunto de competencias secundarias que van con sus descripciones más o menos detalladas de las pautas que debe seguir ese rol basándose en la experiencia.
5. Es necesario tener en cuenta para la selección para los roles de liderazgo dentro del equipo, que no siempre la persona más competente dentro de un puesto de trabajo o actividad es la que mejor puede dirigir al grupo que se desempeña en la misma actividad. Al igual que no siempre es factible que la persona que ocupa un segundo puesto es el más indicado para asumir el liderazgo en un momento determinado.
6. La mejor tendencia a la hora de la selección del personal, no es enfocarlo hacia un puesto X, sino hacia varios puesto. Significando flexibilidad y multihabilidades en la persona, y no solo orientado al conocimiento sino a las habilidades personales.

Teniendo en cuenta los roles especificado por RUP para el desarrollo de un software y las responsabilidades de cada uno de ellos en la etapa de desarrollo, más lo expresado en el TSP sobre las funcionalidades y responsabilidades que tienen estos dentro del equipo de desarrollo, además de aplicar las ideas antes expresadas para poder seleccionar un personal que independientemente de sus conocimientos pudiera compenetrarse y trabajar como un equipo para obtener el resultado esperado por el cliente se seleccionaron un grupo de estudiantes a los cuales se les asignaron roles dentro del equipo según algunas pruebas y la observación de donde se desempeñaban mejor cada una de ellos que fue posible producto a que todos prevenían del mismo grupo docente. Pero es muy importante resaltar que en la conformación de un equipo son muy importantes las pruebas de aptitud dirigidas a conocer cual o cuales pueden los roles que puede desempeñar un compañero atendiendo a sus competencias tanto cognitivas, como emocionales.

1.3.6 Estimulación dentro de un Equipo de desarrollo.

Es muy importante tener en cuenta por parte de cualquier directivo o grupo de dirección de un grupo de desarrollo la palabra estimulación, como logra que un miembro del equipo se sienta estimulado por el trabajo desarrollado y a su vez no pierda la noción de que ese es su deber dentro del equipo. Pero antes de hablar de estimulación hay que hablar de motivación que no es más que todas las acciones realizadas por la dirección del proyecto encaminadas a ilusionar a los miembros del equipo con vista a conseguir de ellos un fuerte compromiso con el trabajo que van a desarrollar.

Cuando se trabaja en equipos siempre se debe tener dos esquemas de estimulación uno dirigido al equipo y otro al individuo.

No se puede premiar exclusivamente el éxito individual ya que el equipo exige muchas veces renunciar al lucimiento personal en favor del éxito del equipo. Si únicamente se reconociera al individuo se dañaría el espíritu de equipo: sus miembros, en lugar de sacrificarse por el equipo, tratarían de destacar individualmente.

Pero también se debe establecer una gratificación a título individual ya que dentro del equipo hay distintos niveles de dedicación y de eficiencia. Hay que premiar al empleado que destaque individualmente ya que esto contribuye a crear cierto espíritu competitivo dentro del equipo, lo que redunda en un mejor desempeño.

Es muy importante establecer metas intermedias a la hora de realizar un trabajo ya que el éxito temprano y la satisfacción de ir cumpliendo con lo propuesto estimulan al equipo.

1.3.7 Fundamentación de la utilización del Plan de Iteraciones

1.3.7.1 ¿Por qué un Plan de Iteraciones?

Un proceso iterativo e incremental significa llevar a cabo un desarrollo en pequeños pasos. Para ello:

1. Se escoge una pequeña parte del sistema y se sigue con el todo el ciclo de vida clásico en cascada (planificación, especificación, diseño,…).
2. Si estamos satisfechos con el paso anterior damos otro. Cada uno proporciona retroalimentación.
3. Las iteraciones son distintas. Al principio del proyecto proporcionan una comprensión de los requisitos, del problema, de los riesgos y el dominio de la solución; las últimas nos proporcionan la visión externa (producto para el cliente).

RUP es un proceso de desarrollo, donde su ciclo de vida se caracteriza por ser:

1. Dirigido por casos de uso.
2. Centrado en la arquitectura
3. Iterativo e Incremental: Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

Motivos para adoptar un ciclo de vida iterativo e incremental:

1. Para identificar **riesgos**. Esto ocurre en las dos primeras fases: Inicio y Elaboración, en vez de en la etapa de integración como con el modelo en cascada.
2. La **arquitectura** se establece en la fase de elaboración y eso permite cambiarla si se considera necesario en una etapa temprana del desarrollo, por tanto con pocos costes. En el ciclo de vida en cascada esto se descubre más tarde.
3. Gestión de **requisitos cambiantes**: Gracias a que se hace una integración continua los usuarios disponen desde las primeras iteraciones de versiones ejecutables que permiten un cambio de impresiones en este sentido.

4. **Fallos**: Al igual que en los puntos anteriores, la ventaja de este ciclo de vida es que los fallos se van descubriendo a medida que se implementan nuevas funcionalidades; esto significa que no hay una avalancha de problemas al final.
5. **Aprendizaje**: Con un par de iteraciones es suficiente para que todo el mundo comprenda los diferentes flujos de trabajo.

Planificar siempre es muy importante por lo tanto la realización de un plan de Iteraciones proporciona al equipo de trabajo un claridad de cómo se debe hacer y como esta el producto actualmente.

En el proyecto se aplico un plan basado en la propuesta de iteraciones por fases de RUP.

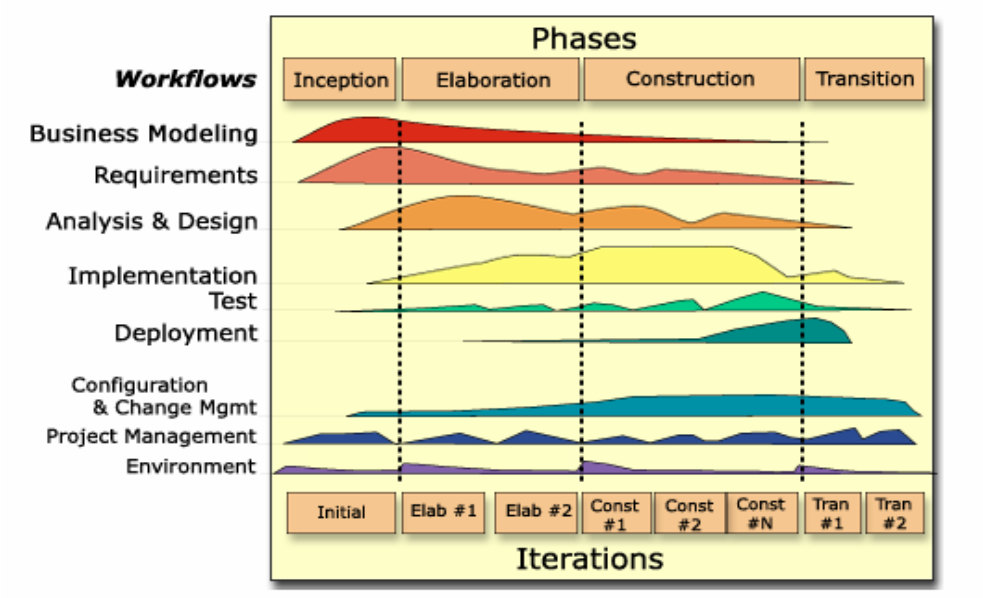

Figura 1: RUP en Dos Dimensiones

1.3.7.2 Plan por Fases

La propuesta del plan por fases realizadas se hace al comienzo del desarrollo del producto, por lo cual la cantidad de iteraciones y el tiempo. La siguiente tabla muestra la forma de realizar la planificación.

Fase	Nro. Iteraciones	Duración
Fase de Inicio	-	-
Fase de Elaboración	-	-
Fase de Construcción	-	-
Fase de Transición	-	-

Tabla 1: Plan por Fases

Los hitos que marcan el final de cada fase se describen en la siguiente tabla.

Descripción	Hito
Fase de Inicio	Se desarrollarán los requisitos del producto desde la perspectiva del usuario, los cuales serán establecidos en el artefacto Visión. Los principales casos de uso serán identificados y se hará un refinamiento del Plan de Desarrollo del Proyecto. La aceptación del cliente/usuario del artefacto Visión y el Plan de Desarrollo marcan el final de esta fase.
Fase de Elaboración	En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y/o críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en el primer release de la fase de Construcción deben estar analizados y diseñados (en el Modelo de Análisis/Diseño). La revisión y aceptación del prototipo de la arquitectura del sistema marca el final de esta fase. La primera iteración tendrá como objetivo la identificación y especificación de los principales casos de uso, así como su realización preliminar en el Modelo de Análisis/Diseño, también permitirá hacer una revisión general del estado de los artefactos hasta este punto y ajustar si es necesaria la planificación para asegurar el cumplimiento de los objetivos. Ambas iteraciones tendrán una duración de cuatro semanas.
Fase de Construcción	Durante la fase de construcción se terminan de analizar y diseñar todos los casos de uso, refinando el Modelo de Análisis/Diseño. El producto se construye en base a 3 iteraciones, cada una produciendo un release a la cual se le aplican las pruebas y se valida con el cliente/usuario. Se

	comienza la elaboración de material de apoyo al usuario. El hito que marca el fin de esta fase es la versión del release 3.0, con toda la capacidad operacional del producto, listo para ser entregada a los usuarios para pruebas beta.
Fase de Transición	En esta fase se prepararán los release para distribución, asegurando una implantación y cambio del sistema previo de manera adecuada, incluyendo el entrenamiento de los usuarios. El hito que marca el fin de esta fase incluye, la entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, la finalización del entrenamiento de los usuarios y el empaquetamiento del producto.

Tabla 2: Descripción de las Fases

1.3.8 Fundamentación de la utilización del Plan de Gestión de Riesgos

El análisis y la gestión del riesgo son una serie de pasos que ayudan al equipo del software a comprender y a gestionar la incertidumbre. Un proyecto de software puede estar lleno de problemas. Un riesgo es un problema potencial –puede ocurrir o no-. Pero sin tener en cuenta el resultado, realmente es una buena idea identificarlo, evaluar su probabilidad de aparición, estimar su impacto, y establecer un plan de contingencia por si ocurre el problema.(PRESSMAN 2005)

Realizar el análisis y la gestión de riesgos es muy importante a la hora de elaborar un software, debido a que la elaboración del mismo es un proceso muy complejo y siempre existe la posibilidad de que salgan cosas mal, de hecho la probabilidad de la ocurrencia de problemas es realmente alta. Por esta razón, una acción eficiente seria el estar preparados, comprender los riesgos, preveer a los mismos con medidas adecuadas según el espacio en que se desarrolle permitiendo así gestionarlos o eliminación. Todos estos factores decisivos en la gestión general del Proyecto Productivo.

Los principales pasos que se deben seguir para la gestión de Riesgos son:

1. El reconocimiento de que algo puede ir mal es el primer paso, llamado identificación del riesgo.
2. Cada riesgo es analizado para determinar la probabilidad de que pueda ocurrir y el daño que puede causar si ocurre.

3. Se priorizan los riesgos, en función de la probabilidad y del impacto.
4. Por último, se desarrolla un plan para gestionar aquellos riesgos con gran probabilidad e impacto.

Existen dos tipos diferenciados de riesgos para cada categoría presentada en el apartado anterior: genéricos y específicos del producto. *Los riesgos genéricos* son una amenaza potencial para todos los proyectos de software. *Los riegos específicos* del producto, sólo los pueden identificar los que tienen una clara visión de la tecnología, el personal y el entorno específico del proyecto en cuestión.

Un método para poder identificar los riegos es la elaboración de una lista de comprobación de elementos de riesgos la cual se concentra en subconjuntos conocidos y predecibles:

- Tamaño del producto: asociados con el tamaño del software a construir o a modificar.
- Impacto en el negocio: riesgos asociados con las limitaciones impuestas por la gestión o por el mercado.
- Características del cliente: asociados con la sofisticación del cliente y la habilidad del desarrollador para comunicarse con el cliente en los momentos oportunos.
- Definición del proceso: asociados con el grado de definición del proceso del software y su seguimiento por la organización de desarrollo.
- Entorno de desarrollo: asociados con la disponibilidad y calidad de las herramientas que se van a emplear en la construcción del producto.
- Tecnología a construir: asociados con la complejidad del sistema a construir y la tecnología punta que contiene el sistema.
- Tamaño y experiencia de la plantilla: asociados con la experiencia técnica y de proyectos de los ingenieros del software que van a realizar el trabajo.

Se elaboró un plan de gestión de riesgos divido en 4 fases:

1.3.8.1 Fase de Identificación de Riesgos

Para la identificación de los riegos se elaboró una lista maestra de riesgos que pude realizarse de diferentes maneras o conjunta entre ellas:

- A través de encuestas.
- A través de entrevistas.
- A través de la literatura.

En el proyecto, fueron aplicadas para la conformación de la lista maestra de riegos las opciones de las entrevistas con el personal de desarrollo y otro personal ajeno al equipo pero con experiencia en el desarrollo de software, para obtener riesgos directos, además de incluir algunos riesgos encontrados en los materiales consultados que son aplicables a la mayoría de los proyectos que desarrollan productos informáticos. No se selecciono la opción de encuestas porque los recursos humanos del proyecto son escasos y realizar encuestas y procesarlas ralentizaría el proceso de identificación de los riesgos.

1.3.8.2 Fase de Análisis

El análisis de los riegos es muy importante, en esta fase es donde se puede determinar a través del impacto y la probabilidad de ocurrencia da cada uno. ¿Cuales son los que presentan mayor nivel de amenaza al proyecto en desarrollo y por lo tanto sobre los cuales hay que centrar mayor atención?

Durante esta etapa todos los integrantes del equipo de desarrollo examinan todos los riegos obtenidos de la fase anterior y les asignan un valor de impacto y probabilidad. Usando esta lista, el líder y el planificador pueden saber cuales son los riesgos a los cuales hay que designar recursos para su eliminación o mitigación dada su importancia o cuales por presentar una amenaza baja pueden quitarse de la lista.

Se decidió tomar como técnicas para asignar las prioridades planteadas por PMI (Project Management Institution), organización mundialmente reconocida. La cual consiste en tomar las decisiones reunidas por el equipo en dos de los elementos de riesgos más universales: probabilidad e impacto y multiplicando estas obtenemos un componente que denominamos exposición al riesgo.

