UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
DIRECCIÓN GENERAL DE LOS ESTUDIOS DE POSTGRADO
ÁREA DE CIENCIAS ADMINISTRATIVAS Y DE GESTIÓN
POSTGRADO EN GERENCIA DE PROYECTOS

TRABAJO ESPECIAL DE GRADO

MODELO DE EVALUACIÓN DE METODOLOGÍAS PARA EL DESARROLLO DE SOFTWARE

Presentado por:
MÉNDEZ NAVA, ELVIA MARGARITA
Para optar al título de:
Especialista en Gerencia de Proyectos

Asesor
GARRIDO RAMÓN

Caracas, julio de 2006

Reconocimiento

A Dios Todopoderoso, por que siempre esta conmigo.

A mi familia por ser lo que me da el valor y la entereza de seguir adelante y cumplir cada una de mis metas.

A mis padres, por su cariño y apoyo incondicional en los momentos más difíciles. Gracias por haber hecho de mí, lo que soy ahora.

A la Sra. Yaritza y Sr. Atilio por el gran apoyo que me han brindado durante la realización y culminación de mi especialización

A mis hermanos, por hacerme sentir la mejor de las hermanas.

A mis padrinos, por ser como unos segundos padres para mí.

Y muy especialmente a mi tutor Ing. Ramón Garrido por que sin el no hubiera realizado esta meta en el tiempo que me lo había previsto. ¡ Gracias!.

Elvia M. Méndez N.

ÍNDICE

Sinopsis....1

CAPÍTULO I. PROPUESTA DEL PROYECTO....2

1.1 Planteamiento y delimitación de la problemática....2
1.2 Objetivos del Proyecto....4
1.2.1 Objetivo General....4
1.2.2 Objetivos Específicos....4
1.3 Justificación del Proyecto....4
1.4 Marco metodológico....5
1.5 Resultado esperados....6

CAPITULO II. MARCO TEÓRICO....7

2.1 Dirección de Proyectos....7
2.1.1 Definición....7
2.1.2 Áreas de conocimiento en la Dirección de proyectos....7
2.1.3 Procesos de la Dirección de proyectos....8
2.2 Dirección de proyectos de software....10
2.2.1 Definición....10
2.2.2 Marco de trabajo para el desarrollo de software....10
2.2.3 Modelos de procesos utilizados en el desarrollo de software....11
2.2.2.1 Modelos convencionales o prescriptitos....12
2.2.2.2 Modelos de desarrollo ágil....19

CAPITULO III. MARCO ORGANIZACIONAL....30

3.1 La empresa....30
3.1.1 Misión....30
3.1.2 Organigrama....31
3.1.3 Políticas de la empresa....31

CAPITULO IV. MATRIZ DE EVALUACIÓN DE METODOLOGÍAS....32

4.1 Identificación de variables en la evaluación de metodologías a ser utilizadas en el desarrollo de software....32
4.2 Diseño de la matriz de evaluación de metodologías....36

CAPITULO V. ANÁLISIS DE RESULTADOS....40

5.1 Resultados obtenidos en las encuestas....40
5.2 Resultados obtenidos en la matriz de evaluación de metodologías....41

CAPITULO VI. CONCLUSIONES Y RECOMENDACIONES....42

6.1 Conclusiones....42
6.2 Recomendaciones....43

BIBLIOGRAFÍA....44

TABLA DE GRÁFICOS

Fig 1. Modelo de cascada 13
Fig 2. Modelo incremental 14
Fig 3. Modelo de desarrollo rápido de aplicaciones (DRA) 15
Fig 4. Modelo de prototipos 16
Fig 5. Modelo en espiral 17
Fig 6. Modelo de desarrollo concurrente 19
Fig 7. Modelo de Programación extrema 21
Fig 8. Modelo de Desarrollo adaptativo de software 22
Fig 9. Modelo de desarrollo de sistemas dinámicos 24
Fig 10. Flujo de proceso de la Melé 25
Fig 11. Desarrollo Conducido por características 26
Fig 13. Áreas de competencia de Synergy Consultores, C.A. 30
Fig 14. Organigrama Synergy Consultores, C.A. 31
Fig 15. Tabla de cálculo de las ponderaciones de las variables 36
Fig 16. Matriz de perfil competitivo 37
Fig 17. Matriz de evaluación de metodologías (convencionales ó prescriptitas) 38
Fig 18. Matriz de evaluación de metodologías (proceso de desarrollo ágil) 39

Universidad Católica Andrés Bello
Titulo: Modelo de Evaluación de Metodologías para el desarrollo de software
Asesor: Ing. Ramón Garrido
Autor: Elvia Margarita Méndez Nava C.I.: 13.868.963 Exp. 84.205
Fecha: 18 de junio de 2006

Resumen Proyecto de Trabajo de Grado de Especialización

Toda empresa dedicada al desarrollo de software en miras de aprovechar las oportunidades que se le presentan, deben definir desde un principio un plan de gestión que permita el éxito de producto o servicio a ofrecer, con la adecuada selección de la metodología a seguir de acuerdo a la naturaleza del proyecto que realizará. Es por ello que se desea diseñar una herramienta que permita facilitar y ayudar a la dirección de proyectos, en la selección de metodologías de desarrollo de software. Con la realización de este trabajo se busca el aseguramiento de la calidad en el desarrollo del proyecto, la disminución o mitigación de los riesgos y la facilidad para definir los entregables de dicho proyecto. El objetivo general del trabajo es el de Diseñar un modelo de evaluación de metodologías para el desarrollo de software, para conseguir este objetivo se plantean tres objetivos específicos: Identificar variables para la evaluación de metodologías aplicadas en el desarrollo de software, diseñar el modelo de evaluación de metodologías de desarrollo de software y desarrollar la matriz de evaluación de metodologías. El tipo de metodología a utilizar en la investigación es la de Investigación y Desarrollo. Se espera poder proveer a futuros líderes o gerentes de proyectos, de una herramienta de fácil comprensión y utilización, que permitirá definir desde un principio cuales son los pasos a seguir para llevar a cabo un proyecto de desarrollo de software. De igual manera este trabajo recopila las diferentes metodologías utilizadas para el desarrollo de software, convencionales o de desarrollo ágil que servirá como guía para conocer las características más importantes de cada una de ellas. Una vez finalizado el Trabajo especial de grado se obtendrá una herramienta o modelo a seguir para la selección de una metodología que puede ser adecuada a las características de cualquier proyecto de desarrollo de software.

Palabras clave: Metodologías aplicadas en el Desarrollo de Software

Sinopsis

Este trabajo especial de grado tiene como finalidad el diseñar una matriz de evaluación de metodologías de desarrollo de software

El Trabajo se encuentra presentado por capítulos, siguiendo las secuencias que se presenta a continuación:

Capitulo I. Propuesta del Proyecto. Se describe la problemática presentada por las industrias del software , los objetivo que se pretenden alcanzar con el estudio realizado, además de conocer los alcances y limitaciones a los que se encuentra sujetos este proyecto, basándonos en la metodología seleccionada.

Capitulo II. Marco Teórico. Posee todo el basamento teórico necesario para comprender los conceptos utilizados, así como la fundamentación de las técnicas y herramientas.

Capitulo III. Marco Organizacional. En este capitulo se hace una pequeña presentación de la empresa donde se llevó a cabo este estudio. En la misma se muestra la actividad de la empresa, su misión, estructura organizativa y políticas

Capitulo IV. Matriz de evaluación de metodologías. Se diseño y desarrollo la matriz de evaluación de metodologías, utilizando variables determinantes a la hora de seleccionar la más adecuada para el desarrollo de un software

Capitulo V. Análisis de Resultados. En este capitulo se realiza un breve análisis de los resultados obtenidos en el capitulo IV.

Capitulo VI. Conclusiones y Recomendaciones. Se muestra los resultados obtenidos del estudio y las recomendaciones para la empresa y los próximos estudios que tengan que ver con esta área.

CAPÍTULO I. PROPUESTA DEL PROYECTO

1.1 Planteamiento y delimitación de la problemática

Un estudio realizado a la Industria del Software en Venezuela por la Empresa Data análisis en el año 2003, refleja los principales factores que contribuyen al éxito de las empresas de este sector, entre las que se destacan la calidad del servicio y el producto entregado; de igual manera hace referencia a las mayores problemáticas que se ven expuestas, entre las que se mencionan: la falta de protección contra la piratería (46,3%), lo reducido del mercado interno (45,0%), la ausencia o carencia de fuentes de financiamiento (43,8%), la falta de una política de estado para la industria del software (31,3%) y el desconocimiento en el ámbito nacional e internacional de los productos y servicios desarrollados en el país (30%).

En el se muestra también las cinco sugerencias mas realizadas por las empresas, que según su juicio y experiencia pudieran contribuir con el crecimiento del sector: el financiamiento del desarrollo de proyectos (45%), incentivar la innovación tecnológica de investigadores y empresas (36,3%), generación de demanda (33,8%), dar créditos oportunos e intereses adecuados para el desarrollo de productos (31,3%) y proveer el financiamiento de riesgo compartido entre el estado y las empresas (21,3%).

En la actualidad el Ejecutivo Nacional viene creando una serie de políticas de estado que permite incentivar la innovación tecnológica de investigadores y empresas que contribuyen al crecimiento de la industria del software en Venezuela: dentro de estas se encuentra la publicación en gaceta del Decreto Nº 3390 de fecha 23 de diciembre de 2004, “mediante el cual se dispone que la administración Pública Nacional empleará prioritariamente software libre desarrollado con estándares abiertos, en sus sistemas, proyectos y servicios informáticos”. Esto permite a las empresas contar con grandes oportunidades de crecimiento, ya que los sistemas serán realizados a la medida del cliente permitiendo a las pequeñas y medianas industrias tener una ventaja competitiva con las grandes empresas que ofrecen herramientas ya desarrolladas y que no poseen la flexibilidad de las aplicaciones desarrolladas en software libre.

Toda empresa dedicada al desarrollo de software en miras de aprovechar las oportunidades que se le presentan, debe revisar sus procesos de manera que deseche todo aquello que no agregue valor al negocio, buscando la mayor eficiencia en el desarrollo de los proyectos a realizar.

El éxito de las empresas dependerá en gran medida de la definición y administración adecuada desde el inicio al fin del desarrollo de software, para lo cual es importante poseer un amplio conocimiento en la gestión de integración, área de conocimiento de la dirección de proyectos, donde se busca la unificación, consolidación y articulación de los distintos procesos, para tener una visión global del desarrollo del mismo

Para lograr una buena gestión de la integración se debe analizar y comprender el alcance del proyecto, conocer en detalle las especificaciones del producto a realizar, crear un plan de gestión y analizar los riesgos que se encuentren latentes, previendo las medidas a tomar, en caso de que se presenten.

Las empresas deben definir desde un principio un plan de gestión que permita el éxito de producto o servicio a ofrecer, con la adecuada selección de la metodología a seguir de acuerdo a la naturaleza del proyecto que realizará.

La selección de la metodología a utilizar en el desarrollo de un software en ocasiones se hace cuesta arriba, ya que se debe poseer mucha experiencia en este ámbito para que sin necesidad de un análisis exhausto, se pueda dar con aquella que es la más adecuada para su aplicación. Las metodologías utilizadas en el desarrollo de software se encuentran bajo un marco general en el cual se encuentran presentes la comunicación, planeación, modelado, construcción y desarrollo, sin embargo, no todas ellas poseen la misma importancia para cada metodología ni tampoco un mismo flujo de trabajo.

