INSTITUTO SUPERIOR MINERO METALURGICO

“Dr. Antonio Núñez Jiménez”.

Facultad de Metalurgia - Electromecánica

Moa, Holguín

Título

“Metodologías Ágiles para el desarrollo de Software.”

Trabajo de diploma para optar por el título de Ingeniería en Informática

Autor: Jonny Luis Caballero Nuñez.

Tutor(s): Ing. Roiky Rodríguez Noa.

Moa, Cuba

2009

Agradecimientos

Agradezco a mi madre por siempre la luz de mi vida, la que guía mis pasos y sobre todo me inspira en cada paso que doy. A mi abuela querida por brindarme todo ese amor de forma incondicional.

También agradecer a mi tutor por dedicarme siempre un espacio para mis necesidades.

A mis compañeros tanto como Obed, Dariel y Ramonita por brindarme su mano en momentos de dudas y problemas. También a mi sincero amigo Lisandro Lamas por su aporte conmigo en la realización de esta tesis.

Dedicatoria

Dedico la realización de esta tesis a mi madre que ha sido la única persona que ha hecho posible mi estancia en la universidad, ocupándose de mí en todos los aspectos de la vida.

También dedico este trabajo a mi abuela querida que es una de las cosas más importante para mí.

Para esas dos personas que menciono todo mi corazón y amor incondicional.

Resumen

Con nuestro trabajo se desea obtener una guía metodológica para la utilización de La Metodología Ágil XP. Para darle cumplimiento a esta tarea primeramente realizaremos un estudio tanto de Las Metodologías Ágiles como Las Tradicionales con el fin de profundizar en el conocimiento de ambos grupos de metodologías. Este objetivo fue cumplido con la realización de nuestro primer capítulo que nos permitió alcanzar una mayor fundamentación teórica da ambas metodologías. También se realizará una comparación entre Las Metodologías Ágiles y Las Tradicionales, este objetivo tuvo cumplimiento junto con la realización del capítulo dos de nuestro trabajo, este nos brindó un mayor conocimiento acerca de las ventajas y desventajas de ambas metodologías, sus principales características, etc. También se realizará una encuesta que nos permitirá conocer acerca de la aceptación de XP, este objetivo fue obtenido durante el desarrollo del capítulo tres del trabajo así nos permitió conocer la opinión que tienen los desarrolladores de software de nuestro entorno sobre las metodologías ágiles, es decir si es eficiente su aplicación en un (X) determinado problema. Y la obtención de una guía metodológica para la utilización de la metodología ágil XP aplicada al desarrollo de software.

ABSTRACT

With our work it is wanted to obtain a methodological guide for the use of The Agile Methodology XP. To give execution to this task firstly will carry out a study so much of The Agile Methodologies as The Traditional ones with the purpose of deepening in the knowledge of both groups of methodologies. This objective was fulfilled the realization of our first chapter that allowed us to reach a bigger theoretical foundation he/she gives both methodologies. He/she will also be carried out a comparison between The Agile Methodologies and The Traditional ones, this objective had execution together with the realization of the chapter two of our work, this it offered us a bigger knowledge about the advantages and disadvantages of both methodologies, its main ones characteristic, etc. will Also be carried out a survey that will allow us to know about the acceptance of XP, this objective it was obtained during the development of the chapter three of the work it allowed this way us to know the opinion that you/they have the developers of software of our environment on the agile methodologies, that is to say if it is efficient its application in a (X) certain problem. And the obtaining of a methodological guide for the use of the agile methodology XP applied to the software development.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Esta década ha empezado con un creciente interés en metodologías de desarrollo. El proceso de desarrollo a marcado énfasis en el control del proceso, a través de una definición de roles, actividades y artefactos, que incluía modelado y la documentación detallada. Esta forma tradicional para abordar el desarrollo de software ha demostrado ser efectivo y necesario para proyectos de gran tamaño (respecto a tiempo y recursos), donde se exige un alto grado de ceremonia en el proceso. Es decir en proyectos donde los requisitos de trabajo se tienen bien definidos, se cuenta con un amplio tiempo para el desarrollo y sobre todo se tendrá un alto capital humano, es decir estará conformado el equipo desarrollador de software.

Al mirar nuestro ámbito sabemos que la metodología de más aplicación es la metodología tradicional ya que es la más conocida y la que goza de mayor popularidad entre nuestros desarrolladores ya sean a nivel nacional o en nuestro entorno. Al decir nuestro entorno se refiere al municipio donde incluimos al Instituto (ISMM) y todas las industrias aledañas, donde la utilización de las metodologías tradicionales ya forma parte de su grupo de herramientas para el desarrollo de software. Ejemplo claro de esto se puede observar en el Instituto (en la carrera de informática), es decir en los estudiantes, que son los que se encargan directamente del desarrollo de software ya sea durante la realización de las prácticas laborales, de un proyecto investigativo, etc., en fin son trabajos que requieren poco tiempo de desarrollo y de una documentación muy extensa, donde los requisitos de trabajo son muy cambiantes y sobre todo no se cuenta con un equipo de trabajo, no existe un equipo desarrollador de software, y todo el trabajo informático (roles) recae sobre el estudiante. Otro caso muy particular es durante el trabajo de diploma (realización de la tesis), en mucho casos son tesis multidisciplinarias que se necesita del apoyo de conocimiento de la persona especializada en el tema, esto conlleva a muchos cambios en los requisitos de trabajo, debido al cambio constante de lo que se quiere y lo que se desea obtener, el tiempo de desarrollo no es el adecuado, tampoco se cuenta con un equipo de desarrollo, todo el trabajo recae sobre una persona.

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. Entre ellas se encuentran las 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. Esta forma de desarrollo no resulta ser la más adecuada para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante y se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Antes las dificultades para utilizar metodologías tradicionales con estas restricciones de tiempo y flexibilidad, en este mismo escenario emergen Las Metodologías Ágiles (AMs) como una posible respuesta para llenar este vacío metodológico. La filosofía de las AMs es centrarse en otras dimensiones como por ejemplo el factor humano o el producto software dándole 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.

Partiendo de las cuestiones antes mencionadas y debido a la no existencia de conocimiento sobre el desarrollo de las Metodologías Ágiles (AMs) aplicadas al software, se dio la tarea de diseñar una guía de cómo utilizar Las (AMs) aplicadas al desarrollo de software, en aquellos proyectos, los equipos de trabajo son pequeños, con lazos reducidos, requisitos volátiles (muy cambiantes) y/o basadas en nuevas tecnologías. Por lo que en el trabajo que se presenta se atenderá al siguiente ***problema científico:***

La Ineficiencia de Las Metodologías Tradicionales para el desarrollo de Software en entornos donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

El **objeto de estudio** en el cual se enmarca el problema planteado lo constituyen las metodologías ágiles para el desarrollo de software.

Se propone como **objetivo general** para dar solución al problema científico, proponer una guía para uso de una metodología ágil para el desarrollo de Software en entornos donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Precisándose como ***campo de acción*** Las Metodologías Ágiles en entornos donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Se plantea la siguiente ***hipótesis de investigación:*** Utilizando las Metodologías Ágiles será más eficiente el desarrollo de Software en entornos donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Para darle respuesta al objetivo general se plantean los siguientes **objetivos específicos**:

1. Fundamentación teórica de Las Metodologías Ágiles (AMs) y Las Metodologías Tradicionales con el fin de profundizar en el conocimiento de ambos grupos de metodologías.
2. Comparación entre Las Metodologías Ágiles y Las Metodologías Tradicionales para obtener las ventajas y desventajas de su aplicación.
3. Aplicación de una encuesta para determinar la aceptación que tienen las metodologías ágiles en nuestro entorno.

La metodología utilizada para darles solución a las tareas propuestas fue una combinación de métodos teóricos y empíricos.

Métodos teóricos:

Análisis y síntesis: permitió procesar la información en la elaboración de los fundamentos teóricos y las conclusiones.

Histórico lógico: Para el estudio de la evolución del problema y el desarrollo de Las Metodologías Ágiles aplicadas al software pero en entornos muy cambiantes y los tiempos de desarrollo son reducidos.

Hipotético deductivo: Para la elaboración de la hipótesis y la deducción de los resultados de la investigación.

Métodos empíricos:

Encuestas: Estas fueron realizadas, para obtener los resultados pertinentes acerca de la opinión que tienen sobre Las Metodologías Ágiles los desarrolladores de software de nuestro entorno, es decir si son eficientes o no.

El trabajo que se presenta consta de tres capítulos: El *Capítulo 1* se titula “Las Metodologías Ágiles como desarrolladoras de software” y contiene los fundamentos de los aspectos relacionados con el objeto de estudio y del uso de Las (AMs) como desaerrolladoras de software, que son las (Ams), como surgen, sus características principales, sus antescedentes y la ejemplificación de las más conocidas con algunas de sus características.

El *Capítulo 2* “Metodologías Ágiles para el Desarrollo de Software. XP – Extreme Programming, Scrum. Porque La Selección de estas Metodologías sobre las Tradicionales.” Este contiene el manifiesto de las (AMs) con sus doce principios, una comparación entre Las Metodologías Ágiles y Las Metodologías Tradicionales, porque usar las (AMs) y una explicación del funcionamiento de las dos Metodologías Ágiles más conocidas que son: XP – Extreme Programming, Scrum.

En el *Capítulo 3* “Utilización de XP en un ejemplo práctico”, se explicaran las prácticas que se tomaron en cuenta para realizar nuestro ejemplo, también el funcionamiento de un ciclo de desarrollo XP para nuestro trabajo y lo que se modela.

Capítulo I: Las Metodologías Ágiles como desarrolladoras de Software.

Las metodologías ágiles surgen como una extensión a las metodologías tradicionales para mejorar el desarrollo de sistemas según el tipo de proyecto y empresa, añadiendo y mejorando (optimizando) las practicas de desarrollo de software. Lo que no se debe olvidar y tener en cuenta es que son muy usadas, son mezcladas con otras y están para quedarse, evolucionar y revolucionar las practicas de ingeniería.

1.1 De Metodología a Metodologías Ágiles (AMs):

Con mucho el desarrollo de software es una actividad caótica, frecuentemente caracterizada por la frase "codifica y corrige". El software se escribe con un mínimo un plan subyacente, y el diseño del sistema se adoquina con muchas decisiones a corto plazo. Esto realmente funciona muy bien si el sistema es pequeño, pero conforme el sistema crece llega a ser cada vez más difícil agregar nuevos aspectos al mismo. Además los bugs llegan a ser cada vez más frecuentes y más difíciles de corregir. La seña típica de tal sistema es una larga fase de pruebas después de que el sistema ha sido "completado". Tal fase larga de pruebas hace estragos con los planes de pruebas y depurado llegando a ser imposible de poner en el programa de trabajo.

Las metodologías ingenieriles han estado presentes durante mucho tiempo. No se han distinguido precisamente por ser muy exitosas. Aún menos por su popularidad. La crítica más frecuente a estas metodologías es que son burocráticas. Hay tanto que hacer para seguir la metodología que el ritmo entero del desarrollo se retarda. [1]

Como una reacción a estas metodologías, un nuevo grupo de metodologías ha surgido en los últimos años. Durante algún tiempo se conocían como las metodologías ligeras, pero el término aceptado ahora es metodologías ágiles.

Para mucha gente el encanto de estas **metodologías ágiles** es su reacción a la burocracia de las metodologías monumentales. Estos nuevos métodos buscan un justo medio entre ningún proceso y demasiado proceso, proporcionando simplemente suficiente proceso para que el esfuerzo valga la pena.

El resultado de todo esto es que los métodos ágiles cambian significativamente algunos de los énfasis de los métodos ingenieriles. La diferencia inmediata es que son menos orientados al documento, exigiendo una cantidad más pequeña

de documentación para una tarea dada. En muchas maneras son más bien orientados al código: siguiendo un camino que dice que la parte importante de la documentación es el código fuente.

Por qué surgen Las Metodologías Ágiles (AMs). [2]

- Dificultad para implantar Metodologías Tradicionales. Sofisticadas herramientas CASE y notaciones (UML).
- Una solución a medida por un segmento importante de proyectos de desarrollo de software.
- Pugnas entre comunidades/gurús.
- "Aceptar el cambio".

Costo de los Cambios en Software.