- **Probabilidad**: para que exista un riesgo este debe tener una probabilidad mayor que cero, sino no seria un riesgo para el proyecto. La tabla siguiente muestra una escala para la utilización a la hora de la definición de la probabilidad de ocurrencia.

Definición de escala de probabilidades de ocurrencia		
Escala	**Definición**	**Descripción**
0.1	Muy improbable	Me sorprendería si ocurre.
0.3	Poco probable	Mas probable que no ocurra a que si.
0.5	Probable	Tan probable que ocurra como que no ocurra.
0.7	Altamente probable	Más probable que ocurra a que no.
0.9	Casi cierto	Me sorprendería si no ocurre.

Tabla 3: Probabilidad de Ocurrencia

- **Impacto**: es el efecto que puede ocasionar un riesgo sobre el costo (aumento), el cronograma de entrega de las etapas (atraso), en la funcionalidad (disminución de nivel de desempeño) y en la calidad (reducción). Para medir se uso una escala de valores no lineales.

Definición de escala de impacto negativo en los objetivos del proyecto					
Objetivos del proyecto	**Escala**				
	Muy poco(0.0.5)	**Poco (0.10)**	**Moderado(0.20)**	**Alto (0.40)**	**Muy alto (0.80)**
Costo	Incremento insignificante	Incremento en costo < 5%	Incremento de costo de 5-10%	Incremento de costo de 10-20%	Incremento del costo >20%
Tiempo	insignificante	Retraso <5%	Retraso global de 5-10%	Retraso global 10-20%	Incremento del tiempo > 20%
Alcance	Reducción escasament e apreciable	Áreas menores de alcances afectadas	Áreas mayores de alcances afectadas	Reducción de alcance inaceptabl e	Proyecto no viable
Calidad	Degradación escasament	Solo aplicacione	Reducción de calidad que	Reducción de calidad	Proyecto no viable

	e apreciable	s muy exigente se afectan	requiere aprobación	inaceptable	

Tabla 4: Escala de Impacto

- **Exposición al riesgo**: se obtiene combinando la probabilidad y el impacto. Es muy práctico crear una matriz que tenga en cuenta las posibles combinaciones de las puntuaciones y las clasifique en las categorías de riesgo alto, medio y bajo.
  - Bajo: los valores de impacto y probabilidad son solo de adivinanzas.
  - Medio: los valores de impacto y probabilidad se consideran precisos, a un nivel aceptable.
  - Alto: los valores de impacto y probabilidad se consideran bastante exactos.

En la matriz el nivel los colores determina el nivel de riesgo, siendo el más alto, la parte más oscura.

Probabilidad	**Amenazas**				
0.9	0.05	0.09	0.18	0.36	0.72
0.7	0.04	0.07	0.14	0.28	0.56
0.5	0.03	0.05	0.10	0.20	0.40
0.3	0.02	0.03	0.06	0.12	0.24
0.1	0.01	0.01	0.02	0.04	0.08
	0.05	**0.10**	**0.20**	**0.40**	**0.80**

Tabla 5: Exposición al riesgo

1.3.8.3 Fase de Planificación de respuestas de Riesgos

Durante esta fase se deben tener en cuenta cuatro alternativas: evitar, mitigar, transferir y aceptar. Consiste en desarrollar un plan para controlar los riesgos más importantes identificados en la fase de análisis.

Una vez planteada las acciones de respuesta a los riegos, se deben planificar los recursos y las actividades de mitigación para los riegos. A estas acciones se le deben asignar responsables que se encargaran del cumplimiento y reporte del nivel de su realización.

1.3.8.4 Seguimiento y control de Riesgo

Esta fase es muy importante, es donde se controlan los riesgos que aparecen en la lista actual de riegos, pero además se debe estar muy atento porque pueden aparecer nuevos riesgos en su entorno a medida que avanza el proyecto. En la tabla que continuación aparece un formulario que es utilizado para el seguimiento y control de los riegos.

ID	Prioridad	Respuesta al riesgo	Estado	Fecha del estado	Aprobado	Comentarios
#	alta, media y baja	Acciones planteadas	Activo o inactivo	fecha	Nombre de la persona que esta haciendo el control de riesgo	

Tabla 6: Seguimiento y control

1.3.9 Fundamentación de la utilización del Plan de métricas del proyecto

Ningún sistema de métricas es igual a otro en el mundo porque depende de las características de las empresas para la cual se cree, estos son hechos a la medida bien ajustados a las entidades y sus entornos de trabajo y desarrollo. Puede que haya dos proyectos que no necesiten medir lo mismo pero lo que le sirve a uno no le sirve a otro.

El Instituto de Ingeniería de Software (IIS), ha desarrollada una guía para establecer un programa de mediciones de software dirigido hacia los objetivos de cada organización en especifico.

1. Identificar los objetivos del negocio.
2. Identificar lo que se desea saber o aprender.
3. Identificar 10s sub-objetivos.
4. Identificar las entidades y atributos relativos a esos sub-objetivos.
5. Formalizar 10s objetivos de la medición.

6. Identificar preguntas que puedan cuantificarse y los indicadores relacionados que se van a usar para ayudar a conseguir 10s objetivos de medición.
7. Identificar 10s elementos de datos que se van a recoger para construir 10s indicadores que ayuden a responder a las preguntas planteadas.
8. Definir las medidas a usar y hacer que estas definiciones sean operativas.
9. Identificar las acciones que serán tomadas para mejorar las medidas indicadas.
10. Preparar un plan para implementar estas medidas.(PRESSMAN 2005)

Documentos normativos sobre métricas: IEEE e ISO:

- IEEE Std 982.1-1988 IEEE Standard Dictionary of Measures to Produce Reliable Software.

  Divide las mediciones en:

  - ✓ Métricas del producto. Causas y efectos de aspectos estáticos y dinámicos tanto de los elementos de proyectos anteriores como los de uso.
  - ✓ Métricas del proceso.
    - Control de la gestión: la evaluación de la orientación del desarrollo y mantenimiento del proceso.
    - Cobertura: la evaluación de la presencia de todas las actividades necesarias para desarrollar o mantener el producto de software.
    - Riesgo, beneficio y evaluación de costos: la evaluación del costo, plan y rendimiento según el proceso.

- En la ISO/IEC 14598-1

  Aparecen requisitos y guías para la selección de los criterios de medición y las métricas para evaluar el producto de software.

  - ✓ Método “Métricas en ocho pasos”.
  - ✓ OPM (Objetivo, Pregunta, Métrica).

Existen muchos métodos para desarrollar métricas, pero el más ampliamente aplicado y mejor conocido es el desarrollado por Víctor Basili el GQM (Goal Question Metric). GQM hizo posible la medición del software. Este enfoque abraza la filosofía de medición de Basili, la cual establece que las métricas se derivan de las metas, limitando la recolección de datos a aquello que es necesario para responder a las preguntas importantes, indicando suposiciones explícitamente, y usando un modelo claro para interpretar los resultados de las mediciones.

El enfoque GQM está basado en el postulado: para que la medición de los productos y procesos de una organización sea útil, se deben primero especificar las metas que la organización tiene, tanto para sí misma, como para sus proyectos. Una vez que las metas son establecidas, la organización debe poder controlarlas usando una jerarquía de preguntas relacionadas a los datos previstos para evaluar esas metas operacionalmente.

Las métricas obtenidas y las medidas deben cumplir con las siguientes características fundamentales:

- *Simple y fácil de calcular*: debería ser relativamente fácil de aprender a obtener la métrica y su cálculo no obligara a un esfuerzo o a una cantidad de tiempo inusuales.
- *Empírica e intuitivamente persuasiva*: la métrica debería satisfacer las nociones intuitivas del ingeniero de software sobre el atributo del producto en cuestión (por ejemplo: una métrica que mide la cohesión de un módulo debería aumentar su valor a medida que crece el nivel de cohesión).
- *Consistente en el empleo de unidades y tamaños*: el cálculo matemático de la métrica debería utilizar medidas que no lleven a extrañas combinaciones de unidades. Por ejemplo, multiplicando el número de personas de un equipo por las variables del lenguaje de programación en el programa resulta una sospechosa mezcla de unidades que no son intuitivamente concluyentes.
- *Independiente del lenguaje de programación*: las métricas deberían apoyarse en el modelo de análisis, modelo de diseño o en la propia estructura del programa. No deberían depender de los caprichos de la sintaxis o semántica del lenguaje de programación.
- *Un mecanismo eficaz para la realimentación de calidad*: la métrica debería suministrar al desarrollador de software información que le lleve a un producto final de superior calidad. No obstante que la mayoría de las métricas de software compensan las características anteriores, algunas de las métricas usualmente empleadas no cumplen una o dos características. Un ejemplo es

el punto funcional (PF), en donde se consigue argumentar que el atributo es *consistente y objetivo* pero falla por que un equipo ajeno independiente puede no ser capaz de conseguir el mismo valor de PF que otro equipo que emplee la misma información del software. En tal caso la siguiente pregunta se manifiesta, pero no por esto se debe desechar la utilización la PF, ya que aporta una visión interna útil y por lo tanto provee de un valor claro, incluso si no satisface un atributo perfectamente. Este es un ejemplo en donde el uso de las métricas depende de los factores individuales del desarrollador o desarrolladores del software.

Teniendo en cuenta lo analizado anteriormente se elaborará de plan simple que contribuya a la evaluación de todo el proceso de desarrollo de Predictor.

1.3.10 Fundamentación de la utilización del Plan de calidad

Ha sido y aún es práctica usual considerar la calidad como algo ajeno a la Dirección y sólo vinculada a la parte puramente técnica; pero se ha visto como es uno de los objetivos básicos De la elaboración y puesta a termino de los proyectos y que en la tríada de costo, tiempo y calidad para un alcance, el análisis debe ser integral y el logro de ellos garantizará ya, de hecho, una cierta calidad que pudiera considerarse como una calidad básica. Por otra parte, el Proyecto una vez terminado, debe garantizar que la explotación sea la debida y que satisfaga finalmente al cliente o usuario. Si se incluyen todos estos aspectos ya el concepto de calidad necesariamente se amplía. La calidad hay que conseguirla a través de todo el proceso del Proyecto con la participación de los recursos humanos que intervienen, tengan conocimiento de lo que es y significa la ***calidad total*** y que apliquen una filosofía común y completa sobre la misma.

¿Qué es calidad total?

- “Cumplir con los objetivos de mayor alcance: los exigidos por los usuarios/consumidores.”(HEREDIA)

- Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente.

Sobre la calidad total o gestión de la calidad total (GCT) es una tendencia iniciada en Japón, que emigró para el mundo occidental alrededor de los setenta y ochenta, se explica que la GCT consta fundamentalmente de 4 pasos.

Si un equipo de software aplica la calidad a todas las actividades de la ingeniería de software, podrá reducir la cantidad de trabajo repetido, lo que supondrá una disminución del tiempo de desarrollo y por ende del costo.

Para poder establecer una buena gestión de la calidad es necesario que:

- Los niveles de dirección, desde la alta dirección hasta el último eslabón, estén convencidos de que la calidad es algo fundamental para la supervivencia del proyecto.
- Voluntad real de poner en práctica lo que se planifica para el logro de la calidad.
- Conocimiento de los requisitos del usuario.
- Conocimiento de la relación costo-calidad para conseguir una calidad que sea compatible con el costo.
- Definición de forma coherente los objetivos (costo, plazo, calidad) para un alcance.
- Los elementos involucrados se sientan motivados económica y profesionalmente.
- Un enfoque de sistema.
- Se consideren y equilibren los diferentes intereses de los elementos presentes en el proyecto.

El papel que juega el grupo de calidad es muy importante:

- Revisa la descripción del proceso para ajustarse a la política de la empresa, los estándares internos del software, los estándares impuestos externamente (por ejemplo: IS0 9001), y a otras partes del plan de proyecto del software.
- Identifica, documenta y sigue la pista de las desviaciones desde el proceso y verifica que se han hecho las correcciones.

- Revisa los productos seleccionados; identifica, documenta y sigue la pista de las desviaciones; verifica que se han hecho las correcciones, e informa periódicamente de los resultados de su trabajo al gestor del proyecto.
- Asegurar que las desviaciones del trabajo y los productos del software se documentan y se manejan de acuerdo con un procedimiento establecido. Las desviaciones se pueden encontrar en el plan del proyecto, en la descripción del proceso, en los estándares aplicables o en los productos técnicos.
- Registra lo que no se ajuste a los requisitos e informar a sus superiores. Los elementos que no se ajustan a los requisitos están bajo seguimiento hasta que se resuelven.
- Además de estas actividades, el grupo de SQA coordina el control y la gestión de cambios y ayuda a recopilar y a analizar las métricas del software.

La importancia de crear un grupo de garantía de la calidad, es que este sería el cliente en casa, y es el encargado establecer y controlar la aplicación del Plan de Calidad, este plan se elabora durante la fase de planificación del proyecto y debe ser revisado por todas las partes interesadas. Un plan se identifica:

- evaluaciones a realizar
- auditorías y revisiones a realizar
- estándares que se pueden aplicar al proyecto
- procedimientos para información y seguimiento de errores
- documentos producidos por el grupo SQA
- realimentación de información proporcionada al equipo de proyecto del software.

El modelo CMM-CMMI: Aseguramiento de la calidad, pone como prioridades de un plan de aseguramiento de la calidad:

- Evaluar objetivamente la ejecución de los procesos, los elementos de trabajo y servicios contra las descripciones de procesos, estándares y procedimientos.
- Identificar y documentar los elementos no conformes.
- Proporcionar información a las personas que están usando los procesos y a los gestores, de los resultados de las actividades del aseguramiento de la calidad.

- Asegurar de que los elementos no conformes son arreglados.

A la hora de definir un Plan de Control de la Calidad este debe ser funcional para todas las fases del proyecto. Y además se puede definir uniendo las definiciones anteriores, que un plan tiene como objetivo:

- Evaluar a través de auditorías y revisiones la ejecución de los procesos, los elementos de trabajo y servicios contra las descripciones y procedimientos.
- Aplicar los estándares que sean aceptables para el proyecto.
- Asegurar el arreglo de los elementos que se detecten con errores o que no concuerden con las descripciones.
- Proveer de realimentación para los miembros del equipo y a las personas interesadas en el proceso de desarrollo.

Para la ejecución del Plan de Calidad se utilizó una plantilla elaborada por la dirección de calidad de la universidad, en la cual se recogen los elementos antes expuestos.