Motivado a esta gran variedad de metodologías y a su particularidades, se desea diseñar una herramienta que permita facilitar y ayudar a la dirección de proyectos, en la selección de metodologías de desarrollo de software; buscando asegurar que la definición del plan de gestión sea acorde a las características del proyecto a llevar a cabo.

1.2 Objetivos del Proyecto

1.2.1 Objetivo General

Diseñar un modelo de evaluación de metodologías para el desarrollo de software

1.2.2 Objetivos Específicos

1.2.2.1. Identificar variables para la evaluación de metodologías aplicadas en el desarrollo de software.

1.2.2.2. Diseñar el modelo de evaluación de metodologías de desarrollo de software.

1.2.2.3. Desarrollar la matriz de evaluación de metodologías.

1.3 Justificación del Proyecto

En búsqueda de poseer estabilidad, control y organización en el proyecto, que de un momento a otro se puede volver caótico, se debe definir desde un principio los pasos a seguir para el desarrollo del mismo, donde se especifiquen las actividades, controles y documentaciones realmente necesarias para el equipo de proyecto y el producto a realizar.

Es por ello que se desea diseñar una herramienta que permita la evaluación y selección de la metodología más adecuada para llevar a cabo los distintos proyectos en los que incursionará la empresa.

Lo que se busca con la selección acertada de la metodología a aplicar en el desarrollo de un software es el aseguramiento de la calidad del producto o servicio a ofrecer, la disminución o mitigación de los riesgos y la facilidad para definir los entregables a realizar.

1.4 Marco metodológico

El tipo de investigación utilizada en el desarrollo de este trabajo especial de grado es la de investigación y desarrollo.

A continuación se presentan las etapas que se siguieron para alcanzar los objetivos específicos de este trabajo especial de grado y que contribuyeron a alcanzar el objetivo general del mismo

1. Identificar variables para la evaluación de metodologías aplicadas en el desarrollo de software.

    a. Se consultó bibliografías, artículos de Internet y revistas, para determinar las variables que de alguna manera son pieza fundamental para la selección o definición de una metodología a seguir en un proyecto de desarrollo de software
    b. Se diseño una encuesta dirigida a personas que poseen algún tipo de experiencia con el desarrollo de proyectos de software
    c. Se aplicaron las encuestas a personas con diferentes niveles de responsabilidad dentro del organigrama en proyectos de desarrollo de software
    d. Una vez obtenido los resultados de las encuestas se analizaron y se obtuvo la lista con las variables y sus respectivas ponderaciones

2. Diseñar el modelo de evaluación de metodologías de desarrollo de software.

    a. Se consulto bibliografías, artículos de Internet y revistas para ubicar herramientas que pudieran ser utilizadas en el diseño del modelo de evaluación de metodologías
    b. Se realizo la adecuación de la herramienta ubicada a los requerimientos de este trabajo especial de grado

3. Desarrollar la matriz de evaluación de metodologías.

    a. Se construyó la matriz de evaluación de metodologías, con las variables encontradas, su ponderación y las metodologías utilizadas para el desarrollo de software
    b. Se calificaron cada una de las metodologías para cada variable identificada con anterioridad.
    c. Aplicación de la herramienta
    d. Análisis de resultados obtenidos

1.5 Resultado esperados

Se espera poder proveer a futuros líderes o gerentes de proyectos, de una herramienta de fácil comprensión y utilización, que permitirá definir desde un principio cuales son los pasos a seguir para llevar a cabo un proyecto de desarrollo de software.

De igual manera este trabajo recopila las diferentes metodologías utilizadas para el desarrollo de software, convencionales o de desarrollo ágil que servirá como guía para conocer las características más importantes de cada una de ellas.

CAPITULO II. MARCO TEÓRICO

2.1 Dirección de Proyectos

2.1.1 Definición

En la actualidad podemos encontrar diferentes denominaciones a lo que llamamos la dirección de proyectos: según el Project Managment Institute, la dirección de proyectos es:

“la aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades de un proyecto para satisfacer los requisitos del proyecto. La Dirección de proyectos se logra mediante la aplicación e integración de los procesos de dirección de proyectos de inicio, planificación, ejecución, seguimiento y control, y cierre.”

Dentro de la dirección de proyectos se contempla nueve áreas de conocimientos que se describen a continuación.

2.1.2 Áreas de conocimiento en la Dirección de proyectos

Gestión de Integración del Proyecto: donde se puede tener una visión global del proyecto, ya que en esta área combino o unifico los diferentes procesos de la dirección de proyectos. Es de gran importancia para la gerencia ya que ayuda a dirigir y coordinar las diferentes actividades de manera que se llegue al resultado esperado.

Gestión del Alcance del Proyecto: Defino lo que quiero obtener en el proyecto, tomando en consideración lo que en realidad se necesita para que al final del proyecto se tenga lo deseado.

Gestión del Tiempo del Proyecto: Todo lo referente al tiempo que me tome el desarrollo del proyecto, tomando en consideración cada una de las actividades y los recursos que posea para culminarlo en los tiempos establecidos.

Gestión Costos del Proyecto: Como su palabra lo dice es todo lo referente a los costos involucrados en el proyecto. Es importante señalar que deben ser bien definidos desde un principio para no tener problemas con el presupuesto asignado para dicho proyecto.

Gestión de la Calidad del Proyecto: La calidad del proyecto se obtendrá en la medida que yo satisfaga las necesidades por las cuales se realiza el proyecto. Esto va ligado a las acciones que se deben tomar para asegurar que el resultado del proyecto sea el mismo por el cual se concibió.

Gestión de los Recursos Humanos del Proyecto: Todos los procesos donde involucro la organización y dirección del personal designado en el proyecto. En el defino los roles y funciones que posea cada uno además de la cuantificación del personal requerido para dicho proyecto.

Gestión de las Comunicaciones del Proyecto: Fundamental para obtener lo que se desea al finalizar el proyecto es una buena comunicación. Se requiere que el intercambio de información sea lo mas fluida posible para que no existan dudas de hacia donde se va. Es por ello que toda la información generada sea almacenada y remitida a los destinos que le corresponde.

Gestión de los Riesgos del Proyecto: Visualizo todos los eventos que pueden modificar algún objetivo en el desarrollo del proyecto, esto ayuda a prevenir o tomar acciones para corregirlas de manera que no se vea afectado el resultado del proyecto.

Gestión de las adquisiciones del Proyecto: Donde se coordina o dirige todo lo referente a la obtención de los recursos necesarios para el desarrollo del proyecto, desde la adquisición del material de oficina hasta la contratación de servicios a otra empresa.

2.1.3 Procesos de la Dirección de proyectos

Procesos de Iniciación: En este primer proceso se identifican las necesidades de la empresa y por la cual se requiere el desarrollo de un nuevo proyecto, en este proceso aclaro los objetivos a perseguir para satisfacer esas necesidades y de igual manera defino el alcance,

la duración y lo que se estima utilizar en recursos, esto con el fin de que sea aprobado y se inicie el desarrollo del proyecto.

Proceso de Planificación: Donde se establece las actividades, los tiempos y los recursos, basado en las experiencias e historial de datos que posea la coordinación de proyectos. Cabe destacar que mientras mas certera sea la planificación, el desarrollo del proyecto se llevará con menores contratiempos.

Proceso de Ejecución: Se lleva a cabo el plan trazado. Cumple con los requisitos de proyecto.

Proceso de seguimiento y control: En este proceso se observa los avances que vaya teniendo el desarrollo del proyecto de manera que se tiene un control de aquellos objetivos que se van cumpliendo en los lapsos estipulados en la planificación. Si existe algún objetivo que no se ha cumplido en el tiempo estimado, se puede tomar las acciones para que no afecte el resultado del proyecto.

Proceso de cierre: Se finaliza el proyecto ó una fase del mismo luego de evaluar el cumplimiento de los objetivos trazados. Se finaliza formalmente con la aceptación del mismo y la iniciación de la implementación.

Estas áreas de conocimientos y procesos que se encuentran descritas en la guía del PMBOK, como partes fundamentales de la dirección de proyectos son base para el desarrollo de cualquier proyecto[1].

Dentro de la amplia diversidad de los tipos de proyectos existentes se tienen los proyectos de desarrollo de software que han sido objeto de grandes estudios y que actualmente se encuentra en continuo crecimiento debido al incremento en el uso de las tecnologías y comunicación.

[1] Proyecto: “Un esfuerzo temporal , emprendido para elaborar un producto o servicio”

Es por ello que se cuenta con la ingeniería del software que no es mas que el conjunto de métodos, técnicas y herramientas que controlan el proceso integral del desarrollo de software y suministra las bases para construir software de calidad de forma eficiente en los plazos adecuados.

Dentro de la ingeniería del software se toma en consideración todos y cada uno de los fundamentos utilizados por la guía del PMBOK, pero adecuados a la naturaleza de los proyectos de desarrollo de software. Es así como encontramos que lo que son los procesos para la dirección de proyectos según la guía del PMBOK, para la ingeniería del software y Roger S. Presuman es llamado marco de trabajo para el desarrollo de software.

A continuación se presenta la dirección de proyectos desde el enfoque de la ingeniería del software

2.2 Dirección de proyectos de software

2.2.1 Definición

Según el libro de Roger Presuman, Ingeniería del software un enfoque práctico " la gestión eficaz de proyectos de software se enfoca sobre las cuatro P: personal, producto, proceso y proyecto", que si lo analizamos con detenimiento es también podrid ser una definición de la dirección de proyectos de forma general.

2.2.2 Marco de trabajo para el desarrollo de software

El marco de desarrollo de software establece la base, donde se identifica un número pequeño de actividades que son aplicables a todos los proyectos de esta índole, sin importar su tamaño o complejidad. A continuación se detallan:

Comunicación: se basa en la comunicación y colaboración con los clientes[2], esta actividad permite el levantamiento de los requerimientos y la compresión de las necesidades del negocio.

Planeación: define los objetivos y determina la estrategia y metodología a seguir. Describe las tareas técnicas que deben realizarse, los riesgos que podrían amenazar el correcto curso del proyecto, los productos a ser entregados y el cronograma de trabajo inicial.

Ejecución: se ejecuta el plan de gestión. En el se crean modelos que permitan al desarrollador y al cliente comprender mejor los requerimientos del software.

Revisar. (Revisión de resultados): en ella se verifica los avances del proyecto de acuerdo al plan. Permite identificar variaciones respecto al plan de gestión del proyecto, de tal forma que se tomen las medidas necesarias para cumplir con los objetivos definidos.

Actuar. (Tomar las acciones necesarias): si al revisar los resultados el trabajo no cumple con lo esperado en el plan de gestión, ó no cumple con los resultados esperados, se deben tomar las acciones apropiadas para su tratamiento.

Los detalles del proceso del software serán muy diferentes en cada caso, pero las actividades dentro del marco permanecerán similares y enmarcándose en las descritas anteriormente.

2.2.3 Modelos de procesos utilizados en el desarrollo de software

Los procesos utilizados para el desarrollo de software se pueden dividir en dos grandes vertientes los modelos convencionales (prescriptivos de proceso) y de desarrollo ágil.

