ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"Sistema de Calidad para proyectos de Desarrollo de Software enfocado en la metodología de Programación Extrema (XP)"

TESIS DE GRADO

Previo a la obtención del Título de:

MAGISTER EN SISTEMAS DE INFORMACIÓN GERENCIAL

PRESENTADA POR:

OMAR FERNANDO GUZMAN ROSERO

GUAYAQUIL - ECUADOR

2012

AGRADECIMIENTO

A todas aquellas personas que en un momento determinado creyeron en mí y me dejaron ejemplos dignos de superación y entrega, porque en gran parte gracias a todas Ellas hoy puedo alcanzar mi meta. A Dios por darme la fortaleza y la sabiduría en los momentos difíciles para seguir adelante.

Omar Guzmán

DEDICATORIA

A mi esposa que supo comprender y apoyar mi deseo de superación y a mis padres por haberlo fomentado.

Omar Guzmán

TRIBUNAL DE GRADO

Ing. Lenin Freire

DIRECTOR DE TESIS

Ing. Carlos Martin

DELEGADO DEL TRIBUNAL

DECLARACIÓN EXPRESA

La responsabilidad del contenido de este Trabajo Final de Graduación me corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL.

(Reglamento de Graduación de la ESPOL)

AUTOR DE LA TESIS

_____

Omar Fernando Guzmán Rosero

RESUMEN

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es 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. Este enfoque está mostrando 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 están revolucionando la manera de producir software, y a la vez generando un amplio debate entre sus seguidores y quienes por escepticismo o convencimiento no las ven como alternativa para las metodologías tradicionales. En este trabajo se presenta resumidamente el contexto en el que surgen las metodologías ágiles, sus valores, principios y comparación con las metodologías tradicionales. Además se describen brevemente las principales propuestas, especialmente Programación Extrema (Extreme Programming, XP) la metodología ágil más popular en la actualidad.

INDICE GENERAL

CAPÍTULO 1

GENERALIDADES

1.1. INTRODUCCION

Con el rápido avance de las metodologías ágiles de desarrollo, algunos desarrolladores sienten que la norma ISO 9001 y otros estándares de aseguramiento de la calidad han llegado a ser irrelevante o al menos no tan necesario. La idea parece ser que un proceso de certificación ISO 9001 es incompatible con el proceso ágil de desarrollo de software. Nuestro objetivo es demostrar que es posible que ambos procesos puedan unificarse y no convivir por separados y que ambos podrían traer beneficios si se logra una combinación de ellos.

La necesidad por un continuo proceso de mejoramiento para asegurar a la organización la entrega repetitiva de entregables de productos de software que cumplan con las necesidades del cliente ha ido en importante crecimiento, el cual ha conducido al desarrollo de software hacia la búsqueda de nuevos modelos de mejoramiento de procesos, entre estos modelos están los estándares ISO.

Muchos potenciales clientes requieren que la compañía que desarrolla software tenga un certificado de calidad que asegure que el producto final cumpla con los estándares de calidad requeridos antes de cerrar un contrato. Requerimiento cada vez más adoptado tanto por las empresas privadas como las públicas. La principal razón para esto es el nivel de confianza creado por un certificado ISO 9001. Es mucho más fácil verificar que una compañía tenga un certificado ISO 9001 a que la misma tenga un buen proceso de desarrollo que garantice la calidad del producto final. Además, otra de las razones para certificar un proceso ágil se debe a que muchas compañías ya tienen un certificado ISO 9001 y quieren mantener el mismo integrando el proceso de desarrollo ágil a sus procesos de desarrollo convencionales.

Este trabajo de investigación se enfoca en el desarrollo usando la metodología de Extreme Programming (programación extrema) un ligero acercamiento al desarrollo de software orientado al cambio y orientado al cliente, el cual es ampliamente usado en las metodologías ágiles. Extreme programming también pone énfasis en la importancia del cliente en sitio, comunicación oral, calidad

del producto, cortas retroalimentaciones del cliente y del usuario final, simplicidad, mínima documentación y la ausencia de sobretiempo. Extreme programming se enfoca en las pruebas continuas, integración y la necesidad de siempre esperar cambios al software de parte de los usuarios, también pone énfasis en la menor documentación posible. XP es exitoso hoy en día porque se enfoca en la satisfacción del cliente y responde rápidamente a las necesidades de cambio especialmente en ambientes donde los requerimientos son inestables. Esto es posible involucrando a los usuarios en el proceso de desarrollo, empoderar a los desarrolladores para que respondan continuamente a las necesidades del cliente y también poniendo énfasis en el trabajo en equipo, no es escalable, y carece de accesorios y documentación.

Sin embargo XP e ISO tienen algunas debilidades, pero las fortalezas de ambos modelos tienen fortalezas que vale la pena considerar, XP satisface al cliente, reduce la tasa de defectos, responde a necesidades de cambios rápidos, reduce los tiempos de desarrollo; mientras que ISO promete previsibilidad, estabilidad, alto aseguramiento de la calidad. Considerando estas fortalezas y ambos con una creciente demanda para certificación de procesos desde procesos conducidos con planificación y disciplina como ISO y también la rápida aparición de Extreme programming en la industria del software a nivel mundial debido a su flexibilidad y la rápida respuesta a los requerimientos de cambios en los productos de software. Llega a ser necesario combinar Extreme Programming e ISO en una forma complementaria tal que se puedan cumplir con los principios de calidad de ISO sin modificar considerablemente la metodología de XP.

1.2. NORMA ISO 9001:2008

Las series de normas ISO relacionadas con la calidad se constituyen en lo que se denomina una familia de normas que abarcan distintos aspectos relacionados con la calidad. La institución encargada de desarrollar y publicar estos estándares de calidad es la Organización internacional para la

Estandarización y que tiene como objeto desarrollar estándares internacionales que faciliten el comercio internacional.

Cuando las organizaciones tienen una forma objetiva de poder analizar los procesos de un proveedor y evaluar la calidad de los mismos, el riesgo de hacer negocios con dicho proveedor se reduce en gran medida, y si los estándares son los mismos para todo el mundo, el comercio entre empresas de diferentes países puede potenciarse en forma significativa.

Estas normas de calidad requieren de sistemas documentados que permitan controlar los procesos que se utilizan para desarrollar y fabricar un producto o brindar un servicio. Estos sistemas de calidad se fundamentan en la idea de que hay ciertos elementos que todo sistema de calidad debe tener bajo control, con el fin de garantizar que los productos y/o servicios se fabriquen en forma consistente y a tiempo.

Las ISO 9000 es definida como un Estándar Internacional que define los principios, conceptos y términos fundamentales bajo el cual se delinea un Sistema de Calidad. No define cómo debe ser un Sistema de Gestión de Calidad de una organización, sino que ofrecen especificaciones de cómo crearlo e implementarlo; éste será diferente en función de las características particulares de las diferentes organizaciones y sus procesos.

Las normas se revisan cada 5 años para garantizar la adecuación a las tendencias y dinámica del contexto mundial. En el año 2000 cobraron vigencia los cambios propuestos para las ISO 9000, los que se tradujeron en las actuales Normas ISO 9000 versión 2000.

Las ISO 9000:2000 quedaron conformadas por tres grandes apartados:

- ISO 9000:2000 Sistemas de Gestión de Calidad: Principios y vocabulario.

- ISO 9001:2000 Requisitos de los Sistemas de Gestión de Calidad

- ISO 9004:2000 Recomendaciones para llevar a cabo las mejoras de calidad

1.2.1. PRINCIPIOS DE CALIDAD DE ISO 9000

Los principios de calidad más importantes de este estándar (Leakey & Restell, 2001) son los siguientes:

Orientación hacia el cliente.- La razón por la cual las organizaciones existen son los clientes, sus necesidades deberían ser claramente definidas y comprendidas. En los negocios de hoy en día no es suficiente conocer las necesidades del cliente, sino que necesidades deberían se satisfechas

Liderazgo.- Los líderes organizacionales deberían establecer un sentido de dirección y también crear un ambiente apropiado para que los empleados puedan explotar por completo sus potenciales en función de la consecución de las metas y objetivos organizacionales.

Involucramiento de las personas.- Los empleados en todas las categorías de la organización deberían ser llevados e involucrados por completo en las actividades de la organización, de esta forma la organización se beneficiara efectivamente de las contribuciones de los empleados.

Mejoramiento continuo.- Todo principio organizacional debería ser el mejoramiento continuo de sus sistema.

Acercamiento a los hechos para la toma de decisiones.- La toma de decisiones debería ser un esfuerzo colectivo; este debería estar basado en los resultados obtenidos luego de un detallado y cuidadoso análisis de datos.

Beneficio mutuo en las relaciones con los proveedores.- Toda organización debería establecer una relación de beneficio mutuo con sus proveedores.

1.2.2. REQUERIMIENTOS DE LOS SISTEMAS DE GESTIÓN DE CALIDAD.

ISO 9000 contiene requerimientos necesarios para claramente implementar un sistema de gestión de calidad ISO. Las organizaciones requieren contar con un sistema de gestión de calidad para observar, mantener trazabilidad y mejorar continuamente los procesos de desarrollo. ISO 9011 es el estándar de mayor aplicación al ciclo de vida del desarrollo de software. Ayuda a las compañías de software a soportar ciertos requerimientos a través de las diferentes fases del desarrollo de software como diseño, desarrollo, producción, instalación y mantenimiento.

Implementar el estándar ISO 9001:2008, se recomienda el siguiente proceso:

1. Identificar todos los procesos afectados.

2. Definir como la secuencia de procesos interactuará

3. Definir técnicas que aseguren que el proceso es efectivamente controlado y gestionado.

4. Ubicar recursos y otras cosas necesarias para coordinar y sostener el proceso mientras esté en operación.

5. Asegurar que el proceso es monitoreado, medido y analizado para cumplir las metas de gestión de calidad.

6. Asegurar que haya un continuo mejoramiento del proceso.

En la implementación del estándar ISO 9001:2008 son necesarios diferentes tipos de documentos, estos incluyen: documentos que contengan información de la organización, documentos de planificación del proyecto, documentos de especificación de requerimientos, documentos que provean información de cómo se realizaran las actividades del proyecto, y también documentos que muestren que se ha completado del proyecto y los logros registrados.

1.2.3. AREAS DE PROCESO ISO.

A continuación se describen las áreas de proceso donde toma acción el estándar ISO (ISO 9001:2008):

1.2.3.1 Gestión del Sistema de Calidad (4.0)

Requerimientos Generales (4.1).- El negocio por entero debe ser considerado como un proceso más allá de algo individual, un proceso significa un conjunto de actividades que utilizan recursos para transformar entradas en salidas. En la práctica esto significa sinergia que debería ser animada entre todas las unidades de negocio de la organización, ellas deberían trabajar colectivamente en la consecución de las metas de la organización. Los negocios de la organización deberían ser un proceso que claramente defina una verdadera imagen de las actividades de la organización, asi también que describa como este proceso es aplicado en la organización y al interrelación entre los subprocesos. Finalmente el sistema de calidad debería mantenerse en un cambio constante reflejando las mejoras continuas.

Requerimientos de Documentación (4.2).- Esta parte describe que es necesario que sea documentado en un sistema de gestión de calidad.

Se espera que el manual de calidad cubra una descripción general del sistema de gestión de calidad, es aconsejable para esto que el manual este estructurado en línea con los requerimientos ISO 9001, pero no es una regla estricta que debe cumplirse. La cláusula de control de documentos o registro aconseja como el documento o registro debe ser mantenido y circulado hacia aquellos que lo necesitan.

1.2.3.2 Responsabilidad de la Administración (5.0)

Compromiso de la Administración (5.1).- Se requiere que la administración o dirección demuestre su compromiso en la implementación y mejoramiento del sistema de calidad de las organizaciones.

Enfoque al Cliente (5.2).- Se requiere que la administración prueba que las necesidades y expectativas de los clientes siempre serán consideradas en el cumplimiento de la satisfacción al cliente.

Política de Calidad (5.3).- La administración debería proteger los requerimientos de calidad.

Planificación (5.4).- La administración debería colocar metas de calidad que sean medibles y estas metas deben estar en concordancia con las políticas de calidad de la organización. También es necesario el compromiso con la administración para la mejora continua del proceso; así mismo se necesita colocar en esta etapa los recursos necesarios para alcanzar estas metas.

Responsabilidad, autoridad y comunicación (5.5).- Se requiere de la administración tener un manual que describa los roles organizacionales de los empleados, responsabilidades y canales de comunicación dentro de las organizaciones y estás deberían ser constantemente actualizadas.

Revisión de la Administración (5.6).- Se requiere de la administración una revisión continua del sistema en lugar de intervalos regulares para asegurar que las metas sean alcanzadas.

1.2.3.3 Gestión de Recursos (6.0)

Provisión de recursos (6.1).- La organización debería colocar todos los recursos necesarios.

Recursos Humanos (6.2).- La organización requiere dirigir un grupo humano con la necesaria competencia y fortaleza.

Infraestructura (6.3).- La organización debe ubicar la infraestructura necesaria para que se consigan los requerimientos del producto.

Ambiente de Trabajo (6.4).- La organización debe crear un ambiente de trabajo que permita conseguir las metas de calidad.

1.2.3.4 Realización del Producto (7.0)

Planificación de la realización del producto (7.1).- El proceso de desarrollo del producto debe seguir los lineamientos del proceso definido en el sistema de gestión de calidad y debería estar documentado acorde a la rutina de trabajo de la organización.

Procesos relacionados con el cliente (7.2).- Los requerimientos del cliente deben ser identificados, revisados y discutidos con el cliente.

Diseño y Desarrollo (7.3).- La organización debería ubicar un plan de diseño y desarrollo que pueda ser controlado. La organización debe definir técnicas o métodos para revisar, verificar y validar los diseños y el desarrollo. Los artículos diseñados y desarrollados deben ser cuidadosamente examinados y modificados si fuese necesario.

Compras (7.4).- Los proveedores deben ser seleccionados basados en el criterio de capacidad de cumplimiento del proveedor para con los requerimientos especificados por la organización.

Producción y Prestación del Servicio (7.5).- El desarrollo de productos debería ser controlado probablemente a través del uso de herramientas, tecnicas y método especializados, y el requerimiento de trazabilidad debería ser administrado y documentado. El proceso de desarrollo es evaluado basado en su capacidad para cumplir requerimientos definidos o específicos.

Dispositivos de monitoreo y medición (7.6).- La organización debería definir lo que es medible, las métricas que serán monitoreadas y los tipos de herramientas de monitoreo y medición necesarias que deberán ser utilizadas.

1.2.3.5 Medición, Análisis y Mejoramiento (8.0)

General (8.1).- La organización debería proveer procedimientos de supervisión y medición que puedan cumplir con los requerimientos del estándar de calidad.

Monitoreo y medición (8.2).- Estadísticas acerca de la satisfacción e insatisfacción del cliente deberían ser analizadas y los resultados entregados a la Administración ya que estos indican como está trabajando el sistema de calidad. El sistema que genera esta clase de información estadística debería ser modelado; el sistema de calidad debería ser auditado y documentado. El proceso de desarrollo y los productos deberían ser continuamente medidos.

Figura 1.0 (Dean Don, 2003)

Control de productos no conformes (8.3).- Productos que no cumplan con los requerimientos del cliente deberían ser identificados a través de pruebas; tales

productos deberían ser re-desarrollados para cumplir con los requerimientos especificados antes de la entrega al cliente.

Análisis de Datos (8.4).- Datos deberían ser confrontados con la satisfacción del cliente, productos y procesos. Los datos deberían ser cuidadosamente analizados e interpretados para determinar la eficiencia del sistema de calidad y para propósito de mejoramiento continuo del proceso.

Mejoramiento (8.5).- La organización debería planificar y diseñar un proceso que pueda ser continuamente mejorado.

1.2.4. FORTALEZAS Y DEBILIDADES DE ISO 9000.

ISO 9000 ayuda a las organizaciones a definir y administrar su sistema de gestión de la calidad para mejorar la calidad de sus productos. De acuerdo a algunos autores calidad es el "grado de cumplimiento que tienen las características del producto para con los requerimientos del cliente". ISO 9001 también ayuda a direccionar los problemas de algunas organizaciones, por ejemplo políticas de comunicación de la oficina, planeación del producto, documentación del proceso, etc. ISO 9000 ayuda a satisfacer al cliente, define la satisfacción del cliente como "una percepción del cliente del grado en el cual sus requerimientos ha sido cumplidos". ISO 9001 también anima a las organizaciones a invertir en sus empleados dándoles adecuados programas de entrenamiento que beneficiaran tanto al empleado como a la organización por un largo tiempo. ISO 9000 ayuda a las organizaciones para mejorar continuamente en los procesos de sus negocios de esta forma los procesos se hacen más eficientes. Cuando un proceso es más eficiente reduce fallas en los productos de software, y así se reducen costos de desarrollo, implementación y mantenimiento. Procesos eficientes producen productos de software estables, previsibles y confiables.

El principal problema con ISO 9000 es que involucra mucha burocracia, y algunas organizaciones solo necesitan la certificación ISO 9000 para propósitos de marketing en contra del mejoramiento de procesos de software

que es el principal objetivo. También ISO 9000 requiere un alto volumen de documentación y la certificación podría ser alcanzada en un ambiente bien documentado, pero no necesariamente significa que el sistema de calidad es eficiente (Jerzy, N, J, W&A, 2005). Requiere muchos recursos (tiempo, costo y esfuerzos) para implementar ISO 9000. Algunas organizaciones de software son escépticas acerca de ISO debido al hecho que es demasiado general y hay una tendencia a creer que el proceso de creación de software es diferente a otras compañías manufactureras.

ESTANDAR ISO 9000	
Fortalezas	**Debilidades**
Escalable	Demasiada burocracia
Promete previsibilidad	Demasiada documentación
Realza confiabilidad y estabilidad del sistema	Inflexible – respuesta baja para necesidades de requerimientos rápidos de cambios
Acercamiento estructurado al desarrollo de software	No se ajusta a equipos pequeños de desarrollo
Planificación efectiva y gestión del riesgo	Muy costoso implementarlo
Requerimientos definidos como documentos de especificación de requerimientos	Demasiado general, no es específico para industria que solo hacen software
Se ajusta a equipos grandes	
Mejoramiento al proceso de software	
Enfocado en el cliente	

Tabla 1.0 Sumario de Fortalezas y Debilidades del estándar ISO 9000

1.3. METODOLOGIAS DE DESARROLLO ÁGIL DE SOFTWARE.

1.3.1. INTRODUCCION.

La atención global ha sido cambiada con tendencia hacia el desarrollo ágil en el desarrollo moderno de software. Esta nueva tendencia en la ingeniería de software fue desarrollada por la alianza ágil, cuya motivación estuvo basada en la complejidad y en el exceso de documentación involucrada en los procesos de desarrollo de software. Debido a la creciente insatisfacción de los métodos utilizados en el desarrollo de software poco a poco se fue introduciendo diferentes metodologías ágiles entre ellas: Extreme Programming, Scrum. Crystal, etc.

La tendencia al desarrollo ágil, por ejemplo, Extreme programming fue propuesta para corregir los desperfectos ocasionados por el desarrollo de software tradicional manejados por un plan enfocándose en rápidos entregables, pronta respuesta a requerimientos de cambios, rápidas iteraciones y desarrollo incremental. El desarrollo ágil contrasta con la metodología tradicional al recomendar desarrolladores experimentados, también recomienda que el cliente debería estar comprometido, ser conocedor, colaborativo, representativo y con el suficiente poder de decisión. Las metodologías ágiles también contradicen a las tradicionales autorizando a los clientes o usuarios a estar en sitio con los desarrolladores con el completo compromiso y dedicación. La metodología ágil se caracteriza por modificaciones frecuentes del sistema cuando la necesidad aparece después de las pruebas, esto discierne con el diseño tradicional de la arquitectura del sistema donde los cambios son considerados para un futuro en lugar de estar pendiente de las necesidades actuales. Como se explica en (Highsmith & Cockburn, 2001), hoy en día las compañías necesitan sobrevivir usando productos que se adapten rápidamente al cambio que demanda el mercado, en una forma tal que puedan ganar un lugar en el mercado a sus competidores, ellos necesitan agilidad para ser capaces de adaptarse y sobrevivir en este ambiente de continuos y nuevos cambios.

La metodología ágil se aplica mejor en pequeños, medianos y proyectos menos complejos. En proyectos de mayor escala con más de 10 miembros en el equipo de trabajo; las metodologías tradicionales de desarrollo son mejor aplicables. Las metodologías ágiles no se recomiendan en sistemas de vida crítica tales como sistemas propietarios como por ejemplo sistemas de control de una aeronave o un sistema de monitoreo cardiaco.

1.3.2. EL MANIFIESTO AGIL EN DETALLE.

En febrero del 2001, diecisiete líderes de esta nueva ola en el desarrollo de software tuvieron una reunión en Salt Lake City, Utah, USA. Ellos estuvieron desarrollando nuevas metodologías desde mediados de los años noventa como Extreme Programming, Scrum, Dynamic Software Development Method, Crystal Methodologies, Lean Development, Adaptive Software Development, y feature Driven Development.

Ellos crearon la alianza ágil para apoyar esta nueva ola de desarrollo de software, más metodologías han sido creadas ahora y esto continúa en crecimiento. El núcleo del desarrollo ágil de software son los valores presentados en el manifiesto ágil y otros doce principios que serán discutidos en esta sección.

1. *Individuos e Iteraciones sobre procesos y herramientas.* El primer punto en cuanto a esta declaración es que enfocarse en individuos en el equipo es mejor que enfocarse en los individuos como recursos del proceso. El segundo punto es que las interacciones y aspectos de comunidad entre la gente experta influencia nuevas soluciones en los desafíos que aparecen en un proyecto. Además, esto significa que la utilización de un proceso no documentado pero con una comunicación excelente es preferible a la utilización de procesos muy documentados con comunicaciones malas entre personas en el trabajo (Cockburn, 2001).

2. *Trabajar en el software que trabajar sobre la documentación.* El trabajo en el software es el producto principal del desarrollo de software donde el cliente puede evaluar si los sistemas han sido construidos o no conforme a sus requerimientos (Cockburn, 2001). El software será probado en intervalos regulares entonces el código se hace más simple y más fácil para cambiar en un futuro, y la documentación debería estar en un nivel más abajo apenas otorgándole la importancia justa (Abrahamasson et al., 2001). Una cantidad enorme de documentación no necesariamente significa que el problema ha sido bien entendido.

3. *Colaboración del cliente sobre la negociación del contrato.* Esta idea fortaleza la necesidad para comunicaciones frecuentes y la colaboración entre el cliente o usuario y los desarrolladores más allá de únicamente tratar negociaciones del contrato entre los interesados que podrían dificultar la implementación de requerimientos específicos. Sin embargo se debe considerar que para los proyectos más grandes siempre será necesario contratos mejor definidos (Abrahamasson et al., 2001), mas allá de que la colaboración siempre será necesaria para una mejor comprensión de las necesidades del cliente.

4. *Respondiendo al cambio sobre el seguimiento del plan.* Esta idea desanima la planificación a largo plazo considerando la planificación a corto plazo para ser más efectivos al momento de responder a necesidades de cambio rápido. Generalmente se recomienda un plan detallado para dos semanas y un plan general para un periodo de tres meses.

1.3.3. LOS PRINCIPIOS EN EL MANIFIESTO AGIL.

En esta sección se explicará los principios del manifiesto ágil.

- **Nuestra más alta prioridad es satisfacer al cliente a través de prontos y continuos entregables de software de valor.**

El objetivo de las metodologías ágiles es entregar software que satisfaga las necesidades y expectativas del cliente procurando siempre servir a los objetivos del negocio (Fowler & Highsmith, 2001). Las entregas regulares son necesarias para obtener un continuo espíritu ganador en el equipo y rápidamente obtener una retroalimentación de parte del cliente y de ser necesario cambiar las prioridades a lo largo del desarrollo. El intervalo de las entregas regulares debería ser negociado con el cliente para ajustar el proyecto a un ritmo constante (Cockburn, 2001).

- **Los requerimientos de cambio son bienvenidos, inclusive si aparecen tardíamente. Los procesos ágiles toman el cambio como una ventaja competitiva para el cliente.**

  Las metodologías ágiles tienen algunas técnicas para tratar con cambios en cada etapa del desarrollo de software por ejemplo el uso de continuas y prontas entregas con desarrollo iterativo. Si la compañía puede encarar el cambio rápidamente entonces esto se vuelve una ventaja competitiva en el mercado tanto para la compañía cliente como para la compañía proveedora (Cockburn, 2001). Es mejor enfrentar el cambio que tratar de prevenirlo (Fowler & Highsmith, 2001).

- **Entregas frecuentes de software que trabaje, desde un par de semanas hasta un par de meses, con preferencia por la escala de tiempo mas corta.**

  En este principio están definidos los intervalos de trabajo para entregar software que trabaje.

- **Gente de negocios y desarrolladores trabajando juntos diariamente a lo largo del proyecto.**

  Si la gente que maneja el negocio no pasa el tiempo suficiente con los desarrolladores el proyecto puede fallar (Cockburn, 2001). Si no hay una comunicación base diaria entre ellos el proyecto puede fracasar. Los técnicos de las metodologías ágiles empiezan a trabajar con requerimientos que no están bien definidos, es por eso que se vuelve necesaria la cooperación de la gente de negocios, justamente para llenar este espacio.

- **Construir proyectos alrededor de gente motivada. Si se les da el ambiente y apoyo que ellos necesitan, de seguro conseguirán hacer su trabajo.**

  Los individuos son claves en el proyecto, sin ellos no hay ningún proyecto. La motivación puede estar unida con una buena comunión, amistad, y satisfacción en el trabajo Por lo tanto maximizar el factor "gente" tiene una alta importancia en el desarrollo ágil. La confianza entre gerentes y el equipo es un elemento necesario para el éxito en un proyecto específico (Fowler & Highsmith, 2001).

- **El método más eficiente y efectivo de llevar información hacia el interior del equipo de trabajo es a través de la conversación cara a cara.**

  La principal cosa al momento de transferir información no es el alto volumen de documentación empleada, sino que la gente haya comprendido lo que se quiere comunicar. Sin embargo, el problema no es la documentación enorme contra la ausencia de documentación, sino como balancear la cantidad de documentación y la transferencia del conocimiento (Fowler & Highsmith, 2001).