1.3.11 Fundamentación de la utilización del Plan de solución de problemas

Como bien se sabe un problema es cualquier situación indeseable en un proceso o en su resultado. También podría existir una situación indeseable si un proceso o resultado actual no cumple con los requisitos futuros del cliente.

Resolver problemas, mejora la satisfacción del cliente y reduce el precio del incumplimiento (PDI). Se trabaja sobre cinco pasos fundamentales que proporcionan un método sistemático para eliminar la causa o causas raíz de un problema.

1.3.11.1 Definir la situación

La descripción de un problema, es un enunciado claro del problema en términos de un incumplimiento específico.

¿Cuál es la situación indeseable? ¿Qué sucedió que no debió haber sucedido? ¿Cuáles requisitos no se están cumpliendo? ¿Cuándo es que no se cumplen? ¿Con qué frecuencia no se cumplen? ¿Cuál es el Precio del Incumplimiento (PDI)?

Describir Claramente el Problema.

- Concentrarse en los datos y no en la causa.
- Especificar el incumplimiento sin buscar culpables.

Planear la Solución

- Decidir quien es la gente necesaria.
- Determinar el criterio de resolución.
- Estimar la fecha de resolución.

1.3.11.2 Remediar temporalmente.

En este paso se necesita analizar detenidamente las consecuencias que se identificaron en el Paso 1 para determinar qué tan rápido se necesita un remedio temporal o para evaluar la efectividad del que quizá ya esté implantado. Un remedio temporal es un paso para mantener el proceso funcionando.

Un remedio temporal minimiza las consecuencias de un problema, no se dirige a la causa o causas. El problema con el remedio temporal es que es un gasto innecesario. Si no se resuelve el problema, se tendrá que seguir remediando y costará más y más. No es una solución permanente del

problema. El remedio temporal gana tiempo, hasta que la causa o causas se puedan identificar y eliminar.

1.3.11.3 Identificar la(s) causa(s) raíz.

Identificar la causa o causas raíz, se desarrolla un plan para recolectar datos. Después los datos se recolectan, se organizan y se analizan.

Planear y Recolectar Datos

Técnicas

- Diagrama de causa y efecto.

El diagrama de causa y efecto es una técnica poderosa usada para representar gráficamente las posibles causas. Puede utilizarse para sintetizar lo que se conoce sobre el problema, identificar qué información adicional se necesita y para hacer una tormenta de ideas sobre las posibles causas El diagrama tiene anotada la descripción del problema a la derecha como "efecto". Las ramas que salen de los brazos que van hacia el efecto son las posibles causas.

Uno por uno, todos los participantes tienen su turno para hablar y expresar las posibles causas del problema. Esto continua hasta que se hayan plasmado en el diagrama todas las posibles causas. Es útil nombrar a los brazos del diagrama y dibujar ramas poder categorizar las posibles causas de acuerdo a los brazos. Lo más importante a recordar es usar el diagrama para anotar todas las sugerencias posibles. Así pueden considerarse más posibles causas y nada se pasa por alto cuando intentamos identificar la causa o causas raíz del problema. Para usar esta herramienta el grupo sigue las reglas de la tormenta de ideas.

Diagrama de Causa y Efecto o de Ishikawa

- Materiales

- Medio ambiente
- Mano obra
- Mediciones
- Método
- Efecto

Cuando la lista está completa, el equipo puede comenzar a evaluar cada idea e intentar llegar a una decisión por consenso.

Tormenta de ideas

- Fijar un tiempo límite
- Registrar todas las ideas
- Alentar todas las sugerencias
- Construir sobre las ideas de los demás

1.3.11.4 Tomar acción correctiva

En este punto debe implantarse una solución permanente para el problema. Primero, si es necesario reunir a la gente que pueda decidir más efectivamente sobre ideas u opciones para la acción correctiva. Cuando se han planteado todas las opciones hay que elegir, planear, comunicar e implantar la mejor acción.

Reunir a la Gente Clave

- Aquellos con conocimiento y autoridad.
- Clientes y/o proveedores.
- Aquellos quienes son responsables de resolver el problema.

Generar Posibles Acciones Correctivas

Una vez identificada bien la causa o causas raíz, es el momento de tomar acción correctiva. Aunque muchas veces la medición muestra que hay más de una causa raíz.

Cuando esto sucede, se necesita seleccionar la cantidad de causas a trabajar. Puede no ser efectivo tratar con todas las causas simultáneamente y se puede evaluar mejor y dar seguimiento a los esfuerzos si se corrige una causa a la vez. Además, con frecuencia hay más de una acción que eliminará cualquier causa raíz. Por lo tanto, es importante observar todas las acciones posibles.

Cuando sea posible, las acciones correctivas deben ser "a prueba de errores", esto es modificar o aumentar los mecanismos en un proceso para que sea imposible producir o entregar a un cliente un producto defectuoso. Ejemplos de mecanismos "a prueba de errores" (Poke Yoke) son:

- Programas de cómputo que no permiten seguir adelante si se comete un error.
- Mecanismos que aseguran que un número específico de componentes esté presente o que ciertos pasos se tomen antes de que el proceso siga adelante.
- Un dispositivo que evita que pase un producto defectuoso al siguiente paso del proceso.

Elegir la Acción Correctiva

- Costo
- Complejidad
- Tiempo
- "A prueba de errores"

Planear, Comunicar e Implantar

Una vez que se ha elegido la acción correctiva, debe desarrollarse un plan para su implantación. Este plan puede incluir:

- Acciones que se van a tomar
- Asignar responsabilidades

- Fecha de resolución.
- Clientes afectados
- Implantación de acuerdo con el plan.

1.3.11.5 Evaluar

Se ha analizado la definición de un problema, luego la identificación y eliminación de la causa o causas raíz. Las acciones para asegurar que el problema se ha eliminado para siempre son muy importantes dentro de este proceso.

Con frecuencia hay una sensación de alivio y satisfacción después de que se ha tomado la acción correctiva. Sin embargo, un problema no está completamente resuelto hasta que se haya evaluado esa acción correctiva para ver si fue efectiva y se le haya dado seguimiento para asegurar que siga operando.

Evaluar

Cuando la situación se determinó en el Paso 1, se estableció un criterio de resolución. Este criterio se usó para evaluación. Evaluar la acción correctiva determina si el problema está resuelto o no. La primera acción para evaluar la acción correctiva es examinar el remedio temporal. Puede ser necesario terminar con el remedio temporal para que los datos recolectados para evaluación no se vean afectados. Una vez que se recolectaron y se analizaron los datos, es posible ver si el criterio de resolución se ha cumplido o no.

- Revisar el remedio temporal.
- Recolectar y analizar los datos
- Cumplir con el criterio de resolución
- Auditar.
- Encuestar a clientes y proveedores
- Revisión informal

En algunas situaciones cuando se identificó más de una causa raíz, se toma la decisión de no resolver todas las causas al mismo tiempo. Cuando así sucede, es necesario revisar el criterio de resolución para apoyar esa decisión. Por ejemplo, el criterio de resolución podría incluir una corrección que diga que el problema se considerará resuelto cuando no haya habido ningún incumplimiento debido a una causa específica por un período determinado.

1.4 Conclusión

Actualmente para dirigir un proyecto informático hay que tener en cuenta todas las tendencias, metodologías, herramienta y procesos que hacen que un equipo de trabajo pueda llegar a obtener de forma eficiente un producto.

Tras el análisis realizado en este capitulo sobre la gestión de proyecto y las elaboraciones de planes que contribuyan a la orientación, control y seguimiento de las tareas a cada uno de los miembros del equipo. Se definió que el proceso por el cual, el desarrollo de la aplicación a desarrollar se guiara es RUP, adaptándolo al entorno de Predictor.

Capítulo II: Líder de proyecto. Desempeño dentro de Predictor.

2.1 Introducción

Para lograr el desempeño profesional como líder de proyecto se realizó la elaboración de los planes que se incluyen dentro de la gestión del proyecto, esto planes proveen a todos los miembros del equipo y al líder, de una herramienta muy eficiente para poder lograr el desarrollo organizado a lo largo de todo el ciclo de vida del producto. La elaboración de los planes suele ser un proceso complejo, por lo cual manteniendo RUP como guía del proceso de desarrollo, se ha adaptado al proyecto todo lo necesario para que se agilice su construcción.

La evaluación del proceso de desarrollo también es posible gracias a estos planes, en criterio de: cuanto se cumplió o no, cuales fueron las desviaciones ocurridas y cual es el estado actual del proyecto, proporcionan a todos los miembros del equipo y al cliente de cuanto a ocurrido dentro de la elaboración del software.

2.2 Plan de iteraciones

2.2.1 Plan por Fases

La propuesta del plan por fases realizadas se hizo al comienzo del desarrollo producto por lo cual la cantidad de iteraciones y el tiempo varió atendiendo a cambios realizados. La siguiente tabla muestra la planificación realizada.

Fase	Nro. Iteraciones	Duración
Fase de Inicio	1	3 semanas

Fase de Elaboración	2	5 semanas
Fase de Construcción	3	4 semanas
Fase de Transición	1	2 semanas

Tabla 7: Plan por Fases de Predictor

2.2.2 Plan de Iteraciones del proyecto

Inicio (2/10/2006)

Objetivos

- Definición del ámbito y descripción del proyecto a realizar.
- Asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.
- Estimar el coste y programación del proyecto completo
- Estimar riesgos potenciales
- Establecer una descripción inicial de la arquitectura derivada desde los escenarios significativos

Entregables

- **Modelo del negocio**: Modelo de Casos de Uso del Negocio, Diagramas de Actividad, Glosario de Términos, Lista de Riesgos, Documento Visión, Reglas del Negocio, Modelo de Objeto del Negocio.
- **Requisitos:** Casos de Uso, Modelo de Casos de Uso, Glosario de términos, prototipo de interfaz usuario, Documento Visión (Refina), Descripción de la arquitectura (vista del modelo de casos de uso).
- **Gestión de proyectos**: Plan de Desarrollo de Software, Plan de Iteración, Plan de Iteración siguiente, Evaluación de la Iteración, Lista de Riesgos.

Disciplinas/Artefactos generados o modificados durante la Fase de Inicio	Comienzo	Aprobación
Modelado del Negocio		
Modelo de Casos de Uso del Negocio y Modelo de Objetos del Negocio	Semana 1	Semana 2
Diagramas de Actividad	Semana 1	Semana 2
Glosario de Términos	Semana 1	Semana 2
Lista de Riesgos	Semana 1	Semana 2
Reglas del Negocio	Semana 1	Semana 2
Documento Visión	Semana 2	Semana 3
Requisitos		
Documento Visión(Refinamiento)	Semana 2	Semana 3
Modelo de Casos de Uso	Semana 3	Semana 3
Especificación de Casos de Uso	Semana 3	Semana 3
Especificaciones Adicionales	Semana 2	Semana 3
Descripción de la arquitectura	Semana 3	siguiente fase
Análisis/Diseño		
Modelo de Análisis	Semana 3	siguiente fase
Modelo de Diseño	siguiente fase	siguiente fase
Modelo de Datos	Semana 2	siguiente fase
Implementación		
Prototipos de Interfaces de Usuario	Semana 2	siguiente fase
Modelo de Implementación	Semana 2	siguiente fase
Pruebas		
Casos de Pruebas Funcionales	Semana 2	siguiente fase
Despliegue		
Modelo de Despliegue	Semana 2	siguiente fase
Gestión de Cambios y Configuración	Durante todo el proyecto	
Gestión del proyecto		
Plan de Desarrollo del Software en su versión 1.0 y planes de las Iteraciones.	Semana 1	Semana 2
Ambiente	Durante todo el proyecto	

Tabla 8: Iteración 1, Fase de Inicio

Elaboración (1) (23/10/2006)

Objetivos

- Asegurar que los requisitos y planes son estables para la fase de construcción.
- Establecer la línea base de la arquitectura.
  - ✓ Refinar la arquitectura y seleccionar los componentes a hacer/comprar/reusar. Dicha selección puede conllevar una reevaluación de la arquitectura.
- Establecer un entorno estable de soporte describiendo las políticas de cambios, estructuras de directorio, esquemas de nombrado y políticas de salvaguarda.

Entregables

- **Modelo del negocio:** Modelo de Casos de Uso del Negocio (Refinar, Diagramas de Actividad (Refinar), Glosario de Términos (Refinar), Lista de Riesgos (Refinar), Reglas del Negocio (Refinar), Modelo de Objeto (Refinar).
- **Requisitos**: Casos de Uso (Refinar), Modelo de Casos de Uso (Refinar), Documentos Visión (Refinar), Especificación Suplementaria.
- **Análisis y Diseño**: Modelo de Análisis, Modelo de Diseño, Modelo de Datos, Creación de la Línea Base de la Arquitectura.
- **Gestión de Configuraciones**: Plan de Gestión de Configuración, Repositorio del Proyecto, Workspace.
- **Gestión de Proyecto**: Plan de Desarrollo de Software (Refinar), Plan de Iteración siguiente, Evaluación de la Iteración, Lista de Riesgos (Refinar)
- **Implementación**: Modelo de Implementación (primera versión)

Disciplinas/Artefactos generados o modificados durante la Fase de Elaboración – Iteración 1 (2 semanas de duración)	**Comienzo**	**Aprobación**
Modelado del Negocio		
Modelo de Casos de Uso del Negocio y Modelo de Objetos del Negocio	Semana 1	Semana 2
Diagramas de Actividad	Semana 1	Semana 2
Glosario de Términos	Semana 1	Semana 2
Lista de Riesgos	Semana 1	Semana 2
Reglas del Negocio	Semana 1	Semana 2
Documento Visión	Semana 2	Semana 3

Requisitos		
Documento Visión(Refinamiento)	Semana 2	Semana 3
Modelo de Casos de Uso	Semana 3	Semana 3
Especificación de Casos de Uso	Semana 3	Semana 3
Especificaciones Adicionales	Semana 2	Semana 3
Descripción de la arquitectura	Semana 3	Semana 5
Análisis/Diseño		
Modelo de Análisis	Semana 3	Semana 5
Modelo de Diseño	Semana 4	siguiente iteración
Modelo de Datos	Semana 2	siguiente iteración
Implementación		
Prototipos de Interfaces de Usuario	Semana 2	siguiente iteración
Modelo de Implementación	Semana 2	siguiente iteración
Pruebas		
Casos de Pruebas Funcionales	Semana 2	siguiente iteración
Despliegue		
Modelo de Despliegue	Semana 2	siguiente iteración
Gestión de Cambios y Configuración	Durante todo el proyecto	
Gestión del proyecto		
Plan de Desarrollo del Software en su versión 2.0 y planes de las Iteración 2, de elaboración.	Semana 4	Semana 4
Ambiente	Durante todo el proyecto	

Tabla 9: Iteración 1, Fase de Elaboración

Elaboración (2) (7/11/2006)

Objetivos

- Asegurar que los requisitos y planes son estables para la fase de construcción.
- Establecer la línea base de la arquitectura.
  - ✓ Refinar la arquitectura y seleccionar los componentes a hacer/comprar/reusar. Dicha selección puede conllevar una reevaluación de la arquitectura.
- Establecer un entorno estable de soporte describiendo las políticas de cambios, estructuras de directorio, esquemas de nombrado y políticas de salvaguarda.

