Universidad de las Ciencias Informáticas
Facultad 3

Título: Referencia para la aplicación de la metodología ágil Programación Extrema a proyectos DESOFT SS.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Martha Karina Pardillo Castañeda

Tutor(es): Ing. Karina Pérez Teruel

Ms. Lídice Álvarez

Asesor: Lic. Jorge Fardales Pérez

Junio de 2007

"Si buscas resultados distintos, no hagas siempre lo mismo."

Albert Einstein.

DECLARACIÓN DE AUTORÍA

Declaro que yo, Martha Karina Pardillo Castañeda soy el único autor de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los 30 días del mes de junio del año 2007.

Martha Karina Pardillo Castañeda

_____

Firma del autor

Lídice Álvarez

_____

Firma del tutor

Karina Pérez Teruel

_____

Firma del tutor

AGRADECIMIENTOS

A la Revolución, por darme la oportunidad de realizar lo que realmente imagine para mí, especialmente a la Universidad de Ciencias Informáticas por concretar este anhelo.

A mis padres y hermano que siempre han estado ahí apoyándome en todo, aconsejándome, siempre de mi lado y que han hecho tanto como yo para lograr esta gran tarea que ahora ya culmina. Especial para ustedes que me han dado fuerzas y porque también es uno de sus sueños. Gracias por confiar en mí.

A mi tía Maricela: por apoyarme en cada uno de mis pasos y darme ánimos para continuar.

A Mami: abuela, por preocuparte en todo momento por mi.

A mi abuelo: aunque ya no estés aquí físicamente sigues tan firme en mi corazón como siempre, lo que siempre me inculcaste aquí lo termino para ti.

A Lídice: muchísimas gracias por tu ayuda, gran parte de esto te lo debo a ti, gracias por todo, de corazón.

A Fardales: gracias por tu importante contribución, por tu ayuda.

A Churri, mi amor, por estar siempre ahí tratando de solucionar mis problemas, por sobrellevar cada uno de mis muchos percances durante el transcurso de este ardua tarea, a ti, gracias por estar a mi lado.

A Blanquita, Tati, Ada por cada granito de arena que pusieron en esta gran montaña.

A Heidy y Ray por advertirme, ayudarme y enseñarme muchísimas cosas.

A Ibelys y Víctor por su preocupación y cooperación.

A todos aquellos que de una forma u otra contribuyeron a que este, mi sueño, se realizara.

DEDICATORIA

"A las personas que quiero, por apoyarme siempre."

RESUMEN

Las metodologías de desarrollo permiten establecer una organización del proceso de desarrollo de software por lo que representan un eslabón fundamental para lograr éxito en el proceso de producción. Las metodologías ágiles son una nueva tendencia surgida como solución para algunas de las dificultades que las metodologías tradicionales, creadas anteriormente, están presentando. A través del siguiente trabajo se brinda la propuesta de una referencia para la aplicación de la metodología ágil Programación Extrema para la empresa Desoft SS (Sancti Spíritus), en cuya entidad se prevé la reestructuración del proceso de desarrollo a través de la implementación de una nueva metodología que se adapte a las condiciones existentes en dicha empresa siendo las metodologías ágiles la vía de solución para el logro satisfactorio del proceso de desarrollo de software, pero de cuya metodología no se tiene el conocimiento suficiente para su implementación. La misma está encaminada a suministrar una guía para determinar cuándo se está en condiciones de aplicar una metodología ágil y de qué forma hacerlo y así proporcionar una mayor eficiencia al proceso de producción de software de acuerdo a las dificultades que la empresa presenta. Dicha referencia fue aplicada en la empresa Desoft SS arrojando resultados satisfactorios que indican la posibilidad que brinda la misma para aquellos principiantes que se adentran en el mundo de la agilidad para los procesos de desarrollo de software.

TABLA DE CONTENIDOS

Introducción

Las notaciones de modelado y las herramientas implementadas para ser utilizadas en el desarrollo de software surgieron con el objetivo de propiciar una solución fundamental para lograr éxito durante la producción del mismo. No obstante, de acuerdo a los resultados obtenidos después de la aplicación de estos medios, quedó demostrado que los mismos no eran suficientes. Todo esto surge como consecuencia de la exclusión de las metodologías como pieza esencial en el proceso de producción de software. El hecho de contar con potentes y eficaces notaciones y herramientas no significa que estas se utilicen de la forma más eficiente posible. Se necesita entonces de un procedimiento que especifique cómo aplicarlas si se quiere obtener un producto satisfactorio en relación con la utilización de ambos métodos. Este hecho ha traído consigo un gran interés en las metodologías de desarrollo las cuales se han iniciado con un gran auge(LETELIER and PENADÉS 2006).

Para el proceso de producción de software se emplean diferentes metodologías con el objetivo de estructurar el proceso de desarrollo utilizándolas como un marco referencial para lograr una forma ordenada y predecible de trabajo en la cuál los resultados se podrán garantizar.

Es entonces cuando resulta muy común preguntarse qué metodología utilizar. Cada proceso lleva en sí una forma lógica a seguir si se quiere obtener los resultados que se esperan, el proceso de desarrollo de software en proyectos específicos es muy complicado y difícil de controlar, es entonces cuando se debe elegir qué método utilizar para lograr el objetivo. Sin embargo muchas veces esta técnica no es aplicada de la mejor forma. (SÁNCHEZ 2004)

Las empresas que desarrollan software deben tener presente que el modelo que cada una adopte tiene implicaciones relevantes en la eficiencia general del proceso de producción del software, donde precisamente, la producción del software es un entorno sistémico. Donde los componentes deben funcionar armónicamente, alineados con las características, cultura y estrategia de la organización, para maximizar la homogeneidad y calidad de resultados.

El problema que pueden encontrar quienes deciden implantar métodos más eficientes es caer en la desorientación ante el abanico de modelos de calidad, de procesos y de técnicas de trabajo desplegado en la última década (Modelo de Capacidad de Madurez -CMM-, ISO 15504, Programación Extrema, Scrum), o escoger cualquier alternativa que se presente (sin tener en cuenta las necesidades) como “solución” de eficiencia y calidad. (PALACIO 2005)

Durante los últimos años se ha desarrollado el debate en relación con las metodologías de desarrollo de software. Las opiniones se encuentran polarizadas entre las metodologías ágiles y las tradicionales.

Los procesos de desarrollo de software están mayormente guiados por metodologías tradicionales donde muchas veces debido a su gran planificación y rigidez resulta muy engorrosa su aplicación en proyectos donde se necesita obtener rápidamente un producto. Para resolver esta situación se presentan las metodologías ágiles con una forma más flexible de desarrollo.

Las metodologías ágiles están revolucionando la manera de producir software. Muchas de las empresas productoras de software están encaminando su proceso de software bajo estas metodologías debido a las facilidades que presentan. Estas constituyen una solución a medida para muchos proyectos de software, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto. (LETELIER and PENADÉS 2006)

Debido a esta vigente variabilidad de metodologías de desarrollo con que ahora se cuenta se está más propenso a incurrir en la incertidumbre respecto a cuál de estos estándares utilizar para organizar el proceso de producción de software. Al ser el uso de las prácticas ágiles la nueva tendencia que se está adoptando ampliamente en el mundo del software se hace conveniente obtener una guía a través de la cual el equipo de desarrollo sea capaz de reconocer con facilidad cuándo asumir una postura agilista frente al proceso de desarrollo de software y cómo llevarla a cabo o cuándo mantener las formas más tradicionales de desarrollo; todo de acuerdo a las necesidades de ambas partes (cliente y desarrollador).

La presente investigación está centrada en la obtención de una referencia para la aplicación de la metodología ágil Programación Extrema a proyectos de desarrollo de software en la empresa Desoft SS.

La empresa Desoft de la provincia SS es una entidad creada con el fin de suministrar soluciones para determinadas instituciones de la provincia SS a través de la producción de software correspondiente a las necesidades de los clientes. Los proyectos que se desarrollan en Desoft son mayormente sistemas de información para automatizar diferentes procesos en determinadas entidades de la provincia con el objetivo de brindar un mayor servicio tanto a la población como a la misma empresa en cuanto a la gestión de su información interna. Los miembros que conforman al equipo de desarrollo se encuentran entre 5 y 8 personas, los cuales todos son graduados desde hace aproximadamente entre 5 y 10 años de las Ingeniería Informática, Cibernética y con categorías de Master. Dicha empresa rige su proceso de desarrollo bajo la metodología tradicional RUP (Rational Unified Process - Proceso Racional Unificado) la

cual ha sido utilizada desde hace algunos años debido a que esta corresponde con los requerimientos de la empresa y a través de la cual se suministra al cliente un producto que satisface sus necesidades.

Actualmente en la empresa se denota la existencia de una gran demanda de servicios que prestar por parte de la empresa por lo que se hace necesario acelerar el proceso de producción manteniendo la calidad del producto final. En base a esto se tienen una serie de inconvenientes que se derivan de la metodología que se utiliza. Uno de estos aspectos es que la documentación que se genera es extensa por lo que consume gran parte del tiempo de desarrollo, también se tiene que la planificación que se hace es muy estricta y controlada por lo que no permite ajustar diferentes aspectos con el objetivo de acortar el proceso de la producción, además en relación con la planificación, los cambios que se pueden producir sobre el proyecto pueden implicar retraso y a raíz de este un alto costo

Como consecuencia de lo anteriormente planteado se genera la siguiente **situación problemática**: La empresa Desoft adopta la posición de un cambio de estrategia a través de la implantación de una nueva metodología de desarrollo que le permita ajustarse de una forma más conveniente a las nuevas condiciones que presenta el proceso de desarrollo de la misma. Las metodologías ágiles con su forma flexible de desarrollo representan una nueva alternativa para el proceso de desarrollo de software que en consecuencia corresponde. Pero en relación a la agilidad la empresa no posee el conocimiento suficiente para determinar bajo qué condiciones y de qué forma aplicar este tipo de metodología.

A causa de esta situación se deriva el siguiente **problema científico**: ¿Cómo aplicar una metodología ágil en la empresa Desoft para hacer más eficiente el proceso de desarrollo de software?

Tenemos entonces que:

Se definió para la realización de la investigación como **objetivo general** desarrollar una referencia que viabilice y facilite el proceso de aplicación de la metodología ágil Programación Extrema a proyectos de desarrollo de software en la empresa Desoft SS.

Los **objetivos específicos** que se derivan para dar cumplimiento al objetivo general son los siguientes:

1. Diagnosticar la situación actual para el uso de las metodologías ágiles en proyectos de desarrollo de software.
2. Desarrollar una referencia para la aplicación de la metodología ágil Programación Extrema a proyectos de desarrollo de software en la empresa Desoft SS.

3. Probar la efectividad de dicha referencia en proyectos de desarrollo de software del entorno específico.

El **objeto de estudio** es: El proceso de desarrollo de software.

Campo de acción: Aplicación de la metodología ágil Programación Extrema en proyectos de desarrollo de software en la empresa Desoft SS.

La **hipótesis** que se plantea es la siguiente:

Si se desarrolla una referencia para la aplicación de la metodología ágil Programación Extrema a proyectos de desarrollo de software en la empresa Desoft SS entonces se logrará una mayor calidad y eficiencia en el proceso de producción de software en dicho centro.

Variables Independientes: Referencia utilizada como guía para la aplicación de la metodología ágil Programación Extrema a proyectos de desarrollo de software en la empresa Desoft SS.

Variables dependientes: Calidad y productividad en el proceso de desarrollo de software.

Tareas de la investigación

1. Estudio del estado de arte de las metodologías ágiles más utilizadas en el desarrollo de software (surgimiento, evolución, desarrollo, aplicación, ventajas, deficiencias, resultados, etc.).
2. Aplicación de encuestas que contribuyan a determinar la posibilidad de llevar a cabo la utilización de las prácticas ágiles en proyectos de desarrollo de software.
3. Justificación de la metodología ágil escogida como base para la referencia a desarrollar.
4. Elaboración de un esquema que represente la dinámica de cómo debe llevar a cabo la aplicación de una metodología ágil como parte de la referencia a desarrollar.
5. Explicación detallada de cada uno de los pasos con que consta el esquema de referencia.
6. Aplicación de la referencia desarrollada en un proyecto de la empresa Desoft SS.
7. Obtención de resultados de acuerdo a la referencia aplicada.
8. Analizar los resultados de la aplicación de esta referencia a proyectos de desarrollo de software en la empresa Desoft SS.

El siguiente trabajo constará para su estructuración de 3 capítulos. A continuación una breve reseña de lo que se tratará en cada uno de los capítulos correspondientes.

Capitulo 1: En este capítulo se hace un estudio de la evolución y desarrollo de los diferentes tipos de metodologías ágiles y su relación con las metodologías tradicionales en el proceso de desarrollo de software. También se hace una evaluación de la empresa Desoft en cuanto a la situación existente como base para la propuesta de solución a desarrollar.

Capitulo 2: Se construye la referencia propuesta como solución a través de un esquema que es detalladamente descrito mediante varias fases que reflejan como queda estructurado el proceso de desarrollo de software bajo el uso de la metodología ágil Programación Extrema.

Capítulo 3: En este capítulo se hace un análisis de los resultados obtenidos a través de la aplicación de la referencia desarrollada como propuesta.

CAPITULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se hace una revisión bibliográfica acerca del surgimiento y desarrollo de los nuevos métodos de desarrollo de software, refiriéndonos así a las metodologías ágiles de desarrollo, haciendo una ejemplificación de los métodos más comunes, exponiendo las ventajas que estos presentan sobre los métodos tradicionales y su marco de aplicación dentro del proceso de desarrollo de software.

Para esto se debe comenzar por plantear una definición de Metodologías Ágiles, en lo adelante MAs.

1.1 Metodologías ágiles

Las Metodologías Ágiles o "ligeras" constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración. (GALLO and VERGARA 2006)

Tras toda esta nueva tendencia que estaba surgiendo en cuanto a los procesos de desarrollo de software, en febrero del 2001 se reúnen en Utah- EE.UU. diecisiete críticos de los modelos de mejora del desarrollo de software basados en procesos convocados por Kent Beck. Todo este acontecimiento organizado con el objetivo de trabajar sobre nuevas técnicas y procesos para el desarrollo de software que dieran solución a los problemas vigentes proponiendo una alternativa a los métodos tradicionales caracterizados como rígidos y robustos por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo. Es entonces que se asume el término "Métodos Ágiles" aplicado al desarrollo del software.(LETELIER and PENADÉS 2006)

Los integrantes de la reunión bajo el nombre de "Alianza Ágil" (organización dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos) resumieron los principios sobre los que se basan los métodos alternativos en cuatro postulados, lo que ha quedado denominado como "**Manifiesto Ágil"** (BECK 2001)**,** el cual expone que se ha llegado a valorar:

- A los individuos y su interacción, por encima de los procesos y las herramientas.
- El software que funciona, por encima de la documentación exhaustiva.
- La colaboración con el cliente, por encima de la negociación contractual.

- La respuesta al cambio, por encima del seguimiento de un plan

Aunque hay valor en los elementos de la derecha, en las metodologías ágiles se valoran más los de la izquierda.

Los valores anteriores inspiran los doce principios del manifiesto. Estos principios son las características que diferencian un proceso ágil de uno tradicional.(BECK 2001)

1. Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.
4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
7. El software que funciona es la principal medida del progreso.
8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica enaltece la agilidad.
10. La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan.
12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

Lo que las MAs tienen en común es su modelo de desarrollo incremental (pequeñas entregas con ciclos rápidos), cooperativo (desarrolladores y usuarios trabajan juntos en estrecha comunicación), directo (el método es simple y fácil de aprender) y adaptativo (capaz de incorporar los cambios). Las claves de las

MAs son la velocidad y la simplicidad. De acuerdo con ello, los equipos de trabajo se concentran en obtener lo antes posible una pieza útil que implemente sólo lo que sea más urgente; de inmediato requieren feedback de lo que han hecho y lo tienen muy en cuenta. Luego prosiguen con ciclos igualmente breves, desarrollando de manera incremental. Estructuralmente, las MAs se asemejan a los RADs (Desarrollo Rápido de Aplicaciones) más clásicos y a otros modelos iterativos, pero sus énfasis son distintivos y su combinación de ideas es única.(REYNOSO 2004)

Estas metodologías ágiles están encaminadas a reducir el tiempo de desarrollo conservando la alta calidad del producto como requisito indispensable para obtener satisfacción tanto por parte del cliente como del equipo de desarrollo y así un buen producto de software.

1.2 Metodologías Ágiles vs. Metodologías Tradicionales

En este epígrafe se realiza una comparación entre los métodos ágiles y los tradicionales, resumiéndose los aspectos más importantes de dicha comparación en la tabla 1.1.

Los métodos ágiles cambian significativamente algunos de los énfasis de los métodos ingenieriles. La diferencia inmediata es que son menos orientados al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada. En muchas maneras son más bien orientados al código: siguiendo un camino que dice que la parte importante de la documentación es el código fuente.(FOWLER 2003)

Los métodos ágiles son adaptables en lugar de predictivos. Los métodos ingenieriles tienden a intentar planear una parte grande del proceso del software en gran detalle para un plazo grande de tiempo, esto funciona bien hasta que las cosas cambian. Así que su naturaleza es resistirse al cambio. Para los métodos ágiles, no obstante, el grado de adaptabilidad es muy alto donde el cambio es bienvenido. Intentan ser procesos que se adaptan y crecen en el cambio, incluso al punto de cambiarse ellos mismos.(FOWLER 2003)

Para muchos clientes esta flexibilidad será una ventaja competitiva y porque estar preparados para el cambio significa reducir su coste. Estas metodologías nos llevan a retrasar las decisiones tanto como sea posible (de una manera responsable), porque será una ventaja tanto para la empresa de desarrollo como para el cliente teniendo en cuenta que el éxito de una empresa de desarrollo está muy relacionado con el éxito de sus clientes.(SÁNCHEZ 2004)

Los métodos ágiles son orientados a la gente y no orientados al proceso. La meta de los métodos ingenieriles es definir un proceso que funcionará bien con cualquiera que lo use. Los métodos ágiles

afirman que ningún proceso podrá nunca maquillar las habilidades del equipo de desarrollo, de modo que el papel del proceso es apoyar al equipo de desarrollo en su trabajo. Explícitamente puntualizan el trabajar a favor de la naturaleza humana en lugar de en su contra y enfatizan que el desarrollo de software debe ser una actividad agradable**.**(FOWLER 2003)

Metodologías Tradicionales	**Metodologías Ágiles**
Más Roles, más específicos	Pocos Roles, más genéricos y flexibles
Más Artefactos. El modelado es esencial, mantenimiento de modelos	Pocos Artefactos. El modelado es prescindible, modelos desechables.
Existe un contrato prefijado	No existe un contrato tradicional o al menos es bastante flexible
La arquitectura es esencial: Se promueve que la arquitectura se defina tempranamente en el proyecto	Menos énfasis en la arquitectura: La arquitectura se va definiendo y mejorando a lo largo del proyecto
Cierta resistencia a los cambios (estos pueden provocar grandes costos)	Sujeta a cambios en cualquier etapa del proyecto
Dirigidas al proceso: roles, actividades y artefactos	Dirigidas a las personas: el individuo y el trabajo en equipo
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos	Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños y trabajando en el mismo sitio
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo

Tabla1.1 Resumen de la Comparación entre MAs y las tradicionales.

