UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
Facultad 9

LA METODOLOGÍA XP APLICABLE AL DESARROLLO DEL SOFTWARE EDUCATIVO EN CUBA

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores: Maite Rodríguez Corbea
Meylin Ordóñez Pérez

Tutor: Ing. Yaneisis Pérez Heredia

Co-tutor: Dtor. Ernesto González

Ciudad de la Habana, julio, 2007
"Año 49 de la Revolución"

AGRADECIMIENTOS

De Maite

Agradezco a mi madre por ser siempre la luz que guía mis pasos, a mi padre por darme todo su apoyo, a mi querida abuela por sus valiosos consejos, a toda mi familia en general y especialmente a Carmen, Héctor, Marileidis, Andrea, Susana.
Agradezco a todos mis amigos, a los buenos, a los de verdad…a Denis por esa paciencia que siempre me tuvo, a Susan por ser mi ángel de la guarda, a Lianna, mi querida compañera de cuarto que me soportó durante todo este tiempo y por supuesto agradezco a mi compañera de tesis Meylin, la mejor colega que he tenido.
Agradezco a mi tutora por aguantar mis ataques de histeria. A Fidel Castro, que me dio la posibilidad de estar hoy aquí y a todas esas personas que me ayudaron a hacer esta larga travesía, son muchos, pero a todos los tengo en mi corazón.

De Meylin

Agradezco a mis padres por apoyarme en todo momento, mi madre por su comprensión y dedicación a contribuido en la finalización de mis estudios
y mi padre por ser aquella persona la cual siempre he llevado como ejemplo y como guía para lograr todas y cada una de mis metas.
A mi novio Amado por su amor, por ser de gran apoyo y sustento en mi vida. A todas mis tías y a mi suegra Maritza.
A toda mi familia, a nuestro Comandante Fidel Castro**,** a mi tutora Yaneisis y a todos mis amigos**,** en especial a esos que siempre me apoyaron: a Maite mi super colega, Yaima , Susan, Carmen, Cesar, Silenay, Denys

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de Julio del 2007.

Maité Rodríguez Corbea	Meylin Ordoñez Pérez	Yaneisis Pérez Heredia
Autora	Autora	Tutora

OPINIÓN DEL TUTOR

Como tutora del Trabajo de Diploma “La Metodología XP aplicable al desarrollo del Software Educativo en Cuba”, luego de haber culminado la realización del mismo, considero que las autoras Meylín Ordóñez Pérez y Maite Rodríguez Corbea, han desarrollado un conjunto de habilidades que les permitirán darle solución adecuadamente a cualquier tipo de necesidad de informatización que se presente en su vida profesional.

Durante la realización del presente trabajo las estudiantes han demostrado un alto grado de responsabilidad ante el cumplimiento en tiempo de las tareas que se les programaron. Han trabajado organizadamente dando muestras de poseer responsabilidad y compromiso en la realización de su tesis. Su desempeño ha manifestado que han desarrollado un valioso nivel de asimilación de nuevas metodologías, llegando a alcanzar un profundo conocimiento y una gran capacidad para la toma de decisiones correctas.

La originalidad, la elegancia en el trabajo y la independencia, han sido cualidades dignas de destacar a lo largo de la realización del trabajo realizado. Las estudiantes manifestaron laboriosidad a lo largo del cumplimiento de las tareas programadas, logrando resultados satisfactorios.

Por otra parte, el elemento investigativo del documento, estuvo desde el inicio muy bien orientado y estructurado, basado en una gran cantidad de bibliografía actualizada. Cada contenido se ha expuesto con claridad y aporta grandes conocimientos al lector.

Por todo lo anteriormente planteado, considero que las diplomantes están aptas para ejercer como Ingenieros Informáticos; y propongo al tribunal que se le otorgue al Trabajo de Diploma la máxima calificación.

RESUMEN

Con el desarrollo de este trabajo se propone lograr una mayor eficiencia en el desarrollo de software educativo en nuestro país, a través del uso de la metodología propuesta.
Se realiza un estudio de la actualidad cubana sobre el desarrollo de software educativo con el objetivo de satisfacer las necesidades del mismo como medio de enseñanza-aprendizaje en la escuela cubana.
Se hace un estudio exhaustivo de la Metodología XP (Programación Extrema) y a partir de la existencia de un caso de estudio, se obtienen resultados que demuestran que es eficientemente realizable.
Estructuración del contenido con una breve explicación de sus partes.
En el Capitulo 1 se hace referencia a los conceptos fundamentales que sustentan la base de los capítulos restantes.
En el Capitulo 2 se realiza un estudio profundo sobre la Metodología XP, donde se hace énfasis en las cuatro fases que la integran y se describen todos los aspectos fundamentales que permiten la total comprensión de cómo es que funciona dicha metodología.
En el Capitulo 3 se aplica todo lo planteado en el capitulo 2 al caso de estudio Topografía.
Por ultimo las Conclusiones ponen a consideración las valoraciones sobre la metodología estudiada, criterio sobre la robustez y eficiencia en la organización del proceso de Ingeniería de software aplicada a Software Educativo y se analiza el cumplimiento o no del objetivo propuesto.

PALABRAS CLAVES

Proceso de Software, Ingeniería de Software, Software Educativo, Multimedia, Metodologías de desarrollo, Metodologías Ágiles**,** Programación Extrema, Pruebas Funcionales XP, Pruebas de aceptación XP, Pruebas Unitarias XP, Prácticas XP, Historias Usuario XP, Plan Entrega XP, Plan de Iteración XP.

INDICE DE TABLAS

INDICE DE FIGURAS

INDICE

INTRODUCCIÓN

El siglo XXI se enfrenta a la creciente implantación de la sociedad del conocimiento. La era del conocimiento en que vivimos no solo está cambiando la sociedad en sí misma, sino que los nuevos modelos de negocios requieren la reformación de nuevos conceptos. La ingeniería de software no es una excepción y por ello requiere no solo una actualización de conceptos, sino también una comprensión y una formulación del nuevo conocimiento existente en torno a las nuevas innovaciones y teorías de dicha disciplina.

Utilizar la informática como apoyo a los procesos de enseñanza/aprendizaje ha sido una problemática que durante mucho tiempo ha sido investigada y probada por muchas instituciones y docentes. Su asimilación dentro de instituciones educativas, ha aumentado en los últimos años, con lo que la demanda de software educativo (SWE) de calidad es cada vez mayor.

En el presente trabajo se eligió la metodología XP para hacer un estudio profundo de la misma y proponer su aplicación en el desarrollo de SWE que cumpla con las condiciones requeridas para la aplicación de la misma.

XP (eXtreme Programing) nace como nueva disciplina de desarrollo de software en el año 1996, y ha causado un gran revuelo entre el colectivo de programadores del mundo. Surge de la mente de Kent Beck, tomando ideas recopiladas junto a Ward Cunningham, y utilizando conceptos como el de Chief Programmer creado por IBM en la década de los ’70. Esta metodología pretende que el desarrollo de un proyecto de software sea un proceso de desarrollo ágil, aunque disciplinado, y aporte soluciones sencillas.

Nuestro país en los últimos años se ha visto enfrascado en una revolución tecnológica donde la esfera de la educación no ha estado exenta en toda una serie de inversiones que ha realizado el Estado Cubano en equipos y tecnología. Se han instalado en todas las escuelas computadoras y otros equipos para que se haga uso de las Tecnologías de la Informática y las Comunicaciones (TIC) en los procesos de enseñanza-aprendizaje, para que sea más ameno y divertido

El Departamento Nacional de Software Educativo del Ministerio de Educación que forma parte de la Dirección de Computación Educacional tiene dentro de sus funciones dirigir, coordinar, organizar y controlar la producción e introducción en la práctica de software educativo para la escuela cubana, desde el ámbito de la red de Centros de Estudio de Software Educativo de los Institutos Superiores Pedagógicos, bajo un esquema de investigación-producción.

Se han creado proyectos de desarrollo de software en los Institutos Superiores Pedagógicos donde trabajan Master en Ciencias y Licenciados junto a programadores y técnicos en Ciencias de la Informática entre otros especialistas, con el objetivo de satisfacer las necesidades de SWE como medio de enseñanza-aprendizaje en la escuela cubana.

Como resultado de estos proyectos en los últimos años se han desarrollado más de 78 productos, otros se están desarrollando y algunos están en proyecto para un futuro no muy lejano. Nuestras escuelas cuentan con una diversidad de SWE que los alumnos y docentes utilizan de forma satisfactoria en su preparación.

Actualmente se trabaja con “Multisaber”, colección que tiene un enfoque curricular y multidisciplinario por su relación con los contenidos de los programas de cada asignatura del currículo de estudio de la Educación Primaria.

”El Navegante”, colección de 10 software para la Educación Secundaria.

“Futuro”, colección de 19 software para la Educación Preuniversitaria.

“Jugar y Aprender”, este nuevo software es el resultado de un proyecto de investigación que se lleva a cabo en nuestro país por un equipo de especialistas relacionado con la inclusión de la computación en las edades preescolares.

Además se han desarrollado varios software monotemáticos de diferentes educaciones que son utilizados según el tema específico para el cual fueron diseñados e implementados.

Estos proyectos no hacen uso de una metodología que guíe el proceso de desarrollo que en ellos se realiza y si analizamos que un proceso de desarrollo de software requiere por

un lado un conjunto de conceptos, una metodología y un lenguaje propio. Podemos decir que la metodología es parte fundamental dentro del proceso; pero como cada software puede tener sus características específicas, tenemos que elegir para cada tipo de software la metodología que nos brinde mejor eficiencia en su desarrollo.

Este trabajo es un estudio profundo de la metodología XP por lo que se pretende resolver la siguiente problemática:

¿Cómo lograr mayor eficiencia en el desarrollo de software educativo a través del uso de la metodología XP al contexto productivo cubano?

Como objeto de estudio de esta investigación se planteó el proceso de desarrollo de software en Cuba.

Para esto se hizo necesario profundizar en el estudio del campo de acción, metodología de desarrollo de Software Educativo en el país.

Se ha realizado un profundo análisis del tema y el objetivo fundamental que se ha propuesto es el siguiente:

- Aplicar en el proceso de desarrollo del Software Educativo ¨Topografía¨ la Metodología XP.

Idea a defender:

La utilización de la metodología XP para el desarrollo productivo del SWE Topografía en nuestro país producirá una mejora en el proceso de desarrollo del Software Educativo.

Para dar cumplimiento al objetivo planteado se ha propuesto cumplir las siguientes tareas:

1. Estudio de las metodologías internacionales para el desarrollo de software educativo.
2. Estudio del software educativo a nivel internacional.
3. Estudio de las condiciones actuales en la cuales se desarrolla el software educativo en Cuba.

4. Estudio de la metodología XP.
5. Aplicar la metodología XP a un caso de prueba “Topografía”.

Algunos de los métodos de investigación que se utilizan son:

Teórico:

- El método Entrevista para conocer acerca de la existencia de un proyecto que se realiza en la universidad, el cual utiliza la Metodología XP.

Empírico:

- El Método Histórico para investigar la existencia de las distintas metodologías existentes para el desarrollo de software educativo en nuestro país.
- El método de análisis para comprender los fundamentos de la metodología y el método de síntesis para poder describir y resumir las características esenciales en cada etapa que conforman la misma.

CAPÍTULO 1

FUNDAMENTACIÓN DEL TEMA.

«Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar.» **Kent Beck.**

En el presente capítulo se describe de forma general los aspectos relacionados con el objeto de estudio y el campo de acción en que se trabaja. Este capítulo constituye la base teórica para la comprensión del trabajo que se desarrolla y los principales aspectos que se abordan son:

- Proceso de Ingeniería de Software.
- Metodologías de desarrollo.
- Proceso de desarrollo de software educativo.
- Clasificaciones de software educativo

1.1 Proceso de Ingeniería de Software.

1.1.1 Software.

Se denomina software, programática, equipamiento lógico o soporte lógico al conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware). Esto incluye aplicaciones informáticas tales como un procesador de textos y software de sistema como un sistema operativo, facilitando la interacción con los componentes físicos y el resto de aplicaciones. (PRESSMAN. 2006)

Probablemente la definición más formal de software es la atribuida a la IEEE : «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». Bajo esta definición, el concepto de software va más allá de los programas de cómputo en sus distintas formas: código fuente, binario o ejecutable, además de su documentación: es decir, todo lo intangible.

El término «software» fue usado por primera vez en este sentido por John W. Tukey en 1957. En las ciencias de la computación y la ingeniería de software, el software es toda la información procesada por los sistemas informáticos: programas y datos. La teoría que forma la base de la mayor parte del software moderno fue propuesta por vez primera por Alan Turing en su ensayo de 1936, los números computables, con una aplicación al problema de decisión.

1.1.2 Ingeniería de Software.

La Ingeniería de Software es una tecnología multicapa en la que, según Pressman, 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).

Figura 1: Representación de Ingeniería de Software

La Ingeniería de Software trata con áreas muy diversas de la informática y de las ciencias de la computación , tales como construcción de compiladores , sistemas operativos o desarrollos en Intranet \ Internet aplicables a una infinidad de áreas tales como : investigaciones científicas , medicina , producción, logística, banca , control de tráfico , metereología, la red de redes Internet, redes Intranet y Extranet. (PRESSMAN. 2006)

1.1.3 Proceso de Ingeniería de Software.

El proceso de ingeniería de software se define como "un conjunto de etapas parcialmente ordenadas con la intención de logra un objetivo, en este caso, la obtención de un producto de software de calidad".(JACOBSON 1998)

El proceso de desarrollo de software "es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo".(JACOBSON 1998)

El proceso de desarrollo de software requiere de un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del software que comprende cuatro grandes fases: concepción, elaboración, construcción y transición. La concepción define el alcance del proyecto y desarrolla un caso de negocio. La elaboración define un plan del proyecto, especifica las características y fundamenta la arquitectura. La construcción crea el producto y la transición transfiere el producto a los usuarios.

Sintéticamente puede describirse como: la obtención de los requisitos del software, el diseño del sistema de software (diseño preliminar y diseño detallado), la implementación, las pruebas, la instalación, el mantenimiento y la ampliación o actualización del sistema.

1.1.4 Proceso de Ingeniería de Software Educativo.

Según Alvaro Galvis de la Universidad de los Andes, es muy coherente y pertinente considerar la producción de software educativo dentro de la Ingeniería de software.

¿Qué tan valida es la propuesta de Alvaro Galvis, donde nos habla de la ingeniería del software, se puede considerar una Ingeniería de software educativo aparte de la Ingeniería del Software o es un caso particular donde el análisis del dominio del problema es educativo?

Se considera que la Ingeniería del Software educativo es una adecuación al área de las ciencias informáticas enfocadas a la creación de software que facilite el proceso de aprendizaje-enseñanza como herramienta pedagógica. En el proceso de software educativo se tiene como uno de las fundamentales tareas la inclusión de los aspectos pedagógicos, siguiendo las pautas de las teorías educativas y de la comunicación subyacente.

1.2 Metodologías de desarrollo de software.

En un proyecto de desarrollo de software la metodología define Quién debe hacer, Qué, Cuándo y Cómo debe hacerlo. La metodología constituye la columna vertebral del proceso de desarrollo de software. No existe una metodología universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable.(MARQUÈS 1995)

Hay que destacar que el proceso de elaboración del software educativo no es un proceso lineal, sino iterativo: en determinados momentos de la realización se comprueba el funcionamiento, el resultado, se evalúa el producto... y frecuentemente se detecta la conveniencia de introducir cambios.

1.2.1 Concepto de metodología de desarrollo.

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. Los métodos son vías que facilitan el descubrimiento de conocimientos seguros y confiables para solucionar los problemas que la vida nos plantea.

La metodología dependerá de los postulados que el investigador considere como válidos; de aquello que considere objeto de la ciencia y conocimiento científico, pues será a través de la acción metodológica como recolecte, ordene y analice la realidad estudiada. Es pues, una etapa, una parte del proceso.

Las metodologías son el camino a seguir para desarrollar software de una manera sistemática, las metodologías persiguen tres necesidades principales:

- Mejores aplicaciones, conducentes a una mejor calidad.
- Un proceso de desarrollo controlado.
- Un proceso normalizado en una organización, no dependiente del personal.

En cuanto al concepto de metodologías de desarrollo varios autores han tratado el tema:

Se define metodología como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información. (MADDISON 1983)

Se llega a la definición de metodología de desarrollo como “un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar nuevo software”. (PIATTINI 1996)

1.2.2 Características de las metodologías de desarrollo.

Cada metodología tiene características particulares pero de forma general se puede enumerar una serie de características que debe tener la metodología y que influirán en el entorno de desarrollo: (PIATTINI 1996)

Reglas predefinidas

- Determinación de los pasos del ciclo de vida.
- Verificaciones en cada etapa.
- Planificación y control.
- Comunicación efectiva entre desarrolladores y usuarios.
- Flexibilidad: aplicación en un amplio espectro de casos.
- De fácil comprensión.
- Soporte de herramientas automatizadas.
- Que permita definir mediciones que indiquen mejoras.
- Que permita modificaciones.
- Que soporte reusabilidad del software

1.2.3 Elementos a tener en cuenta para elegir o construir una metodología.

En el momento de adoptar un estándar o construir una metodología, se han de considerar unos requisitos deseables, por lo que seguidamente se proponen una serie de criterios de evaluación de dichos requisitos.

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

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

- Investigación.

- Análisis de requisitos.

- Diseño.

- Implementación.

- Prueba.

- Instalación y Mantenimiento.

- Ampliación o actualización del sistema.

3. La metodología debe integrar las distintas fases del ciclo de desarrollo.

- Rastreabilidad. Es importante poder referirse a otras fases de un proyecto y fusionarlo con las fases previas. Es importante poder moverse no sólo hacia adelante en el ciclo de vida, sino hacia atrás de forma que se pueda comprobar el trabajo realizado y se puedan efectuar correcciones.

- Fácil interacción entre las fases del ciclo de desarrollo. Es necesaria una validación formal de cada fase antes de pasar a la siguiente. La información que se pierde en una fase determinada queda perdida para siempre, con un impacto en el sistema resultante.

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

5. La metodología debe soportar la determinación de la exactitud del sistema a través del ciclo de desarrollo.
La exactitud del sistema implica muchos asuntos, incluyendo la correspondencia entre el sistema y sus especificaciones, así como que el sistema cumple con las necesidades del usuario. Por ejemplo, los métodos usados para análisis y especificación del sistema deberían colaborar a terminar con el problema del entendimiento entre los informáticos,

los usuarios, y otras partes implicadas. Esto implica una comunicación entre usuario y técnico, amigable y sencilla, exenta de consideraciones técnicas.

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

7. La metodología debe funcionar en un entorno dinámico orientado al usuario. A lo largo de todo el ciclo de vida del desarrollo se debe producir una transferencia de conocimientos hacia el usuario. La clave del éxito es que todas las partes implicadas han de intercambiar información libremente. La participación del usuario es de importancia vital debido a que sus necesidades evolucionan constantemente. Por otra parte la adquisición de conocimientos del usuario la permitirá la toma de decisiones correctas. Para involucrar al usuario en el análisis, diseño y administración de datos, es aconsejable el empleo de técnicas estructuradas lo más sencillas posible. Para esto, es esencial contar una buena técnica de diagramación.

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

9. La metodología debe poder emplearse en un entorno amplio de proyectos software.

- **Variedad**. Una empresa deberá adoptar una metodología que sea útil para un gran número de sistemas que vaya a construir. Por esta razón no es práctico adoptar varias metodologías en una misma empresa.

- **Tamaño, vida**. Las metodologías deberán ser capaces de abordar sistemas de distintos tamaños y rangos de vida.

- **Complejidad**. La metodología debe servir para sistemas de distinta complejidad, es decir puede abarcar un departamento, varios de departamentos o varias empresas.

- **Entorno**. La metodología debe servir con independencia de la tecnología disponible en la empresa.

10. La metodología se debe de poder enseñar.
En una organización sencilla, serán muchas las personas que la van a utilizar, incluso los que se incorporen posteriormente a la empresa. Cada persona debe entender las técnicas específicas de la metodología, los procedimientos organizativos y de gestión que la hacen efectiva, las herramientas automatizadas que soportan la metodología y las motivaciones que subyacen en ella.
11. La metodología debe estar soportada por herramientas CASE.
La metodología debe estar soportada por herramientas automatizadas que mejoren la productividad, tanto del ingeniero de software en particular, como la del desarrollo en general.
El uso de estas herramientas reduce el número de personas requeridas y la sobrecarga de comunicación, además de ayudar a producir especificaciones y diseños con menos errores, más fáciles de probar, modificar y usar.

12. La metodología debe soportar la eventual evolución del sistema.
Normalmente durante su tiempo de vida los sistemas tienen muchas versiones, pudiendo durar incluso más de 10 años. Existen herramientas CASE para la gestión de la configuración y otras denominadas "Ingeniería inversa" para ayudar en el mantenimiento de los sistemas no estructurados, permitiendo estructurar los componentes de éstos facilitando así su mantenimiento.

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

1.2.4 Que metodología es la correcta.

Cuando una empresa encara proyectos de desarrollo de software, ¿que la impulsa a seleccionar una metodología?

La elección respecto a la utilización o no de una determinada metodología depende principalmente del grado de predictibilidad que se desee tener en el desarrollo.

Si una empresa está interesada en que cada proyecto sea como una aventura, en que se desconozca el resultado y existan una gran cantidad de riesgos, no necesitará proceso alguno. Por otro lado, existen organizaciones que poseen manuales de procedimientos para cada una de las tareas relacionadas con el desarrollo de software.

Al analizar estos dos enfoques de desarrollo de software se puede decir que existen dos extremos : el de *Burocracia*, representado por las organizaciones con procesos rígidos y definidos hasta el más mínimo nivel de detalles, y el de *Adhocracia*, que representa el desarrollo caótico sin ningún proceso ni visibilidad sobre el estado y el rumbo de los proyectos. En un extremo, se está en la claridad cegante de un proceso tan pautado que termina sin satisfacer las necesidades de los usuarios, y en el otro se está en la oscuridad completa y aleatoriedad total que deviene del caos permitiendo únicamente beneficios o pérdidas a corto plazo.

Al trazar una línea uniendo la *Burocracia* con la *Adhocracia* se puede ubicar el grado de formalidad dado por las metodologías ágiles. Es decir, una organización que se identifique con uno de estos enfoques, cuáles serían las metodologías que elegiría para el desarrollo de software.