- **El software trabajando es la primera medida de progreso.**

  Confíe en el software que está trabajando en lugar de la documentación y los planes. Las metodologías ágiles acentúan el hecho del desarrollo rápido y evolutivo. Los proyectos pueden estar divididos en pequeñas paquetes a fin de desarrollar y probar incrementalmente (Cockburn, 2001). La utilización de estas técnicas hace que el cliente pueda ver un progreso temprano y útil en el proyecto y obtener una retroalimentación de cuales con los riesgos que el proyecto afronta (Fowler & Highsmith, 2001).

- **El proceso ágil promueve desarrollo sostenible. Los sponsors, desarrolladores y usuarios deberían ser capaz de mantener un paso constante indefinidamente.**

  Este principio implica que el equipo debería trabajar con regularidad, no con horas extras continuas, porque esto puede afectar el progreso del trabajo debido a que la gente está más cansada no sólo durante horas extras sino también con el trabajo en horario regular. Este puede causar esto mayores defectos se encuentren en el desarrollo durante un día de trabajo regular (Cockburn, 2001).

  Además, el desarrollo sostenible (Fowler & Highsmith, 2001) significa trabajar con el mismo intervalo de horas diarias durante todo el proyecto entero y permanecer de esta forma saludable y creativo.

- **La atención continua a la excelencia técnica y un buen diseño realzan la agilidad.**

  Si el diseño es bien realizado con buenos patrones que resultan en un diseño bien definido, entonces es muy fácil hacer cambios en el desarrollo tanto en el presente como en el futuro. Además, el diseño es probado y mejorado con regularidad para una mejor comprensión en posteriores entregables (Cockburn, 2001). El diseño es realizado a través de todo el proceso de desarrollo y en cada parte incremental, por

lo tanto la calidad del diseño es esencial para mantener la agilidad (Fowler & Highsmith, 2001).

- **Simplicidad, el arte de maximizar el trabajo no hecho es esencial.**

  Hacer el diseño simple en un proyecto de software donde los cambios puedan ser realizados, pero es mejor para el mantenimiento y desarrollo tener un diseño donde los cambios no puedan ser hechos efectivamente o tomen mucho más tiempo poderlos hacer, ya que es más fácil añadir algo al código que está trabajando (Cockburn, 2001), (Fowler & Highsmith, 2001).

- **Las mejores arquitecturas, requerimientos y diseños emergen de equipos que se auto-organizan.**

  Todas estas propiedades emergen (Cockburn, 2001) debido a las pocas reglas y a las muchas iteraciones en el equipo de trabajo, y así mismo se va incrementando conforme avanza el proyecto.

- **En intervalos regulares, el equipo reflexiona en como llegar a ser más efectivo, luego sintoniza y ajusta su comportamiento como consecuencia de esto.**

  Las metodologías ágiles no son cuestiones que el equipo puede adoptar y copiar fácilmente. Para empezar se puede adoptar una parte de la metodología, pero las metodologías pueden ser adaptadas para cada proyecto específico, ellas no son universales para cada proyecto (Fowler & Highsmith, 2001).

1.4. EXTREME PROGRAMMING.

Extreme Programming es el modelo más popular de la familia de metodologías ágiles. Fue presentado por Kent Beck en los años noventas y luego más tarde Beck y Ward Cunningham experimentaron con una forma más simple y eficiente para desarrollar software. En marzo de 1996 Beck empezó un

proyecto con su equipo en Chrysler que dio nacimiento a Extreme Programming, al que Ron Jeffries (Jeffries, A, H, 2008) define de la siguiente manera: "Es una disciplina de desarrollo de software basado en valores de simplicidad, comunicación, retroalimentación y coraje. Trabaja trayendo a todo el equipo junto hacia practicas simples, con la suficiente retroalimentación para que el equipo sea capaz de ver donde ellos están y sintonizar esas prácticas a su situación en particular". "Todo el equipo" aglutina a todos los contribuyentes del proyecto, y el equipo incluye al cliente que trabaja con este diariamente a lo largo del proyecto. El cliente conduce el proyecto, el entrega los requerimientos y los prioriza con la asistencia del analista o probador. Este equipos también lo hacen los programadores, probadores, administradores, etc.

La filosofía de Extreme Programming dice "permitan que el desarrollo de software sea divertido, simple, flexible, previsible, menos riesgoso, eficiente y mas científico".

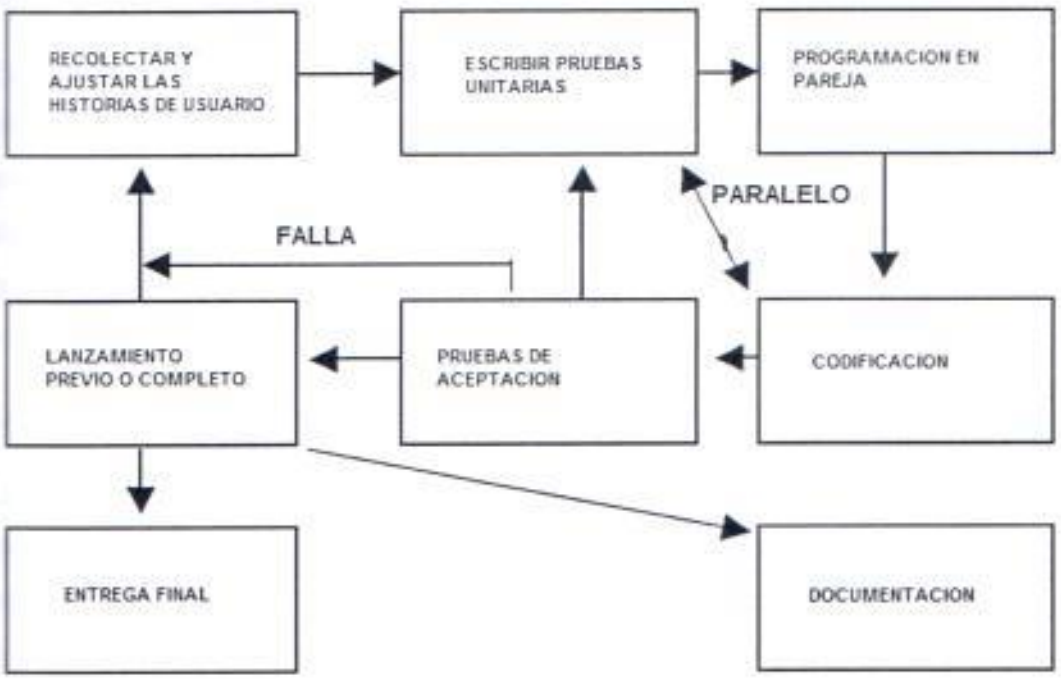

Figura 2.0 (METODO DE EXTREME PROGRAMMING)

XP esta compuesta por un conjunto de valores, prácticas, principios y actividades que serán descritas en las siguientes secciones de este capítulo.

1.4.1. VALORES DE EXTREME PROGRAMMING.

Los valores que están detrás de los cinco principios de XP son: simplicidad, comunicación, retroalimentación y Coraje (Beck, 2004).

Simplicidad

Hacer del desarrollo de software algo simple. Los programadores deben darle importancia a lo que ellos están escribiendo para el momento, no pensando en el futuro. Si más adelante son necesarios nuevas funcionalidades, entonces están se harán sin mucho esfuerzo ya que el código es simple y fácil de comprender. Hacer simple las cosas complejas, es igual que en las artes marciales donde el aprendiz practica movimientos simples todos los días, para hacer movimientos complejos en el futuro.

Estas afirmaciones contradicen las viejas prácticas de programación donde se piensa que los programadores deben hacer el código general para que puedan soportar las necesidades del futuro, pero esta premisa transforma el código en algo más complejo de entender, usando la premisa de simplicidad de XP el proyecto puede ir más rápido en un mundo extremadamente rápido como es el actual.

Comunicación

La comunicación es un parte importante en el desarrollo del software, por ejemplo si hay ausencia de comunicación entre clientes y desarrolladores, entre el administrador y cliente, entre administrados y desarrolladores o entre desarrolladores es muy probable que el proyecto fracase (Beck, 2004).

Los clientes en XP deben está comprometidos con el desarrollo, ellos deberán ser uno de los miembros del equipo. No más la cultura de "ellos" y "nosotros". El equipo de desarrollo de software debe actual como un equipo de futbol

donde todos trabajan para ganar el partido, así mismo el equipo de XP debe trabajar para tener éxito con el desarrollo. El desarrollo de software no es un acto donde el cliente es el adversario y no es parte del equipo.

Para llegar a un acuerdo entre lo que el cliente quiere y lo que los desarrolladores pueden hacer, el proyecto XP basa sus relaciones usando cuatro variables: alcance, calidad, recursos y tiempo.

Alcance es lo que el sistema hace y lo que debería hacer en el futuro. *Calidad* son las medidas usadas para cumplir con las necesidades y expectativas del cliente, como por ejemplo, confiabilidad, exactitud, seguridad, etc. *Recursos* son las personas, equipamiento técnico y el sitio donde el proyecto XP tendrá lugar. *Tiempo* es cuanto durará el proyecto.

La comunicación puede estar relacionada con la simplicidad porque si el desarrollador hace aplicaciones más simples, estas son más simples de comunicar y comprender por parte del cliente.

Retroalimentación.

La retroalimentación desde un punto de vista es como los usuarios y desarrolladores ven como está avanzando el proyecto o aplicación. De esta forma el equipo de trabajo toma más confianza (Beck, 2004) en cuanto a lo que Ellos están haciendo si ellos tienen una retroalimentación exacta de la aplicación.

Los desarrolladores pueden hacer pruebas unitarias para conocer donde están las fallas del sistema así más tarde poderlas corregir y mejorar la calidad del producto. Ellos obtienen con las pruebas unitarias una retroalimentación de cómo la aplicación está avanzando.

Los clientes hacen pruebas funcionales o de aceptación donde ellos pueden ver si el comportamiento de la aplicación y si esta cumple con los requerimientos, así ellos obtienen una retroalimentación del sistema. Además, los clientes crean historias de usuario, los desarrolladores al mismo tiempo hacen estimaciones para cada historia de usuario y de esta forma el cliente

obtiene retroalimentación de cuanto tomara desarrollar una historia específica (Beck, 2004).

Para el lanzamiento del aplicativo es necesario que el 100% de las historias de usuario hayan sido probadas y hayan pasado las pruebas.

Desde otro punto de vista la retroalimentación no está solamente relacionada con las pruebas, está relacionada con la comunicación continua entre clientes y desarrolladores trabajando juntos en el mismo espacio debido a que ellos están siempre comunicándose entre unos y otros obteniendo retroalimentación de cada parte.

Más tarde los desarrolladores pueden cambiar fácilmente en el sistema lo que el cliente quiere. Esto es debido a que el código es simple ya que se obtuvo retroalimentación del cliente. Así es como estos tres valores de XP: comunicación, simplicidad y retroalimentación están relacionados.

Coraje.

Estos tres valores trabajando juntos se convierten en coraje. Kent Beck muestra una historia donde ellos detectaron problemas en la arquitectura del sistema una pocas semanas antes de la entrega. Ellos solucionaron el problema pero eso se convirtió en más problemas porque Ellos tuvieron que reconstruir el 50% de las pruebas, pero lo hicieron. Más tarde ellos empezaron a rediseñar todo el código que tenian debido a cambios en las pruebas.

Ese coraje es formado por el buen entendimiento del sistema debido a la simplicidad, retroalimentación con el cliente y buenos canales de comunicación (Beck, 2004).

1.4.2. PRINCIPIOS DE EXTREME PROGRAMMING.

Los principales principios de XP son cinco: rápida retroalimentación, asumir simplicidad, cambio incremental, abrazar el cambio, y trabajo de calidad.

Rápida retroalimentación.

La rápida retroalimentación puede ser dividida en dos partes: la retroalimentación desde el punto de vista del cliente y el desarrollador.

Desde el punto de vista de los desarrolladores con la retroalimentación ellos aprenden a producir código, diseño y pruebas de buena calidad, esto ocurre rápidamente en un marco de tiempo de segundos o minutos (Beck, 2004).

Además, los clientes aprenden como hacer una buena contribución para desarrollar la aplicación en un rápido marco de tiempo en días.

El núcleo de este principio es "conseguir retroalimentación, interpretarla, y poner todo lo aprendido dentro del sistema tan rápido como sea posible.

Asumir simplicidad.

Los desarrolladores deben hacer el diseño y el código muy simple usando prácticas XP y haciendo refactoring al código, haciendo pruebas y complementándolo con una buena comunicación (Beck, 2004). Los desarrolladores deberían hacer el código simple y tener la certeza de que una mayor complejidad puede aparecer más tarde.

Cambio incremental.

El cambio se hace a través de pequeños paso a lo largo del desarrollo y la planeación (Beck, 2004). La misma práctica debería hacerse para llevar cambios dentro del equipo y la implementación de XP dentro de la organización.

Abrazar el cambio.

El equipo debe tratar de cambiar únicamente las áreas problemáticas en la aplicación y mantener las otras partes (Beck, 2004).

Trabajo de calidad.

La calidad refleja que el equipo trabaja haciendo las cosas bien hechas usando las otras tres variables: alcance, costo y tiempo (Beck, 2004).

1.4.3. PRACTICAS DE EXTREME PROGRAMMING.

Las practicas son las cosas que un equipo XP hace todos los días durante el desarrollo, es decir es su rutina de actividades diarias.

Estas prácticas no pueden ser fructíferas si los valores no viene junto, por ejemplo la programación en pareja no tendría sentido si no es practicada con el objetivo de lograr comunicación entre los miembros del equipo, para obtener retroalimentación y simplificar el sistema, si estas prácticas no se ajustan a los valores de XP entonces dejan de ser útiles.

De acuerdo a cada situación se determina la clase de práctica que deberá ser adoptada, si la situación cambia la práctica también se ajustará a la actual. Los cambios en las prácticas no necesariamente cambian los valores aplicados.

Las prácticas son un vector de donde uno está y donde uno puede estar con XP. Prácticas son las vías para lograr desarrollar software efectivo, es un proceso gradual que empieza en un punto en particular. También la práctica es una forma de hacer mejoramiento continuo, las practicas de XP trabajan fuertemente juntas teniendo cuidado de aplicar una práctica diferente a la vez que eventualmente conducirá hacia el mejoramiento.

Las siguientes son las doce prácticas más comunes en XP:

El juego de la planeación.- Esta práctica determina el alcance de la iteración actual. La mayor tarea es determinar que es lo más necesario para el cliente y luego implementar esto en primer lugar. Las principales actividades incluyen: determinar el alcance del proyecto, priorizar funcionalidades, implementar las funcionalidades y fijar las fechas de entrega.

Pequeños entregables.- Ela idea detrás de esto es conseguir que el sistema entre a producción a tiempo para obtener una constante retroalimentación de del cliente para evitar riesgo y minimizar esfuerzo para efectuar los cambios, lo cual se convierte en bajos costos de producción a largo plazo. Los pequeños entregables trabajan estrechamente con el juego de planificación y prácticas de diseño simple.

Diseño simple.- Se debe mantener el código simple para resolver necesidades presentes que pasaran luego las pruebas de aceptación del cliente, hacer esto permite que el sistema entre en operación a tiempo. La mayoría de los clientes no conocen exactamente los que ellos tienen que hacer al momento de participar en el diseño del sistema, pese a su experiencia, pero siempre requerirán cambios que vayan a favor de cumplir con sus requerimientos.

Pruebas.- Esta es una práctica muy importante que sirve para comprobar si las funcionalidades del sistema están trabajando como se espera, y es también a través de las pruebas que se demostrará si el producto aún trabaja después del refactoring o de la integración. Se recomienda que esta práctica debería ser automatizada para generar confianza a una re-arquitectura, re-diseño, refactoring, o inclusive hasta suprimir código sin romper el funcionamiento del sistema.

Integración continua.- Este es un proceso de integración incremental de código dentro del sistema, el cual luego es probado como un sistema completo al menos una vez al día. Esperar hasta el final del proyecto antes de integrar todos los módulos dentro del sistema final puede ser tedioso, consumir más tiempo, muy costoso y podría resultar en una pobre calidad del sistema.

Refactoring.- Si el código no funciona, entonces hay que cambiarlo. Si el código es difícil de comprender, podría no ser claro, no tiene sentido alguno, o es muy difícil de modificar, significa que el código no funciona. Es mejor re-escribir el código lo cual significa hacer refactoring. Siempre es mejor hacer refactoring al código constantemente para su fácil comprensión y modificación,

por esto se debe crear un espacio dentro del proyecto para ejecutar constantes mejoras al código.

1.4.4. DEBILIDADES Y FORTALEZAS DE EXTREME PROGRAMMING.

Extreme programming es un proceso adaptivo y también muy flexible el cual permite al desarrollo de software mantener el paso con rápidos cambios en las necesidades del negocio dentro de la competencia global. Esta práctica permite a las organizaciones acomodarse a los frecuentes cambios en los requerimientos del producto. Extreme programming reduce tareas de administración y costos, realza la productividad del equipo de trabajo y satisface las necesidades del cliente, en comparación con los procesos de desarrollo tradicionales manejados por un plan.

Por otro lado extreme programming tiene algunas debilidades, es un procesos para equipos de desarrollo pequeños de 10 a 12 miembros, tampoco es escalable, lo cual hace difícil adoptar XP en proyecto de gran escala. También XP carece de herramientas y documentación, lo cual hace difícil de mantener a un sistema desarrollado usando la metodología XP, a diferencia de los procesos manejados por un plan que llevan a la práctica la realización de un diseño más avanzado a través de herramientas y extensa documentación en todo el ciclo de vida del desarrollo. Debido a la carencia de una planificación avanzada, comparar XP con procesos tradicionales manejados por un plan no tiene lugar dentro del desarrollo de sistemas críticos tamales como aplicaciones de seguridad. Esto requiere de desarrolladores experimentado para una implementación exitosa.

EXTREME PROGRAMMING	
FORTALEZA	**DEBILIDAD**
Entrega rápida de un prototipo	No escalable
Acercamiento iterativo al desarrollo	Demasiado énfasis en entregar resultados tempranos
Respuesta rápida a necesidades de requerimiento de cambio	La programación en pareja es costosa de practicar
Crea un espacio para diseños experimentales	La programación guiada por pruebas extiende el tiempo de desarrollo
Realza la confiabilidad del sistema	Requerimientos no definidos
El Refactoring realza la calidad del software	Lineamiento no estructurado para el desarrollo
Alta tasa de producción de código	Carencia de previsibilidad
Código de calidad	Carencia de planificación
Propiedad colectiva del código	Requiere de desarrolladores experimentados
Acceso a usuarios dedicados	Carencia de documentación
Bajos costos operativos	Carencia de presupuesto operativo planificado
Se ajusta a equipos de tamaños pequeños y medianos	No se ajusta a equipos de gran tamaño
Un ambiente cohesivo de desarrollo	
Permite flexibilidad	

TABLA 4.0 SUMARIO DE FORTALEZAS Y DEBILIDADES DE EXTREME PROGRAMMING
(FrVreed, 2006)

Característica	Extreme Programming	Estándar ISO
Aplicación de Software		
Metas primarias	Respuesta rápida a los requerimientos de cambio	Previsibilidad, estabilidad, confiabilidad y alto aseguramiento a la calidad
Tamaño	Equipos de proyecto medianos y pequeños	Equipos de proyecto grandes
Ambiente	Proyecto y requerimientos inestables	Proyectos y requerimientos estables
Administrativo		
Usuarios y clientes	Usuarios y clientes dedicados o en sitio enfocados en priorizar los avances del proyecto	Interacción con el cliente cuando la demanda esta enfocada en negociaciones del contrato
Planificación y control	Planes del proyecto interiorizados por los desarrolladores – controles cualitativos	Planes del proyecto documentados – controles cuantitativos
Comunicaciones	Transferencia tacita de conocimiento y la distribución del mismo, no hay documentación	Explicita transferencia del conocimiento y compartirlo a través de la documentación
Técnico		
Requerimientos	Informal definición de requerimientos - se presentan historias que son priorizadas por el cliente pero no existe documentación formal	Formal definición de requerimientos – formalmente documentados en el documento de especificaciones de requerimientos
Desarrollo	Diseño simple que trabaja para las necesidades actuales, incrementos cortos, pequeños entregables, refactoring frecuente y de bajo costo	Diseño avanzado, grandes incrementos y entregables, refactoring tardío que puede ser muy costoso
Pruebas	Pruebas frecuentes, plan de pruebas escritos antes de la codificación, casos de pruebas ejecutables	Planes de pruebas documentados que se caracterizan con procedimientos oficiales.

Personal		
Clientes / Usuarios	Compromiso y colaboración con los desarrolladores	No hay trabajo en conjunto con los desarrolladores en la mayoría de casos
Desarrolladores	Se hace necesario contar con miembros experimentados en el equipo desde el inicio hasta la finalización del proyecto	Los miembros experimentados son necesarios al inicio del proyecto, pero podrían participar miembros no experimentados conforme el proyecto avanza con la supervisión de los pocos miembros experimentados
Cultura	Empoderamiento, libertad, gente enfocada. Caracterizada por el caos	Libertad limitada, proceso enfocado. Caracterizado por el orden.

TABLA 4.1 CARACTERISTICAS GENERALES DE EXTREME PROGRAMMING E ISO
(BarryTurner, 2003)

1.5. CASO DE ESTUDIO DE LA TESIS.

Para desarrollar la presente tesis hemos considerado plantear un modelo del proceso de desarrollo aplicando la metodología de Extreme Programming. Este modelo es descrito en el Capítulo 2 y su aplicación dentro de un caso de estudio se detalla en el Capitulo 3. El objetivo principal es analizar cada uno de los enunciados descritos en nuestro proceso de calidad junto a los principios de la metodología seleccionada (XP) dentro del proyecto donde se aplico el modelo planteado.

El objetivo principal de esta tesis es investigar cada una de las características y prácticas de XP e ISO, confrontarlas y plantear un modelo para que las empresas de software puedan beneficiarse de ellas.

La investigación planteada en el caso de estudio responderá a las siguientes preguntas:

1. ¿Son los valores y principios de las metodologías ágiles de desarrollo complementario con los principios de los sistemas de gestión de la calidad y con ISO 9001:2008, o ellos se contradicen el uno al otro?

2. ¿Cuales son las fortalezas y debilidades de cada modelo?

3. ¿Cuáles prácticas, actividades y roles XP fueron desarrollados por el equipo del proyecto?

4. ¿Un proyecto XP es condescendiente con el estándar ISO 9001:2008?

5. ¿Pueden estos modelos complementarse el uno al otro? y ¿En cuales áreas?

6. ¿Cuáles son los beneficios de usar estos modelos juntos?

Respondiendo estas preguntas podremos ver como estos modelos pueden complementarse el uno al otro. Las respuestas también definirán las relaciones entre prácticas XP y las cláusulas de ISO 9001; y el propósito de cómo una organización con certificación ISO puede construir sus procesos para responder fácilmente al cambio usando XP. Esto se logra a través de una comparación y análisis del caso de estudio.

En el presente caso de estudio que consideramos para aplicar el objetivo de esta tesis hemos definido las siguientes instancias para su desarrollo:

- Definición del modelo planteado. Capitulo 2.
- Desarrollo del caso de estudio. Capitulo 3.
- Desarrollar conclusiones y recomendaciones. Capitulo 4.

El tipo de caso de estudio desarrollado fue de tipo exploratorio, hecho para responder al objetivo planteado en la Introducción del Capítulo 1. Es principalmente exploratorio porque aún no se ha definido ningún modelo oficial que estandarice la aplicación de los principios de la norma ISO 9001:2008 dentro de una metodología XP para desarrollo ágil de software y queremos presentar una propuesta más a las varias ya existente hasta el día de hoy, las cuales también plantean un modelo a considerarse para su estandarización.

CAPÍTULO 2

DESARROLLO DEL SISTEMA DE CALIDAD.

2.1 ESTRATEGIA PARA MODIFICAR XP DESDE LA PERSPECTIVA ISO.

Para lograr que las prácticas de Extreme Programming sean compatibles con el punto de vista de ISO hay la necesidad de encausar las prácticas de Extreme Programming de tal forma que cumplan con los requerimientos de calidad plasmados en la norma ISO 9001:2008. Esta combinación ayudará a que tanto ISO y Extreme Programming se complementen logrando superar las debilidades propias de XP y haciendo que la organización ISO 9000 pueda responder rápidamente a las necesidades del cliente, especialmente en un ambiente de requerimientos dinámicos. De esta forma se guiará a una organización que practica Extreme Programming hacia un proceso de certificación exitoso.

En esta sección describiremos cada una de las fases que conforman el proceso de desarrollo aplicando las políticas de calidas ISO 9001 en cada una de ellas. Las prácticas XP que serán modificadas para reflejar las necesidades ISO 9001:2008 se darán en las siguientes áreas claves dentro del proceso para ISO:

- Realización del producto (Planificación y ejecución del proyecto), y
- Mediciones y monitoreo (Monitoreo y control del proyecto).

La razón para considerar estas áreas del proceso como claves es el hecho de que las cláusulas ISO contienen requerimientos para la metodología a ser implementada o considerada en el presente proyecto.

2.1.1 REALIZACION DEL PRODUCTO

Las prácticas de Extreme Programming necesitan ser modificadas para que reflejen las necesidades ISO 9001-2008 en el área de realización del producto (Jerzy, N, J, W&A, 2005) adoptando el modelo de madurez XP (XPMM), el cual es normalmente usado en la introducción de la metodología Extreme

Programming en una organización y que es similar a CMM (Modelo de Capacidades y Madurez) de SEI. El modelo de madurez XP será explicado brevemente más adelante para una comprensión apropiada de porque este modelo fue adoptado en el cumplimiento de los requerimientos de ISO. El modelo de madurez XP es necesario para asegurar en la actualidad que la programación practicada está correctamente implementada sin comprometer calidad cuando se ejecuten proyectos de Extreme Programming. Las prácticas XP fueron agrupadas en cuatro niveles del modelo de madurez (Nawrocki, Walter & Wojciechowski2001): básico, relación con el cliente, aseguramiento de la calidad del producto y desempeño del proyecto tal como lo muestra la figura 2.1:

Figura 2.1

Los Niveles 2, 3 y 4 del modelo XPMM: relación con el cliente, calidad del producto y desempeño del proyecto serán considerados para cumplir con las necesidades de ISO en el capítulo 7 de la norma 9001-2008, y esto incluye: Administración de la Relación con el Cliente (Nivel 2), Aseguramiento de la Calidad del Producto (Nivel3) y Metodología de Desarrollo XP (Nivel4).