Los modelos convencionales, llamados también “prescriptivos”, determina un conjunto de elementos del proceso: actividades del marco de trabajo, acciones de ingeniería del software, tareas, productos del trabajo, aseguramiento de la calidad, y mecanismos de control de

[2] Cliente, cualquier persona que posee algún tipo de interés en el desarrollo del sistema

cambio para cada proyecto; cada modelo prescribe también un flujo de trabajo, donde los elementos del proceso se interrelacionan entre si. Entre algunos de estos modelos podemos nombrar: modelo en cascada, modelo de proceso incremental como el desarrollo rápido de aplicaciones DRA, modelos de procesos evolutivos como el prototipazo y en espiral, modelos especificados de procesos como el desarrollo basado en componentes y el orientado a aspectos.

Los modelos de desarrollo ágil combina una filosofía y un conjunto de directrices de desarrollo. Busca la satisfacción del cliente y la entrega temprana de software incremental; por lo general se realiza con equipos pequeños de proyectos que poseen una alta motivación, utilizan métodos informales; y una simplicidad general del desarrollo. La comunicación entre los desarrolladores y los clientes durante el desarrollo del proyecto es activa y continua. Entre ellos podemos mencionar: programación extrema (PE), desarrollo adaptativo del software (DAS), el modelo de desarrollo de sistemas dinámicos (MDSD), el modelo Melé, el desarrollo conducido por características (DCC) y el proceso unificado de Rational

2.2.2.1 Modelos convencionales o prescriptitos

Los modelos convencionales surgen para dar orden al caos en los proyectos de desarrollo de software, a través de actividades, acciones, tareas, fundamentos y productos de trabajo que permitirán llevar un mayor control y seguimiento de los mismos. Los modelos convencionales en esencia son más estrictos o rígidos que los de desarrollo ágil, pero nos da mayor seguridad cuando se desea desarrollar un software de alta calidad.

A continuación se presenta diferentes modelos convencionales o prescrititos de proceso.

Tipos de modelos convencionales

a Modelo de cascada

En este tipo de modelo las fases se desarrollan de forma lineal ó secuencial. Es ideal cuando se desea realizar adaptaciones o mejoras bien definidas a software ya existentes, en caso de

que sea nuevo se hace imprescindible que se tengan los requerimientos bien claros y definidos (ver figura 1).

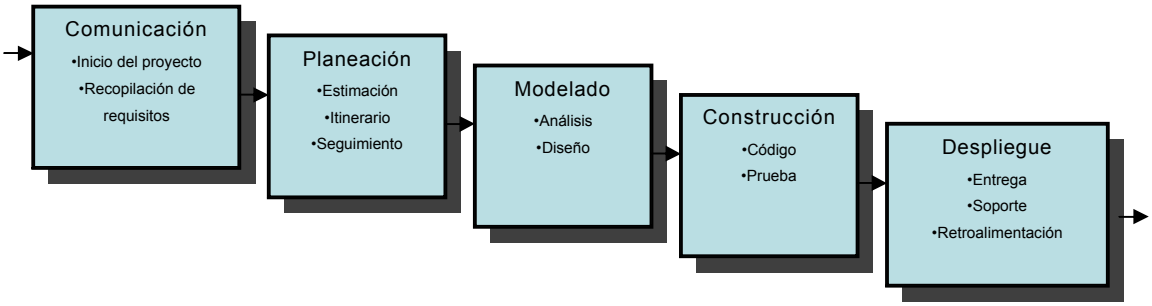

Fig 1. Modelo de cascada

Entre los problemas que se presentan con el modelo en cascada se mencionan los siguientes:

- Por lo general los proyecto de desarrollo de software, difícilmente siguen un modelo secuencial, y a pesar de que este modelo permite realizar iteraciones, se realizan de manera indirecta, confundiendo al equipo de proyecto, que se encuentra actuando.
- En muchas oportunidades al cliente se le hace difícil definir los requisitos de manera explicita.
- La primera versión que tendrá el cliente del producto se obtendrá cuando el proyecto se encuentre muy avanzado, así que un error grave será de gran impacto para el desarrollo del mismo.

b Modelos de proceso incrementales

En ocasiones se tienen bien definido en forma razonable los requisitos iniciales del software, pero se desea tener con celeridad un grupo de funcionalidades requeridas por el usuario, lo que conduce a ordenar las entregas de una manera incremental.

b.1Modelo incremental

Esta metodología utiliza el modelo de cascada de forma incremental como se muestra en la figura 2

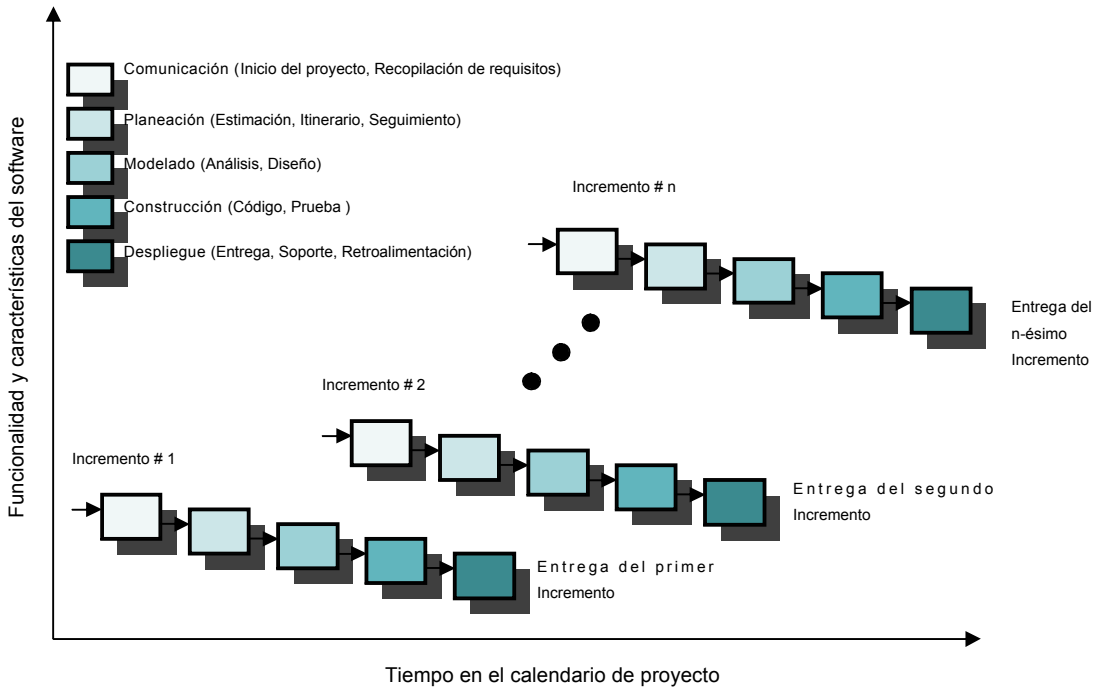

Fig 2. Modelo incremental

Ella permite priorizar las funcionalidades del sistema requeridas por los usuarios e ir desarrollándolas en función de las necesidades. Pero a diferencia de los modelos evolutivos que veremos mas adelante este modelo se centra en cada entregable a realizar, sin revisar o mejorar funcionalidades desarrolladas en iteraciones anteriores.

b.2Modelo de desarrollo rápido de aplicaciones (DRA)

El desarrollo rápido de aplicaciones (DRA) es un proceso de desarrollo incremental que resalta un ciclo de desarrollo corto. En el las actividades de modelado y construcción se realizan en forma de escala, teniendo que definir “n” número de equipos que trabajarán en un lapso de tiempo restringido entre los 60 y 90 días como se muestra en la figura 3, para

culminar con la integración de todas las funcionalidades desarrolladas en la etapa de despliegue.

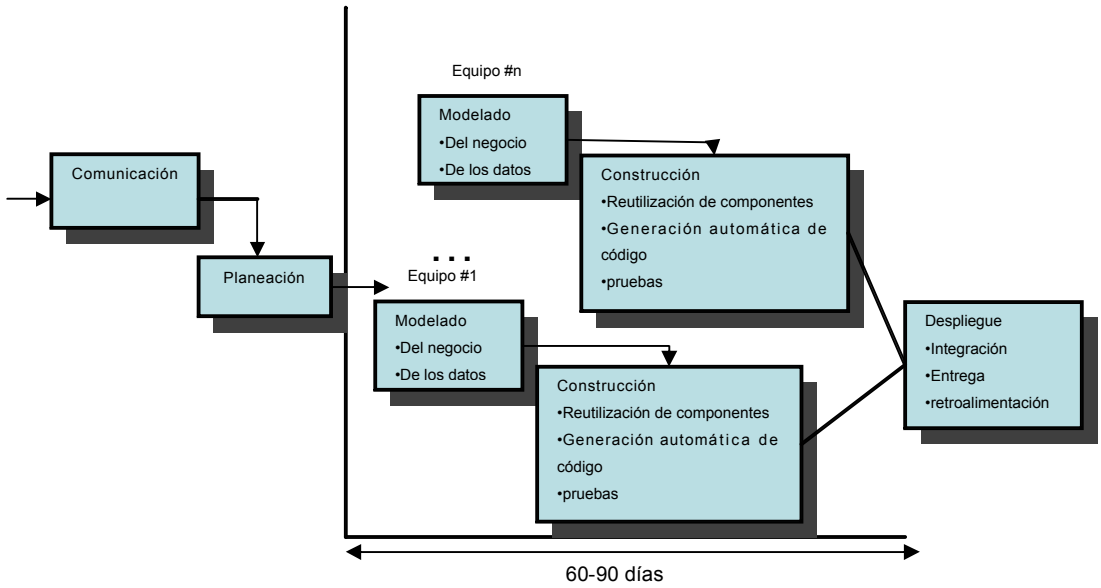

Fig 3. Modelo de desarrollo rápido de aplicaciones (DRA)

Esta metodología posee algunos inconvenientes:

- En proyectos grandes escalables se requiere de suficiente recursos humanos para definir el número correcto de equipos.
- Debe existir compromiso entre los clientes y los desarrolladores de las actividades rápidas necesarias para completar el sistema en un lapso breve de tiempo
- Esta metodología no es apropiada para cuando se tienen altos riesgos.

c Modelos de procesos evolutivos

Para los sistemas complejos, que evolucionan con el tiempo, es necesario utilizar una metodología que permita que los requisitos del negocio y productos vayan cambiando conforme se realiza el proyecto. Por lo tanto una ruta lineal que conduce a un producto final no da flexibilidad al producto para ir adecuándose a las reales necesidades del cliente.

Los modelos evolutivos permiten a los desarrolladores crear versiones cada vez más complejas del software. Entre los modelos evolutivos se tiene: la construcción de prototipos, modelos en espiral y el modelo de desarrollo concurrente.

c.1 Construcción de prototipos

Este modelo de proceso surge cuando no se tienen detallados los requisitos, como se llevará a cabo el procesamiento, ni lo que se tendrá al finalizar del mismo. Este modelo permite a los desarrolladores y clientes comprender en mayor escala lo que se desea buscar con el resultado de la construcción cuando los requisitos se encuentren cubiertos.

El modelo de construcción de prototipos inicia con la actividad de comunicación, continua con la realización de un plan rápido y un modelado ó diseño rápido, para luego construir el prototipo y desarrollarlo. Una vez que se tenga listo es entregado al cliente para recibir la retroalimentación que servirá para aclarar los requisitos o funcionalidades que debe poseer el sistema, en la figura 4 se muestra el ciclo de vida para este modelo.