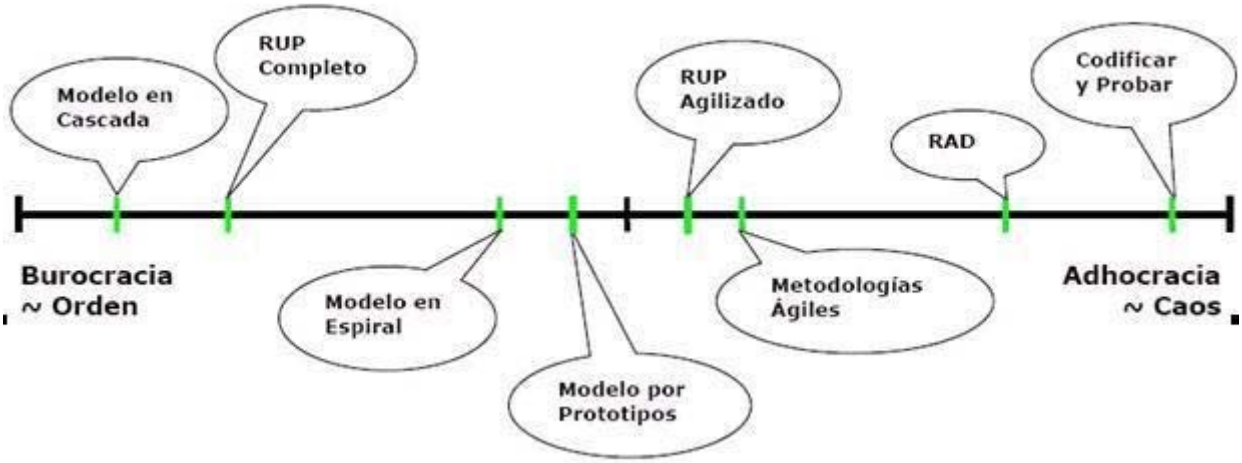


Figura 2: Representación de las distintas metodologías

Desde el punto de vista hacia el Caos se encuentra el tema de las Metodologías ágiles, las cuales reconocen las características inherentes de complejidad del software y el carácter empírico que debe tener un proceso de desarrollo del mismo.

Los procesos más burocráticos tienden a tener definidos todos los aspectos del desarrollo. Esto se contradice con la naturaleza compleja del software, para lo cual se adaptaría con mayor eficiencia un proceso empírico. Por esto se hace referencia a un proceso en que se desconocen muchos aspectos, se reconocen a las personas como componentes de primer orden totalmente impredecibles, y simplemente se intenta guiar el desarrollo hacia un objetivo que puede no permanecer constante en el tiempo a medida que aumenta el conocimiento de la aplicación a ser construida.(HERNÁN. 2004)

En los procesos empíricos se conocen las buenas prácticas probadas con la experiencia, y los resultados que surgen generalmente de tomar ciertas entradas o de realizar ciertas actividades. Estos modelos surgen de la observación y la experiencia obtenida a lo largo de los años, sin basarse en leyes de la naturaleza o principios teóricos fundamentados. Es por esto que se les asocia a un enfoque pragmático para desarrollar software.(HERNÁN. 2004)

1.2.5 Metodologías Ágiles de Desarrollo.

A principios de la década del ’90, surgió un enfoque que fue bastante revolucionario para su momento ya que iba en contra de la creencia de que mediante procesos altamente definidos se iba a lograr obtener software en tiempo, costo y con la requerida calidad. Consistía en un entorno de desarrollo altamente productivo, en el que participaban grupos pequeños de programadores utilizando herramientas que generaban código en forma automática tomando como entradas sintaxis de alto nivel. En general, se considera que este fue uno de los primeros hitos en pos de la agilidad en los procesos de desarrollo. Cabe mencionar que las metodologías ágiles no inventaron la noción de los procesos iterativos e incrementales, los cuales eran usados desde décadas pasadas.

Las Metodologías Ágiles o “ligeras” constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales (ISO-9000, CMM, etc.) 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.(HERNÁN. 2004)

Las Metodologías Ágiles valoran:

1. Al individuo y las interacciones en el equipo de desarrollo más que a las actividades y las herramientas.

2. Desarrollar software que funciona más que conseguir una buena documentación, implica minimalismo respecto del modelado y la documentación del sistema.

3. La colaboración con el cliente más que la negociación de un contrato.

4. Responder a los cambios más que seguir estrictamente una planificación.

¿Por qué surgen las Metodologías Ágiles?

1. Dificultad para implantar metodologías tradicionales. Sofisticadas herramientas CASE y notaciones (UML).

2. Una solución a medida para un segmento importante de proyectos de desarrollo de software

3. Aceptar el cambio.

Comparación Ágil - No-Ágil

Metodología Ágil	**Metodología No Ágil (Tradicional)**
Pocos artefactos	Más artefactos
Pocos roles	Más roles
No existe un contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo (además in-situ)	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio	Grupos grandes
Menos énfasis en la arquitectura	La arquitectura es esencial

Tabla 1: Comparación entre metodologías ágiles y tradicionales

La historia de las Metodologías Ágiles y su apreciación como tales en la comunidad de la ingeniería de software tiene sus inicios en la creación de una de las metodologías utilizada como arquetipo:

XP - eXtreme Programming. XP.

Entre las metodologías ágiles más destacadas hasta el momento podemos nombrar:

• XP – Extreme Programming

• Scrum

• Crystal Methodologies.

• DSDM – Dynamic Systems Development Method

• FDD – Feature Driven Development

• ASD – Adaptive Software Development

• Feature-Driven Development (FDD)

• Lean Development (LD)

Se puede decir que XP le dio conciencia al movimiento actual de metodologías ágiles.

1.3 El Software Educativo.

Por la importancia que se brinda al Software Educativo, se le dedicará el presente epígrafe para mostrar todos los aspectos fundamentales sobre este tipo de software que se encontró dentro de la tipología de software de aplicación, porque se puede distinguir el software arbitrariamente de la siguiente forma:

1.3.1 Tipologías del software

- **Software de sistema**, que permite funcionar al hardware. Su objetivo es aislar tanto como sea posible al programador de aplicaciones de los detalles del computador particular que se use, especialmente de las características físicas de la memoria, dispositivos de comunicaciones, impresoras, pantallas, teclados, etcétera. Incluye entre otros:
    - Sistemas operativos
    - Controladores de dispositivo
    - Herramientas de diagnóstico
    - Servidores
    - Sistemas de ventanas
    - Utilidades

- **Software de programación**, que proporciona herramientas para ayudar al programador a escribir programas informáticos y a usar diferentes lenguajes de programación de forma práctica. Incluye entre otros:

o Editores de texto

o Compiladores

o Intérpretes

o Enlazadores

o Depuradores

- **Software de aplicación**, que permite a los usuarios llevar a cabo una o varias tareas más específicas, en cualquier campo de actividad susceptible de ser automatizado o asistido, con especial énfasis en los negocios. Incluye entre otros:

o Aplicaciones de automatización industrial.

o Aplicaciones ofimáticas.

o Software educativo.

o Software médico.

o Bases de datos.

o Videojuegos.

o

1.3.2 Concepto de Software Educativo.

El software educativo se puede definir como entornos de trabajo en formato digital orientado temático y metodológicamente al proceso de formación. Los avances tecnológicos han enriquecido enormemente las posibilidades de trabajo al integrar elementos multimedia y nuevas concepciones pedagógicas. El apoyo de estos programas a la labor educativa puede ser catalogado como diverso dependiendo por un lado de las posibilidades ofertadas por el software y por otro la iniciativa metodológica del docente.

El software educativo se ha enfocado principalmente en dos polos:

El software educativo de tipo algorítmico que es aquel en donde predomina el aprendizaje por transmisión de conocimientos, dentro de estos se pueden considerar los denominados tutoriales, entrenadores y libros electrónicos.

El software educativo de tipo heurísticos que es donde el estudiante descubre el conocimiento interactuando con el ambiente de aprendizaje que le permite llegar a él, aquí se encuentran los simuladores, juegos educativos, sistemas expertos y sistemas tutoriales inteligentes.

Cada uno de estos software tienen sus cualidades y limitaciones que se deben tener en cuenta a la hora de seleccionar uno de ellos, dada una necesidad educativa.

Dentro de una gran variedad de software este trabajo se enfoca en el desarrollo de software educativo, **el cual se define como un programa automatizado diseñado con una intencionalidad educativa para ser utilizado en el proceso de aprendizaje, utiliza procedimientos para que el estudiante aprenda, se fomenta el análisis de problemas, facilita el trabajo en grupo, provee soporte en actividades docentes y en el sentido más amplio, mejora las habilidades del pensamiento y la resolución de problemas. (MARQUÈS 1995)**

1.3.3 Clasificaciones de los software educativos.

El software educativo es uno de los pilares del sistema de educación a distancia y se perfila como la herramienta base de las próximas generaciones de educandos. Es la base fundamental de la enseñanza y aprendizaje en la actualidad y se presentan de diferentes formas: juegos, laboratorios, libros etc.

De las 7 clasificaciones planteadas, se abordaran tres de ellas, las cuales son: (MARQUÈS 1995)

1- Programas Tutoriales.

Son programas que en mayor o menor medida dirigen, autorizan, el trabajo de los alumnos. Pretenden que, a partir de unas informaciones y mediante la realización de

ciertas actividades previstas de antemano, los estudiantes pongan en juego determinadas capacidades, y aprendan o refuercen conocimientos y habilidades. Estos tienen cuatro clasificaciones:

-Programas lineales, que presentan al alumno una secuencia de información ó ejercicios (siempre la misma o determinada aleatoriamente) con independencia de la corrección o incorrección de sus respuestas. Herederos de la enseñanza programada, transforman el ordenador en una máquina de enseñar transmisora de conocimientos y adiestradora de habilidades. No obstante, su interactividad resulta pobre y el programa se hace largo de recorrer.

-Programas ramificados, basados inicialmente también en modelos conductistas, siguen recorridos pedagógicos diferentes según el juicio que hace el ordenador sobre la corrección de las respuestas de los alumnos o según su decisión de profundizar más en ciertos temas. Ofrecen mayor interacción, más opciones, pero la organización de la materia suele estar menos compartimentada que en los programas lineales y exigen un esfuerzo más grande al alumno. Pertenecen a éste grupo los programas multinivel, que estructuran los contenidos en niveles de dificultad y previenen diversos caminos, y los programas ramificados con dientes de sierra, que establecen una diferenciación entre los conceptos y las preguntas de profundización, que son opcionales.

-Entornos tutoriales. En general están inspirados en modelos pedagógicos cognitivitas, y proporcionan a los alumnos una serie de herramientas de búsqueda y de proceso de la información que pueden utilizar libremente para construir la respuesta a las preguntas del programa. Este es el caso de los entornos de resolución de problemas, "problem solving", donde los estudiantes conocen parcialmente las informaciones necesarias para su resolución y han de buscar la información que falta y aplicar reglas, leyes y operaciones para encontrar la solución.

-Sistemas tutoriales expertos, como los Sistemas Tutores Inteligentes (Intelligent Tutoring Systems), que, elaborados con las técnicas de la Inteligencia Artificial y teniendo en cuenta las teorías cognitivas sobre el aprendizaje, tienden a reproducir un diálogo auténtico entre el programa y el estudiante, y pretenden comportarse como lo haría un tutor humano: guían a los alumnos paso a paso en su proceso de aprendizaje, analizan

su estilo de aprender y sus errores y proporcionan en cada caso la explicación o ejercicio más conveniente.

2- Simuladores

Presentan un modelo o entorno dinámico (generalmente a través de gráficos o animaciones interactivas) y facilitan su exploración y modificación a los alumnos, que pueden realizar aprendizajes inductivos o deductivos mediante la observación y la manipulación de la estructura subyacente; de esta manera pueden descubrir los elementos del modelo, sus interrelaciones, y pueden tomar decisiones y adquirir experiencia directa delante de unas situaciones que frecuentemente resultarían difícilmente accesibles a la realidad (control de una central nuclear, contracción del tiempo, pilotaje de un avión...).