2.1.1.1 Administración de la Relación con el Cliente.

Extreme programming remarca con énfasis su compromiso de satisfacer al cliente, y esto cae dentro del nivel 2 de XPMM, y las prácticas XP que trabajan en la satisfacción del cliente incluyen entre las principales:

1. El cliente o usuario cuenta historias para describir sus necesidades y expectativas. Estas historias contadas frecuentemente no son documentadas formalmente como un documento de especificaciones de requerimiento, como se requiere en procesos tradicionales de desarrollo de software. En la sección 2.4.2.1 de este capítulo describiremos como desarrollar las historias de usuario.

2. Obteniendo constante retroalimentación del cliente. El proceso de desarrollo de Extreme Programming está dividido en cortos entregables por un periodo que normalmente va entre 6 a 9 semanas. Cada entregable es más adelante dividido en dos iteraciones de 2 a 3 semanas de duración cada una, tiempo en el cual se implementa partes de las historias del cliente. De esta forma un conjunto de historias son iterativamente implementadas y empaquetadas como un producto entregable que ha sido completado y entregado al cliente, el proceso continúa de esta forma y se repite hasta que todos los entregables hayan sido implementados. Esta práctica ayuda a crear retroalimentación efectiva entre los desarrolladores y el cliente que mejorará los productos de software.

3. Extreme Programming permite al cliente participar activamente en el proyecto y en la toma de decisiones contribuyendo de esta manera mucho más de lo que los programadores pudiesen lograr solos, y esto a su vez se revierte en una buena planificación de la entrega parcial (release) . El desarrollador usa este acercamiento para evaluar las historias de los clientes que le permiten entonces calcular o determinar el esfuerzo necesario para implementar estas historias. Este proceso también implica hacer análisis de riesgo y administración. Esta planificación y análisis de riesgo ayuda al cliente a priorizar las historia que van a ser implementadas después de considerar el costo y el riesgo involucrado en cada implementación.

4. Para que el cliente se llevado a lo largo del proceso de desarrollo, una metáfora, y normalmente una palabra clave es adoptada por el equipo de Extreme Programming, tal metáfora es lo que el cliente entiende para si mismo de cómo conducir un proyecto. Esta práctica crea una comunicación efectiva entre el cliente y los desarrolladores. Las prácticas de Extreme Programming siempre animan a decidir en la clase de funcionalidad necesaria en los sistemas esperados. Esto hace posible evitar funcionalidades en sistemas impuestos al cliente.

5. Es requerido como una práctica de Extreme Programming que el cliente sea colocado con los programadores para una efectiva planeación, comunicación y retroalimentación.

6. Otra de las prácticas requeridas por Extreme Programming es la medición de la velocidad del proyecto, entrega parámetros de entrada para la planificación del proyecto lo cual es totalmente útil para el cliente, el proceso de planificación y la toma de decisiones.

7. Crear un ápice arquitectónico ayuda a reducir los riesgos del proyecto ya que permite evaluaciones tempranas del factor riego. El ápice arquitectónico es la primera fase del proyecto donde se conceptualiza de forma general el proyecto y donde se presentan los primeros estimados del mismo.

La única cosa que se pierde en las prácticas de XP es la forma de medir la satisfacción del cliente desde el punto de vista de ISO 9001:2008 ya que en XP el cliente está informado constantemente sobre la situación del proyecto y puede intervenir rápidamente si el desarrollo se aleja de sus necesidades. En lo que se refiere a las cláusulas: 7.1, 2.2.2, 7.3.2 de la norma ISO 9001 (EuroComStd, 2000), hay la necesidad evidente de requerimientos documentados, si bien no se demanda explícitamente que los documentos deberían ser provistos, pero existe la necesidad de contar de los mismos (Jerzy, N, J, W&A, 2005). Esto contradice las prácticas de Extreme programming de Historias de Usuario no necesariamente bien documentadas.

2.1.1.2 Aseguramiento de la Calidad del Producto.

El aseguramiento de la calidad en XP no se basa en formalismos en la documentación, sino en controles propios y una comunicación fluida con el cliente. Una de las características que XP promueve es el de elaborar las pruebas que aseguren obtener un software de calidad, y dentro de las actividades XP orientadas a este objetivo, las siguientes deben ser prácticas comunes y las hemos clasificados en dos grupos:

AUDITORIAS DE CALIDAD

1. Las pruebas son creadas antes de empezar la codificación. Esto hace que sea más fácil y más rápido codificar después de crear la prueba. Esto también ayuda a los programadores a preveer y entender lo que es necesario que sea hecho, crear la primera prueba ayuda a que los requerimientos a ser efectivamente tenidos en cuenta y de esta forma evitar errores por la perdida de algunos requerimientos.

2. Se incluyen pruebas de regresión; las pruebas de regresión es la prueba hecha nuevamente a un código ya probado para asegurar que la modificación hecha al sistema no introducirá errores al código original.

3. Extreme programming también promueve que una prueba debe ser creada tan pronto una incorrección (bug) sea detectada, esta práctica ayuda a facilitar las pruebas de regresión. Esto asegura que la incorrección en el código sea completamente eliminada antes de que el producto sea finalmente liberado.

ANALISIS DE PROCESOS

1. El código debe ser escrito en un procesos sistemático ya definido, esto hace más fácil de leer, comprender y reescribir el código de alguien más.

2. El código debe ser programado en parejas, de esta forma se genera código mejor y más eficiente, animando de esta manera a que se comparta el conocimiento. Así evitamos el monopolio del conocimiento, esto hace que sea fácil para alguien más modifica cualquier código cuando sea necesario, nadie es propietario del código.

3. El uso de la administración de versiones de código es también soportado por la práctica de desarrollo en pareja, esto ayuda a eliminar el monopolio del código y asegura que las parejas de programación estén continuamente siendo integradas con apertura y transparencia.

4. Los sistemas son integrados continuamente todo el tiempo, esto da una rápida retroalimentación en errores de integración y también ayuda a evitar integraciones tardías que podrían resultar en errores fatales que luego serán muy difíciles de detectar o eliminar.

5. Se sugiere que sean los probadores (testers) quienes tomen la responsabilidad de documentar las historias de usuario en proyectos Extreme Programming. Esto asegura que las pruebas sean escritas con mayor precisión y permite a los desarrolladores concentrarse únicamente en el código. Su elección se debe a que trabaja frecuentemente con los clientes, el define las pruebas funcionales del cliente, y ejecuta estas pruebas con el cliente.

Estas prácticas de Extreme Programming claramente satisfacen los requerimiento ISO 9000 como se especifica en las cláusula 7.3.5 que dice que la verificación debe ser realizada para mostrar claramente que los resultados de diseño y desarrollo cumplen con los requerimientos de entrada del diseño y desarrollo. Esta práctica clave de Extreme Programming como aporte al aseguramiento de la calidad del producto sigue la misma dirección de ISO 9000.

La práctica de programación en pareja es la única de Extreme Programming que no tiene una relación clara con alguna cláusula ISO 9000, y con el propósito de hacer relevante esta práctica desde una perspectiva ISO 9000, la programación en pareja debería ser realizada en una oficina abierta o en un

espacio de trabajo como el que es requerido en la cláusula "6.3 infraestructura y 6.4 espacio de trabajo". Esto anima a una comunicación efectiva entre el equipo de Extreme Programming, donde cada miembro del equipo tendrá la suficiente libertad para compartir ideas.

La documentación necesaria para verificar la calidad y resultados de las prueba no tiene que ser muy compleja o pesada; lo más importante es la utilidad del documento, ISO necesita registros de que estas actividades han sido probadas para una evaluación. Las historias de usuario podrían ser almacenadas electrónicamente y las tarjetas de historia (story cards) podrían ser colocadas en una herramienta electrónica dentro de un sistema de información, de esta forma se podría reemplazar la cartelera donde se las coloca habitualmente. Además, esto permitiría que aparte de la información que comúnmente se muestra en una tarjeta de historia se pueda informar sobre el progreso de cada tarea e historia, información en confirmación del cliente para completar la historia, resultados de la integración inicial y la integración final. Así mismo los resultados de las pruebas unitarias, funcionales, de regresión y aceptación deberían ser electrónicamente almacenadas.

2.1.1.3 Desempeño del Proceso.

XP no ofrece herramientas claras para efectuar los trabajos de medición y monitoreo del proceso de desarrollo, sin embargo el uso de modelos e indicadores de gestión es lo que nos va a permitir medir un proyecto y compararlo contra el tiempo programado y el costo estimado del mismo.

"No podemos monitorear y mejorar lo que no se mide", esta es una premisa básica del desempeño. Por eso es necesario el uso de técnicas que nos permitan establecer una línea base de control, lo más simple acorde a la metodología, como es:

1. Medición de la velocidad del proyecto, esto da un parámetro de entrada para la planificación del proyecto, lo cual es totalmente útil para el cliente para la planificación y toma de decisiones.

2. Cronograma de iteraciones. Las iteraciones son periodos cortos de análisis, diseño, desarrollo y prueba donde participan el equipo de desarrollo y el cliente. Las iteraciones deben ser cortas, no menos de 2 y no más de cuatro semana.

3. Establecer los entregables como hitos del cronograma.

4. Crear puntos de revisión. Hechas en cooperación con el cliente para identificar requerimientos completados y mejoras futuras. Esto puede tambien ser visto como una retrospectiva, ayuda a reducir el riesgo en un proyecto, y a su vez permite evoluciones tempranas del factor de riesgo..

5. Establecer una linea base de costos en base a los recursos utilizados y el tiempo asignado a cada uno de ellos. Esto permitirá que en caso de que el proyecto se atrase poder asignar recursos a la actividad con atraso dependiendo de las holguras del proyecto sin que haya un impacto en el costo total del mismo.

Estás técnicas de control para la administración del proyecto y valorar su desempeño no abarcan el ciclo de vida del producto, es decir operaciones de mantenimiento y mejoras posteriores, tan solo el proceso de desarrollo del producto, software en este caso específico.

2.1.2 MEDICION Y MONITOREO.

Esta sección describirá como Extreme Programming implementará ISO 9001-2008 el área de medición y monitoreo para las actividades propias de la realización del producto (software).

Las cláusulas 8.2.3 dicen que la organización necesita mostrar o probar cuan capacitado esta un proceso para lograr sus metas o resultados. Las siguientes

prácticas de Extreme Programming pueden ayudar a satisfacer este requerimiento recolectando y registrando los siguientes procesos:

1. Estimación del sobretiempo del proyecto. El tiempo extra puesto en el proyecto, aun considerando que dentro de los valores de XP no está permitido el sobretiempo.

2. La disponibilidad del cliente o la accesibilidad de su representante, para que los desarrolladores puedan hacer sugerencias y clarificar cualquier tema malentendido como un aporte al desarrollo del producto.

3. Velocidad del proyecto, métrica que actualmente determina el número de horas por día/semana/meses que el equipo de desarrollo ocupa en una tarea o trabajo en particular. Estas métricas son normalmente horas, días y meses. Por ejemplo un miembro del equipo debe ocupar un promedio de cinco horas diarias de codificación en el proyecto. Esta métrica ayuda a hacer una planificación efectiva.

4. Bitácora de integración. Esta métrica describe la tasa de integración de los sistemas y ayuda a medir en índice porcentual en la cual una pieza de código esta siendo integrado con un código que ya está trabajando. XP promueve a una frecuente integración diaria de código.

5. Modo de producción de código. Esta métrica indica si una parte de código fue producida en pareja o individualmente.

6. Velocidad de programación. Para segurar la velocidad de la programación, datos para la métrica como: líneas de código escritas por hora o día, casos de prueba escritos por hora o día, etc. deberían de ser recolectados.

ISO 9001-2008 en su cláusula 8.2.4 "Seguimiento y medición del producto" dice que desde una perspectiva de desarrollo de software debería haber un reporte que muestre las pruebas unitarias y de aceptación que hayan pasado satisfactoriamente. Esta cláusula es muy directa en su cometido; Extreme Programming puede satisfacer esta cláusula manteniendo registros de pruebas

unitarias y pruebas de aceptación que han pasado con resultados satisfactorios. También ISO 9001:2008 en la cláusula 8.3 "Control del producto no conforme" requiere que un producto que no ha satisfecho los requerimientos del cliente sea reverificado para cumplir con tales exigencias. Extreme Programming cumple con este requerimiento asegurando que los sistemas satisfacen o pasan las pruebas unitarias antes de que el producto sea liberado al mercado. Finalmente la cláusula 8.5.2 de ISO 9001-2008 "Acciones Correctivas" ayuda a chequear los defectos del producto y Extreme Programming satisface estos requerimientos creando casos de prueba por los defectos detectados. Esto es necesario para que el defecto sea eliminado antes de que el producto sea liberado al cliente.

Los datos o métricas antes mencionados habilitan a que el proceso sea monitoreado desde la iniciación del proyecto con la presentación de las historias de usuario, a través de la implementación de esas historias y la verificación de que las características del producto cumplan acorde a las historias del cliente aparejado con la calidad incrementando el número de pruebas unitarias, ajustándose exactamente a las historias presentadas, estimando exactamente las tareas a realizarse así como reduciendo el número de defectos y fallas satisfaciendo claramente la cláusula 8 de ISO "Medición, análisis y mejora"

Las organizaciones del estándar ISO y la organización Extreme programming que quieren obtener la certificación ISO 9000 pueden adoptar el siguiente proceso discutido en este capítulo y también satisfacer los requerimientos de gestión de la calidad descritos en la cláusula 4 de la norma, este acercamiento ayuda a que el estándar ISO y Extreme programming se complemente el uno al otro combinando fortalezas compatibles tanto del proceso como de estándar en la entrega de producto de software de calidad.

2.2 DESARROLLO DE LA POLITICA DE CALIDAD.

ISO 9000:00 define Política de Calidad como el conjunto de intenciones globales y orientación de una organización relativa a la calidad, expresada formalmente por la Dirección, a partir de ella, se despliega la estrategia, y de la estrategia se ha de deducir el resto de sistema que haga posible la consecución de los resultados esperados. No se debe iniciar la implantación de un sistema de gestión sin hacer de la Política la primera piedra del edificio

La Política de Calidad es un documento que debe ser entendido en todos los niveles del equipo de trabajo dentro de la Organización ejecutante del proyecto. En este caso de estudio, la organización ejecutante es la empresa desarrolladora de software o la unidad encargada de administrar el proyecto dentro de una empresa. Ej. Departamento de TI de una Organización Comercial.

El equipo de proyecto, y más específicamente su administración deben estar orientados a seguir un plan de gestión de calidad. Gestionar la calidad no es otra cosa que asegurarse que los principios establecidos en un manual o guia se incorporen al trabajo del día a día del proyecto.

2.2.1 DEFINICIÓN DE LOS PRINCIPIOS DE CALIDAD

Basados en los principios de calidad de ISO 9001-2008 y los principios seguidos por la metodologia XP todo plan de calidad debe considerar, al menos, los siguientes principios:

1. DESARROLLO ENFOCADO EN EL CLIENTE.

Todo el desarrollo del sistema se basa en las historias de usuario, es decir, visualizar el sistema desde la perspectiva del cliente sin perder de vista los objetivos principales del mismo. El equipo de desarrollo necesita comprender que Ellos dependen altamente de las relaciones con el cliente. De esta forma

para tener éxito el equipo de desarrollo necesita comprender las necesidades actuales y futuras del cliente así como sus expectativas y requerimientos.

Este principio es la conexión con la definición de calidad indicado en ISO 9001-2008; e inicia el proceso de cómo satisfacer los deseos de un cliente con la fabricación y desarrollo de un producto.

2. LIDERAZGO.

La Dirección del equipo de trabajo deberá establecer las políticas, misión y metas del equipo de desarrollo de esta forma todos los miembros deberán seguir sus lineamientos. La dirección también es responsable de crear el ambiente necesario para todo el personal que trabaja en el equipo de desarrollo, así las personas estarán más comprometidas con las metas del grupo.

3. PARTICIPACION DE LA GENTE.

Es necesario crear un ambiente de participación para obtener una continua satisfacción del cliente con mejoramientos de calidad. Todos los miembros del equipo deben estar comprometidos y participar en forma activa en este acercamiento. Esto no solo se reflejará en la calidad del producto sino en todo el proceso de desarrollo.

El núcleo de toda organización es la gente, y su compromiso hace posible que sus habilidades puedan ser desarrolladas para beneficio de la empresa, y en este caso del equipo de desarrollo. Ciertas características individuales de los miembros del equipo que puedan resultar relevantes dentro del proceso deberán ser tomadas en consideración y desarrolladas para crear colaboración, por ejemplo: independencia, capacidades conversacionales, resolución de conflictos, co-creatividad, y capacidad de aprender de la experiencia.

4. ACERCAMIENTO A LOS PROCESOS.

El proceso debe ser visto como un todo y no como entidades individuales dentro del trabajo global. Un proceso es un conjunto de actividades interrelacionadas que son repetidas en el tiempo. Los buenos resultados son alcanzados con eficiencia cuando las actividades y sus recursos son administrados como un proceso.

5. COMPRENDER EL SISTEMA DENTRO DE LA GESTION ADMINISTRATIVA.

La administración de procesos interrelacionados como todo un sistema. La comprensión, determinación y administración de los procesos que están relacionados entre ellos como un sistema entero hace que la organización sea más efectiva y eficiente en alcanzar sus metas y objetivos.

6. MEJORAMIENTO CONTINUO.

El proceso de mejoramiento continuo es un objetivo clave dentro de la organización o equipo de trabajo, y es un proceso permanente necesario para su desarrollo. Si la organización no hace de este proceso algo permanente entonces corren el riego de no buscar mejores formas de hacer una actividad o trabajo.

La base del mejoramiento continuo es el ciclo "planificar - hacer - revisar", es decir, planificar una actividad o tarea, hacerla, y luego revisar los resultados y puntos donde se puede mejorar las practicas realizadas.

7. APROXIMACION A LOS HECHOS PARA LA TOMA DE DECISIONES.

Las decisiones deberán ser tomadas usando el análisis de hechos concretos que deben estar bien fundamentados. Los hechos recolectados deben de ser tanto de carácter cualitativo como cuantitativo; sin embargo en este análisis de hechos es necesario saber diferenciar entre las variaciones naturales y

variaciones por causas distinguibles lo cual requiere de conocimiento y habilidad para poderlo hacer.

Es necesaria un estrategia que permita ejecutar el vínculo entre las decisiones que deban ser tomadas y los hechos relacionados a ellas, tal cual lo hacen las empresas de manufactura a través de herramientas administrativas y estadísticas que permiten observar de una forma más clara el vínculo entre las decisiones y los hechos.

8. RELACIONES BENEFICIOSAS MUTAS CON EL PROVEEDOR.

Si el equipo de desarrollo necesita proveerse de los servicios de un tercer participante dentro del proyecto empezando una relación cliente – proveedor, se buscará siempre el beneficio mutuo que mejore la capacidad de crear valor.

Todo proveedor externo al equipo de trabajo puede terminar afectando seriamente el resultado final de forma negativa si las relaciones con este no son satisfactorias y bien manejadas por parte de la dirección del equipo de desarrollo.

2.2.2 DEFINICIÓN DE LOS OBJETIVOS DE CALIDAD

Desarrollar una política de Calidad implica tener definidas las metas que queremos lograr con su implementación. Para la buena implementación, y posterior seguimiento, al sistema de calidad de toda organización debemos estructurar los objetivos de calidad y definirlos claramente.

Lo más importante dentro del plan de calidad, es tener alineados a la política de calidad y sus objetivos, tal como lo indica la norma. Los objetivos de calidad deben de ser coherentes con la política de calidad. Si el compromiso de la organización es "brindar un servicio oportuno o producto util", entonces los objetivos deben de ser tal que ayuden a cumplir con ese compromiso.

Figura 2.2

El desafío que enfrenta el desarrollo de la presente tesis es crear una sinergia entre la política de calidad basada en los principios generales de la norma ISO 9001-2008. Y los objetivos que persigue la metodología para desarrollo de software Extreme Programming.

Se decidió definir los objetivos de calidad del sistema propuesto basándonos en el Núcleo objetivo de Extreme Programming :

1. **Objetivo**: Satisfacer al cliente a través de prontas y continuas entregas de software de valor.

   Principio: ENFOQUE AL CLIENTE

2. **Objetivo**: Construir procesos ágiles de desarrollo que flexibilicen el software ante requerimientos de cambios que logren crear en el cliente una ventaja competitiva.

   Principio: ENFOQUE AL CLIENTE

3. **Objetivo**: Lograr sinergia y empatía entre el cliente y los desarrolladores de tal forma que sea una actividad natural trabajar juntos en el proyecto diariamente.

   Principio: ENFOQUE AL CLIENTE

4. **Objetivo**: Construir proyectos alrededor de individuos motivados, dándoles el ambiente y soporte que ellos necesitan y la confianza para conseguir hacer su trabajo.

   Principio: GESTION ADMINISTRATIVA

5. **Objetivo**: Obtener el método mas efectivo y eficiente para comunicar información cara a cara dentro de un equipo de desarrollo.

   Principio: LIDERAZGO

6. **Objetivo**: Mantener un estado de cooperación permanente entre sponsors, desarrolladores y usuarios que promuevan el desarrollo sostenible.

   Principio: PARTICIPACION DE LA GENTE

7. **Objetivo**: Continua atención a la excelencia técnica y el buen diseño que realcen la agilidad del proceso.

   Principio: MEJORAMIENTO CONTINUO

8. **Objetivo**: Maximizar la simplicidad en el trabajo que esta por hacerse.

   Principio: MEJORAMIENTO CONTINUO

9. **Objetivo**: Lograr equipo autoorganizados que propendan las mejores arquitecturas, requerimientos y diseños

   Principio: ACERCAMIENTO A LOS PROCESOS

2.3 ESTRUCTURA DE LAS FASES DEL PROCESO DE DESARROLLO DEL SOFTWARE.

Extreme Programming, como toda metodología ágil, intenta reducir la complejidad del software por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción.

El flujo de desarrollo en XP, no nos permite determinar una secuencia clara de procesos como lo podemos ver en las metodologías de desarrollo tradicionales ya que el flujo o ciclo de vida de un proyecto en XP es secuencial y recursivo, además hay ciertos conceptos que no son familiares en las metodologías

tradicionales como historias de usuario, iteración, pruebas de aceptación, etc. Como es necesario identificar claramente los procesos del flujo de desarrollo se ha identificado las actividades del proceso de desarrollo XP y se tomado los items más importantes de ISO 9001 especialmente en lo que tiene que ver con documentación necesaria, implementación, validación y verificación del sistema de software; luego se ha agrupado estas actividades en cuatro fases, que generalmente son las más identificadas con XP: la fase de exploración, la fase de planificación, la fase de producción y la fase de mantenimiento.

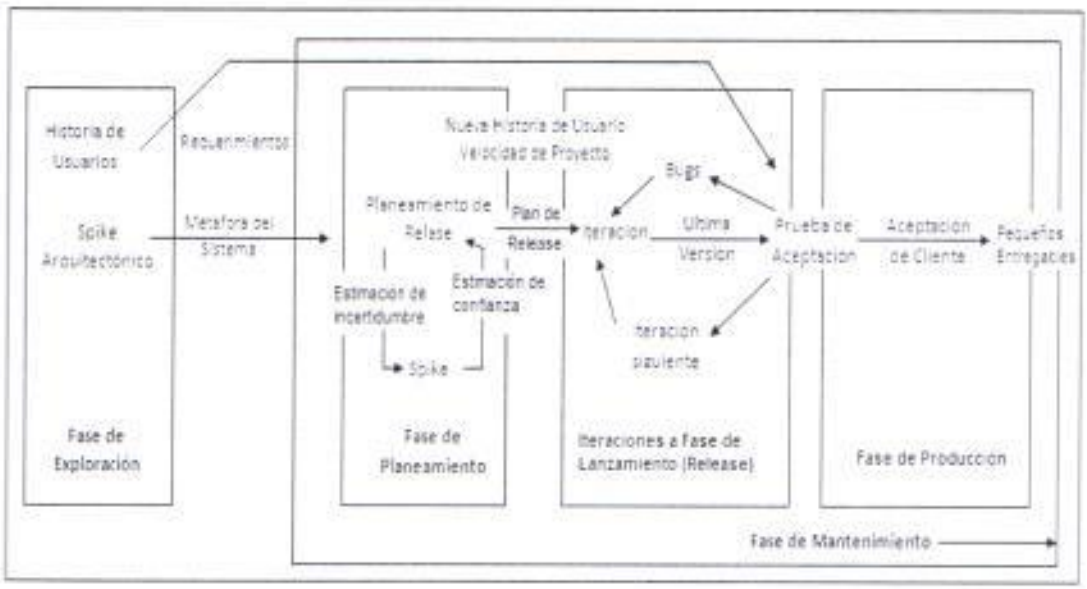

Figura 2.3

2.3.1 FASE DE EXPLORACION.

Dentro de todo el proceso de desarrollo XP intenta minimizar el riesgo por medio de la disposición permanente de un representante competente del cliente a disposición del equipo de desarrollo. Generalmente este representante debería estar en condiciones de contestar rápida y correctamente a cualquier pregunta del equipo de desarrollo de modo que no se retrase la toma de decisiones. Esta persona se convierte en un elemento trascendental es esta fase de planificación.

En esta fase se da lo que en XP se llama "Ápice Arquitectónico" (arquitectural spike). Donde lo que se busca es conceptualizar de manera general el proyecto, el equipo de trabajo revisa de forma general los requerimientos del proyecto que son de interés para el cliente y se constituyen en la base del software a desarrollar, explora todas las áreas de un problema, rediciendo de esta forma cualquier riesgo que pudiese ocurrir más tarde. Estos requerimientos serán acogidos dentro del alcance del proyecto y describirán escenarios sobre el funcionamiento del software.

Figura 2.4

En XP la fase de exploración se limita a determinar de forma general los estimados tanto en tiempo como recursos que tendrá cada historia de usuario considerando el problema central a ser solucionado, sin embargo considerando los requerimientos de calidad que deberá cumplir el procesos se hace necesario considerar las siguientes actividades dentro de la fase de exploración:

2.3.1.1 DETERMINAR EL ALCANCE DEL PROYECTO

No todos los proyectos son iguales, cada uno tiene al menos pequeñas variaciones con respecto a otros, pero finalmente lo que se busca al administrar un proyecto es cumplir en tiempo, en costo y en forma, con el objetivo del proyecto.