Entregables

- **Requisitos**: Casos de Uso (Refinar), Modelo de Casos de Uso (Refinar), Documentos Visión (Refinar), Especificación Suplementaria.
- **Análisis y Diseño**: Modelo de Análisis, Modelo de Diseño, Modelo de Datos, Creación de la Línea Base de la Arquitectura.
- **Gestión de Configuraciones**: Plan de Gestión de Configuración, Repositorio del Proyecto, Workspace.
- **Gestión de Proyecto**: Plan de Iteración siguiente, Evaluación de la Iteración, Lista de Riesgos (Refinar)
- **Implementación**: Modelo de Implementación (primera versión)

Disciplinas/Artefactos generados o modificados durante la Fase de Elaboración – Iteración 2 (3 semanas de duración)	**Comienzo**	**Aprobación**
Modelado del Negocio		
Modelo de Casos de Uso del Negocio y Modelo de Objetos del Negocio	Semana 1	Semana 2
Diagramas de Actividad	Semana 1	Semana 2
Glosario de Términos	Semana 1	Semana 2
Lista de Riesgos	Semana 1	Semana 2
Reglas del Negocio	Semana 1	Semana 2
Documento Visión	Semana 2	Semana 3
Requisitos		
Documento Visión(Refinamiento)	Semana 2	Semana 3
Modelo de Casos de Uso	Semana 3	Semana 3
Especificación de Casos de Uso	Semana 3	Semana 3
Especificaciones Adicionales	Semana 2	Semana 3
Descripción de la arquitectura	Semana 3	Semana 5
Análisis/Diseño		
Modelo de Análisis	Semana 3	Semana 5
Modelo de Diseño	Semana 5	Semana 8
Modelo de Datos	Semana 7	Semana 8
Implementación		
Interfaces de Usuario	Semana 8	Semana 9
Modelo de Implementación (primera versión)	Semana 8	Semana 9

Pruebas		
Casos de Pruebas Funcionales	Semana 9	siguiente fase
Despliegue		
Modelo de Despliegue	Semana 9	siguiente fase
Gestión de Cambios y Configuración	Durante todo el proyecto	
Gestión del proyecto		
Plan de Desarrollo del Software en su versión 3.0 y planes de las Iteración 1 de construcción	Semana 7	Semana 7
Ambiente	Durante todo el proyecto	

Tabla 10: Iteración 2, Fase de Elaboración

Construcción (1) (29/11/2007)

Objetivo

- Completar análisis, diseño, desarrollo y pruebas de la funcionalidad esencial.
- Obtener versión útil (1).
- Iterativa e incrementalmente desarrollar un primer prototipo del producto.
- Puesta en práctica de las políticas de gestión de cambios para aquellas solicitudes de cambio procedentes de la ejecución de las pruebas.

Entregables

- **Análisis y Diseño**: Arquitectura Software(Refinar), Modelo de Diseño (Refinar), Modelo de Datos (Refinar)
- **Gestión de Configuraciones**: Plan de Gestión de Configuración (Refinar), Repositorio del Proyecto (Refinar)
- **Gestión de Proyecto**: Plan de Iteración siguiente, Evaluación de la Iteración, Lista de Riesgos (Refinar)
- **Pruebas:** Plan de pruebas, casos de prueba, implementación casos de prueba, evaluación casos de prueba.
- **Implementación:** Modelo de Implementación (Refinar), prototipo del sistema, implementación de cambios.

Disciplinas/Artefactos generados o modificados durante la Fase de Construcción – Iteración 1 (2 semanas de duración)	Comienzo	Aprobación
Modelado del Negocio		
Modelo de Casos de Uso del Negocio y Modelo de Objetos del Negocio	Semana 1	Semana 2
Diagramas de Actividad	Semana 1	Semana 2
Glosario de Términos	Semana 1	Semana 2
Lista de Riesgos	Semana 1	Semana 2
Reglas del Negocio	Semana 1	Semana 2
Documento Visión	Semana 2	Semana 3
Requisitos		
Documento Visión(Refinamiento)	Semana 2	Semana 3
Modelo de Casos de Uso	Semana 3	Semana 3
Especificación de Casos de Uso	Semana 3	Semana 3
Especificaciones Adicionales	Semana 2	Semana 3
Descripción de la arquitectura	Semana 3	Semana 5
Análisis/Diseño		
Modelo de Análisis	Semana 3	Semana 5
Modelo de Diseño	Semana 5	Semana 8
Modelo de Datos	Semana 7	Semana 8
Implementación		
Interfaces de Usuario (refinamiento)	Semana 8	Semana 9
Modelo de Implementación (refinamiento)	Semana 8	Semana 9
Implementación v1.0	Semana 8	Semana 10
Pruebas		
Casos de Pruebas Funcionales	Semana 10	Semana 10
Despliegue		
Modelo de Despliegue	Semana 10	Semana 10
Gestión de Cambios y Configuración	Durante todo el proyecto	
Gestión del proyecto		
Plan de Desarrollo del Software en su versión 4.0 y planes de las Iteración 2 de Construcción	Semana 10	Semana 10
Ambiente	Durante todo el proyecto	

Tabla 11: Iteración 1, Fase de Construcción

Construcción (2) (14/12/2007)

Objetivos

- Obtener versión útil (2).
- Completar análisis, diseño, desarrollo y pruebas de la funcionalidad esencial.
- Iterativa e incrementalmente desarrollar un segundo prototipo del producto, casi listo para la instalación.

Entregables

- **Análisis y Diseño**: Arquitectura Software (Refinar), Modelo de Diseño (Refinar), Modelo de Datos (Refinar).
- **Gestión de Configuraciones**: Plan de Gestión de Configuración (Refinar), Repositorio del Proyecto (Refinar).
- **Gestión de Proyecto:** Plan de Iteración siguiente, Evaluación de la Iteración, Lista de Riesgos (Refinar)
- **Pruebas:** Plan de pruebas (Refinar), casos de prueba, implementación casos de prueba, evaluación casos de prueba.
- **Implementación:** modelo de Implementación (Refinar), prototipo del sistema, implementación de cambios.

Disciplinas/Artefactos generados o modificados durante la Fase de Construcción – Iteración 2 (1 semanas de duración)	Comienzo	Aprobación
Modelado del Negocio		
Modelo de Casos de Uso del Negocio y Modelo de Objetos del Negocio	Semana 1	Semana 2
Diagramas de Actividad	Semana 1	Semana 2
Glosario de Términos	Semana 1	Semana 2
Lista de Riesgos	Semana 1	Semana 2
Reglas del Negocio	Semana 1	Semana 2
Documento Visión	Semana 2	Semana 3
Requisitos		

Documento Visión(Refinamiento)	Semana 2	Semana 3
Modelo de Casos de Uso	Semana 3	Semana 3
Especificación de Casos de Uso	Semana 3	Semana 3
Especificaciones Adicionales	Semana 2	Semana 3
Descripción de la arquitectura	Semana 3	Semana 5
Análisis/Diseño		
Modelo de Análisis	Semana 3	Semana 5
Modelo de Diseño	Semana 10	Semana 10
Modelo de Datos	Semana 7	Semana 8
Implementación		
Interfaces de Usuario (refinamiento)	Semana 8	Semana 10
Modelo de Implementación (refinamiento)	Semana 8	Semana 10
Implementación v2.0	Semana 10	Semana 11
Pruebas		
Casos de Pruebas Funcionales	Semana 11	Semana 11
Despliegue		
Modelo de Despliegue	Semana 10	Semana 10
Gestión de Cambios y Configuración	Durante todo el proyecto	
Gestión del proyecto		
Plan de Desarrollo del Software en su versión 5.0 y planes de las Iteración 3 de Construcción	Semana 11	Semana 11
Ambiente	Durante todo el proyecto	

Tabla 12: Iteración 2, Fase de Construcción.

Construcción (3) (9/1/2007)

Objetivos

- Obtener versión útil (3).
- Completar análisis, diseño, desarrollo y pruebas de la funcionalidad esencial.
- Iterativa e incrementalmente desarrollar un segundo prototipo del producto, casi listo para la instalación.

Entregables

- **Análisis y Diseño**: Arquitectura Software (Refinar), Modelo de Diseño (Refinar), Modelo de Datos (Refinar).
- **Gestión de Configuraciones**: Plan de Gestión de Configuración (Refinar), Repositorio del Proyecto (Refinar).
- **Gestión de Proyecto:** Plan de Iteración siguiente, Evaluación de la Iteración, Lista de Riesgos (Refinar)
- **Pruebas:** Plan de pruebas (Refinar), casos de prueba, implementación casos de prueba, evaluación casos de prueba.
- **Implementación:** modelo de Implementación (Refinar), prototipo del sistema, implementación de cambios.

Disciplinas/Artefactos generados o modificados durante la Fase de Construcción – Iteración 3 (1 semanas de duración)	Comienzo	Aprobación
Modelado del Negocio		
Modelo de Casos de Uso del Negocio y Modelo de Objetos del Negocio	Semana 1	Semana 2
Diagramas de Actividad	Semana 1	Semana 2
Glosario de Términos	Semana 1	Semana 2
Lista de Riesgos	Semana 1	Semana 2
Reglas del Negocio	Semana 1	Semana 2
Documento Visión	Semana 2	Semana 3
Requisitos		
Documento Visión(Refinamiento)	Semana 2	Semana 3
Modelo de Casos de Uso	Semana 3	Semana 3
Especificación de Casos de Uso	Semana 3	Semana 3
Especificaciones Adicionales	Semana 2	Semana 3
Descripción de la arquitectura	Semana 3	Semana 5
Análisis/Diseño		
Modelo de Análisis	Semana 3	Semana 5
Modelo de Diseño	Semana 10	Semana 10
Modelo de Datos	Semana 7	Semana 8
Implementación		
Interfaces de Usuario (refinamiento)	Semana 8	Semana 10

Modelo de Implementación (refinamiento)	Semana 8	Semana 10
Implementación v3.0	Semana 11	Semana 12
Pruebas		
Casos de Pruebas Funcionales	Semana 11	Semana 11
Despliegue		
Modelo de Despliegue	Semana 10	Semana 10
Gestión de Cambios y Configuración	Durante todo el proyecto	
Gestión del proyecto		
Plan de Desarrollo del Software en su versión 6.0 y planes de las Iteración 1 de transición	Semana 12	Semana 12
Ambiente	Durante todo el proyecto	

Tabla 13: Iteración 3, Fase de Construcción

Transición (1) (17/1/2007)

Objetivo

- Entregar un release listo para una instalación real.
- Corregir errores que puedan surgir a la hora de la instalación.
- Despliegue del producto.
- Soporte y transferencia de tecnología al cliente.

Entregables

- Release listo para la instalación.
- Documentación de ayuda y soporte.

Disciplinas/Artefactos generados o modificados durante la Fase de Transición – Iteración 1 (2 semanas de duración)	Comienzo	Aprobación
Modelado del Negocio		
Modelo de Casos de Uso del Negocio y Modelo de Objetos del Negocio	Semana 1	Semana 2
Diagramas de Actividad	Semana 1	Semana 2
Glosario de Términos	Semana 1	Semana 2

Lista de Riesgos	Semana 1	Semana 2
Reglas del Negocio	Semana 1	Semana 2
Documento Visión	Semana 2	Semana 3
Requisitos		
Documento Visión(Refinamiento)	Semana 2	Semana 3
Modelo de Casos de Uso	Semana 3	Semana 3
Especificación de Casos de Uso	Semana 3	Semana 3
Especificaciones Adicionales	Semana 2	Semana 3
Descripción de la arquitectura	Semana 3	Semana 5
Análisis/Diseño		
Modelo de Análisis	-	-
Modelo de Diseño (refinamiento v3.0)	-	-
Modelo de Datos	-	-
Implementación		
Interfaces de Usuario (refinamiento)	-	-
Modelo de Implementación (refinamiento)	-	-
Implementación	-	-
Pruebas		
Casos de Pruebas Funcionales	-	-
Despliegue		
Modelo de Despliegue	-	-
Entrega del release	-	-
Despliegue del producto	Semana 13	Semana 14
Soporte y transferencia	Semana 13	Semana14
Gestión de Cambios y Configuración	Durante todo el proyecto	
Ambiente	Durante todo el proyecto	

Tabla 14: Iteración 1, Fase de Transición

2.3 Plan de gestión de riesgos

2.3.1 Fases de la Gestión de Riesgo

2.3.1.1 Identificación de los riesgos

La realización de las entrevistas a los miembros del equipo y a otras personas arrojó la lista de riesgos mostrada a continuación la cual presenta una descripción de todos los riesgos identificados

1. *Cambio constante de los requerimientos*: Actualmente los requerimientos son inestables, no existe formalmente un proceso estructurado de captura de requerimientos, pues se hace de manera informal y no se firma ningún contrato de acuerdo con los clientes; si esto ocurre el cliente tiene el poder de cambiar de idea constantemente, sin importar el estado actual de los requerimientos pactados.
2. *Tiempo inadecuado para planificar*: Este proceso, es una de las partes más importantes del desarrollo de software. Actualmente el tiempo que se da a la planificación es bien poco; si se tiene en cuenta que el personal encargado de esto, cumple otras funciones en la organización. Esto trae como consecuencia una pobre estimación de las tareas.
3. *Planificación ambiciosa del calendario de trabajo*: Los retos que se enfrentan durante el desarrollo de una aplicación a lo largo de tres meses, son muy diferentes a los acarreados por un desarrollo superior al año. Fijar un plan excesivamente agresivo predispone a que el proyecto falle por falta de tiempo. También supone excesiva presión para los desarrolladores.
4. *Dependencia de las tareas*: Los líderes de cada equipo, subdividen las tareas principales en pequeñas sub-tareas. En ocasiones existen tareas que dependen de otras. Esto es un problema en el momento de planificar, pues afectan el desarrollo del software, si no se tiene en consideración.
5. *Falta de flexibilidad en el diseño*: Los diseños deben ser lo suficientemente flexibles para ser reusados en otras partes o aplicaciones. En ocasiones por falta de tiempo estos son muy acoplados a la aplicación en específico.
6. *Demora en el proceso de revisión del código*: El personal debe revisar el código para luego ser integrado. El gran problema de esta fase es que suele consumir mucho tiempo y a veces no se llega a completar una revisión precisa por falta de tiempo.
7. *Problemas inesperados en la etapa de integración*: En la integración se juntan los pequeños módulos de software desarrollados y que parece que funcionan. En ocasiones surgen errores a la hora de integrarlos y se debe tener la capacidad de afrontarlos rápidamente.
8. *Inexperiencia del personal:* La mayoría del personal presenta mucho potencial pero muy poca experiencia; debido a que todos los miembros son estudiantes de Pregrado de la especialidad de

Ingeniería en Ciencias Informáticas. Esto ocasiona que a veces exista personal con ciertas dosis de carga, con algunas tareas.