También se pueden considerar simulaciones ciertos videojuegos que, al margen de otras consideraciones sobre los valores que incorporan (generalmente no muy positivos) facilitan el desarrollo de los reflejos, la percepción visual y la coordinación psicomotriz en general, además de estimular la capacidad de interpretación y de reacción ante un medio concreto.

Al igual que la anterior esta tiene dos clasificaciones:

-Modelos físico-matemáticos: Presentan de manera numérica o gráfica una realidad que tiene unas leyes representadas por un sistema de ecuaciones deterministas. Se incluyen aquí los programas-laboratorio, algunos trazadores de funciones y los programas que mediante un convertidor analógico-digital captan datos analógicos de un fenómeno externo al ordenador y presentan en pantalla un modelo del fenómeno estudiado o informaciones y gráficos que van asociados. Estos programas a veces son utilizados por profesores delante de la clase a manera de pizarra electrónica, como demostración o para ilustrar un concepto, facilitando así la transmisión de información a los alumnos, que después podrán repasar el tema interactuando con el programa.

-Entornos sociales: Presentan una realidad regida por unas leyes no del todo deterministas. Se incluyen aquí los juegos de estrategia y de aventura, que exigen una estrategia cambiante a lo largo del tiempo.

3- Programas de ejercitación o entrenadores

Su finalidad es que el estudiante practique mediante una repetición de preguntas y ejercicios. Responden a la necesidad de aprender destrezas específicas sencillas. Los entrenadores se diseñan con diferentes niveles de complejidad, en dependencia del fin que se persiga con el mismo (de aplicación reproductiva o productiva). Su principal objetivo es la adquisición por parte del estudiante de habilidades que lo conduzcan implícitamente a la reafirmación o consolidación de conocimientos.

Durante el proceso de diseño de los programas de ejercitación deben tomarse decisiones en torno al nivel contenido y estructura de las tareas a realizar. También debe decidirse el control del progreso en función del número de aciertos obtenidos en cada nivel.

El uso del software educativo en Cuba. Su inserción en el proceso pedagógico.

El Programa de Informática Educativa del MINED contempla, tanto la formación Informática de nuestros niños y jóvenes, como la introducción progresiva del software educativo como medio de enseñanza a todos los niveles de educación. Si bien en lo primero es donde se ha acumulado la mayor experiencia durante una década, no es así en el segundo.

A partir del cambio de la tecnología y la introducción de las computadoras en todos los niveles de enseñanza, en el curso 2001-2002, se implementan acciones concretas para transitar progresivamente hacia un uso masivo de estos recursos como medio de enseñanza.

La presencia de las computadoras en las instituciones escolares cubanas es un hecho palpable y su empleo como medio de enseñanza se encuentra en un período de tránsito.

Este tránsito se caracteriza por un uso progresivo de software educativo y sistemas autorizados de recuperación informativa (SARI) combinado con las habilidades informáticas adquiridas por los alumnos en las clases de Computación.

A nivel curricular debe lograrse que la dosificación de la asignatura Computación en los diferentes grados, garantice el dominio de procedimientos bien definidos de las diferentes tipologías de software que pueden ser usados por el resto de las asignaturas del grado en los tiempos de máquina planificados.

Hoy contamos con computadoras, un conjunto de software educativo instalados en los centros educacionales, profesores con una preparación informática adecuada en todas las escuelas en los diferentes niveles, para poder avanzar en la utilización del software educativo como medio de enseñanza.

Una aplicación multimedia educativa en la actualidad, resulta un excelente medio de aprendizaje en tanto que puede presentarle a un estudiante, material proveniente de diferentes fuentes: Textos, gráficos, audio, video, animación, simulaciones, fotografías, esquemas, mapas contextuales, etc. Cuando estos recursos se combinan a través de la interactividad se crean las posibilidades para el desarrollo de un entorno educativo realmente efectivo y tan centrado en el estudiante que más que llamarlo medio de enseñanza, resultaría más correcto denominarlo medio de aprendizaje.

En las escuelas de oficios el objetivo del uso de las computadoras es contribuir al trabajo correctivo - compensatorio, es por ello que los beneficios de la utilización en los estudiantes se multiplican y constituyen un recurso con grandes posibilidades educativas: enriquecen su aprendizaje, acentúan sus fortalezas, eliminan el sentido del fracaso y ayudan a identificar áreas de talentos e intereses vocacionales.

1.3.4 Multimedia

Conocida como una de las áreas de mayor importancia de la Informática, permite combinar diversos medios como texto, sonido, vídeo y gráficas, en una sola aplicación, que, junto con la técnica del hipertexto, permite agregar interactividad; lo que hace que el

usuario pueda navegar a través de la aplicación, a su libre elección, de acuerdo con sus intereses o necesidades de aprendizaje.

Multimedia evita las muchas dificultades que en países poco desarrollados se presentan al hacer uso de la red de redes, quien ofrece sistemas de enseñanza más sofisticados y con mayores posibilidades, no obstante debemos tener en cuenta que la solución mejor será siempre la que se adapte a las condiciones de cada lugar, modelo pedagógico, alcance tecnológico, etc., ningún modelo o tecnología será mejor o peor fuera del contexto en que se intente aplicar.

Ningún contenido multimedia, educativo o no, puede ser llevado con éxito a grandes escalas sin una previa organización del proceso de producción, más al tener en cuenta la amplia gama de recursos que intervienen y la necesidad de la sincronización de los esfuerzos.

Dentro del grupo de los materiales multimedia están los materiales multimedia educativos, que son los materiales multimedia que se utilizan con una finalidad educativa. (GRAELLS. 1999)

Atendiendo a su estructura, los materiales didácticos multimedia se pueden clasificar en programas tutoriales, de ejercitación, simuladores, bases de datos, constructores, programas herramienta..., presentando diversas concepciones sobre el aprendizaje y permitiendo en algunos casos (programas abiertos, lenguajes de autor) la modificación de sus contenidos y la creación de nuevas actividades de aprendizaje por parte de los profesores y los estudiantes. Con más detalle, la clasificación es la siguiente: (GRAELLS. 1999)

- Materiales formativos directivos. En general siguen planteamientos conductistas. Proporcionan información, proponen preguntas y ejercicios a los alumnos y corrigen sus respuestas.

- Programas de ejercitación. Se limitan a proponer ejercicios autocorrectivos de refuerzo sin proporcionar explicaciones conceptuales previas.

Su estructura puede ser: lineal (la secuencia en la que se presentan las actividades es única o totalmente aleatoria), ramificada (la secuencia depende de los aciertos de los

usuarios) o tipo entorno (proporciona a los alumnos herramientas de búsqueda y de proceso de la información para que construyan la respuesta a las preguntas del programa).

- Programas tutoriales. Presentan unos contenidos y proponen ejercicios autocorrectivos al respecto. Si utilizan técnicas de Inteligencia Artificial para personalizar la tutorización según las características de cada estudiante, se denominan tutoriales expertos.

- Bases de datos. Presentan datos organizados en un entorno estático mediante unos criterios que facilitan su exploración y consulta selectiva para resolver problemas, analizar y relacionar datos, comprobar hipótesis, extraer conclusiones... Al utilizarlos se pueden formular preguntas del tipo: ¿Qué características tiene este dato? ¿Qué datos hay con la característica X? ¿Y con las características X e Y?

- Programas tipo libro o cuento. Presenta una narración o una información en un entorno estático como un libro o cuento.

- Bases de datos convencionales. Almacenan la información en ficheros, mapas o gráficos, que el usuario puede recorrer según su criterio para recopilar información.

- Bases de datos expertas. Son bases de datos muy especializadas que recopilan toda la información existente de un tema concreto y además asesoran al usuario cuando accede buscando determinadas respuestas.

- Simuladores. Presentan modelos dinámicos interactivos (generalmente con animaciones) y los alumnos realizan aprendizajes significativos por descubrimiento al explorarlos, modificarlos y tomar decisiones ante situaciones de difícil acceso en la vida real. Al utilizarlos se pueden formular preguntas del tipo: ¿Qué pasa al modelo si modifico el valor de la variable X? ¿Y si modifico el parámetro Y?

- Modelos físico-matemáticos. Presentan de manera numérica o gráfica una realidad que tiene unas leyes representadas por un sistema de ecuaciones deterministas. Incluyen los programas-laboratorio, trazadores de funciones y los programas que con un convertidor analógico-digital captan datos de un fenómeno externo y presentan en pantalla informaciones y gráficos del mismo.

- Entornos sociales. Presentan una realidad regida por unas leyes no del todo deterministas. Se incluyen aquí los juegos de estrategia y de aventura

- Constructores o talleres creativos. Facilitan aprendizajes heurísticos, de acuerdo con los planteamientos constructivitas. Son entornos programables (con los interfaces convenientes se pueden controlar pequeños robots), que facilitan unos elementos simples con los cuales pueden construir entornos complejos. Los alumnos se convierten en profesores del ordenador. Al utilizarlos se pueden formular preguntas del tipo: ¿Qué sucede si añado o elimino el elemento X?

- Constructores específicos. Ponen a disposición de los estudiantes unos mecanismos de actuación (generalmente en forma de órdenes específicas) que permiten la construcción de determinados entornos, modelos o estructuras.

- Lenguajes de programación. Ofrecen unos "laboratorios simbólicos" en los que se pueden construir un número ilimitado de entornos.

Hay que destacar el lenguaje LOGO, creado en 1969 por Seymour Papert, un programa constructor que tiene una doble dimensión: proporciona a los estudiantes entornos para la exploración y facilita el desarrollo de actividades de programación, que suponen diseñar proyectos, analizar problemas, tomar decisiones y evaluar los resultados de sus acciones.

- Programas herramienta. Proporcionan un entorno instrumental con el cual se facilita la realización de ciertos trabajos generales de tratamiento de la información: escribir, organizar, calcular, dibujar, transmitir, captar datos...

- Programas de uso general. Los más utilizados son programas de uso general (procesadores de textos, editores gráficos, hojas de cálculo...) que provienen del mundo laboral. No obstante, se han elaborado versiones "para niños" que limitan sus posibilidades a cambio de una, no siempre clara, mayor facilidad de uso.

- Lenguajes y sistemas de autor. Facilitan la elaboración de programas tutoriales a los profesores que no disponen de grandes conocimientos informáticos.

Atendiendo a su concepción sobre el aprendizaje en los materiales didácticos multimedia podemos identificar diversos planteamientos: la perspectiva conductista

(B.F.Skinner), la teoría del procesamiento de la información (Phye), el aprendizaje por descubriendo (J. Bruner), el aprendizaje significativo (D. Ausubel, J. Novak), el enfoque cognitivo (Merrill, Gagné, Solomon...), el constructivismo (J.Piaget), el socio-construcivismo (Vigotsky). (GRAELLS. 1999)

- **La perspectiva conductista.** Desde la perspectiva conductista, formulada por B.F.Skinner hacia mediados del siglo XX y que arranca de Wundt y Watson, pasando por los estudios psicológicos de Pavlov sobre condicionamiento y de los trabajos de Thorndike sobre el refuerzo, intenta explicar el aprendizaje a partir de unas leyes y mecanismos comunes para todos los individuos.

- **Condicionamiento operante**. Formación de reflejos condicionados mediante mecanismos de estímulo-respuesta-refuerzo. Aprendizaje = conexiones entre estímulos y respuestas.

- **Ensayo y error con refuerzos y repetición**: las acciones que obtienen un refuerzo positivo tienden a ser repetidas.

- **Asociacionismo**: los conocimientos se elaboran estableciendo asociaciones entre los estímulos que se captan. Memorización mecánica.