En esta etapa se da la estimación preliminar de tiempo, costos y recursos asegurando que el proyecto satisfaga las necesidades del cliente con la funcionalidad y calidad requerida.

En esta etapa temprana del proyecto es donde se debe definir el alcance. Existen diversas teorías de cómo desarrollar un alcance, que formato debe tener o que aspectos debe incluir, sin embargo la finalidad es la misma: la definición de lo que está y no está incluido en el proyecto.

Los puntos a considerarse al momento de elaborar un alcance con el fin de cumplir las cláusulas ISO son:

1.- Desarrollar un escrito o documento formal.

2.- Detallar claramente que actividades y procesos son parte del proyecto, es decir, el trabajo que debe ser realizado con el fin de entregar un producto con las características y especificaciones solicitadas.

3.- Definir los requerimientos funcionales y no funcionales generales del proyecto.

4.- Determinar las ventajas que se ofrecerá a las partes interesadas

5.- Definir los entregables que tendrá el proyecto de acuerdo a las historias de usuario estimadas.

6.- Determinar los criterios que se utilizaran para determinar si el entregable ha finalizado exitosamente, es decir los criterios de aceptación.

7.- Al definir el alcance, tener en consideración que lo que no está en el alcance está fuera del proyecto.

8.- Formalizar la aceptación del alcance con el Cliente.

2.3.1.2 DESARROLLAR EL PLAN DE EJECUCIÓN DEL PROYECTO

El plan de ejecución del proyecto ayudará al equipo de trabajo a determinar el tamaño y la duración del proyecto, identificar los objetivos y metas factibles y cuantificables; además permite llegar a acuerdos entre los miembros del equipo de desarrollo en cuanto a tiempos y recursos necesarios, así como el desarrollo de métodos para el seguimiento de la ejecución del proyecto.

Entre los principales componentes básicos a ser considerados dentro de la elaboración del plan de ejecución de proyecto de software XP, están:

1. Documento Alcance del proyecto

2. Política de Calidad.

3. Plan de Comunicaciones

4. Cronograma General del Proyecto detallando el número de iteraciones necesarias por cada entregable.

5. Definición de los roles del equipo de trabajo.

Otros elementos importantes y definiciones que podrían encontrarse en el plan de ejecución del proyecto podrían ser:

1.- Identificar los riesgos del proyecto y su plan de respuestas.

2.- Determinar el uso del personal necesario en cada una de las etapas del proyecto

3.- Establecer un sistema de seguimiento y evaluación del proyecto.

9.- Criterios de aceptación del plan de ejecución del proyecto.

2.3.2 FASE DE PLANIFICACION.

XP le da a esta fase el nombre informal de "El juego de la planeación", y esta es la técnica clave que se usa para crear un plan del proyecto. Aunque realmente hay dos etapas de planeación: la planeación inicial y la planeación del lanzamiento del producto entregable "release". La figura 2.5 muestra el flujo general de la fase de planificación en un proyecto XP.

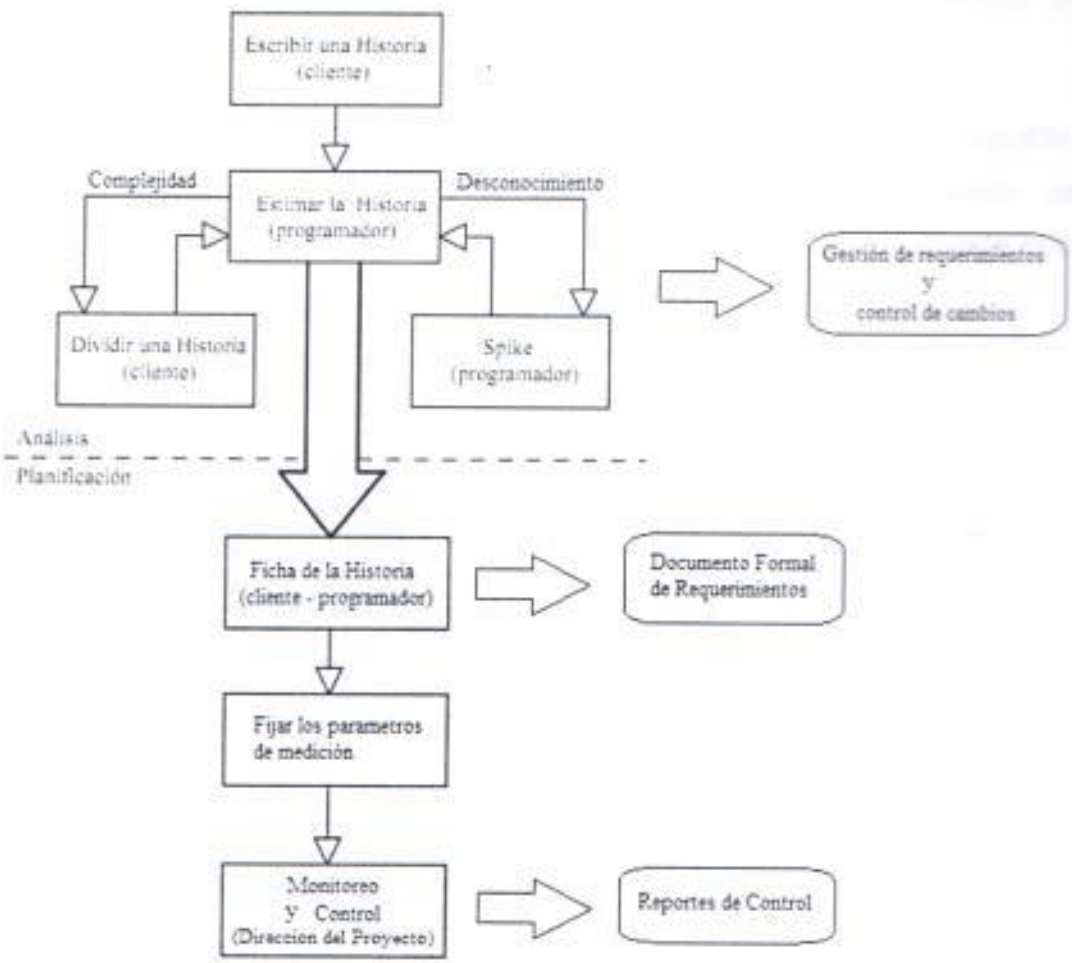

Figura 2.5

La planeación inicial es usada frecuentemente antes de que el equipo de desarrollo ha sido identificado y conformado; mientras que la planeación del lanzamiento del entregable "release" se da cuando todo el equipo del proyecto ya está conformado puesto que su participación se hace necesaria.

En la etapa de planeación el cliente y el desarrollador se sientan en un cuarto juntos y empiezan un procedimiento con actividades que son realmente simples (Jeffries, A, H, 2008) y que podemos resumir como:

1. El cliente presenta las historias de usuario.
2. El equipo trabaja en tareas de ingeniería de software a través de una lluvia de ideas
3. El desarrollador estima en una unidad de medida (que puede ser simplemente un punto) el peso de cada historia.

Basada en algún nivel de recursos predeterminado, el desarrollador identifica aproximadamente cuantas unidades de medida (Puntos) pueden ser completados en cada iteración de N semanas cada una (N es típicamente entre una y cuatro)

2.3.2.1 GESTION DE REQUERIMIENTOS

La gestión de requisitos corre alrededor de las historias de usuario creadas por el cliente y documentadas en conjunto con el equipo de desarrollo.

Es importante tener claro algunos conceptos en relación a la elaboración de las historias de usuario ya que dentro del proceso propuesto para la elaboración de un proyecto de software que cumpla con los requerimientos XP y con la norma ISO, las historias de usuario se convierte en el principal medio documentado para formalizar los requerimientos del cliente y del sistema a desarrollarse.

La historia de usuario se convierte en el punto de partida para el desarrollo de cada entregable del proyecto. Identificamos tres actividades básicas en esta parte del proceso de desarrollo:

A. Escribir las Historias de Usuario.

En esta fase el cliente desarrollará en conjunto con el grupo de programadores las historias de usuario detalladas en forma general en el documento Alcance del Proyecto que se realizó en la Fase de Exploración.

Una historia de usuario describe la funcionalidad deseada desde la perspectiva del cliente (el usuario), así mismo debe detallar otras características como: quien la necesita, como se va a utilizar y porque se va a utilizar. Los componentes básicos de Una historia de usuario se pueden resumir en tres elementos:

a. Tarjeta.- es la descripción escrita de la historia, que sirve como identificación, recordatorio y también ayuda a planificar.

b. Conversación.- es el núcleo de la historia; es el dialogo que ocurre con los usuarios, notas, grabaciones, prototipos y documentos.

c. Confirmación.- el criterio de las pruebas de aceptación que el usuario va a utilizar para confirmar que la historia fue terminada.

Una historia de usuario no es un documento técnico pero si puede ser un documento formal de requerimientos que facilitara al equipo del proyecto de tener una visión más clara de la funcionalidad del entregable y de las expectativas del cliente. Las historias de usuario se usan para capturar la esencia del valor de negocio de un sistema y están escritas en un lenguaje cotidiano

Una buena historia de usuario también sigue el modelo de INVEST: Independiente, Negociable, Estimable, Pequeña (Small), y Testeable. Veamos lo que significa.

Independiente - una historia debería ser independiente de otras (lo más posible). Las dependencias entre las historias hace que sea más difícil planificar, priorizar y estimar. A menudo, se puede reducir las dependencias haciendo una combinación de historias, o partiendo historias de forma diferente.

Negociable - una historia de usuario es negociable. La "tarjeta" de la historia es tan sólo una descripción corta que no incluye detalles. Los detalles se trabajan durante la etapa de "Conversación". Una tarjeta con demasiados detalles limita la conversación con el cliente.

Valiosa - cada historia tiene que tener valor para el cliente (para el usuario o para el comprador). Una forma muy buena de generar historias valiosas es hacer que el cliente la escriba. Una vez que el cliente se de cuenta que la historia no es un contrato y es negociable, van a sentirse mucho más cómodos para escribir historias.

Estimable - los desarrolladores necesitan poder estimar una historia de usuario para permitir que se pueda priorizar y planificar la historia. Los problemas que pueden impedirle a los desarrolladores estimar una historia son: falta de conocimiento del dominio (en cuyo caso se necesita más Negociación / Conversación); o si la historia es muy grande (en cuyo caso se necesita descomponer la historia en historias más pequeñas).

Pequeña - una buena historia debe ser pequeña en esfuerzo, generalmente representando no más de 2-3 personas/semana de trabajo. Una historia que es más grande va a tener más errores asociados a la estimación y alcance.

Testeable - una historia necesita poder probarse para que ocurra la etapa de "Confirmación". Recordemos que desarrollamos aquello que no podemos probar. Si no podemos probarlo, nunca vamos a saber si lo terminamos. Un ejemplo de historia no testeable sería: "el software tiene que ser fácil de usar".

B. Estimar el Esfuerzo Necesario por cada Historia de Usuario.

La piedra de tope de la planificación es la estimación de cuanto un trabajo tomará tiempo en realizarse (Auer&Miller, 2001) Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores usando como medida el punto. Un punto, equivale a una semana ideal de programación. Para planificar el trabajo desde el punto de vista técnico las historias de usuario deben ser descompuestas en tareas pequeñas para las cuales también se realiza una estimación (el esfuerzo asociado a estas tareas nunca supero los tres días de programación). Las tareas siempre se ponderaron en un máximo de 1 punto, aunque generalmente no debería tomar más de dos días en terminarse (Auer&Miller, 2001). Cada historia de usuario debe estimarse en no más de 3 puntos.

Los entregables están divididos en iteraciones de no más de 3 semanas, asignando a cada iteración un conjunto de historias de usuario que serán implementadas. Durante el desarrollo de la iteración y en particular al final de cada iteración se realizará un seguimiento del plan de la iteración y del entregable. Es de esperar situaciones relativas a historias de usuario tales como: aparición de nuevas, postergación en cuanto a la iteración en la cual se implementa, no finalización en la iteración planificada y modificación durante la iteración o una vez completada en una iteración previa.

Las iteraciones son espacios de tiempo donde se juntan esfuerzos tanto del cliente como del equipo de desarrollo para sacar adelante una historia de usuario (objetivo). Las iteraciones pueden ser para planificar la funcionalidad de un sistema, para desarrollar una funcionalidad nueva y para mejorar una ya existente. Una iteración necesita ser lo suficientemente grande para hacer progresos significativos, pero también lo suficientemente pequeña como para que esos progresos no se hagan sin haber sido revisados correctamente, es decir sin que se pierda el control de calidad de los mismos. El tamaño recomendable de una iteración está entre dos y cuatro semanas, ni más ni menos que eso.

C. Registro Formal de Requerimientos.

Si bien es cierto XP plantea el uso de tarjetas de historia de usuario (story cards), donde de forma breve el usuario describe el proceso sobre el cual debe trabajar el equipo de desarrollo, estas tarjetas no se constituyen en un documento formal de requerimientos del cliente, por lo tanto en este punto vamos a formalizar el requerimiento del usuario documentando el mismo e indicando cada uno de los elementos importantes que deberán ser considerados por cada historia de usuario. Si el equipo de desarrolladores considera que algún elemento nuevo no considerado en esta sección deba agregarse, lo podrá hacer, finalmente lo que aquí damos es un modelo general que nos permita cumplir con las exigencias de calidad del estándar ISO

	Historia de Usuario
Número: 2	**Usuario:** Asistente Comercial
Nombre historia: Selección de pedidos a procesar (cliente preferente)	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: XXXXXXXX	
Descripción:	
Observaciones:	
Requerimientos Funcionales:	**Requerimientos No Funcionales:**

Tabla 2.1 Plantilla modelo par registrar las historia de usuario

El equipo del proyecto deberá documentar el requerimiento (historia de usuario) usando una plantilla similar a la de la tabla 2.1, indicando detalles como: prioridad y esfuerzo estimado, entre otros.

El procedimiento formal necesario para documentar el requerimiento deberá al menos considerar los siguientes pasos:

1. Determinar los requerimientos funcionales y no funcionales para cada historia de usuario.

2. El cliente y el equipo de programadores darán prioridad a las historias.

3. El programador realizará las estimaciones del esfuerzo necesario para cada una de las historia. Realmente esta planificación se da por cada entregable (release) como se lo mencionó anteriormente. Teniendo en cuenta el esfuerzo asociado a las historias de usuario y las prioridades que tenga el cliente será quien defina los entregables del sistema, que le serán de valor.

4. El cliente definirá el alcance de cada release o entregable y en conjunto con el equipo de trabajo elaboran el documento de requerimiento donde se hace una narrativa de la historia seleccionada, la misma que es aprobada por el cliente. Se tomaran acuerdos en conjunto sobre el contenido de cada entrega, así como el cronograma aprobado por el cliente.

5. Se determinará el tiempo en que una entrega deba obtenerse y el tiempo de duración de la fase de planificación del entregable (release). La gestión de requisitos se la ha diseñado para que sea lo más simple posible. El equipo de desarrollo identifica los requisitos funcionales y no funcionales del sistema elaborando a su vez un documento de requisitos.

Este documento pasará a formar parte de los requerimientos de las especificaciones funcionales, que luego servirá como documento de referencia para la gestión de calidad, una vez firmado por el cliente y el programador.

2.3.2.2. EJECUTAR ITERACIONES, DESARROLLO Y PRUEBAS UNITARIAS.

Ya en el desarrollo de las iteraciones es necesario planificar el trabajo de una semana a la vez. Dentro del trabajo a realizar esta el de mantener reuniones no mayores a diez minutos para sesiones de diseño donde el primer objetivo será elaborar un diseño lo más sencillo posible y hacerlo incrementalmente, es decir invertir tiempo diario en revisiones al diseño que luego derive en el desarrollo de código simple sin descuidar el resolver las necesidades presentes del cliente y pasar las pruebas de aceptación del cliente harán que el sistema entre en operación a tiempo cumpliendo con lo planificado. Mientras el cliente no tenga algo con el cual pueda realmente experimentar en la práctica, sus requerimientos no podrán ajustarse a sus deseos.

Luego de contar con el diseño el programador deberá escribir las primeras pruebas para el código de acuerdo a los requerimientos del cliente, estas, bajo este esquema se produce una característica clave de la metodología XP que es el desarrollo orientado a las pruebas, y en esta fase las primeras pruebas a superarse son las pruebas del programador o también llamadas unitarias.

La idea es que antes de escribir código se deba tener claro cual es el objetivo del código a través de pruebas creadas con antelación a la codificación. Bajo este esquema la primera prueba fallará al no haber código creado y luego los programadores irá creando código para pasar las pruebas, y luego retornaran al ciclo para ir creando más pruebas seguidas por el código que se deberá crear. Aquí reside en uno de los mayores beneficios de XP, ya que con esta práctica ya que el 100% del código que se obtiene está probado.

Se recomienda seguir los siguientes pasos para escribir las pruebas unitarias:

1. Crear un ambiente de pruebas (en los lenguajes de programación orientado a objetos podría ser una clase de pruebas,.
2. Crear programas u objetos que implementen la clase de pruebas
3. Escribir la prueba o código modelo.

4. Ejecutar la prueba
5. Modificar el código modelo
6. Liberar el código modelo.

2.3.2.3. MONITOREAR LA VELOCIDAD DEL PROYECTO.

El equipo de desarrollo mantendrá un registro de la "velocidad" de desarrollo, 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.

La planificación se realizará basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuantas historias se pueden implementar antes de una fecha determinada o cuanto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuantos puntos se pueden completar. Al planificar según el alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el numero de iteraciones necesarias para su implementación.

2.3.3 FASE DE PRODUCCIÓN

En esta fase se requiere asegurar la calidad del producto previo su puesta en producción. El producto entregable deberá pasar por pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

EN la fase de producción se ha considerado la implementación de documentos formales tales como los resultados de las pruebas y la aceptación formal del cliente. La figura 2.6 muestra el flujo de la fase de producción.

2.3.3.1 IMPLANTACION.

Tener un plan de implementación del software se constituye en una parte importante de la fase de producción porque ayuda al equipo de trabajo a tener una guía para la implantación de software una vez concluidas las pruebas unitarias hachas por el equipo de desarrollo.

Las actividades principales que deben ser consideradas dentro de un plan de implantación del software son las siguientes:

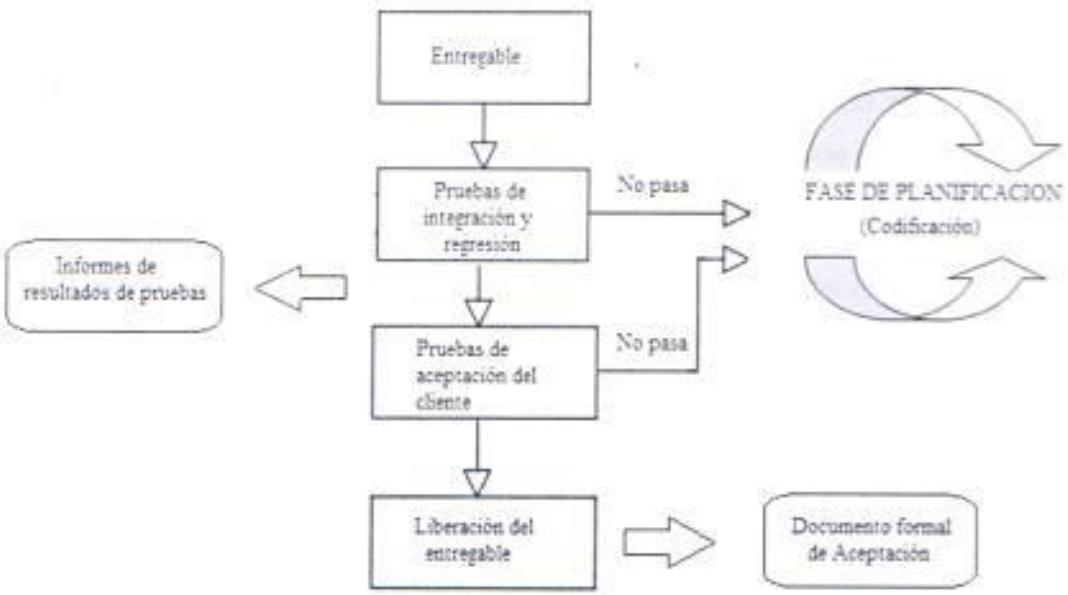

Figura 2.6

A. Pruebas de Regresión e Integración.

Las pruebas de regresión es la actividad que ayuda a asegurar que los cambios (debido a pruebas u otros motivos) no introduzcan un comportamiento no deseado o errores adicionales, por ello se prueban nuevamente componentes evaluados anteriormente, asegurando que los cambios introducidos en el software no han afectado su funcionamiento. El objetivo de las pruebas de regresión es eliminar el efecto onda, es decir, comprobar que los cambios sobre un componente de un sistema de información, no introducen un comportamiento no deseado o errores adicionales en otros componentes no modificados.

Las pruebas de regresión intentan descubrir las causas de nuevos errores (bugs), carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software, inducidos por cambios recientemente realizados en partes de la aplicación que anteriormente al citado cambio no eran propensas a este tipo de error.

Este tipo de cambio puede ser debido a prácticas no adecuadas de control de versiones, falta de consideración acerca del ámbito o contexto de producción final y extensibilidad del error que fue corregido (fragilidad de la corrección), o simplemente una consecuencia del rediseño de la aplicación.

Cada pareja de programadores debería ser responsable de chequear que el nuevo código o corrección que se está implementado no afectará a su propio código anterior o al de otra pareja de programadores, a través de ciertas técnicas para mitigar los riesgos. En la tabla 2.2 de detallan los puntos principales a ser considerados en esta actividad dentro del plan de implementación.

- Una muestra representativa de pruebas que ejercite todas las funciones de software.
- Pruebas adicionales que se centran en la funciones del software que se van a ver probablemente afectadas por el cambio

- Pruebas que se centran en los componentes del software que se han cambiado

Esta puntos de observación a seguir en las pruebas de regresión se vuelven muy necesarias en sistemas que requieren de un alto nivel de integración entre los componente entregables que forman parte del proyecto.

Cuando hablamos de integrar en XP nos referimos a esta actividad como un proceso de integración incremental de código dentro del sistema, el cual luego es probado como un sistema total al menos una vez por día. Esperar hasta que finalice el proyecto para integrar todos los módulos en un sistema podría ser tedioso, consumir mucho tiempo, altamente costoso y podría resultar en una pobre calidad del sistema, el modelo ideal de integración continua permite que la construcción y ejecución de pruebas sea realizada cada vez que el código cambia.

Lo más difícil para el programador es conocer el impacto real de una actualización fundamental sobre las funcionalidades de la aplicación. La integración continua permite al desarrollador tener esta visión más global de la aplicación ya que las pruebas de la aplicación se hacen en un entorno similar de producción.

No.	Descripción	Resultado
1	Describir el proceso de integración de los entregables para con el sistema	Listado de resultados observables
2	Definir la acciones a seguir cuando los resultados no son satisfactorios	Listado de resultados de pruebas no satisfactorias
3	Identificar los responsables de las pruebas y las correcciones	Matriz de responsabilidades

Tabla 2.2 Pasos para obtener un conjunto de pruebas de aceptación

En esta actividad es muy útil y recomendable contar con herramientas de software y hardware para manejar la integración de software. Existen servidores de integración continua que permiten integrar código fuente de manera más ágil y limpia que haciéndolo manualmente donde existe más riesgo de cometer errores.

B. Pruebas de Aceptación.

El propósito de las pruebas de aceptación es garantizar que se han tenido en cuenta todos los casos de pruebas posibles para validar la solución informática a un requerimiento nuevo o solicitud de cambio, garantizando que al momento de entregar el producto este cuente con un nivel de calidad apropiado.

No.	Descripción	Resultado
1	Identificar todos los posibles resultados observables de la historia	Listado de resultados observables
2	Identificar los resultados que terminan la historia y los que permiten continuar dentro de la historia	Listado de resultados observables clasificador en terminales y no terminales
3	Identificar los caminos de ejecución posibles	Listado de caminos de ejecución posibles y a cual de los resultados identificados conduce
4	Asignar un conjunto de valores válidos y valores del entorno a cada camino de ejecución para obtener el resultado esperado.	Listado de caminos de ejecución, con sus resultados esperados y los valores que permiten obtener dicho resultado.
5	Eliminación de caminos redundantes.	Listado de caminos de ejecución, valores de prueba y resultados que se convertirán en pruebas de aceptación.

Tabla 2.3 Pasos para obtener un conjunto de pruebas de aceptación

Cada historia de usuario que representa una característica en el desarrollo XP ha sido asociada a una prueba de aceptación que es determinada por el cliente

e implementada por el equipo de desarrollo. Generalmente en XP está pruebas son automatizadas ya que el factor tiempo es crítico como para realizar pruebas manuales al sistema.

Las pruebas de aceptación necesitan estar disponibles en la misma iteración en que la historia de usuario es programada, de esta forma, a través de pruebas funcionales automatizadas, los programadores podrán saber claramente cuales son las necesidades y, así mismo, el cliente podrá ver los progresos de un sistema en funcionamiento

Las correcciones al sistema son mostradas al cliente una vez que todas las pruebas pasan. Consecuentemente el sistema está continuamente creciendo y mejorando. Si existiese dentro del equipo un área de Control de Calidad está podría implementar sus propias herramientas para las pruebas de aceptación.

Los pasos que proponemos en este trabajo para que el cliente pueda generar un conjunto de pruebas de aceptación que verifiquen completamente la funcionalidad de una historia de usuario las resumimos en la tabla 2.3.

C. Liberar el Producto.

Una de las partes más importantes de trabajar en propiedad de código colectivo es el proceso de liberar el producto entregable o los cambios realizados al sistema. Cuando una pareja de programadores XP está trabajando, su código camina hacia tres fases en este proceso: si es un entregable local, solo para uso del equipo de programación; es un entregable candidato a entrega, disponible para el resto del equipo de desarrollo; o, listo para entrar a producción. La tabla 2.4 explica más en detalle la definición de cada fase.

Una vez que el código fuente es publicado como versión oficial, el entregable es puesto a disposición del cliente en el ambiente de producción.

No.	Descripción	Resultado
1	Local	Esta es la primera fase del desarrollo, la pareja ha empezado a trabajar y sus cambios en el código no están disponibles para ningún otro programador
2	Candidato a entrega	Los programadores han finalizado su tarea, y están listos para empezar el proceso de lanzar sus cambios a otros programadores
3	Lanzado al repositorio común	En este momento ya es una versión actual oficial, lo que significa que el código está garantizado para trabajar, todas las pruebas unitarias pasaron al 100%. Los cambios entregados estarán disponibles para todos los otros programadores

Tabla 2.4 Fases del proceso de lanzamiento.