Luego de hacer una definición de las MAs y compararlas con las metodologías tradicionales se hace necesario tratar de manera separada cada tipo de metodología ágil.

1.3 Diferentes tipos de MAs

Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios, cada metodología tiene características propias y hace énfasis en algunos aspectos más específicos.

En este epígrafe se hace una descripción de diferentes tipos de MAs y al final se brinda una comparación entre las mismas.

1.3.1 Programación Extrema (Extreme Programming-XP-)

Las raíces de la XP yacen en la comunidad de Smalltalk, y en particular de la colaboración cercana de Kent Beck y Ward Cunningham a finales de los 1980s. Ambos refinaron sus prácticas en numerosos proyectos a principios de los 90s, extendiendo sus ideas de un desarrollo de software adaptable y orientado a la gente. El paso crucial de la práctica informal a una metodología ocurrió en la primavera de 1996 cuando Kent Beck recomendó derribar la base del código el proyecto de nómina C3 para Chrysler en su totalidad y empezar desde el principio debido a la baja calidad de la base del código. El proyecto entonces reinició bajo su dirección y subsecuentemente se volvió el buque insignia temprano y el campo de entrenamiento de la XP. (FOWLER 2003)

La Programación Extrema es una metodología de desarrollo ligera (o ágil) basada en una serie de valores y de prácticas que persigue aumentar la productividad a la hora de desarrollar software basado en un método donde los desarrollos sean más sencillos.

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Se han seleccionado aquellas prácticas que han considerado mejores y han profundizado en sus relaciones y en cómo se refuerzan unas con otras. El resultado de esta selección ha sido esta

metodología única y compacta. Aunque no está basada en nuevos principios, sí que el resultado es una nueva manera de ver el desarrollo de software. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre.(BECK 1999)

La Programación Extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. XP considera que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Los objetivos de XP son muy simples y precisos: la satisfacción del cliente(dar al cliente el software que él necesita y cuando lo necesita) y potenciar al máximo el trabajo en grupo (tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software).(SOLÍS. 2003)

XP define cuatro variables para proyectos de software: coste, tiempo, calidad y alcance. Sólo tres de estas variables pueden ser establecidas por las fuerzas externas (jefes de proyecto y clientes), mientras que el valor de la cuarta variable debe ser establecido por los programadores en función de las otras tres. XP propone que se verifiquen todas las partes implicadas en el proyecto hasta que el valor que alcancen las cuatro variables sea el correcto para todas las partes. Estas cuatro variables no guardan una relación tan directa como en principio pueda parecer. El incremento del número de programadores no repercutirá de manera lineal en el tiempo de desarrollo del proyecto.(SOLÍS. 2003)

El coste del cambio en el desarrollo de un proyecto se incrementaba exponencialmente en el tiempo. Lo que XP propugna es que esta curva ha perdido validez y que con una combinación de buenas prácticas de programación y tecnología es posible lograr que la curva sea la contraria. Si se decide emplear XP como proceso de desarrollo de software, se debe adoptar basándose en dicha curva. La idea fundamental aquí es que, en vez de diseñar para el cambio, diseñaremos tan sencillo como sea posible, para hacer sólo lo que sea imprescindible en un momento dado, pues la propia simplicidad del código, junto con conocimientos de refactorización y, sobre todo, las pruebas y la integración continua, hacen posible que los cambios puedan ser llevados a cabo tan a menudo como sea necesario.(ACEBAL and LOVELLE 2002)

La Programación Extrema, lejos de ser un proceso incontrolado, es una metodología muy disciplinada y que se apoya en cuatro valores fundamentales: Comunicación, Simplicidad, Retroalimentación, Coraje(BERRUETA 2006).

Un proyecto XP tiene éxito cuando el cliente selecciona el valor de negocio a implementar basado en la habilidad del equipo para medir la funcionalidad que puede entregar a través del tiempo.

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos(JEFFRIES R *et al.* 2001) (LETELIER and PENADÉS 2006):

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. (BECK 1999)

Las prácticas de la Programación Extrema (actividades que el equipo de un proyecto lleva a cabo cada día) tienen su origen en prácticas bien conocidas en la ingeniería del software y en el sentido común, y por tanto, no pueden resultar extrañas. Sin embargo, lo que caracteriza a este conjunto es la cohesión de todos los elementos, y que cada práctica ha sido llevada al extremo.

1.3.2 Scrum

Scrum fue aplicado por Jeff Sutherland y elaborado más formalizadamente por Ken Schwaber. Poco después Sutherland y Schwaber se unieron para refinar y extender Scrum (REYNOSO 2004).

SCRUM es una forma de gestionar proyectos de software. No es una metodología de análisis, ni de diseño, como podría ser RUP, es una metodología de gestión del trabajo. Una de las características más importantes es que es muy fácil de explicar y de entender, lo que ayuda mucho a su implantación. Por otra parte SCRUM puede ser aplicado a distintos modelos de calidad (como podría ser CMMI) puesto que estos definen qué se tiene que hacer, es decir, definen que se tiene que gestionar el proyecto, pero no definen cómo. Ahí es donde entra SCRUM como modelo de gestión del proyecto.(GRACIA 2006)

Aunque surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados sistemas de software.

Scrum no está concebido como método independiente, sino que se promueve como complemento de otras metodologías, incluyendo XP, MSF o RUP. Como método, Scrum enfatiza valores y prácticas de gestión, sin pronunciarse sobre requerimientos, implementación y demás cuestiones técnicas; de allí su deliberada insuficiencia y su complementariedad. Scrum se define como un proceso de gestión y control que implementa técnicas de control de procesos; se lo puede considerar un conjunto de patrones organizacionales.

La dimensión del equipo total de Scrum no debería ser superior a diez ingenieros. El número ideal es siete, más o menos dos, una cifra canónica en ciencia cognitiva. Si hay más, lo más recomendable es formar varios equipos. No hay una técnica oficial para coordinar equipos múltiples, pero se han documentado experiencias de hasta 800 miembros, divididos en Scrums de Scrum, definiendo un equipo central que se encarga de la coordinación, las pruebas cruzadas y la rotación de los miembros.(HIGHSMITH 2002)

El ciclo de vida de Scrum es el siguiente:

1. Pre-Juego: 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 (backlog) del producto inicial y los ítems estimados, así como la arquitectura de alto nivel, el diseño exploratorio y los prototipos. El registro de acumulación es de alto nivel de abstracción.
2. Pre-Juego: Montaje (Staging). El propósito es identificar más requerimientos y priorizar las tareas para la primera iteración. Las actividades son planificación, diseño exploratorio y prototipos.
3. Juego o Desarrollo: El propósito es implementar un sistema listo para entregar en una serie de iteraciones de treinta días llamadas “corridas” (sprints). 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 de Scrum.
4. Pos-Juego: Liberación. El propósito es el despliegue operacional. Las actividades, documentación, entrenamiento, mercadeo y venta.

Algunos textos sobre Scrum establecen una arquitectura global en la fase de pre-juego; otros dicen que no hay una arquitectura global en Scrum, sino que la arquitectura y el diseño emanan de múltiples corridas (SCHWABER and BEEDLE 2002). No hay una ingeniería del software prescripta para Scrum; cada quien puede escoger entonces las prácticas de automatización, inspección de código, prueba unitaria, análisis o programación en pares que le resulten adecuadas.

Es habitual que Scrum se complemente con XP; en estos casos, Scrum suministra un marco de gestión basado en patrones organizacionales, mientras XP constituye la práctica de programación, usualmente orientada a objetos y con fuerte uso de patrones de diseño. Uno de los nombres que se utiliza para esta alianza es XP@Scrum. También son viables los híbridos con otros MAs (REYNOSO 2004)

1.3.3 Evolutionary Project Management (Evo)

Evo, creado por Tom Gilb, es el método iterativo ágil más antiguo. En las breves iteraciones de Evo, se efectúa un progreso hacia las máximas prioridades definidas por el cliente, liberando algunas piezas útiles para algunos participantes y solicitando su retroalimentación (feedback). Esta es la práctica que se ha llamado Planeamiento Adaptativo Orientado al Cliente y Entrega Evolutiva. Otra idea distintiva de Evo es la clara definición, cuantificación, estimación y medida de los requerimientos que necesitan mejoras. En Evo se espera que cada iteración constituya una re-evaluación de las soluciones en procura de la más alta relación de valor contra costo, teniendo en cuenta tanto la retroalimentación como un amplio conjunto de estimaciones métricas.

Evo requiere, igual que otras MAs, activa participación de los clientes. A diferencia de otras MAs como Agile Modeling (AM), donde la metodología es meticulosa pero discursiva, en Evo hay una especificación semántica y una pragmática rigurosa, completamente alejadas del sentido común, pero con la fundamentación que les presta derivarse de prácticas productivas suficientemente probadas.

La esencia de Evo se recoge en los siguientes principios:

1. Se entregarán temprano y con frecuencia resultados verdaderos, de valor para los participantes reales.
2. El siguiente paso de entrega de Evo será el que proporcione el mayor valor para el participante en ese momento.
3. Los pasos de Evo entregan los requerimientos especificados de manera evolutiva.

4. No se puede saber cuáles son los requerimientos por anticipado, pero pueden descubrirse más rápidamente intentando proporcionar valor real a participantes reales.
5. Evo es ingeniería de sistemas holística (todos los aspectos necesarios del sistema deben ser completos y correctos) y con entrega a un ambiente de participantes reales (no es sólo sobre programación; es sobre satisfacción del cliente).
6. Los proyectos de Evo requieren una arquitectura abierta, porque se habrá de cambiar las ideas del proyecto tan a menudo como se necesite hacerlo, para entregar realmente valor a nuestros participantes.
7. El equipo de proyecto de Evo concentrará su energía como equipo hacia el éxito del paso actual. En este paso tendrán éxito o fracasarán todos juntos. No se gastará energías en pasos futuros hasta que hayan dominado los pasos actuales satisfactoriamente.
8. Evo tiene que ver con aprendizaje a partir de la dura experiencia, tan rápido como se pueda: qué es lo que verdaderamente funciona, qué es lo que realmente entrega valor. Evo es una disciplina que hace confrontar los problemas tempranamente, pero que permite progresar rápido cuando probadamente se ha hecho bien.
9. Evo conduce a una entrega temprana, a tiempo, tanto porque se lo ha priorizado así desde el inicio, y porque aprendemos desde el principio a hacer las cosas bien.
10. Evo debería permitir poner a prueba nuevos procesos de trabajo y deshacerse tempranamente de los que funcionan mal.

A diferencia de otras MAs que son más experimentales y que no tienen mucho respaldo de casos sistemáticamente documentados, Evo es una metodología probada desde hace mucho tiempo en numerosos clientes corporativos: la NASA, Lockheed Martin, Hewlett Packard, Douglas Aircraft, la Marina británica. (REYNOSO 2004)

1.3.5 Crystal Methods

Las metodologías Crystal fueron creadas por el “antropólogo de proyectos” Alistair Cockburn, el autor que ha escrito los que tal vez sean los textos más utilizados, influyentes y recomendables sobre casos de uso.

La familia Crystal dispone un código de color para marcar la complejidad de una metodología: cuanto más oscuro un color, más “pesado” es el método. Cuanto más crítico es un sistema, más rigor se requiere. El código cromático se aplica a una forma tabular elaborada por Cockburn que se usa en muchos MAs para situar el rango de complejidad al cual se aplica una metodología.

Para la evaluación de las pérdidas que puede ocasionar la falla de un sistema y el método requerido según este criterio se tiene los siguientes parámetros: Comodidad (C), Dinero Discrecional (D), Dinero Esencial (E) y Vidas (L).

Los métodos se llaman Crystal evocando las facetas de una gema: cada faceta es otra versión del proceso, y todas se sitúan en torno a un núcleo idéntico. Hay cuatro variantes de metodologías: Crystal Clear (“Claro como el cristal”) para equipos de 8 o menos integrantes; Amarillo, para 8 a 20; Naranja, para 20 a 50; Rojo, para 50 a 100. Se promete seguir con Marrón, Azul y Violeta.

Crystal Clear (CC) puede ser usado en proyectos pequeños de categoría D6, aunque con alguna extensión se aplica también a niveles E8 a D10.

El otro método elaborado en profundidad es el Naranja, apto para proyectos de duración estimada en 2 años.

Los siete valores o propiedades de CC son:

1. Entrega frecuente.
2. Comunicación osmótica
3. Mejora reflexiva.
4. Seguridad personal.
5. Foco.
6. Fácil acceso a usuarios expertos.
7. Ambiente técnico con prueba automatizada, gestión de configuración e integración frecuente. (COCKBURN 2004)

Los métodos Crystal no prescriben las prácticas de desarrollo, las herramientas o los productos que pueden usarse, pudiendo combinarse con otros métodos como Scrum, XP y Microsoft Solutions Framework (MSF). En su comentario a (HIGHSMITH 2000), Cockburn expresa en que cuando imaginó CC pensaba proporcionar un método ligero; comparado con XP, sin embargo, CC resulta muy pesado. CC es más fácil de aprender e implementar; a pesar de algunos aspectos controversiales XP es más disciplinado; pero si un equipo ligero puede tolerar sus rigores, lo mejor será implementar XP.

1.3.6 Feature Driven Development (FDD)

Feature Oriented Programming (FOP) es una técnica de programación guiada por rasgos o características (features) y centrada en el usuario, no en el programador; su objetivo es sintetizar un programa conforme a los rasgos requeridos.

FDD, en cambio, es un método ágil, iterativo y adaptativo. A diferencia de otros MAs, no cubre todo el ciclo de vida sino sólo las fases de diseño y construcción y se considera adecuado para proyectos mayores y de misión crítica. Es un método de desarrollo de ciclos cortos que se concentra en la fase de diseño y construcción. En la primera fase, el modelo global de dominio es elaborado por expertos del dominio y desarrolladores; el modelo de dominio consiste en diagramas de clases con clases, relaciones, métodos y atributos. Los métodos no reflejan conveniencias de programación sino rasgos funcionales. Aunque hay coincidencias entre la programación orientada por rasgos y el desarrollo guiado por rasgos, FDD no necesariamente implementa FOP.

FDD no requiere un modelo específico de proceso y se complementa con otras metodologías. Enfatiza cuestiones de calidad y define claramente entregas tangibles y formas de evaluación del progreso.(REYNOSO 2004)

Los principios de FDD son pocos y simples:

1. Se requiere un sistema para construir sistemas si se pretende escalar a proyectos grandes.
2. Un proceso simple y bien definido trabaja mejor.
3. Los pasos de un proceso deben ser lógicos y su mérito inmediatamente obvio para cada miembro del equipo.
4. Vanagloriarse del proceso puede impedir el trabajo real.
5. Los buenos procesos van hasta el fondo del asunto, de modo que los miembros del equipo se puedan concentrar en los resultados.
6. Los ciclos cortos, iterativos, orientados por rasgos son mejores.

FDD consiste en cinco procesos secuenciales durante los cuales se diseña y construye el sistema (Desarrollar un Modelo Global, Construir una Lista de los Rasgos, Planear por Rasgo, Diseñar por Rasgo, Construir por Rasgo). La parte iterativa soporta desarrollo ágil con rápidas adaptaciones a cambios en requerimientos y necesidades del negocio. Cada fase del proceso tiene un criterio de entrada, tareas,

pruebas y un criterio de salida. Típicamente, la iteración de un rasgo insume de una a tres semanas. FDD suministra un rico conjunto de artefactos para la planificación y control de los proyectos.

Algunos agilistas sienten que FDD es demasiado jerárquico para ser un método ágil, porque demanda un programador jefe, quien dirige a los propietarios de clases, quienes dirigen equipos de rasgos. Otros críticos sienten que la ausencia de procedimientos detallados de prueba en FDD es llamativa e impropia. Los promotores del método aducen que las empresas ya tienen implementadas sus herramientas de prueba, pero subsiste el problema de su adecuación a FDD. Un rasgo llamativo de FDD es que no exige la presencia del cliente.(REYNOSO 2004)

Se sugiere su uso para proyectos nuevos o actualizaciones de sistemas existentes, y recomiendan adoptarlo en forma gradual.

1.3.7 Dynamic Systems Development Method (DSDM)

Originado en los trabajos de Jennifer Stapleton, directora del DSDM Consortium, DSDM se ha convertido en el marco de trabajo de Desarrollo Rápido de Aplicaciones (RADs) más popular de Gran Bretaña y se ha llegado a promover como el estándar de facto para desarrollo de soluciones de negocios sujetas a márgenes de tiempo estrechos.

Además de un método, DSDM proporciona un marco de trabajo completo de controles para RAD y lineamientos para su uso. DSDM puede complementar metodologías de XP, RUP o Microsoft Solutions Framework, o combinaciones de todas ellas. DSDM es relativamente antiguo en el campo de los MAs y constituye una metodología madura. Se expresa que actualmente las iniciales DSDM significan Dynamic Solutions Delivery Method. Ya no se habla de sistemas sino de soluciones, y en lugar de priorizar el desarrollo se prefiere enfatizar la entrega.(STAPLETON 2003)

La idea dominante detrás de DSDM es explícitamente inversa a la que se encuentra en otras partes, y al principio resulta contraria a la intuición; en lugar de ajustar tiempo y recursos para lograr cada funcionalidad, en esta metodología tiempo y recursos se mantienen como constantes y se ajusta la funcionalidad de acuerdo con ello.

DSDM consiste en cinco fases:

1. Estudio de viabilidad.
2. Estudio del negocio.
3. Iteración del modelo funcional.

4. Iteración de diseño y versión.
5. Implementación.

Las últimas tres fases son iterativas e incrementales. De acuerdo con la iniciativa de mantener el tiempo constante, las iteraciones de DSDM son cajas de tiempo. La iteración acaba cuando el tiempo se consume. Se supone que al cabo de la iteración los resultados están garantizados. Una caja de tiempo puede durar de unos pocos días a unas pocas semanas.(REYNOSO 2004)

A diferencia de otros MAs, DSDM ha desarrollado sistemáticamente el problema de su propia implantación en una empresa. El proceso de Examen de Salud (Health Check) de DSDM se divide en dos partes que se interrogan, sucesivamente, sobre la capacidad de una organización para adoptar el método y sobre la forma en que éste responde a las necesidades una vez que el proyecto está encaminado. .

Desde mediados de la década de 1990 hay abundantes estudios de casos, sobre todo en Gran Bretaña, y la adecuación de DSDM para desarrollo rápido está suficientemente probada(PEKKA ABRAHAMSSON 2002) . El equipo mínimo de DSDM es de dos personas y puede llegar a seis, pero puede haber varios equipos en un proyecto. El mínimo de dos personas involucra que un equipo consiste de un programador y un usuario. El máximo de seis es el valor que se encuentra en la práctica. DSDM se ha aplicado a proyectos grandes y pequeños. La precondición para su uso en sistemas grandes es su partición en componentes que pueden ser desarrollados por equipos normales.(REYNOSO 2004)

También hay documentos conjuntos con participación de Jennifer Stapleton, que demuestran la compatibilidad del modelo DSDM con RUP, a despecho de sus fuertes diferencias terminológicas.