- **Enseñanza programada.** Resulta especialmente eficaz cuando los contenidos están muy estructurados y secuenciados y se precisa un aprendizaje memorístico. Su eficacia es menor para la comprensión de procesos complejos y la resolución de problemas no convencionales.

- En muchos materiales didácticos multimedia directivos (ejercitación, tutoriales) subyace esta perspectiva.

- **Teoría del procesamiento de la información.** La teoría del procesamiento de la información, influida por los estudios cibernéticos de los años cincuenta y sesenta, presenta una explicación sobre los procesos internos que se producen durante el aprendizaje. Sus planteamientos básicos, en líneas generales, son ampliamente aceptados. Considera las siguientes fases principales:

- **Captación y filtro** de la información a partir de las sensaciones y percepciones obtenidas al interactuar con el medio.

- **Almacenamiento momentáneo** en los registros sensoriales y entrada en la memoria a corto plazo, donde, si se mantiene la actividad mental centrada en esta información, se realiza un reconocimiento y codificación conceptual.

- **Organización y almacenamiento definitivo** en la memoria a largo plazo, donde el conocimiento se organiza en forma de redes. Desde aquí la información podrá ser recuperada cuando sea necesario.

- En muchos materiales didácticos multimedia directivos (ejercitación, tutoriales) subyace esta perspectiva.

- **Aprendizaje por descubrimiento**. La perspectiva del aprendizaje por descubrimiento, desarrollada por J. Bruner, atribuye una gran importancia a la actividad directa de los estudiantes sobre la realidad.

- **Experimentación directa** sobre la realidad, aplicación práctica de los conocimientos y su transferencia a diversas situaciones.

- **Aprendizaje por penetración comprensiva.** El alumno experimentando descubre y comprende lo que es relevante, las estructuras.

- **Práctica de la inducción**: de lo concreto a lo abstracto, de los hechos a las teorías.

- **Utilización de estrategias heurísticas**, pensamiento divergente.

- **Currículum en espiral**: revisión y ampliación periódica de los conocimientos adquiridos.

- Esta perspectiva está presente en la mayoría de los materiales didácticos multimedia no directivos (simuladores, constructores...)

- **Aprendizaje significativo** (D. Ausubel, J. Novak) postula que el aprendizaje debe ser significativo, no memorístico, y para ello los nuevos conocimientos deben relacionarse con los saberes previos que posea el aprendiz. Frente al aprendizaje por descubrimiento de Bruner, defiende el aprendizaje por recepción donde el profesor estructura los contenidos y las actividades a realizar para que los conocimientos sean significativos para los estudiantes.

- Condiciones para el aprendizaje:

...significabilidad lógica (se puede relacionar con conocimientos previos)
...significabilidad psicológica (adecuación al desarrollo del alumno)
...actitud activa y motivación.

- **Relación de los nuevos conocimientos con los saberes previos.** La mente es como una red proposicional donde aprender es establecer relaciones semánticas.

- **Utilización de organizadores previos** que faciliten la activación de los conocimientos previos relacionados con los aprendizajes que se quieren realizar.

- Diferenciación-reconciliación integradora que genera una memorización comprensiva.

- Funcionalidad de los aprendizajes, que tengan interés, se vean útiles.

- Esta perspectiva está presente en la mayoría de los materiales didácticos multimedia.

- **Enfoque cognitivo. Psicología cognitivista**. El cognitivismo (Merrill, Gagné, Solomon...), basado en las teorías del procesamiento de la información y recogiendo también algunas ideas conductistas (refuerzo, análisis de tareas) y del aprendizaje significativo, aparece en la década de los sesenta y pretende dar una explicación más detallada de los procesos de aprendizaje, distingue:

- El aprendizaje es un proceso activo. El cerebro es un procesador paralelo, capaz de tratar con múltiples estímulos. El aprendizaje tiene lugar con una combinación de fisiología y emociones. El desafío estimula el aprendizaje, mientras que el miedo lo retrae.

El estudiante representará en su mente simbólicamente el conocimiento, que se considera (igual que los conductistas) como una realidad que existe externamente al estudiante y que éste debe adquirir. El aprendizaje consiste en la adquisición y representación exacta del conocimiento externo. La enseñanza debe facilitar la transmisión y recepción por el alumno de este conocimiento estructurado.

Posteriormente cuando se haga una pregunta al estudiante se activarán las fases: recuerdo, generalización o aplicación (si es el caso) y ejecución (al dar la respuesta, que si es acertada dará lugar a un refuerzo)

- **Condiciones internas** que intervienen en el proceso: motivación, captación y comprensión, adquisición, retención.

- **Condiciones externas**: son las circunstancias que rodean los actos didácticos y que el profesor procurará que favorezcan al máximo los aprendizajes.

- En muchos materiales didácticos multimedia directivos (ejercitación, tutoriales) subyace esta perspectiva.

- **Constructivismo**. J. Piaget, en sus estudios sobre epistemología genética, en los que determina las principales fases en el desarrollo cognitivo de los niños, elaboró un modelo explicativo del desarrollo de la inteligencia y del aprendizaje en general a partir de la consideración de la adaptación de los individuos al medio.

- **Considera tres estadios de desarrollo cognitivo universales:**

Sensoriomotor, estadio de las operaciones concretas y estadio de las operaciones formales. En todos ellos la actividad es un factor importante para el desarrollo de la inteligencia.

- Construcción del propio conocimiento mediante la interacción constante con el medio. Lo que se puede aprender en cada momento depende de la propia capacidad cognitiva, de los conocimientos previos y de las interacciones que se pueden establecer con el medio. En cualquier caso, los estudiantes comprenden mejor cuando están envueltos en tareas y temas que cautivan su atención. El profesor es un mediador y su metodología debe promover el cuestionamiento de las cosas, la investigación...

- Reconstrucción de los esquemas de conocimiento. El desarrollo y el aprendizaje se produce a partir de la secuencia***:*** equilibrio - desequilibrio – reequilibrio (que supone una adaptación y la construcción de nuevos esquemas de conocimiento).

Aprender no significa ni reemplazar un punto de vista (el incorrecto) por otro (el correcto), ni simplemente acumular nuevo conocimiento sobre el viejo, sino más bien transformar el conocimiento. Esta transformación, a su vez, ocurre a través del pensamiento activo y

original del aprendiz. Así pues, la educación constructivista implica la experimentación y la resolución de problemas y considera que los errores no son antitéticos del aprendizaje sino más bien la base del mismo.

El constructivismo considera que el aprendizaje es una interpretación personal del mundo (el conocimiento no es independiente del alumno), de manera que da sentido a las experiencias que construye cada estudiante. Este conocimiento se consensúa con otros, con la sociedad.

- Esta perspectiva actualmente está presente en muchos materiales didácticos multimedia de todo tipo, especialmente en los no tutoriales.

- **Socio-constructivismo.** Basado en muchas de las ideas de Vigotski, considera también los aprendizajes como un proceso personal de construcción de nuevos conocimientos a partir de los saberes previos (actividad instrumental), pero inseparable de la situación en la que se produce. Tiene lugar conectando con la experiencia personal y el conocimiento base del estudiante y se sitúa en un contexto social donde él construye su propio conocimiento a través de la interacción con otras personas (a menudo con la orientación del docente). Enfatiza en los siguientes aspectos:

- **Importancia de la interacción social** y de compartir y debatir con otros los aprendizajes. Aprender es una experiencia social donde el contexto es muy importante y el lenguaje juega un papel básico como herramienta mediadora, no solo entre profesores y alumnos, sino también entre estudiantes, que así aprenden a explicar, argumentar... Aprender significa "aprender con otros", recoger también sus puntos de vista. La socialización se va realizando con "otros" (iguales o expertos).

- **Incidencia en la zona de desarrollo próximo**, en la que la interacción con los especialistas y con los iguales puede ofrecer un "andamiaje" donde el aprendiz puede apoyarse.

Actualmente el **aprendizaje colaborativo y el aprendizaje situado,** que destaca que todo aprendizaje tiene lugar en un contexto en el que los participantes negocian los significados, recogen estos planteamientos. El aula debe ser un campo de interacción de ideas, representaciones y valores. La interpretación es personal, de manera que no hay

una realidad compartida de conocimientos. Por ello, los alumnos individualmente obtienen diferentes interpretaciones de los mismos materiales, cada uno construye (reconstruye) su conocimiento según sus esquemas, sus saberes y experiencias previas su contexto.

- Esta perspectiva actualmente está presente en algunos materiales didácticos multimedia no tutoriales.

Otras clasificaciones. Además de considerar la "estructura", los materiales didácticos multimedia se pueden clasificar según múltiples criterios:(GRAELLS. 1999)

- Según los contenidos (temas, áreas curriculares...)

- Según los destinatarios (criterios basados en niveles educativos, edad, conocimientos previos...)

- Según sus bases de datos: cerrado, abierto (= bases de datos modificables)

- Según los medios que integra: convencional, hipertexto, multimedia, hipermedia, realidad virtual.

- Según su "inteligencia": convencional, experto (o con inteligencia artificial)

- Según los objetivos educativos que pretende facilitar: conceptuales, procedimentales, actitudinales (o considerando otras taxonomías de objetivos).

- Según las actividades cognitivas que activa: control psicomotriz, observación, memorización, evocación, comprensión, interpretación, comparación, relación (clasificación, ordenación), análisis, síntesis, cálculo, razonamiento (deductivo, inductivo, crítico), pensamiento divergente, imaginación, resolución de problemas, expresión (verbal, escrita, gráfica…), creación, exploración, experimentación, reflexión metacognitiva, valoración...

- Según el tipo de interacción que propicia: recognitiva, reconstructiva, intuitiva/global, constructiva (Kemmis)

- Según su función en el aprendizaje: instructivo, revelador, conjetural, emancipador. (Hooper y Rusbhi)

- Según su comportamiento tutor, herramienta, aprendiz. (Taylor)

- Según el tratamiento de errores: tutorial (controla el trabajo del estudiante y le corrige), no tutorial.

- Según sus bases psicopedagógicas sobre el aprendizaje: conductista, cognitivista, constructivista (Begoña Gros)

- Según su función en la estrategia didáctica: entrenar, instruir, informar, motivar, explorar, experimentar, expresarse, comunicarse, entretener, evaluar, proveer recursos (calculadora, comunicación telemática)...

- Según su diseño: centrado en el aprendizaje, centrado en la enseñanza, proveedor de recursos. (Hinostroza, Mellar, Rehbein, Hepp, Preston)

- Según el soporte: disco, web

1.3.5 Funciones del software multimedia educativos.

Los materiales multimedia educativos, como los materiales didácticos en general, pueden realizar múltiples funciones en los procesos de enseñanza y aprendizaje. Las principales funciones que pueden realizar los recursos educativos multimedia son las siguientes:(GRAELLS. 1999)

Informativa: Estos materiales, a través de sus actividades, presentan unos contenidos que proporcionan información estructurada de la realidad, a los estudiantes.

Instructiva Entrenadora: Todos los software educativos orientan y regulan el aprendizaje de los estudiantes ya que, explícita o implícitamente, promueven determinadas actuaciones de los mismos encaminadas a este fin. Además, mediante sus códigos simbólicos, estructuración de la información e interactividad condicionan los procesos de aprendizaje.

Motivadora: La interacción con el ordenador suele resultar por sí misma motivadora.

Estos programas incluyen además elementos para captar la atención de los alumnos, mantener su interés y focalizarlo hacia los aspectos más importantes.