D. Cierre del Proyecto.

En esta fase 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 (requerimientos no funcionales). Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. El cierre del proyecto también puede ocurrir debido a que el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

2.3.4 FASE DE MANTENIMIENTO

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del

sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

2.3.4.1 ELABORAR UN PLAN DE MANTENIMIENTO.

El plan de mantenimiento ayudará al equipo de trabajo a tener una guía operativa para los trabajos de mantenimiento una vez que el software ha pasado a producción para ser utilizado por los usuarios.

Las actividades principales a ser consideradas dentro de un plan de implantación del software son las siguientes:

A. Manejo de Errores.

Los defectos reducen el valor del software para el cliente, lo cual va en contra del principal propósito de la metodología XP el cual es entregar valor al cliente a través del software. El proceso para manejar los defectos del software debe ser lo más sencillo y ágil posible, para lo cual nosotros lo hemos dividido en tres etapas:

1. Reportar los problemas. Tarea que le corresponde al usuario y cliente.
2. Programar las correcciones. Es importante que el cliente priorice los problemas y los programe en futuras iteraciones. Generalmente los problemas más urgentes son los más complejos por lo tanto se hará necesario programarlos en la iteración que está corriendo.
3. Prueba y corrección del problema. Identificado el problema se debe hacer una prueba de aceptación y sobre esta corregir el código actual.

B. Refactoring.

El refactoring no es otra cosa que la técnica de mejorar el código sin cambiar su funcionalidad. Haciendo refactoring se aplica el mejoramiento continuo

dentro del proyecto mejorando el diseño constantemente. Los programadores evaluan continuamente el diseño y recodifican lo necesario. Si el código no funciona adecuadamente o puede ser mejorado, entonces hay que cambiarlo. Si el código es difícil de entender, puede no ser claro, no tiene mucho significado o es difícil de modificar quiere decir que el código no funciona adecuadamente y es mejor reescribirlo, lo cual significar hacer "refactoring". La finalidad es mantener un sistema enfocado a proveer el valor de negocio mediante la minimización del código duplicado y/o ineficiente. Es recomendable hacer refactoring al código constantemente para su fácil comprensión y modificación, algunas veces se hace necesario crear un espacio exclusivo para llevar a cabo las tareas de mejoramiento del código.

Ningún sistema podrá ser considerado obsoleto con un mantenimiento constante, además de ayudar a que la propiedad del código pertenezca a todo el grupo de desarrolladores y no a uno solo o una sola pareja, lo cual hace mucho más fácil de sobrellevar la ausencia temporal o definitiva de un programador.

C. Gestión de Cambios.

En esta parte es importante definir en el plan de mantenimiento los procedimientos por los cuales se aprobaran o rechazaran los cambios en el proyecto. Un cambio al alcance original generalmente incluye cambios dentro del sistema que si no son bien gestionados pueden derivar en la inestabilidad del sistema originando problemas posteriores más graves. Es recomendable incluir en este proceso una bitácora de incidentes ("Issue Log") compartida y accesible por todos los involucrados, un sistema de seguimiento de los cambios aprobados, y un mecanismo de niveles de aprobación para autorizar los cambios, como así también un sistema para actualizar el plan original del proyecto con los cambios agregados en la ejecución.

Estos procedimientos deben estar disponibles no solo cuando el proyecto finaliza y empieza la etapa de mantenimiento dentro del ciclo de vida del producto, sino también cuando el proyecto aún esta en la etapa de ejecución, lo

cual haría que el cambio solicitado entre como un requerimiento en las etapas de planificación y producción del proyecto.

2.4 ACTORES Y RESPONSABILIDADES.

Los roles en un proyecto XP pueden tener muchas perspectivas, todas ellas encaminadas a que ocurra un desarrollo efectivo de software. La principal práctica que sugiere XP es que el equipo completo de trabajo este conformado por una variedad de personas trabajando juntas, más allá de los conceptos de cada rol, si estos no cumplen esta primera práctica, el equipo sufre, por eso los roles de cada proyecto no deben ser fijos y rígidos. El éxito es tener a todos contribuyendo con lo mejor que cada uno tiene que ofrecer para alcanzar el éxito del equipo, si bien es cierto los roles fijos ayudan a que las personas se especialicen cada vez más en sus habilidades, interfiere en el hecho de que algunos miembros del equipo pueden resultar muy buenos realizando actividades ajenas al rol asumido, por ejemplo: Los programadores pueden escribir historias si ellos son buenos escribiendo historias claras y precisas; un Project Manager podría sugerir mejoras en la arquitectura del sistema si El está en una mejor posición para poderlo hacer.

Vamos a tomar como modelo de roles dentro del equipo de desarrollo aquellos que recomienda Kent Beck de manera general, considerando que dependiendo el proyecto de software a desarrollarse es posible que haya necesidad de eliminar algún rol, que dos roles estén dentro de uno solo o inclusive adicionar un rol nuevo no considerado.

1. Responsables de Pruebas (Testers)

El encargado de pruebas ayuda al cliente como identificar y escribir las pruebas funcionales (pruebas de cliente), además de entrenar a los programadores en técnicas de pruebas de software. El tipo de programación "probar-primero" dan

como resultado un conjunto de pruebas que ayudan a que el proyecto se mantenga estable. El rol de los probadores toma importancia tempranamente en el desarrollo, ayudando a definir y especificar que se constituirá en un funcionamiento aceptable del sistema antes de que la funcionalidad haya sido implementada.

En un ciclo semanal la primera cosa que le sucederá a las historias elegidas es que ellas se convertirán en pruebas automatizadas a nivel del sistema. Esto se convierte en una buena palanca para fortalecer habilidades para escribir pruebas. Los Clientes pueden tener una buena idea del comportamiento general que ellos quieren ver en el sistema, pero los probadores son buenos localizando los mejores caminos y preguntando que sucedería si algún error ocurre.

Una vez que las pruebas para la semana son escritas y entregadas, los probadores continúan escribiendo nuevas pruebas como implementaciones con nuevos detalles no cubiertos que necesitan ser especificados. También el escribe las pruebas de aceptación y guía al cliente que realiza estas pruebas; el probador es muy entendido en el trabajo con los clientes, desarrolladores y administración con respecto a este punto.

Probadores pueden también trabajar más adelante para automatizar y mejorar pruebas. Finalmente cuando un programador está detenido en un problema de testeo dentro del cual no puede salir, un probador puede hacer pareja con el programador para ayudarlo a resolver el problema.

2. Programadores.

Los programadores son el núcleo de XP, sin ellos no hay desarrollo de software. El programador es responsable de dos cosas:

- *Hacer las mejores estimaciones posibles de tal forma que el cliente pueda hacer las mejores decisiones posibles de negocios.*

- Entregar funcionalidad que se ajuste confiablemente a los requerimientos implicados en la mayoría de historias de usuario.

El desarrollador no es responsable por usar la última tecnología posible o de entregar las segundas historias más importantes. El desarrollador es responsable por informar al cliente el impacto que tomar atajos podría tener en las futuras historias.

XP deberá practicar mejoramiento del diseño, diseño simple, aprender a programar en pareja, desarrollo conducido por pruebas (pruebas unitarias), hacer las estimaciones a partir de las historias de usuario, y determinar cuales son las tareas que deberán ser tomadas en orden para desarrollar cada historia de usuario. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

3. Cliente.

El cliente es el otro 50% del corazón de XP, conforme lo describe Beck: "El programador sabe como programar, el cliente conoce que programar. El buen cliente es aquel que usará el sistema a ser desarrollado". Las tareas del cliente son: escribir las historias de usuario y las pruebas funcionales para validar su implementación, asignar la prioridad a las historias de usuario y decidir cuáles se implementarán en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.

4. Encargado de seguimiento (Tracker) .

El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones.

También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes, esto lo hace a través de estimar la velocidad del proyecto usando la retroalimentación de los programadores preguntando y escuchándolos acerca de lo que Ellos está haciendo en el momento. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración. El deberá ser cuidadoso de no interrumpir el proyecto demasiadas veces.

5. Diseñadores de Iteración.

Los diseñadores de iteración en un equipo XP eligen las metáforas en general para el sistema, escriben historias, y evalúan el uso del sistema desplegado para encontrar oportunidades de nuevas historias. Direccionar las preocupaciones de los usuarios eventuales es una prioridad del equipo. Las herramientas del diseño de iteración tales como la personas y objetivos ayudan al equipo a tomar conciencia del mundo del usuario, sin embargo ellos no son sustitutos para la conversación con la gente real.

Muchos de los consejos para los diseñadores de iteraciones están basado en un modelo de desarrollo por fases: los diseñadores de la primera iteración entienden lo que el sistema se supone que debe hacer, y luego los programadores hacen eso. Las fases reducen la retroalimentación y restringen el flujo de valor. El beneficio mutuo es posible entre el diseño de la iteración y el resto del equipo XP sin separar el desarrollo en fases.

En un equipo XP, los diseñadores de iteración trabajan con el cliente, ayudando a escribir y clarificar historias. Los diseñadores de iteración pueden utilizar todas las herramientas usuales durante este proceso. Ellos también analizan el actual uso del sistema para decidir cuales son las siguientes necesidades del sistema.

6. Arquitectos.

Los arquitectos en un equipo XP buscan y ejecutan refactorings a gran escala, escriben pruebas a nivel sistema que fuercen la arquitectura, e implementan historias. Los arquitectos aplican su experiencia en el tiempo a lo largo del proyecto. Ellos dirigen la arquitectura del proyecto a través de su evolución. La arquitectura para un sistema pequeño no debería ser la misma que para uno grande. Mientras el sistema es pequeño el arquitecto se asegura que el sistema solo tenga la poca arquitectura necesaria. En tanto el sistema va creciendo, el arquitecto se asegura de que la arquitectura mantenga el tamaño acorde.

Una de las tareas importantes para el arquitecto en un equipo XP es la de particionar sistemas. Particionar no es una tarea que se da por adelantado. Primero un equipo pequeño crea un sistema pequeño, luego el equipo encuentra las líneas de fraccionamiento naturales y divide el sistema en partes independientes relativamente para expansión. Los arquitectos ayudan a elegir las líneas de fraccionamiento apropiadas y luego siguen al sistema como un todo, manteniendo en mente la gran figura, así como los grupos se enfocan en su más pequeña sección.

7. Administradores del Proyecto.

Los administradores de proyecto en un equipo XP tienen la función de ser facilitadores de la comunicación dentro del grupo y coordinar la comunicación con el cliente, proveedores, y el resto de la organización.

Los administradores de proyecto actúan como historiadores del equipo recordando cuanto se ha progresado en el proyecto y cuanto falta por hacerse. Los administradores del proyecto tienen que ser creativos al momento de empaquetar la información que debe ser mostrada a los ejecutivos y sus pares.

En XP la planificación es una actividad más y es constante, no es solo una fase. Los administradores de proyecto son los responsables de mantener los planes sincronizados con la realidad y, dado su mejor posición, son quienes

deben de conducir hacia el mejoramiento a todo el equipo en el proceso de planificación mismo.

La información fluye en ambas ibas, hacia dentro y fuera del equipo de trabajo, los administradores de proyecto facilitan la comunicación entre los clientes, patrocinadores, proveedores y usuarios con todo el equipo del proyecto. Una forma de hacer esto es saber escoger a la persona correcta dentro del equipo de proyecto con la persona correcta fuera del equipo según se den las necesidades de comunicación en reuniones de trabajo.

8. Administradores del Producto.

Los administradores del producto escriben historias, recogen temas e historias quincenalmente y semanalmente, y responden interrogantes en áreas no especificadas y no cubiertas en la implementación de la historias.

El administrador del producto tiene la tarea de ayudar a decidir prioridades en base a los requerimientos asumidos y los actuales, adaptándose a lo que realmente esta sucediendo en la Organización.

Los administradores del producto animan la comunicación entre clientes y programadores, asegurando que las cosas más importantes concernientes al cliente son escuchadas y puestas en acción por parte del equipo.

9. Ejecutivos.

La principal función del staff ejecutivo es la de proveer al equipo del proyecto del animo, confianza y responsabilidad necesaria para lograr el éxito deseado, logrando alinear las metas del equipo del proyecto con las metas de la compañía.

Los ejecutivos son libres de pedir un informe sobre cualquier aspecto del proyecto en cualquier momento, y este debe ser dado en forma clara y precisa de parte del equipo de trabajo.

Entre las actividades principales del staff ejecutivo estan las siguientes:

- Defender el proyecto
- Obtener presupuestos para el proyecto
- Aceptar la responsabilidad de problemas extendidos del administrador del proyecto
- La firma de documentos formales

10. Escritores Técnicos.

Son los encargados de elaborar todas las publicaciones de carácter técnico relativas al proyecto en desarrollo. La principal labor es la de proveer una temprana retroalimentación acerca de las características del producto y las de crear relaciones cercanas con los usuarios.

La primera parte del rol viene debido a que los escritores técnicos en el equipo son los primeros en ver las características, frecuentemente mientras ellas aún son solo un esbozo. Este proceso de aprendizaje en conjunto con el equipo de desarrollo crea un sistema de retroalimentación dentro del equipo mismo.

La segunda parte de rol es crear relaciones cercanas con los usuarios: ayudándolos a aprender acerca del producto, escuchando sus observaciones y dirigiendo sus confusiones en nuevas publicaciones técnicas o en la implementación de nuevas historias.

Las publicaciones técnicas pueden tomar formas de: tutoriales, manuales de referencia, revisiones técnicas, video y audio.

El trabajo del escritor técnico es dinámico porque el sistema cambia constantemente, las especificaciones no se congelan al inicio del proyecto, sino que cambian en el transcurso del desarrollo del proyecto, por lo tanto actualizar las publicaciones técnicas debe ser un trabajo constante. No es muy práctico plasmar las publicaciones técnicas en papel, salvo que el proyecto se haya

entregado y tenga algún tiempo en funcionamiento, lo más práctico y aconsejable es que se encuentre en un medio digital que pueda ser actualizable fácilmente o mejor aún de ser posible publicarlo en la intranet de la organización.

11. Usuarios

Los usuario en un equipo XP ayudan a escribir y levantar las historias, además de ayudar a tomar decisiones de dominio público. Los usuarios son muy valorados si ellos tienen conocimiento y experiencia trabajando en sistemas similares. Los usuarios deben tener en mente que ellos están hablando por una comunidad entera y que sus decisiones pueden tener amplias consecuencias negativas en el futuro si ellos no están seguros sobre lo que se está decidiendo.

2.5 INDICADORES DE MEDICION Y MONITOREO.

Esta sección describirá como Extreme Programming implementará las áreas de monitoreo y medición de ISO 9001-2008.

La cláusula 8.2.3 de ISO 9001 dice que la Organización necesita mostrar o probar como es capaz un proceso de alcanzar sus logros o resultados. Las siguientes prácticas de pueden ayudar a satisfacer este requerimiento recolectando y registrando los siguientes procesos:

1. INDICADORES DE MEDICION

Indicador	Descripción
Sobre tiempo del proyecto	Estimar el tiempo extra dado que XP no permite realizar sobretiempo en un proyecto
Disponibilidad del cliente	La accesibilidad a un representante del cliente para que los desarrolladores clarifiquen o hagan sugerencias sobre temas no comprendidos como aporte al desarrollo del producto
Velocidad del Proyecto	Métrica para determinar el número de horas por día, semana o mes, que el equipo de desarrollo gasta en una tarea en particular. Esta métrica se muestra normalmente como días, semanas o meses. Por ejemplo, un miembro del equipo debe gastar en promedio 5 horas diariamente codificando en el proyecto. Esta métrica ayuda a hacer la planificación más eficiente.
Bitácora de Integración	Esta métrica describe la tasa de integración del sistema, ayuda a medir la tasa en el cual una pieza de código esta siendo integrada con un código ya implementado y en funcionamiento, XP anima a la integración frecuente y diaria del código.
Modo de producción del código	Esta métrica nos indica si una pieza de código fue producida individualmente o en pareja.
Velocidad de Programación	Esta métrica mide la velocidad de programación en función de las líneas de código escritas por hora o día, casos de prueba creados por hora o día, etc. deberían de ser recolectados

2. INDICADORES DE MONITOREO

Cláusula ISO 9001:2008	Descripción
Cláusula 8.2.4: "Monitoreo y medición del producto" dice, en perspectiva de desarrollo de software, que debería haber un reporte que muestre los productos que pasaron las pruebas unitarias y de aceptación.	Esta cláusula es estricta, y XP puede satisfacerla manteniendo registros de los desarrollos que pasaron las pruebas unitarias y de aceptación.
Cláusula 8.3: "Control de no-conformidades". Requiere que un producto no cumpla con las expectativas del cliente sea reverificado para cumplir con estas no-conformidades.	Extreme Programming cumple con el requerimiento asegurando sistemas que satisfagan o pasen las pruebas unitarias antes de que el producto sea lanzado al mercado.
Cláusula 8.3: "Acciones Correctivas". Ayuda a chequear la recurrencia de defectos del producto	Extreme Programming satisface este requerimiento creando casos de prueba para defectos detectados. Esto es necesario para que el defecto sea eliminado completamente antes de relanzar el producto al consumidor.

Estás métricas o información a monitorear habilita al proceso para ser monitoreado desde el inicio con la presentación de las historias del cliente, a través de la implementación de las historias y verificación de que las características y funcionalidad del producto están acorde a los requerimientos del cliente en las historias de usuario, todo esto emparejado con el manejo de la calidad incrementando el número de pruebas unitarias, con el exacto cumplimiento de las historias presentadas y la exacta estimación de tareas a

realizarse se logra reducir el número de defectos o fallas, satisfaciendo claramente la cláusula 8.0 acerca de Medición, análisis y mejoramiento.

Las organizaciones de estándar ISO y la organización de Extreme Programming que quieren obtener la certificación ISO 9000 pueden adoptar el proceso discutido en este capítulo y también satisfacer los requerimientos de gestión de calidad como se describen en la sección 4.3, este acercamiento ayuda al estándar ISO 9000 y Extreme Programming a complementarse el uno al otro combinando fortalezas de ambos: estándar y proceso, entregando de esta forma productos de software de calidad.

2.6 DOCUMENTACIÓN

Uno de los puntos con los cuales XP entra en conflicto con la norma ISO, ya que difiere de las buenas prácticas de Extreme programming como reconocimiento a la satisfacción del cliente en la norma ISO 9001-2008 en las cláusulas 7.1, 7.2.2 y 7.3.2, es la evidente necesidad de requerimientos documentados. Esto contradice las prácticas de Extreme Programming de historias de usuario que no se documentan bien, aunque los principios de Extreme Programming no prohíben la documentación, la única cosa que difiere de estos principios es elaborar demasiada documentación que no es necesaria, que podría distraer a los programadores de la codificación y el cumplimiento de metas de cada iteración. La principal cuestión es como introducir documentación en Extreme Programming, lo cual no se convierte en la mejor práctica al ser una metodología de peso ligero. La esencia de esto es permitir al programador una concentración total en la codificación de los programa, lo cual debería ser su principal enfoque.

El propósito de esta sección no es la de plantear modelos o plantillas de la documentación formal necesaria para la ejecución de un proyecto XP, en este punto el tema se vuelve un poco contradictorio con la filosofía XP, ya que la misma reduce al mínimo toda documentación necesaria sobre el proyecto y

más bien anima a producir un muy reducida documentación básica necesaria, ya que bajo la premisa de que el proyecto es dinámico y cambiante se requieren de continuas planificaciones, revisiones y cambios debidamente organizados que basan su desempeño en las buenas practicas y la experiencia de la dirección del proyecto.

Sin embargo, debiendo cumplir con requisitos de la norma ISO 9001 que nos indica que el sistema de gestión de calidad debe incluir la siguiente documentación para calificar como tal:

- Declaraciones documentadas de una política de la calidad y de objetivos de la calidad.
- Manual de Calidad.
- Procedimientos documentados requeridos de esta norma internacional.
- Los documentos, incluidos los registros que la organización determina que son necesarios para asegurarse la planificación, operación y control de sus procesos

Nos centraremos en los dos últimos puntos ya que los dos primeros puntos se refieren a documentación que debe tener la organización en general dentro de su Sistema de Gestión de Calidad, tales como: el alcance del sistema de calidad, procedimientos documentados de las áreas donde se aplica el sistema de calidad y una descripción de la interacción entre los procesos del sistema de gestión de calidad.

Este trabajo se va a centrar en la documentación necesaria para cumplir con el capítulo 7 de la Norma: "Realización del Producto", y garantizar una gestión de calidad en el proceso de desarrollo y ejecución. Para este efecto hemos dividido el ciclo del proyecto en tres grupos generales de procesos: el inicio del proyecto; el desarrollo y ejecución; y, el monitoreo y control

2.6.1 INICIO DEL PROYECTO.

En esta etapa es importante documentar todos los acuerdos previos al desarrollo del proyecto. Estos documentos deben proporcionar información

relativa a las estimaciones preliminares del proyecto, los objetivos y la funcionalidad del producto final.

1. **El alcance del sistema a desarrollarse.** Este documento debe considerarse el más importante en la parte previa al desarrollo del producto, ya que un proyecto en el cual no se realiza la definición del alcance en forma correcta, es un proyecto condenado a grandes problemas en su planificación, ejecución y control. En esta etapa se recomienda usar la herramienta más distintiva y fundamental que es la Estructura de División del Trabajo o WBS por sus siglas en inglés. Esta técnica consiste en identificar los principales entregables, incluyendo aquellos entregables relacionados a la administración del proyecto. Prácticamente es la base para poder saber que se requiere hacer y de realizar los planes de tiempo, costo, calidad, riesgo, adquisiciones, comunicaciones y recursos humanos, de forma integral.

2. **El acta de constitución del proyecto.** Desarrollar el documento necesario que autorice formalmente el proyecto. En el mismo debería quedar reflejado de forma general los requisitos iniciales que satisfagan las necesidades y expectativas de los interesados. El fin principal del documento es establecer la colaboración entre la organización ejecutante y la organización solicitante (cliente).

3. **Plan de Ejecución del Proyecto.** En esta instancia se declara lo que el software debe hacer o lograr a través de los distintos procesos y funciones que han de automatizarse, con indicación de las entradas y salidas. En esta instancia no se avanza con el diseño técnico de detalle, Ej. Bases de datos. Modulo, programas y procedimientos, arquitectura de seguridad y control, etc. Las especificaciones deben contemplar todas las características del sistema que se consideren críticas respecto a la funcionalidad requerida para la solución. Cada una de ellas en mayor

detalle será tratada en las historias de usuario durante la fase de desarrollo.

2.6.2 DESARROLLO Y EJECUCIÓN DEL PROYECTO.

Durante todas las fases de un proyecto XP la planificación se vuelve como una ola envolvente, es decir siempre está presente la actividad de planificar y más aún en el desarrollo del producto.

El alcance del proyecto es un documento importante a considerar por parte del equipo de desarrollo al momento de la ejecución del mismo porque le da una visión de las expectativas de los Patrocinadores del proyecto.

Los principales documentos necesarios para desarrollar esta etapa del ciclo del proyecto son los siguientes:

1. **Requerimientos de Usuario.** El desarrollo de los requerimientos del entregable quedan plasmados dentro de las historias de usuario. Este documento se convierte en la principal guía para el equipo de programadores del proyecto. El mismo debe considerar el siguiente contenido: Objetivo, puntos de esfuerzo necesarios, narrativa de la historia, requerimientos funcionales, no funcionales, y prioridad entre otros.

2. **Informe del Cronograma del Proyecto.** El cronograma es la representación técnica del proyecto ya que nos permite relacionar en un gráfico los objetivos, las actividades, los tiempos y los plazos. Con el cronograma describimos las actividades o tareas que se llevaran a cabo durante el desarrollo del proyecto con sus fechas de inicio y fin, precedencias de cada tarea, recursos requeridos por cada una de ellas, todo esto representado en un diagrama de Gantt.

   En el plan de ejecución el cronograma debe elaborarse conjuntamente con el equipo responsable del desarrollo del proyecto para que den su

aprobación o se modifiquen las fechas dependiendo de las necesidades de la organización, de los calendarios de las personas responsables o de cualquier otra circunstancia que pueda entorpecer el desarrollo del proyecto.

El informe de cumplimiento del cronograma del proyecto nos indicara el avance del proyecto en porcentaje conforme a los días o semanas de trabajo cumplidas contras el tiempo programado para terminar una tarea. Los cambios al cronograma establecido deben ser consensuados por todo el equipo de trabajo.

De todas las habilidades requeridas para administrar un proyecto, la gestión del cronograma es quizás la más fundamental. Dependiendo de la dinámica del proyecto, el líder puede estar en una posición que requiera usar constantemente su experiencia y creatividad para poder concluir el proyecto de acuerdo a las expectativas. En caso de atrasos, se deben evaluar los recursos y opciones disponibles para regresar el proyecto a lo planificado.

2.6.3 MONITOREO Y CONTROL.

El Monitoreo y Control nos permite seguir el desempeño del proyecto en cada paso de su ejecución, de forma que se puede identificar los posibles problemas oportunamente y adoptar las acciones correctivas que permitan mantener el proyecto dentro de los límites establecidos en las líneas base: Alcance, Costo, Tiempo y Calidad.

1. **Informe de Pruebas de Aceptación.** Detalla las pruebas de aceptación e integración realizadas por el equipo de calidad o pruebas del proyecto y los resultados obtenidos en cada una de ellas. Este documento se convierte en una herramienta de control ya que asegura el cumplimiento de los requerimientos funcionales del proyecto acorde con lo detallado en las historias de usuario del entregable. El documento de las pruebas se

convierte en los registros de productos satisfactorio y no satisfactorio requeridos en la norma ISO.

2. **Reportes de Control de Cambios.** Esta es una actividad de control que se desarrolla a lo largo del proceso de software. Los cambios dentro del desarrollo del software pueden ocurrir en cualquier momento y se debe estar preparado para identificar el cambio, controlar ese cambio y garantizar que el cambio quede bien implantado. Asi mismo la actividad de control debe garantizar la integridad con el sistema completo ante cualquier modificación del software. Este conjunto de actividades de seguimiento y control inician cunado comienza un proyecto de ingeniería de software y terminan solo cuando queda fuera de circulación.

CAPÍTULO 3

CASO DE ESTUDIO.