Se ha elaborado en particular la combinación de DSDM con XP y se ha llamado a esta mixtura EnterpriseXP, término acuñado por Mike Griffiths.

1.3.8 Adaptive Software Development (ASD)

James Highsmith III, consultor de Cutter Consortium, desarrolló ASD hacia el año 2000 con la intención primaria de ofrecer una alternativa a la idea, propia de CMM Nivel 5, de que la optimización es la única solución para problemas de complejidad creciente.

La estrategia entera se basa en el concepto de emergencia, una propiedad de los sistemas adaptativos complejos que describe la forma en que la interacción de las partes genera una propiedad que no puede ser explicada en función de los componentes individuales.(HOLLAND 1995)

ASD presupone que las necesidades del cliente son siempre cambiantes. La iniciación de un proyecto involucra definir una misión para él, determinar las características y las fechas y descomponer el proyecto en una serie de pasos individuales, cada uno de los cuales puede abarcar entre cuatro y ocho semanas. Los pasos iniciales deben verificar el alcance del proyecto; los tardíos tienen que ver con el diseño de una arquitectura, la construcción del código, la ejecución de las pruebas finales y el despliegue.

Aspectos claves de ASD son:

1. Un conjunto no estándar de “artefactos de misión”, incluyendo una visión del proyecto, una hoja de datos, un perfil de misión del producto y un esquema de su especificación
2. Un ciclo de vida, inherentemente iterativo. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje.
3. Cajas de tiempo, con ciclos cortos de entrega orientados por riesgo.

Un ciclo de vida es una iteración; este ciclo se basa en componentes y no en tareas, es limitado en el tiempo, orientado por riesgos y tolerante al cambio. Que se base en componentes implica concentrarse en el desarrollo de software que trabaje, construyendo el sistema pieza por pieza. En este paradigma, el cambio es bienvenido y necesario, pues se concibe como la oportunidad de aprender y ganar así una ventaja competitiva; de ningún modo es algo que pueda ir en detrimento del proceso y sus resultados.

La idea subyacente a ASD (y de ahí su particularidad) radica en que no proporciona un método para el desarrollo de software sino que más bien suministra la forma de implementar una cultura adaptativa en la empresa, con capacidad para reconocer que la incertidumbre y el cambio son el estado natural. El problema inicial es que la empresa no sabe que no sabe, y por tal razón debe aprender.

ASD se concentra más en los componentes que en las tareas; en la práctica, esto se traduce en ocuparse más de la calidad que en los procesos usados para producir un resultado.

Hay ausencia de estudios de casos del método adaptativo, aunque las referencias literarias a sus principios son abundantes. Como ASD no constituye un método de ingeniería de ciclo de vida sino una visión cultural o una epistemología, no califica como marco de trabajo suficiente para articular un proyecto.(REYNOSO 2004)

Entre las empresas que han requerido consultoría adaptativa se cuentan AS Bank de Nueva Zelanda, CNET, GlaxoSmithKline, Landmark, Nextel, Nike, Phoenix International Health, Thoughworks y Microsoft.

1.3.9 Agile Modeling (AM)

Agile Modeling (AM) fue propuesto por Scott Ambler no tanto como un método ágil cerrado en sí mismo, sino como complemento de otras metodologías, sean éstas ágiles o convencionales. Ambler recomienda su uso con XP, Microsoft Solutions Framework (MSF), RUP. En el caso de XP y MSF los practicantes podrían definir mejor los procesos de modelado que en ellos faltan, y en el caso de RUP el modelado ágil permite hacer más ligeros los procesos que ya usan. AM es una estrategia de modelado (de clases, de datos, de procesos) pensada para contrarrestar la sospecha de que los métodos ágiles no modelan y no documentan. Se lo podría definir como un proceso de software basado en prácticas cuyo objetivo es orientar el modelado de una manera efectiva y ágil.

Los principales objetivos de AM son:

1. Definir y mostrar de qué manera se debe poner en práctica una colección de valores, principios y prácticas que conducen al modelado de peso ligero.
2. Enfrentar el problema de la aplicación de técnicas de modelado en procesos de desarrollo ágiles.
3. Enfrentar el problema de la aplicación de las técnicas de modelado independientemente del proceso de software que se utilice.

Como AM se debe usar como complemento de otras metodologías, nada se especifica sobre métodos de desarrollo, tamaño del equipo, roles, duración de iteraciones, trabajo distribuido y criticidad, todo lo cual dependerá del método que se utilice.(REYNOSO 2004)

1.3.10 Lean Development (LD) y Lean Software Development (LSD)

Lean Development (LD) es el método menos divulgado entre los reconocidamente importantes. La palabra “lean” significa magro, enjuto; en su sentido técnico apareció por primera vez en 1990 en el libro de James Womack “La Máquina que Cambió al Mundo.”

LD se inspira en valores centrados en estrategias de gestión (HIGHSMITH 2002):

1. Satisfacer al cliente es la máxima prioridad.
2. Proporcionar siempre el mejor valor por la inversión.
3. El éxito depende de la activa participación del cliente.
4. Cada proyecto LD es un esfuerzo de equipo.
5. Todo se puede cambiar.

6. Soluciones de dominio, no puntos.
7. Completar, no construir.
8. Una solución al 80% hoy, en vez de una al 100% mañana.
9. La necesidad determina la tecnología.
10. El crecimiento del producto es el incremento de sus prestaciones, no de su tamaño.
11. Nunca empujes LD más allá de sus límites.

Dado que LD es más una filosofía de gestión que un proceso de desarrollo no hay mucho que decir del tamaño del equipo, la duración de las iteraciones, los roles o la naturaleza de sus etapas. Últimamente LD ha evolucionado como Lean Software Development (LSD)

Otra preceptiva algo más amplia es la de Mary Poppendieck (POPPENDIECK 2001):

1. Eliminar basura – Entre la basura se cuentan diagramas y modelos que no agregan valor al producto.
2. Minimizar inventario – Igualmente, suprimir artefactos tales como documentos de requerimiento y diseño.
3. Maximizar el flujo – Utilizar desarrollo iterativo.
4. Solicitar demanda – Soportar requerimientos flexibles.
5. Otorgar poder a los trabajadores.
6. Satisfacer los requerimientos del cliente – Trabajar junto a él, permitiéndole cambiar de ideas.
7. Hacerlo bien la primera vez – Verificar temprano y refactorizar cuando sea preciso.
8. Abolir la optimización local – Alcance de gestión flexible.
9. Asociarse con quienes suministran – Evitar relaciones de adversidad.
10. Crear una cultura de mejora continua.

LD y LSD han sido pensados como complemento de otros métodos, y no como una metodología excluyente a implementar en la empresa. LD prefiere concentrarse en las premisas y modelos derivados de Lean Production, que hoy constituyen lo que se conoce como el canon de la Escuela de Negocios de Harvard. Para las técnicas concretas de programación, LD promueve el uso de otros MAs que sean consistentes con su visión, como XP o sobre todo Scrum.

Existen abundantes casos de éxito documentados empleando LD y LSD, la mayoría en Canadá. Algunos de ellos son los de Canadian Pacific Railway, Autodesk y PowerEx Corporation. Se ha aplicado prácticamente a todo el espectro de la industria.(REYNOSO 2004)

1.3.11 Microsoft Solutions Framework y los Métodos Ágiles.

Microsoft Solutions Framework es “un conjunto de principios, modelos, disciplinas, conceptos, lineamientos y prácticas probadas” elaborado por Microsoft. MSF se encuentra en estrecha comunión de principios con las metodologías ágiles, algunas de cuyas intuiciones y prácticas han sido anticipadas en las primeras versiones de su canon, hacia 1994. La documentación de MSF no deja sombra de dudas sobre esta concordancia.

Aunque MSF también se presta como marco para un desarrollo fuertemente documentado y a escala de misión crítica que requiere niveles más altos de estructura, como el que se articula en CMMI(Capability Maturity Model Integration), PMBOK o ISO9000, sus disciplinas de ningún modo admiten la validez (o sustentan la vigencia) de modelos no iterativos o no incrementales.

Muchos de los codificadores de MAs han procurado poner en claro sus vínculos con MSF, entendiendo que éste es una parte importante en la arena metodológica, y que en estas prácticas es común que disciplinas de alto nivel se articulen y fomenten con metodologías más precisas, e incluso con más de una de ellas. Por ejemplo, Agile Modeling es, en las palabras de Scott Ambler, un complemento adecuado a la disciplina de MSF, en la misma medida en que es también armónico con RUP o XP.

Hay herramientas para trabajar en términos a la vez conformes a MSF y a las prácticas ágiles, como csUnit para .NET, un verificador de regresión del tipo que exigen los procesos de XP, o NUnitAsp para Asp.NET.

En suma, muchos de los principios de los MAs son consistentes no sólo con los marcos teóricos, sino con las prácticas de Microsoft, las cuales vienen empleando ideas adaptativas, desarrollos incrementales y metodologías de agilidad (algunas de ellas radicales) desde hace mucho tiempo. Se puede hacer entonces diseño y programación ágil en ambientes Microsoft, o en conformidad con MSF (ya sea en modalidad corporativa como en DSDM, o en modo hacker como en XP), sin desnaturalizar a ninguna de las partes.(REYNOSO 2004)

Microsoft tiene el nuevo producto Visual Studio Team System, que se presenta como la solución para la optimización y mejora de los entornos de producción de software. Microsoft Solutions Framework no es

un método ágil ni un método ortodoxo. Es un marco de trabajo flexible, válido para ambas teorías. y Microsoft Studio Team System al trabajar con este marco es una herramienta apropiada para unos y otros.(NAVEGAPOLIS 2006a)

MSF Agile (Microsoft Solution Framework Agile) es la nueva propuesta de Microsoft en el mundo de procesos y prácticas ágiles de desarrollo de software. “Microsoft Agile” renueva al ya exitoso MSF V3.0, que es un marco de trabajo en cascada y espiral, implementando las mejores prácticas del mundo de desarrollo ágil de software.

MSF Agile se considera más una metodología que un marco de trabajo y tiene como principales características el ser altamente insistente, de planificación adaptable a los cambios y enfocado a las personas.

Uno de los aspectos más interesantes acerca de las metodologías MSF Agile y MSF CMMI, es el hecho de que ya vienen integradas a la plataforma de desarrollo de Microsoft Visual Studio Team System (VSTS). Por medio de VSTS es posible contar con una serie de plantillas y guías adaptadas a cada metodología y orientadas a los roles definidos en cada una. Al alinear tanto la herramienta de desarrollo de VSTS como los procesos ágiles de MSF Agile y MSF CMMI, Microsoft se asegura un puesto importante dentro de la escena mundial de las prácticas ágiles de desarrollo de software.

1.3.12 Comparación entre los Diferentes tipos de MAs

Los métodos que se han examinado anteriormente no son fáciles de comparar entre sí conforme a un pequeño conjunto de criterios. Algunos, como XP, han definido claramente sus procesos, mientras que otros, como Scrum, son bastante más difusos en ese sentido, limitándose a un conjunto de principios y valores. Lean Development también presenta más principios que prácticas; como la satisfacción del cliente, están menos articulados que en Scrum, por ejemplo. Ciertos MAs, como FDD, no cubren todos los pasos del ciclo de vida, sino unos pocos de ellos. Varios métodos dejan librada toda la ingeniería concreta a algún otro método sin especificar.

Por más que exista una homología estructural en su tratamiento del proceso, se diría que en un primer análisis hay dos rangos o conjuntos distintos de MAs en la escala de complejidad. Por un lado están los MAs declarativos y programáticos como XP, Scrum, LD, AM y RAD; y por el otro, las piezas mayores finamente elaboradas como Evo, DSDM y Crystal. En una posición intermedia estaría ASD, cuya naturaleza es muy peculiar. No se calificaría a RUP como metodología ágil en plenitud, sino más bien

como un conjunto enorme de herramientas y artefactos, al lado de unos (pocos) lineamientos de uso, que acaso están mejor articulados en AM que en la propia documentación nativa de RUP. Los recursos de RUP se pueden utilizar con mayor o menor conflicto en cualquier otra estrategia, sin necesidad de crear productos de trabajo nuevos una y otra vez. En cualquier caso, se debe investigar en qué casos los métodos admiten hibridación por ser semejantes y complementarios en un mismo plano, y en qué otros escenarios lo hacen porque se refieren a distintos niveles de abstracción.

También se podrían considerar combinaciones múltiples. Se propone que MSF se utilice como marco general, Planguage como lenguaje de especificación de requerimiento, Scrum (con sus patrones organizacionales) como método de gestión, XP (con patrones de diseño, programación guiada por pruebas y refactorización) como metodología de desarrollo, RUP como abastecedor de artefactos, ASD como cultura empresarial y CMM como metodología de evaluación de madurez.(REYNOSO 2004)

1.4 Tipos de Proyectos a los que se pueden aplicar MAs

La filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas muestran su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Las metodologías ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.(LETELIER and PENADÉS 2006)

Las características de los proyectos para los cuales las metodologías ágiles han sido especialmente pensadas se ajustan a un amplio rango de proyectos de desarrollo de software; aquellos en los cuales los equipos de desarrollo son pequeños, donde el proyecto no es crítico, con plazos reducidos, requisitos volátiles, la cultura de la organización es flexible, personal preparado, y/o basados en nuevas tecnologías.(REYNOSO 2004)

Los acercamientos adaptables son buenos cuando sus requisitos son inciertos o volátiles. Si se tienen requisitos estables, entonces no se está en posición de tener un plan estable y seguir un proceso planeado. En estas situaciones un proceso adaptable puede ser menos cómodo, pero será más eficaz. A menudo la barrera más grande aquí es el cliente. Es importante para el cliente entender que seguir un proceso predictivo cuando los requisitos cambian es arriesgado tanto para ellos como para el desarrollo (FOWLER 2003)

En cuanto al tamaño de que los equipo de desarrollo deben ser pequeños, en general (3 – 20), pero cada una de las metodologías ágiles define en sí el tamaño del equipo que esta asimila de acuerdo a sus necesidades y técnicas que implementa para su desarrollo y puesta en práctica.(LETELIER and PENADÉS 2006)

Especialmente en XP los equipos son considerados en un rango de 3-10 programadores según la funcionalidad del software a desarrollar(BECK 2000). Aunque siempre valorando la posibilidad de que si se quiere utilizar un equipo mayor es posible pero que seria necesario reestructurar un conjunto de sus prácticas para poder establecer un adecuado control sobre lo que se esta produciendo, pues al ser un grupo mayor de personas estas técnicas se tornan más complicada pues ellas están precisamente pensadas para su utilización en equipo pequeños. Es válido aclarar que el tener equipos mayores no se deduce mayor productividad ni rapidez, con equipos menores que trabajen desarrollando consecuentemente estas prácticas se han logrado resultados que con equipos mayores no pudieron ser.(COCKBURN 2006)

Una de los posibles conflictos de estas nuevas metodologías, en algunos casos, es cómo manejan equipos más grandes. Como muchas nuevas tendencias, tienden a ser usadas primero a escala pequeña antes que a gran escala, pero es que precisamente están orientadas a equipos pequeños. Es importante señalar que los equipos de software pueden reducirse en tamaño sin reducir su productividad total. (FOWLER 2003)

Cada uno de los datos que anteriormente se expresan, los cuales definen un marco general que se debe tener como base para aplicar metodologías ágiles se cuantifican a través del método que a continuación se describe para lograr una mayor certeza de la utilización de las mismas.

1.4.1 Método Estrella de Barry Boehm y Richard Turner.

Uno de los métodos más usados para determinar cuándo se está en condiciones de implantar agilidad a un proyecto de software es el Método de Estrella de Barry Boehm y Richard Turner. En este método se proponen 5 parámetros fundamentales que se deben tener en cuenta para aplicar metodologías ágiles. Ellos son: Nivel profesional del equipo de desarrollo, criticidad del proyecto, tamaño del equipo de desarrollo, cultura de la organización y el dinamismo de los requisitos del proyecto.

Este método recoge las características esenciales que se necesitan para aplicar MAs, proporcionando una forma muy sencilla y eficiente para estimar si es preciso o no aplicar metodologías ágiles. (Anexo 1 Método de Estrella de Barry Boehm y Richard Turner)

1.5 Empresas de software que utilizan metodologías ágiles vs. tradicionales

Existen muchas empresas productoras de software que utilizan metodologías tradicionales para organizar su proceso de producción pero también existen otras que poco a poco se van adentrando en el mundo de la agilidad y cada vez son más las que utilizan metodologías ágiles demostrando que estas nuevas estrategias de desarrollo se adaptan a muchas de las condiciones existentes y logran resolver muchos de las dificultades que actualmente se enfrenta en la producción de software.(NAVEGAPOLIS 2006b)

Diferentes empresas que utilizan cada uno de las metodologías antes mencionadas.(Anexo 2 Empresas que utilizan metodologías ágiles vs. empresas que utilizan metodologías tradicionales.)

1.5.1 Encuesta realizada en enero 2006 por CM Crossroads sobre el uso de metodologías ágiles

CM Crossroads (Comunidad de desarrolladores dirigida a la administración de la configuración y del ciclo de vida de la aplicación) realizó una encuesta sobre el uso de metodologías ágiles en el desarrollo de software, entre los lectores de Configuration Management Journal (Diario de Administración de Configuración) de la cual se obtuvo que la mayoría de los desarrolladores que la utilizan lo hacen para reducir el tiempo de desarrollo y que la de mayor aceptación es la Programación Extrema entre otros datos. (Anexo 3 Datos de encuesta realizada por CM Crossroads sobre el uso de metodologías ágiles.)

1.5.2 Encuesta realizada por Digital Focus sobre metodologías ágiles.

Digital Focus, empresa líder en el desarrollo y consultoría en torno a metodologías ágiles, publicó un informe con las conclusiones de la encuesta que realizó en la Agile 2005 Conference.

En esta encuesta participaron 136 personas de 128 organizaciones donde los resultados obtenidos muestran que la adopción de metodologías ágiles esta creciendo tanto en medianas como en grandes empresas, donde la habilidad para responder a los cambios impulsa el uso de las metodologías ágiles

siendo el conocimiento y la experiencia son las principales barreras, negocio y tecnología deben trabajar más estrechamente. (Anexo 4 Encuesta realizada por Digital Focus sobre metodologías ágiles.)

1.6 Caracterización de la empresa Desoft SS

DeSoft S.A., es una empresa estatal que trabaja para lograr informatizar la sociedad de manera acelerada, comercializando e implementando software y servicios que le permita a las entidades, organizar y gestionar adecuadamente sus recursos.

El perfil del negocio de la Corporación se orienta hacia la satisfacción de las necesidades del mercado a través de soluciones informáticas, es decir que su tecnología fundamental está en productos y servicios informáticos (software) que permitan a los empresarios, especialistas y trabajadores tener información para la toma de decisiones.

Misión

Brindar soluciones integrales eficaces en tecnologías de la información a las organizaciones, para contribuir eficientemente al desarrollo de la sociedad cubana.

Visión

Líderes en soluciones integrales en tecnología de la información, con reconocimiento en el mercado internacional.