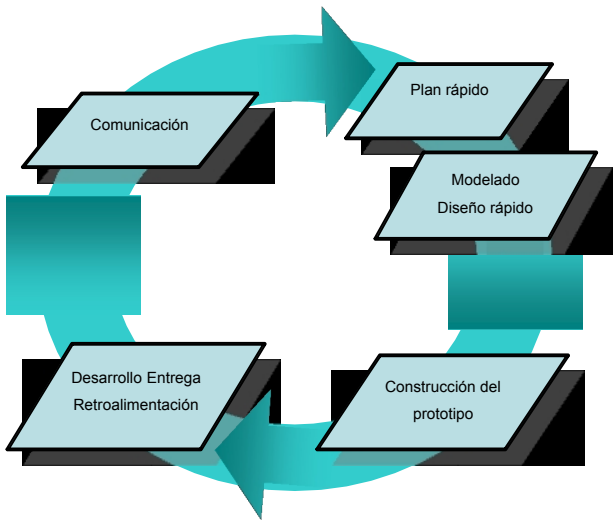

Fig 4. Modelo de prototipos

El prototipo sirve para que los clientes vean el sistema real en poco tiempo y los desarrolladores construir algo de inmediato. Pero la construcción de prototipos puede poseer algunas desventajas como lo son:

- Por la rapidez en que se está desarrollando el sistema se puede sacrificar la calidad en la construcción del mismo, lo que puede lograr que se dificulte el mantenimiento del mismo a largo plazo.
- También por la premura en realizar el prototipo no podemos encontrar con una utilización del sistema operativo o lenguaje de programación inadecuada, solo por que se encuentra disponible y es conocido.

La construcción de prototipos sin embrago puede ser efectivo si se definen las reglas desde un principio, que permita al desarrollador y cliente ponerse de acuerdo a la construcción del mismo y que sirva como mecanismo para la definición de requisitos

c.2 Modelo en espiral

Pertenece a los modelos de proceso evolutivos. En él, el software se desarrolla en una serie de versiones increméntales. Durante las primeras iteraciones. La versión incremental podría ser un modelo en papel o un prototipo. A medida que se va incrementando el número de iteraciones, se producen versiones cada vez mas completas.

El modelo en espiral se divide en un número de actividades estructurales, también llamadas regiones de tareas. Generalmente, existen entre tres y seis regiones de tareas. Las cuales son: Comunicación con el cliente, la planificación, el análisis de riesgo, la ingeniería, la construcción y adaptación y por último la evaluación del cliente (ver figura 5).

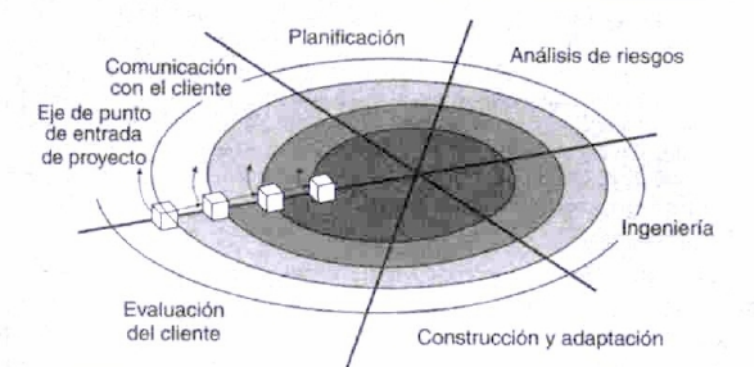

Fig 5. Modelo en espiral

Como ventajas de este modelo se tiene:

- Puede adaptarse y aplicarse a lo largo de la vida del software.
- Como el software evoluciona, a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos.
- Permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto.
- Demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto.
- Reduce los riesgos antes de que se conviertan en problemáticos.

Al igual que el resto de los modelos que no son perfectos, el modelo en espiral posee las siguientes desventajas:

- Demostrar al cliente "exigente" (bajo contrato) que el enfoque evolutivo es controlable.
- Requiere gran habilidad y experiencia para valorar el riesgo y saber cuando detener la evolución

c.3 Modelo de desarrollo concurrente

El modelo de proceso concurrente se puede representar en forma de esquema como una serie de actividades técnicas importantes, tareas y estados asociados a ellas, como se muestra en la figura 6.

El modelo de proceso concurrente define una serie de acontecimientos que dispararan transiciones de estado a estado para cada una de las actividades de la ingeniería del software.

Es utilizado en todos los tipos de desarrollo de software y proporciona una visión certera del estado actual del proyecto.

Cada actividad, acción o tarea dentro de la red existe de manera simultánea con otras. Los sucesos generados dentro de una actividad dada o algún otro lado de la red de actividad inicia las transiciones entre los estado de una actividad.

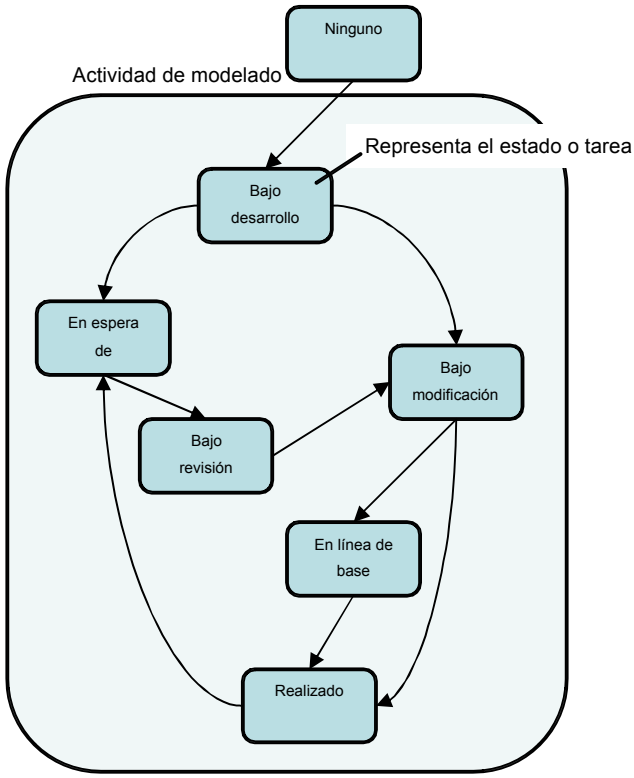

Fig 6. Modelo de desarrollo concurrente

2.2.2.2 Modelos de desarrollo ágil

Los modelos de desarrollo ágil no son la antitesis de la práctica sólida de la ingeniería del software, es decir de la vertiente convencional de los modelos que anteriormente fueron descritos, ya que todos estos modelos se basan de alguna u otra forma en uno o varios de los procesos prescriptivos, para la definición de sus pasos.

En la actualidad las condiciones del mercado se mantienen en un constante cambio, haciendo que las necesidades de los usuarios finales evolucionen, ya que cada día surgen nuevas amenazas competitivas que emergen sin previo aviso. En muchas ocasiones no es posible definir por completo los requisitos antes de que se inicie el proyecto

Las características esenciales del proceso de desarrollo ágil para la mayoría de los proyectos son las siguientes:

1. Dificultad para predecir los requerimientos que persistirán y cuáles cambiarán.
2. El diseño y la construcción están intercalados y deben realizarse de manera conjunta de modo que puedan ser aprobados conforme se crean.
3. El análisis, el diseño y la construcción no son predecibles.

Motivado a estas características el proceso ágil de software debe adaptarse de manera incremental. La idea es que se mantenga un canal de retroalimentación con el cliente, a través de entregas de prototipos ejecutable o porción de un sistema operacional, en periodos cortos para que la adaptabilidad mantenga un buen ritmo con el cambio.

A continuación se presenta diferentes modelos ágiles de proceso. Entre ellas existe mucha similitud sin embargo cada una posee característica únicas que las diferencian de las demás.

Tipos de modelos de desarrollo ágil

a. Programación extrema (PE)

Es una de las metodologías de desarrollo de software utilizadas en la actualidad para proyectos de corto plazo, con un equipo de proyecto pequeño. La metodología consiste en una programación rápida o extrema, utiliza un enfoque orientado a objetivos, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Esta metodología se encuentra compuesta de cuatro (4) actividades del marco del trabajo: planeación, diseño, codificación y pruebas (ver figura 7).

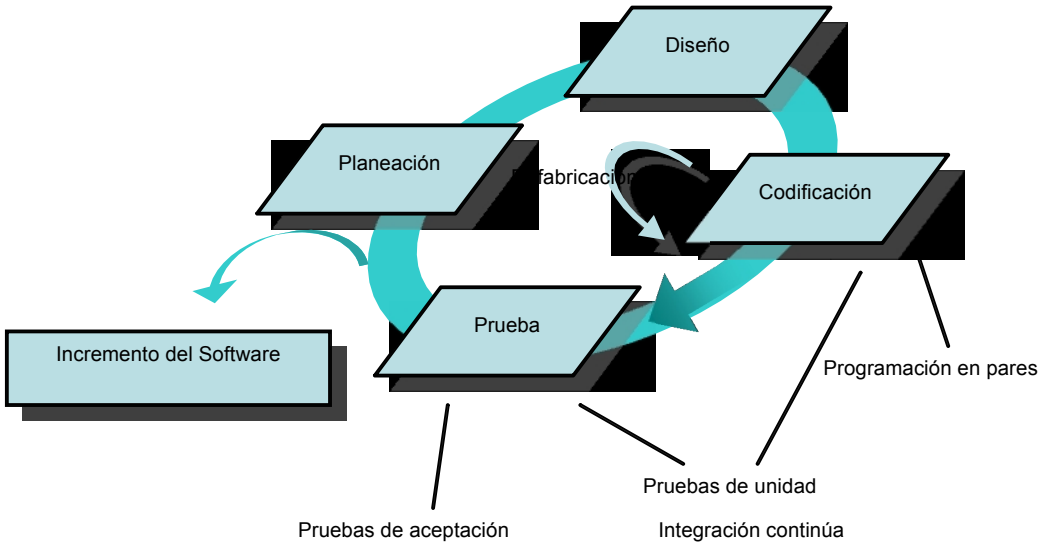

Fig 7. Modelo de Programación extrema

Las características más relevantes de la programación extrema son las siguientes:

Refabricación: se basa en la utilización repetitiva de código, para lo cual se establecen patrones, permitiendo mayor flexibilidad al cambio.

Programación en pares: Consiste en que dos desarrolladores trabajen para un proyecto en la misma estación de trabajo.

Pruebas: la fase de prueba se compone de dos tipos, las pruebas de unidad y las pruebas de aceptación. Las pruebas de unidad se basa en las pruebas realizadas a los principales procesos y las pruebas de aceptación son realizadas por los clientes y se enfoca en las característica generales del sistema de su parte visible y su funcionalidad como tal.

b. Desarrollo adaptativo de software (DAS)

El desarrollo adaptativo de software (DAS) fue propuesto por Jim Highsmith como una metodología para desarrollar software y sistemas muy complejos. El se centra en la colaboración humana y la organización del equipo.

El ciclo de vida del DAS se conforma de tres fases como se muestra en la figura 8, especulación, colaboración y aprendizaje.