3.1 INTRODUCCIÓN

El proyecto que se tomo como modelo para la implementación de un sistema de calidad aplicando la metodología XP fue el desarrollo de un sistema de para la administración de la gestión de pedidos y ventas para una empresa perteneciente a la industria del Plástico. El objetivo del proyecto es desarrollar un módulo de negociación que se deberá instalar en la oficina del distribuidor para que este pueda gestionar los pedidos de sus clientes finales negociando en línea, a través del sistema, condiciones comerciales con el proveedor desde que es ingresada la solicitud de cotización del pedido hasta que este se convierta en orden de venta en firme pasando por todo un proceso de gestión.

El proyecto fue desarrollado por la empresa MEDIALOGIC y su equipo de desarrollo. El equipo de desarrollo en su inicio estaba conformado por un Líder de Proyecto, cuatro programadores, un técnico de hardware y dos implementadores quienes además trabajaron en labores de control de calidad.

Cada uno de los miembros fue preparado en la metodología de programación extrema con especial énfasis en cada uno de los roles que iban a desempeñar cada uno de Ellos. Al finalizar el periodo de capacitación Ellos ya tenían un conocimiento bastante adecuado para emprender el desarrollo del proyecto usando la metodología de XP. Así mismo se adecuó un cuarto para que en el mismo todos los miembros del equipo de trabajo puedan trabajar juntos.

Una vez que el equipo de desarrollo fue capacitado en la metodología de Programación Extrema, se procedió a conformar el equipo de calidad encargado de desarrollar: la política de calidad, los objetivos y el proceso de desarrollo; todos ellos basados en la norma ISO 9001:2008 aplicado a Programación Extrema. El equipo lo conformaron: un auditor ISO 9001:2008, un especialista en XP, el Líder de Proyecto y un representante del cliente-beneficiario final.

Cuando la política, objetivos y proceso de desarrollo quedaron definidos se hicieron conocer los mismos a todo el equipo de trabajo XP.

El desarrollo del proyecto duró 20 semanas desde Noviembre del 2010, hasta Mayo del 2011. El lenguaje de programación usado fue Visual Basic.net, para los desarrollos en ambiente escritorio; y, HTML y ASP.net para aplicaciones basadas en los navegadores web.

La metodología de desarrollo usada en este proyecto es Programación Extrema, con entregables continuos y definidos. Las iteraciones del proyecto no tuvieron la misma duración, algunas veces fueron de una y media semana, otras veces tres semanas, pero la mayoría fueron de dos semanas. Hubo seis iteraciones en el proyecto y se habían proyectado ocho para finalizar el mismo.

3.2 CONFORMACION DEL EQUIPO DE TRABAJO.

Cuando el proyecto empezó, los roles que estuvieron definidos desde el inicio fueron: Director del Proyecto, el Lider (Coach), Responsable del Seguimiento (Tracker), Probador (Tester). Luego fue necesario una revisión de los roles conforme avanzaba el proyecto y el equipo completo quedo finalmente configurado de la siguiente forma:

1. Líder de Proyecto.

Los roles previo de Director de Proyecto y Coach quedaron finalmente representado en una sola persona, el Líder de Proyecto. Para una mejor comprensión el rol del Director de Proyecto es igual a la figura del Gran Jefe, pero dentro del equipo no había una persona con las características necesarias para cumplir con este Rol; por eso, se definió el rol de Líder de Proyecto, que es el cargo más similar al de Coach en Extreme Programming.
El trabajo del Líder de Proyecto es planificar el proyecto y que cada cosa vaya en la misma dirección de forma que el proyecto llegue a ser exitoso. El es responsable de la comunicación externa del proyecto, programar y estar a

cargo de las reuniones del equipo, y tratar de resolver los problemas que el equipo pueda llegar a tener.

2. Responsable del Cliente.

Es el nexo en sitio del cliente que debería actuar como un cliente XP debido a la imposibilidad de tener al cliente real en el lugar de trabajo del equipo. El comprendía las necesidades del cliente y estuvo a cargo de las relaciones con este. Su responsabilidad fue siempre estar en contacto con el cliente real y así poder responder las preguntar del desarrollador acerca del producto.

También fue el responsable de desarrollar y planificar las historias de usuario y las tareas a realizarse.

3. Equipo Técnico.

El equipo técnico estaba conformado por cuatro programadores y un técnico de hardware.

3.1. Desarrolladores de código fuente. Los desarrolladores fueron los responsables de desarrollar el sistema y hacer las estimaciones para su trabajo. Ayudaron también en las reuniones de planificación y diseño de la aplicación, además de ser responsable de la configuración del sistema.

3.2. Desarrollador GUI. Fue el desarrollador responsable por el desarrollo de la interfase gráfica de usuario, especialmente de la interfaces web. Asegurándose de que la capa del sistema fuese consistente a lo largo de desarrollo.

3.2. Desarrollador de Seguridades y Bases de Datos. Fue el programador responsable de desarrollar las seguridades del sistema a nivel de los programas de usuario y de la base de datos, así como de la creación de los

componentes de las bases de datos como son tablas, roles de usuario, procedimientos almacenados, etc.

3.3. Técnico de Hardware. Estaba a cargo de la gestion requerimientos de hardware del sistema, de la implementación de las soluciones de hardware y de administrar los equipos de computación de desarrollo y pruebas así como los respaldos de los fuentes y de las bases de datos.

4. Equipo de Calidad.

El equipo técnico estaba conformado por 2 probadores de calidad (testers) y una persona que documenta.

4.1. Probadores de Calidad. La responsabilidad de probar los entregables del proyecto asegurándose de la calidad del producto y los procesos. Era tarea de Ellos también asegurar que las métricas estén acorde a los que se esta midiendo, y que sean usadas a lo largo de las iteraciones del proyecto, así como en los reportes de progreso.

Era el responsable de las entregas finales de la aplicación. El tenía que asegurarse de que el producto fue desarrollado, probado e integrado antes de las presentaciones en cada iteración. Muchas veces también asumieron la responsabilidad de la configuración del sistema.

4.2. Documentador. Era el responsable de generar la documentación del proyecto y de distribuirla a las personas involucradas en el mismo de acuerdo a las necesidades de información que tuviesen cada una de ellas. Eran responsables de la actualización y archivo de cada documento generado en el proyecto como son estándares, políticas, procedimientos, acuerdos, minutas de reuniones, etc.

ORGANIGRAMA DEL PROYECTO

Figura 3.2

3.3 PRACTICAS XP USADAS.

El objetivo de este capitulo es evaluar la aplicación y cumplimiento de las doce prácticas propuestas por la metodología XP. Examinamos todas y les otorgamos una calificación entre 0 y 3; 0 cuando no existió aplicación, 1 cuando la aplicación fue básica, 2 media y 3 completa:

Practica 1: El cliente en sitio.	Calificación: 1
No existió realmente un cliente en el espacio de trabajo del proyecto. Se definió un representante del cliente (responsable de la relación con el cliente) dado que el desarrollo del proyecto no se llevo a cabo en las instalaciones de la empresa cliente sino en las oficinas de la empresa desarrolladora. Este representante del cliente estuvo en constantes reuniones con los funcionarios de la empresa cliente transmitiendo los acuerdos entre partes y requerimientos al equipo de desarrollo.	

Practica 2: Diseño simple.	Calificación: 2
El equipo trabajó casi siempre aplicando un diseño simple al productos, sin embargo algunas veces fue necesario agregar cierta complejidad al código debido a los requerimientos funcionales del software.	

Practica 3: Propiedad colectiva del código.	Calificación: 2
Para la mayoria de parte del código existió propiedad colectiva del código, esto debido a que no siempre se programó en parejas.	

Practica 4: Estándares de Codificación.	Calificación: 3
El equipo de desarrollo compartió estándares de codificación dentro de todo el grupo.	

Practica 5: Mejoramiento del Diseño.	Calificación: 3
El equipo hizo mejoras en el diseño una vez o varias veces por día, sin embargo en muchas ocasiones no se cumplió con la adecuada documentación de los cambios realizados al diseño previo. Al final si se contó con un documento del diseño actualizado.	

Practica 6: Juego de Planeación.	Calificación: 3
El equipo trabajó con el juego de planeación en cada iteración, y lo hizo desde el primer día de cada iteración. Esta práctica fue la mejor entendida y ejecutada por el equipo de desarrollo.	

Practica 7: Desarrollo conducido por pruebas.	Calificación: 1
Uno de los problemas más críticos para el cumplimiento de esta práctica se debió a que el cliente no quería hacer las pruebas de aceptación. El desarrollador junto con el responsable del cliente desarrollaron las pruebas de aceptación, pero estas no fueron chequeadas por el cliente o usuario final,	

porque no las entendieron. Las pruebas unitarias fueron hechas para algunas partes del sistema. Los problemas aparecieron cuando había que definir como hacer para probar las interfaces gráficas y las aplicaciones web, por ejemplo, o como probar los componentes relacionados con las interfaces de usuario. El equipo desarrolló sus propias técnicas de pruebas, pero aún así fue difícil poder probar cada componente

Practica 8: Metáfora del sistema.	Calificación: 1
El equipo de desarrollo presentó una metáfora del sistema. Ellos tuvieron el conocimiento para hacer aquello, pero no supieron como usarlo y como obtener un beneficio de el. Según muchos autores este es una de las prácticas más difíciles de entender en XP y se puso de manifiesto en las prácticas XP del equipo de desarrollo.	

Practica 9: Pequeños entregables.	Calificación: 3
Todos los entregables no tuvieron el mismo tamaño, el desglose de trabajo en pequeños paquetes fue aceptable y los tiempos de trabajo fueron desde 1.5 semanas a 3 semanas, y funcionó para lo que el equipo quería con el cliente. Hubieron ciertos problemas al inicio ya que muchos entregables no tenían el valor agregado suficiente para ponerlos en práctica y necesitaron de otros elementos entregables para complementarlos. En este caso el diseño y la funcionalidad de esos entregables fue mal concebido.	

Practica 10: Programación en pareja.	Calificación: 2
Se crearon dos parejas de programación y uno tuvo pareja de programación en el proyecto. La mayor dificultad se encontró con una pareja donde sus miembros no tuvieron igual nivel de habilidades en programación, lo cual hacía muchas veces demorar el trabajo de codificación.	

Practica 11: Integración continua.	Calificación: 3
Existieron diferentes medios físicos donde se guardó el código desarrollado para lo cual se definió el espacio individual y el espacio colectivo. Cuando el código esta en desarrollo y pruebas de programación el mismo se almacena en el espacio individual asignado a cada pareja de programación. Cuando las pruebas de programación terminan el código se pasa al espacio colectivo listo para ser probado e integrado con el otro código por el grupo de pruebas.	

Practica 12: Ritmo sostenido.	Calificación: 3
El equipo usó las 40 horas semanales de desarrollo, pero cuando el personal falto por enfermedad o por alguna otra causa o razón personal, debieron trabajar horas extras para llenar la brecha que se formo en el cronograma del proyecto debido a las ausencias.	

3.4 ACTIVIDADES XP USADAS.

Las actividades analizadas en este capítulo tienen que ver con las principales tareas que se realizan por el equipo de desarrollo. Las actividades son confrontadas con los requerimientos que tiene XP sobre las mismas y su calificación esta dada por el grado o nivel de cumplimiento estipulado por la metodología y está entre 0 y 3; 0 cuando su cumplimiento fue nulo, 1 cuando su cumplimiento fue básico, 2 medio y 3 completo.

Actividad 1: Codificación.	Calificación: 2
Es muy difícil que un grupo de 10 personas compartan una comprensión total del código. En las primeras iteraciones todos trabajaron comprendiendo y conociendo el código completo, pero conforme el proyecto fue avanzando y el tamaño se fue incrementando, la propiedad colectiva no fue posible hacerla. Hubieron algunas personas que fueron especialistas en una parte del código y	

ellos obtuvieron un gran conocimiento en esa parte. Otros tuvieron un gran conocimiento en varias partes del código pero no en su totalidad. Usualmente las personas trabajaron con su propia parte del código.

Actividad 2: Pruebas.	Calificación: 1
En la primera iteración se hicieron las pruebas unitarias y de aceptación para todo el código escrito. El cliente debía escribir las pruebas de usuario pero Ellos no estaban interesados en cumplir esa parte y al inicio no fue entendida por parte del cliente. El equipo de desarrollo empezó escribiendo por ellos mismos las pruebas de aceptación pero cuando fue notorio el desinterés de la parte de cliente por las pruebas, entonces el equipo de desarrollo lo dejo de hacer desde la tercera iteración. En relación a las pruebas de programación el equipo hizo pruebas unitarias para una parte del sistema, pero por otra parte hay personas del grupo de pruebas que hicieron revisiones de código y pruebas de integración. Esto es todo lo que se hizo para el cumplimiento de pruebas, no hay más desarrollo conducido por pruebas.	

Actividad 3: Escuchar al cliente.	Calificación: 3
El equipo tuvo reuniones con el cliente al final de cada iteración, donde ellos obtuvieron una retroalimentación de parte del cliente. Ellos mostraban el producto (entregable) y lo que Ellos habían hecho desde la última iteración y obtenían la clase de cambio que el cliente necesitaba. También habían reuniones con cierta frecuencia donde todos podían hablar acerca de sus problemas, las soluciones encontradas y como el desarrollo podía ser mejorado. El sistema fue chequeado casi diariamente por el cliente, debido a que se creo un ambiente en línea para pruebas y revisión. Mucha retroalimentación fue recibida por parte del cliente.	

Actividad 4: Diseño.	Calificación: 3
El tipo de diseño tradicional (diseño en cascada) fue desarrollado solo cuando fue necesario, por ejemplo, cuando un grupo de personas estuvieron	

trabajando sobre los mismos componentes. Ellos decidieron como estos componentes se iban a relacionar el uno con el otro, En la primera iteración el equipo tuvo un alto nivel de desarrollo de diseño en cascada para la arquitectura, pero el diseño fue simple sin entrar en mucho detalle. No se gastaba más de 30 minutos en diseño cascada por cada historia de usuario.

El equipo hizo grandes mejoras al diseño de la arquitectura antes de que el software sea liberado. Hubo tres personas encargadas de esto casi cuatro días antes de la entrega del software. Esto fue hecho dos veces en todo el proyecto, sin embargo no fue un problema al momento de añadir nuevas funcionalidades, debido a que no se hicieron grandes cambios en esta instancia del proyecto. Finalmente, casi a diario se hicieron algunas pequeñas mejoras al diseño en el código propio para mejorar rendimiento y estabilidad.

3.5 PROGRAMA DE METRICAS.

Se definieron dos tipos de métricas en este proyecto. Primero las métricas de productividad y progreso, y segundo las métricas de pruebas.

Las métricas de productividad y progreso están relacionadas con las métricas usadas en Extreme Programming (Tabla 3.5.1):

No.	Métrica	Descripción	Fórmula
1	Punto de Tarea (Task Point)	Refleja las horas usadas para desarrollar una determinada tarea. TP estuvo configurada en 8 horas en el inicio del desarrollo, pero podía cambiar de acuerdo a las habilidades personales de cada uno.	TP = 8
2	Velocidad de tarea individual (individual task velocity)	Es una forma de medir cuanto trabajo es desarrollado por un individuo. , donde H es el número de horas trabajadas en tareas propias, y TP es el número total de puntos de tarea completadas en la iteración previa.	VTI = TP/H

3	Punto de historia de usuario	Equivale a 40 horas ideales	PHU = 40
4	Horas de exploración	Horas que los desarrolladores deben ocupar investigando para implementar una historia de usuario	HE
5	Velocidad del Proyecto	Es el número de puntos de historias de usuario estimados usados para trabajar en una iteración usando el conocimiento de las iteraciones previas	VP

Tabla 3.5.1

Las métricas de pruebas (Tabla 3.5.2) no fueron desarrolladas hasta la quinta iteración. Estas métricas no existen en XP ya que la metodología está más enfocada en lel software trabajando que en el mejoramiento del software, pero fue necesario definirlas para cumplir con el capítulo de Aseguramiento de la Calidad de la norma ISO 9000:

No.	Métrica	Descripción	Fórmula
1	Pruebas Satisfactorias	Refleja el porcentaje de pruebas de aceptación que pasaron en relación al total de pruebas de aceptación de una iteración a otra.	%PS = (PTA – PS) / PTA x 100
2	Pruebas no satisfactorias	Refleja el porcentaje de pruebas de aceptación que no pasaron en relación al total de pruebas de aceptación de una iteración a otra.	%PNS = (PTA – PNS) / PTA x 100
3	Log de Integración	Refleja el porcentaje de pruebas de integración que no pasaron en relación al total de pruebas de integración de una iteración a otra.	%PI = (PTI – PI) / PTI x 100

Tabla 3.5.2

3.6 PLAN DE COMUNICACIÓN IMPLEMENTADO.

Los principales objetivos que se cumplieron con el plan de comunicación implementado fueron:

1. Responder a las necesidades concretas del proyecto y atender las peticiones que cada miembro del proyecto pueda tener.

2. Informar de forma general e individual a cada uno de los miembros del equipo.

3.6.1. DEFINICION DEL PLAN

La tabla 3.2 muestra la matriz de responsabilidades para generar la información de reportes del proyecto.

Tipo de Reporte	Responsable	Dirigido a	Frecuencia
Project Charter	Líder de Proyecto	Stakeholders del Proyecto	Inicio del Proyecto
Reporte de Requerimientos Funcionales del Entregable	Líder Funcional	Líder de Desarrollo	Inicio del Proyecto
Cronograma del proyecto (cumplimiento)	Equipo de Desarrollo	Líder de Proyecto	Una vez a la semana
Reporte de Estado del Proyecto	Líder del Proyecto	Responsable del Cliente.	Quincenal
Reporte de pruebas satisfactorias	Probadores de Calidad	Líder de Proyecto y Equipo de Desarrollo	Una vez a la semana
Reporte de pruebas no satisfactorias	Probadores de Calidad	Líder de Proyecto y Equipo de Desarrollo	Una vez a la semana
Acta de Entrega – Recepción	Líder de Proyecto	Responsable del Cliente	Al finalizar el proyecto

Tabla 3.6.1

3.6.2. DESCRIPCION DE REPORTES.

Tipo de Reporte	Contenido	Proposito	Método de Difusión
Project Charter	Información del acta de constitución del proyecto	Definir los alcances del Proyecto	Medio impreso
Reporte de Requerimientos Funcionales del Entregable	Alcance de cada entregable del proyecto	Definir la funcionalidad que cada entregable debe tener al momento de liberarlo para el cliente	Medio impreso
Cronograma del proyecto (cumplimiento)	Avances del cronograma, holguras y estado de cada actividad del proyecto	Informar al cliente sobre los tiempos para el desarrollo del proyecto	Medio electrónico e impreso
Reporte de Estado del Proyecto	Estado del proyecto en términos fechas, milestones, bugs resueltos y pendientes, riesgos, control de cambios, riesgos	Informar al Cliente del proyecto sobre el avance del desarrollo del proyecto	Medio electrónico e impreso
Reporte de pruebas satisfactorias	Pruebas de integración y aceptación del usuario que pasaron satisfactoriamente luego de que el área de programación libero en entregable para el control de calidad	Medir el desempeño y control de calidad del área de programación	Correo Electrónico
Reporte de pruebas no satisfactorias	Pruebas de integración y aceptación del usuario que no pasaron satisfactoriamente luego de que el área de programación libero en entregable para el control de calidad y que regresaron al área de programación	Medir el desempeño y control de calidad del área de programación	Correo Electrónico
Acta de Entrega – Recepción	Términos en que se finaliza el proyecto y firmas de los directivos que lo certifican	Formalizar la entrega del proyecto finalizado al cliente y dejar constancia que se cumplieron todos los requerimientos en el proyecto.	Medio impreso

Tabla 3.6.2

3.7 CUMPLIMIENTO DE LOS REQUERIMIENTOS ISO.

Para evaluar los resultados del sistema de calidad implementado se utilizó la guía TickIT. El esquema TickIT es un conjunto de normas que ayuda a las organizaciones de software a desarrollar sistemas de gestión de calidad que son pertinentes a sus procesos de negocios y que cumplen con los requisitos de la norma ISO 9001.

La siguiente tabla fue desarrollada usando la información obtenida de los resultados y las entrevistas a todos los miembros del equipo de desarrollo del proyecto, además se hizo cross-checking para verificar y validar los resultados con los usuarios del proyecto.

La ponderación que se uso en la calificación fue la siguiente: 0 cuando no se cumplió , 1 cuando el cumplimiento fue básico y sin registro, 2 cuando se cumplió pero no se registró y 3 cumplimiento completo:

Requerimientos ISO	Grado de cumplimiento	Comentarios acerca de XP
4. Sistema de Gestión de Calidad		
4.1 Requerimientos Generales	2	Algunos de los requerimientos generales fueron desarrollados por el equipo, pero no por XP mismo como nos podremos dar cuenta en las siguientes líneas de esta tabla
4.2 Requerimientos de Documentación		
4.2.1 General	1	Algunos documentos fueron establecidos y mantenidos de acuerdo al estándar, pero en el caso de la documentación de diseño estuvieron en un nivel muy alto. Las historias de usuario fueron considerados como especificaciones de requerimientos. Registros de las reuniones no estuvieron disponibles, y los registros de pruebas fueron establecidos solamente en la última parte del proyecto

4.2.2 Manual de Calidad	3	El manual de calidad fue establecido y mantenido
4.2.3 Control de Documentos	2	Documentación tal como documentos de calidad e historias de usuario fueron revisadas por una persona diferente al autor. Documentos de diseño no fueron elaborados.
4.2.4 Control de Registros	1	Las reuniones no fueron grabadas, los registros de pruebas fueron desarrollados luego de que el desarrollo empezó. XP no dice nada acerca del control de registros
5. Responsabilidad de la Dirección		
5.1 Compromiso de la Administración	3	El Líder del Proyecto estuvo comprometido con el proyecto trabajando en reportes de progreso; codificando y desarrollando las políticas y el manual de calidad; y en reuniones con el cliente. El tuvo un alto compromiso en el proyecto.
5.2 Enfoque al Cliente	3	La satisfacción del cliente fue obtenida desde la visión XP, mostrando una parte del producto en cada iteración.
5.3 Política de Calidad	3	La política de calidad fue desarrollada por el líder del proyecto y comprendido por el resto del equipo.
5.4 Planificación		
5.4.1 Objetivos de Calidad	3	Se definieron como objetivos de calidad los planteados en la presente tesis.

5.4.2 Planificación de Sistemas de Gestión de Calidad	-	El tema acerca de la planificación en toda la organización dentro de una visión estratégica no es aplicable aquí, debido a que Ellos no pertenecen a una compañía. Ellos fueron contratados para un proyecto. Acerca de la planificación en un nivel funcional están especificados los planes de calidad, objetivos de calidad, estándares de codificación, etc. Por otro lado los documentos del producto están pobremente definidos, para XP la documentación son el código y las pruebas, que han sido hechas a través del desarrollo.
5.5 Responsabilidad, Autoridad y Comunicación		
5.5.1 Responsabilidad y Autoridad	3	En lugar de que recaiga la completa responsabilidad sobre una sola persona, todo el equipo es responsable por lo que se va a entregar. Hay roles y responsabilidades definidas, así como todas las fases del proyecto.
5.5.2 Representación de la Administración	3	Fue credo el rol del grupo de calidad que es responsable por el buen funcionamiento del sistema de gestión de calidad. Esto fue agregado a XP, más no es parte de El.
5.5.3 Comunicación Interna	3	Uno de los puntos más altos en XP es que hay un gran nivel de comunicación entre los miembros del equipo debido a que Ellos trabajan dentro del mismo espacio físico
5.6 Revisión de la Administración		
5.6.1 General	1	En el proyecto no existió un buen sistema para manejar las revisiones de la administración. XP está basado en revisiones diarias debido a la programación en pareja, pero esto no se documentó.

5.6.2 Entrada de la Revisión	1	Solamente se hizo revisiones sobre las historias de usuario con las personas involucradas en cada una de ellas.
5.6.2 Salida de la Revisión	0	No existió
6. Administración de Recursos		
6.1 Provisión de Recursos	3	El espacio de trabajo se lo determinó conforme a los requerimientos de XP. Se contrato personal de acuerdo al las exigencias de conocimiento y habilidades técnicas de las herramientas tanto de software y hardware requeridos en el proyecto y definidas por el Líder del proyecto
6.2 Recursos Humanos		
6.2.1 General	3	El personal estuvo trabajando como un grupo de prueba, para asegurar la calidad y la satisfacción del cliente. En un proyecto puro de XP el equipo debería trabajar como una sola entidad para mejorar la calidad del producto y la satisfacción del cliente.
6.2.2 Competencia, concienciación y entrenamiento	3	A través del proyecto, algunas de la gente con más experiencia estuvo dedicado a el. El entrenamiento en XP fue desarrollado por charlas que se dictaron antes de empezar el proyecto
6.3 Infraestructura	3	Cumple los requisitos de XP. Un espacio de trabajo de 6 x 5 mt. con una sala de reuniones adjunta de 3 x 3mt. Cuentan con una línea telefónica y servicio de Internet inalámbrico
6.4 Ambiente de Trabajo	3	Área de trabajo con aire acondicionado y aislado del ruido de la calle, adecuada iluminación y limpieza diaria. Dos baños para los miembros del equipo.
7. Realización del Producto		

7.1 Planificación de la Realización del Producto	1	Como una diferencia entre XP y la forma tradicional de hacer planeación es que XP esta enfocado en los detalles de uno o dos iteraciones. No es posible planificar más allá de eso.
7.2 Procesos relacionados con el Cliente		
7.2.1 Determinación de Requerimientos relacionados con el producto	2	Como ya se especificó antes, los requerimientos estuvieron especificados en un alto nivel, y como se los priorizó no fue definido. El equipo los priorizo en un forma especifica durante las reuniones de planeación. Los requerimientos o funcionales no fueron considerados sino hasta la parte final del proyecto..
7.2.2 Revisión de requerimientos relacionados con el producto	3	Las técnicas de revisión no están bajo control. Ellas estuvieron definidas pero no concretamente aplicadas en el proyecto. La actividad de revisión es ejecutada directamente por el Grupo de Pruebas (Tests) y el cliente.
7.2.3 Comunicación con el Cliente	3	Uno de los puntos que XP remarca es la retroalimentación con el cliente a través de la práctica del cliente en sitio. En lugar de que ellos tengan un intermediario que se contacte con el cliente real, es mucho mejor que el equipo de desarrollo se contacte con el cliente a diario. Se permite el acceso web al cliente para que puede probar el sitio web creado.
7.3 Diseño y Desarrollo		
7.3.1 Planificación del diseño y desarrollo	1	Como se mencionó antes, planificación es desarrollada casi al momento de la iteración. El diseño es planeado en un alto nivel de abstracción. Esto se debe a que XP propone refactorizar el código para hacer un buen diseño.