Valores Internacionales

- Compromiso con la organización
- Trabajo en equipo
- Profesionalismo y ética
- Creatividad
- Proactividad
- Innovación

Las líneas de negocio que se desarrollan en estos momentos son:

- Sistemas de gestión empresarial
- Desarrollo de proyectos de software
- Sistemas de gestión de información
- Formación

- Implementación del sistema contable Versat Sarasola

Los servicios se orientan a todos los organismos y asociaciones nacionales de sectores empresariales, presupuestados y gubernamentales y a entidades extranjeras con representación en Cuba. Estos servicios son brindados a través de un total de 17 gerencias ubicadas en todas las provincias del país. Específicamente para la validación de la referencia se escogió la división de Sancti-Spíritus.

DeSoft Sancti Spíritus representa una estructura jerárquica que cuenta con 40 trabajadores, de ellos 24 están vinculados directamente a la actividad informática. Existe un Master en computación aplicada y otro en Ciencias de la Computación y con respecto a la docencia hay 8 que son profesores adjuntos del Centro de Superación Nacional de la Actividad Informática (CENSAI) y 3 que además son profesores adjuntos del Centro Universitario de Sancti-Spíritus (CUSS).

UNIDAD	CTDAD
GERENTE	1
ADM. RED	1
ESP. B GEST. CALIDAD	1
NEGOCIACIÓN	2
DESARROLLO	13
IMPLEMENTACION Y SERVICIOS	11
ECONOMIA	3
REC. HUMANOS	2
DPTO INTERNO	6

Tabla 1.2 Distribución del personal de la empresa Desoft SS

1.7 Evaluación de la empresa Desoft SS para aplicar MAs

Se hace una evaluación de la empresa Desoft con el objetivo de determinar en qué estado se encuentra la empresa para implantar metodologías ágiles a su proceso de desarrollo. Este proceso de evaluación es de gran ayuda y brinda información muy importante la cual es utilizada como base para las posibles transformaciones a implementar en dicha institución.

Para tener bien esclarecidos todos y cada uno de los aspectos que se han de tener en cuenta para conocer en qué condiciones se encuentra la empresa para aplicar agilidad es necesario realizar cada uno de los siguientes pasos:

- Recopilación de información: Para obtener esta información se aplican varias encuestas con el fin de determinar los aspectos fundamentales en los que la empresa presenta dificultades en su proceso de desarrollo de software. En este sentido, se desarrollaron dos encuestas que recogen una serie de parámetros a tener en cuenta. Ver los modelos de dichas encuestas en (Anexo 4 Encuesta realizada por Digital Focus sobre metodologías ágiles).
- Análisis de la información recopilada: Una vez que se recopila la información, la misma se debe organizar en un formato que la resuma y que facilite su análisis para la posterior toma de decisiones.

De los 40 trabajadores con los que cuenta la empresa Desoft SS, 24 de estos están vinculados a la actividad informática, 11 dedicados a la parte de implementación y 13 al desarrollo de software. Para aplicar las encuestas desarrolladas se tomó una muestra intencional de 13 personas. Dicha muestra es intencional debido a que se tuvo en cuenta encuestar sólo a aquellas personas que están vinculadas directamente con el proceso de producción de software el cual es el tema que realmente encierra la investigación y a partir del cual se puede obtener resultados que sustenten la misma.

Según los datos obtenidos de las encuestas realizadas se obtiene que un 84.62 % de las ocasiones los encuestados utilizan la metodología tradicional RUP, estándar de la empresa Desoft, haciendo ciertas adaptaciones en la misma para llevar a cabo el proceso de desarrollo de software principalmente estas adaptaciones debido a que alrededor de un 70 % expresa que la documentación es demasiada y un 30 % que se genera un poco más de la necesaria coincidiendo el 100 % de los encuestados que el principal es el documento de captura de los requisitos. Teniendo en cuenta además que más del 80 % de estos considera que el coste de los cambios para dicha metodología utilizada es alto. Todo lo cual refleja la situación existente en la que está basada dicha investigación y del cual se genera la situación problemática.

Al mismo tiempo se obtiene que:

Alrededor del 61.54 % de los encuestados manifiestan que no tienen ningún conocimiento en cuanto al tema de las metodologías ágiles y que sólo un 38.46 % ha escuchado hablar acerca de estas. Por tanto se muestra la necesidad del desarrollo de la solución propuesta a través de la creación de la referencia para la aplicación de la metodología ágil Programación Extrema.

De manera general:

La cantidad de personas que están vinculadas directamente con la producción del software formando parte del equipo de desarrollo se encuentra en un rango de 5 a 20 personas considerando el 53.85 % que mayormente ha trabajado en equipos de cómo máximo hasta 8 personas donde en todos los proyectos que estos han trabajado se hace una especificación por roles y más del 80 % de los encuestados consideran que el rol de programador es el que mayor peso tiene en un proyecto de desarrollo de software. Cerca del 85 % del personal involucrado en el desarrollo del software tiene un alto grado de experiencia en el desarrollo de software especialmente en software empresarial donde ha trabajado el 69.23 % así como un alto grado de conocimiento en la producción que se lleva a cabo en la cual esta totalmente centrada en el control de los procesos según los encuestados. Más del 90 % de los desarrolladores consideran que debe existir una mayor comunicación entre los miembros del equipo de desarrollo donde más del 75 % plantea que estas relaciones que se establecen no son ni tan formales ni tan informales.

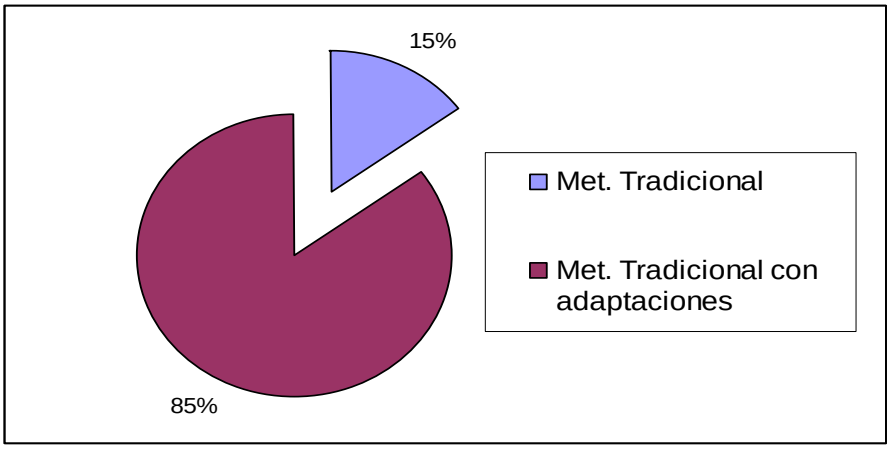

Figura 1.1 Situación de la empresa Desoft SS

A partir de cada uno de los datos cuantitativos expresados anteriormente se define sobre una base real las condiciones en que se encuentra la empresa y en qué aspectos hacer posibles modificaciones para adecuar la empresa a las condiciones de desarrollo actual, donde mediante la aplicación de la referencia desarrollada se propone una sugerencia para tratar de erradicar las diferentes dificultades que la misma afronta.

Como se puede apreciar (Anexo 5 Encuestas realizadas a los desarrolladores de la empresa Desoft) el contenido de cada cuestionario está elaborado de modo que a partir de la información obtenida se descubren los posibles cambios, mejoras y adecuaciones que puede tener la empresa. Las encuestas brindan detalles explícitos de cómo se ve el proceso que actualmente se está desarrollando y las posibilidades que existen de adaptarse a nuevas reformas.

Mediante la caracterización y evaluación de la organización antes realizada se llega a la conclusión que se puede hacer uso de metodologías ágiles en dicha empresa según las condiciones existentes y que para esto se necesita de una guía que dirija el camino de dicha aplicación.

1.8 Conclusiones parciales

En este capítulo se hizo una revisión exhaustiva de las metodologías ágiles de desarrollo de software que en la actualidad tienen mayor divulgación y aplicación por parte de los desarrolladores de software en dicho proceso; cuya revisión está en relación con el desarrollo y evolución de cada uno de las mismas a través de sus particularidades. Además, teniendo en cuenta, los tipos de proyectos a los que estas metodologías en general pueden ser aplicadas.

De acuerdo a toda la investigación realizada se arriba que la metodología que cubre la mayor cantidad de aspectos necesarios para lograr un estructurado y eficaz desarrollo de software en comparación con las demás es Programación Extrema. Esta integra de una forma muy efectiva cada una de las prácticas de desarrollo que propone y las complementa con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo lo que posibilita un desarrollo equilibrado. Organiza todo el proceso previendo cada uno de los detalles en cuanto al proceso general del desarrollo de software .Se sustenta en bases muy sólidas para la aplicación de sus prácticas, donde cada una de ellas esta muy bien definido el por qué de su utilización y los beneficios que esta representa. Mientras que por otro lado la mayoría de las restantes metodologías ágiles a pesar de que también son de gran utilidad y presentan una serie de importantes funcionalidades mas bien son puesta en práctica como complementos

unas de otras e incluso mayormente como complemento de XP para reforzar tal vez en algún rasgo que en dependencia del proyecto a desarrollar se necesite. Por lo que la misma se utiliza como base para la propuesta de solución a desarrollar.

Además se analizó la situación en que se encuentra la empresa Desoft a través de la que se permite conocer el estado actual del proceso de desarrollo de software que se lleva a cabo en dicha empresa identificando los principales problemas y las futuras adaptaciones en base a estos; utilizándose dichos resultados como marco para la propuesta a desarrollar.

CAPITULO 2: REFERENCIA PARA LA APLICACIÓN DE LA METODOLOGÍA ÁGIL PROGRAMACIÓN EXTREMA A PROYECTOS DESOFT SS.

En este capítulo se construye la referencia para la aplicación de la metodología ágil Programación Extrema que se persigue como objetivo general del trabajo. Para llevar a cabo la misma se describe un esquema que consta de una serie de fases que describen un proceso lo necesariamente estructurado para ser utilizado como guía para la utilización de dicha metodología ágil en los procesos de producción de software. El esquema general de esta referencia se muestra en la Figura 2.1.

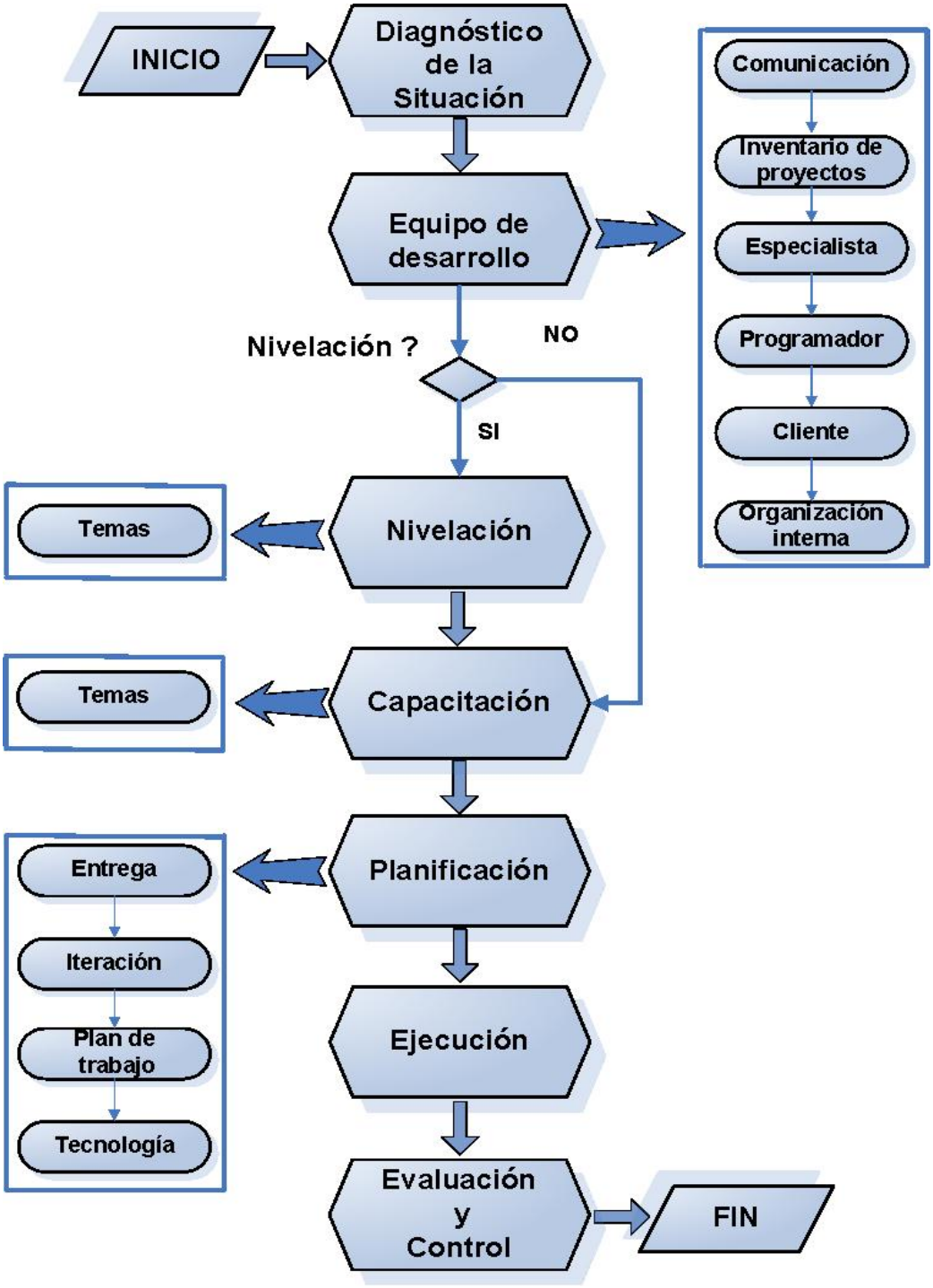

Figura 2.1 Esquema general de la referencia para la aplicación de la metodología ágil Programación Extrema.

Como se puede apreciar, el esquema se describe a través de varias fases las cuales serán descritas a continuación muy brevemente para obtener una visión general del proceso, pues más adelante se describe de forma detallada cada una de las características que cada una de estas recoge:

1. Inicio: Precondiciones a tener en cuenta para el desarrollo del software.
2. Diagnóstico de la situación: Condiciones de la empresa para implantar agilidad al proceso de desarrollo.
3. Equipo de desarrollo: Selección del personal que estará relacionado de una u otra forma con la producción del software.
4. Nivelación: Miembros del equipo de desarrollo al mismo nivel en cuanto a conocimientos técnicos se trata.
5. Capacitación: Capacitación de cada uno de los miembros del equipo de desarrollo en cuanto al uso y conocimiento de las metodologías ágiles así como los temas relaciones con el software específico a desarrollar.
6. Planificación: Actividades bajo las cuales esta estructurado el proceso de desarrollo.
7. Ejecución: Se tienen en cuenta cada de una de las prácticas ágiles seleccionadas para llevar a cabo la producción.
8. Evaluación y control: Comprobar la correspondencia entre lo planificado y lo real.
9. Fin: Culminación del proyecto.

A continuación se comienza la descripción de la referencia a través de la explicación de cada una de las fases en que la misma se sustenta.

2.1 Inicio

El inicio de la producción del determinado software se enmarca en el contexto de ir preconcibiendo todos aquellos aspectos que servirán de base y guía para el proceso que se de ha llevar a cabo y así poder encaminar dicho proceso de acuerdo a las condiciones y necesidades de ambas partes, clientes y desarrolladores.

Los aspectos que se deben ir teniendo en cuenta según el proceso de desarrollo de software que las MAs definen en general son las siguientes:

Se tiene en cuenta que se va a efectuar un proceso pensado en base a la adaptabilidad, donde la variabilidad de los requisitos que puedan ir surgiendo a medida que el proceso de desarrollo avanza no provocan grandes pérdidas, donde el costo de los cambios es mínimo, lo cual es una tarea difícil de ejecutar pero que su efectiva ejecución trae resultados considerablemente satisfactorios. En particular se requiere de un equipo eficaz de desarrolladores. El equipo necesita ser eficaz tanto en la calidad de los individuos como en la manera en que funcionan juntos en equipo, siendo este último un aspecto esencial.

La adaptabilidad requiere de un buen equipo en todos los sentidos. La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta puesto que hay muchas variables en juego, debe ser flexible para poder adaptarse a los cambios que puedan surgir. También la estructura del software debe ser flexible para poder incorporar los cambios sin demasiado coste.

Ningún proceso podrá nunca opacar las habilidades del equipo de desarrollo, de modo que el papel del proceso es apoyar al equipo de desarrollo en su trabajo. Explícitamente puntualizan el trabajar a favor de la naturaleza humana en lugar de en su contra.

Otra característica del proceso de desarrollo es que debe ser guiado por el cliente, donde existe una estrecha colaboración entre ambas partes.

Todos los integrantes del equipo deben ser capaces de mantener una paz constante. No se trata de desarrollar lo más rápido posible, sino de mantener el ritmo de desarrollo durante toda la duración del proyecto, asegurando en todo momento que la calidad de lo producido es máxima.

Cada uno de estos aspectos se centra en el objetivo de lograr una productividad eficiente. A partir de esta se logra suministrar al cliente un producto final que se encuentra en correspondencia con las especificidades que este planteó al comienzo de la producción y el equipo de desarrollo puede desempeñarse de una forma más eficiente y al mismo tiempo agradable.

2.2 Diagnóstico de la situación empresarial

Según lo expresado anteriormente cada uno de estos parámetros definidos inicialmente pueden influir negativa o positivamente en el resultado final de acuerdo a cómo estos se manifiesten. Entre ellos el fundamental es la elección de una metodología de desarrollo que satisfaga las necesidades tanto de los clientes al obtener el producto deseado como del equipo de desarrollo que logra establecer un proceso

estructurado de la producción sin mayores dificultades; se necesita una metodología que se adecue con el proceso que se va a desarrollar y a las condiciones existentes.

Se tienen en cuenta para esto las habilidades que deben tener los posibles miembros a formar el equipo de desarrollo de software, la experiencia que se tiene, cómo han sido las relaciones que se establecen entre los miembros del equipo y cómo es la organización interna entre ellos.

Para diagnosticar si la empresa está en condiciones de aplicar una metodología ágil se propone el empleo del Método de Estrella de Barry Boehm y Richard Turner (Anexo 5 Método de Estrella de Barry Boehm y Richard Turner.) Este método define un gráfico en forma de estrella de cinco puntas donde cada una de estas expresa una característica diferente que se debe cumplir para utilizar este tipo de metodologías. Los 5 aspectos que se definen son:

- Personal: define el nivel profesional que debe tener cada uno de los miembros del equipo donde se dividen en técnicos competentes y expertos frente al de principiantes y menos hábiles.
- Criticidad: define la criticidad que tiene el proyecto, el nivel de pérdidas que pueden producir los errores cometidos en el desarrollo.
- Tamaño: define el número de personas que debe tener el equipo de desarrollo
- Cultura: define la cultura de la empresa para adaptarse a entornos caóticos.
- Dinamismo: define el cambio de los requisitos, si cambian fácilmente y con rapidez.

Cada uno de estos datos se expresa en forma cuantitativa para que estos sean más precisos y así utilizarlos con mayor seguridad.