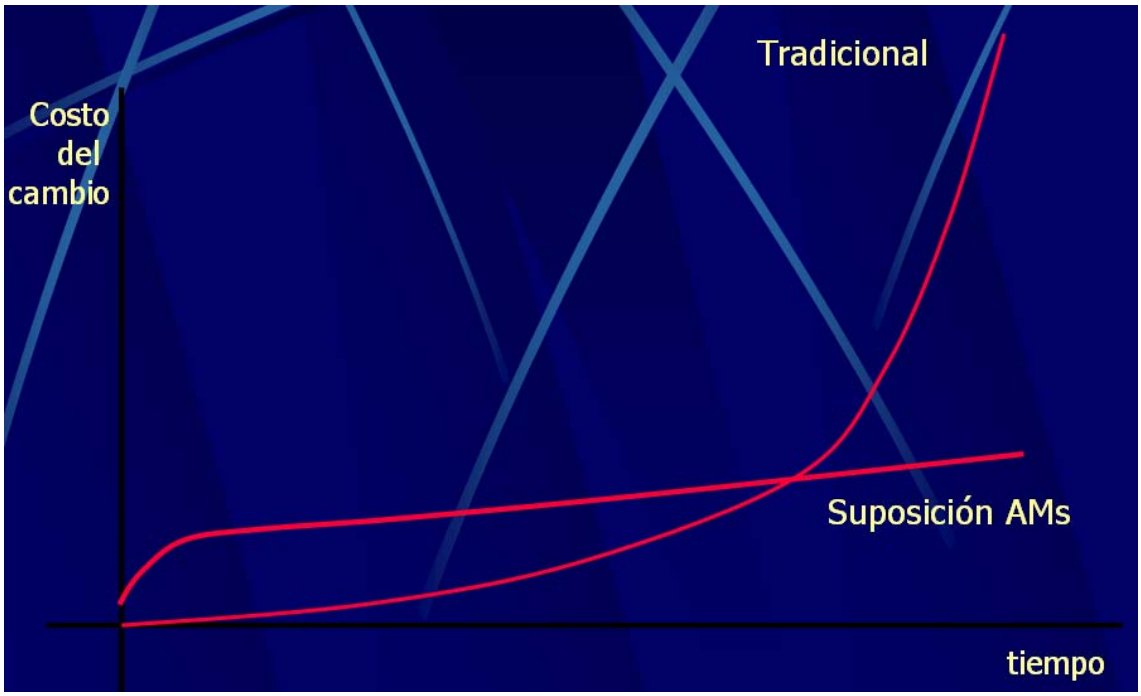


Figura N°1 Costo de los cambios en SW.

1.2 Metodologías Ágiles (Ams):

Las Metodologías Ágiles son una supuesta solución a las falencias e inconvenientes que muestran o presentan los tradicionales métodos de desarrollo (en otras palabras una extensión de los métodos comunes). [3]

Los procesos ágiles ofrecen un enfoque diametralmente opuesto para gestionar y controlar el desarrollo de productos y proyectos de software.

Se basan en principios tales como la temprana y continua entrega de software de valor; la actitud positiva frente al cambio de requerimientos en función de las necesidades del cliente (aún en etapas tardías del desarrollo); la motivación individual de los integrantes del proyecto; la comunicación fluida entre el personal de desarrollo y de negocios; la excelencia técnica; la simplicidad; la comunicación personal; la emergencia y la autoorganización. [4]

Las metodologías ágiles de desarrollo de software son una agrupación de las prácticas tradicionales pero llevadas al extremo, tomando la esencia y aplicándolas buscando la calidad en el desarrollo desde el inicio, entregas oportunas y la entrega final del sistema, teniendo en cuenta el soporte, mantenimiento, auditoria y capacitaciones al usuario final.[3]

Las Metodologías Ágiles (AMs) valoran [2]:

- Al individuo las interacciones en el equipo de desarrollo más que a las actividades y las herramientas.
- Desarrollar software que funciona más que conseguir una buena documentación. Minimalismo respecto del modelado y la documentación del sistema.
- La colaboración con el cliente más que la negociación de un contrato.
- Responder a los cambios más que seguir estrictamente una planificación.

1.2.1 Principales Características:

Las Ams. Se caracterizan por:

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

Los métodos ágiles son orientados a la gente y no orientados al proceso. La meta de los métodos ingenieriles es definir un proceso que funcionará bien con cualquiera que lo use. Los métodos ágiles afirman que ningún proceso podrá nunca maquillar las habilidades del equipo de desarrollo, de modo que el papel del proceso es apoyar al equipo de desarrollo en su trabajo. Explícitamente

puntualizan el trabajar a favor de la naturaleza humana en lugar de en su contra y enfatizan que el desarrollo de software debe ser una actividad agradable.
Que estas metodologías están apuntando a ser las sustitutas de las metodologías en cascadas.

1.2.2 Antecedentes de las Ams:

El principal antecedente de esta revolución en la forma de desarrollar hoy en día el software es el Proceso Unificado (RUP) fue desarrollado por La Racional como el proceso complementario al UML. El RUP es un armazón de proceso y como tal puede acomodar una gran variedad de procesos.
Es considerado por sus creadores un proceso completo en el desarrollo de software respaldado por más de tres décadas de desarrollo y de aplicaciones prácticas.
Decir que este puede usarse en un estilo muy tradicional de cascada o de manera ágil. Como resultado este puede usarse, como un proceso ágil, o como un proceso pesado, todo depende de cómo se adapte al ambiente.
Otra tachuela en el RUP Ágil es el proceso dX. El proceso dX es una versión totalmente dócil del RUP que simplemente es idéntico a la XP. El dX está diseñado para gente que tiene que usar el RUP pero quiere usar XP. Como tal es a la vez XP y RUP y por tanto un buen ejemplo del uso ágil del RUP. [5]
A partir del surgimiento de este, se tiene un Proceso Unificado para soportar todo un ciclo de vida de un proceso de software, el cual se sigue mejorando con la ayuda de muchas empresas y desarrolladores.
Pero que los líderes de RUP en la industria enfaticen su acercamiento al desarrollo de software, ya que la gente que usa RUP ha planteado que están usando un proceso de desarrollo estilo cascada.

1.3 Metodologías ágiles en si:

Varias metodologías encajan bajo el estandarte de ágil. Mientras todas ellas comparten muchas características, también hay algunas diferencias significativas.
A continuación alguna de las características fundamentales de las metodologías ágiles más destacadas hasta el momento, parte de su funcionamiento en si y los nombres de los responsables del surgimiento de

estas metodologías así como la última publicación realizada por ellos mismos sobre los posibles cambios que puedan sufrir estas.

XP – Extreme Programming.

Scrum.

Crystal Clear.

DSDM – Dynamic Systems Development Method.

FDD – Feature Driven Development.

ASD – Adaptive Software Development.

XBreed.

Extreme Modeling.

1.3.1 XP (Programación Extrema):

De todas las metodologías ágiles, ésta es la que ha recibido más atención. Esto se debe en parte a la notable habilidad de los líderes XP, en particular Kent Beck, para llamar la atención. También se debe a la habilidad de Kent Beck de atraer a las personas a este acercamiento, y tomar un papel principal en él. De algunas maneras, sin embargo, la popularidad de XP se ha vuelto un problema, pues ha acaparado la atención fuera de las otras metodologías y sus valiosas ideas.

Las raíces de la XP yacen en la comunidad de Smalltalk, y en particular la colaboración cercana de Kent Beck y Ward Cunningham a finales de los 80. Ambos refinaron sus prácticas en numerosos proyectos a principios de los 90, extendiendo sus ideas de un desarrollo de software adaptable y orientado a la gente. [3]

El paso crucial de la práctica informal a una metodología ocurrió en la primavera de 1996. A Kent se le pidió revisar el progreso del proyecto de nómina C3 para Chrysler. El proyecto estaba siendo llevado en Smalltalk por una compañía contratista, y estaba en problemas. Debido a la baja calidad de la base del código, Kent recomendó tirar la base del código en su totalidad y empezar desde el principio. El proyecto entonces reinició bajo su dirección y subsecuentemente se volvió el buque insignia y el campo de entrenamiento de la XP.

La XP empieza con cuatro valores: Comunicación, Retroalimentación, Simplicidad y Coraje. Construye sobre ellos una docena de prácticas que los

proyectos XP deben seguir. Muchas de estas prácticas son técnicas antiguas, tratadas y probadas, aunque a menudo olvidadas por muchos, incluyendo la mayoría de los procesos planeados. Además de resucitar estas técnicas, la XP las teje en un todo sinérgico dónde cada una refuerza a las demás.

Una de las más llamativas, así es su fuerte énfasis en las pruebas. Mientras todos los procesos mencionan la comprobación, la mayoría lo hace con muy poco énfasis. Sin embargo la XP pone la comprobación como el fundamento del desarrollo, con cada programador escribiendo pruebas cuando escriben su código de producción. Las pruebas se integran en el proceso de integración continua y construcción, lo que rinde una plataforma altamente estable para el desarrollo futuro.

En esta plataforma XP construye un proceso de diseño evolutivo que se basa en refactorar un sistema simple en cada iteración. Todo el diseño se centra en la iteración actual y no se hace nada anticipadamente para necesidades futuras. El resultado es un proceso de diseño disciplinado, lo que es más, combina la disciplina con la adaptabilidad de una manera que indiscutiblemente la hace la más desarrollada entre todas las metodologías adaptables.

1.3.2 Los Puntos más interesantes de XP son:

- Desarrollo iterativo e incremental.
- Pruebas unitarias continuas, frecuentemente repetidas y automáticas.
- Programación en parejas.
- Frecuentemente interacción del equipo de programación con el cliente o usuario.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Hacer entregas frecuentes.
- Refactorización del código.

Propiedad del código compartida: promueve el que todo personal pueda corregir y extender cualquier parte del proyecto.

Simplicidad en el código.

Sus objetivos:

La satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita.

Potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software.

Una de las cosas que a los desarrolladores les tiene que quedar claro es que en el ciclo de vida del desarrollo de un proyecto de software, es que cambios pueden a aparecer: cambiarán los requisitos, las reglas de negocio, el personal, la tecnología, todo puede a cambiar. Por tanto el problema no es el cambio en si, ya que se asume que este va a suceder, sino la incapacidad de enfrentarnos a estos cambios. XP toma en cuenta cuatro valores fundamentales para enfrentar estas situaciones: Comunicación, Simplicidad, Retroalimentación y Respeto.

1.4 Scrum [3]:

Scrum ha estado durante algún tiempo en los círculos orientados a objetos. De nuevo se enfoca en el hecho de que procesos definidos y repetibles sólo funcionan para atacar problemas definidos y repetibles con gente definida y repetible en ambientes definidos y repetibles.

Scrum divide un proyecto en iteraciones (que ellos llaman carreras cortas) de 30 días. Antes de que comience una carrera se define la funcionalidad requerida para esa carrera y entonces se deja al equipo para que la entregue. El punto es estabilizar los requisitos durante la carrera.

Sin embargo la gerencia no se desentiende durante la carrera corta. Todos los días el equipo sostiene una junta corta (quince minutos), llamada scrum, dónde el equipo discurre lo que hará al día siguiente. En particular muestran a los bloques de la gerencia: los impedimentos para progresar que se atraviesan y que la gerencia debe resolver. También informan lo que se ha hecho para que la gerencia tenga una actualización diaria de dónde va el proyecto.

La literatura de Scrum se enfoca principalmente en la planeación iterativa y el seguimiento del proceso. Es muy cercana a las otras metodologías ágiles en muchos aspectos y debe funcionar bien con las prácticas de código de la XP.

Después de mucho tiempo sin un libro, finalmente Ken Schwaber y Mike Beedle escribieron el primer libro de scrum. Ken Schwaber también aloja controlChaos.com qué probablemente es la mejor apreciación global sobre SCRUM. Jeff Sutherland siempre ha tenido un sitio Web activo sobre temas de

tecnologías de objetos e incluye una sección sobre SCRUM. Hay también una buena apreciación global de las prácticas de Scrum en el libro PLoPD 4. Scrum tiene una lista de discusión en yahoo.

1.4.1 Principales características de Scrum.

El desarrollo de software se realiza mediante iteraciones, denominadas sprints o carreras cortas, con una duración de 30 días. Antes de que comience una carrera se define la funcionalidad requerida para esa carrera y entonces se deja al equipo para que la entregue. El punto es estabilizar los requisitos durante la carrera. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. [3]

La segunda característica importante son las reuniones a lo largo proyecto. La reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración, en la que todos cuentan qué han hecho ayer, qué van hacer hoy y qué problemas tiene para hacer lo que están haciendo.