Evaluadora: La posibilidad de "feed back" inmediato a las respuestas y acciones de los alumnos, hace adecuados a los programas para evaluarles. Esta evaluación puede ser:

-Implícita: el estudiante detecta sus errores, se evalúa a partir de las respuestas que le da el ordenador.

-Explícita: el programa presenta informes valorando la actuación del alumno.

Explorar Experimentar: Les presentan a los estudiantes interesantes entornos donde explorar, experimentar, investigar, buscar determinadas informaciones, cambiar los valores de las variables de un sistema, etc.

Innovadora: Los programas educativos pueden desempeñar esta función ya que utilizan una tecnología actual y, en general, suelen permitir muy diversas formas de uso. Esta versatilidad abre amplias posibilidades de experimentación didáctica e innovación educativa en el aula.

1.4 Conclusiones.

El proceso de desarrollo de software requiere de un conjunto de conceptos, una metodología y un lenguaje propio.

El software educativo es un producto tecnológico diseñado para apoyar procesos educativos .Se diseñan para alcanzar diversos propósitos en el ámbito de la educación, desde bases de datos, programas de apoyo didáctico para exposición de algún contenido temático o alguna materia.

Los conceptos expuestos a lo largo de este capítulo son la fundamentación teórica que garantiza el cumplimiento del objetivo específico de este trabajo: Aplicar en el proceso de desarrollo del Software Educativo ¨Topografía¨ la metodología XP.

CAPÍTULO 2

SOLUCIÓN PROPUESTA.

"la programación extrema es una forma ligera, eficiente, flexible, predecible, científica y divertida de generar software"

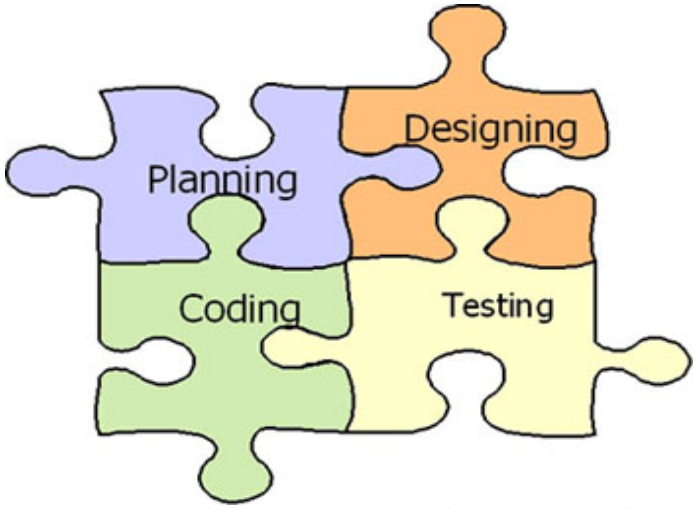


Figura 3: Representación de la metodología Programación Extrema

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.

XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. .XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

La programación extrema se basa en una serie de reglas y principios que se han ido gestando a lo largo de toda la historia de la ingeniería del software**.**(BECK. 1999)

2.1 Características de la Metodología XP.

- El desarrollo bajo XP tiene características que lo distinguen claramente de otras metodologías:
- Los diseñadores y programadores se comunican efectivamente con el cliente y entre ellos mismos.
- Los diseños del software se mantienen sencillos y libres de complejidad o pretensiones excesivas.
- Se obtiene retroalimentación de usuarios y clientes desde el primer día gracias a las baterías de pruebas.
- El software es liberado en entregas frecuentes tan pronto como sea posible.
- Los cambios se implementan rápidamente tal y como fueron sugeridos.
- Las metas en características, tiempos y costos son reajustadas permanentemente en función del avance real obtenido.
- Empieza en pequeño y añade funcionalidad con retroalimentación continua
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.

Derechos del Cliente.

- Decidir que se implementa.
- Saber el estado real y el progreso del proyecto.
- Añadir, cambiar o quitar requerimientos en cualquier momento.

- Obtener lo máximo de cada semana de trabajo.
- Obtener un sistema funcionando cada 3 o 4 meses

Derechos del Desarrollador.

- Decidir como se implementan los procesos.
- Crear el sistema con la mejor calidad posible.
- Pedir al cliente en cualquier momento aclaraciones de los requerimientos.
- Estimar el esfuerzo para implementar el sistema.
- Cambiar los requerimientos en base a nuevos descubrimientos.

Con estas características no es sorprendente que XP sea la metodología más apropiada para un entorno caracterizado por requerimientos cambiantes originados por un mercado fluctuante y los propios avances de la tecnología y los negocios.

2.2 Valores, prácticas y principios de la Metodología XP.

La programación extrema es una metodología de desarrollo ligera basada en una serie de valores, principios y una docena de prácticas que propician un aumento en la productividad a la hora de generar software.

2.2.1 Valores que promueve.

Los valores representan aquellos aspectos que se consideran fundamentales para garantizar el éxito de un proyecto de desarrollo de software. Los cuatro valores de XP son:

- Comunicación

Existen problemas en los equipos de desarrollo por falta de comunicación, por no comentar un cambio crítico en el diseño, por no preguntar lo que se piensa al cliente. La mala comunicación no surge por casualidad y hay circunstancias que conducen a la ruptura de la misma, como aquel jefe de proyecto que reprende al programador cuando

éste le comunica que hay un fallo en el diseño. XP ayuda mediante sus prácticas a fomentar la comunicación.

- Coraje

El coraje es un valor muy importante dentro de la programación extrema. Un miembro de un equipo de desarrollo extremo debe de tener el coraje de exponer sus dudas, miedos, experiencias sin embellecer éstas de ninguna manera. Esto es muy importante ya que un equipo de desarrollo extremo se basa en la confianza para con sus miembros. Detrás de este valor se encuentra el lema "si funciona, mejóralo", que choca con la práctica habitual de no tocar algo que funciona, por si acaso. .

- Simplicidad

Dado que no se puede predecir como va a ser en el futuro el software que se esta desarrollando; un equipo de programación extrema intenta mantener el software lo más sencillo posible. Esto quiere decir que no se va a invertir ningún esfuerzo en hacer un desarrollo que en un futuro pueda llegar a tener valor. En el XP frases como "...en un futuro vamos a necesitar..." no tienen ningún sentido ya que no aportan ningún valor en el momento. La simplicidad y la comunicación se complementan, cuanto mas simple es tu sistema menos tienes que comunicar de el.

- Retroalimentación.

La retroalimentación actúa junto con la sencillez y la comunicación, cuanto mayor es la retroalimentación más fácil es la comunicación. Cuanto mas simple un sistema, mas fácil de probar. Escribir pruebas nos orienta como simplificar un sistema, hasta que las pruebas funcionen, cuando las pruebas funcionen habrá un gran avance. Esta retroalimentación se toma del cliente, de los miembros del equipo, en cuestión de todo el entorno en el que se mueve un equipo de desarrollo ágil. (FERRER 2002)

2.2.2 Prácticas.

La mayoría de estas prácticas no son nuevas, sino que han sido reconocidas por la industria como las mejores prácticas durante años. En la XP, dichas prácticas son llevadas al extremo para que se obtenga algo mucho mejor que la suma de las partes.

1. **Planificación Incremental**

La Programación Extrema asume que la planificación nunca será perfecta, y que varía en función de cómo varíen las necesidades del negocio. Por tanto, el valor real reside en obtener rápidamente un plan inicial, y contar con mecanismos que permitan conocer con precisión la situación actual del proyecto. Como es lógico, la planificación es iterativa: un representante del negocio decide al comienzo de cada iteración qué características concretas se van a implementar.

2. **Testing**

La ejecución automatizada de tests es un elemento clave de la XP. Existen tanto tests internos (o tests de unidad), para garantizar que el mismo es correcto, como tests de aceptación, para garantizar que el código hace lo que debe hacer. El cliente es el responsable de definir los tests de aceptación, no necesariamente de implementarlos. Él es la persona mejor cualificada para decidir cuál es la funcionalidad más valiosa.

El hecho de que los tests sean automatizados es el único modo de garantizar que todo funciona. Desde el punto de vista de la XP, si no hay tests, las cosas sólo funcionan en apariencia.

El objetivo de los tests no es corregir errores, sino prevenirlos. Los tests siempre se escriben antes que el código a testear, no después: esto aporta un gran valor adicional, pues fuerza a los desarrolladores a pensar cómo se va a usar el código que escriben, situándolos en la posición de consumidores del software. Elaborar los tests exige pensar por adelantado cuáles son los problemas más graves que se pueden presentar, y cuáles son los puntos dudosos.

Un efecto lateral importante de los tests es que dan una gran seguridad a los desarrolladores: es posible llegar a hacer cambios más o menos importantes sin miedo a

problemas inesperados, dado que proporcionan una red de seguridad. La existencia de tests hace el código más seguro.

3. Programación en parejas

La XP incluye, como una de sus prácticas estándar, la programación en parejas. Nadie programa en solitario, siempre hay dos personas delante del ordenador. Ésta es una de las características que más se cuestiona al comienzo de la adopción de la XP dentro de un equipo, pero en la práctica se acepta rápidamente y de forma entusiasta. El hecho de que todas las decisiones las tomen al menos dos personas proporciona un mecanismo de seguridad enormemente valioso.

Con dos personas responsabilizándose del código en cada momento, es menos probable que se caiga en la tentación de dejar de escribir tests. Es muy difícil que dos personas se salten tareas por descuido o negligencia ya que el código siempre está siendo revisado por otra persona.

4. Refactorización

El código de la mayor parte de las aplicaciones empieza en un razonable buen estado, para luego deteriorarse de forma progresiva. El coste desorbitado del mantenimiento, modificación y ampliación de aplicaciones ya existente se debe en gran parte a este hecho.

Uno de los objetivos de la XP es mantener la curva de costes tan plana como sea posible, por lo que existen una serie de mecanismos destinados a mantener el código en buen estado, modificándolo activamente para que conserve claridad y sencillez. A este proceso básico para mantener el código en buena forma se le llama refactorización.

La refactorización no sólo sirve para mantener el código legible y sencillo: también se utiliza cuando resulta conveniente modificar código existente para hacer más fácil implementar nueva funcionalidad.

5. Diseño simple

Otra práctica fundamental de la Programación Extrema es utilizar diseños tan simples como sea posible. El principio es "utilizar el diseño más sencillo que consiga que todo funcione". Se evita diseñar características extra porque a la hora de la verdad la experiencia indica que raramente se puede anticipar qué necesidades se convertirán en reales y cuáles no. La XP exige no tener la ilusión de que un diseño puede resolver todas o gran parte de las situaciones futuras: lo que parece necesario cambia con frecuencia, es difícil acertar a priori.

La XP define un "diseño tan simple como sea posible" como aquél que:

Pasa todos los tests.

No contiene código duplicado.

Deja clara la intención de los programadores (enfatiza el qué, no el cómo) en cada línea de código.

Contiene el menor número posible de clases y métodos.

6. Propiedad colectiva del código

La XP aboga por la propiedad colectiva del código. En otras palabras, todo el mundo tiene autoridad para hacer cambios a cualquier código, y es responsable de ellos. Esto permite no tener que estar esperando a otros cuando todo lo que hace falta es algún pequeño cambio.

Cada cuál es responsable de las modificaciones que haga. El principio básico es "quien lo rompe, lo arregla, no importa si está en el código propio o en el de otro".

Hay que tener en cuenta que la existencia de tests automatizados impide que se produzca un desarrollo anárquico, al ser cada persona responsable de que todos los tests se ejecuten con éxito al incorporar los cambios que ha introducido al programa.