7.3.2 Entradas para el diseño y desarrollo	1	Requerimientos funcionales fueron especificados en las historias de usuario, pero requerimientos no funcionales como confiabilidad, portabilidad, usabilidad, fueron desarrollados solamente desde las dos últimas iteraciones. XO no maneja requerimientos no funcionales
7.3.2 Salidas del diseño y desarrollo	1	Una detallada especificación de diseño será provista usando una herramienta de ingeniería reversa. Especificaciones de la arquitectura del sistema fueron desarrolladas en un alto nivel de abstracción. Las pruebas no fueron escritas en el inicio del desarrollo, en las dos últimas iteraciones antes del final del proyecto Ellos escribieron las pruebas para algunas historias de usuarios. Código ejecutable y código fuente será entregado al cliente.
7.3.4 Revisión al diseño y desarrollo	1	La mayoría del código y el diseño de la arquitectura del sistema fue revisado por el Grupo de Pruebas. El diseño detallado no fue revisado, debido a que ha sido un proceso continuo de diseño y mejoramiento de código.
7.3.5 Verificación del diseño y desarrollo	1	El proceso de verificación fue desarrollado por el Grupo de Pruebas. La trazabilidad de los requerimientos a codificar no fue definida. La mayoría de las pruebas unitarias no fueron definidas no así las pruebas de aceptación e integración.
7.3.6 Validación del diseño y desarrollo	3	La validación fue ejecutada en reuniones con el cliente, donde Ellos mostraron un prototipo del producto objeto del proyecto a través del tiempo
7.3.7 Control de cambios en diseño y desarrollo	1	Control de diseño no fue ejecutado debido a que el equipo de desarrollo entregaría un diseño cuando el proyecto este completado. El diseño solo está definido en el nivel más alto (sin detalle). Se maneja los cambios en el código usando registrando en una bitácora diaria lo que se ha cambiado en cada programa.
7.4 Compras		

7.4.1 Proceso de compras	-	No aplicable
7.4.2 Información de Compras	-	No aplicable
7.4.3 Verificación del producto comprado	-	No aplicable
7.5 Producción y provisión de servicios		
7.5.1 Control de producción y provisión de servicios	-	No aplicable
7.5.2 Validación de procesos para producción y provisión de servicios	-	No aplicable
7.5.3 Identificación y trazabilidad	1	Como fue detallado antes, la trazabilidad no fue desarrollada desde los requerimientos al código. XP no explica cerca de eso, pero por ejemplo, es posible trazar un requerimiento como una historia de usuario y quien fue el propietario
7.5.4 Propiedad del Cliente	3	El código fuente del sistema es propiedad intelectual del cliente
7.5.5 Preservación del Producto	-	No aplicable
7.6 Control de monitoreo y dispositivos de medición	-	No aplicable
8 Medición, análisis y mejoramiento		
8.1 General	2	Nuevamente, la medición de requerimientos no funcionales no fue desarrollada hasta la parte final del proyecto, debido a la ausencia detallada de especificaciones de requerimientos en XP. Los requerimientos funcionales fueron probados sin ningún problema usando la información en las historias de usuario. Problemas con las medidas, análisis y monitoreo del producto están detallados en las cláusulas siguientes.

8.2 Monitoreo y Medición		
8.2.1 Satisfacción del Cliente	2	La satisfacción del cliente fue desarrollada a través de reuniones con el cliente al final de cada iteración donde la retroalimentación fue obtenida directamente cara a cara con el cliente, y se deja constancia de sus observaciones en la minuta posterior a cada reunión, pero no es medible en ninguna forma.
8.2.2 Auditoría Interna	-	No aplicable, hubo únicamente gente del proyecto trabajando en aquello, no fue asignado una auditoria interna, solo el rol de revisores que tuvo el grupo de pruebas.
8.2.3 Monitoreo y medición del proceso	1	Mediciones del proceso fueron desarrollados usando los puntos de tarea y velocidad de cada desarrollador, ellos no tienen otras medidas para monitorear el proceso
8.2.4 Monitoreo y medición del producto	2	Como fue detallado antes, ellos no monitorearon requerimientos no funcionales, los funcionales si lo hicieron. En la última iteración Ellos ejecutaron la medición de requerimientos no funcionales.
8.3 Control de producto no conforme	1	Ellos no hicieron ningún control sobre la parte del producto que no cumplía con los requerimientos del cliente. Pero en la parte final del proyecto si hubieron informes de pruebas satisfactorias y no satisfactorias
8.4 Análisis de Datos	1	Los mejoramientos fueron hechos sin el análisis de datos, solamente debido a la investigación de alguien sobre algún tema especifico
8.5 Mejoramiento		
8.5.1 Mejoramiento Continuo	2	Mejoramiento continuo fue mantenido en las reuniones de la mañana, en las reuniones con el cliente al final de cada iteración. Los resultados de las reuniones no fueron guardadas, problemas con acciones preventivas y correctivas se describen en las cláusulas 8.5.2 y 8.5.3

8.5.2 Acciones correctivas	1	El equipo ejecuta acciones correctivas con los resultados que salen luego de las reuniones cada lunes, miércoles y viernes y luego de las reuniones con el cliente. No hay revisiones de acciones correctivas que han sido tomadas. Lo resultados no se guardan, únicamente se responde ante los errores que se presentan. No se tiene un seguimiento de ellos.
8.5.3 Acciones preventivas	1	Se toman acciones preventivas con la información obtenida en las reuniones breves, reuniones diarias, y reuniones con el cliente. No hay revisiones de las acciones preventivas que se han tomado. Los resultados no fueron almacenados

3.8 CUMPLIMIENTO DE LOS OBJETIVOS DE CALIDAD.

Conforme a la calificación realizada. en cada uno de los puntos de evaluación de la guia TickIT, podemos determinar el nivel de cumplimiento de los objetivos de calidad. El cumplimiento de los mismos no se basa en si fue o no documentado o registrado según la norma ISO, sino más bien en las practicas y procesos que se siguieron para lograr los objetivos.

La ponderación que se uso en la calificación fue la siguiente: 0 cuando no se cumplió, 1 cuando el cumplimiento fue a medias y 2 cuando el cumplimiento fue total.

OBJETIVOS DE CALIDAD	CALIFICACION	SITUACION EN EL PROYECTO
Satisfacción al cliente	1	Una buen parámetro para medir la satisfacción del cliente hubiese sido las pruebas de aceptación, pero estas únicamente fueron desarrolladas en las parte final del proyecto
Procesos ágiles de desarrollo	2	Todos los procesos: de diseño, de desarrollo, de pruebas, de implementación y mejoramiento fueron concebidos pensando en la agilidad y cumplimiento de los tiempos del cronograma
Sinergia y empatia entre el cliente y los desarrolladores	2	Desde el primer momento hubo empatia y sinergia entre los desarrolladores y el responsable del cliente
Equipo de trabajo motivado	2	El equipo de trabajo se motivo al estar involucrado 100% con el cliente y sus necesidades. Vieron resultados positivos de las acciones tomadas dia a dia y eso fue completamente motivante para el grupo
Métodos efectivos de comunicación	2	La comunicación en el equipo fue muy sencilla y directa. Generalmente las cosas más importantes se comunicaban en persona y se registraban en una minuta luego de cada reunión. El email era usado generalmente para la comunicación con los sponsors.
Estado de cooperación permanente entre sponsors, desarrolladores y usuarios	1	Los usuarios finales tuvieron un nivel medio de cooperación ya que su disponibilidad no era al 100%. De los sponsors casi nunca se necesito colaboración durante el desarrollo.
Atención a la excelencia técnica	2	El mejoramiento continuo fue una política de todo el equipo de desarrollo

OBJETIVOS DE CALIDAD	CALIFICACION	SITUACION EN EL PROYECTO
Simplicidad en el trabajo	2	El trabajo en codificación siempre tendió a lo más simple y exacto con los requerimientos del sistema
Autoorganización en el equipo de trabajo	1	La autoorganización no fue uno de los puntos más altos en el equipo de desarrollo. A partir de la mitad del proyecto se logro organización basado en la constante práctica de actividades XP.

3.9 REVISION DEL PROCESO.

El desarrollo del proyecto siguió lineamientos definidos en este documento – tesis, sin embargo dentro del proceso fue evidente la contraposición en muchos puntos de los requerimientos de calidad de la norma ISO y las practicas XP que a continuación pasamos a detallar.

3.9.1 FASE DE EXPLORACION

3.9.1.1 Actividad 1: ALCANCE DEL PROYECTO.

Luego del levantamiento de información inicial relacionada con el desarrollo del sistema, la empresa desarrolladora del software entrego un documento alcance del proyecto que considero los siguientes puntos dentro del mismo.

1. Descripción del sistema
2. Objetivos del desarrollo del sistema.
3. Descripción de la funcionalidad del sistema. Detalle de lo que forma parte del proyecto y aquello que no se considera como parte del mismo.
4. Recurso humando necesario para el proyecto y el presupuesto mensual por cada uno de ellos
5. Tiempo de Entrega final del proyecto
6. Costo total del proyecto

7. Términos de las condiciones de pago

Este documento fue discutido entre los directivos de la empresa proveedora del desarrollo y los sponsors del proyecto y no se realizó modificación alguna al mismo. El alcance fue aceptado tal cual se presentó.

El documento alcance no considero los siguientes puntos:

1. Definición de los entregables del proyecto
2. Criterios de aceptación de los entregables

3.9.1.2 Actividad 2: PLAN DE EJECUCION DEL PROYECTO.

El equipo de desarrollo formado por el Líder de proyecto, el Líder de programación, el Líder de pruebas y el responsable del cliente tuvieron la primera reunión de trabajo donde definieron el plan de ejecución del proyecto que tuvo como contenido lo siguiente:

1. Tecnología a utilizarse en el desarrollo del producto final
2. Requerimientos de hardware para el equipo de trabajo
3. Perfil requerido para los programadores
4. Nivel de capacitación en metodología XP
5. Identificación de los entregables del proyecto
6. Estimación del número de iteraciones por cada entregable

En una segunda reunión con los programadores del equipo de desarrollo y el equipo de pruebas se determinó lo siguiente:

1. Cronograma general del proyecto

3.9.1.3 Sumario - Fase de Exploración.

En la fase de exploración del proyecto hubo las siguientes observaciones al proceso propuesto:

1. Fue difícil estimar los entregables del proyecto al momento de elaborar el alcance del proyecto, ya que el mismo fue realizado únicamente por la empresa desarrolladora del software y no en conjunto con el cliente.

2. Al no poder definirse los entregables del proyecto no se pudo llegar al detalle de las tareas a realizarse en cada uno de ellos y las actividades en conjunto con el cliente.

3. El sponsor no tuvo claro como se iba a desarrollar el proyecto, simplemente tuvo la información general de la funcionalidad del software, el tiempo de entrega y el costo total del proyecto.

4. El requerimiento de un responsable del cliente que trabaje de forma permanente con el equipo de desarrollo solo fue comunicado de forma personal a los sponsors y se confirmo vía email.

3.9.2 FASE DE PLANIFICACION.

3.9.2.1 Actividad 1: GESTION DE REQUERIMIENTOS.

Para la gestión de requerimientos se partió del alcance del proyecto y luego se hicieron más específicos los requerimientos funcionales a través de las historias de usuario.

Cada historia de usuario contenía lo siguiente:

1. Un número secuencial que la identificaba
2. Nombre descriptivo de la historia

3. Prioridad dentro de los objetivos principales del negocio
4. Riesgo en el desarrollo
5. Puntos estimados de esfuerzo
6. Número de iteración asignada.
7. Programador responsable
8. Descripción detallada de la historia
9. Observaciones
10. Requerimientos funcionales

Al principio las historia de usuario abarcaban mucho trabaja para el entregable. Esto no era muy bien entendido por el cliente y el equipo de desarrollo. Las iteraciones de trabajo eran muy complejas por lo que fue necesario recomponer las primeras historias para reducirlas al mínimo posible de tal forma que se obtenga un entregable útil en el menor tiempo posible. El cliente fue aprendiendo a definir las historias de usuario de una forma más clara para los programadores.

Los requerimientos no funcionales no fueron considerados en el formato escrito para detallar la historia de usuario. Estos requerimientos solo fueron considerados en la parte final del proyecto y fueron generales para todo el sistema.

3.9.2.2 Actividad 2: EJECUCION, DESARROLLO Y PRUEBAS.

El cliente siempre fue quien elegía el conjunto de historias de mayor valor a ser implementadas en la iteración planeada.

Las iteraciones al principio se dieron con más frecuencia de lo programado, es decir las cosas no estaban muy claras y era necesario reuniones frecuentes entre los miembros del equipo responsables de ejecutar un entregable. Esto también se origino debido a la generación de historias de usuario no adaptables al esquema XP.

Fue importante para el equipo entender lo esencial de las iteraciones para poder crear procesos adaptables que puedan tratar con los cambios en los requerimientos iniciales. El desarrollo iterativo da un fundamento firme en cada iteración que puede usarse para las planificaciones posteriores. La tendencia siempre fue crear iteraciones tan cortas como se puedan manejar ya que esto proporciona retroalimentación más frecuente, asi le daba una idea más clara al equipo de donde se encontraba.

La codificación desde el principio fue desarrollado en parejas y la primera labor que tenían los programadores era dividir las historias de usuario en tareas más pequeñas. Luego cada pareja de programadores elegían las tareas que deseaban implementar, las analizaban en mayor detalle y realizaban una estimación del tiempo de desarrollo necesario.

Finalmente el cliente ordenaba en función de sus necesidades las Historias estimadas, dejando para iteraciones posteriores aquellas que sobrepasen la capacidad productiva de la iteración.
Un problema que surgió fue el de administrar las dependencias entre las historias de usuario. XP considera que las historias son entidades independientes, por lo tanto no había un modelo a seguir para esta situación. Se consideraron iteraciones especiales para dar tratamientos a situaciones de dependencia.

Las pruebas unitarias se hicieron al principio del desarrollo del proyecto pero conforme fue avanzando el proyecto las pruebas unitarias también fueron evolucionando volviéndose cada vez más complejas y por lo tanto su mantenimiento exigía cada vez más tiempo y esfuerzo. Se empezó a dejar de lado las pruebas unitarias a mitad del proyecto.

3.9.2.3 Actividad 3: MONITOREAR LA VELOCIDAD DEL PROYECTO.

El monitoreo de la velocidad del proyecto fue bastante sencillo ya que en base a los puntos estimados por cada historia usuario se puedo determinar si las semanas estimadas en un conjunto de historias de usuario se cumplieron o no.

En general su cumplimiento fue de aproximadamente el 85%, siendo necesario cumplir horas extras en el proyecto para que el desfase no fuese mayor.

3.9.2.3 Sumario - Fase de Planificación.

En la fase de planificación del proyecto hubo las siguientes observaciones al proceso propuesto:

1. El cliente tuvo dificultades al escribir las primeras historias de usuario, luego el aprendizaje hizo subir el nivel de elaboración de cada historia. Es real que a muchos clientes les cuesta detallar sus requerimientos con un alto nivel de claridad y detalle.
2. La programación en parejas fue una práctica que tuvo pocas dificultades al momento de ejecutarse. Las programadores tenían un nivel técnico muy parecido y su nivel de conocimiento con XP solo había sido a través de charlas breves de capacitación. En el proceso fueron desarrollando habilidades de desarrollo en Extreme Programming. Fueron un aporte fundamental para el cliente al momento de escribir las historias de usuario y estimar el esfuerzo necesario de cada una.

3. El punto más débil para el equipo en esta fase fueron las pruebas: las pruebas unitarias se hicieron al inicio y luego conforme el desarrollo fue avanzando las pruebas se descontinuaron al volverse más complejo su mantenimiento.

4. XP no contempla el tratamiento para dependencia entre historias de usuario, por lo tanto hay que considerar iteraciones adicionales dentro del proceso exclusivamente para darle tratamiento a situaciones de este tipo.

3.9.3 FASE DE PRODUCCION.

3.9.3.1 Actividad 1: IMPLEMENTACION DEL SOFTWARE CREADO.

La fase de producción consistió básicamente en las pruebas finales a ser realizadas por el cliente y el grupo de pruebas para que finalmente en entregable sea liberado. Nos encontramos con las siguientes observaciones dentro de las actividades correspondientes a esta fase:

1. Escribir pruebas de aceptación fue algo totalmente nuevo para el cliente y a pesar de tener el apoyo del grupo de pruebas estas solo se hicieron luego que el proyecto había transcurrido en su mayoría. Al inicio las pruebas de aceptación consistían simplemente en simular el proceso dentro del contexto de la historia de usuario.

2. Cada pareja de programadores manejaba su propio código. El código en común fue escrito directamente en la base de datos, para lo cual había un programador de base de datos que proveía de la funcionalidad al sistema requerida por los programadores.

3. Cuando el desarrollo del entregable involucraba hacer una modificación o mejora a un entregable anterior, entonces este requerimiento pasaba por refactoring (mejora del código) dentro de la fase de mantenimiento.

4. Las pruebas de regresión no fueron bien entendidas y se volvió compleja su aplicación. Al inicio solo cuando el cliente consideraba que podía haber una afectación a entregables anteriores se verificaba la consistencia de la información, esto originó algunos problemas con los

entregables en producción y muchos de ellos tuvieron que regresar a mantenimiento. Luego cuando se empezaron a escribir las pruebas de aceptación se utilizaron las pruebas de procesos anteriores como base para determinar las pruebas de regresión.

3.9.3.2 Sumario - Fase de Producción.

Como principal observación el equipo no estaba lo suficientemente preparado para ejecutar las pruebas de aceptación y regresión, es indudable que al ser su primera experiencia con este tipo de actividades fue, al principio, un poco complicado entenderlas y ejecutarlas. Luego ya hubo un mayor aprendizaje de las mismas y al final del proyecto se realizaron satisfactoriamente. En general el proceso de producción se llevo adelante sin mayores observaciones

3.9.4 FASE DE MATENIMIENTO.

3.9.4.1 Actividad 1: PLAN DE MANTENIMIENTO DEL SOFTWARE CREADO.

Dentro de la fase de mantenimiento del software se considera dos actividades principales y generales que son: el manejo de errores y el refactoring del código. Si bien es cierto lo ideal hubiese sido contar con un plan que describa actividades, papeles y responsabilidades, sin embargo si se contó con una guía para gestionar los errores encontrados y las mejoras al código del sistema basada en los lineamientos de XP.

1. El usuario reportaba al grupo de pruebas el error detectado
2. El tester describía el error y lo reportaba al programador encargado
3. Dependiendo de la prioridad del error se programaba su corrección ya sea en una iteración posterior (si es de prioridad baja), o en la iteración actual (si tiene alta prioridad).

4. El programador corregía el código y lo pasaba nuevamente al grupo de pruebas.
5. El software era liberado una vez que pasaba las pruebas de aceptación hechas por el tester y el usuario.

Conforme el sistema iba evolucionando los programadores planificaban el refactoring para el código entregado anteriormente. Sin embargo solo se consideraba hacer refactoring bajo las siguientes condiciones.

1. Cuando se creaban nuevas funciones al sistema que ameritaba una optimización al código escrito
2. Cuando el código ya tenia un tiempo prudencial en ejecución y se habían detectado puntos de mejora importantes en el.

3.9.4.2 Sumario - Fase de Mantenimiento

La fase de mantenimiento solo consideraba dos actividades: el manejo de errores y el refactoring del código. Si bien es cierto no hubo mayores inconvenientes en la gestión de estas actividades la debilidad principal constituyo la poca documentación de el trabajo realizado en cada una de ellas. XP no da ninguna directiva para documentar el manejo de errores y el refactoring, sin embargo en el proceso se considero el reporte de producto no satisfactorio, lo que se consideraba como reporte de errores encontrados, pero no había un detalle del trabajo realizado sino simplemente el resultado final luego de las correcciones. EN el caso de refactoring era un trabajo que solo se planificaba dentro del grupo de programadores sin dejar registro alguno.

CAPÍTULO 4

ANALISIS DE RESULTADOS.

4.1 INTRODUCCIÓN.

La industria de software carece de reconocimiento al mejoramiento de procesos, una de las razones más comunes para esto es que se pone mucho énfasis en la programación debido a las mejoras que se están haciendo en esta área y por la muy buena experiencia de los programadores en la industria, pero muchas organizaciones no ven que tienen problemas con el área de planificación del proyecto, administración y análisis de riesgo y administración de proyectos.

Este capítulo presenta las comparaciones desde el análisis realizado en los capítulos 2 y 3 en un caso de estudio, desde el punto de vista de XP y desde el punto de vista ISO, así como las fortalezas y debilidades encontradas en el proyecto. Tomando en consideración los resultados aquí obtenidos estaremos en capacidad de reforzar las áreas más débiles para certificar el modelo propuesto conforme al estándar ISO.

4.2 COMPARANDO XP E ISO EN EL CASO DE ESTUDIO.

En esta sección se hará un análisis comparativo entre las prácticas y actividades XP definidas en la metodología contra aquellas encontradas en el proyecto del caso de estudio. Así mismo se hace una comparación del cumplimiento de los requerimientos de calidad ISO confrontándolas con su cumplimiento dentro del desarrollo del proyecto.

Al final del proyecto se hizo una encuesta con cada uno de los miembros del equipo de desarrollo, incluido el cliente, ya que al ser la primera vez que tenían una experiencia aplicando la metodología XP en un proyecto de desarrollo de software cuyos resultados se muestran en el gráfico 4.2.

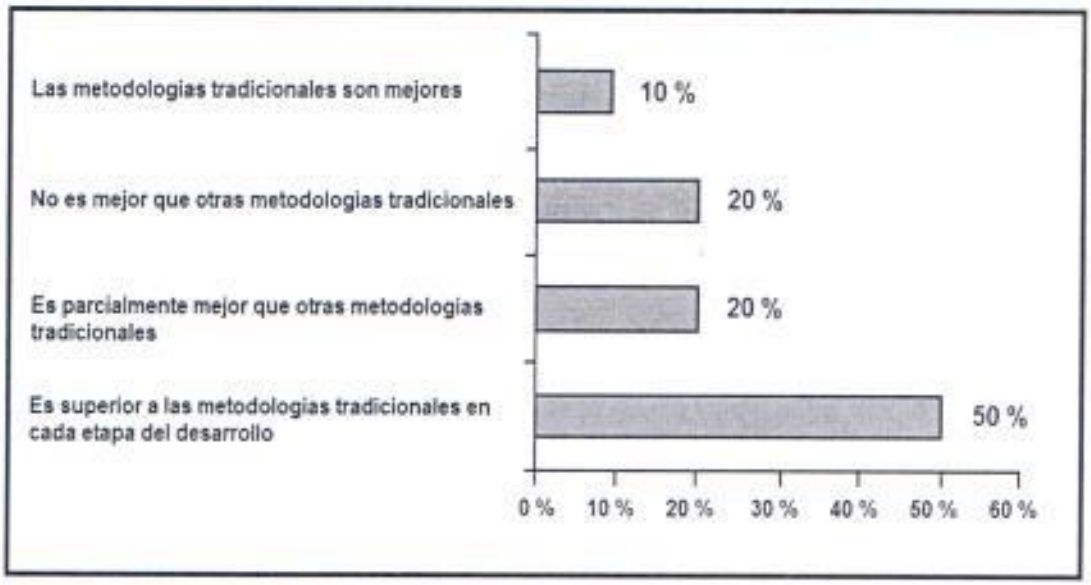

Gráfico 4.2

4.2.1 COMPARANDO LAS PRÁCTICAS XP.

En la siguiente tabla 4.0 se podrá observar el número de prácticas XP agrupadas por el nivel de cumplimiento dentro del proyecto de caso de estudio.

Grado de cumplimiento	Total de Prácticas XP
1	3
2	3
3	6
	12

TABLA 4.0

Uno puede ver en la tabla anterior que el proyecto de caso de estudio tiene una ejecución aceptable en cuanto a trabajar con las prácticas XP.

Sin embargo, al comparar con los resultados obtenidos detallados en el Capitulo 3 el proyecto falla en los siguientes puntos importantes en la aplicación de la metodología XP:

- **Cliente en sitio.** Al momento de calificar a todo el equipo debido a que no tienen el cliente en sitio para un comunicación directa y efectiva, y para ejecutar las pruebas de aceptación, en su lugar tuvieron una persona dentro del equipo de trabajo que hacia de enlace entre el equipo y el cliente. Otra de las fallas importantes fue que las pruebas de aceptación se hicieron manualmente y no automáticamente como lo dice XP.

- **Desarrollo conducido por pruebas.** La mayor diferencia con respecto a las prácticas XP se da en las pruebas de código, ya que estas solo se hicieron luego de que el código fue creado, sin embargo las prácticas de XP en el juego de la planeación la situación se da a la inversa.

- **Propiedad colectiva de código.** Otra diferencia es acerca de la programación en pareja, en el proyecto solo una parte del equipo desarrollo esta práctica. Hay una falencia en cuanto a programar en pareja que se da debido a la diferencia de nivel y habilidades técnicas entre los programadores lo que hacia retrasar el desarrollo de las aplicaciones y llevó al equipo a ser condescendiente con el cumplimiento de la práctica de la propiedad colectiva del código.

- **Metáfora del sistema.** Finalmente, la última observación en las prácticas XP es que no se comprendió muy bien la práctica de la metáfora del sistema.

La prácticas arriba mencionadas fue donde el equipo tuvo mayores dificultades tanto en la comprensión con en su aplicación. Podemos observar también que estás prácticas son aquellas que resultan completamente nuevas para un desarrollador que no ha tenido experiencia en metodologías ágiles sino en los métodos tradicionales de desarrollo.

Las otras prácticas fueron entendidas y aplicadas en un nivel bastante aceptable

4.2.2 COMPARANDO LAS ACTIVIDADES XP CON LOS RESULTADOS DEL CASO DE ESTUDIO.

En la siguiente tabla 4.1 tenemos el número de actividades XP agrupadas por el nivel de cumplimiento que obtuvieron en el proyecto de caso de estudio.