A medida que la estrella se dirija más al centro esto expresa que se encuentra en territorio donde puede aplicar agilidad mientras que cuando esta se expande denota que se acerca más a la formalidad. Si esta tiene muy poca simetría entonces ninguno de los 2 procesos está en condiciones de ser aplicado.

De acuerdo a lo obtenido según este método, si las condiciones están totalmente creadas para implementar este nueva forma de desarrollo, tales como un equipo de desarrollo muy bien estructurado y con personal con habilidades, con una buena comunicación entre los miembros, estructura organizacional acorde a las características y a las necesidades, proyecto bajo poca criticidad, con un numero adecuado de personal, se tienen todos los recursos disponibles, etc. entonces se esta listo para aplicar metodologías ágiles.

Las encuestas realizadas para evaluar las condiciones en que se encuentra la empresa Desoft para la aplicación de metodologías ágiles son utilizadas para obtener una fundamentación lo suficientemente sólida para la elaboración de la propuesta de solución a desarrollar y que se pueden utilizar de forma general con el mismo propósito de determinar el uso de una metodología ágil en el proceso de desarrollo de software. (Anexo 4 Encuestas realizadas a los desarrolladores de la empresa Desoft)

2.3 Conformación del equipo de desarrollo

La conformación del equipo de desarrollo es una de las primeras actividades que se realiza cuando se procede a desempeñar cualquier actividad productiva, en especial del tema que se trata, la producción de software.

Esta se basa en la selección de todo aquel personal que estará vinculado con la producción del software a desarrollar. En este caso se tiene por una parte a los desarrolladores (llamados así para agruparlos de una forma más general, pero que posteriormente se hace una distinción entre estos), y son aquellos que están implicados directamente con la producción del software y por otra parte a los de los especialistas en cada uno de los temas que estén relacionados con el entorno del software a producir. Es necesario enfatizar en este punto que el cliente también está vinculado con la producción del software. El cual estará formando parte del equipo pero de una forma muy particular en relación con el personal anteriormente mencionado. Este aspecto será detallado más adelante.

Las personas son el principal factor de éxito de un proyecto software. Si se sigue un buen proceso de desarrollo, pero el equipo falla, el éxito no está asegurado; sin embargo, si el equipo funciona, es más fácil y posible conseguir el objetivo final, aunque no se tenga un proceso completamente definido. Se necesitan desarrolladores que se adapten bien al trabajo en equipo, enfatizando que este punto es una base fundamental. También estos deben constar con cierta madurez, experiencia y talento debido a que las prácticas que propone la metodología ágil presentan una particular forma de implementación, donde es cierto que se necesita de experiencia y talento para establecer un desarrollo consecuente de acuerdo a las mismas. En varios casos se específica que estas metodologías solo funcionarían con integrantes que posean estas cualidades debido a lo planteado anteriormente pero estas metodologías también pueden ser aplicadas a otra escala del personal, una muestra de esto es el estudio realizado en la Universidad Autónoma de Aguascalientes, México (SAAVEDRA *et al.* 2006). Así mismo, las herramientas son importantes para mejorar el rendimiento del equipo, pero el disponer más recursos que los estrictamente

necesarios también pueden afectar negativamente. A partir de lo que se obtiene que sea más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades. De acuerdo a lo explicado anteriormente se evidencia que este proceso de desarrollo es centrado en las personas y sus interacciones y no en los procesos y las herramientas. (LETELIER and PENADÉS 2006)

Cada uno de los miembros del equipo de desarrollo debe cumplir con una serie de valores que son indispensables para llevar a cabo el uso correcto de metodologías ágiles y en los cuales se sustenta la eficacia del equipo. Se debe garantizar que cada uno de dichos integrantes sea: consagrado, responsable, comunicativo y que tenga coraje para llevar a cabo decisiones de gran trascendencia.

2.3.1 Comunicación entre los miembros del equipo de desarrollo

La comunicación uno de los aspectos fundamentales que hay que tener en cuenta en el momento de conformar el equipo de desarrollo de software, esta debe involucrar a todos los participantes en el proyecto, ser efectiva, libre y sincera.

La comunicación abierta y sin restricciones permite el diálogo, la capacidad de los miembros del equipo para suspender los supuestos e ingresar en un auténtico pensamiento conjunto. El diálogo entre clientes, desarrolladores, y toda persona vinculada al proyecto, permite una visión más general del sistema a construir.

Las mejores estrategias que puede aplicar la organización para mejorar su desempeño dependen del desarrollo de sus procesos comunicativos internos entre los miembros del equipo de desarrollo. Los Procesos de Comunicación se refieren a las relaciones entre los diferentes caminos de comunicación por efecto de las muchas formas de interactuar que se generan en la organización como resultado del operar de los elementos constituyentes de la misma. Estos procesos afectan directamente las posibilidades de éxito o fracaso que se puedan dar en la organización.(MATERÓNA 2005)

2.3.2 Inventario de proyectos

En este aspecto se hace una revisión de todos los proyectos por los cuales se han incursionado a través de todo el ciclo de producción de software con el objetivo de tener una visión mucho más clara de las posibilidades con que realmente se cuenta para enfrentar un determinado proyecto. Además se

determina qué proyecto tiende a ser desarrollado con más facilidad en base a experiencias anteriores, las habilidades con que cada desarrollador está más relacionado. A través de este inventario se tiene una referencia del desempeño de cada desarrollador. De acuerdo a esto se puede determinar la experiencia, la madurez y capacidad con que cuenta el personal involucrado en la producción del software y así utilizar esta información como base para la selección correcta.

La tabla a continuación representa como estos datos serán recogidos donde se muestra el nombre de los proyectos realizados, a que rama están dirigidos y una breve descripción de lo que cada uno de estos.

Inventario de proyectos				
No	Nombre del proyecto	Tipo Proyecto	Integrantes	Descripción
1				
2				

Tabla 2.1 Inventario de proyectos

2.3.3 Especialistas

En este paso se hace la selección de los especialistas. Estos son aquellas personas que no están vinculadas directamente con la producción del software y no forman parte del equipo de desarrollo sino más bien se caracterizan por ser una base de apoyo para llevar a cabo dicho proceso. Pueden ser conocidos también como consultores según se define en los roles antes citados. Cada uno de los requerimientos propuestos por el cliente se encuentra en función del tema al cual estará vinculado dicha producción. En dependencia de esto se tendrá en cuenta los especialistas en el tema correspondiente que se necesitarán para que los mismos hagan llegar a los desarrolladores la información pertinente en base al tema que se trata. Se requiere entonces gestionar cada uno de los mismos, ya sea por parte del cliente o la empresa dedicada a la producción. De igual forma se encuentran aquellos especialistas que están vinculados con la parte técnica del proceso, los cuales también se tienen en cuenta en caso de necesitar ayuda técnica. En la tabla que a continuación se muestra se define una manera muy efectiva y rápida de gestionar este personal con el objetivo de tener un conocimiento de cada uno de los especialistas que serán solicitados para responder a esta tarea de acuerdo a su clasificación y a partir de lo cual será mucho más rápida y eficiente su estructuración. En esta se recoge el nombre de los especialistas que se

necesitaran de acuerdo a la necesidad del proyecto y la especialidad a la que cada uno de estos pertenece.

Especialistas por temas		
No	Nombre del especialista	Especialidad
1		
2		

Tabla 2.2 Especialistas por temas

2.3.4 Programadores

La selección de los programadores se hace teniendo en cuenta que estos van a trabajar en pares. Es necesario aclarar que esta práctica no es aplicada para todo tipo de programador de acuerdo a su estrategia de aplicación, y es en este instante en que se debe tener en cuenta la serie de valores que anteriormente fueron expresados y en los cuales esta práctica debe ser muy bien sustentada. Es necesario especificar que la cantidad de los programadores debe oscilar en un rango de 3 a 10 teniendo en cuenta la propuesta original de Kent Beck en (BECK 2000).

Todo el código será desarrollado en parejas, dos personas compartiendo un solo monitor y teclado, ambos frente al ordenador. Quien codifica estará pensando en el mejor modo de implementar un determinado método, mientras que su compañero lo hará de una manera más estratégica pensando si se está siguiendo el enfoque apropiado, qué podría faltar, cuáles son las posibles pruebas que se pueden hacer, de qué forma se puede simplificar el sistema que se está desarrollando.

Los roles entre la pareja son intercambiables, de manera que en cualquier momento quien observaba puede tomar el teclado para ejemplificar alguna idea o, simplemente, para turnar a su compañero. Igualmente, la composición de las parejas cambiará siempre que uno de los dos sea requerido por algún otro miembro del equipo para que le ayude con su código. Puede estar emparejado en una ocasión con una persona e indistintamente con otra, si se tiene un trabajo sobre un área que no se conoce muy bien se puede unir con otro miembro del equipo que conozca esa área. Cualquier miembro del equipo se puede emparejar con cualquiera.(ACEBAL and LOVELLE 2002)

La implementación de esta práctica trae consigo numerosas ventajas entre las que se encuentran que el promedio de errores al terminar el producto es mucho menor ya que la mayoría de estos son detectados al insertar el código y rápidamente corregidos, otra de las ventajas que logra una mayor comunicación, la comunicación cara a cara hace que la información se divulgue más fácilmente, hay un intenso intercambio entre las ideas de los programadores y así una transferencia de conocimientos por parte de cada uno de los miembros del equipo donde se logra un mayor flujo de información, la dinámica entre el equipo. Esta técnica posibilita que otras se garanticen, esto es en el caso de que en algún momento pase por alto un paso que si la pareja está atenta esto se puede solucionar rápidamente, aunque es importante aclarar que esto no significa que no se cometan errores pero sí mucho menos. Esta práctica no es para que cada miembro de la pareja aprenda del otro como una forma de tutores, pero si uno tiene un poco más de experiencia que el otro al pasar el tiempo cada uno va aprendiendo del otro y al final se obtiene una pareja fuerte en conocimiento.

2.3.5 Clientes

Los clientes (o representantes del cliente) se encuentran formando parte del equipo de desarrollo. Es fundamental enfatizar en la importancia que tiene la presencia de estos durante la producción del software. Esta es una práctica que sustenta mucha de las demás que propone XP, a partir de esta las demás logran un mayor sentido de su utilización.

El cliente aquí forma parte del equipo de desarrollo, este se encuentra mucho más cerca del proceso de desarrollo. Con su activa participación, la fase inicial de recopilación de requerimientos toma otro sentido. Inicialmente se hace la captura de requisitos pero de forma muy distinta. El cliente describe las características que se necesita que tenga el producto final, plasmándolas en las historias de usuarios. Una historia es precisamente una lista de todas las funcionalidades que el cliente especifica que el sistema debe cumplir y a través de la cual se establecen las prioridades de cada una de las funcionalidades a desarrollar. También se permite que estos requisitos se vayan recogiendo a lo largo del proyecto, de manera ordenada. De esta forma se posibilita que el cliente pueda ir cambiando de opinión sobre la marcha del desarrollo, pero a cambio este ha de estar siempre disponible para solucionar las dudas del equipo de desarrollo. Además los clientes son los encargados de escribir y correr las pruebas de aceptación para verificar la funcionalidad que va logrando la aplicación en desarrollo.

2.3.6 Organización interna

Teniendo en cuenta que el proceso de desarrollo de software es un proceso adaptable entonces la necesidad de organizarse para el cambio requiere asimismo un alto grado de descentralización. La razón es que el equipo tiene que estructurarse para tomar decisiones rápidamente. Y esas decisiones tienen que basarse en cercanía a la ejecución, al mercado, a la tecnología, a los muchos cambios que hay en la sociedad, en el ambiente, en la demografía y en el conocimiento, que brindan oportunidades para la innovación, si la empresa es capaz de verlas y se utilizan.(MATERÓNA 2005)

La organización interna de la empresa en general específicamente el equipo de desarrollo debe implementar diferentes aspectos para poder abordar un proyecto utilizando metodologías ágiles y poder alcanzar resultados satisfactorios; existen varias estrategias que se han de seguir para tener garantías de éxito en un proyecto de este tipo.

Se debe mantener un estilo de gestión basado en el liderazgo y la colaboración. Para asumir un proyecto ágil la posición clásica de dar órdenes y controlar al personal no tiene sentido. Las jerarquías estrictas del orden y el control, jefe-ordena y empleado-obedece, acaban indefectiblemente en situaciones en que las personas no se sienten motivadas con el trabajo que realizan, sin estimulación por hacer cosas nuevas y de obtener productividad y buenos resultados en lo que se produce. Estas se limitan solo a hacer sus propias actividades sin aportar nada por lo que no tenga un valor monetario. La influencia de la agilidad, a través del liderazgo colaborativo logrará que el equipo logre una gran colaboración, comunicación y compromiso, dando lo máximo de sí.

La estructura de la organización debe ser orgánica, donde la organización sea flexible, reflexiva, participativa y que facilite la cooperación. Cada persona es una parte importante dentro del equipo, cada cual tiene indistintamente su responsabilidad pero el resultado funciona como un todo, estrechamente relacionado.

La cultura organizacional tiene que ser centrada en las personas que conforman el equipo de trabajo y no en los procesos. El éxito está en la interacción de las personas con los procesos y con la tecnología y no en la visión única de control de procesos. La empresa debe ser organizada en torno a las personas, que son las que pueden dar la agilidad al proceso de desarrollo, esa capacidad de adaptación a situaciones nuevas que no hayan sido previamente pensadas y modeladas en un proceso, pero que son vitales para la supervivencia de la entidad.

La gestión del conocimiento del personal debe ser tácito. El conocimiento suele estar siempre en los cerebros de las personas que conforman el equipo de desarrollo, pero siempre se tiende a intentar guardarlo de forma explícita, absolutamente todo y al máximo nivel de detalle posible, eso trae como consecuencia que haya que documentar periódicamente el conocimiento y claro no se hace como se explicaría de una forma más informal, porque se ha de ser formal y explícito. Precisamente este efecto de documentarlo todo, es lo que en ocasiones incide negativamente pues para muchos se pierde un poco de tiempo en documentar en muchas ocasiones algo más que lo necesario en lugar de invertirlo en crear, en adquirir nuevo conocimiento. Es mucho más efectivo hacer una pequeña anotación de alguna muy buena idea y llevarla rápidamente a desarrollar en lugar de tener cantidades de documentos repletos de superficialidades.

La comunicación entre los miembros debe ser informal si se pretende que el equipo colabore, comparta conocimiento y funcione al unísono, no se puede establecer comunicaciones rígidas, formales y llenas de burocracia si se quiere trabajar ágilmente.

La visión de que el cliente se encuentra solo en la fase de inicio, donde se recogen los requisitos que el sistema debe cumplir, aquí se establece de una forma distinta en relación con lo que se pretende lograr. En el caso de las metodologías ágiles el cliente forma parte indisoluble del equipo de desarrollo, este es una pieza fundamental en el proyecto. El cliente pasa a ser parte implicada en el equipo de desarrollo. Tiene un papel importante de interacción con el equipo de programadores, sobre todo después de cada cambio, y de cada posible problema localizado, mostrando las prioridades, expresando sus sensaciones, etc.

Los roles que se definen en el proyecto deben ser intercambiables, el poder intercambiar roles y funciones en un mismo proyecto, da la posibilidad de desarrollar nuevos puntos de vista y así atacar los mismos problemas desde diferentes puntos de vista. De esta forma las personas relacionadas con el desarrollo del software crecerán en motivación, en conocimiento y la empresa será más adaptable. (GONZÁLEZ 2006)

Cada uno de los miembros del equipo de desarrollo asume un rol en específico pero es importante aclarar que estos roles son utilizados para lograr una mayor organización entre el equipo y no para que cada cual desarrolle su trabajo independientemente, cada quien juega su papel indistintamente en un instante de tiempo donde a medida que avanza el proceso de desarrollo cada una de sus acciones son integradas de forma indisoluble.

A continuación se especifican las tareas principales que cada uno de estos soporta. Cada uno de los cuales está basado en la propuesta original de Kent Beck acerca de la Programación Extrema. (BECK 1999)

1. Programador: Escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.
2. Cliente: Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
3. Encargado de pruebas: Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
4. Encargado de seguimiento (Tracker): Proporciona retroalimentación al equipo en el proceso XP.
5. Entrenador: Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
6. Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.
7. Gestor: Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Esta técnica establece una gran comunicación entre clientes y desarrolladores al estar todo el tiempo presente para aclarar alguna duda que se manifieste por parte de algún desarrollador o resolver alguna inquietud. Además posibilita que se implementen sólo las funcionalidades que exactamente necesite el cliente si al existir alguna confusión este está presente para saldarla, logrando así también simplicidad, uno de los valores que es necesario tener en cuanto a la producción del código logrando también que la documentación sea la necesaria.

2.4 Nivelación

La nivelación es un proceso que se lleva a cabo con el fin de adquirir o afianzar los conocimientos en algunas materias técnicas básicas que sean imprescindibles para el proyecto, es decir, que cada uno de los miembros del equipo de desarrollo se encuentre relativamente al mismo nivel respecto a las habilidades y conocimientos que se requieren para el desarrollo del software a ejecutar. Puede verse desde el punto de vista desde el inicio del proyecto, cuando se seleccione el equipo de desarrollo o se puede tener en cuenta cuando algún miembro del equipo bajo diferentes razones abandona el proceso de producción del software y se incorpora otro nuevo miembro al mismo.

Esta nivelación trae como consecuencia una gran estabilidad en el equipo de desarrollo y por consiguiente una mayor productividad y eficiencia en el trabajo que se lleva a cabo. Además si todos se encuentran nivelados, la comunicación (aspecto fundamental de las MAs) que se logra se hace más fructífera y productiva pues esta puede ser mucho más informal, especificándose exactamente lo que se necesita, donde el contenido de lo que se expresa es asimilado de forma directa por el interlocutor si se habla bajo las mismas condiciones.

2.4.1 Selección de temas de nivelación

Para llevar a cabo este proceso de nivelación se hace necesario una verificación respecto a los temas que están vinculados a la parte técnica del desarrollo del software donde se presenta algún tipo de deficiencia con el fin de corregir la misma. Para saber que temas son necesarios se debe hacer una especie de reunión donde se recojan las mayores dificultades que se afrentan para llevar a cabo el proyecto y en base a esta dificultad entonces se planificarán diferentes cursos con el objetivo de satisfacer dichas necesidades y así poder obtener un mayor rendimiento en la producción.

A continuación se muestra la forma en que cada uno de estos datos de los temas a tratar serán recogidos para una mayor velocidad en el proceso y actualización de los mismos en caso de ser necesario y una mayor organización del proceso.

Este proceso de nivelación se llevará a cabo a través de cursos que se impartirán al equipo de desarrollo por parte de los especialistas anteriormente seleccionados. Estos cursos deben ser estructurados en base a la fecha que estos comenzarán, a la duración que estos requerirán y que

actividad tratarán, teniendo en cuenta el tema que en especifico se abordará. La tabla que a continuación se muestra describe cómo estos datos serán recogidos.

Plan de nivelación técnica					
Tipo de proyecto	Tema	Actividad	Fecha	Duración	Especialista

Tabla 2.3 Plan de nivelación técnica

2.5 Capacitación

La capacitación es un proceso que se realiza con el objetivo de suministrar nuevos conocimientos a los individuos en determinado tema que resulte necesario en este caso para llevar a cabo el desarrollo de un software en específico.