7. Integración continua

En muchos casos la integración de código produce efectos laterales imprevistos, y en ocasiones la integración puede llegar a ser realmente traumática, cuando dejan de funcionar cosas por motivos desconocidos. La Programación Extrema hace que la integración sea permanente, con lo que todos los problemas se manifiestan de forma inmediata, en lugar de durante una fase de integración remota.

8. Cliente en el equipo

El cliente siempre está disponible para resolver dudas y para decidir qué y qué no se hace en cada momento, en función de los intereses del negocio. Debido a su inmersión dentro del equipo, y a que es él quien decide qué y qué no se hace, junto con los tests que verifican si la funcionalidad es la correcta y deseada, el cliente obtiene información absolutamente realista del estado del proyecto.

9. Releases pequeñas

Siguiendo la política de la XP de dar el máximo valor posible en cada momento, se intenta liberar nuevas versiones de las aplicaciones con frecuencia. Éstas deben ser tan pequeñas como sea posible, aunque deben añadir suficiente valor como para que resulten valiosas para el cliente.

10. Semanas de 40 horas

La Programación Extrema lleva a un modo de trabajo en que el equipo está siempre al 100%. Una semana de 40 horas en las que se dedica la mayor parte del tiempo a tareas que suponen un avance puede dar mucho de sí, y hace innecesario recurrir a sobreesfuerzos -excepto en casos extremos.

Además, el sobreesfuerzo continuado pronto lleva a un rendimiento menor y a un deterioro de la moral de todo el equipo.

11. Estándares de codificación

Para conseguir que el código se encuentre en buen estado y que cualquier persona del equipo pueda modificar cualquier parte del código es imprescindible que el estilo de codificación sea consistente. Un estándar de codificación es necesario para soportar otras prácticas de la XP.

12. Uso de Metáforas

La comunicación fluida es uno de los valores más importantes de la Programación Extrema, para conseguir que esto ocurra es imprescindible, entre otras cosas, utilizar el vocabulario del negocio. También es fundamental huir de definiciones abstractas. Dicho de otro modo, la XP no pretende seguir la letra de la ley, sino su espíritu. Dentro de este enfoque es fundamental buscar continuamente metáforas que comuniquen intenciones y resulten descriptivas, enfatizando el qué por delante del cómo.

El objetivo final debe ser aplicar todas las prácticas, ya que representan un conjunto completo, "si no las aplicas todas no se hace eXtreme Programming".

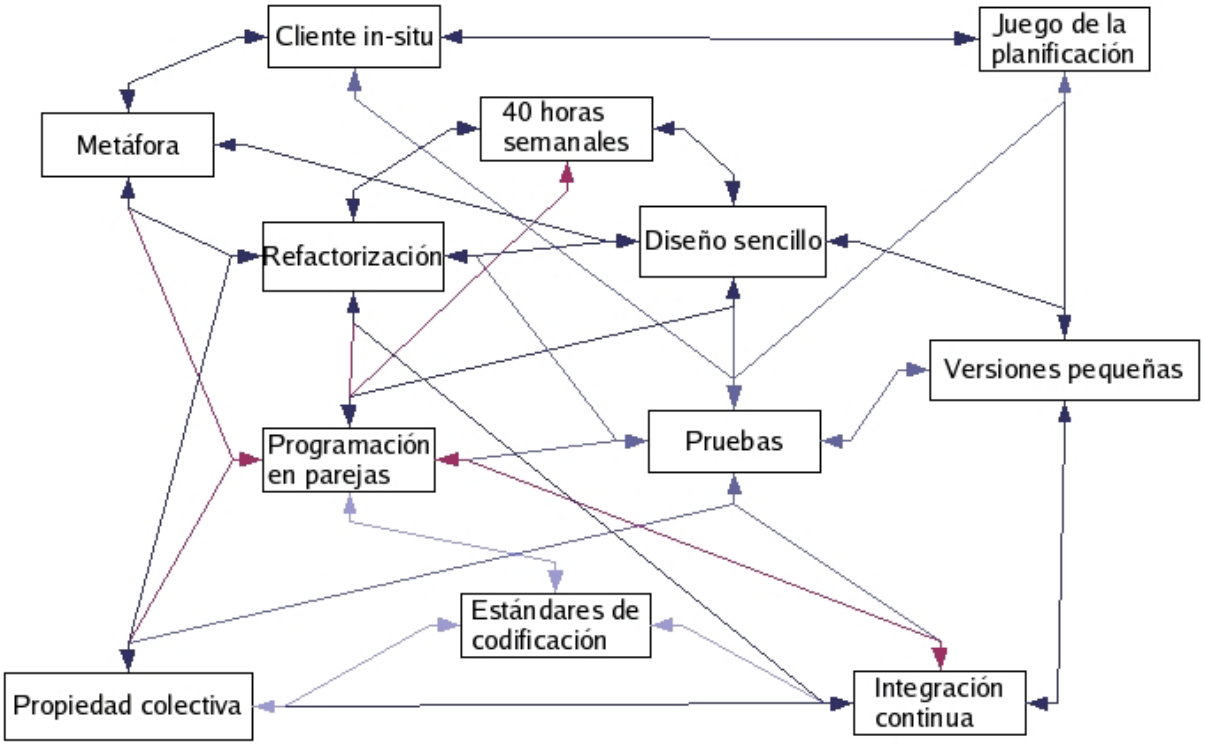


Figura 4: Combinación de todas las prácticas.

Muchas de las prácticas propuestas contribuyen a maximizar la comunicación entre las personas, permitiendo de esa forma una mayor transferencia de conocimiento entre los desarrolladores y con el cliente, quien también es parte del equipo. Esto es logrado en la práctica gracias a la disposición física del lugar de trabajo.

La idea es reunir a todas las personas en una misma oficina manteniendo una distribución denominada cavernas y común En la misma se observan escritorios dispuestos en el centro con varias computadoras, cada una con dos sillas para permitir la programación de a pares. En las paredes se observan pequeños boxes o escritorios con sillas, los cuales pueden ser usados por los programadores en forma privada, para realizar llamados, consultar mail, o simplemente descansar la mente. (Anexo 3)

2.2.3 Principios de la Programación Extrema.

Los principios fundamentales se apoyan en los valores y también son cuatro. Se busca:

- Realimentación veloz
- Modificaciones incrementales
- Trabajo de calidad
- Asunción de simplicidad.

Los principios suponen un puente entre los valores (algo intrínseco al equipo de desarrollo) y las prácticas, que están más ligadas a las técnicas que se han de seguir. (FERRER 2002)

2.3 Las cuatro variables.

XP define cuatro variables para cualquier proyecto software: coste, tiempo, calidad y alcance.

De estas cuatro variables, sólo tres de ellas podrán ser fijadas por las fuerzas externas al proyecto (clientes y jefes de proyecto), mientras que el valor de la cuarta variable será establecido por el equipo de desarrollo en función de los valores de las otras tres.

Normalmente los clientes y jefes de proyecto se creen capaces de fijar de antemano el valor de todas las variables:

«Quiero estos requisitos satisfechos para el día uno del mes que viene, para lo cual cuentan con este equipo. ¡Ah, y ya saben que la calidad es lo primero!»

Cuando esto ocurre, la calidad es lo primero que se pierde de la ecuación por una sencilla razón, pues frecuentemente se ignora que nadie es capaz de trabajar bien cuando está sometido a mucha presión. XP hace a las cuatro variables visibles para todo el mundo, programadores, clientes y jefes de proyecto, de manera que se pueda jugar con los valores de la entrada hasta que la cuarta variable tenga un valor que satisfaga a todos.(FERRER 2002)

Las cuatro variables no guardan entre sí una relación tan obvia como a menudo se quiere ver. XP hace especial énfasis en equipos de desarrollo pequeños (diez o doce

personas como mucho) que, naturalmente, se podrán ir incrementando a medida que sea necesario, pero no antes, o los resultados serán generalmente contrarios a lo esperado. Es bueno incrementar el **coste** del proyecto en aspectos como máquinas más rápidas, más especialistas, técnicos en determinadas áreas o mejores oficinas para el equipo de desarrollo.

Con la **calidad** también sucede otro fenómeno extraño: frecuentemente, aumentar la calidad conduce a que el proyecto pueda realizarse en menos tiempo. En efecto, en cuanto el equipo de desarrollo se habitúe a realizar pruebas intensivas y se sigan estándares de codificación, poco a poco comenzará a avanzar mucho más rápido de lo que lo hacía antes, mientras la calidad del proyecto se mantiene asegurada por las pruebas al 100%, lo que conlleva mayor confianza en el código y, por tanto, mayor facilidad para adaptarse al cambio, lo que hace que se programe más rápido y así sucesivamente.

Frente a esto, está la tentación de sacrificar la calidad interna del proyecto –la que es apreciada por los programadores – para reducir el tiempo de entrega del proyecto, en la confianza de que la calidad externa, la que notan los clientes, no se vea demasiado afectada. Sin embargo, ésta claro que todo el mundo trabaja mucho mejor cuando le dejan hacer trabajo de calidad. No tener esto en cuenta conduce a la desmoralización del equipo y, con ello, a la larga, a la ralentización del proyecto mucho más allá del **tiempo** que hubiera podido ganarse al principio con esta reducción de calidad. (FERRER 2002)

En cuanto al **alcance** del proyecto, es una buena idea dejar que sea esta la variable libre, de manera que, una vez fijadas las otras tres, el equipo de desarrollo determine el alcance mediante:

• La estimación de las tareas a realizar para satisfacer los requisitos del cliente.

• La implementación de los requisitos más importantes primero, de manera que el proyecto tenga en cada instante tanta funcionalidad como sea posible.

El coste del cambio

Una de las suposiciones establecidas en la industria del software es que el coste de los cambios en un software crece exponencialmente con el tiempo.

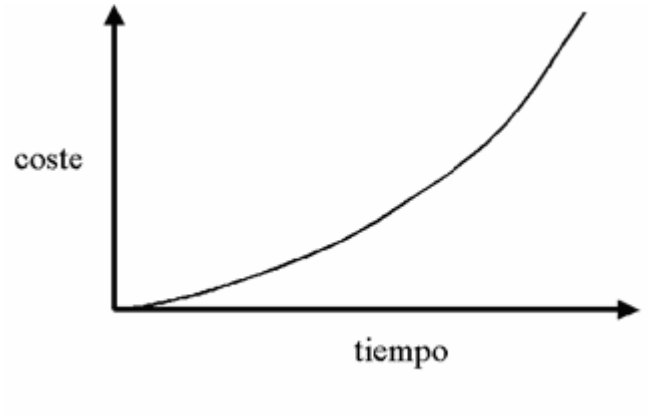


Figura 5: Representación del valor del coste del cambio

XP propone que si el sistema que empleas hace que el coste del software aumente con el tiempo debes de actuar de forma diferente a cómo lo haces. XP propugna que esta curva ha perdido validez y con una combinación de buenas practicas de programación y tecnología es posible lograr que la curva sea la contraria.

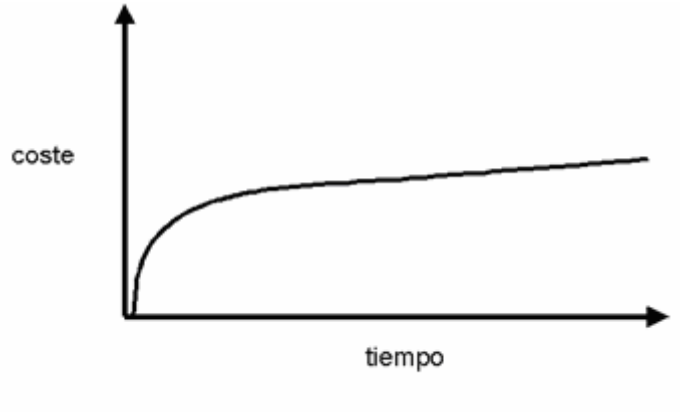


Figura 6: Representación del valor del coste del cambio en XP

2.4 Roles en XP

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

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

Encargado de pruebas (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 realimentación 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 las prácticas 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: Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es la coordinación.