Como vemos, Scrum no dice nada de si hacer o no hacer diseño, si hacer o no hacer pruebas, si hacer o no hacer documentación, si trabajar en parejas o no, etc. Scrum únicamente nos indica cómo conseguir que todos trabajen con el mismo objetivo, a corto plazo y deja bastante visible como avanza el proyecto día a día. No es propiamente un método o metodología de desarrollo, e implantarlo como tal resulta insuficiente, Scrum define métodos de gestión y control para complementar la aplicación de otros métodos ágiles como XP o alguna otra metodología de desarrollo.

1.5 Desarrollo manejado por Rasgos (FDD)

El Desarrollo Manejado por Rasgos (FDD por sus siglas en inglés) fue desarrollado por Jeff De Luca y el viejo gurú de la OO Peter Coad. Como las otras metodologías adaptables, se enfoca en iteraciones cortas que entregan funcionalidad tangible. En el caso del FDD las iteraciones duran dos semanas.

El FDD tiene cinco procesos. Los primeros tres se hacen al principio del proyecto.

Desarrollar un Modelo Global

Construir una Lista de los Rasgos

Planear por Rasgo

Diseñar por Rasgo

Construir por Rasgo

Los últimos dos se hacen en cada iteración. Cada proceso se divide en tareas y se da un criterio de comprobación.

Los desarrolladores entran en dos tipos: dueños de clases y programadores jefe. Los programadores jefe son los desarrolladores más experimentados. A ellos se les asignan rasgos a construir. Sin embargo ellos no los construyen solos. Solo identifican qué clases se involucran en la implantación de un rasgo y juntan a los dueños de dichas clases para que formen un equipo para desarrollar ese rasgo. El programador jefe actúa como el coordinador, diseñador líder y mentor mientras los dueños de clases hacen gran parte de la codificación del rasgo. [3]

Hasta recientemente, la documentación sobre FDD era muy escasa. Finalmente hay un libro completo sobre FDD. Jeff De Luca, el inventor primario, ya tiene un portal FDD con artículos, blogs y foros de discusión. La descripción original estaba en el libro UML in Color de Peter Coad et al. Su compañía, TogetherSoft, también da consultoría y entrenamiento en FDD.

1.6 Método de Desarrollo De Sistema Dinámico. (DSDM)

El DSDM empezó en Gran Bretaña en 1994 como un consorcio de compañías del Reino Unido que querían construir sobre RAD [N. del T. Desarrollo Rápido de Aplicaciones] y desarrollo iterativo. Habiendo empezado con 17 fundadores ahora tiene más de mil miembros y ha crecido fuera de sus raíces británicas. Siendo desarrollado por un consorcio, tiene un sabor diferente a muchos de los otros métodos ágiles. Tiene una organización de tiempo completo que lo apoya con manuales, cursos de entrenamiento, programas de certificación y demás. También lleva una etiqueta de precio, lo qué ha limitado mi investigación sobre su metodología. Sin embargo Jennifer Stapleton ha escrito un libro que da una apreciación global de la metodología. [6]

El método empieza con un estudio de viabilidad y negocio. El estudio de viabilidad considera si DSDM es apropiado para el proyecto. El estudio de negocio es una serie corta de talleres para entender el área de negocio dónde tiene lugar el desarrollo. También propone arquitecturas de esbozos del sistema y un plan del proyecto.

El resto del proceso forma tres ciclos entretejidos: el ciclo del modelo funcional produce documentación de análisis y prototipos, el ciclo de diseño del modelo diseña el sistema para uso operacional, y el ciclo de implantación se ocupa del despliegue al uso operacional.

DSDM tiene principios subyacentes que incluyen una interacción activa del usuario, entregas frecuentes, equipos autorizados, pruebas a lo largo del ciclo. Como otros métodos ágiles usan ciclos de plazos cortos de entre dos y seis semanas. Hay un énfasis en la alta calidad y adaptabilidad hacia requisitos cambiantes.

No se ha visto mucha evidencia de uso fuera del Reino Unido, pero DSDM es notable por tener mucha de la infraestructura de las metodologías tradicionales más maduras, al mismo tiempo que sigue los principios de los métodos ágiles.

Conclusiones:

- Las metodologías ágiles de desarrollo de software son una agrupación de las prácticas tradicionales pero llevadas al extremo.
- El principal antecedente de las metodologías ágiles, es el Proceso Unificado (RUP) fue desarrollado por La Racional como el proceso complementario al UML.
- La metodología XP dentro de las metodologías ágiles de desarrollo de software es la que más popularidad ha alcanzado entre los desarrolladores.

Capítulo II. Metodologías Ágiles para el Desarrollo de Software. XP – Extreme Programming, Scrum. Porque La Selección de estas Metodologías sobre las Tradicionales.

2.1 Manifiesto Ágil.

En Marzo del 2001 diecisiete críticos de los modelos de mejoras del desarrollo de software basados en procesos convocados por Kent Beck quien había publicado un par de años antes Extreme Programming Explained, libro en el que se exponía una nueva metodología denominada Extreme Programming, se reunieron en Salt Lake City para tratar sobre técnicas y procesos para desarrollar el software. En la reunión se acuñó el término “Métodos Ágiles” para definir a los métodos que estaban surgiendo como alternativas a las metodologías formales (CMMI, SPICE) a las que consideraban excesivamente “pesadas” y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo.

Los integrantes de la reunión resumieron los principios sobre los que se basan los métodos alternativos en cuatro postulados lo que ha quedado dominado como Manifiesto Ágil [3].

2.1.1 Valores del Manifiesto Ágil:

Comienza enumerando los principales valores del desarrollo ágil. Estos valores son [7: p.8]:

- **Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.** La gente es el principal factor de éxito de un proyecto de software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comente el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que este configure su propio entorno de desarrollo en base a sus necesidades.
- **Desarrollar software que funciona más que conseguir una buena documentación.** La regla a seguir es “no producir documentos a menos que sean necesarios de forma inmediata para tener una decisión

importante". Esto documentos deben ser cortos y centrase en lo fundamental.

- **La colaboración con el cliente más que la negociación de un contrato.** Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.
- **Responder a los cambios más que seguir estrictamente un plan.** La habilidad de responder a los cambios que puede surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta [3].

2.1.2 Principios del Manifiesto

Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de un tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tiene que ver con el proceso a seguir y el equipo de desarrollo, en cuanto metas a seguir y organización del mismo. Los principios son:

1) *La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.*
2) *Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.*
3) *Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con le menor intervalo de tiempo posible entre entregas.*
4) *La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto*
5) *Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.*
6) *El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.*
7) *El software que funciona es la medida principal de progreso.*

8) *Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.*
9) *La atención continua a la calidad técnica y al buen diseño mejora la agilidad.*
10) *La simplicidad es esencial.*
11) *Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.*
12) *En intervalos regulares, el equipo reflexiona respecto a como llegar a ser más efectivo, y según esto ajusta su comportamiento.*

2.2 Metodologías Ágiles versus Metodologías Tradicionales.

La tabla N°1 recoge esquemáticamente las principales diferencias de las metodologías ágiles con respecto a las tradicionales (no ágiles). Estas diferencias que afectan no solo el proceso en sí, sino también al contexto del equipo así como a su organización.

Tabla N°1 Diferencias entre Metodologías Ágiles y no Ágiles.

Basadas en heurísticas provenientes de practicas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparadas para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho mas controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en le mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tener Metodologías diferentes para aplicar de acuerdo con el proyecto que se desarrolle resulta una idea interesante. Estas Metodologías pueden involucrar prácticas tanto de metodologías ágiles como de metodologías tradicionales. De esta manera podíamos tener una metodología para cada proyecto, la problemática seria definir cada una de las prácticas y en el momento preciso definir parámetros para saber cual usar.

Es importante tener en cuenta que el uso de un método ágil no es para todos. Sin embargo, una de las principales ventajas de los métodos ágiles es su peso inicialmente ligero y por eso las personas que no estén acostumbradas a seguir procesos encuentran estas metodologías bastantes agradables.

2.3 Por qué usar las Metodologías Ágiles.

Tomando las ideas de la tabla N°1 se puede decir que las metodologías tradicionales presentan los siguientes problemas a la hora de abordar proyectos. [8]

- Existen unas costosas fases previas de especificación de requisitos, análisis y diseño. La corrección durante el desarrollo de errores introducidos en estas fases será costosa, es decir, se pierde flexibilidad ante los cambios.
- El proceso de desarrollo esta encorsetado por documentos firmados.
- El desarrollo es más lento. Es difícil para los desarrolladores entender un sistema complejo en su globalidad.

Las Metodologías Ágiles de Desarrollo (AMs) están especialmente indicadas en proyectos con requisitos pocos definidos y cambiantes. Estas presentan diversas ventajas, entre las que podemos destacar.

- Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo.
- Entrega continua y en plazos breves de software funcional.
- Trabajo conjunto entre el cliente y el equipo de desarrollo.
- Importancia y la simplicidad, eliminando el trabajo innecesario.
- Atención continúa a la excelencia técnica y el buen diseño.
- Mejora continua de los procesos y el equipo de desarrollo.

2.3.1 XP - eXtreme Programming.

XP es la primera metodología ágil y la que dio conciencia al movimiento actual de metodologías ágiles. De la mano de Kent Beck, XP ha conformado un extenso grupo de seguidores en todo el mundo, disparando una gran cantidad de libros a los que dio comienzo el mismo Beck en [7]. Inclusive Addison – Wesley ha creado un serie de libros denominada *The XP Series.*

La imagen mental de Beck al crear XP [7] era la de perrillas en un tablero de control. Cada perrilla representa una práctica que de su experiencia sabía que trabajaba bien. Entonces, Beck decidió girar todas las perillas al máximo para ver que ocurría. Así fue como dio inicio XP. [6]

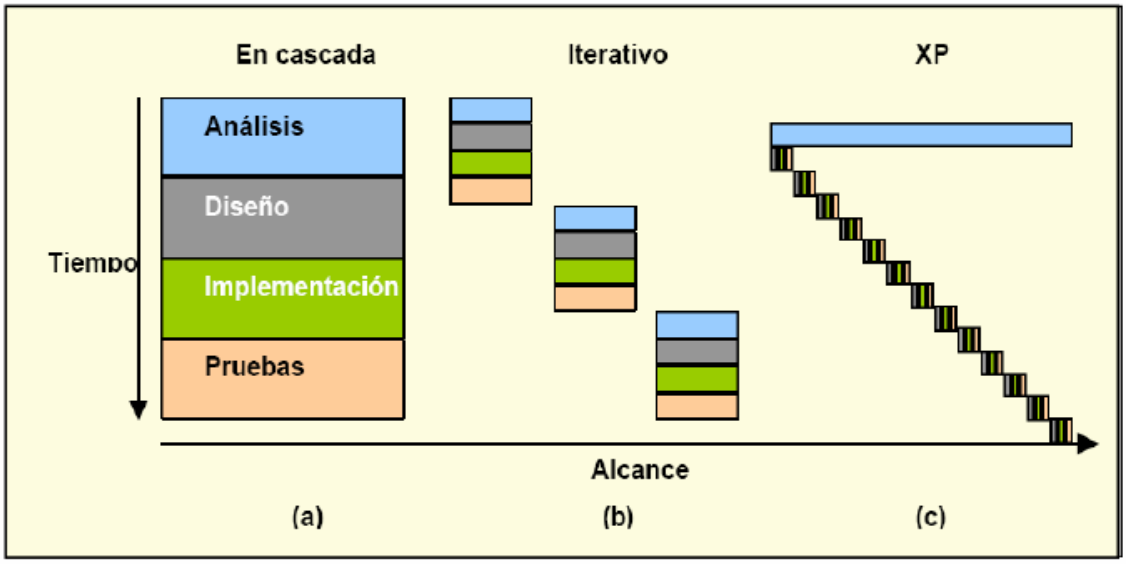


Figura N°2. Evolución de los largos ciclos de desarrollo en cascada (a) a ciclos iterativos más cortos (b) y a la mezcla que hace XP.

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

2.3.1.1 Características de XP:

Las características esenciales de XP organizadas en sus tres apartados son: historia de usuario, roles, proceso y practicas. [6]

Las Historias de Usuarios.

Es la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no. El tratamiento de las historias de usuarios es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en una semana. Beck en su libro [3] presenta un ejemplo de ficha (*customer story and task card*) en el cual pueden reconocerse los siguientes contenidos: fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, numero de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción y una lista de seguimiento con la fecha, estado cosas por terminar y comentarios. A efectos de planificación, las historias pueden ser de una a tres semanas de tiempo de programación (para no superar el tamaño de una iteración). Las historias de usuario son descompuestas en tareas de programación (task card) y asignadas a los programadores para ser implementadas durante una iteración.