9. *Pobre planificación del programa de formación de gestión de procesos*: Incorporar metodologías requiere entrenamientos, la gente esta acostumbrada a trabajar de una forma y se debe dar seguimiento de cómo trabajar de otra.
10. *La existencia de un proceso de evaluación personal*: No existe un proceso de evaluación personal, es decir no hay métricas para medir su desempeño y poder dar retroalimentación de su trabajo.
11. *Gestión de riesgos insuficientes*: No existe un proceso de gestión de riesgos. Se ha pensado en una lista de riesgos en un futuro, pero no existe un proceso definido de cómo analizar y controlar los riesgos.
12. *Falta de disponibilidad efectiva de los recursos*: Afecta el desarrollo del software y afecta la calidad del producto.
13. *Problemas en el proceso de definición y control de interfaces*: Al no existir estándar de diseño; aspectos como el cambio de requerimientos dan origen a un diseño riesgoso.
14. *Pobre participación del usuario*: Los proyectos que desde un inicio no implican al usuario corren el riesgo de no comprender los requerimientos del proyecto.
15. *Falta de participación entre los implicados*: Todos los principales participantes del desarrollo de la aplicación deben implicarse. En ocasiones no existe una buena participación entre los implicados, debido a la falta de tiempo del personal para asumir la implementación de este proceso de construcción del software.
16. *Problemas de confiabilidad de software*: Debido al poco tiempo que se tiene en todo el ciclo de desarrollo del código, se pueden presentar problemas de inestabilidad del código.
17. *Cambio constante de estimación del plan*: Un tipo de reestimación es responder inapropiadamente al retraso del plan. En este caso, cuando ocurren factores sorpresas, se presiona al personal a trabajar a su máxima capacidad y cumplir con su fecha de entrega, sin tener en cuenta la calidad final del producto.
18. *Ambiente inadecuado para el desarrollo de las aplicaciones*: En ocasiones se presentan en el desarrollo de las aplicaciones; como por ejemplo: problemas con el sistema operativo o con las herramientas de desarrollo utilizadas.

19. *Cumplimiento del horario de trabajo*: En ocasiones producto de actividades docentes de la universidad o extracurriculares existe desaprovechamiento del tiempo destinado a la producción.
20. *Rotura de una Computadora*: Se puede presentar la situación de que una computadora de las cuales tiene asignada el proyecto para desarrollar el producto software deje de funcionar.
21. *Ausencia de un integrante del equipo de desarrollo*: En ocasiones a algunos estudiantes se le asignan otras tareas por parte de la facultad o de la universidad y deben dejar temporalmente el proyecto.

En la lista Maestra de riesgos que aparece a continuación están los riesgos definidos por las entrevistas y los sacados de la bibliografía los cuales coincidieron, por lo tanto se dejaron con el nombre que ya tenían para mejor comprensión.

	Lista Maestra de Riesgos
1.	Cambio constante de los requerimientos
2.	Tiempo inadecuado para la planificación
3.	Planificación ambiciosa del calendario de trabajo
4.	Dependencia de las tareas
5.	Falta de flexibilidad en el diseño
6.	Demora en el proceso de revisión del código
7.	Problemas inesperados en la etapa de integración
8.	Inexperiencia del personal
9.	Pobre planificación del programa de formación de gestión de procesos
10.	La existencia de un proceso de evaluación personal
11.	Gestión de riesgos insuficientes
12.	Falta de disponibilidad efectiva de los recursos
13.	Problemas en el proceso de definición y control de interfaces
14.	Pobre participación del usuario
15.	Falta de participación entre los implicados
16.	Problemas de confiabilidad de software
17.	Cambio constante de estimación del plan

18.	Ambiente inadecuado para el desarrollo de las aplicaciones
19.	Cumplimiento del horario de trabajo
20.	Rotura de una computadora.
21.	Ausencia de un integrante del equipo de desarrollo.

Tabla 15: Lista Maestra de Riesgos

2.3.1.2 Análisis de Riesgos

Teniendo en cuenta la manera definida sobre la cual se hacer el análisis de los riesgos y con la lista maestra de riesgos obtenida en la fase anterior podemos establecer el orden de prioridad que debe seguirse a la hora de seleccionar los 10 primeros riesgos a los cuales de les debe realizar la planeación de respuesta.

Evolución de Riesgos					
	Riesgo	**Probabilidad**	**Impacto**	**Exposición al riesgo**	**Orden de prioridad**
1.	Cambio constante de los requerimientos	0.7	0.80	0.56	1
2.	Tiempo inadecuado para planifica	0.5	0.40	0.20	4
3.	Planificación ambiciosa del calendario de trabajo	0.5	0.40	0.20	4
4.	Dependencia de las tareas	0.9	0.10	0.09	8
5.	Falta de flexibilidad en el diseño	0.7	0.20	0.14	6
6.	Demora en el proceso de revisión del código	0.7	0.20	0.14	6
7.	Problemas inesperados en la etapa de integración	0.5	0.40	0.20	4
8.	Inexperiencia del personal	0.7	0.40	0.28	3
9.	Pobre planificación del programa de formación de gestión de procesos	0.5	0.20	0.10	10
10.	La existencia de un proceso de evaluación	0.3	0.10	0.03	11

	personal				
11.	Gestión de riesgos insuficientes	0.7	0.80	0.56	1
12.	Falta de disponibilidad efectiva de los recursos	0.5	0.20	0.10	7
13.	Problemas en el proceso de definición y control de interfaces	0.5	0.10	0.05	10
14.	Pobre participación del usuario	0.7	0.40	0.28	3
15.	Falta de participación entre los implicados	0.5	0.80	0.40	2
16.	Problemas de confiabilidad de software	0.5	0.40	0.20	4
17.	Cambio constante de estimación del plan	0.7	0.80	0.56	1
18.	Ambiente inadecuado para el desarrollo de las aplicaciones	0.3	0.20	0.06	9
19.	Cumplimiento del horario de trabajo	0.9	0.20	0.18	5
20.	Rotura de una computadora.	0.5	0.40	0.20	4
21.	Ausencia de un integrante del equipo de desarrollo.	0.7	0.40	0.28	3

Tabla 16: Evaluación de Riegos

2.3.1.3 Fase de Planeación de respuestas de Riesgos

A continuación se exponen las estrategias a seguir para cada uno de los riegos identificados. Teniendo en cuenta los 10 riesgos que mayor orden de prioridad tienen:

1. **Riesgo:** Cambio constante de requerimientos.
   Tipo: Organizacional
   Probabilidad: 0.7 **Impacto:** 0.80 **Riesgo Percibido:** 0.56
   Detalle: Actualmente los requerimientos son inestables.
   Respuesta al riesgo: Mitigar.

Estrategia Recomendada: Realizar un buen proceso de captura de requisitos. Una vez que el cliente identifique sus necesidades, se deberá negociar directamente con el área de desarrollo con el propósito de identificar la viabilidad de los requerimientos. Finalmente la lista actualizada deberá regresar al cliente y este confirmará si esta de acuerdo, firmando finalmente un contrato para cerrar el negocio.

2. **Riesgo:** Gestión de Riesgos Insuficiente.
   Tipo: Organizacional
   Probabilidad: 0.7 **Impacto:** 0.80 **Riesgo Percibido:** 0.56
   Detalle: No existe un proceso de gestión de riesgos.
   Respuesta al riesgo: Mitigar.
   Estrategia Recomendada: Implementar una cultura en la gestión de los riesgos en todo el personal del proyecto. Elaborara un plan de gestión de riesgos que permita identificar los principales riesgos a los cuales esta expuesto el equipo de desarrollo y controlar. Para esto es necesario llevar a cabo reuniones e informes periódicos sobre el estado de los riesgos.

3. **Riesgo:** Cambio constante de estimación del plan.
   Tipo: Organizacional
   Probabilidad: 0.7 **Impacto:** 0.80 **Riesgo Percibido:** 0.56
   Detalle: Un tipo de reestimación es responder inapropiadamente al retraso del plan.
   Respuesta al riesgo: Mitigar
   Estrategia Recomendada: Desarrollar un plan que tenga en cuenta todas las contingencias que pueden ocurrir a lo largo del proyecto, haciéndolo flexible.

4. **Riesgo:** Falta de participación entre los implicados.
   Tipo: Organizacional
   Probabilidad: 0.5 **Impacto:** 0.80 **Riesgo Percibido:** 0.40
   Detalle: Todos los principales participantes del desarrollo de la aplicación deben implicarse.
   Respuesta al riesgo: Mitigar

Estrategia Recomendada: Lograr que todos los miembros implicados en el desarrollo del producto se comprometan con este. Establecer desde un principio sesiones de trabajo donde cada uno de los implicados rinda cuenta de las tareas que tienen.

5. **Riesgo:** Pobre participación del usuario
   Tipo: Organizacional
   Probabilidad: 0.7 **Impacto:** 0.40 **Riesgo Percibido:** 0.28
   Detalle: Los proyectos que desde un inicio no implican al usuario corren el riesgo de no comprender los requerimientos del proyecto.
   Respuesta al riesgo: Mitigar.
   Estrategia Recomendada: Contar con la participación del cliente en el proceso de planificación y de las iteraciones. Se requiere que el cliente participe en cada iteración a lo largo del proyecto para proveer de retroalimentación al equipo, garantizando el cumplimiento de las expectativas que tiene.

6. **Riesgo:** Inexperiencia del personal
   Tipo: Organizacional
   Probabilidad: 0.7 **Impacto:** 0.40 **Riesgo Percibido:** 0.28
   Detalle: La mayoría del personal presenta mucho potencial pero muy poca experiencia.
   Respuesta al riesgo: Mitigar
   Estrategia Recomendada:
   1. Trabajo en parejas: Está basado en el principio de que dos personas trabajando juntas pueden hacer más que por separado. El resultado de esto es una mejora en la calidad de las tareas y además no supone tardar más tiempo.
   2. Rotar el personal del equipo por los diferentes puestos de trabajo, ya que todo el grupo no puede depender de una persona que sea la única que tenga el conocimiento sobre un área en específico del producto. Esto permitiría flexibilidad y se contaría con personal preparado para realizar cualquier reasignación.
   3. Programación de talleres y conferencias para la superación del personal.

7. **Riesgo:** Problemas de confiabilidad de software.
   Tipo: Organizacional
   Probabilidad: 0.5 **Impacto:** 0.40 **Riesgo Percibido:** 0.20
   Detalle: Debido al poco tiempo que se tiene en todo el ciclo de desarrollo del código, se pueden presentar problemas de inestabilidad del código.
   Respuesta al riesgo: Mitigar.
   Estrategia Recomendada: Realizar un correcta planificación del ciclo de desarrollo del código lo cual a los desarrolladores poder realizar su trabajo de manera cómoda y realizarle pruebas a lo que están desarrollando antes de someter el producto a las pruebas de integración.

8. **Riesgo:** Problemas inesperados en la etapa de integración.
   Tipo: Organizacional
   Probabilidad: 0.5 **Impacto:** 0.40 **Riesgo Percibido:** 0.20
   Detalle: En ocasiones surgen errores a la hora de integrar los módulos y se debe tener la capacidad de afrontarlos rápidamente.
   Respuesta al riesgo: Mitigar.
   Estrategia Recomendada: Adoptar un método de desarrollo basado en las pruebas para asegurar que el código se comporta según lo esperado.
   Asumir la propiedad colectiva del código, esto significa que la propiedad del sistema en cierto modo la decide el colectivo en conjunto.
   Integración continúa, esto quiere decir que debemos actualizar cada pocas horas el código existente en el repositorio, así todos los desarrolladores cuando vayan a hacer integraciones los harán sobre las versiones más actualizadas.

9. **Riesgo:** Planificación ambiciosa del calendario de trabajo.
   Tipo: Organizacional.
   Probabilidad: 0.5 **Impacto:** 0.40 **Riesgo Percibido:** 0.20
   Detalle: Fijar un plan excesivamente agresivo predispone a que el proyecto falle por falta de tiempo. También supone excesiva presión para los desarrolladores.
   Respuesta al riesgo: Mitigar.

Estrategia Recomendada: Elaborar una planificación que se ajuste a las condiciones del producto a desarrollar y además a las condiciones que tienes los miembros del equipo, estudiantes de 5to año de la carrera, los cuales deben tener un horario flexible ya que constantemente están apareciendo actividades extracurriculares.

10. **Riesgo:** Tiempo inadecuado para planificar.
    Tipo: Organizacional.
    Probabilidad: 0.5 **Impacto:** 0.40 **Riesgo Percibido:** 0.20
    Detalle: Actualmente el tiempo que se da a la planificación es bien poco.
    Respuesta al riesgo: Mitigar.
    Estrategia Recomendada: Aumento de la etapa de planificación, para que el personal pueda estimar correctamente las actividades a realizar. Para esto de 4 semanas que consta la iteración de la fase de inicio se extiende a 5 semanas, para lograr una planificación adecuada.

2.3.1.4 Fase de Seguimiento y control de Riesgo

Esta fase es muy importante ya que es donde se controlan los riesgos que aparecen en la lista actual de riegos, pero además se debe estar muy atento ya que pueden aparecer nuevos riesgos en su entorno a medida que avanza el proyecto. En la tabla que continuación aparece un formulario que es utilizado para el seguimiento y control de los riegos.