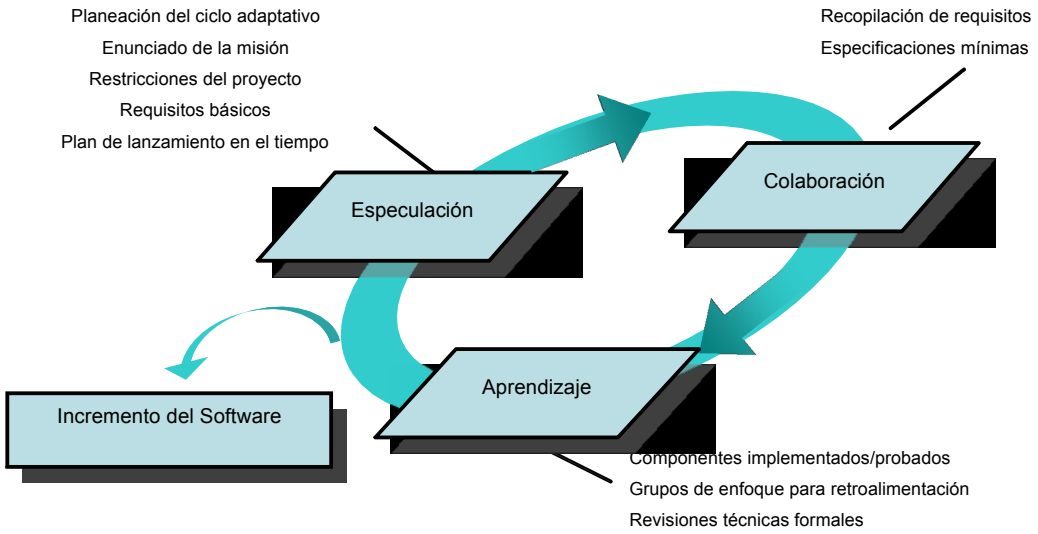

Fig 8. Modelo de Desarrollo adaptativo de software

En la fase de especulación se inicia el desarrollo del proyecto. En ella se utiliza información como la misión del cliente, las restricciones del proyecto y los requisitos básicos para definir el conjunto de ciclos en el que se harán los incrementos del software.

Para la fase de colaboración se busca que el equipo no solo se comunique o se encuentren completamente integrados, se desea que exista confianza, donde se pueda realizar críticas constructivas y ayudar sin resentimientos, trabajar tan duro como sea posible, comunicar de una forma oportuna los problemas que se presenten para tomar acciones efectivas y poseer un conjunto de actitudes que contribuyan al trabajo que se encuentran realizando.

El aprendizaje permite mejorar el entendimiento real sobre la tecnología, los procesos utilizados y el proyecto. El aprendizaje individual permite al equipo tener mayor posibilidad de éxito.

c. Modelo de desarrollo de sistemas dinámicos (MDSD)

El método de desarrollo de sistemas dinámicos (MDSD) permite la construcción de sistemas con restricción de tiempo, realizando prototipos incrementales en un ambiente de proyecto controlado.

Este modelo se compone de dos actividades que se realizan primero y consecuente con ellas se realizan tres ciclos de vida adicionales, las dos actividades primarias son el estudio de factibilidad en donde se establecen los requisitos básico del negocio y las restricciones asociadas a la metodología de manera de evaluar si la misma puede ser realizada bajo el esquema MDSD, y la segunda es el estudio del negocio donde se establecen los requerimientos funcionales y la arquitectura básica de la aplicación.

La iteración de modelo funcional, se realizan diversos prototipos incrementales que permite mostrar al cliente la funcionalidad del sistema. Con cada iteración se recopilan requisitos adicionales que pueden ir siendo incluidos en el prototipo.

La iteración de construcción y diseño, contribuye a agregar el valor operativo del negocio para los usuarios, a través de la construcción de prototipos durante la iteración del modelo funcional.

Implementación, los prototipos que vayan surgiendo de la fase de construcción y diseño, se Irán colocando en ambientes operativos (prototipos operacionales).

En la figura 9 se muestra la secuencia de estas actividades y las características de cada una de estas iteraciones.

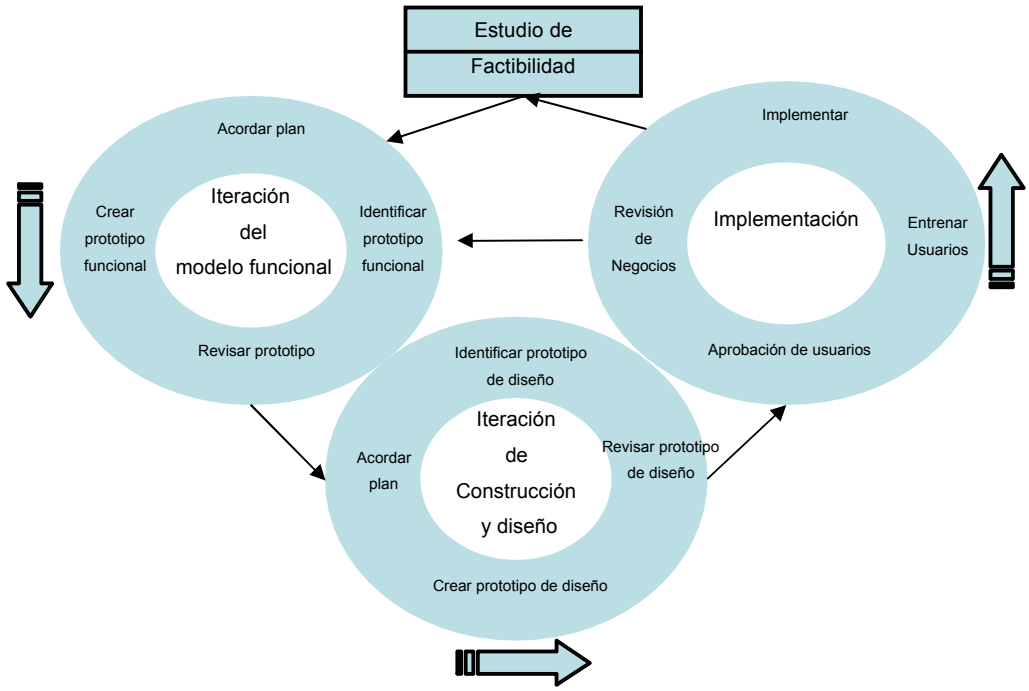

Fig 9. Modelo de desarrollo de sistemas dinámicos

d. Melé

Es un modelo ágil de proceso desarrollado por Jefff Suterland y su equipo a los comienzos de la década de los 90 (la palabra Melé proviene de una jugada de rugby). También es conocido con el nombre de Scrum.

Posee los siguientes principios que van en concordancia con los métodos de desarrollo ágiles:

- Equipos auto-dirigidos y auto-organizados.
- Una vez elegida una tarea, no se agrega trabajo extra. En caso que se agregue algo, se recomienda quitar alguna otra cosa.
- Encuentros diarios con las tres preguntas indicadas en la figura 10. Se realizan siempre en el mismo lugar, en círculo.
- Iteraciones de treinta días; aunque se pueden realizar con mas frecuencia.
- Demostración a participantes externos una vez culminada cada iteración.
- Al principio de cada iteración, planeamiento adaptativo guiado por el cliente.

El ciclo de vida de Melé es el siguiente:

Planeamiento: El propósito es establecer la visión, definir expectativas y asegurarse la financiación. Las actividades son la escritura de la visión, el presupuesto, el registro de acumulación o retraso del producto inicial y los ítems estimados, así como la arquitectura de alto nivel, el diseño exploratorio y los prototipos.

Montaje: El propósito es identificar más requerimientos y priorizar las tareas para la primera iteración. Las actividades son planificación, diseño exploratorio y prototipos.

Desarrollo: El propósito es implementar un sistema listo para entrega en una serie de iteraciones de treinta días llamadas “corridas”. Las actividades son un encuentro de planeamiento de corridas en cada iteración, la definición del registro de acumulación de corridas y los estimados, y encuentros diarios.

Liberación: El propósito es el despliegue operacional. Las actividades, documentación, entrenamiento, mercadeo y venta

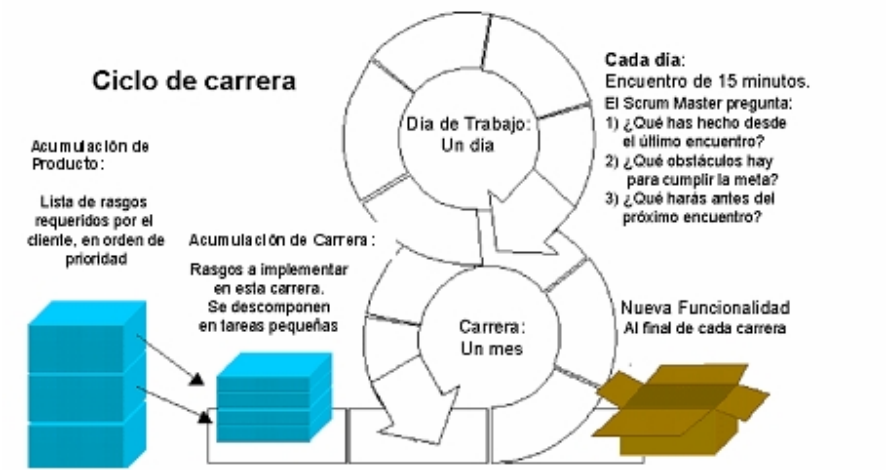

Fig 10. Flujo de proceso de la Melé

e. Desarrollo Conducido por características (DCC)

Es un modelo de proceso práctico para la ingeniería del software orientada a objetivos. Es aplicado en proyectos de software de tamaño moderado y grande. Para la metodología una característica es una función validad por el cliente y que puede ser implementada el dos o menos semanas. La definición de estas características permite a la metodología poseer los siguientes beneficios:

- Las características son pequeños bloques de funcionalidad entregables, los usuarios las describen con mayor facilidad, permitiéndoles revisarlas de mejor manera en búsqueda de errores u omisiones.
- Las características pueden ser agrupadas por orden jerárquico relacionado con el negocio.
- La característica es el incremento del software entregable, así que el equipo desarrolla características operativas cada dos semanas.
- La planificación del proyecto lo guía la jerarquía de la característica, en lugar de hacerlo un conjunto de tareas de la ingeniería de software adaptado en forma arbitraria

El DCC concede una mayor importancia a las directrices y técnicas de la gestión del proyecto que muchos otros métodos ágiles, ya que enfatiza cuestiones de calidad y define claramente entregas tangibles y formas de evaluación del progreso.

Se encuentra definido por cinco fases o actividades que se muestran a continuación (ver figura 11)

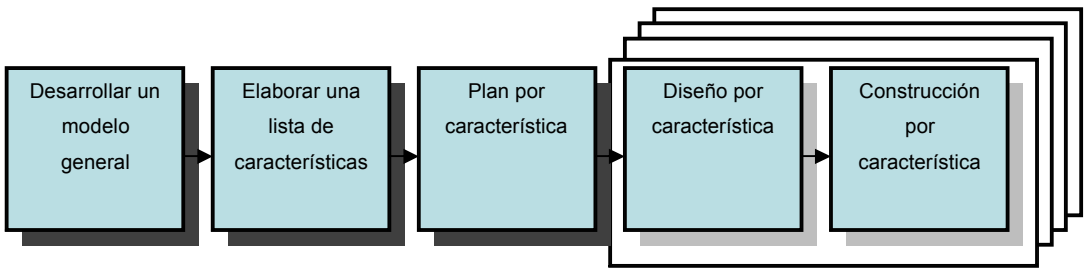

Fig 11. Desarrollo Conducido por características

Desarrollo de un modelo general, en esta fase ya se tiene el dominio de la visión, el contexto y los requerimientos del sistema a construir. En este momento se posee información básica de las especificaciones funcionales.