Roles XP.

Los roles de acuerdo con la propuesta original de Beck (Creador) son:

Programador. El programador escribe las pruebas unitarias y produce el código del sistema.

Cliente. Escribe las historias de usuarios y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historia de usuario y decide cuales se implementan en cada iteración centrándose en aportar mayor valor al negocio.

Encargado de Prueba (***Tester***). Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Encargado de Seguimiento (*Tracker*). Proporciona retroalimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo

real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.

Entrenador (*Coach*). Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen la practicas XP y se siga el proceso correctamente.

Consultor. Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.

Gestor (*Big Boss*). Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Para observar como interactúan todos estos papeles a interpretarse durante todo un día de trabajo XP, remitirse a anexo 1.

Proceso XP.

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1) El cliente define el valor del negocio a implementar.
2) El programador estima el esfuerzo necesario para su implementación.
3) El cliente selecciona que construir, de acuerdo con sus prioridades y las restricciones del tiempo.
4) El programador construye ese valor de negocio.
5) Vuelve al paso1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe forzar al programador realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible en cada iteración.

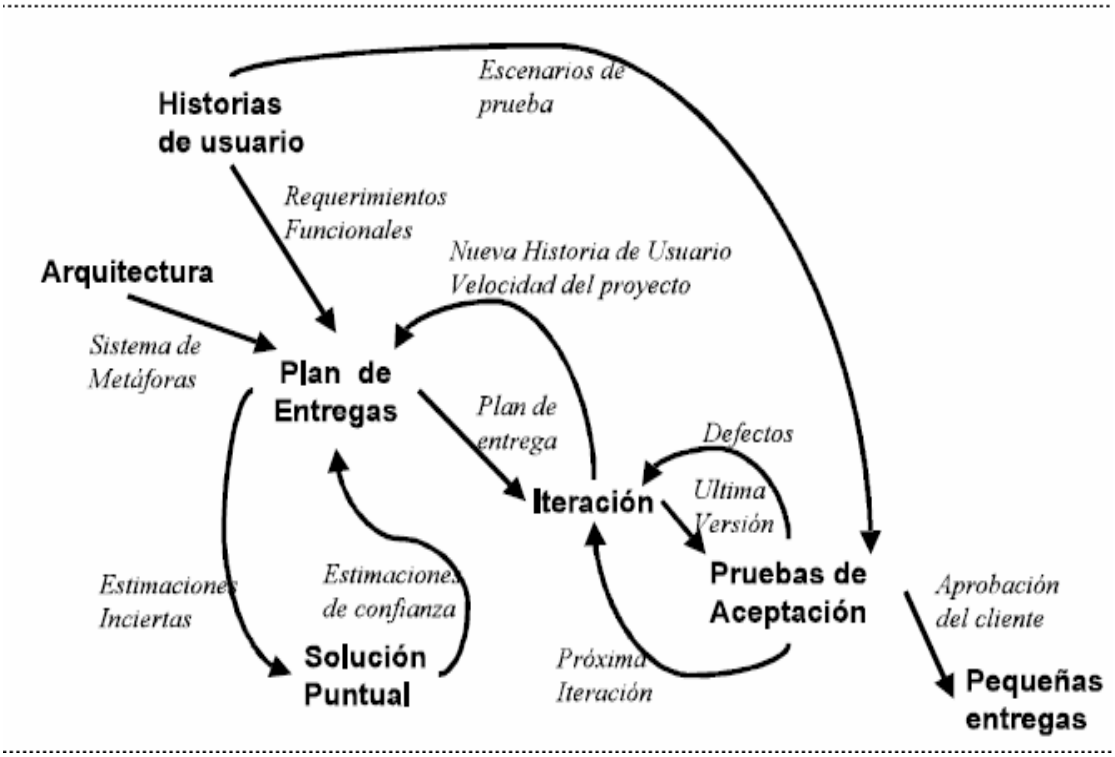


Figura N°3 Proceso XP.

En si el ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (*Release*), Iteraciones, Producción, Mantenimiento y Muerte del proyecto.

Prácticas XP

La principal suposición que realiza XP es la posibilidad de disminuir la mítica curva exponencial del costo de cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas.

El juego de la planificación. Hay una comunicación frecuente, el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.

Entregas pequeñas. Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más 3 meses.

Metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).

Diseño Simple. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un memento determinado del proyecto.

Pruebas. La producción de código esta dirigida por las pruebas unitarias. Estas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.

Refactorización (*Refactoring*). Es una actividad constante de reestructuración del código con le objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo mas flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.

Programación en parejas. Toda la programación de código debe realizarse con trabajo de parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores). Para Observar como sería el ambiente a convivir durante la programación en parejas, remitirse al anexo 2.

Propiedad colectiva del código. Cualquier programador puede cambiar cualquier parte del código en cualquier momento.

Integración continua. Cada pieza de código es insertada en el sistema una vez que esté lista. Así el sistema puede llegar a ser integrado y construido varias veces en un mismo día.

40 horas por semana. No se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo.

Cliente in-situ. El cliente tiene que estar presente todo el tiempo y disponible para el equipo. Este es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportará mayor del

negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita.

Estándares de programación. XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

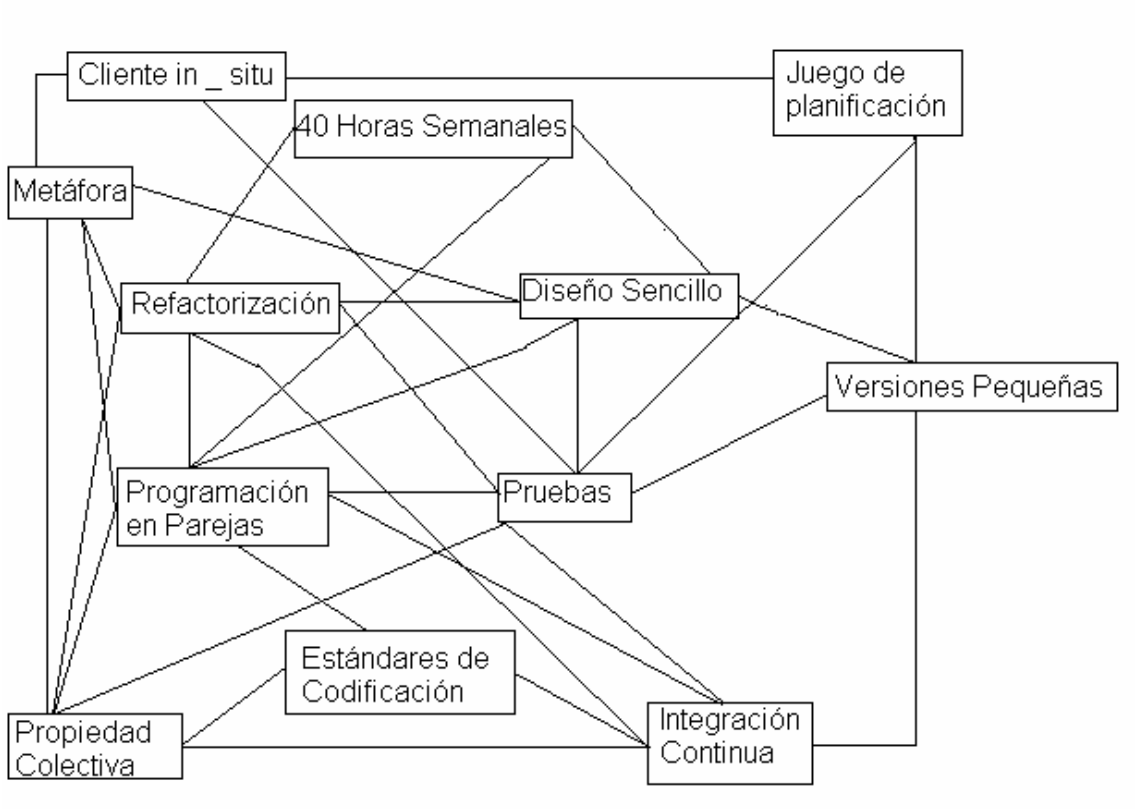


Figura N°4 Las Prácticas se refuerzan entre sí.

El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras. Esto se ilustra en la figura N°4, donde una línea entre dos prácticas significa que las prácticas se refuerzan entre sí.

La mayoría de las prácticas propuestas por XP no son novedosas sino que en alguna forma ya habían sido propuestas en ingeniería de software e incluso demostrado su valor en la práctica. El mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva el negocio, los valores humanos y el trabajo en equipo.

Muchas personas ven esta metodología como algo como algo innovador, es verdad que las ideas que propone no son nuevas: pero XP agrupa un conjunto de buenas prácticas y la lleva al extremo.

La innovación (logro) de XP consiste en:

- Agrupar todas las prácticas.
- Asegurarse de que esas prácticas se lleven a cabo.
- Asegurarse de que las prácticas se soporten entre si en el mayor grado posible.

Si bien XP es la metodología ágil de más renombre en la actualidad se diferencia de las demás metodologías que forman este grupo en un aspecto en particular: El alto nivel de disciplina de las personas que participan en proyecto.

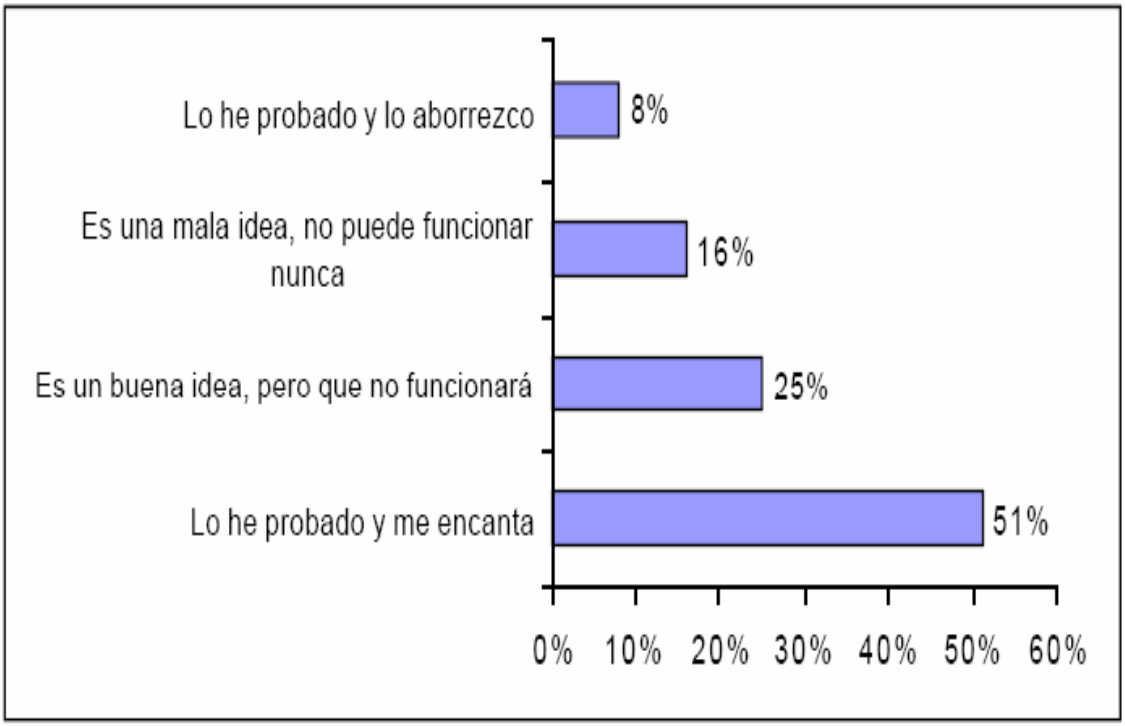


Figura N°5 Encuesta de IBM.