ID	Prioridad	Respuesta al riesgo	Estado	Fecha del estado	Aprobado	Comentarios
#	alta, media y baja	Acciones planteadas	Activo o inactivo	fecha	Nombre de la persona que esta haciendo el control de riesgo	

Tabla 17: Seguimiento y control

Para más información sobre la planeación completa de los riesgos y el seguimiento y control de los mismos puede dirigirse al Trabajo de Diploma: Predictor: Sistema de descarga y procesamiento automatizado de patentes. Rol de Planificador" de la autora Yailin López Batista.

2.4 Plan de métricas del proyecto

Como bien se planteo en el capitulo anterior, el plan de métricas se debe ajustar a las necesidades de la organización o el proyecto, de lo contrario entorpecería el trabajo de los desarrolladores.

2.4.1 Métricas al Modelo de casos de uso del Sistema

El Modelo de Métricas para Casos de Uso presentado tiene por objetivo medir la calidad de los casos de uso generados en este proyecto de software. Dicho modelo define cuatro atributos genéricos de propiedades de calidad: consistencia, correctitud, completitud y complejidad, que tienen un significado concreto de acuerdo al tipo de artefacto software y al nivel de abstracción que éste describe. Un atributo se analiza en términos de un conjunto de factores cada uno de los cuales tendrá asociada una métrica.

Para el Modelo de casos de uso del Sistema se definieron tres atributos principales y de acuerdo a las métricas relacionadas con cada uno de los factores asociados a dichos atributos se evaluó la calidad de los mismos. Las tablas que muestran los factores y las métricas por las cuales se realizo la evolución de la calidad, se muestran en los anexos:

Anexó A: Tabla 1: Definición de factores por nivel de abstracción.
Anexó A: Tabla 2: Definición de Métricas por Factores.

Para más información aplicación de estas métricas puede dirigirse al Trabajo de Diploma: Sistema de descarga y procesamiento automatizado de patentes. Rol de Análisis" de las autora Yaniet Piñeiro y Mayrelis Plaza.

2.4.2 Medición de las líneas de código

Para esta tarea se realizarán pruebas unitarias que tributarán a la evaluación del correcto funcionamiento de un módulo de código. El propósito principal es: aislar cada parte de programa, mostrar que las partes individuales son correctas, proporciona un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan 5 ventajas básicas:

- **Fomentan el cambio:** Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.
- **Simplifica la integración**: debido que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.
- **Documenta el código**: Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.
- **Separación de la interfaz y la implementación.**
- **Los errores están más acotados y son más fáciles de localizar**: dado que tenemos pruebas unitarias que pueden desenmascararlos.

Para ello se uso la herramienta NUnit, esta herramienta permite diseñar los casos de prueba que serán puestos en práctica de forma independiente en el sistema.

2.4.3 Métricas del mantenimiento

Las métricas para el mantenimiento del software existente. El estándar IEEE 982.1-1988 sugiere el índice de madurez del software (IMS) que proporciona una indicación de la estabilidad de un producto

software basada en los cambios que ocurren con cada versión del producto. Con el IMS se determina la siguiente información:

- MT= Número de módulos en la versión
- actualFc= Número de módulos en la versión actual que se han cambiado
- Fa= Número de módulos en la versión actual que se han añadido
- Fe= Número de módulos en la versión actual que se han eliminado 50

El índice de madurez del software se calcula de la siguiente manera:

- IMS= [MT - (Fc + Fa + Fe)]/MT

A medida que el IMS se aproxima a 1 el producto se empieza a estabilizar.

2.4.5 Métricas de diseño aplicadas

- Métricas de diseño arquitectónico

Son consideradas métricas de alto nivel. Miden la eficiencia de la arquitectura y los módulos que conforman el sistema. No se necesita conocer el funcionamiento interno de los módulos para evaluarlos, por ello se consideran como métricas de caja negra. Para medir estas métricas se defines 3 niveles de complejidad.

La complejidad estructural: definida como: $S(i) = f^{2}out(i)$, donde i es el módulo y $f^{2}_{out}(i)$ es la expansión de i.

La complejidad de datos: proporciona una indicación de la complejidad en la interfaz interna de un módulo, se define como: $D(i) = v(i)/[f_{out}(i)+1]$, donde $v(i)$ es el número de variables de entrada del módulo i.

La complejidad del sistema: que es la suma de las complejidades estructural y de datos, se define como: $C(i) = S(i) + D(i)$.

Si aumentan los valores de complejidad, entonces la complejidad arquitectónica del sistema también aumenta, provocando que aumente el esfuerzo para la integración y las pruebas.

- Métricas de Acoplamiento

El acoplamiento de un módulo proporciona una indicación de la conectividad del mismo con otros módulos, datos globales y el entorno exterior. A continuación se muestran las medidas que se necesitan para aplicar esta métrica:

- ✓ *Acoplamiento de flujo de datos y de control:* d_i (número de parámetros de datos de entrada), c_i (número de parámetros de control de entrada), d_o (número de parámetros de datos de salida) y c_o (número de parámetros de control de salida).
- ✓ *Acoplamiento global:* g_d (número de variables globales usadas como datos) y g_c (número de variables globales usadas como control).
- ✓ *Acoplamiento de entorno:* w (número de módulos llamados (expansión)) y r (número de módulos que llaman al módulo en cuestión (concentración)).

Para esta métrica se define como indicador de acoplamiento del módulo de la siguiente manera: $m_c = k/M$, donde k es una constante que toma valor 1 y $M = d_i + a * c_i + d_o + b * c_o + g_d + c * g_c + w + r$, donde $a = b = c = 2$. Cuanto mayor es el valor de m_c, menor es el acoplamiento del módulo.

- Métricas para Sistemas Orientados a Objetos

La POO ha alcanzado gran auge en las últimas décadas, esta permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar. Junto con la explosión de la POO también

surgieron varias métricas con el fin de solucionar el problema de las mediciones a los sistemas OO.

- ✓ Métricas propuestas por Lorenz y Kidd

  Según lo propuestos por estos autores, se utilizara las métricas orientadas a tamaño, las cuales se centran en contar los atributos y operaciones de cada clase, Tamaño de clase (TC).

- ✓ Métricas propuestas por Chidamber y Kemerer

  Es uno de los conjuntos de métricas más difundidos y conocidas como las CK, también se les llama MOOSE. Adoptan tres criterios a la hora de definirlas: capacidad de satisfacer propiedades analíticas, aspecto intuitivo a los profesionales y facilidad para su recogida automática. En el proyecto se utilizó: Profundidad del árbol de herencia (PAH).

Para más información sobre la aplicación de estas métricas puede dirigirse al Trabajo de Diploma: Sistema de descarga y procesamiento automatizado de patentes. Rol de Diseñador" del autor Alberto Limia.

2.5 Plan de calidad

Tiene como propósito el control de los procesos y actividades que se realizan para la construcción del software Predictor: Sistema para la descarga y procesamiento automatizado de patentes. El plan tiene una gran importancia ya que permite controlar el desarrollo del producto para que satisfaga las necesidades del cliente.

2.5.1. Referencias

Título	Fecha	Autor	Ubicación (anexo, documento)
Plan de pruebas	3/11/2006	Vlamir Rodríguez Fernández	
Plan de desarrollo del proyecto	20/10/2006	Vlamir Rodríguez Fernández	
Plan de gestión de riesgos	6/10/2006	Vlamir Rodríguez Fernández, Yailin López Batista	
Plan de Estimación	4/10/2006	Yailin López Batista	

Tabla 18: Referencia a documentos con relación con la calidad

2.5.2. Gestión

2.5.2.1. Organización

La documentación de calidad se encuentra en el repositorio de información del proyecto como un componente del expediente de proyecto, esta organizada en carpetas independientes los planes (gestión de riesgos, iteraciones, pruebas, el propio plan de calidad), listas de chequeo, etc.

El proyecto consta de 9 integrantes, por lo tanto no se organizó grupo de calidad debido a la escasa cantidad de recursos humanos, elemento que propició llevar a cabo un plan de calidad con la estructura siguiente:

- Líder de proyecto hizo función de *administrador de la calidad* el cual es el encargado de dirigir, organizar y diseñar todas las actividades de aseguramiento de la calidad en el proyecto, así como de cuidar que sean cumplidas por todos sus integrantes en aras de lograr fomentar la calidad total tan necesaria para obtener resultados exitosos en la elaboración del software.

- Uno de los programadores se especializó en las actividades de calidad y fue el encargado de cumplir el rol de *diseñador de pruebas* que tiene como responsabilidad diseñar los casos de uso de prueba, Planificar las pruebas, y evaluar los resultados.
- Además se designaron dos compañeros indistintamente para que realizaran el rol de *revisor técnico*: Se encarga de realizar las revisiones técnicas y las auditorías al proyecto.
- Probador: Ejecuta las pruebas diseñadas y anota los resultados obtenidos.

2.5.2.2. Tareas y responsabilidades

Tarea de Aseguramiento de calidad	Entrada	Salida	Responsable	Comentarios
Revisión del modelo de negocio	Artefactos del Modelos de Negocio	Plantilla de evaluación.	Sergio J. García	
Revisión de los requisitos	Documentos de especificación de requisitos de sofware(ERS)	Plantilla de evaluación.	Vlamir Rodríguez Sergio J. García	
Revisión del plan de Mitigación de riesgos	Plan de Mitigación de riesgos	Plantilla de evaluación.	Vlamir y compañeros seleccionados del proyecto	
Revisión del análisis	Artefactos del Modelo de Análisis	Plantilla de evaluación.	Alberto Limia	
Auditoría interna del proceso #1	Artefactos generados al final de la fase de elaboración.	Plantilla de evaluación General que incluye la evaluación de las anteriores.	Mileisis Placencia	
Revisión del diseño preliminar	Descripción de Diseño del Software (Los artefactos generados)	Plantilla de evaluación.	Mileisis Placencia	
Revisión del Diseño Final	Descripción de Diseño del Software Final (Los artefactos generados)	Plantilla de evaluación.	Mileisis Placencia	
Revisión del		Plantilla de	Yoandry Castellanos	

modelo de implementación		evaluación.		
Auditoría interna del proceso #2	Artefactos generados en la fase de elaboración como el modelo de diseño, Modelo de despliegue, línea base de la arquitectura, etc.	Plantilla de evaluación General que incluye la evaluación de las anteriores.	Vlamir Rodríguez Yoandry Castellanos	
Auditoría funcional	Producto, ERS	Plantilla de evaluación.	Yaniet Piñeiro Mayrelis Plaza	
Auditoría física	Producto y Documentación.	Plantilla de evaluación.	Mileisis Placencia Sergio J. García.	
Certificación final del producto	Prueba Producto	Plantilla de evaluación de prueba	Vlamir Rodríguez	
Certificación por parte del Cliente	Producto	Evaluación echa por el cliente.	Yailin López	
Evaluación de la satisfacción del cliente	Despliegue del producto	Aval y valoración del cliente	Vlamir Rodríguez	

Tabla 19: Tareas y responsabilidad

2.5.3. Estándares

- *Codificación:* en nuestro proyecto se define un estándar de codificación basado en el lenguaje de programación C#, sobre la plataforma .NET, en la definición del estándar trabaja el arquitecto y los programadores.

  - ✓ Indentación.
  - ✓ Comentarios.
  - ✓ Nomenclatura de las clases, variables, atributos y métodos.
  - ✓ Prácticas de programación

- *Control de Documentación:* la utilización de un repositorio del proyecto, donde un compañero destinado para esta tarea, debe controlar la buena organización y control de los documentos dentro de este.
- *Interfaz de Usuario:* La interfaz generada en el proyecto debe cumplir con los estándares establecidos por la universidad para todos los productos que se realizan en esta.
- *Formato de la documentación:* se define todo los aspectos sobre los cuales se deben escribir todo documento generado por el proyecto, la configuración de la página, el tamaño de letra, tipo de fuente, interlineado y el tipo de viñetas.
- *Estándar IEEE:* para desarrollar el plan de calidad utilizamos el estándar IEEE Std 730-1998 que es una versión revisada del IEEE Std 730-1989. El mismo describe un conjunto de pasos para la confección del plan de aseguramiento de calidad que constituyen una excelente guía para lograr la organización y planificación de todas las tareas de calidad que nuestro proyecto requiere.

2.5.5. Plan de Revisiones y Auditorías

1. Revisión del modelo del negocio: el modelo del negocio permite la identificación de todos los procesos del negocio y actividades fundamentales que ocurren dentro de cada uno de ellos. La información que proporciona es muy útil para la futura captura de requisitos. Dentro de los aspectos fundamentales que se tienen en cuenta en la lista de comprobación definida para este artefacto a continuación se listará una selección de ellos:
    - Referente al Diagrama de Actividad:
        - ✓ Cada Diagrama de Actividad tiene que tener uno y sólo un estado inicial y como mínimo un estado final.
        - ✓ De toda actividad hay que transitar siempre a: otra actividad, una decisión, una barra de sincronización o un estado final.
        - ✓ Toda actividad debe estar precedida por: otra actividad, una decisión, una barra de sincronización o un estado inicial.

- ✓ En el diagrama debe quedar claro el orden en que ocurren las actividades y además debe quedar claro cuando el orden de las actividades no es significativo (pueden ejecutarse en paralelo) utilizando para ello, barras de sincronización.
- ✓ Todas las actividades que están descritas dentro de un Diagrama de Actividad tienen que ser posibles.
- ✓ El flujo de actividades del diagrama debe ser claro y fácil de comprender.

- Diagrama de Clases del Modelo Objeto de Negocio:

  - ✓ Sólo debe incluir las entidades de negocio, la relación que existe entre ellas y los trabajadores de negocio que interactúan con dichas entidades.

2. Revisión de los Requisitos Software: Su objetivo es asegurar la adecuación, factibilidad técnica y completitud de los requisitos incluidos en la Especificación de Requisitos Software, esta revisión es conjunta entre el cliente y los desarrolladores, el responsable de esta tarea es el analista líder (Yaniet Piñeiro). Al finalizar la revisión deben estar completamente firmado todos los requisitos por el cliente y desarrolladores, y puestos en el repositorio del proyecto, en le documento de levantamiento de requisitos.
3. Revisión del análisis: El modelo de análisis brinda una vista interna del sistema estructurada por clases donde se detallan las clases interfaz, controladora y entidad de cada caso de uso lo que permite una mejor visión para la transición hacia el diseño. Los principales aspectos que se revisan son los relacionados con el modelo de casos de uso y la descripción de la arquitectura para lo cual se siguen las siguientes pautas:

   - Todas las clases del diseño están definidas correctamente en correspondencia con lo descrito en la especificación de requisitos y en el modelo de casos de uso del sistema.
   - Están correctamente descritos los escenarios en el caso de uso.
   - Están descritas todas las clases del análisis.
   - Se encuentran los diagramas de caso de uso del análisis.
   - Están definidos los diagramas de interacción para cada caso de uso.