Este proceso de capacitación permite que cada uno de los integrantes del equipo de desarrollo tenga una base mucho más sólida en relación tanto con el tema del software a desarrollar como en aspectos técnicos, lo que trae como consecuencia que la implementación de dicho software sea mucho más eficaz si se tienen bien esclarecidos cada una de las actividades que se necesitan automatizar.

2.5.1 Selección de los temas de capacitación

Al igual que en el proceso de nivelación para definir cuáles son los temas que se requieren para que constituyan

Este proceso de capacitación se lleva a cabo en base a diferentes temas:

- Dominio del problema en cuestión del software a desarrollar.
- Técnicos

Cada software que se desarrolla está orientado a proveer mejoras en diferentes empresas a través de la automatización de determinados procesos que se desarrollan en dicha entidad, donde cada uno de estos procesos está relacionado con un tema es específico. Por lo que se hace necesario en la mayoría

de los casos que cada uno de los integrantes del equipo de desarrollo se identifique con el tema en cuestión para poder llevar a cabo la producción del software especificado.

La misma situación ocurre cuando alguna nueva tecnología será utilizada para llevar a cabo la producción de dicho software. Se refiere a tecnología cualquier herramienta, proceso, modelo, procedimiento, metodología, técnica que se utilice para la producción de software. En este caso particular tenemos el uso de una metodología ágil, el cual es un aspecto esencial a tener en cuenta, pues al ser una nueva alternativa en el desarrollo del software su utilización para muchos desarrolladores resulta nueva en cuanto al uso de sus prácticas integradas. Por lo que se hace necesario llevar a los desarrolladores una visión general de cómo se manifiestan las prácticas que estas proponen.

Por lo tanto se hace necesario que cada uno de los miembros del equipo enfrente un proceso de capacitación en cuanto a cada uno de los temas antes mencionados siempre que sea necesario para garantizar el éxito en el proceso de producción

Se estructura un sistema de cursos que serán impartidos por un especialista determinado en dependencia del tema necesitado, en los cuales cada uno de los integrantes del equipo de desarrollo debe apropiarse de los conocimientos necesarios y suficientes para poder ser capaz de afrontar un proceso de desarrollo guiado bajo la agilidad.

A continuación se muestra cómo se gestionará la información de forma rápida para realizar estos cursos de capacitación.

Plan de capacitación técnica					
Tipo de proyecto	Tema	Actividad	Fecha	Duración	Especialista

Tabla 2.4 Plan de capacitación técnica

Plan de capacitación del dominio del problema					
Tipo de proyecto	Tema	Actividad	Fecha	Duración	Especialista

Tabla 2.5 Plan de capacitación del dominio del problema

2.6 Planificación

Al llegar a este paso se tienen todos los factores externos necesarios para comenzar con la planificación del proyecto en sí. A partir de aquí se tiene claramente el proyecto en cuestión a desarrollar y se comienza a desglosar toda una serie de actividades específicas del proceso de desarrollo del software que guiarán dicha producción.

La etapa de planificación se manifiesta de forma general como una permanente comunicación entre el cliente y los programadores, en la que los primeros decidirán el alcance – ¿qué es lo realmente necesario del proyecto?–, la prioridad –qué debe ser hecho en primer lugar –, la composición de las versiones –qué debería incluir cada una de ellas – y la fecha de las mismas. En cuanto a los programadores, son los responsables de estimar la duración requerida para implementar las funcionalidades deseadas por el cliente, de informar sobre las consecuencias de determinadas decisiones, de organizar la cultura de trabajo y, finalmente, de realizar la planificación detallada dentro de cada versión.(ACEBAL and LOVELLE 2002)

En este punto se tendrá que elaborar la planificación por etapas, donde se aplicarán diferentes iteraciones. Para hacerlo será necesaria la existencia de reglas que se han de seguir por las partes implicadas en el proyecto para que todas las partes tengan participación y se sientan realmente partícipes de la decisión tomada. La planificación que se realiza no es de forma estricta sino que puede ser modificada de acuerdo a los cambios que vayan surgiendo a medida que se avanza en el desarrollo del software. No se puede saber todo lo que se va a ser necesario ni evaluar los tiempos correctamente. La planificación deberá revisarse y modificarse continuamente a lo largo del proyecto.

2.6.1 Planificación de la entrega

Inicialmente tanto el cliente como el programador necesitan determinar diferentes aspectos en función de los cuales se centrará la producción. Esta actividad se enmarca durante la planificación de la entrega. Una entrega representa de una a tres meses de trabajo, equivalente al rango de 2 a 5 iteraciones.(COCKBURN 2006). Cuyo objetivo es ayudar al cliente a identificar los rasgos del software que estos necesitan, ha entender lo que el sistema hace, dar la oportunidad a los programadores de explorar la tecnología que se necesita para llevar a cabo la producción del software, hacer las estimaciones pertinentes y suministrar como una planificación general para todo el equipo de desarrollo.

Los requisitos del cliente son recogidos a través de historias de usuario. Una historia de usuario en un texto de una o dos frases en las que se dice algo que debe hacer el sistema. Es más extensa que un requisito (que suele ser una frase corta) y menos que un caso de uso (que puede ser de una o dos páginas). Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

De acuerdo a la complejidad del sistema, debe haber al menos una historia por cada característica importante, y propone realizar una o dos historias por programador por mes. Si se tienen menos, probablemente sea conveniente dividir las historias, si se tienen más lo mejor es disminuir el detalle y agruparlas. Para efectos de planificación, las historias pueden ser de una a tres semanas de tiempo de programación (para no superar el tamaño de una iteración).(JEFFRIES R *et al.* 2001)

Estas historias de usuarios constituyen la documentación esencial que se genera durante todo el proceso de desarrollo ya que estas recogen los datos que son considerados esenciales para llevar a cabo la producción y que resumen de manera precisa las características que el sistema debe tener y los datos que serán realmente utilizados para el desarrollo de dicho software.

Respecto de la información contenida en las historias de usuario y las tareas, existen varias plantillas sugeridas pero no existe un consenso al respecto. Beck en su libro (BECK 1999) presenta un ejemplo de ficha historias de usuario y tareas de programación en la cual pueden reconocerse los siguientes contenidos: fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, número de historia,

prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción, notas y una lista de seguimiento con la fecha, estado cosas por terminar y comentarios.

A continuación se muestra una plantilla de historia de usuario en la cual se han seleccionado algunas de las características consideradas más importantes. Es necesario aclarar que debido a que no existe un estándar para estas plantillas cada equipo de desarrollo puede establecer la estructura deseada e incluir las características que considere pertinentes. De forma semejante puede hacerse para las tareas de programación.

Historia de usuario	Fecha:
Nombre de la historia:	
Número:	
Tipo de actividad Nueva____ Corrección_____ Mejora____	
Prioridad: _____	
Estimación técnica:	
Referencia a historia previa: ____	
Riesgo:	
Descripción:	

Figura 2.2. Historia de usuario.

Es posible que en un principio no se identifiquen todas las historias de usuario pero en cada iteración los cambios serán analizados y se incorporan para las próximas iteraciones. A partir de lo anterior:

Se escriben las historias de usuario por parte de los clientes donde se describen brevemente los rasgos del sistema, esto debe mantenerse de la forma más simple posible para que el programador pueda estimar las historias.

De acuerdo a las historias escritas los programadores estiman el esfuerzo asociado a la implementación de las mismas utilizando como medida el punto de historia. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Si la estimación indica que la implementación de esta se toma más de 3 semanas entonces pasa nuevamente a manos del

cliente para que esta la divida en 2 o más historias debido a su complejidad, posibilitando esto mayor flexibilidad a la hora de la implementación.

Si los programadores no saben cómo estimar alguna cosa se hace una rápida programación exploratoria, donde rápida puede significar minutos a horas o posiblemente una pareja de días. El resultado es suficiente conocimiento para intentar nuevamente una estimación. Una forma también muy simple de estimar las historias es compararla con una historia similar que ya se ha entregado, a través de la cual observando su similitud se estima la nueva historia.(BECK; and FOWLER 2000)

Teniendo en cuenta lo anteriormente explicado se planifica la siguiente entrega priorizando una lista de historias concurrentemente planificadas para ser incluidas en la próxima entrega. Las historias son ordenadas por el cliente de acuerdo al valor del negocio que estas aporten, estableciendo prioridades, desde el mayor al menor valor. Se clasifican de la siguiente forma:

- sin las que el sistema no puede funcionar
- las que son menos importantes pero que suministran valor al negocio
- las que serían agradables tener

Los clientes pueden cambiar sus prioridades en cualquier momento.

Mientras que los programadores ordenan estas según el riesgo técnico en 3 categorías:

- las que puede estimar precisamente (bajo)
- las que puede estimar razonablemente bien (medio)
- las que no puede estimar del todo o en absoluto (alto)

El programador declara la velocidad que define cuántos puntos de historia puede implementar. 1/3 de puntos de usuario por semana.

El cliente elige el alcance a través de la selección de las historias para la próxima entrega. Cuán largo podría ser el esfuerzo del desarrollo.

Si se necesitan nuevas historias durante el desarrollo se escriben, los programadores estiman la nueva historia y el cliente elimina la estimación equivalente desde el plan restante e inserta la nueva historia. (WAKE 2000)

También aquí se escribe la metáfora del sistema la cual se define como una historia, de la cual tanto los programadores como los clientes podrán expresarse acerca del funcionamiento del sistema. La metáfora expresa la visión evolutiva del proyecto que define el alcance y propósito del sistema. Su principal objetivo es mejorar la comunicación entre todos los integrantes del equipo, al crear una visión

global y común de lo que se quiere desarrollar. Esta puede cambiar a medida que se vaya aprendiendo del sistema y del entendimiento que se va teniendo del mismo. A través de la metáfora del sistema se captura parte de la arquitectura del software.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto mantenida por el equipo de desarrollo se utiliza para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias la cual esta establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación. (LETELIER and PENADÉS 2006)

Las entregas del producto se tienen que hacer cuanto antes mejor, es decir, cada vez que se tenga un nuevo rasgo que pueda ser suministrado al cliente, y con cada iteración, el cliente ha de recibir una nueva versión. Donde cada una debe de ser tan pequeña como fuera posible, conteniendo los requisitos de negocios más importantes logrando tener sentido como un todo en el producto final.

Cuanto más tiempo se tarde en introducir una parte esencial en el software, menos tiempo se tendrá para trabajar con ella después. Se aconseja muchas entregas y muy frecuentes. De esta manera un error en la parte inicial del sistema tiene más posibilidades de detectarse rápidamente.

2.6.2 Planificación de la iteración

Cada iteración necesita también ser planificada, es lo que se llama plan de iteración, donde el objetivo es tomar las historias de usuario que el equipo seleccionó para ser implementadas en dicha iteración, cuyas historias son aquellas que suministren mayor valor al cliente, las que se consideren esenciales. Luego estas historias serán divididas en pequeñas tareas de programación acorde a las historias seleccionadas para implementar. Estas se recogen también en tarjetas como las historias de usuario, las cuales se asignan a los programadores, estos las estiman y luego pasan a ejecutarse. Este plan de iteración se repite siempre antes de comenzar una nueva iteración.

En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de

esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio).

Al comienzo de cada iteración estarán registrados los cambios en las historias de usuario y según eso se planificará la siguiente iteración. Las iteraciones son de 1 a 3 semanas y al final de cada una el sistema tiene que estar listo para entregar, funcionando bajo las pruebas de aceptación.

Tan rápido como estén las pruebas funcionales listas y las tareas para una historia completada se pasa a verificar dicha historia haciendo correr estas pruebas para comprobar que la historia ha sido implementada correctamente. Por lo que al final de cada iteración el sistema tiene que estar a punto para entregar

Debido a esto se añade agilidad al proceso de desarrollo y evita que se mire demasiado hacia delante, desarrollando trabajos que aún no han estado programados.

El plan de iteración es un proceso continuo donde este se realiza cada vez que se termina una iteración con el objetivo de pasar a la siguiente y poder obtener para esta próxima a realizar tareas que queden pendientes. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior.

2.6.3 Plan de trabajo

Para tener un control de la actividad que realiza cada integrante del equipo se propone que cada uno de los miembros se trace un plan de trabajo consecuente con las actividades que realiza. Este, resume de una forma sencilla, muy eficaz y con claridad el desempeño de cada miembro. Se realiza con el objetivo que en caso de de ser necesario puede ser consultado sin mayor dificultad obteniendo los datos más imprescindibles de los desarrolladores.

En la tabla a continuación se muestra el prototipo que se propone para llevar a cabo este plan de trabajo. El cual especifica como aspecto fundamental la actividad que esté realizando en el proceso de desarrollo determinado integrante y cómo es el avance de la misma a través de dicho proceso.

Plan de trabajo		
Nombre :		
Período :		
Actividad que realiza	Estado (Inicial/Medio/Terminado)	Producto(VMF)

VMF: Versión media funcional

Tabla 2.6. Plan de trabajo

2.6.4 Selección de la tecnología a utilizar

Se selecciona todo el equipamiento necesario para llevar a cabo la producción del software. Hay que tener en cuenta que esta es una decisión tanto del cliente como de los desarrolladores. El equipo de desarrollo se familiariza con las herramientas y tecnologías que se utilizarán en el proyecto organizando todo esto en torno al equipo de desarrollo. Esta tecnología será estandarizada para todo el equipo de desarrollo con el objetivo de que cada uno de los integrantes trabaje sobre la misma base, pues algunas de las prácticas que se llevan a cabo utilizando metodologías ágiles están vinculadas con la tecnología a utilizar. En este caso tenemos la integración continua, estándares de codificación y la propiedad colectiva del código, las cuales más adelante se especificarán con un nivel de detalle mayor.

2.7 Ejecución

El proceso de desarrollo de software se basa en producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con toda la funcionalidad pretendida para el sistema pero si que constituyan un resultado de valor para el negocio. Por lo que a partir de aquí el proceso es guiado a través de versiones.

Cada versión debe de ser tan pequeña como sea posible, conteniendo los requisitos de negocios más importantes donde estas deben tener sentido como un todo en el producto final.(SOLÍS. 2003)

El cliente y el equipo de desarrollo se beneficiarán de la retroalimentación que produce un sistema funcionando, y esto se reflejará en sucesivas versiones.

La figura muestra una representación de cómo este proceso se lleva a cabo, donde los desarrolladores trabajan hasta obtener una versión del producto que exprese características mínimas funcionales a través de iteraciones, donde luego entregan esta al cliente para que este verifique su funcionalidad y adecuación de acuerdo a las necesidades requeridas por el mismo.

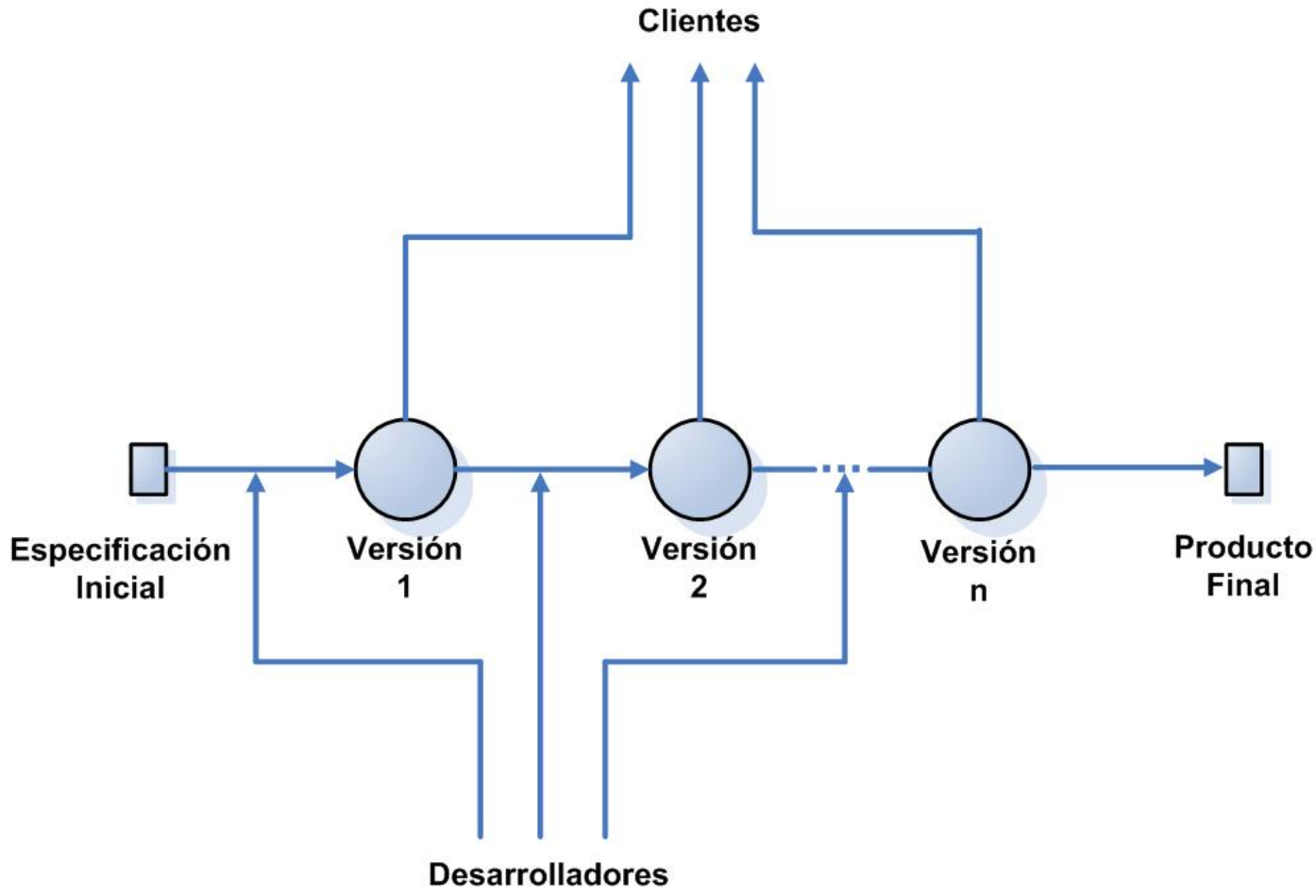

Figura 2.3 Proceso de ejecución

Cada versión está compuesta por varias iteraciones, las cuales se efectuarán según se planificaron en el plan de iteración. Para la primera iteración, se escoge un juego de historias simples y básicas que se espera obligará a que cree la arquitectura completa del sistema. Entonces se llevan a cabo las historias de la manera más simple que se puede trabajar donde al final se obtendrá una arquitectura. (BECK 2000)

Luego de realizada la planificación de acuerdo a lo expresado en el epígrafe Planificación, donde ya se tienen las primeras historias que serán implementadas se comienza un ciclo que consta de varias prácticas que definirán como se llevará a cabo este proceso.

Inicialmente se definirán diferentes pruebas antes de codificar con el objetivo de proporcionar una gran seguridad de que lo que se implementa está correcto.