Construcción de la lista de características, Los ensayos, modelos de objeto y documentación de requerimientos proporcionan la base para construir una amplia lista de características. Las funciones se agrupan conforme a diversas actividades en áreas de dominio específicas. La lista de características es revisada por los usuarios para asegurar su validez y exhaustividad.

Diseño y Construcción por característica, Se selecciona las características a desarrollar y los equipos dispuestos por cada una de ellas. Luego se procede iterativamente hasta que se producen las características seleccionadas.

f. Proceso Unificado de Rational

Es una metodología que posee un poco de controversia, ya que cuenta con características esenciales de los procesos de desarrollo ágil como es el crecimiento iterativo o que se encuentra centrado en la arquitectura, pero a la vez tiende a caer en las rigidez de los métodos convencionales, de igual manera será estudiado en este capitulo motivado a su importancia y utilización en la actualidad para el desarrollo de software.

El proceso unificado de Rational implementa las siguientes mejores prácticas asociadas al proceso de Ingeniería de Software:

- Desarrollo Iterativo
- Manejo de los Requerimientos
- Uso de una Arquitectura basada en componentes
- Modelización Visual
- Verificación Continua de la Calidad
- Manejo de los Cambios

La metodología RUP, divide en 4 fases el desarrollo del software. Cada Fase tiene definido un conjunto de objetivos y un punto de control especifico. A saber:

Inicio, Se especifican los objetivos del ciclo de vida del proyecto y las necesidades de cada participante. Se establecer el alcance, las limitaciones y los criterios de aceptabilidad. Se identifican los casos de uso que orientarán la funcionalidad. Se diseñan las arquitecturas probables y se estima la duración.

Elaboración, se define el plan del proyecto. La fase de elaboración brinda una arquitectura, requerimientos y planes suficientemente sólidos y estables. Se describen en detalle la infraestructura y el ambiente de desarrollo. Se debe crear un prototipo de ella. Al final de la fase se realiza un análisis para determinar los riesgos.

Construcción, Se desarrollan, integran y verifican todos los componentes y rasgos de la aplicación. Los resultados de esta fase (las versiones alfa, beta y otras versiones de prueba) se crean tan rápido como sea posible. Se debe compilar también una versión de entrega. Es la fase más prolongada de todas.

Transición, Comienza cuando el producto está suficientemente maduro para ser entregado. Se corrigen los últimos errores y se agregan los rasgos pospuestos. La fase consiste en prueba beta, piloto y entrenamiento a usuarios. Se produce también la documentación.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

En la figura 12 se muestra gráficamente las fases e interrelación entre cada una de ellas durante el ciclo de vida del proyecto.

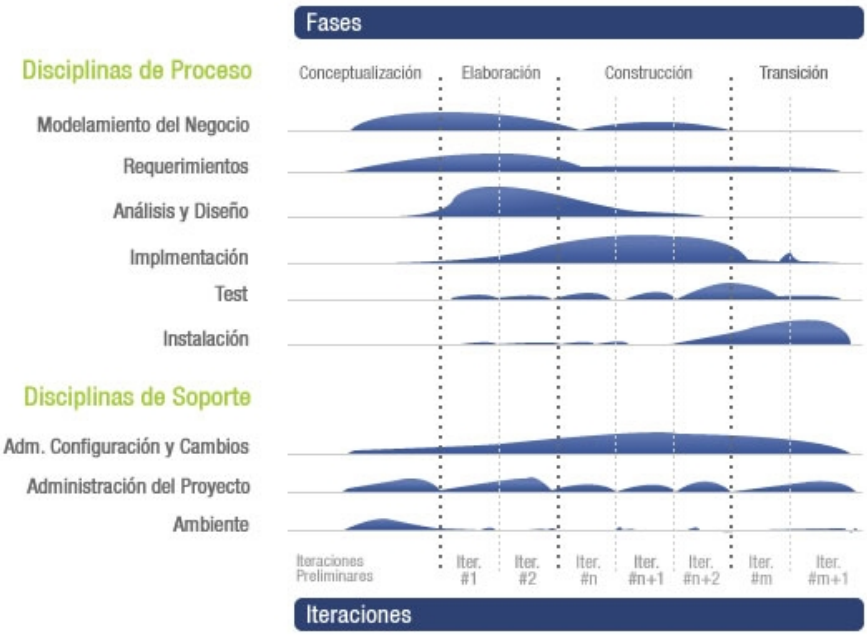

Fig 12. Proceso unificado de Rational

CAPITULO III. MARCO ORGANIZACIONAL

3.1 La empresa

Synergy Consultores, C.A. fundada en 1995, es una firma de consultoría especializada en tecnología de información, procesos y estrategia de negocio. Representante exclusivo de Garner para Venezuela, Colombia, Centro América y el Caribe.

Sus áreas de competencias se ilustran a continuación (ver figura 13):

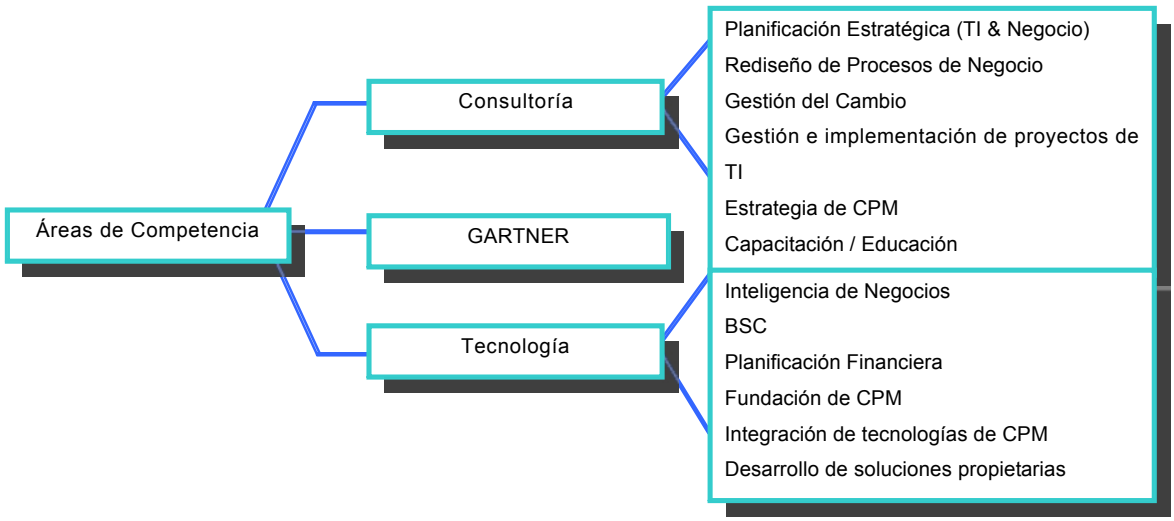

Fig 13. Áreas de competencia de Synergy Consultores, C.A.

3.1.1 Misión

Ayudar a sus clientes a lograr el éxito en sus negocios a través del uso inteligente, eficiente y oportuno de la tecnología de información.

3.1.2 Organigrama

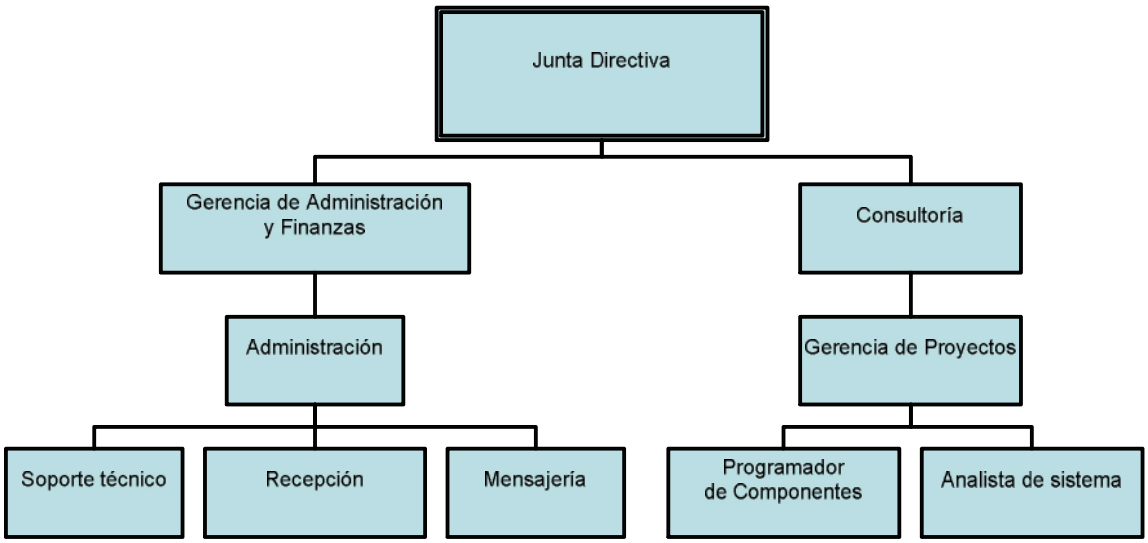

Fig 14. Organigrama Synergy Consultores, C.A.

3.1.3 Políticas de la empresa

- Enfoque balanceado: procesos, entorno, Tecnología de la Información, capital humano=valor
- Visión de Tecnología de la información como “impulsador” de la misión de la institución
- Capacidad integral: plan, diseño y ejecución
- Minimización de costos y riesgos con la incorporación de las mejores prácticas a nivel mundial
- Eficiencia e innovación en sus servicios profesionales
- Focalización y adaptación de metodologías al negocio de los clientes

CAPITULO IV. MATRIZ DE EVALUACIÓN DE METODOLOGÍAS

En el marco teórico se describió todo lo referente a la gerencia de proyectos según la guía del PMBOK, de igual manera se enfatizo en la gerencia de proyectos de software su marco de trabajo y los distintos modelos de procesos existentes. Esto permitió tener una idea de cuales son algunas de las variables que se encuentran presentes en cada una de ellas y que permiten una adecuación entre el proyecto a ejecutar y la naturaleza del mismo.

Paralelo a esto se obtuvo información de distintas páginas Web y material bibliográfico sobre las variables que se deben tomar en cuenta para la creación o selección de una metodología de desarrollo de software y donde se ubicó un reportaje que engloba con gran certeza las variables mencionadas anteriormente.

4.1 Identificación de variables en la evaluación de metodologías a ser utilizadas en el desarrollo de software

Según el autor Rafael Menéndez - Barzanallana Asensio[3] para construir o elegir una metodología se han de considerar unos requisitos deseables, que se mencionan a continuación

1. La metodología debe ajustarse a los objetivos. Cada aproximación al desarrollo de software está basada en unos objetivos. Por ello la metodología que se elija debe recoger el aspecto filosófico de la aproximación deseada, es decir que los objetivos generales del desarrollo deben estar implementados en la metodología de desarrollo.