   Descripción de la arquitectura:

- Todos los mecanismos del análisis se han identificado y se han descrito los subsistemas.
- Las dependencias entre los subsistemas y paquetes corresponden a las relaciones de la dependencia entre las clases contenidas.

4. Auditoría del proceso #1: La primera auditoría del proceso podemos clasificarla como de rutina y se lleva a cabo después de la culminación de la fase de análisis con el propósito de tomar muestras del trabajo realizado hasta el momento.
5. Revisión del Diseño Preliminar: Su objetivo es asegurar la adecuación del diseño preliminar tal y como aparece en una versión preliminar de la Descripción de Diseño del Software, antes de comenzar con el diseño detallado, esta revisión debe ejecutarse entre el diseñador y un revisor designado, el responsable da la tarea es el revisor. Los resultados de la revisión deben aparecer en un documento en el repositorio del proyecto para que el diseñador puede auxiliarse para el documento de diseño final.
6. Revisión del Diseño Final: Su objetivo es asegurar la adecuación del diseño detallado tal y como aparece en la versión final de la Descripción de Diseño del Software, antes de comenzar con la codificación, esta tarea deben igualmente cumplirla el diseñador conjuntamente con el revisor, y deben obtener como resultado el documento de Descripción de Diseño del Software.
7. Auditoría interna del proceso #2: el objetivo de esta auditoría es la verificación del acoplamiento de los elementos del diseño con la especificación de requisitos así como la descripción de la arquitectura que al finalizar la etapa de diseño.
8. Auditoría Funcional: Esta auditoría se lleva a cabo antes de la entrega del software para comprobar que se han satisfecho todos los requisitos especificados en la ERS, el responsable de esta auditoría es analista jefe, conjuntamente con varios revisores definidos para realizarle las pruebas funcionales al proyecto, el documento que se obtiene es un resumen de todas las pruebas realizadas.
9. Auditoría Física: Esta auditoría se lleva a cabo para comprobar que el software y su documentación son consistentes internamente y están listos para su entrega. Para ello se compara el código con su documentación de apoyo, se obtiene un documento que certifique detalladamente la calidad de del código contra la documentación que dio origen.

10. Auditorias del Proceso: Son auditorías en las que se examinan muestras de los diferentes productos del desarrollo para comprobar la consistencia del producto según evoluciona a través del proceso de desarrollo. Con ello se obtienen medidas de lo bien que funciona el proceso, esta tarea es realizada por el planificador del proyecto el cual lleva a cabo un control utilizando las herramientas definidas par dicha tarea.
11. Revisiones de gestión: Estas revisiones se llevan a cabo periódicamente para valorar la ejecución de este Plan, esta revisión es controlada por el líder del proyecto, conjuntamente con el planificador par evaluar la ejecución del proyecto, esta revisión obtiene como documentación, un documento Word las incidencias de cumplimientos e incumplimientos, además de la revisión del documento Gantt.
12. Revisión de los documentos de usuario: Esta revisión tiene como objetivo controlar la calidad de los documentos que se le entregaran al usuario como parte de la entrega del producto final, esta revisión es realizada por varias personas: analista, diseñador, arquitecto, para chequear que la documentación adjunta al proyecto contenga todas las explicaciones útiles para los usuarios.
13. Certificación final del producto: esta evaluación se realiza por parte del personal designado para realizarle las pruebas al producto final antes de ser liberado para despliegue, esta debe ser aprobada por el líder de proyecto.
14. Certificación por parte del Cliente (Pruebas de aceptación): estas pruebas la realiza un personal de despliegue conjuntamente con el cliente para obtener el criterio de posibles errores en el producto.
15. Evaluación de la satisfacción del cliente: esta es el aval que realiza el cliente luego de haber realizado la prueba anterior lo cual se almacena para tener constancia de ala aceptación del producto o modulo según corresponda.

2.5.6. Herramientas, Técnicas y Metodologías

Las herramientas utilizadas son:

1. Microsoft Project, el Gantt para poder llevar un control de las actividades y asegurar la calidad del proceso calidad.
2. NUnit que se utiliza para el control de pruebas de unidad en la programación.
3. Se utilizan algunas listas de comprobación para la evaluación de algunos artefactos.

Algunas de las técnicas que pueden ayudar a la evaluación o mejora de la calidad son:

1. Los estándares de codificación, para poder realizar una buena codificación y que existan pocos errores que atenten contra la calidad a la hora de la integración de los módulos.
2. Las inspecciones a las diferentes áreas de trabajo, se harán de forma controlada par que no se entorpezca con el desempeño y avance del proyecto.

Las metodologías de Garantía de Calidad serán conjuntos integrados de técnicas, de entre las anteriores.

Además es importante especificar que la aparición de problemas en las revisiones, auditorias y controles que se tienen previstas dentro del plan de calidad se le aplican, al igual que a lo otros problemas aparecidos en el desarrollo del proyecto, plan de resolución de problemas ya definido.

2.5.7. Gestión de Configuración

La forma en que se llevará a cabo la gestión de la configuración para el control de la calidad es una tarea importante:

- Los documentos salidos de las revisiones, auditorías y pruebas previstas en el plan de calidad, cuenta con una carpeta dentro del repositorio, que como bien se explica se utiliza el *SVN*.
- Las carpetas deben documentarse con fecha y descripción de la actividad.
- Los documentos deben definir artefacto de entrada y artefacto de salida.
- El control de la que todas las actividades sean documentadas correctamente y mantenga una traza en el repositorio es un responsable asignado a cumplir estas tareas, en este caso la compañera Mileisis Placencia.

2.6 Plan de aceptación del producto

2.6.1 Tareas de aceptación del producto

2.6.1.1 Criterio para la aceptación del producto

1. Levantamiento de los Requerimientos chequeados por el cliente y con su visto de conformidad con ellos.
2. Betas cumplan con la función definidas para el periodo:

   Beta v 1.0 funciones de descarga de las bases de datos 1 y 2 de patentes.

   Beta v 2.0 funciones de descarga de la base de datos 3 de patentes y implementación Base de Dato local.
3. Entrega de producto final con las funcionalidades especificadas en los requerimientos.

2.6.1.2 Revisión de la Configuración Física

Los artefactos definidos como de interés del cliente son:

1. Gestión de los requisitos funcionales y no funcionales del proyecto.
2. Los betas elaborados en cada iteración de la fase de implementación para la evaluación del comportamiento.
3. Producto terminado.
4. Documentos de transición de tecnología.

2.6.1.3 Revisión de la Configuración funcional

1. Gestión de requisitos: el método que se utilizará para la evaluación es las reuniones reiteradas

con los clientes hasta lograr la aceptación de todo el proceso con la firma de los usuarios del documento de levantamiento de requisitos.

2. Proceso de revisión y prueba de los betas elaborados, primero con todo el proceso de pruebas internas echas por los miembros del proyecto y las pruebas de funcionalidad echa por los usuarios.
3. Revisión de todos los documentos de transición tecnológica, los cuales deben tener un nivel muy alto de explicación para que sirvan de material auxiliar a la hora del trabajo.

2.6.2 Requerimientos de recursos

2.6.2.1 Requerimientos de Hardware

1. Es necesario que exista una red local que permita la comunicación entre las máquinas donde estará el Sistema y el servidor de Base de Datos, además de conexión a Internet.
2. Es Recomendado para ello una PC PENTIUM III o superior con 300 MHz como mínimo, con 128 MB de RAM, y mínimo 20 GB de disco duro.
3. Por razones de seguridad el servidor de Base de datos será instalado en otra máquina, cuyas características de hardware, son superiores a las PC donde se ejecutará el sistema.
4. La comunicación entre las máquinas donde estará el sistema y el servidor de Bases de Datos será a través del protocolo de TCP/IP.

2.6.2.2 Requerimientos de Software

1. Los requerimientos mínimos de software necesarios son una computadora personal con plataforma del sistema operativo Windows XP o superior.
2. En la PC donde se encuentre el Servidor de Bases de Datos debe tener instalado el Microsoft SqlServer 2000.
3. Las PCs donde se ejecute el sistema debe tener instalado el FrameWork. NET 2.0.

2.7 Valoración de los resultados del desempeño del Líder

2.7.1 Evaluación del Plan de Iteraciones

Durante el desarrollo del plan se pudo valorar su acierto, elaborado en la fase de inicio del proyecto permitió a los miembros del equipo constatar si su desempeño se desvió de los cálculos previstos. Este permite al equipo seguir una línea definida. En la valoración del plan desarrollado se puede verificar que gracias a la efectiva gestión de riesgos, se definió agregarle algunos días más a la iteración de la fase de inicio para obtener mayor tiempo para la planificación; como se muestra en la tabla 20.

Fase	Nro. Iteraciones	Duración
Fase de Inicio	1	3 semanas
Fase de Elaboración	2	12 semanas
Fase de Construcción	3	7.5 semanas
Fase de Transición	1	3 semanas

Tabla 20: Plan real por Fases de Predictor

Como se puede observar en la tabla 20 la cantidad de semanas previstas para la fase de elaboración en sus dos iteraciones aumentaron también, producto a que en ese periodo el programador principal y analista del proyecto tuvo que salir a cumplir tareas por parte de la UCI fuera del país, por lo tanto produjo que se atrasara el tiempo, previendo cualquier fallo en el análisis y diseño, se extendió a 12 semanas toda la fase, la cual estaba programada para 8 semanas, pero es importante que se conozca que la extensión de la segunda iteración proporcionó un vista más clara de lo que se quería implementar por lo tanto se pudo compensar el tiempo perdido en esta fase, con una disminución notable en las dos iteraciones de la fase de construcción donde de 10 semanas previstas para realizar el trabajo se generaron las iteraciones propuestas en 7.5 semanas. Estas 10 semanas enunciadas anteriormente no fueron previstas en el plan original, sino producto al problema mencionado, a lo que los programadores

respondieron de manera eficiente. Se debe aclarar que la inserción de la prueba de nivel de Ingeniería de Software realizada por 5to año, al cual pertenece la totalidad del proyecto, retraso el desarrollo de esta fase también, caso conciliado y comprendido por el cliente no teniendo repercusión alguna en lo acordado.

2.7.2 Evaluación del Plan de gestión de riesgos

El plan de mitigación y gestión de riesgos realizado proporcionó al desarrollo del producto una solides a la hora de enfrentar los problemas identificados como riesgos y establecer las respuestas necesarias, llevando un control de cuales eran los riesgos que con mayor potencial podían afectar durante el ciclo de vida. El plan permitió también poder controlar, eliminar e incrementar riesgos a medida que iba avanzando el proyecto. En particular nuestro plan tuvo sus aciertos y desviaciones:

Aciertos

Se aumento el tiempo de planificación en la fase de inicio para mitigar el riesgo 2 identificado en la Tabla 15: Lista Maestra de Riesgos.

Desviaciones

El riesgo 8 identificado en la Tabla 6.Lista Maestra de Riesgos, fue controlado bastante bien, aunque se produjo un desviación, que aun estando prevista se escapo del control: el principal programador del proyecto fue seleccionado para cumplir tareas fuera del territorio nacional y provocando un atraso en la fase de elaboración, la desviación fue provocada por razones administrativas ajenas al proyecto y relacionada con la disposición de los recursos humanos de la facultad.

2.7.3 Evaluación del Plan de solución de problemas

Como bien se explicó en el Capitulo I existe un método que consta de 5 pasos para poder elaborar planes de solución a diferentes problemas que se presentan a lo largo del ciclo de vida de desarrollo.

Este método se le aplicó a problemas de diferentes índoles. Además a los que se le aplico este método, son aquellos que afectarían el desempeño del proyecto en gran medida, debido que si se le realiza a todos los problemas retrazaría el desarrollo del software. Un ejemplo de problema tratado a través de este método:

1. Definir la situación.

Atraso del proyecto en la Fase de elaboración.

En el proyecto se produjo en atraso en las iteraciones de la fase de elaboración. Donde se había planificado que esta fase durara 8 semanas y duró 12 semanas en tiempo real.

2. Describir el remedio temporal.

Las soluciones temporales adoptadas fueron:

- ✓ El compañero continuó asesorando el trabajo desde el exterior.
- ✓ Se incorporó otro compañero para ocupar el rol de diseñador de sistema.
- ✓ El otro desarrollador impartió cursos rápidos y específicos sobre la herramienta de programación.