Para conocer acerca de otros títulos (bibliografía) pero sobre la metodología ágil XP, observar anexo3

2.3.2 Scrum.

Scrum define un proceso empírico, iterativo e incremental de desarrollo que intenta obtener ventajas respecto a los procesos definidos (cascada, espiral, prototipo, etc.) mediante la aceptación de la naturaleza caótica del desarrollo de software, y la utilización de prácticas tendientes a manejar la impredictibilidad y el riesgo a niveles aceptables. El mismo surge en 1996, de un artículo de La Harvard Business Review titulado “The New Product Development Game” de

Hirotaka Takeuichi e Ikujiro Nonoka, que introducía las mejores prácticas más utilizadas en 10 compañías japonesas altamente innovadoras. A partir de ahí y tomando referencias al juego de rugby, Ken Scwaber y Jeff Sutherland formalizan el proceso conocido como Scrum en el año 1995. [5]

Scrum es un método iterativo e incremental que enfatiza prácticas y valores de Project management por sobre las demás disciplinas de desarrollo. Al principio del proyecto se define el Product Backlog, que contiene todos los requerimientos funcionales y no funcionales que debería satisfacer el sistema a construir. Los mismos estarán especificados de acuerdo a las convenciones de la organización ya sea mediante: features, casos de usos, diagramas de flujos de datos, incidentes, tareas, etc. El Product Backlog será definido mediante reuniones de planeamientos con los stakeholders. A partir de ahí se definirán las iteraciones, conocidas como Sprint en la juerga de Scrum, en la que se ira evolucionando la aplicación evolutivamente. Cada Sprint tendrá su propio Sprint Backlog que será un subconjunto del Product Backlog con los requerimientos a ser construidos en el Sprint correspondiente. La duración recomendada para el Sprint es de un mes. [7: p. 18]

Dentro de cada Sprint el Scrum master (equivalente al líder del proyecto) llevará a cabo la gestión de la iteración, convocando diariamente el Scrum Daily Meeting que representa una reunión de avance diaria de no más de 15 minutos con el propósito de tener retroalimentación sobre las tareas de los recursos y los obstáculos que se presentan. Al final de cada Sprint, se realizara un Sprint Review para evaluar los artefactos construidos y comentar el planeamiento del próximo Sprint.

Cuando acaba cada reunión diaria de 15 minutos los desarrolladores dan respuesta a las tres preguntas siguientes:

1. Que hizo desde la última reunión.
2. Que dificultades concretas tiene en el desarrollo de la tarea.
3. Que ve hacer hasta la próxima reunión diaria.

El proyecto es abierto hasta la fase de clausura la entrega puede ser replanificada en cualquiera de las fases anteriores manteniéndose abierto a la complejidad del ambiente, la competencia, tiempo, calidad y presiones financieras, durante el desarrollo de dichas fases.

La entrega del proyecto es calculada sobre la influencia del ambiente.

La metodología Scrum está diseñada para ser flexible durante el desarrollo de los sistemas. Provee de mecanismos de control para planificar una entrega del producto y manejar las variables en el progreso del proyecto. Esto permite la organización para cambiar el proyecto y las entregas en cualquier punto del desarrollo, entregando la versión más apropiada.

La metodología Scrum libera a los desarrolladores a inventar las más ingeniosas soluciones durante el proyecto, mientras aprenden y el ambiente cambia. Los equipos de desarrolladores (idealmente alrededor de 7 miembros) pueden intercambiar los conocimientos acerca de los procesos de desarrollo siendo esto una magnífica oportunidad de entrenamiento en el ambiente.

2.3.2.1 Aplicando Scrum al desarrollo de software.

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

Jeff Sutherland aplicó el modelo Scrum al desarrollo de software en 1993 Easel Corporation (Empresa que en los macro-juegos de compras y fusiones se integraría en VMARK, luego en Informix y finalmente en Ascential Software Corporation).

En 1996 lo presentó junto con Ken Scwaber como proceso formal, también para gestión del desarrollo de software en OOPSLA 96. En el desarrollo de software Scrum está considerado como modelo ágil por la Agile Alliance.Como Observarán en la figura N°6 la metodología resulta sencilla definiendo algunos roles y artefactos que contribuyen a tener un proceso que maximiza el feedback para mitigar cualquier riesgo que pueda presentarse.

La intención de Scrum es la de maximizar la realimentación sobre el desarrollo pudiendo corregir problemas y mitigar riesgos de forma temprana. Su uso se está extendiendo cada vez más dentro de la comunidad de Metodologías Ágiles, siendo combinado con otras – como – XP para completar sus carencias. Cabe mencionar que Scrum no propone el uso de ninguna práctica de desarrollo en particular; sin embargo, es habitual emplearlo como un Framework ágil de

administración de proyectos que puede ser combinado con cualquier de la metodologías mencionadas.

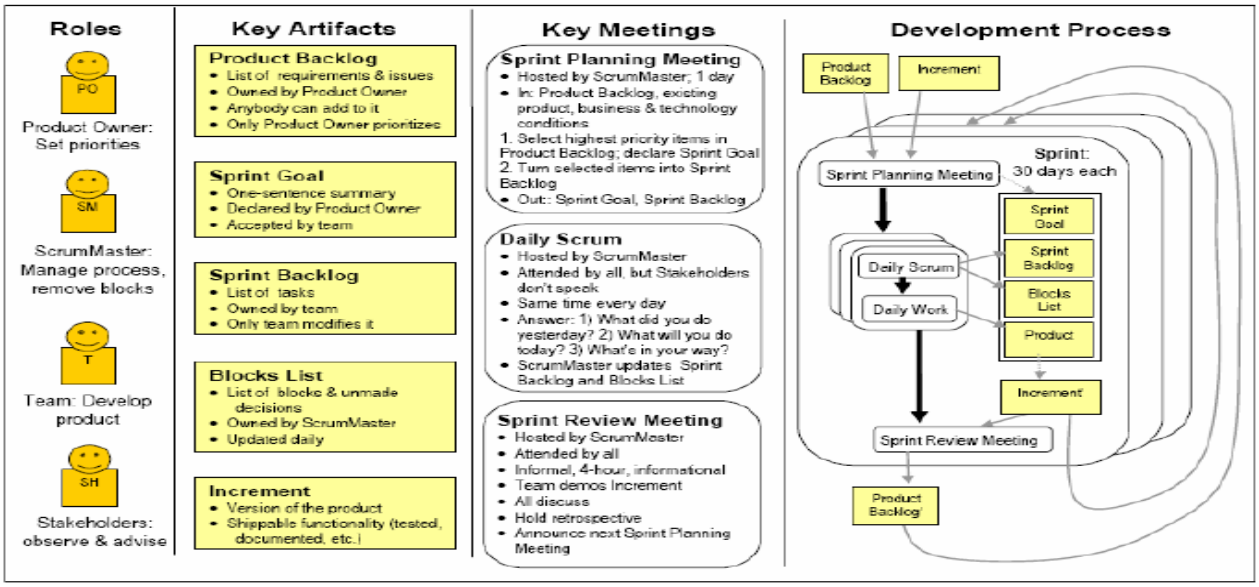


Figura N°6 Roles, artefactos, reuniones y proceso de desarrollo de Scrum.

2.3.2.2 Ciclo de Vida de Scrum. [7: p. 20]

El ciclo de vida de Scrum es el siguiente:

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

Pre – Juego Montaje (Staging). El propósito es identificar más requerimientos y priorizar las tareas para la primera iteración. Las actividades son planificación, diseño exploratorio y prototipos.

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

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

Usualmente los registros de acumulación se llevan en planillas de cálculo comunes, antes que en una herramienta sofisticada de gestión de proyectos. Los elementos del registro pueden ser prestaciones del software, funciones, corrección de bugs, mejoras requeridas y actualizaciones de tecnología. Hay un registro total del producto y otro específico para cada corrida de treinta días. En la juerga de Scrum se llaman “paquetes” a los objetos o componentes que necesitan cambiarse en la siguiente iteración.

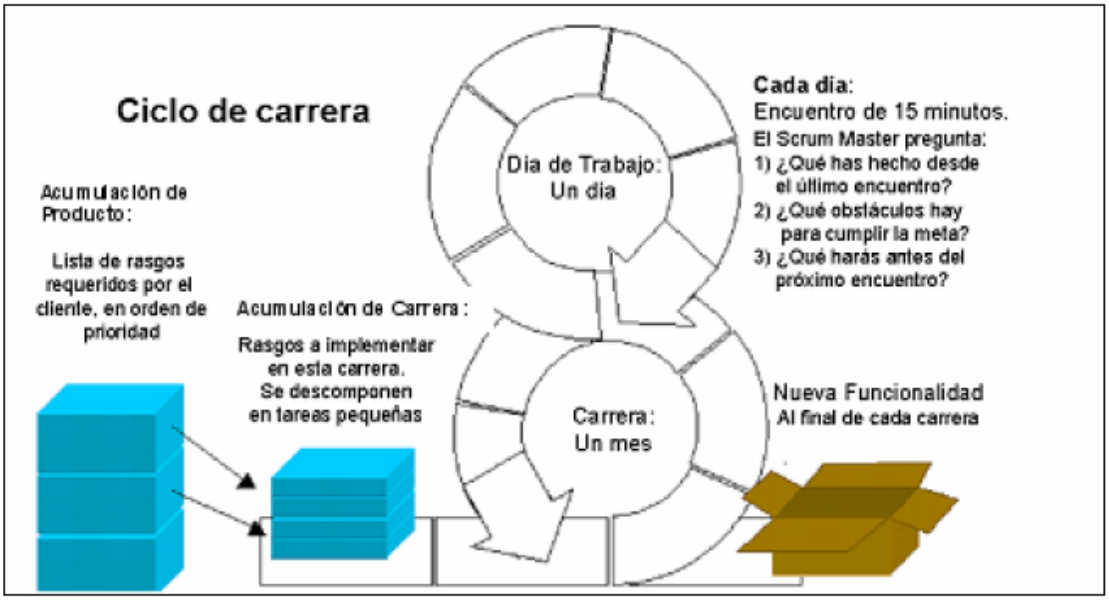


Figura N°7 Ciclo de Carrera o vida (Sprint) de Scrum.

La lista de Acumulación del Producto contiene todos los rasgos, tecnologías, mejoras y lista de bugs, que a medidas que se desenvuelven, constituyen la futura entrega del producto. Los rasgos más urgentes merecerán mayor detalle, los que puedan esperar se tratarán de manera más sumaria. La lista se origina a pedir de una variedad de fuentes. El grupo de mercadeo genera los rasgos o la función; la gente de venta genera elementos que harán que el producto sea más competitivo; los de ingeniería aportarán paquetes que lo harán más robusto; el cliente ingresará debilidades o problemas que deberán resolverse. El propietario de la administración y el control del backlog en productos comerciales bien puede ser el product manager; para desarrollos in – house

podría ser el Project manager o alguien designado por el. Se estima que priorizar adecuadamente una lista de productos puede resultar dificultosa al principio del desarrollo pero deviene más fácil con el correr del tiempo.

Al fin de cada iteración de treinta días hay una demostración a cargo de Scrum Master. Las presentaciones en PowerPoint están prohibidas. En los encuentros diarios, las gallinas deben estar fuera del círculo. Todos tienen que ser puntuales; si alguien llega tarde, se le cobra una multa que se destinará a obras de caridad. Es permitido usar artefactos de los métodos a los que Scrum acompañe, por ejemplo Listas de Riesgos si se utiliza UP, PLenguage si el método es EVO, o los planes de Proyecto Sugeridos en la disciplina de Gestión de Proyectos de Microsoft Solutions Framework. No es legal, en cambio, el uso de instrumentos tales como diagramas PERT, porque éstos parten del supuesto de que las tareas de un proyecto se pueden identificar y ordenar; en Scrum el supuesto dominante es que le desarrollo es semi – caótico, cambiante y tiene demasiado ruido como para que se le aplique un proceso definido.

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

Como ya se ha mencionado antes es muy habitual que Scrum se complemente con XP; en estos casos, Scrum suministra un marco de management basado en patrones organizacionales, mientras XP constituye la práctica de programación, usualmente orientada a objetos (OO) y con fuerte uso de patrones de diseño. Uno de los nombres que se utiliza para esta alianza es XP@Scrum. También son viables los híbridos con otras metodologías ágiles.

Conclusiones Parciales.

- Las Metodologías Ágiles más conocidas como son Scrum y XP - eXtreme Programming.
- XP es una Metodología Ágil centrada en potenciar las relaciones interpersonales como clave para el éxito para el desarrollo de software,

promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

- Scrum define un proceso empírico, iterativo e incremental de desarrollo que intenta obtener ventajas respecto a los procesos definidos (cascada, espiral, prototipo, etc.) mediante la aceptación de la naturaleza caótica del desarrollo de software, y la utilización de prácticas tendientes a manejar la impredictibilidad y el riesgo a niveles aceptables.
- Las Metodologías Ágiles de Desarrollo (AMs) están especialmente indicadas en proyectos con requisitos pocos definidos y cambiantes. Estas presentan diversas ventajas.

Capítulo III: Utilización de XP en un ejemplo práctico.