Las pruebas se dividen en: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente. Para crear estas pruebas, se debe emplear algún marco de trabajo de pruebas automáticas para poder hacer varias simulaciones del sistema en funcionamiento. Para hacer estas simulaciones automatizadas, se pueden utilizar Ambientes de Prueba (Unit testing frameworks) en cualquiera de sus versiones para diferentes lenguajes. Un buen ejemplo de un ambiente de prueba es el JUnit para Java. También existen otros ambientes de pruebas para lenguajes como C, C++, Delphi, JavaScript, XML y servicios Web, etc.

No debe existir ninguna característica en el programa que no haya sido probada, los programadores escriben pruebas unitarias para chequear el correcto funcionamiento del programa mientras que los clientes realizan pruebas de aceptación para verificar que el software satisface sus necesidades de acuerdo a lo especificado en los requisitos. El resultado es un programa más seguro, que conforme pasa el tiempo es capaz de aceptar nuevos cambios.

Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. Por medio de pruebas al software se mantendrá informado del grado de fiabilidad del sistema. La retroalimentación actúa junto con la simplicidad y la comunicación, cuanto mayor retroalimentación más fácil es la comunicación. Cuanto más simple un sistema más fácil de probar. Escribir pruebas orienta como simplificar un sistema, hasta que las pruebas funcionen, cuando las

pruebas funcionen mucho de lo implementado demuestra su factibilidad logrando una muestra de avance seguro del desarrollo del software.

Después de estar las pruebas listas se comienza con la codificación. Todo el código que se produce debe estar de la forma más simple posible. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra debe ser removido inmediatamente. Según Kent Beck, un diseño adecuado para el software es aquel que: supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos.(BECK 2000)

Este se basa en la filosofía de que el mayor valor de negocio es entregado por el programa más sencillo que cumpla los requerimientos. Se enfoca en proporcionar un sistema que cubra las necesidades inmediatas del cliente, solo debe implementarse lo que se necesita en ese momento sin pensar en lo que tal vez se necesite después. Este proceso permite eliminar redundancias y rejuvenecer los diseños obsoletos de forma sencilla.

El código que se desarrolla es con propiedad compartida. Cualquier integrante del equipo que crea que puede aportar valor al código en cualquier parte puede hacerlo. Mientras se encuentre más personal trabajando en una pieza, menos errores aparecerán. Siempre que vea una posibilidad de simplificar, mediante refactorización, cualquier clase o cualquier método, hayan sido o no escritos por el miembro que lo detecte, deberá hacerlo. Después de que el código es cambiado se le deben aplicar las pruebas para asegurarse que todo continua funcionando. El uso de estándares de codificación y la seguridad que brindan las pruebas de que todo continúa funcionando perfectamente tras una modificación, hacen que esto no cause ningún inconveniente. La definición de estos estándares es decisiva para poder plantear con éxito la propiedad colectiva del código. La colectividad del código no tendría sentido sin una codificación basada en estándares que haga que para todos los integrantes del equipo no se manifiesten diferencias en cuanto al código escrito por cualquier otro miembro del equipo. Uno de los efectos significativos que provoca la colectividad del código es que la parte de este que es complejo no permanece mucho tiempo, ya que cada programador tiene acceso completo al sistema aquel que lo identifique y tenga la posibilidad de simplificarlo debe hacerlo.

A cada código implementado se le debe aplicar refactorización, técnica que consiste en dejar el código en el estado más simple posible de forma que no pierda ni gane funcionalidad y que se sigan ejecutando correctamente todas las pruebas. Esto permite que en el momento en que sea necesario

agregar o modificar una funcionalidad sea mucho más flexible, permitiendo a los equipos de programadores mejorar el diseño del sistema a través de todo el proceso de desarrollo. Su objetivo es mantener un sistema enfocado a proveer el valor de negocio mediante la minimización del código duplicado y/o ineficiente. De igual forma después de que el código es refactorizado se le deben aplicar nuevamente las pruebas para comprobar que todo sigue funcionando adecuadamente.

Este código será integrado continuamente. Cada pocas horas o al cabo de un día de programación, como máximo, se integra el sistema completo Para ello existirá un máquina así llamada, de integración, a la que se acercará una pareja de programadores cada vez que tengan una clase que haya sido probada unitariamente. Si al añadir la nueva clase junto con sus pruebas unitarias, el sistema completo sigue funcionando correctamente los programadores darán por finalizada esa tarea. Si no, serán los responsables de dejar el sistema de nuevo con las pruebas funcionando al 100%. Si después de un cierto tiempo no son capaces de descubrir qué es lo que falla, se desecha ese código y se comienza nuevamente. Esta integración reduce el riesgo del proyecto, pues si se tienen diferentes ideas sobre alguna operación esto puede ser solucionado rápidamente cuando se produce la integración. Durante la integración pueden surgir diferentes colisiones entre las clases y los métodos implementados pero esta situación es resuelta a través de las pruebas. Por lo que se debe tener una herramienta que soporte una rápida integración y las pruebas necesarias para verificar que lo que se está integrando funciona.(ACEBAL and LOVELLE 2002)

Los programadores deben trabajar un promedio de 40 horas semanales, teniendo en cuenta que la sobrecarga de trabajo conlleva al cansancio del personal y provoca que se pierda calidad en la producción. Esta no es una medida rígida en cuanto a que no se puedan trabajar horas extras, pero si que sean solo aquellas que sean realmente necesarias pero esto no significa una cantidad de días excesiva. Trabajar días extras puede ser un síntoma de que algo anda mal. Naturalmente, las 40 horas no es una regla fija, puede variar de 35 a 45. Esta definición es con el objetivo de mantener al equipo de trabajo vinculado a la producción el tiempo que estos pueden aportan un valor real al software y con calidad, no tiene sentido se continúe trabajando si se está exhausto, pues no se logrará un buen resultado.

Cada vez que se consigue codificar y que funcione una historia de usuario, se le da al cliente para que la revise, la pruebe y añada las posibles modificaciones para las siguientes versiones. Cuando se realiza un versión completa (compuesta por varias de las historias de usuario), incluso se entrega al

cliente final para que empiece a trabajar con ella y reporte incidencias o mejoras. Este ciclo se va repitiendo una y otra vez hasta que el cliente se de por satisfecho y cierre el proyecto.

Durante todo el transcurso del proceso de producción del software la documentación que se produce es la mínima necesaria. Todo con motivo de aprovechar el mayor tiempo posible tratando de suministrarle al cliente un producto que pueda evaluar tangiblemente y que satisfaga sus necesidades. En este caso las tarjetas de historias de usuario y de tareas son la documentación que mayor objetividad tiene.

2.8 Evaluación y control.

La evaluación y control del proyecto se basa en una serie de pequeñas reuniones informales realizadas cada día. Todas con el objetivo de fomentar la comunicación, para lograr que los desarrolladores puedan expresar los problemas a los que se enfrentan durante el desarrollo del software y para verificar cómo va el desempeño de sus tareas. En esta cada miembro describe en que está trabajando, que está funcionando bien, los problemas que pueden existir, el trabajo que se ha ido terminando y lo con que podría necesitar ayuda. Al final de cada iteración, se celebra otra reunión en la que se repasa lo que se ha hecho bien y en lo que se ha de trabajar en la próxima iteración. (COCKBURN 2006)

Encargado de seguimiento es el que está más relacionado con el control de las actividades realizadas. Su labor es comprobar la relación entre lo planificado y lo realizado, comunicando los resultados obtenidos para tener en cuenta en las estimaciones siguientes. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

2.9 Fin

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

3.4 Conclusiones parciales

En este capítulo se describió la estructura del esquema que representa la referencia desarrollada a través de las fases que la misma consta y bajo las cuales la organización debe estar dirigida para estructurar su proceso de desarrollo de software. Cada una de estas fases define una serie de pasos que explican exactamente lo que según las condiciones de la organización se debe hacer, donde la unión de todos estos tienen el objetivo de suministrar a dicha organización una guía para organizar el proceso de producción de software bajo la aplicación de una metodología ágil para los proyectos de desarrollo.

Capítulo 3: VALIDACIÓN DE LA REFERENCIA PARA LA APLICACIÓN DE LA METODOLOGÍA ÁGIL PROGRAMACIÓN EXTREMA A PROYECTOS DESOFT SS.

Una vez desarrollada la referencia para la aplicación de la metodología ágil Programación Extrema, es necesario validar su funcionamiento. Con este objetivo se seleccionó la empresa DeSoft Sancti – Spíritus y se utilizó la referencia en un proyecto de desarrollo de software. Paralelamente se desarrolla otro proyecto aplicando una metodología tradicional donde al final se analizan los resultados obtenidos llegando a conclusiones que suministran valor para futuras aplicaciones.

Con el fin de validar la referencia desarrollada se hace necesario seleccionar 2 proyectos para ser ejecutados paralelamente, uno utilizando la referencia desarrollada como guía para la aplicación de la metodología ágil Programación Extrema y otro utilizando la metodología tradicional que se emplea en Desoft. Para llevar a cabo la elección de estos proyectos se selecciona un grupo de características que a partir de las cuales se puedan establecer parámetros de similitud a tener en cuenta para elegir dichos proyectos que serán ejecutados. Todo con el objetivo de a través del progreso del desarrollo del software marcar determinadas diferencias para establecer una comparación y obtener resultados que permitan demostrar con cuál de ambos procesos utilizados se obtiene mayor eficiencia en base a las características y necesidades de la empresa.

3.3 Características de similitud entre proyectos a ejecutar

Es necesario garantizar que inicialmente ambos proyectos a desarrollar se encuentren relativamente bajo las mismas condiciones con el objetivo de mostrar a través del desarrollo de ambos las diferencias que van surgiendo en base a la estrategia de desarrollo utilizada y así poder verificar la trascendencia de la utilización de la referencia para la aplicación de una metodologías ágiles en relación con la metodología tradicional que supone Desoft. En este sentido se deben elegir un conjunto de indicadores a tener en cuenta para establecer dicho objetivo. Los indicadores que se eligieron en esta investigación son los siguientes:

1. Complejidad del sistema a desarrollar: La complejidad del sistema se mide en cuanto a la cantidad de tablas que son generadas, cantidad de pantallas de entrada de datos, cantidad de

reportes de salida, cantidad de validaciones por pantalla de entrada de datos, cantidad de requisitos y complejidad de los procedimientos de cálculo. Esta se divide de acuerdo a 3 categorías:

Complejidad	Tablas	Pantallas	Reportes	Validac/pantalla	Complejidad proced. de cálculo	Requisitos
Alta	+100	+20	+50	5	Alta	+125
Media	+50-100	+10 -20	+25 -50	3	Medio	+60-125
Baja	-50	-10	-25	-3	Bajo	-60

Tabla 3.1 Complejidad del sistema

2. Cantidad de desarrolladores que conforman el equipo: Se establece la cantidad de personas que conformaran el equipo de desarrollo de acuerdo a las características que requiera el proceso de desarrollo. La diferencia entre la cantidad de personas entre un equipo de desarrollo y otro debe ser de una persona.
3. Conocimientos de los integrantes del equipo de desarrollo: Cada uno de los integrantes debe tener dominio general de las plataformas de desarrollo sobre las que se implementa el sistema y de los temas complementarios para el desarrollo del mismo.
4. Tiempo estimado para el desarrollo del software: El tiempo para el desarrollo del software se divide en 3 categorías:
    - Corto: - 7 meses
    - Medio: +7 -1,5 año
    - Largo: +1,5 año
5. Tecnología: En cuanto a la tecnología se tienen en cuenta los requerimientos de hardware y requerimientos de software que se utilizan para la producción del software. Debe existir la relación hombre-máquina en base a la técnica utilizada para la implementación del código donde cada una de las máquinas tengan condiciones técnicas similares.

Es de suma importancia aclarar que cada una de los parámetros establecidos a través de datos cuantitativos para enmarcar las características de similitud correspondientes a los proyectos a desarrollar fueron establecidos teniendo en cuenta las técnicas, prácticas y experiencia de desarrollo que utiliza la empresa Desoft SS para su producción de software.

3.4 Características de diferenciación entre los proyectos a ejecutar

Con el fin de lograr obtener una comparación que muestre valores significantes para demostrar la efectividad de la referencia desarrollada para la aplicación de metodologías ágiles respecto al uso de una metodología tradicional se debe garantizar un monitoreo del comportamiento de diferentes parámetros durante el transcurso de los procesos involucrados en cada uno de los proyectos. Este registro debe servir de base para establecer comparaciones que permitan tomar decisiones sobre la bondad o debilidad de dicha referencia con respecto a una metodología tradicional. En cuanto a esto se eligen un conjunto de indicadores que permitan llevar a cabo dicho objetivo. Los indicadores que se eligieron en esta investigación son los siguientes:

1. Tiempo real de ejecución del proyecto.
2. Tiempo invertido en reunión con el cliente
3. Cantidad de funcionalidades suministradas al cliente
4. Cantidad de documentación generada
5. Cantidad de pruebas unitarias

Cada uno de estos aspectos se seleccionó teniendo en cuenta que estos tienen un peso significante durante todo el proceso de desarrollo y a partir de los cuales se pueden obtener conclusiones que sustenten la investigación realizada donde se podrán tomar decisiones que permitirán a la empresa dirigir su desarrollo según las necesidades y condiciones de la empresa.

3.5 Elección de los proyectos a ejecutar

Después de tener las características en base a la semejanza que se debe tener en cuenta para los proyectos a ejecutar se pasa a la elección de los mismos con el fin de poner en práctica la referencia para la aplicación de la metodología ágil Programación Extrema y analizar sus resultados. Se seleccionaron 2

proyectos para su ejecución, uno de estos será desarrollado a través de la utilización de dicha referencia y otro utilizando la metodología tradicional que se utiliza en la empresa Desoft.

Para ello debe aclararse que los proyectos seleccionados representan Sistemas de Gestión de Bases de Datos teniendo en cuenta que ambos deben estar enmarcados en el mismo ámbito debido a que en estos deben presentar características bastantes similares para poder establecer una comparación en cuanto al proceso de ejecución de los mismos.

Los proyectos correspondientes a llevar a cabo son:

1. Control de boletines en Astro: Es un sistema para controlar los boletines que son vendidos ya sea por reservación y por lista de espera y aquellos que unas vez vendidos pueden ser reintegrados o anulados con los % de rebajas que se aplican. Se generan una serie de reportes de salida relacionados con la información de entrada. Este será desarrollado bajo la referencia para la aplicación de metodologías ágiles.
2. Control de la información del CITMA: Es un sistema para gestionar la información relacionada con los trabajos que se presentan en los forum, los niveles de generalización y aplicación. Además de los centros e investigadores implicados en los trabajos para poder contar con un histórico de estas informaciones. Este será desarrollado bajo la referencia metodología tradicional que emplea Desoft.

Es necesario aclarar que el proyecto que se desarrollará utilizando dicha referencia es de baja complejidad ya que como la estrategia de desarrollo utilizada es nueva se requiere comenzar primero con proyectos sencillos con el objetivo de que el equipo de desarrollo se vaya adaptando y tomando habilidades en cuanto a esta nueva forma de desarrollo y poco a poco ir escalando en el proceso de desarrollo.

3.6 Características iniciales de los proyectos seleccionados

En la tabla que a continuación se muestra se describen las características iniciales que presentan cada uno de los proyectos seleccionados con el objetivo de ir observando la evolución que cada uno de estos proyectos tendrán bajo determinada estrategia de desarrollo y de acuerdo a esto arribar a conclusiones.

El proyecto de Control de Boletines de Astro se desarrollará bajo la referencia desarrollada para la aplicación de metodologías ágiles mientras que el proyecto de Control de la información del CITMA se implementará utilizando la metodología tradicional utilizada por la empresa Desoft SS.

Características iniciales		
Característica	Control de boletines en Astro	Control de la información del CITMA
Complejidad del sistema a desarrollar	56 requisitos, 48 tablas, 9 pantallas de entrada, 23 reportes de salida, aproximadamente, 18 validaciones.	53 requisitos, 45 tablas, 8 pantallas de entrada, 22 reportes de salida, aproximadamente, 16 validaciones.
Conocimientos de los integrantes del equipo de desarrollo	SQL Server, Delphi Conocimiento general del funcionamiento de Astro en cuanto al control de los boletines. Metodologías ágiles.	SQL Server. Delphi Conocimiento general de la información gestionada sobre los trabajos investigativos presentados en los forum del CITMA.
Cantidad de desarrolladores que conforman el equipo	6	7
Tiempo estimado para el desarrollo del software	6	
Tecnología	3 máquinas en total, 1 por pareja de programadores. Máquinas Intel, Pentium 4, CPU 2.40 GHz, 512 MB de RAM. SO Windows XP, SQL Server. Delphi 2005/2006 –Delphi Win32, DUnit	7 máquinas en total, 1 por cada programador individual. Máquinas Intel, Pentium 4, CPU 2.40 GHz, 512 MB de RAM. SO Windows XP, SQL Server. Delphi 2005/2006 –Delphi Win32-

Tabla 3.2 Características iniciales de los proyectos a desarrollar

A continuación se explica cómo se desarrollaron cada una de las características representadas anteriormente a través de la referencia para la aplicación de la metodología ágil Programación Extrema con el fin de mostrar el proceso que se lleva a cabo mediante la misma.

Después de haber realizado la caracterización y evaluación de la empresa Desoft para aplicar metodologías ágiles y llegar a resultados que indican las posibilidades de aplicar agilidad se conformó el equipo de desarrollo basándose en cada uno de los aspectos que propone la referencia desarrollada, teniendo en cuenta que la comunicación entre los miembros del equipo debe ser la más factible posible y es uno de los aspectos fundamentales a garantizar, además de tener un coeficiente intelectual desarrollado que le permita adaptarse y desempeñarse adecuadamente de acuerdo a las condiciones de trabajo, con habilidades y experiencia en el desarrollo de software. Donde la organización del equipo de desarrollo debe ser adaptable y que facilite la cooperación. Se escogieron los especialistas en el tema que capaces de transmitir a cada uno de los integrantes del equipo de desarrollo el funcionamiento que se lleva a cabo en Astro para el control de los boletines así como el los temas correspondientes al aspecto técnico, en este caso al tratamiento de los sistemas de base de datos y del uso de las metodologías ágiles. Además se determina que el equipo tiene condiciones para llevar a cabo este proyecto debido a que se tiene experiencia y habilidad en proyectos similares anteriormente desarrollados en la empresa según el inventario de proyectos realizado. En el equipo los programadores son distribuidos por parejas.

Para lograr que la preparación técnica de cada uno de los integrantes del equipo de desarrollo se encuentre equilibrada se llevó a cabo el proceso de nivelación en cuanto a algunos aspectos de la seguridad que emplea SQL Server ya que se existían algunas dudas de cómo llevar a cabo dicha operación por parte de algunos desarrolladores. Esta nivelación se realizó por parte del especialista seleccionado en dicho tema a través de un curso preparado según las actividades requeridas con el objetivo de lograr una mayor eficiencia en el proceso de desarrollo debido a que de este modo el trabajo se hace con mayor fluidez y rapidez ya que se esta trabajando sobre actividades conocidas.