3. Identificar las causas raíz.

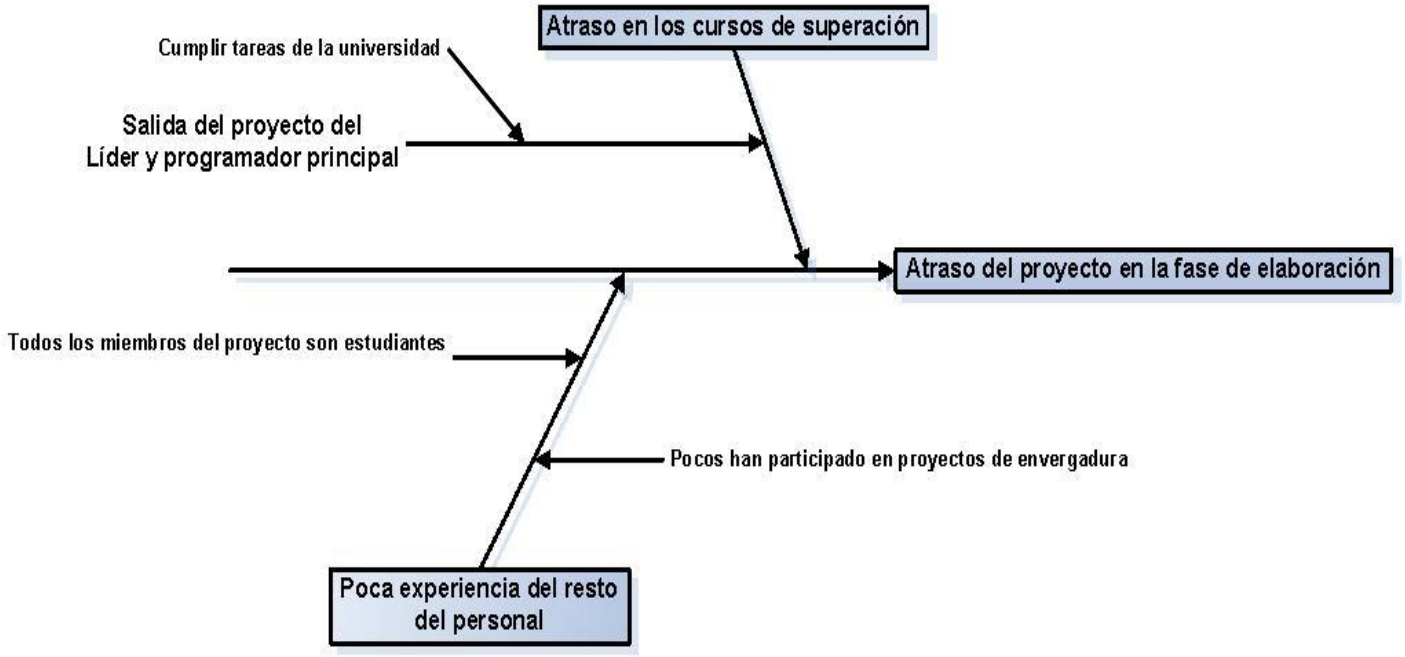

Figura 1: Diagrama Causa-Efecto del problema de la salida del el programador principal

4. Tomar acción correctiva.

Programa de implementación de acciones correctivas

1. Impartir mayor cantidad de cursos:
    1. Análisis y diseño enfoques desde la práctica.
    2. Patrones de diseño.
    3. Programación en C#.
    4. Facilidades de la plataforma .NET.
2. Elaboración de los artefactos y entregables en parejas o equipos.
3. La documentación de las acción correctiva se subirán para el repositorio del proyecto donde estarán las evoluciones de los cursos impartidos, además el jefe de proyecto evaluará el desempeño de los equipos de trabajo, con la ayuda del planificador que mantendrá en el repositorio el Gantt del avance del proyecto.
4. Las acciones especificas se tomarán para verificar que la acción correctiva está operando
    1. El planificador lleva el control del desarrollo de las actividades designadas los equipos que se chaquean semanalmente.

2. Los compañeros que imparten los cursos deben poner la nota de los mismos en el repositorio.

5 Evaluar.

En las reuniones semanales el planificador reporta al jefe del proyecto el estado de las tareas en realización.

2.8 Valoración de estado del proyecto

Para controlar el estado del proyecto se genera un documento que se presenta periódicamente en el cual intervienen el Líder del Proyecto y el Planificador, este último es el encargado de llevar el control y seguimiento de las tareas, manteniendo al tanto al primero del estado del desarrollo del producto, en el ultimo control realizado:

2.8.1 Recursos

2.8.1.1 Planilla de personal

La distribución del personal que se realizó para el proyecto, funciono de manera eficiente, logrando que:

- 1 Líder del proyecto.
- 2 Programadores.
- 2 Analistas de Sistemas.
- 2 Arquitectos.
- 1 Planificador.
- 1 Diseñador de Sistemas.

Pudieran desarrollar bien su rol y desplegarse hacia otros necesarios en el desarrollo del producto, como tareas de calidad, y de investigación. Además se aseguro con esta distribución que los estudiantes ganaran en mayor madurez sobre el trabajo en equipo.

2.8.2 Estado de los hitos mayores

A continuación se presenta el estado por fecha de los hitos mayores.

Hitos	Fecha	Estado
Modelo de Casos de uso del negocio	25/05/07	Terminado
Modelo de objetos del negocio	25/05/07	Terminado
Lista de riesgos	25/05/07	Terminado
Documento Visión	25/05/07	Terminado
Modelo de casos de uso	25/05/07	Terminado
Modelo de Análisis	25/05/07	Terminado
Modelo de Diseño	25/05/07	Terminado
Modelo de Datos	25/05/07	Terminado
Modelo de Implementación	25/05/07	Terminado
Modelo de despliegue	25/05/07	Terminado
Despliegue del Producto		No terminado

Tabla 21: Estado de los Hitos más grandes.

Esta tabla demuestra el control y aseguramiento de cada uno de los hitos mayores representados fundamentalmente por la entrega de los artefactos más importante de cada iteración o fase del desarrollo y en el estado que se encuentran actualmente.

2.8.3 Alcance del producto o Proyecto total

El estado actual del proyecto es bueno. El desempeño de los desarrolladores hasta el momento ha sido satisfactorio. Por otra parte, betas del producto están siendo utilizados por los clientes, los cuales han expresado su satisfacción, aunque no se ha terminado completamente funcionalidades del producto final, en una carta de aceptación expuesta en el Anexo B. Además se ha presentado en varios eventos como son el evento de las BTJ, obteniendo mención, el diploma esta expuesto en el mismo anexo antes

mencionado. Predictor también se presentó, junto a otros proyectos, como representación de la UCI en la Feria Internacional de Informática 2007.

2.9 Conclusiones

En el Capitulo anterior se demuestra que la utilización y adaptación de RUP como guía del proceso del desarrollo realizado, fue satisfactoria. Además la utilización de los planes, aseguraron al Líder y al resto del equipo de Predictor, de una herramienta muy efectiva para controlar el ciclo de vida del producto. La utilización de otras herramientas para el seguimiento de las tareas como: *SVN* y el diagrama de Gantt del Microsoft Project, fueron igualmente efectivas.

La evaluación de los planes para saber donde ocurrieron desviaciones y la capacidad del líder como máximo responsable de la organización de poder contrarrestar cualquier tipo de problema hasta alcanzar el desarrollo completo del producto.

Conclusión

Al concluir esta investigación se puede afirmar que se han cumplido todos los objetivos planteados para su desarrollo.

Se realizó un estudio de las competencias y actividades realizadas por un líder dentro de un proyecto informático, además de realizarse un estudio de las metodologías para regir el proceso de desarrollo de software, de las cuales se opto por RUP para desarrollar este producto, así como un estudio de las mejores formas de realización de planes que contribuyen a la organización y control del proceso de software.

Se aplicaron y desarrollaron los planes definidos y adaptados al entorno del proyecto, además de permitir el desempeño de las competencias y actividades del líder dentro del proyecto, siendo capaz de dirigir y controlar al grupo de compañeros para obtener un buen producto.

Se aplicaron métodos y herramientas que permitan el control y seguimientos de los planes y de las tareas planteadas por ellos, dentro de las cuales podemos mencionar el *SVN* para el control del repositorio, y el modelo de Gantt incluido dentro del Microsoft Project.

También se evaluaron los planes desarrollados para medir el avance del producto y la efectividad de los planes utilizados, además de evaluar el estado actual del proyecto donde se muestra el alto nivel de aceptación del producto por parte del cliente, con todos los betas entregados y funcionales.

Por lo que se concluye que el rol de líder, es indispensable en el trabajo de un equipo. Pero este será realmente efectivo, siempre y cuando se sigan los principales criterios para la selección de los recursos humanos y de las técnicas fundamentales para mantener la buena relación entre los miembros del equipo, además de la utilización de herramientas, técnicas y metodologías para controlar todo el proceso de desarrollo del software.

Recomendaciones

Tras haber cumplido con los objetivos propuestos para el desarrollo del trabajo de Líder del Proyecto Delfos, se recomienda:

- La creación de métricas propias para el proceso de desarrollo en la UCI en aras de uniformar el proceso de medición.
- Establecer en la Universidad cursos de formación, preparación y superación para líderes de proyectos.

Bibliografía Citada

1. HEREDIA, R. Dirección Integrada a Proyecto. p.
2. HUMPHREY, W. S. Introduction to the Team Software Process. California, Addison Wesley., 1999. p.
3. JODOCHA. ¿Qué es un líder de proyectos informáticos?, 2005. [Disponible en: http://www.programacion.com/blogs/49_jodocha
4. PRESSMAN, R. S. Ingeniería de Software. Un enfoque práctico., 2005. p.

Bibliografía Consultada

1. Líder del proyecto Debian. 17 de ene de 2007. [3 febrero 2007]. Disponible en: http://www.debian.org/devel/leader.es.html
2. Los Estandares de Calidad ISO para Desarrollo de Software Disponible en: http://tecnomaestros.awardspace.com/estandares_iso.php
3. ANTONIO, A. D. GESTIÓN, CONTROL Y GARANTÍA DE LA CALIDAD DEL SOFTWARE.
4. Un vistazo al P-CMM (People Capability Maturity Model). 13 noviembre 2006. [17 enero 2007]. Disponible en: http://www.navegapolis.net/content/view/485/59/
5. BURCH, S. Sociedad de la información/ Sociedad del conocimiento, 2006. [26 noviembre 2006]. Disponible en: http://www.vecam.org/article.php3?id_article=518
6. COMUNICACIONES, D. D. T. D. I. Y. Líder de Proyectos Informáticos, 7 abril 2006. [18 diciembre 2006]. Disponible en: http://www.uned.ac.cr/redti/cuarta/art8.pdf
7. GIRALDO, O. P. Métricas, Estimación y Planificación en Proyectos de Software. Disponible en: www.willydev.net/Descargas/WillyDEV_PlaneaSoftware.Pdf
8. GIUDICE, I. E. L. INFORMÁTICA Y LIDERAZGO EN LAS ORGANIZACIONES, 18 diciembre 2006]. Disponible en: http://www.gestiopolis.com/recursos/documentos/fulldocs/ger1/infolidor.htm
9. ---. Sobre el Líder, 4 Mayo 2006. [8 enero 2007]. Disponible en: http://www.wikilearning.com/sobre_el_lider-wkccp-12390-4.htm
10. GRACIA, J. CMM - CMMI Nivel 2, 26 de Noviembre de 2005. Disponible en: http://www.ingenierosoftware.com/calidad/cmm-cmmi-nivel-2.php
11. HERNÁNDEZ, R. G. La información de marcas como indicador de innovación tecnológica, 2006.
12. HUMPHREY, W. S. Introduction to the Team Software Process. California, Addison Wesley., 1999. p.

13. IEEE COMPUTER SOCIETY, S. E. T. C. IEEE Standard for Software Quality Assurance Plans. 1989. p.
14. JODOCHA. ¿Qué es un líder de proyectos informáticos?, 11 agosto 2005. [25 noviembre 2006]. Disponible en: http://www.programacion.com/blogs/49_jodocha
15. LAMAS, D. G. Motivacion de equipos de trabajo, 26 diciembre 2006]. Disponible en: http://www.emagister.com/motivacion-equipos-trabajo-cursos-317293.htm
16. LOPEZ, L. F. A. Como diseñar sistemas de estimulación para las organizaciones. 18 diciembre 2006]. Disponible en: http://www.emagister.com/como-disenar-sistemas-estimulacion-para-organizaciones-cursos-317464.htm
17. ---. Los problemas de la estimulación al trabajo y el diseño de sistemas de estimulación en las organizaciones 11 diciembre 2003. [25 noviembre 2006]. Disponible en: http://tutorial.emagister.com/frame.cfm?id_user=98963283799599144096238667928563&id_centro=35596070040551554868665657484551&id_curso=331200700405526968675767496745 49&url_frame=http://www.ilustrados.com/publicaciones/EpZEkpyyulpGMgqvPv.php
18. ---. PROBLEMÁTICAS ACTUALES DE LA ESTIMULACIÓN AL TRABAJO; REFLEXIONES Y ACCIONES, 25 enero 2007]. Disponible en: http://usuarios.lycos.es/direccion/manuales/problemas%20estimulacion.doc
19. LOVELLE, J. M. C. Calidad del Software, 1999. [Disponible en: http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Calidad_software.PDF
20. MARTÍNEZ, M. F. J. La Función de Calidad en los Proyectos de Desarrollo de Software. Disponible en: www.ati.es/gt/calidad-software/SIMO00/SIMO2000-Resp-Calidad.ppt
21. NAVARRO, M. T. CMM Y RUP una perspectiva común., noviembre 2003. Disponible en: http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo_CMM_y_RUP_04.pdf
22. PEDRO Y. PIÑERO PÉREZ, Y. P. P., MARIETA PEÑA ABREU Desarrollando un proyecto de software, una experiencia práctica., 2006.
23. PRESSMAN, R. S. Ingeniería de Software. Un enfoque práctico., 2005. p.
24. VARGAS, E. PLAN DE MÉTRICAS EN OCHO PASOS. Disponible en: www.eulatic.org/docs/MetricasCAPR.pdf
25. ZALLIO, L. Virtudes de un líder de grupo, 20 julio 2005. [20 enero 2007]. Disponible en: http://ele-zeta.com.ar/2005/07/20/administrar-proyectos-virtudes-de-un-lider-de-grupo/

Glosario

LDC: líneas de código producidas.

Sinergia: es la integración de elementos que da como resultado algo más grande que la simple suma de éstos, es decir, cuando dos o más elementos se unen sinérgicamente crean un resultado que aprovecha y maximiza las cualidades de cada uno de los elementos.

RUP: Proceso Unificado de Desarrollo. Es un proceso de desarrollo de software que una varia prácticas y metodologías.

Microsoft Excel: es un programa del paquete de Microsoft Office, distribuido por la empresa del mismo nombre. Es la hoja de cálculo más utilizada en estos momentos.

Microsoft Project: es un programa que pertenece al paquete de Microsoft Office, es la solución que brinda la empresa Microsoft para la administración de proyectos.

Diagrama de Gantt: es una herramienta que consiste en la representación grafica en dos ejes, el vertical para las tareas del proyecto y el horizontal para la línea de tiempo.

Subversión (SVN): es una aplicación para el control de versiones que nos permite gestionar los cambios y versiones que se realizan en el desarrollo de un proyecto o actividad.

SPI: Software de Interés Público, es una organización que no persigue beneficios que se fundó para ayudar a proyectos de desarrollo de software para la comunidad. Algunos proyectos se soportan a través de donaciones monetarias o de hardware que se han hecho a SPI.

USPTO: United States Patent and Trademark Office: Oficina de Patentes y Marcas de EUA.

Esp@cenet: es un servicio gratuito en línea para buscar patentes y solicitudes. Fue desarrollado por la EPO junto a los estados miembros de la Organización Europea de Patentes.