2. La metodología debe cubrir el ciclo entero de desarrollo de software. Para ello la metodología ha de realizar unas etapas: Investigación, Análisis de requisitos, Diseño.

---

[3] Rafael Menéndez - Barzanallana Asensio. Profesor de la Universidad de Murcia. Asignatura Informática Aplicada a la Gestión Pública. Curso 2005/2006 del Departamento Informática y Sistemas.

3. La metodología debe integrar las distintas fases del ciclo de desarrollo. Rastreabilidad. Es importante poder referirse a otras fases de un proyecto y fusionarlo con las fases previas. Es importante poder moverse no sólo hacia adelante en el ciclo de vida, sino hacia atrás de forma que se pueda comprobar el trabajo realizado y se puedan efectuar correcciones. Fácil interacción entre etapas del ciclo de desarrollo. Es necesaria una validación formal de cada fase antes de pasar a la siguiente. La información que se pierde en una fase determinada queda perdida para siempre, con un impacto en el sistema resultante.

4. La metodología debe incluir la realización de validaciones. La metodología debe detectar y corregir los errores cuanto antes. Uno de los problemas más frecuentes y costosos es el aplazamiento de la detección y corrección de problemas en las etapas finales del proyecto. Cuanto más tarde sea detectado el error más caro será corregirlo. Por lo tanto cada fase del proceso de desarrollo de software deberá incluir una actividad de validación explícita.

5. La metodología debe soportar la determinación de la exactitud del sistema a través del ciclo de desarrollo. La exactitud del sistema implica muchos asuntos, incluyendo la correspondencia entre el sistema y sus especificaciones, así como que el sistema cumple con las necesidades del usuario. Por ejemplo, los métodos usados para análisis y especificación del sistema deberían colaborar a terminar con el problema del entendimiento entre los informáticos, los usuarios, y otras partes implicadas. Esto implica una comunicación entre usuario y técnico amigable y sencillo, exento de consideraciones técnicas.

6. La metodología debe ser la base de una comunicación efectiva. Debe ser posible gestionar a los informáticos, y éstos deben ser capaces de trabajar conjuntamente. Ha de haber una comunicación efectiva entre analistas, programadores, usuarios y gestores, con pasos bien definidos para realizar progresos visibles durante la actividad del desarrollo.

7. La metodología debe funcionar en un entorno dinámico orientado al usuario. A lo largo de todo el ciclo de vida del desarrollo se debe producir una transferencia de conocimientos hacia el usuario. La clave del éxito es que todas las partes implicadas han de intercambiar información libremente. La participación del usuario es de importancia vital debido a que sus necesidades evolucionan constantemente. Por otra parte la adquisición de conocimientos del usuario la permitirá la toma de decisiones correctas. Para involucrar al usuario en el análisis,

diseño y administración de datos, es aconsejable el empleo de técnicas estructuradas lo más sencillas posible. Para esto, es esencial contar una buena técnica de diagramación.

8. La metodología debe especificar claramente los responsables de resultados. Debe especificar claramente quienes son los participantes de cada tarea a desarrollar, debe detallar de una manera clara los resultados de los que serán responsables.

9. La metodología debe poder emplearse en un entorno amplio de proyectos software. Variedad. Una empresa deberá adoptar una metodología que sea útil para un gran número de sistemas que vaya a construir. Por esta razón no es práctico adoptar varias metodologías en una misma empresa. Tamaño, vida. Las metodologías deberán ser capaces de abordar sistemas de distintos tamaños y rangos de vida. Complejidad. La metodología debe servir para sistemas de distinta complejidad, es decir puede abarcar un departamento, varios de departamentos o varias empresas. Entorno. La metodología debe servir con independencia de la tecnología disponible en la empresa.

10. La metodología se debe de poder enseñar. Incluso en una organización sencilla, serán muchas las personas que la van a utilizar, incluso los que se incorporen posteriormente a la empresa. Cada persona debe entender las técnicas específicas de la metodología, los procedimientos organizativos y de gestión que la hacen efectiva, las herramientas automatizadas que soportan la metodología y las motivaciones que subyacen en ella.

11. La metodología debe estar soportada por herramientas CASE. La metodología debe estar soportada por herramientas automatizadas que mejoren la productividad, tanto del ingeniero de software en particular, como la del desarrollo en general. El uso de estas herramientas reduce el número de personas requeridas y la sobrecarga de comunicación, además de ayudar a producir especificaciones y diseños con menos errores, más fáciles de probar, modificar y usar.

12. La metodología debe soportar la eventual evolución del sistema. Normalmente durante su tiempo de vida los sistemas tienen muchas versiones, pudiendo durar incluso más de 10 años. Existen herramientas CASE para la gestión de la configuración y otras denominadas

"Ingeniería inversa" para ayudar en el mantenimiento de los sistemas no estructurados, permitiendo estructurar los componentes de éstos facilitando así su mantenimiento.

13. La metodología debe contener actividades conducentes a mejorar el proceso de desarrollo de software. Para mejorar el proceso es básico disponer de datos numéricos que evidencian la efectividad de la aplicación del proceso con respecto a cualquier producto software resultante del proceso. Para disponer de estos datos, la metodología debe contener un conjunto de mediciones de proceso para identificar la calidad y coste asociado a cada etapa del proceso. Sería ideal el uso de herramientas CASE.

Apoyándonos en estas variables perfectamente identificadas por el profesor Barzallana, se creo una encuesta, que permitió ordenar o priorizar cada una de ellas dándole una ponderación en función a la importancia que los distintos involucrados con desarrollo de software le da a cada una de estas características.

Estas ponderaciones permitirán crear una matriz de evaluación de las metodologías existentes en el desarrollo de software y que fueron detalladas en el capitulo II.

La encuesta se diseño de manera que las personas que la llenaron pudieran seleccionar el grado de importancia que le daba a cada una de las características a través de la siguiente escala:

1 = Totalmente en desacuerdo
2 = Desacuerdo
3 = indeciso/neutral
4 = De acuerdo
5 = Totalmente de acuerdo

Esta escala a su vez fue ponderada, de manera que se pudiera obtener los resultados donde 5 era el 100%, 4 el 80%, 3 el 60%, 2 el 40% y el 1 el 10%

Para la obtención de los resultados se creo una hoja de Excel que me permite ir agregando los resultados de las encuesta, de manera que se puedan ir obteniendo los resultados de

forma automática con las ponderaciones respectivas para cada una de las variables (ver figura 15).

	Puntuación obtenida					Ponderada					Total
						0,2	0,4	0,6	0,8	1	
Variables	1	2	3	4	5	1	2	3	4	5	
La metodología debe ajustarse a los objetivos			1	3	3	0	0	0,1429	0,4286	0,4286	0,857
La metodología debe cubrir el ciclo entero de desarrollo de software			1	1	5	0	0	0,1429	0,1429	0,7143	0,914
La metodología debe integrar las distintas fases del ciclo de desarrollo				3	4	0	0	0	0,4286	0,5714	0,914
La metodología debe incluir la realización de validaciones			1	3	3	0	0	0,1429	0,4286	0,4286	0,857
La metodología debe soportar la determinación de la exactitud del sistema a través del ciclo de desarrollo.			2	3	2	0	0	0,2857	0,4286	0,2857	0,800
La metodología debe ser la base de una comunicación efectiva.		1		3	3	0	0,14	0	0,4286	0,4286	0,829
La metodología debe funcionar en un entorno dinámico orientado al usuario		2	2	1	2	0	0,29	0,2857	0,1429	0,2857	0,686
La metodología debe especificar claramente los responsables de resultados		1		3	3	0	0,14	0	0,4286	0,4286	0,829
La metodología debe poder emplearse en un entorno amplio de proyectos software				4	3	0	0	0	0,5714	0,4286	0,886
La metodología se debe de poder enseñar			1	5	1	0	0	0,1429	0,7143	0,1429	0,800
La metodología debe estar soportada por herramientas CASE			3	3	1	0	0	0,4286	0,4286	0,1429	0,743
La metodología debe soportar la eventual evolución del sistema		1	1	2	3	0	0,14	0,1429	0,2857	0,4286	0,800
La metodología debe contener actividades conducentes a mejorar el proceso de desarrollo de software.			3	2	2	0	0	0,4286	0,2857	0,2857	0,771

Fig 15. Tabla de cálculo de las ponderaciones de las variables

Una vez obtenido los resultados para la ponderación de las variables y su ordenamiento según su grado de importancia para las personas involucradas en el desarrollo de software, que fueron sometidas a la encuesta se procede a diseñar la matriz de evaluación

4.2 Diseño de la matriz de evaluación de metodologías

Para el diseño de la matriz se indago sobre aquellas herramientas de trabajo que permitieran una comparación entre las metodologías y que sirviera como base para el desarrollo de esta herramienta de evaluación.

De las herramientas ubicadas, se consideró la utilización de la matriz de perfil competitivo, utilizada en el área de planificación estratégica, donde se identifica los factores claves de éxito de las empresas para su posterior comparación. En nuestro caso serán los factores claves de éxito las características con las que debería contar una metodología, para su creación o selección.

La matriz de perfil competitivo identifica los principales competidores, en este caso las metodologías, así como sus fortalezas y debilidades particulares.

Los pasos a seguir para la utilización de la matriz de perfil competitivo se presenta a continuación:

1. Identificar los factores claves de éxito
2. Asignar ponderación a cada factor clave de éxito.
3. Se asigna fortaleza o debilidad a cada factor por competidor (metodología)
   3.1. Debilidad grave = 1
   3.2. Debilidad menor = 2
   3.3. Fortaleza menor = 3
   3.4. Fortaleza importante = 4
4. Calcular los resultados ponderados
5. Sumar resultados

A medida que se van realizando lo pasos antes descrito, se obtiene la información requerida para ir llenando la matriz de perfil competitivo que se muestra en la figura 15

Matriz de Perfil Competitivo

Factores clave de éxito	Ponderación	M1		M2		M3	
		Puntuación	Resultado ponderado	Puntuación	Resultado ponderado	Puntuación	Resultado ponderado

Fig 16. Matriz de perfil competitivo

Con la información obtenida a través de las encuestas, se logró ponderar todos los factores claves de éxito que deben estar presente en la selección de la metodología, luego se construyo la matriz con todas las metodologías estudiadas ó analizadas en este trabajo especial de grado para su posterior evaluación en la herramienta como se muestra a continuación (ver figuras 17 y 18).