Introducción:

Este capítulo brinda una guía práctica para la utilización de XP. Nos apoyaremos en la implementación por dos programadores de un software para el reprocesamiento de datos meteorológicos, el desarrollo de este software le servirá a sus desarrolladores igual que para el autor de este trabajo para la defensa de su Tesis de Grado. Lo que se desea es modelar es un contaminante desprendido a la atmósfera a través de una chimenea. Teniendo en cuenta lo ante planteado el problema consiste en como facilitar la entrada de los datos e interpretación de los resultados en el ISCST3.

3.1 Prácticas de XP.

Tomando en cuenta las ventajas [8] que ofrece XP y como bien dice, cuando nos encontramos en la solución de un problema este puede ser implementado en forma parcial es decir (elegir solo alguna de las prácticas), no es necesario adoptar a XP en su forma completa, sino que pueden utilizarse varias de sus prácticas en forma independiente. Esto hace que el costo de su implementación sea mucho más accesible que el de otras metodologías. Es decir solo tomaremos en cuenta las prácticas que serán de utilidad para el desarrollo del problema planteado.

3.1.1 Interacción con el cliente.

Primero observaremos la importancia que tiene la interacción del cliente con el equipo de trabajo. Este no solo apoya al equipo de trabajo sino que podíamos decir que es parte de el, su presencia es capital a la hora de abordar las historias de usuarios y las reuniones de planificación. Además es su tarea realimentar al equipo de desarrolladores después de cada iteración con los problemas con los que se han encontrado. Al mostrar sus prioridades, expresando sus sensaciones. [9]

Como resumen podemos decir que el cliente se encuentra más cercano el proceso de desarrollo. Se elimina la fase inicial de captura de requisitos y se permite que estos se vayan defiendo de una forma ordenada durante el tiempo que dura el proyecto. El cliente puede cambiar de opinión sobre la marcha y a

cambio debe encontrase siempre disponible para resolver dudas del equipo de desarrollo y para detallar los requisitos especificados cuando sea necesario.

Ahora se explican las prácticas de XP que se tuvieron en cuenta para la realización de nuestro ejemplo práctico.

El juego de la planificación. Esta consiste: al reunirse el **programador** con el **cliente**, en este caso sería el diplomante con el jefe de departamento (CEMA), con el objetivo de realizar las historias de usuario que en sí son la captura de requisitos, ya sea funcionales o no. El cliente le plantea al programador cuales serán las tareas que él quiere que se realicen en cada iteración, buscando aportar mayor valor al negocio. Retomamos nuestro ejemplo práctico, en este los requisitos generales del sistema se consideran principalmente gestión y procesamiento de la data de dispersión de contaminantes. También el programador deberá reunirse con el **entrenador** (tutor(es)) que es en sí el que le dará la idea general para la implementación de esta. Después de esto ya debe quedar conformada la historia de usuario.

Entregas Pequeñas: Estas entregas son realizadas por el programador (diplomante) el cual determina el tiempo aproximado de duración para la realización. Para la implementación de la entrega el programador debe reunirse con el **consultor** (especialista en medio ambiente) para suplir su falta de conocimiento con respecto al tema. También en esta parte el cliente juega un papel muy importante ya que él es quien selecciona qué construir de acuerdo con lo que quiere implementar en esta entrega y tomando en cuenta el tiempo de duración que determinó el programador para esta tarea. Después de esto el programador vuelve a retomar el control y entonces realiza la implementación Siempre estas tareas son seguidas por el **Encargado de Seguimiento** (tutor(es)) ya que este verifica las estimaciones realizadas y el tiempo real dedicado para mejorar futuras estimaciones.

Diseño simple: Esta es realizada por el programador en un momento determinado del proyecto y es la solución más simple, es decir es una entrega rápida al cliente pero eficiente, es decir, esta funciona.

Pruebas: Estas pruebas son elaboradas por el cliente en conjunto con el **encargado de pruebas** (tutor(es)) que es el que ayuda al cliente a elaborarlas y

es el responsable de las herramientas para estas. Estas pruebas son ejecutadas constantemente antes cada modificación del sistema.

Cliente in-situ: Esto consiste en la participación constante del **cliente** (Jefe del departamento (CEMA)) en el proyecto, es decir este siempre tiene que estar presente y disponible todo el tiempo para el equipo, esto debe funcionar ya que es uno de los principales factores para el éxito de XP. Que permite esto, que el **programador** (diplomante) siempre tenga a su alcance a la persona que determina lo que se realizará en cada historia de usuario.

Metáforas:

Fuente: Es la fuente de contaminante que se refiere en el problema.

Fichero de entrada: Un fichero que tiene datos meteorológicos.

Fichero de salida: Un fichero que tiene datos meteorológicos procesados.

3.1.2 Ciclo de Desarrollo XP.

Luego de la explicación de las prácticas que serán utilizadas en nuestro ejemplo, veremos como queda el ciclo de desarrollo que se realiza en XP, siguiendo cada una de sus etapas. Para esto también debemos decir que nuestro ejemplo práctico no es una base de datos sino un software implementado en la base de Java. Esto ejemplo se realizará para los requisitos generales del sistema donde se consideran principalmente ***Gestión*** y ***Procesamiento*** de la data de dispersión de contaminantes. [10]

Estos requerimientos funcionales como es el caso de La **Gestión de la data de dispersión de contaminantes** están conformados por una serie de pasos a seguir, aunque en algunos casos no sea necesario darle cumplimiento a todos, pero siempre en el caso que sea (requisito funcional o no) hay que darle cumplimiento a estos. En este caso se tomaron en consideración los siguientes pasos a seguir:

- Insertar la data.
- Guardar la data.
- Editar la data.
- Cargar la data.

El otro requisito por mencionar es **Procesar la data de dispersión de contaminantes** que también es un requisito funcional. Se dieron los siguientes pasos a seguir.

- Crear fichero de entrada del ISCST3.
- Pasarle al ISCST3 el fichero creado.
- Leer la salida del ISCST3.

Como bien se planteó nuestro objetivo no es aplicar tan solo la metodología XP a este proyecto sino que las personas aprendan a través de este ejemplo como aplicarla a un problema que cumpla con los requisitos que esta metodología exige.

Por lo tanto nos centraremos en estos dos requisitos funcionales que por la importancia que se le fue otorgada se consideran generales.

Estos requisitos por el nivel de importancia que tienen en nuestro ejemplo se puede decir que son historias de usuarios, las cuales están compuestas por tareas que en este caso son los pasos a seguir. Estas historias surgen de lo que plantearemos a continuación.

Pero primero observaremos la importancia que tiene hacer correctamente la captura de requisitos, lo que implica que tenemos que tener el conocimiento necesario para realizarlo de una manera correcta (ágil) pero eficiente.

3.1.3 Captura de requisitos.

El proceso de captura de requisitos en XP gira entorno a una lista de características que el cliente desea que existan en el sistema final. Cada una de estas características recibe el nombre de **Historias de Usuarios,** y su definición consta de dos fases [10: p.6]:

1. El cliente describe con sus propias palabras las características y el responsable del equipo de desarrollo le informa de la dificultad técnica de cada una de ellas y por lo tanto de su coste. A través del diálogo resultante el cliente deja por escrito un conjunto de historias y las ordena en función de la prioridad que tiene para él. Entonces ya podemos definir algunos hitos y fechas aproximadas para ellas.
2. Esta consiste en coger las primeras historias que serían implementadas (primera iteración) y dividirlas en las tareas necesarias para llevarlas a cabo. El cliente también participa, pero hay más peso por parte del equipo de

desarrollo, que daría como resultado una planificación más exacta. En cada iteración se repetirá esta segunda fase para las historias planificadas para ella.

3.1.4 Historia de usuario.

El **gestor del proyecto** (doctorante) prevee la reunión entre el programador (diplomante) y el cliente (Jefe del departamento (CEMA)), donde el cliente define el valor del negocio a implementar, es decir la historia de usuario a realizar, luego el programador estima el esfuerzo necesario para su implementación, el decide el tiempo de duración de esta y de las tareas planteadas en la historia de usuario cual será implementada.

Luego de esta reunión ya queda conformada la historia de usuario, que daría de la siguiente manera:

Una historia de usuario en general contiene como ítems los siguientes atributos.Fecha, tipo de actividad (nueva,corrección,mejora), prueba funcional, numero de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción, notas y una lista de seguimiento con las fechas, estado, cosas por terminar y comentarios.

Tabla N°3. Historia de Usuario (Captura de requisitos).

Historia de Usuario	
Número: 1	**Usuario: Jefe de Departamento (CEMA)**
Nombre historia: Gestión de la data de dispersión de contaminantes.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** (Alta / Media / Baja)
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Equipo XP	
Descripción: Debemos encontrar todos los datos que la conformarán en sí, para entonces darle cumplimiento a todos los pasos a seguir.	
Observaciones: *CONFIRMADO con el cliente.*	

Luego el cliente selecciona que construir, de acuerdo con sus prioridades y las restricciones de tiempo. Es decir este tomara la tarea que el desea implementar en esta primera parte de acuerdo con sus necesidades pero siempre tomando

en cuenta la restricción de tiempo puesta por el programador para la implementación de esta tarea. A partir de este momento podemos llamarla tarea(s), varias tareas conforman una historia de usuario. Entonces es que surgen las llamadas tarjetas de tareas, estas son las que se reparten entre los programadores, es decir lo que se modela entre ellos.

Tabla N°4. Tarjeta de Tarea.

	Tarea
Número tarea: 1	**Número historia:** 1
Nombre tarea: Insertar data.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 3
Fecha inicio: 17 octubre 2009	**Fecha fin:** 4 noviembre 2009
Programador responsable: Equipo XP	
Descripción: Comprobar que todos los datos han sido insertados en el fichero correctamente.	

Tabla N°5 Tarjeta de tarea.

	Tarea
Número tarea: 2	**Número historia:** 1
Nombre tarea: Guardar data.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 4
Fecha inicio: 10 octubre 2009	**Fecha fin:** 30 noviembre 2009
Programador responsable: Equipo XP	

Tarea	
Número tarea: 3	**Número historia:** 1
Nombre tarea: Editar data.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 2
Fecha inicio: 2 diciembre 2009	**Fecha fin:** 10 enero 2010
Programador responsable: Equipo XP	
Descripción: Nos permitirá modificar algún dato que se tenga que cambiar, borrar, etc.	

Tabla N°6 Tarjeta de tarea.

Tabla N°7 Tarjeta de tarea.

Tarea	
Número tarea: 4	**Número historia:** 1
Nombre tarea: Cargar data.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 2
Fecha inicio: 2 diciembre 2009	**Fecha fin:** 10 enero 2010
Programador responsable: Equipo XP	
Descripción: Nos permitirá abrir la data una vez guardada.	

Después que se le entrega la tarjeta de tarea al programador (diplomante) entonces este construye ese valor del negocio, es decir implementa la tarea asignada por el cliente según su necesidad. Todo este ciclo realizado para poder obtener la historia de usuario y luego las tarjetas de tareas son lo que constituye en sí un **ciclo de desarrollo.**

Luego para el próximo requisito funcional se debe realizar el mismo proceso lo que cambiaría serian las tarjetas en algunos de sus aspectos ya que no es la misma situación por lo tanto no debe resolverse de la misma manera. Para obtener las tarjetas para el requisito siguiente se deberá realizar nuevamente el ciclo de desarrollo.

Historia de Usuario	
Número: 2	**Usuario: Jefe de Departamento (CEMA)**
Nombre historia: Procesar la data de dispersión de contaminantes.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** (Alta / Media / Baja)
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Equipo XP	
Descripción: Es tomar la data insertada por el usuario y organizarla, etc.	
Observaciones: *CONFIRMADO con el cliente.*	

Tabla N°8 Historia de usuario

Entonces volvemos a dividir esa historia de usuario en tarjetas de tareas, es decir los pasos que conforman este requisito.

Tabla N°9 Tarjeta de tarea.

Tarea	
Número tarea: 1	**Número historia:** 2
Nombre tarea: Crear fichero de entrada del ISCST3.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 3
Fecha inicio: 10 enero 2010	**Fecha fin:** 30 enero 2010
Programador responsable: Equipo XP	
Descripción:	

Tabla N°10 Tarjeta de tarea.

Tarea	
Número tarea: 2	**Número historia:** 2
Nombre tarea: Pasarle al ISCST3 el fichero creado.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 3

Fecha inicio: 31 enero 2010	**Fecha fin:** 15 febrero 2010
Programador responsable: Equipo XP	
Descripción:	

Tabla N°11 Tarjeta de tarea.