También se necesitó del proceso de capacitación en cuanto a las herramientas para realizar las pruebas automáticas ya que estas eran generalmente poco conocidas por los desarrolladores. Este es de gran importancia es necesario tener el conocimiento necesario ya que es algo complejo y tiene un gran peso. Además de la capacitación respecto al uso de la metodología ágil Programación Extrema ya que es una nueva estrategia de desarrollo que la empresa estableció utilizar y por tanto los desarrolladores no tienen el conocimiento suficiente para emplearla. En este se definen las prácticas ágiles a ser utilizadas.

Luego de realizada cada una de las actividades anteriormente descritas se comienza según la referencia la planificación como tal del proyecto y seguidamente cada uno de los pasos que plantea la dicha referencia desarrollada hasta la terminación del proyecto.

3.7 Análisis de los resultados al aplicar una metodología ágil contra el uso de una metodología tradicional en 2 proyectos de desarrollo de software en Desoft SS.

Después de llevar a cabo la ejecución de ambos proyectos siguiendo cada uno de los pasos según la estrategia de desarrollo seleccionada de acuerdo al proyecto se evaluaron los indicadores que se manifestaron de forma diferente en cada uno de los proyectos desarrollados. A continuación se muestra cómo se desarrollo el comportamiento de estos parámetros en cada uno de los proyectos desarrollados de acuerdo a los indicadores iniciales que fueron establecidos.

3.7.1 Proyecto de Control de boletines en Astro

1. Tiempo real de ejecución del proyecto: 4.5 meses de ejecución.
2. Tiempo invertido en reunión con el cliente: Este parámetro se encuentra en cero debido a que se logró que un representante del cliente se sintiera como parte del equipo de desarrollo exponiéndole las ventajas que esto podría traer por lo que estuvo disponible la mayor parte del tiempo de desarrollo para aclarar cualquier dificultad de acuerdo a alguna especificidad o desacuerdo en alguna funcionalidad que deba tener el sistema.
3. Cantidad de funcionalidades suministradas al cliente: cada 3 semanas como máximo el cliente recibe una versión pequeña de lo que se ha implementado de acuerdo a los requisitos especificados. Esta versión el cliente la puede ir probando a través de las pruebas funcionales que estos mismos realizan para verificar su funcionalidad y que satisface sus necesidades. Esta representa una funcionalidad del sistema como un todo. A los 2,5 meses se hace una entrega formal del producto.
4. Cantidad de documentación generada: la principal documentación que se genera apartando al código producido, son las tarjetas de historias de usuarios y las tareas de programación donde se recogen los requerimientos del cliente y se hace una especificación de cada una de las

tareas en que son divididas para la implementación las historias de usuario respectivamente. Se generan también algunos pequeños diagramas para representar y esclarecer algunas actividades pero estos no constituyen documentación ya que a través de estos no se puede obtener una concreta retroalimentación en el sentido que a través de estos no se puede conocer si las pruebas desarrolladas funcionan ni si el código que este soporta es simple, esto solo se puede obtener mediante la codificación. Por lo tanto cada uno de estos está respaldado por el código correspondiente y entonces no constituyen documentación después de haberse generado el código. Aunque hay que aclarar que si algún tipo de código es mejor expresado a través de un diagrama de debe mantener entonces como tal y ser manejado por las herramientas CASE

5. Cantidad de pruebas: se realizan pruebas unitarias por parte de los programadores por cada nueva clase y por parte del cliente se realizan las pruebas de aceptación para comprobar la funcionalidad del sistema.

3.7.2 Control de la información del CITMA

1. Tiempo de ejecución del proyecto: 6 meses de ejecución
2. Tiempo de reunión con el cliente: 1 reunión semanal (cada lunes) de 4 horas para un total de 104 horas.
3. Cantidad de funcionalidades suministradas al cliente: todas las versiones que se hacen no constituyen versiones que suministran funcionalidad al cliente.
    - 1ra versión – Diagrama de Clases (2 semanas)
    - 2da versión – Prototipos de interfaz (4 semanas)
    - 3ra versión – Pantalla de entradas con funcionamiento (9 semanas)
    - 4ta versión – Reportes de salida (6 semanas)
    - 5ta entrega final – Sistema completo (2 semanas)
4. Cantidad de documentación generada: Se obtiene el documento de captura de requisitos, la planificación del proyecto, los diagramas de caso de uso, actividad, clases, secuencia, caso de uso del sistema, despliegue.

5. Cantidad de pruebas: Se le realizan pruebas a las versiones que representan funcionalidad para el cliente, por lo que estas son a un largo plazo de acuerdo a cuando se implemente dicha funcionalidad según la planificación. En este caso a partir de la 3ra entrega.

3.7.3 Comparación de los resultados

Después de describir el comportamiento de los diferentes indicadores que se especificaron como base para la comparación en cada uno de los proyectos desarrollados se comparan los resultados obtenidos.

Es necesario aclarar que la variabilidad de los requisitos a través de la realización de ambos proyectos no fue significante ya que estos son proyectos de poco complejidad aunque esta sea uno de los factores principales de aplicar MAs pero si es necesario aclarar que el proceso esta estructurado de forma tal que este sea adaptable y que se adecue a las necesidades de cualquier otro proyecto que necesite de un proceso adaptable.

El tiempo de ejecución para ambos proyectos se estimó que sería de 6 meses al tener características similares teniendo en cuenta la técnica de desarrollo empleada por Desoft para tener una base de cómo sería el comportamiento de cada uno de estos proyectos y así poder establecer un margen cuantitativo de lo que puede significar el tiempo que realmente cada proyecto utilizó. Se tiene entonces que el proyecto del Control de los boletines de Astro se desarrolló en 1,5 meses menos que el de Control de la información de los proyectos del CITMA. Aquí teniendo en cuenta que en el proyecto correspondiente a Astro se empleó tiempo en la capacitación de las prácticas ágiles y al uso de las herramientas de prueba automática, además de nivelar al equipo de desarrollo. No obstante se muestra un tiempo de desarrollo más corto para el proyecto de Astro.

Tiempo de reunión con el cliente: En el proyecto de Control de los boletines de Astro no se considera este tiempo, mientras que para la realización del proyecto de Control de la información de los proyectos del CITMA se emplearon 104 horas en total para aclarar con el cliente cada una de los requerimientos. En el proyecto de Astro el cliente (representante del cliente) se mantuvo la mayor parte del tiempo de desarrollo como miembro del equipo por lo que estuvo todo el tiempo en contacto con cada una de las actividades realizadas. Este esclarece las ideas de los programadores cuando existe algún tipo de discordancia en algún aspecto de la funcionalidad del software lo que es muy positivo en el sentido de que se logra una mayor comunicación entre cliente – desarrollador a través de la cual se puede responder

mucho mejor a las necesidades que el cliente plantea donde cada uno de los cambios necesarios que se pueden hacer se logran incluso antes de que alguna funcionalidad sea implementada logrando una mayor reutilización del tiempo de desarrollo y una mayor productividad.

Cantidad de funcionalidades entregadas al cliente: Las funcionalidades entregadas en el proyecto de Control de los boletines de Astro son mayores que en el proyecto Control de la información de los proyectos del CITMA. El número de funcionalidades entregadas al cliente en el proyecto de Astro son como promedio 6 pequeñas versiones y 2 entregas formales. Esto posibilita que se vaya chequeando constantemente el progreso del sistema y que cumpla con los requerimientos que el cliente plantea. Mientras que siguiendo el otro proyecto todas las versiones que se consiguen no suministran una funcionalidad al cliente por lo que en ocasiones este no puede comprobar si lo que se ha hecho corresponde con lo que se necesita, solo a partir de la tercera versión el cliente recibe alguna funcionalidad. La entrega constante de versiones garantiza que el sistema que se esta produciendo responde a lo estipulado por parte del cliente.

Cantidad de documentación generada: En el proyecto desarrollado utilizando la referencia (Control de los boletines de Astro) es generada más documentación que en el proyecto utilizando la metodología que emplea la empresa Desoft (Control de la información de los proyectos del CITMA). En el primero solo se tiene en cuenta lo que se realmente se necesita para llevar a cabo la producción de dicho software recogiendo estos datos en las historias de usuarios las que hacen una pequeña descripción de cada requisito por parte del cliente y donde estos más tarde son divididos en tareas de programación. Mientras que en el otro en ocasiones la documentación que se obtiene es abundante en relación con la complejidad del proyecto que se desarrolla debido a la cantidad de diagramas que según anteriormente se especificó.

Cantidad de pruebas unitarias: El número de pruebas implementadas en el proyecto de Control de los boletines de Astro es mucho mayor que en el proyecto de Control de la información de los proyectos del CITMA. Puesto que siguiendo la referencia cada vez que se genera una porción de código esta es comprobada mediante las pruebas unitarias además de que el cliente por su parte realiza las pruebas de aceptación con el objetivo ambas de ir comprobando si lo que se esta implementando esta correcto y así disminuir el número de errores obteniendo un sistema mucho más seguro mientras que bajo el otro proyecto las pruebas son a un plazo un poco más largo lo que puede traer como consecuencia que algunos errores sean encontrados un poco tarde y que el coste del cambio en alguno de esto sea muy elevado. Las constantes pruebas garantizan, además, que cada uno de los muchos cambios en los

requerimientos que pueden ser generados durante el proceso de desarrollo del software, al tener las posibilidades de ser un proceso adaptable, responden a través de estas a los nuevos requerimientos del cliente al poder comprobar rápidamente las nuevas funcionalidades creadas.

Cada uno de los aspectos que anteriormente se establecieron fueron tomados para demostrar con mayor claridad la evolución y desempeño de la utilización de la referencia desarrollada pero es válido aclarar que a medida que se fue aplicando dicha propuesta uno de los aspectos necesarios a tener en cuenta para ser modificados según las encuestas realizadas era establecer una mayor comunicación entre cada uno de los integrantes del equipo de desarrollo, aspecto que se mejoró considerablemente demostrado en los resultados obtenidos ya que sin esta muchos de esas prácticas no se hubiesen podido consolidar y obtener dichos resultado, así como establecer una organización interna flexible y un modo de trabajo colaborativo aunque se considera que esta todavía debe mejorar si se espera trabajar bajo un proceso ágil.

Otro aspecto a considerar después de la aplicación de dicha referencia en el proyecto de Control de boletines de Astro es lo referido a la práctica de la programación en parejas donde dicha práctica fue realizada de forma evolutiva a medida que avanzaba el proyecto integrando poco a poco a aquellas parejas que tenían mayor compatibilidad inicialmente debido a que es una técnica que requiere de un gran entendimiento entre los desarrolladores. Todavía hay que hacer un trabajo mayor para que dicha práctica logre completamente los resultados que esta propone.

3.8 Conclusiones parciales

A través del desarrollo de un proyecto bajo la aplicación de la referencia desarrollada y su comparación con otro proyecto de características similares bajo la estrategia de desarrollo que emplea la empresa Desoft SS, se obtiene que en base a las condiciones que presenta dicha institución la aplicación de esta referencia logra resultados satisfactorios de acuerdo a cada uno de los indicadores que fueron establecidos teniendo en cuenta que en general los proyectos que se desarrollaron son proyectos de poca complejidad con el objetivo de tener una base de las posibilidades que representa la utilización de esta referencia para aquellos que pretenden adentrarse en el mundo de la agilidad en este caso la empresa Desoft SS con el fin de resolver los problemas que esta presenta con la utilización de su estrategia de desarrollo bajo nuevas situaciones. Se demuestra que si se tienen las condiciones para implantar agilidad al proceso de desarrollo de determinado software, estas metodologías ágiles se pueden aplicar y ajustarse

en muchas ocasiones de una mejor manera a las condiciones de desarrollo y a las necesidades de los clientes y desarrolladores, en este caso incluso más que la metodología empleada actualmente por la empresa.

Conclusiones generales

De manera general a través de todo el estudio realizado se concluye que es de gran importancia tener en cuenta la metodología de desarrollo que se establezca utilizar debido a que a partir de dicha elección depende el resultado de todo el proceso de producción y de lograr los resultados satisfactorios esperados tanto para el cliente como para el equipo de desarrollo.

Se comenzó por realizar un estudio detallado primeramente en base a las generalidades de de las metodologías ágiles y una comparación de las mismas con las metodologías tradicionales con el fin de enmarcar cada uno de los aspectos en que ambas metodologías presentan inconcordancia para así poder establecer las ventajas que representa una sobre la otra de acuerdo al proceso de desarrollo de software a que se adaptan. Luego se realizó una caracterización de cada una de las metodologías ágiles más utilizadas a nivel mundial, con el fin de seleccionar la que se utilizaría como base para la solución propuesta así como los tipos de proyectos donde estas metodologías pueden ser aplicadas y algunas de las empresas que las utilizan todo con el objetivo de esclarecer cada uno de los aspectos que se debían tener en cuenta para la referencia a desarrollar. Toda la investigación teniendo en cuenta la importancia de seleccionar la metodología adecuada según el proceso de desarrollo que se establezca.

Dando cumplimiento al primer objetivo se obtienen datos cuantitativos a través del diagnóstico realizado en la empresa Desoft que permiten conocer las condiciones existentes en dicha entidad y las posibilidades de mejoras, donde se refleja la necesidad de establecer una nueva estrategia de desarrollo para la empresa y de la creación y aplicación de una referencia que encamine dicho desarrollo mediante el uso de una metodología ágil como nueva estrategia a utilizar.

El segundo objetivo se logra a través de la creación de dicha referencia para la aplicación de la metodología ágil Programación Extrema, la cual logra guiar al equipo de desarrollo bajo la utilización de una metodología ágil para el proceso de desarrollo de software. Esta se concibe a través de fases que debe tener en cuenta la organización para establecer dicha aplicación, fases que a su vez determinan una serie de pasos describiendo detalladamente las actividades que cada uno de los mismos propone. Dicha referencia se sustenta en la metodología ágil Programación Extrema de Kent Beck de acuerdo a las prácticas que la misma propone para el desarrollo de software que satisfacen las necesidades y se adecuan a la empresa Desoft en base a la problemática a solucionar.

Teniendo en cuenta el vencimiento del tercer objetivo se puso en práctica la referencia desarrollada en uno de los proyectos de la empresa Desoft para la cual fue concebida al mismo tiempo que se desarrollaba otro proyecto de acuerdo a la práctica de desarrollo estándar de la empresa. A partir de dicha aplicación se estableció una comparación en cuanto a algunos aspectos de ambos desarrollos donde se obtuvieron resultados que manifiestan que dicha referencia desarrollada presenta aspectos válidos en su aplicación y que su utilización puede contribuir a un mayor rendimiento en el proceso de desarrollo de Desoft.

Recomendaciones

Después de tener una visión general de lo que representa el proceso de desarrollo de software bajo el uso de una metodología ágil a través de la referencia desarrollada se debe profundizar mucho más en cada una de las prácticas de desarrollo que esta propone y las herramientas a utilizar para implementar cada una de las mismas, ya que la referencia solo muestra un marco general para aquellos principiantes con el fin de que a partir de esta se observe como se puede desarrollar un proyecto utilizando MAs y a partir de ese punto tomar como base la implantación de la agilidad para el desarrollo del software si estas se adecuan a las condiciones de la empresa en cuestión.

De acuerdo a lo anterior se propone extender la utilización de la referencia desarrollada hacia otras entidades que pretendan hacer uso de las metodologías ágiles para el proceso de producción de software, empleando dicha referencia como base inicial de aplicación de dichas metodologías.

Además de considerar la utilización de la referencia propuesta en la Universidad de Ciencias Informáticas para los proyectos productivos que en la misma se desarrollan haciendo algunas adaptaciones teniendo en cuenta que la mayoría del personal vinculado a la producción es estudiantado y no tiene la experiencia suficiente y que se desempeñan en dos actividades fuertes, tanto en la labor docente como productiva.

También se sugiere aplicar dicha referencia a una mayor cantidad y variedad de proyectos con el fin de demostrar su validez y efectividad en los procesos de desarrollo de software.

Bibliografía

1. ACEBAL, C. F. and J. M. C. LOVELLE. *Extreme Programming (XP):un nuevo método de desarrollo de software*. *Novática,*http://www.ati.es/novatica/, 2002.

2. BECK, K. *Agile Manifesto*,http://www.agilemanifesto.org/, 2001.
3. ---. *Extreme Programming Explained. Embrace Change*. Addison-Wesley, 1999. p.
4. ---. *Extreme Programming Explained. Embrace Change*. 2000. p.

5. BECK;, K. and M. FOWLER. *Planning Extreme Programming*. 2000. p.

6. BERRUETA, D. *Programación extrema y software libre*, 2006.

7. COCKBURN, A. *Agile Software Development: The Cooperative Game*. 2006. p.
8. ---. *Crystal Clear. A human-powered methodology for small teams, including The Seven Properties of Effective Software Projects*. 2004. p.

9. FOWLER, M. *La Nueva Metodología* 2003.

10. GALLO, E. and M. VERGARA. *Metodologías de Desarrollo de Software Ágiles*. *European Software Institute*, http://www.esi.es/Berrikuntza, 2006.

11. GONZÁLEZ, J. F. *7 paradigmas a romper para ser ágiles* 2006.

12. GRACIA, J. *Gestión de proyectos con SCRUM*, 2006.

13. HIGHSMITH, J. *Agile Software Development Ecosystems*. Addison Wesley, 2002. p.
14. ---. *Extreme Programming*. 2000. p.

15. HOLLAND, J. *Hidden Order: How Adaptation builds Complexity*. Addison Wesley., 1995. p.

16. JEFFRIES R; ANDERSON A*, et al. Extreme Programming Installed*. 2001. p.

17. LETELIER, P. and M. C. PENADÉS. *Métodologías ágiles para el desarrollo de software: Extreme Programming (XP)*2006.

18. MATERÓNA, J. J. R. *Los procesos de aprendizaje en las metodologías ágiles de desarrollo*, 2005.

19. NAVEGAPOLIS. *Microsoft Solutions Framework, CMMI y los Métodos Ágiles* www.navegapolis.net, 2006a.
20. ---. *¿Usas modelos ágiles para desarrollar software?* , www.navegapolis.net, 2006b.

21. PALACIO, J. *Gestión y modelos para la eficiencia en empresas de desarrollo de software*, 2005.

22. PEKKA ABRAHAMSSON, O. S., JUSSI RONKAINEN,JUHANI WARSTA. *Agile Software Development Methods*. Suecia, VTT Publications 478, 2002. p.

23. POPPENDIECK, M. *Lean Programming*, http://www.agilealliance.org/articles/articles/LeanProgramming.htm,, 2001.

24. REYNOSO, C. *Métodos Heterodoxos en Desarrollo de Software* 2004.

25. SAAVEDRA, A. B.; F. J. A. RODRIGUEZ*, et al. Software Process Improvement trough Extreme Programming practices*, 2006.

26. SÁNCHEZ, J. P. *Metodologías Ágiles: La ventaja competitiva de estar preparado para tomar*
27. *decisiones lo más tarde posible y cambiarlas en cualquier momento*, http://www.agile-spain.com 2004.

28. SCHWABER, K. and M. BEEDLE. *Agile software development with Scrum*. 2002. p.

29. SOLÍS., M. C. *Una explicación de la programación extrema (XP)*, http://www.apolosoftware.com/, 2003.

30. STAPLETON, D. C. Y. J. *DSDM: Business Focused Development*. 2a edición Addison-Wesley, 2003. p.

31. WAKE, W. C. *Extreme Programming Explored* 2000. p.