Matriz de evaluación de metodologías (convencionales ó prescriptivas)

Factores clave de éxito	Ponderación	Cascada		Incremental		DRA		Prototipos		Espiral		Concurrente	
		Puntuación	Resultado ponderado	Puntuación	Resultado ponderado	Puntuación	Resultado ponderado	Puntuación	Resultado ponderado	Puntuación	Resultado ponderado	Puntuación	Resultado ponderado
La metodología debe ajustarse a los objetivos	0,080	1	0,080	1	0,080	1	0,080	1	0,080	2	0,160	2	0,160
La metodología debe cubrir el ciclo entero de desarrollo de software	0,086	2	0,171	3	0,257	2	0,171	3	0,257	3	0,257	3	0,257
La metodología debe integrar las distintas fases del ciclo de desarrollo	0,086	3	0,257	3	0,257	3	0,257	3	0,257	3	0,257	2	0,171
La metodología debe incluir la realización de validaciones	0,080	1	0,080	1	0,080	1	0,080	2	0,160	3	0,241	3	0,241
La metodología debe soportar la determinación de la exactitud del sistema a través del ciclo de desarrollo.	0,075	2	0,150	2	0,150	2	0,150	3	0,225	2	0,150	2	0,150
La metodología debe ser la base de una comunicación efectiva.	0,078	2	0,155	2	0,155	2	0,155	2	0,155	3	0,233	3	0,233
La metodología debe funcionar en un entorno dinámico orientado al usuario	0,064	3	0,193	2	0,128	2	0,128	3	0,193	3	0,193	3	0,193
La metodología debe especificar claramente los responsables de resultados	0,078	3	0,233	3	0,233	3	0,233	3	0,233	2	0,155	3	0,233
La metodología debe poder emplearse en un entorno amplio de proyectos software	0,083	1	0,083	1	0,083	2	0,166	2	0,166	3	0,249	4	0,332
La metodología se debe de poder enseñar	0,075	4	0,299	4	0,299	4	0,299	4	0,299	4	0,299	4	0,299
La metodología debe estar soportada por herramientas CASE	0,070	1	0,070	1	0,070	1	0,070	1	0,070	2	0,139	1	0,070
La metodología debe soportar la eventual evolución del sistema	0,075	1	0,075	1	0,075	1	0,075	2	0,150	4	0,299	3	0,225
La metodología debe contener actividades conducentes a mejorar el proceso de desarrollo de software.	0,072	1	0,072	1	0,072	1	0,072	1	0,072	4	0,289	3	0,217
			1,917		1,939		1,936		2,316		2,920		2,778

Fig 17. Matriz de evaluación de metodologías (convencionales ó prescriptitas)

Matriz de evaluación de metodologías (procesos de desarrollo ágil)

Factores clave de éxito	Ponderación	Programación extrema		DAS		MDSD		Melé		DCC		PUR	
		Puntuación	Resultado ponderado	Puntuación	Resultado ponderado	Puntuación	Resultado ponderado	Puntuación	Resultado ponderado	Puntuación	Resultado ponderado	Puntuación	Resultado ponderado
La metodología debe ajustarse a los objetivos	0,080	4	0,321	3	0,241	2	0,160	1	0,080	4	0,321	4	0,321
La metodología debe cubrir el ciclo entero de desarrollo de software	0,086	3	0,257	3	0,257	2	0,171	3	0,257	4	0,342	4	0,342
La metodología debe integrar las distintas fases del ciclo de desarrollo	0,086	3	0,257	2	0,171	3	0,257	3	0,257	4	0,342	4	0,342
La metodología debe incluir la realización de validaciones	0,080	4	0,321	3	0,241	3	0,241	4	0,321	3	0,241	4	0,321
La metodología debe soportar la determinación de la exactitud del sistema a través del ciclo de desarrollo.	0,075	3	0,225	3	0,225	2	0,150	3	0,225	3	0,225	4	0,299
La metodología debe ser la base de una comunicación efectiva.	0,078	4	0,310	3	0,233	2	0,155	4	0,310	3	0,233	3	0,233
La metodología debe funcionar en un entorno dinámico orientado al usuario	0,064	4	0,257	4	0,257	3	0,193	4	0,257	3	0,193	3	0,193
La metodología debe especificar claramente los responsables de resultados	0,078	4	0,310	3	0,233	3	0,233	3	0,233	3	0,233	3	0,233
La metodología debe poder emplearse en un entorno amplio de proyectos software	0,083	2	0,166	2	0,166	1	0,083	3	0,249	2	0,166	3	0,249
La metodología se debe de poder enseñar	0,075	4	0,299	4	0,299	3	0,225	4	0,299	4	0,299	2	0,150
La metodología debe estar soportada por herramientas CASE	0,070	2	0,139	2	0,139	2	0,139	2	0,139	2	0,139	4	0,278
La metodología debe soportar la eventual evolución del sistema	0,075	3	0,225	3	0,225	2	0,150	3	0,225	3	0,225	3	0,225
La metodología debe contener actividades conducentes a mejorar el proceso de desarrollo de software.	0,072	2	0,144	2	0,144	3	0,217	3	0,217	3	0,217	3	0,217
			3,230		2,829		2,372		3,067		3,174		3,401

Fig 18. Matriz de evaluación de metodologías (proceso de desarrollo ágil)

CAPITULO V. ANÁLISIS DE RESULTADOS

5.1 Resultados obtenidos en las encuestas

En las encuestas realizadas, los cinco (5) primeros factores que se obtuvieron en mayor escala fueron los que a continuación se presentan:

La metodología debe ajustarse a los objetivos	0,857
La metodología debe cubrir el ciclo entero de desarrollo de software	0,914
La metodología debe integrar las distintas fases del ciclo de desarrollo	0,914
La metodología debe incluir la realización de validaciones	0,857
La metodología debe poder emplearse en un entorno amplio de proyectos software	0,886

En ella se puede observar que para las personas encuestadas la prioridad lo tiene el de cubrir el ciclo entero de desarrollo de software e integrar las distintas fases del ciclo. Para luego dar importancia a que la metodología pueda ser empleada en un entorno amplio de los proyectos.

Sin embargo, estos resultados, dejan por fuera factores que de alguna manera son de gran importancia a la hora de diseñar una metodología como los es, que la metodología debería funcionar en un entorno dinámico orientado al usuario.

5.2 Resultados obtenidos en la matriz de evaluación de metodologías

Para la aplicación de la matriz de evaluación de metodologías se obtuvo que la mayor ponderación se presento en los procesos de desarrollo ágil. Esto obedece a que estas metodologías son la mejora continua de las metodologías ya existentes como son las convencionales o prescriptivas. Las metodologías de desarrollo de software continúan evolucionando y cada vez se adecuan o cubren en mejor medida las necesidades y requerimientos de los proyectos de esta índole.

Dentro de las metodologías de desarrollo ágil, los procesos que obtuvieron la mayor ponderación son la de proceso unificado de Racional y la programación extrema, sin embrago esta dos metodologías son aplicadas para proyectos con diferencias de duración es decir la programación extrema es para proyectos cortos y que se deseen con rapidez y el proceso unificado es para desarrollos de software mas complejos y de mayor duración.

Dentro de las metodologías convencionales o prescriptivas la que obtuvo la mayor ponderación es el modelo en espiral, ya que permite que el sistema evolucione, posee gran interacción con el usuario, entre otros.

CAPITULO VI. CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

- Se identificaron once (11) metodologías utilizadas para el desarrollo de software, con sus características particulares, obteniendo las ventajas y desventajas que cada una de ellas presentan.

- Se identificaron los factores claves que se deben tener en cuenta para la selección de una metodología. Se tomó como referencia las variables determinadas por el profesor Barzanallana de la universidad de Murcia de España, ya que estas, cubren con la mayoría de las características identificadas en la obtención de información.

- El resultado de las encuestas arrojó que los tres (3) factores más importantes fueron que: la metodología debe integrar las distintas fases del ciclo de desarrollo, cubrir el ciclo entero de desarrollo de software y la posibilidad de poder emplearse en un entorno amplio de proyectos software; las ponderaciones para cada uno de ellos fueron de 0,914; 0,914 y 0,886, respectivamente en una escala de 0 a 1.

- Se diseño la matriz de evaluación de metodologías utilizando como base la matriz de perfil competitivo que se emplea en el área de planificación estratégica.

- Se desarrollo la matriz obteniendo los resultados que se presentaron en el capitulo IV, la metodología que satisface en mayor medida a los factores de éxito fue la programación unificada de Rational, que pertenece a los tipos de metodologías de desarrollo ágil con un puntaje de 3,230 en un rango de 1 al 4.

6.2 Recomendaciones

- La encuesta debe realizarse a un número significativo de personas con conocimientos en el área de desarrollo de software que permitan obtener resultados más fidedignos, mientras mas experiencia posean las personas encuestadas en el desarrollo de software, los resultados permitirán obtener con mayor veracidad las ponderaciones de cada uno de los factores evaluados .

- Los factores a evaluar en cada metodología pueden ser creados por el líder del proyecto en función a las características que puede tener un proyecto en específico.

- Para realizar la matriz de evaluación de metodologías es imprescindible que la persona que va a realizar la ponderación conozca a fondo las metodologías a evaluar, ya que esto podría traer como consecuencia no seleccionar la que en realidad es la más conveniente para el tipo de proyecto a desarrollar.

BIBLIOGRAFÍA

1. Proyect Management Institute, INC. (2004).Guía de los fundamentos de la dirección de Proyectos (tercera edición). Pennsylvania: Autor
2. Senn, J. (1992). Análisis y Diseño de Sistemas de Información (Segunda Edición). México: McGraw Hill.
3. Corporación Colombia Digital. Fomento a Industria de TICs. Estudio de la Industria del Software en Venezuela. 01 de mayo de 2005. (http://www.colombiadigital.net/informacion/docs/Soft_ve.pdf)
4. Grupo de Gestión de la Tecnología Escuela Técnica Superior de Ingenieros de Telecomunicación Universidad Politécnica de Madrid. Articulo: El ciclo de vida. 23 de mayo de 2006 . (http://www.getec.etsit.upm.es/docencia/gproyectos/planificacion/cvida.htm)
5. Pressman, Roger S. (2005). Ingenieria del software. Un enfoque práctico (Sexta Edición). México: McGraw Hill.
6. file:///J:/tesis/entregables/Jueves%2007-06-06/visitas/Metodolog%EDa%20Reynox%20%20Divisi%F3n%20Proyectos%20SAP%20%20Reynox%20Soluciones%20Inform%E1ticas.htm (metodología RUP y XP)
7. Metodologías de desarrollo de software ágil http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.asp#11
8. Universidad de Murcia (España). Página del profesor Rafael Barzanallana. Asignatura de informática aplicada a la gestión pública. Metodologías de desarrollo de software. 16 de junio de 2006 http://www.um.es/docencia/barzana/IAGP/Iagp3.html#BM2
9. Wikipedia La enciclopedia libre. Ingeniería de software. 16 de junio de 2006. http://es.wikipedia.org/wiki/IngenierÃ-a_de_software
10. Google. Imágenes. Paradigma del Modelo espiral. Caracas 16 de junio de 2006. http://images.google.co.ve/imgres?imgurl=http://148.202.148.5/cursos/cc321/fundamentos/imagenes/espiral1.jpg&imgrefurl=http://148.202.148.5/cursos/cc321/fundamentos/unidad1/espiral.htm&h=546&w=800&sz=42&hl=es&start=24&tbnid=bEmzGFcviU4CEM:&tbnh=96&tbnw=142&prev=/images%3Fq%3Dmodelo%2Bespiral%26start%3D20%26ndsp%3D20%26svnum%3D10%26hl%3Des%26lr%3D%26sa%3DN
11. Google. Imágenes. El modelo de desarrollo concurrente. Caracas 16 de junio de 2006 http://images.google.co.ve/imgres?imgurl=http://148.202.148.5/cursos/cc321/fundamentos/imagenes/concurrnte.jpg&imgrefurl=http://148.202.148.5/cursos/cc321/fundamentos/unidad1/concurrente.htm&h=564&w=426&sz=25&hl=es&start=1&tbnid=RZuaJYG8f4kiqM:&tbnh=131&tbnw=98&prev=/images%3Fq%3Dmodelo%2Bdesarrollo%2Bconcurrente%26svnum%3D10%26hl%3Des%26lr%3D