	Tarea
Número tarea: 3	**Número historia:** 2
Nombre tarea: Leer La Salida del ISCST3.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 3
Fecha inicio: 20 febrero 2010	**Fecha fin:** 18 marzo 2010
Programador responsable: Equipo XP	
Descripción:	

Con esto logramos tener nuestra primera iteración utilizando la metodología XP. Pero como realmente se planifican estas iteraciones, es decir planificar nuestras primeras entregas al usuario. Para esto debemos seguir una serie de premisas que son las utilizadas en la etapa de planificación de XP. La primordial es que las entregas se hagan cuanto antes y que con cada iteración el cliente reciba una mueva versión. Cuánto más tiempo se tarde en introducir una parte esencial, menos tiempo habrá para trabajar en ella posteriormente. Se aconsejan muchas entregas y muy frecuentes. De esta forma, un error en una parte esencial del sistema se encontrará pronto y por lo tanto se podrá arreglar antes.

Pero lo mejor de todo es que uno a la hora de la planificación se puede equivocar. Es más todo sabemos que lo común es equivocarse y por ello la metodología ya tiene previsto mecanismos de revisión. Por tanto, es normal que cada tres a cinco iteraciones se tenga que revisar las historias de los usuarios y renegociar nuevamente la planificación.

3.2 Tarjeta CRC.

Por último en este proceso tomos la realización de la tarjeta CRC (Clases, Responsabilidades y Colaboración), pero debemos decir que estas son realizadas con el objetivo de facilitar la comunicación y documentar los resultados. Estas permiten que el equipo completo contribuya en la tarea ce diseño. Una tarjeta CRC representa un objeto, por lo tanto es una clase, cuyo nombre se pone en forma de título en la tarjeta, los atributos y las responsabilidades más significativas se colocan a la izquierda y las clases que están implicadas con cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente. Para una mejor comprensión de las mismas decidimos agruparlas por historias de usuarios.

Para la realización de las tarjetas CRC de los requisitos funcionales tomados por nosotros como ejemplo práctico, debemos recordar que no tenemos nada implementado por lo tanto las clases que se verán a continuación serán ideales para representarles de alguna forma como es que se representan.

3.2.1 Gestión de la data de dispersión de contaminantes.

Tabla N°12 Tarjeta CRC.

Nombre de la clase: Gestión _ data	
Tipo de la clase: Lógica del negocio.	
Responsabilidades:	**Colaboradores:**
Insertar la data.	Insert _ Data
Guardar la data.	Guardar _ Data
Editar la data.	Editar _ Data
Cargar la data.	Cargar _ Data

Tabla N°13 Tarjeta CRC.

Nombre de la clase: Insert _ Data	
Tipo de la clase: Utilitaria.	
Responsabilidades:	**Colaboradores:**
Tomar los valores de la data.	

Tabla N°14 Tarjeta CRC.

Nombre de la clase: Guardar _ Data	
Tipo de la clase: Utilitaria	
Responsabilidades:	**Colaboradores:**
Guarda todos los datos de la data.	

Tabla N°15 Tarjeta CRC.

Nombre de la clase: Editar _ Data	
Tipo de la clase: Utilitaria	
Responsabilidades:	**Colaboradores:**
Permitirá cambiar los datos.	

Tabla N°16 Tarjeta CRC.

Nombre de la clase: Cargar _ Data	
Tipo de la clase: Utilitaria.	
Responsabilidades:	**Colaboradores:**
Cargará los datos existentes.	

3.2.2 Procesar la data de dispersión de contaminantes.

Tabla N°17 Tarjeta CRC.

Nombre de la clase: Procesar _ Data	
Tipo de la clase: Lógica del negocio.	
Responsabilidades:	**Colaboradores:**
Crear fichero de entrada del ISCST3.	Crear _ Fichero
Pasarle al ISCST3 el fichero creado.	Pasar _ Fichero
Leer La Salida del ISCST3.	Leer _ Salida

Tabla N°18 Tarjeta CRC.

Nombre de la clase: Crear _ Fichero	
Tipo de la clase: Utilitario.	
Responsabilidades:	**Colaboradores:**
Crear un fichero de entrada del ISCST3.	

Tabla N°19 Tarjeta CRC.

Nombre de la clase: Pasar _fichero	
Tipo de la clase: Utilitaria.	
Responsabilidades:	**Colaboradores:**
Pasarle al ISCST3 el fichero creado.	

Tabla N°20 Tarjeta CRC.

Nombre de la clase: Leer _ Salida	
Tipo de la clase: Utilitario.	
Responsabilidades:	**Colaboradores:**
Leer La Salida del ISCST3.	

Como bien se describió anteriormente más todo lo planteado es lo que conforma en sí la metodología ágil XP. Para realizar un proyecto bajo su forma de desarrollo debemos seguir una serie de pasos de una manera cronológica, donde ellos son los encargados de comunicarte como es el funcionamiento de XP.

Nuestro objetivo fundamental en la tesis es la realización de una guía metodológica para el desarrollo de software pero utilizando la metodología ágil XP. Esta guía quedaría conformada por los siguientes pasos a seguir, es decir ya de una manera explícita.

1. Comprobar que el problema a resolver (proyecto a realizar) cumple con los requisitos para ser desarrollado bajo la teoría de funcionamiento de la metodología ágil XP.

2. Definir correctamente todas las todas las prácticas de XP que pueden ser utilizadas en la realización de nuestro proyecto.
3. Tener una buena interacción con el cliente durante todo el desarrollo del proyecto (convertirlo en otro integrante del quipo de desarrollo)
4. Realizar la captura de requisitos (mediante las historias de usuarios), son realizadas directamente con el cliente.
5. Poder definir de una forma segura cuales son las tareas a realizar por cada historia de usuario definida por el cliente según sus necesidades. Para esto debe realizarse de forma correcta la elaboración de las tarjetas de tareas.
6. Hacer entregas rápidas al cliente pero manteniendo una alta calidad.

3.3 Análisis de la Encuesta.

Primero decir que con la realización de esta encuesta es tener como objetivo fundamental demostrar cual es la metodología más utilizada en el desarrollo de software en nuestro instituto, para esto se realizará una serie de preguntas al claustro de profesores del departamento y a los estudiantes de que cursan en este momento el 5to año de la carrera. Sobre estas personas es que recae el mayor peso en el desarrollo de software en nuestro instituto, ya que son los que realizan una serie de trabajos investigativos y una parte esencial en el desarrollo de estos consiste en la implementación de software. Los objetivos fundamentales que se persiguen con la realización de esta encuesta es obtener información sobre las metodologías ágiles de desarrollo de software como una alternativa metodológica a tener en cuenta en cuanto al desarrollo de software. Si nuestros desarrolladores tienen un basto conocimiento utilizando estas, si conocen sus características principales, etc.:

- Cuantos programadores conocen acerca de las metodologías ágiles (popularidad).
- Cuantas personas utilizan RUP y cuantas metodologías ágiles para el desarrollo de software.
- La opinión que se tiene de RUP en la realización de proyectos donde los equipos de trabajo son bastantes pequeños.

Realizando un análisis por preguntas realizadas a través de la encuesta, obtuvimos los siguientes resultados:

Pregunta 1:

Esta pregunta arrojó que de las 20 personas entrevistadas el 72% aproximadamente ha escuchado mencionar el término metodología ágil para el desarrollo de software. Lo que nos da un resultado satisfactorio entre nuestros desarrolladores ya que un gran número por lo menos ha escuchado el término. Y que el otro 28% restante de los encuestados no ha oído mencionar el término. Lo que demuestra que todavía existe una parte de los desarrolladores que no conocen el término metodología ágil. Por lo que debemos hacer énfasis en el estudio de estas y sus formas de aplicación.

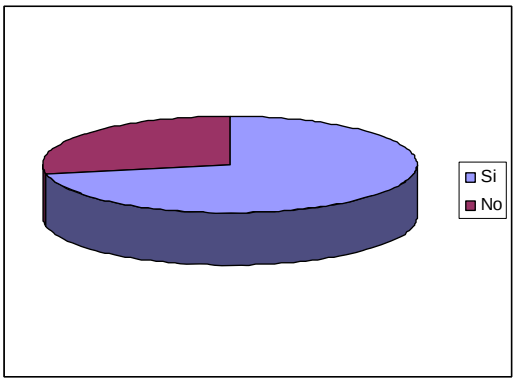


Figura N°8 Gráfico1.

Pregunta 2:

El 67% aproximadamente considera que son útiles en grupos de trabajo pequeño.

Y el resto que solo fue 33% restante no tuvo en consideración ninguna respuesta de las posibles a responder.

Con los resultados obtenidos con la realización de esta pregunta podemos observar que una parte de los encuestados no conoce en que tipo de equipo de trabajo son más eficientes las metodologías ágiles, es decir en los grupos de desarrollo donde se puede implementar software a través de esta metodología.

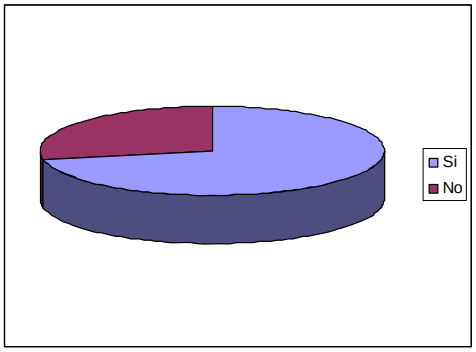


Figura N°9 Gráfico2.

Pregunta 3:

El 33% aproximadamente si considera que las AMs si hacen más énfasis en la arquitectura que las tradicionales.

Un 11% aproximadamente plantea que las ágiles no hacen énfasis en la arquitectura.

El 56% restante aproximadamente no ofreció ninguna respuesta.

Los resultados arrojados por esta pregunta demuestra la falta de conocimiento existente por parte de los desarrolladores con respecto a las metodologías ágiles ya que la menor parte de los encuestados son los que abordaron su respuesta de manera positiva, otra parte pero pequeña respondió de forma incorrecta y la gran mayoría de los encuestados no supo que responder.

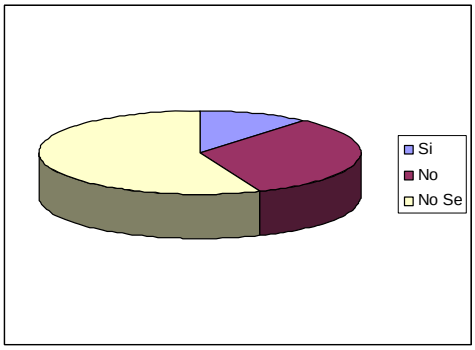


Figura N°10 Gráfico3.

Pregunta 4:

El 72% aproximadamente de los desarrolladores si utiliza RUP.

El 17% aproximadamente de los desarrolladores no utiliza RUP.

El 11% aproximadamente no ofreció ninguna respuesta.

Con esta pregunta se logró conocer que la metodología de desarrollo de software RUP ya forma parte de las herramientas a utilizar por la gran mayoría

de desarrolladores, que solo una parte muy pequeña utiliza otras metodologías y una parte aún más pequeña no abordó respuesta alguna.

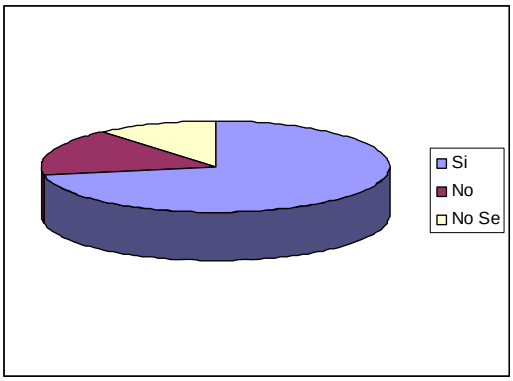


Figura N°11 Gráfico4.

Pregunta 5:

El 17% aproximadamente de los desarrolladores si utiliza las AMs.

El 61% aproximadamente de los desarrolladores no utiliza las AMs.

El 22% aproximadamente no ofreció ninguna respuesta.

Se observa que los desarrolladores no conocen como utilizar las metodologías ágiles aplicadas al desarrollo de software, solo una pequeña parte e los encuestados utiliza las AMs para desarrollar software y que la gran mayoría se inclina por otra(s).

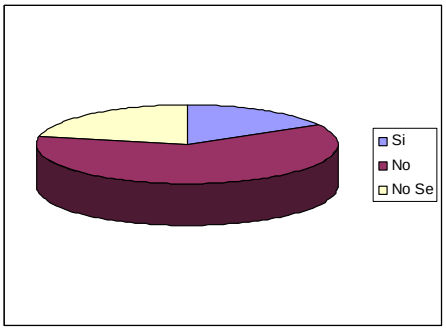


Figura N°12 Gráfico5.