Grado de cumplimiento	Total de Actividades XP
1	1
2	1
3	2
	4

TABLA 4.1

- **Codificación.** La actividad de codificar conforme al modelo XP no se cumplió en su totalidad debido a que en el proyecto del caso de estudio no se cumplió con la propiedad colectiva del código a través de la programación en pareja. Si bien es cierto se pudo formar una pareja de programadores que trabajaron juntos con el resto de programadores no fue posible hacerlo, sin embargo y a pesar de esto dentro del equipo de programadores hubo una muy buena comunicación que permitió al equipo tener un conocimiento bastante avanzado de los componentes de cada aplicativo.

- **Pruebas.** La actividad de prueba del código no es cumplida en el proyecto ya que las pruebas se escribieron después de que se hizo la codificación y no antes como lo dice XP.

Las otras dos actividades: reuniones continuas con el cliente y diseño fueron realizadas cumpliendo con lo que dice XP.

4.2.3 COMPARANDO LOS REQUERIMIENTOS ISO.

En la siguiente tabla 4.2 nuestra los resultados de comparar XP con las cláusulas ISO en el proyecto de caso de estudio.

Grado de cumplimiento	Total de cláusulas ISO
No aplica	9
0	1
1	17
2	7
3	17
	51

TABLA 4.2

- **Sistema de gestión de calidad.** Los requisitos generales y de documentación necesarios para cumplir con esta cláusula fueron cumplidos parcialmente: hubo la documentación de procedimientos, políticas y roles del proyecto, el alcance; y cuando empezó el proyecto se logró documentar los requerimientos funcionales del sistema en cada reunión de planeación conforme XP. No se documentaron los requerimientos no funcionales, ni existió un documento formal del diseño

- **Responsabilidad de la dirección.** La dirección estuvo comprometida a un gran nivel con el proyecto.

- **Administración de recursos.** El equipo humano y los recursos del proyecto fueron manejados con bastante eficiencia por parte de la Administración del proyecto. No fue necesario contratar más recursos que los que se presupuesto al inicio del proyecto, no fue necesario el uso de sobretiempo en el proyecto y al ambiente de trabajo fue el adecuado.

- **Realización del producto.** En este punto es donde se dan las mayores diferencias con el modelo XP. Los procesos relacionados con el cliente se cumplieron bastante bien, más sin embargo los procesos de diseño y desarrollo fue muy difícil poder ajustarlos a los requerimientos del estándar sin modificar sustancialmente los principios XP y el cronograma establecido del proyecto. Se le dio más prioridad a esto último que al estándar ISO en sí.

- **Medición, análisis y mejoramiento.** En el proyecto no se definieron criterios para desarrollar la suficiente cantidad de métricas necesarias para evaluar el avance del proyecto, aseguramiento de la calidad y satisfacción del cliente, sin embargo al ser un proyecto pequeño estos puntos pudieron ser manejados con relativa facilidad. Si bien es cierto en XP hay principios y actividades enfocadas a lograr la satisfacción del cliente y validar la calidad del producto dentro del proceso no se cumplió en su totalidad con los requerimientos ISO. El proceso de mejoramiento continuo si bien se cumplió, no se documento adecuadamente conforme lo requiere ISO.

En su mayoría las cláusulas que no aplicaron fueron aquellas que describen en la sección de compras y provisión de servicios. El puntaje más alto fue obtenido en aquellas cláusulas que tienen que ver con el manejo de la relación con el cliente, comunicación entre el equipo del proyecto y ambiente de trabajo, que son los puntos donde XP tiene una mayor relevancia. Podemos observar también que los puntos más débiles y donde se necesita mayor nivel compatibilidad entre los dos estándares son aquellos que tienen que ver con el monitoreo, control y aseguramiento de la calidad, si bien es cierto XP trabaja mucho en el aseguramiento del software a través de las pruebas escritas antes de la codificación, al equipo del proyecto se le hizo muy difícil definir métricas para medir la satisfacción del cliente conforme a los requerimientos funcionales y no funcionales, tal cual lo pide el estándar ISO.

4.3 FORTALEZAS Y DEBILIDADES PARA CUMPLIR CON EL ESTÁNDAR ISO.

En esta sección son establecidas las fortalezas y debilidades encontradas en el proyecto del caso de estudio en relación al cumplimiento con las cláusulas del estándar ISO.

En el gráfico 4.3 se muestra un gráfico que mide el cumplimiento de cada una de los puntos analizados en cuanto a las exigencias de calidad.

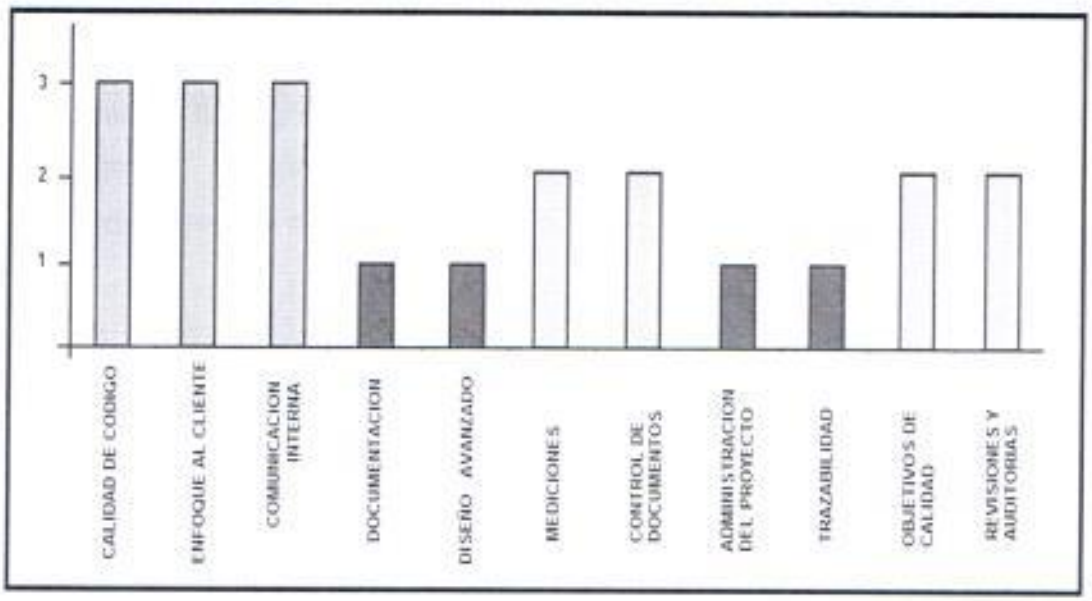

GRÁFICO 4.3

4.3.1 FORTALEZAS.

- **Calidad del código.** La calidad de código fue revisado en las horas de observación por el autor de este trabajo de Tesis. El código tuvo en general una buena calidad dentro del proyecto. Hubieron ciertas observaciones con respecto a este punto, como por ejemplo manejar las conexiones con la base de datos desde el código de las páginas web en lugar de usar procedimientos en la base de datos.

- **Enfoque al cliente.** XP está altamente comprometido con el cliente. En el proyecto a la persona que hizo las veces de cliente se le dio todo el tratamiento dentro del proceso de desarrollo que considera XP y también ISO. La retroalimentación de parte del cliente fue dentro de la medida que podía aportar la persona representante del cliente, sin embargo en las esporádicas reuniones con el cliente real se aporto bastante en este tema. La continuas entregas incrementaron los niveles de satisfacción del cliente

- **Comunicación Interna.** Este fue un aporte muy positivo de XP porque todos en el equipo del proyecto trabajaron físicamente dentro de un mismo lugar y se podían comunicar directamente con cada miembro del grupo. El conocimiento es transferido naturalmente entre todos los miembros del equipo del proyecto.

4.3.2 DEBILIDADES.

En esta sección hay una descripción de donde están los problemas con ISO usando XP como metodología de desarrollo usando la información obtenida en la revisión del proyecto de caso de estudio.

- **Documentación.** El problema principal con la documentación es que ISO que casi todo deba ser documentado y con alto nivel de calidad. En su lugar XP dice que el equipo debe documentar solo aquello que es necesario y en la cantidad suficiente para el proyecto, no tiene que ser todo. La documentación requerida por ISO y que fue elaborada en el proyecto fue:
    - Manual de calidad
    - Contrato
    - Plan de pruebas
    - Plan de implementación

- Plan del proyecto
- Procesos de trabajo y rutinas
- Organización
- Estándar de codificación
- Resultados de las pruebas

Todo esto fue provisto por el equipo de desarrollo como un extra, ya que nada de esto forma parte de XP.

La documentación de requerimientos fue provista por las historias de usuario, si bien no es un documento tradicional para plasmar los requerimientos, cumple con lo mínimo necesario requerido por ISO.

El principal problema fue no haber documentado ningún requerimiento no funcional. Esto realmente no es un problema solo del proyecto, sino un problema de XP en si, donde la documentación debe estar solamente en el código, las pruebas y las historias de usuario.

- **Diseño avanzado.** Al inicio del proyecto quería seguir los lineamientos de XP sobre que el equipo de trabajo no debería trabajar con algún diseño avanzado, más allá de usar solo 30 minutos en cada historia de usuario para desarrollar un diseño y arquitectura para cada una de ellas. La interpretación que dio el equipo a esto fue que ellos estaban continuamente mejorando el diseño.

  Conforme el proyecto fue avanzando el equipo se dio cuenta que es necesario crear un diseño más avanzado para el sistema, debido a su tamaño. No tiene que ser muy detallado pero al menos debe ser algo que sirva de base.

- **Mediciones.** El equipo de trabajo hizo pocas mediciones a lo largo del proyecto conforme se mencionó en las secciones previas, pero una de las mediciones que no se hicieron fue lo que sucedió en las pruebas. Por ejemplo, cuantas pruebas pasaron y no pasaron cada día de trabajo, y en que porcentaje. XP no explica acerca de las mediciones en el

proyecto, sin embargo dentro del proyecto se definieron algunas métricas para medir los avances del proyecto, más no para cumplir con requerimientos ISO.

De acuerdo a recomendaciones de auditoría, las mediciones de prueba al menos deberían de hacer en la integración a nivel del sistema.

Desde el punto de vista del autor de este trabajo, hay algunos problemas en la práctica XP del desarrollo conducido por pruebas en lo que se refiere a las pruebas de integración y las pruebas del sistema, porque en el proyecto se hizo mención en su mayoría solamente a las pruebas unitarias y para el equipo fue realmente difícil tener algunas prácticas de pruebas de integración.

- **Control de documentos.** XP no es específico acerca del control de documentos. El Líder de proyecto está a cargo del control de documentos en el proyecto.

- **Administración de Proyecto.** Existen diferentes formas a seguir para la administración del proyecto, el equipo de trabajo del caso de estudio usó la forma más tradicional donde la gente seguía las ordenes de los líderes de grupos y del líder de proyecto. Ellos tenían reuniones diarias de trabajo, los días lunes para mejoramiento continuo del proyecto, sin embargo se hizo notorio la falta de experiencia de algunos líderes para manejar tareas administrativas. Un poco antes de la mitad del proyecto el estilo cambio a algo más cercano al que usan las metodologías ágiles para el desarrollo de software con el juego de la planeación en cada iteración, reuniones ágiles luego de cada iteración y reuniones diarias para el control del proyecto

- **Trazabilidad.** Trazabilidad desde los requerimientos del cliente hasta el código no está definida en XP. Se supone que el equipo debería de auto-organizarse para responder a los requerimientos de trazabilidad.

Como normalmente pasa en las primeras experiencias el proyecto tuvo problemas con la trazabilidad y control de esto a lo largo del desarrollo.

- **Objetivos de calidad.** En el proyecto se habían documentado los objetivos de calidad como desempeño, estabilidad, etc., esto estaba relacionado a requerimientos no funcionales, sin embargo esto no se probó sino hasta la parte final del proyecto debido al tiempo.

  En XP solo los requerimientos funcionales o técnicos son definidos usando las historias de usuario, pero en ellas no se detalla como tratar con requerimientos no-funcionales.

- **Revisiones y auditorias.** Las revisiones y las auditorías internas son hechas debido a los requisitos ISO para asegurar la calidad del producto, pero ellas no están especificadas en XP. Aunque las revisiones internas en XP puedan ser vistas como la práctica de programar en pareja donde una pareja está chequeando constantemente lo que la otra está escribiendo. En el proyecto del caso de estudio el grupo de pruebas estuvo a cargo de las revisiones de calidad y pruebas del producto.

CONCLUSIONES.

En el análisis comparativo que se realizó con los resultados obtenidos se enfoco en la identificación de las prácticas claves relacionadas en los modelos ISO y Extreme programming, también las fortalezas y debilidades compatibles de ambos modelos fueron identificadas, y las prácticas extreme programming mapeadas con las cláusulas de ISO luego de una cuidadosa explotación de los atributos y características de estos modelos. Estos resultados obtenidos nos permiten responder a las siguientes preguntas del trabajo de investigación:

- *¿Los principios y valores de las tecnologías ágiles de desarrollo son complementarios con los valores y principio de ISO 9001:2008, o existe conflictos entre ellos?*

  Los valores en el desarrollo ágil entra en conflicto en algunas partes con un sistema de calidad basado en ISO, por ejemplo, en el desarrollo ágil las relaciones entre personas tienen alta valoración, mientras que en el Sistema de Calidad los procesos tienen mayor importancia, por esto si en un Sistema de Calidad los procesos tienen mayor relevancia las personas pueden ser fácilmente sustituidas, lo que en el caso de las tecnologías ágiles no se cumple ya que son las personas quienes dan valor al proyecto mejorando la calidad de código y el proceso con su experiencia.

  Si el director del proyecto trata de sustituir en de los miembros de equipo en una instancia del proyecto, El tiene que enseñarle al nuevo miembro acerca de las prácticas XP que son usadas en el proyecto, por ejemplo: desarrollo conducido por pruebas, o modelos de diseño, a fin de hacer mejoras de diseño, por lo tanto hay mucho tiempo involucrado en ello, en cambio en un Sistema de Calidad hay algunos documentos que la persona debería estudiar en un margen de tiempo, no así en XP, y la persona en este caso trabaja sólo en tareas específicas, mientras que en XP debería trabajar en múltiples tareas como programador, probador,

administrador de base de datos, etc. En XP el desarrollador debe tener un amplio conocimiento.

Otra discrepancia es que un Sistema de Calidad confía principalmente en la planificación y documentación, los cuales son descritos en el estándar ISO, ellos no hacen del funcionamiento del software como la esencia del desarrollo de software. De otro punto de vista el enfoque en el cliente es seguido por ambos modelos, ellos concuerdan en esa parte.

Cuando se habla comparativamente de los principios en desarrollo ágil y principios en un Sistema de Calidad con el estándar ISO, va a ocurrir lo mismo cuando queremos comparar los valores del desarrollo ágil e ISO, en algunas partes ellos concuerdan y en otras partes ellos entran en conflicto, sin embargo en otras partes ellos ni concuerdan ni discrepan entre si. Por ejemplo, ellos están de acuerdo en la mejora continua de los procesos, pero discrepan que los mejores entregables llegan como producto de equipos auto-organizados, para los proyectos basados en el estándar ISO es muy valioso el administrador de proyecto como la fuente principal para la calidad en el producto y no el equipo en si mismo. Ellos no están de acuerdo o discrepan en considerar el funcionamiento del software como una medida primaria del progreso en el desarrollo de software.

- *¿Cuales son las fortalezas y debilidades de cada modelo?*

La principal fortaleza del estándar ISO es que basado en este estándar la organización puede desarrollar un Sistema de Calidad para mejoras continuas de procesos, que ofrece a la organización la oportunidad de usar eficientemente sus recursos para incrementar la calidad de los productos de software y los procesos involucrados. Transforma a los procesos haciéndolos más eficientes y así la organización puede crear mejores productos y ganar cuotas de mercado en su negocio.

Las debilidades encontradas son que los procesos llegan a involucrar demasiada documentación y esto no permite la suficiente flexibilidad a la organización para tratar con futuros cambios en el ambiente del mercado y la economía actual. Otra debilidad es el alto costo que la organización debería pagar para entrenar a la gente implicada en el proceso, el costo por certificaciones que debe ser renovado cada dos o tres años, una forma para evaluar esto dentro de la organización es la mejora continua de los procesos. Además, otra debilidad encontrada es que la alta dirección no esta muy comprometida con los procesos y luego estos no alcanzaran la madurez y eficiencia necesaria para encarar cambios. Hay una necesidad del compromiso de la alta dirección, para que el mejoramiento continuo de procesos sea institucionalizado en toda la organización, ya que de esta forma se establecen diferencias con los competidores.

Desde el punto de vista de XP sus fuerzas son que la gente se divierte más trabajando con XP (programación en pareja (Cockburn 2001)), pero cuando fue observado en el proyecto del caso de estudio, algunos de ellos se sintieron muy incómodos con XP, la razón para esto fue que para todos fue el primer proyecto usando la metodología XP, y como todo se necesita una mayor experiencia en XP para lograr destrezas. Otra fortaleza de XP en proyectos de software es que el equipo esta continuamente liberando entregables del producto de software en cada iteración, obteniendo con ello una constante retroalimentación del cliente. Uno punto más a favor es que la calidad del código con XP es alta debido al proceso continuo de mejoras en el diseño.

Las debilidades en XP vienen dadas por la carencia de un diseño avanzado, en XP la codificación tiene más importancia. La carencia de trazabilidad, es otro tema que queda a discreción del equipo que usa XP el implementarlo o no. Para la trazabilidad del diseño los cambios requeridos se usan las iteraciones continuas donde se le muestra al cliente la evolución del código.

El equipo necesita una alta disciplina y experiencia para trabajar con todas las práctica de XP, caso contrario XP no trabajara como debe ser. Otras cuestiones importantes son que como poner en práctica las pruebas de integración y pruebas de sistema, en las varias fuentes escritas que tratan sobre la metodología no se explica sobre esto y en el equipo del proyecto se careció de esto. Las métricas, la gestión de riesgos, y la dirección de proyecto son las otras áreas donde XP muestra carencias. Estas últimas áreas pueden influir en el control del proyecto que el equipo de trabajo debe de tener. Otro punto es sobre problemas de escalabilidad que el equipo tiene, porque ellos auto-organizan el cambio y se adaptan para vencer esta escalabilidad creando un teoría hibrida de XP.

- *¿Cuales prácticas, actividades y roles XP fueron desarrollados por el equipo del proyecto?*

  El equipo usó un alto porcentaje de prácticas XP (alrededor del 75 %), el resto no fue completamente cumplido o simplemente no se cumplió. Entre las prácticas que no se cumplieron del todo estuvo el desarrollo conducido por pruebas y el juego de la planificación. Entre las prácticas que no se cumplieron para nada estuvo el cliente en sitio, y el uso de la metáfora del sistema.

  La actividad que no fue cumplida fue la de pruebas ya que el desarrollo conducido por pruebas no se cumplió por completo.

  Acerca de los roles, no hubo la necesidad de incluir todos los roles de un proyecto XP, pero en general la estructura de funciones se apego bastante a lo recomendado por la metodología.

- *¿El equipo de trabajo ejecutó realmente XP?*

  Siendo rigurosos, el equipo de desarrollo no ejecuto XP, Ellos usaron algunas prácticas de XP con algún tipo de metodología hibrida, como se puede observar en los capítulos 3 y 4. Sin embargo de acuerdo a algunos autores es inevitable confeccionar XP de acuerdo a la situación especial de cada proyecto, y el efecto que se logre, aplicando los principios XP, es lo realmente más importante.

  Desde el punto de vista de este trabajo de investigación, un proyecto debería desarrollar las prácticas XP gradualmente a lo largo del proyecto. Para un equipo cuya experiencia en XP es la primera vez deberían tratar de que al termino del proyecto se hayan usado todas las prácticas XP; no así para un equipo con experiencia en XP donde las prácticas deberían de ser utilizadas desde el inicio del proyecto e irlas midiendo continuamente para lograr una mejor eficiencia. Hay que considerar que en XP todo está interrelacionado y se requiere de disciplina y experiencia para el correcto uso de toda la metodología. Para que un proyecto se diga que es un proyecto XP debe estar usando todas las prácticas, valores, principios y actividades de XP.

- *¿Un proyecto XP se ajusta con el estándar ISO 9001:2008?*

  Al momento de analizar el cumplimiento que tuvo el proyecto XP con los requerimientos ISO se obtuvo como resultado que el 35% de ellos tuvieron problemas de cumplimiento, el 14% de ellos tiene advertencia de incumplimiento parcial, el 17% no aplica al proyecto, y solo el 34% pasó una auditoria ISO. Es obvio que el equipo del proyecto no cumplió de los requerimientos de calidad de ISO 9001:2008. Es necesario adaptar y extender la metodología XP para cumplir con estos requerimientos.

- *¿Pueden ambos modelos complementarse?, ¿En cuales áreas?*

  Las áreas complementarias son: la creación de un sistema de calidad, la satisfacción del cliente debido a la constante retroalimentación que se recibe de ellos, comunicación interna dado que el equipo esta localizado en el mismo espacio físico. Las áreas problemáticas son:

  - Gestión de Riesgos
  - Medición
  - Trazabilidad
  - Documentación
  - Requerimientos no funcionales
  - Administración
  - Diseño avanzado
  - Pruebas de sistema e integración

- *¿Cuáles son los beneficios de usar estos modelos juntos?*

  Usando XP en un sistema de calidad ISO 9001:2008 trae agilidad a la organización ya que puede afrontar ambientes de continuos cambios. Esto significa que la empresa de software puede desarrollar productos de calidad y en menos tiempo que antes, obteniendo elevar su cuota de participación en el mercado al hacer esto.

  La ISO 9001:2000 trae a XP un proceso de mejoramiento continuo para comprobar cuales son las partes del proceso y producto pueden cambiar para lograr un mejor desempeño, calidad, y satisfacción de cliente. La compañía puede reducir con XP el costo de documentación, desarrollando sólo la documentación necesaria para certificar en ISO 9001:2008.

RECOMENDACIONES.

XP se ejecuta mucho mejor en equipos de hasta 10 personas, más allá de este número el equipo empezaría a tener problemas de comunicación entre las diferentes personas. En el libro "Extreme Programming Explained" de Kent Beck, el remarca claramente: "El tamaño importa, Es probable que no se pueda ejecutar XP con cien programadores, ni con cincuenta, ni con veinte probablemente. Diez es definitivamente factible", (Beck 2004). Para proyectos más grandes es necesario complementar XP con otras metodologías ágiles como Scrum o DSDM (Método de desarrollos de sistemas dinámicos), por ejemplo, pero eso ya no será XP.

A fin de cumplir con todas las prácticas XP se requiere un grupo muy disciplinado, con algunas personas de mucha experiencia en el desarrollo de software a fin de compartir su conocimiento con el grupo.

Algunos autores recomiendan balancear los proyectos con prácticas de desarrollo ágil y de desarrollo conducido por planificación (tradicional) en el caso de proyectos con mayor número de personas y con requerimientos exigentes de planificación.

Para seguir un proceso de mejoramiento continuo debería ser buena práctica que en las reuniones ágiles después de cada iteración se tomen consideración las actividades retrospectivas del proyecto para realizar una iteración retrospectiva, sin embargo el proceso debería ser confeccionado en aproximadamente 4 horas. Después de que el proyecto es terminado una retrospectiva completa del proyecto debe considerada.

BIBLIOGRAFIA

[Leakey, D. & Restell, P. (2001)], The TickIT Guide; Usando ISO9001:2000 para Sistemas de Gestión de Calidad, Construcción y Mejoramiento Continuo, 5ta Edición, British Standards Institution

[ISO 9001:2008], Norma ISO 9001:2008, Organización Internacional para la Estandarización

[Dean Don, 2003] Dean Leffingwell, Don Widria, Gestionando Requerimientos: Acercamiento a un caso de utilidad, segunda edición. Mayo 2003

[Jerzy, N, J, W&A, 2005] Jerzy R. Nawrocki, Michael Jasinski, Bartosz Walter, and Adam Wojciechowski, Combinando EXtreme Programming con ISO 9000, (Años 2002-2005).

http://www.cs.put.poznan.pl/jnawrocki/publica/eurasia-ict02.pdf

[Highsmith & Cockburn, 2001], Highsmith, J. A. & Cockburn, A. (2001) , 'Desarrollo Agil de Software: El negocio de la innovación', *Computer* 34(9), pp. 120-122.

[Cockburn, 2001], Cockburn, A. (2001), *Agile Software Development*, The Agile Software Development Series, primera edición, Addison-Wesley, Massachusetts, USA.

[Abrahamasson et al. 2001] Abrahamasson, P., Salo, O., Ronkainen, R. & Warsta, J. (2001), Agile software development methods: Reviews and analysis, Reporte Técnico 478, VTT Electronics.

[Fowler & Highsmith, 2001], Fowler, M. & Highsmith, J. A. (2001), 'El manifiesto agil', *Desarrollo de Software*.

[Beck, 2004], Kent Beck, (2004), *Extreme Programming Explained: Embrace Change*, XP Series, Segunda Edición, Addison-Wesley, Massachusetts, USA.

[Jeffries, A, H, 2008], Ron Jeffries, Ann Anderson, Chet Hendrickson, *Extreme Programming Installed*, 2008, Addison-Wesley, Massachusetts, USA.

[Auer&Miller, 2001], Ken Auer, Roy Miller, *Extreme Programming Applied: Playing to Win*, 2001.

[Beck, 2004], Kent Beck, (2004), *Extreme Programming Installed: Embrace Change*, XP Series, Segunda Edición, Addison-Wesley, Massachusetts, USA.

[Martin, 2002], Martin, R. C. (2002), *Desarrollo Agil de Software: Principios, Patrones, y Prácticas*, Alan Apt Series, Primera edición, Prentice Hall, New Jersey, USA.

[FrVreed, 2006], Fruhling Ann; De Vreede Gert-Jan, Field Experiences with eXtreme Programming: Developing an Emergency Response System, Journal of Management Information Systems Año: 2006.

[BarryTurner, 2003] Boehm, Richard Turner, Observations on Balancing Discipline and Agility, IEEE, Proceedings of the Agile Development Conference (ADC.03) 2003.

[Nawrocki, Walter & Wojciechowski, 2001] Nawrocki, J., Walter, B., Wojciechowski, A.: Toward Maturity Model for eXtreme Programming. Tema de discusión de la 27ava Conferencia EUROMICRO, Los Alamitos.IEEE Computer Society (2001) 233.239;

[EuroComStd, 2000] Comité Europeo para la Estandarización: Requerimientos para el sistema de Gestión de Calidad (ISO 9001:2000).] Comité Europeo para la Estandarización (2000)