Pregunta 6:

El 6% aproximadamente de los desarrolladores tiene una mala opinión de RUP en equipo de trabajos bastantes pequeños.

El 72% aproximadamente de los desarrolladores tiene una opinión regular de RUP en equipo de trabajos bastantes pequeños.

El 22% aproximadamente de los desarrolladores tiene una buena opinión de RUP en equipo de trabajos bastantes pequeños.

Con esta pregunta se comprobó que la gran mayoría de los encuestados solo desarrollan software a través de RUP y no así con otras metodologías. La gran mayoría tiene una opinión regular con respecto a RUP en grupos de trabajo bastante pequeños pero aún así siguen desarrollando mediante este, no han ampliado sus fronteras en la búsqueda de metodologías más eficientes antes este tipo de situación.

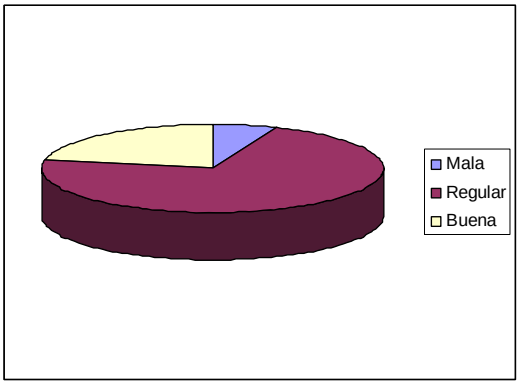


Figura N°13 Gráfico6.

Para ver guía de encuesta revisar anexo 4.

Conclusiones Parciales.

- Con la realización de este capítulo logramos aplicar la metodología ágil XP a un caso de estudio real lo que permitió observar como es el funcionamiento de la misma aplicada al desarrollo de software.
- Nos permitió obtener ya de una manera explícita una guía metodológica para la aplicación de XP en proyectos donde los requisitos son muy cambiantes, los equipos de trabajo son generalmente pequeños y se exige reducir drásticamente los tiempos de entrega pero manteniendo una alta calidad.

Conclusiones Generales.

- En el presente trabajo se ha propuesto la utilización de Las Metodologías Ágiles en el proceso de desarrollo de software y la obtención de una guía metodológica para la utilización de XP aplicada al desarrollo de software.
- Para la obtención de nuestros objetivos desarrollamos conocimiento con respecto a las metodologías ágiles: sus etapas, sus ciclos de vida, sus principales características.
- Logramos obtener la guía metodológica para el uso de la metodología ágil XP aplicada al desarrollo de software pero en entornos donde los requisitos son muy volátiles (cambiantes), los equipos e trabajo son bien reducidos (pequeños) y sobre todo se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una lata calidad.

Recomendaciones.

Una vez concluido el desarrollo de este trabajo investigativo se recomienda:

- La utilización de la Metodología XP en los procesos de desarrollo de software que sean de corto plazo, con un equipo de desarrollo bastante pequeño.
- Realizar un estudio análisis más completo de la fase Implementación y Prueba, las cuales no fueron abordadas profundamente en este trabajo.
- Dar seguimiento a las investigaciones de metodologías de desarrollo de software que se puedan utilizar en los procesos de desarrollo de software en nuestro país ya que cada software tiene características diferentes y otras metodologías podrían brindar para cierto software mayor eficiencia y la obtención de productos con mayor calidad.

Referencias Bibliográficas.

[1] ¿Es RUP un método ágil? Disponible en: **http://www.metodologiatradiconal.com**

[2] Letelier, P. Metodologías Ágiles y XP. Disponible en: www.disc.upr/~letelier/pub´

[3] Canós, J. et al. M. Metodologías Ágiles en el desarrollo de software. Universidad Politécnica de Valencia. Valencia, 2003.

[4] Welicki, L. Implementando Extreme Programming en la plataforma .NET. Disponible en: http://www.programacionextrema./newMethodology.es.html

[5] Las metodologías. **Disponible en:** http://www.nuevametodologia/5/#nuevametodologia_meto_xp

[6] Calderón, A. *Metodologías Ágiles*. Disponible en: http://www.cyta.com. agilar/ta0502/v5n2a1.htm

[7] Beck, Kent, Extreme Programming Explained, Addison-Wesley The XP Series, 2000. Disponible en: http://legnita.wordpress.com/series/project-nagement/metodologias-agilesxp/

[8] Pérez, J. *Metodologías Ágiles: La ventaja competitiva de estar preparado para tomar decisiones lo más tarde posible y cambiarlas en cualquier momento.* Disponible en: http://www.spinec.org/wp-content/metodologiasagiles_01.pdf

[9] Programación extrema, software libre y aplicabilidad. Disponible en: Http: willydev.net//insiteCreation/v1.0descargas/artículos/general/xplibreap.a.spx

[10] Orjela, A. las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería del Software Educativo. Mayo 2008 . Disponible en : http://redalyc.uaemex.mx/redalyc/pdf/849/84934064.pdf

Bibliografía.

Beck, Kent, Extreme Programming Explained, Addison-Wesley The XP Series, 2000. Disponible en: http://legnita.wordpress.com/series/project-nagement/metodologias-agilesxp/

Calderón, A. *Metodologías Ágiles*. Disponible en: http://www.cyta.com. agilar/ta0502/v5n2a1.htm

Canós, J. et al. M. Metodologías Ágiles en el desarrollo de software. Universidad Politécnica de Valencia. Valencia, 2003.

¿Es RUP un método ágil? Disponible en: **http://www.metodologiatradiconal.com**

Las metodologías. **Disponible en:** http://www.nuevametodologia/5/#nuevametodologia_meto_xp

Letelier, P. Metodologías Ágiles y XP. Disponible en: www.disc.upr/~letelier/pub´

Rodríguez Corbea M, Ordóñez Pérez M, *La Metodología XP aplicable al desarrollo del software educativo en Cuba.* Yaneisis Pérez Heredia (tutor). Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Orjela, A. las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería del Software Educativo. Mayo 2008. Disponible en: http://redalyc.uaemex.mx/redalyc/pdf/849/84934064.pdf

Pérez, J. *Metodologías Ágiles: La ventaja competitiva de estar preparado para tomar decisiones lo más tarde posible y cambiarlas en cualquier momento.* Disponible en: http://www.spinec.org/wp-content/metodologiasagiles_01.pdf

Programación extrema, software libre y aplicabilidad. Disponible en:

Http: willydev.net//insiteCreation/v1.0descargas/artículos/general/xplibreap.a.spx

Welicki, L. Implementando Extreme Programming en la plataforma .NET. Disponible en: http://www.programacionextrema./newMethodology.es.html

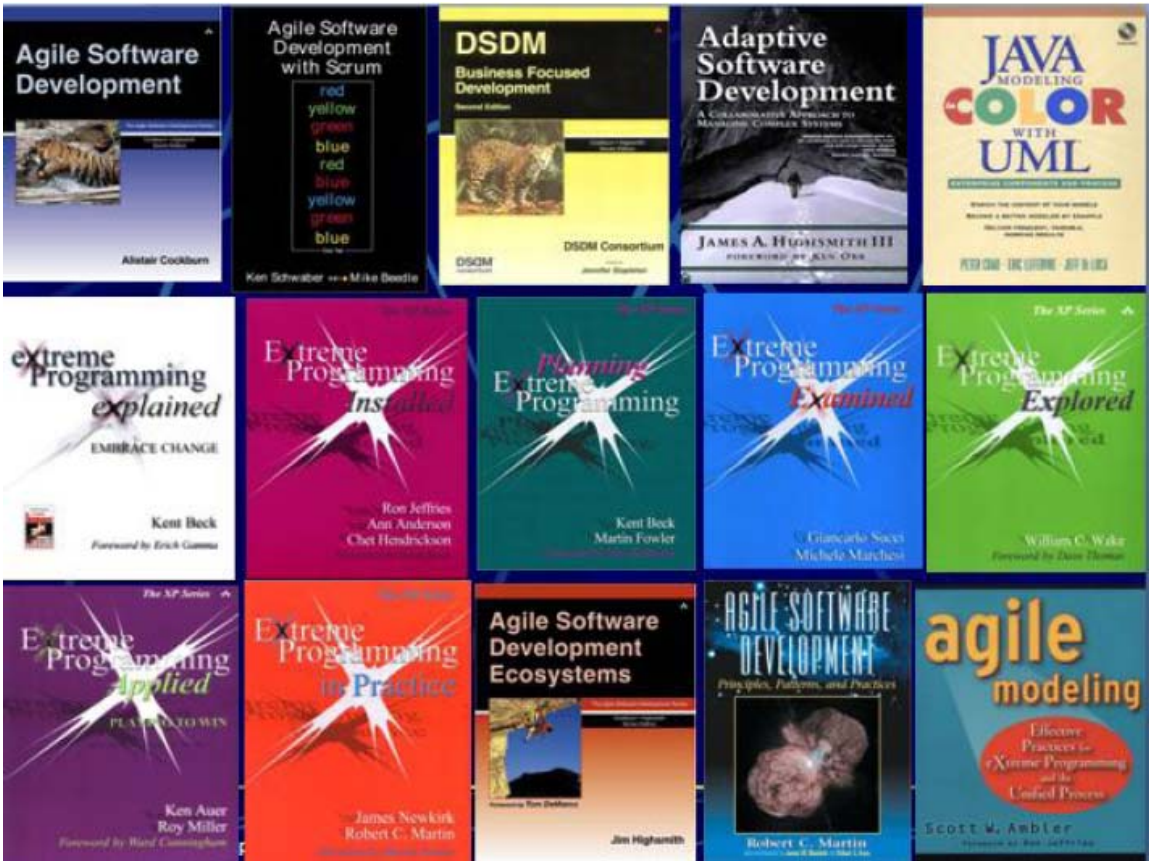
Anexos.

Anexo 1:
Un día de trabajo con XP.

Anexo 2:
Como funciona la programación en parejas.

Anexo 3:

Libros Metodologías Ágiles de desarrollo, específicamente XP.



Anexo 4:

Para obtener toda la información necesaria se realizaron una serie de preguntas con el propósito de alcanzar nuestros objetivos principales perseguidos por nuestra encuesta.

Las preguntas a tener en cuenta fueron:

1) ¿Has oído mencionar el término Metodologías Ágiles para el desarrollo de software?
2) ¿Las Metodologías Ágiles son útiles en grupos de desarrollo pequeños (menos de diez integrantes)?
3) ¿Las Metodologías Ágiles de desarrollo de software hacen más énfasis en la arquitectura que las Tradicionales?
4) ¿Utilizas RUP?
5) ¿Utilizas Metodologías Ágiles?
6) ¿Qué opinión tienes de RUP en la realización de proyectos donde los equipos de trabajo son bastante pequeños?

Glosario.

Software: Es la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo.

Ingeniería de Software: Es una tecnología multicapa en la que, se pueden identificar: los métodos (indican cómo construir técnicamente el software), el proceso (es el fundamento de la Ingeniería de Software, es la unión que mantiene juntas las capas de la tecnología) y las herramientas (soporte automático o semiautomático para el proceso y los métodos).

Metodologías: Se refiere a los métodos de investigación en una ciencia. Se entiende como la parte del proceso de investigación que permite sistematizar los métodos y las técnicas necesarios para llevarla a cabo. Define Quién debe hacer, Qué, Cuándo y Cómo debe hacer.

Metodologías de Desarrollo: Se define como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información.

Metodología Ágil: Constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración.

IBM: International Business Machines Corporation o IBM, conocida coloquialmente como el Gigante Azul, es una empresa que fabrica y comercializa hardware, software y servicios relacionados con la informática. Está constituida desde el 15 de junio de 1911, pero lleva operando desde 1888.

UML: Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software, se usa para detallar los artefactos en el sistema, para documentar y construir.

RUP: El Proceso Unificado Racional o RUP (Rational Unified Process), es un proceso desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Se

caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiados por los caos de usos. Incluye artefactos y roles.

Programación Extrema(XP): Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

Scrum: Define un marco para la gestión de proyectos. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones y la segunda característica importante son las reuniones a lo largo proyecto.

Crystal Methodologies: Conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. Se estableció una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).

Dynamic Systems Development Method (DSDM): Define el marco para desarrollar un proceso de producción de software. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases.

Metáfora: Equivalente de la arquitectura en el universo de XP. La metáfora guía al desarrollo proveyendo de integridad conceptual al diseño y comunicando a todos los involucrados los elementos básicos de la solución y sus relaciones.

Refactoring: Técnica que mantiene intacto el funcionamiento del software mejorando la estructura interna del mismo.