2.5 Etapas de la metodología XP.

2.5.1 - Planificación.

XP plantea la planificación como un permanente diálogo entre la parte empresarial y técnica del proyecto, en la que los primeros decidirán:

Alcance**:** ¿Qué es lo que el software debe de resolver para que genere valor?

Prioridad**:** ¿Qué debe ser hecho en primer lugar?

Composición de las versiones: ¿Cuánto es necesario hacer para saber si el negocio va mejor con software que sin el? En cuanto el software aporte algo al negocio se deben tener lista las primeras versiones.

Fechas de versiones**:** ¿Cuáles son las fechas en la presencia del software?

Consecuencias**:** Informar sobre las consecuencias de la toma de decisiones por parte del negocio.

Procesos**:** ¿Cómo se organiza el trabajo y el equipo?

Programación detallada**:** Dentro de una versión ¿Qué problemas se resolverán primero?

Historias de Usuario

En XP la gestión de requisitos es extremadamente simple, el cliente escribe y prioriza las historias de usuario que expresan las necesidades del sistema. Los programadores estiman el esfuerzo asociado y las dependencias entre ellas. Para planificar el trabajo desde el punto de vista técnico las historias de usuario son divididas en tareas para las cuales también se realiza una estimación. Teniendo en cuenta el esfuerzo asociado a las historias de usuario y las prioridades del cliente se define una versión que sea de valor para el cliente y que tenga una duración de unos pocos meses.

Las historias de usuario tienen el mismo propósito que los casos de uso, pero no son lo mismo. Las escriben los propios clientes, tal y como ven ellos las necesidades del sistema. Por tanto serán descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica.

Conducen el proceso de creación de los test de aceptación. Uno o más de uno de estos test se utilizarán para verificar que las historias de usuario han sido implementadas correctamente.

La principal diferencia es el nivel de detalle. Las historias de usuario solamente proporcionaran los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de dicha historia de usuario. El nivel de detalle de las historias de usuario debe ser el mínimo posible que permita hacerse una ligera idea de cuánto

costará implementar el sistema. Cuando se llegue a la fase de implementación, los desarrolladores podrán acudir al cliente para ampliar detalles.

Los desarrolladores deberán hacer una estimación de cuánto tiempo, idealmente, les llevará implementar cada historia de usuario. Las condiciones ideales son aquellas en las que se codifica la historia de usuario sin otras distracciones y sabiendo exactamente qué es lo que hay que implementar. Como resultado deberíamos obtener un periodo ideal de 1, 2 ó 3 semanas. Más de 3 semanas implica que debemos dividir la historia de usuario en partes. Menos de 1 semana implica que la historia de usuario es demasiado sencilla y tendremos que unir dos o más de ellas. Es de esperar situaciones relativas a historias de usuario tales como: aparición de nuevas, postergación en cuanto a la iteración en la cual se implementa, no finalización en la iteración planificada y modificación durante la iteración o una vez completada en una iteración previa.

Historia de Usuario	
Número:	**Nombre Historia de Usuario:**
Modificación de Historia de Usuario Número:	
Usuario:	**Iteración Asignada:**
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:**
Riesgo en Desarrollo: (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción:	
Observaciones:	

Tabla 2: Representación de una Historia de Usuario

Actualmente se trabaja en base a mejorar el tratamiento dado a las historias de usuario. Se pretende registrar en soporte informático dicho proceso a través del uso de una herramienta denominada Volt.

Volt es una herramienta que permite gestionar historias de usuario a través de un navegador Web incluyendo control de versiones. Para ello la herramienta codifica cada historia de usuario en XML y las registra en un repositorio Subversión. Al mismo tiempo la herramienta permite gestión de usuarios ofreciendo una interfaz personalizada a cada tipo o rol de usuario.

Figura 7: Interfaz personalizada para la gestión de las historias de usuario

En la actualidad solo se cubren los aspectos relacionados con las historias de usuario pero se trabaja en base a ampliar el marco de acción, cubriendo más actividades como las pruebas.

Plan de Entregas

Las historias de usuario servirán para crear el plan estimado de entregas. Se convocará una reunión para crear el plan de entregas. El plan de entregas se usará para crear los planes de iteración para cada iteración. Con cada historia de usuario previamente evaluada en tiempo de desarrollo ideal, el cliente las agrupará en orden de importancia. Una semana ideal es cuánto tiempo costaría implementar dicha historia si no tenemos nada más que hacer, incluyendo la parte de test correspondiente.

De esta forma se puede trazar el plan de entregas en función de estos dos parámetros: tiempo de desarrollo ideal y grado de importancia para el cliente. Las iteraciones individuales son planificadas en detalle justo antes de que comience cada iteración.

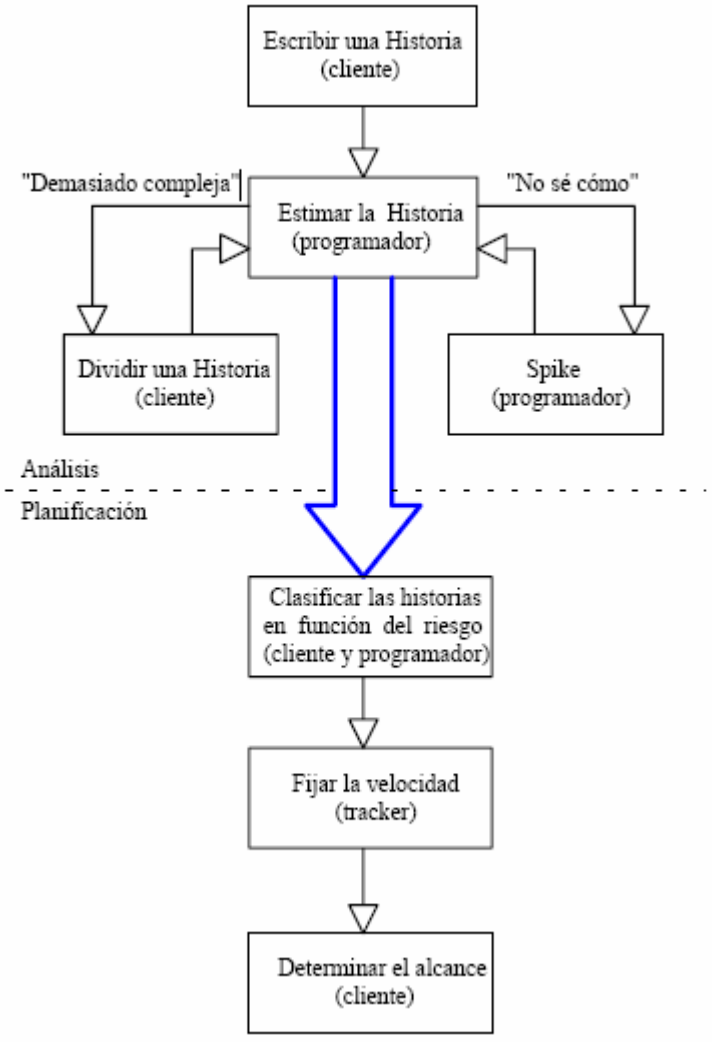


Figura 8: Plan de Entregas

Velocidad del Proyecto

La velocidad del proyecto es una medida de cuan rápido se está desarrollando. La velocidad del proyecto se usa para determinar cuántas historias de usuario pueden ser implementadas antes de una fecha dada (tiempo), o cuánto tiempo es necesario para llevar a cabo un conjunto de historias (alcance). Cuando se realiza una planificación por alcance se divide el numero total de semanas entre la velocidad de proyecto para determinar cuántas iteraciones estarán disponibles.

Iteraciones

Cada iteración corresponde a un periodo de tiempo de desarrollo del proyecto de entre una y tres semanas. De esta forma, un proyecto, se divide en una docena de iteraciones, más o menos. Al principio de cada iteración se debería convocar una reunión para trazar el plan de iteración correspondiente.

Está prohibido intentar adelantarse e implementar cualquier cosa que no esté planeada para la iteración en curso. Habrá suficiente tiempo para añadir la funcionalidad extra cuando sea realmente importante según el plan de entregas.

Se usará la velocidad del proyecto para determinar si una iteración está sobrecargada. La suma de los días que costará desarrollar todas las tareas de la iteración no debería sobrepasar la velocidad del proyecto de la iteración anterior. Si la iteración está sobrecargada, el cliente deberá decidir que historias de usuario retrasar a una iteración posterior. Si, por el contrario, la iteración tiene huecos se rellenará con otras historias de usuario.

Plan de Iteraciones

El plan de iteración consiste en seleccionar las historias de usuario que, según el plan de entregas, corresponderían a esta iteración. También se eligen qué pruebas de aceptación fallidas se corregirán.

Un plan de iteración puede verse como:

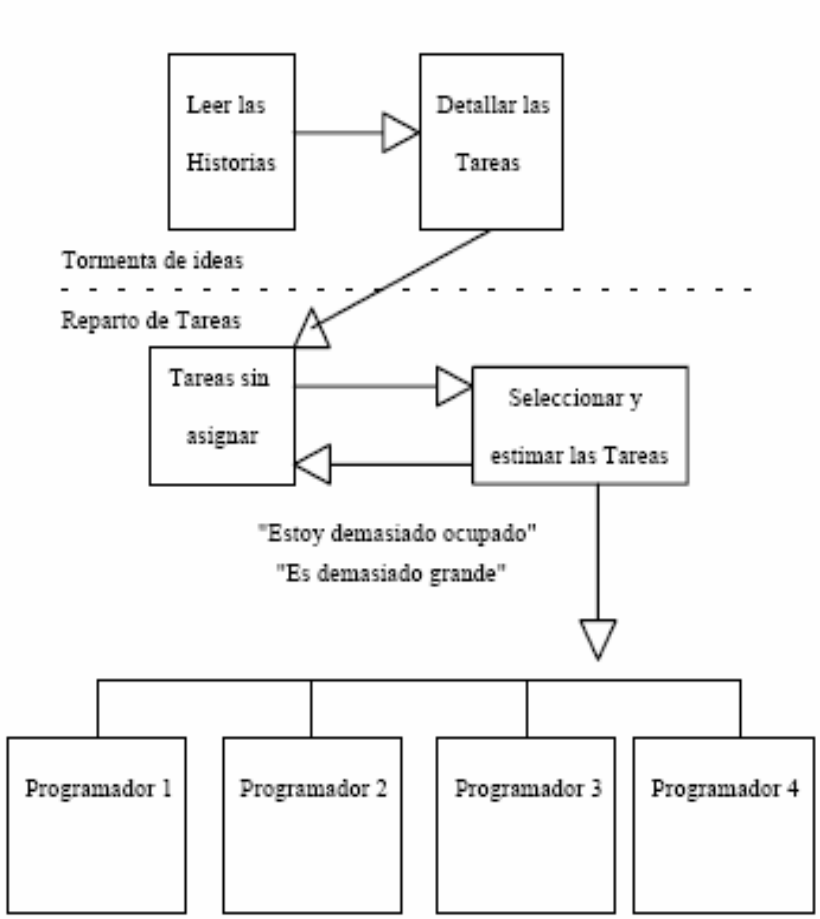


Figura 9: Plan de Iteración

Cada historia de usuario se transformará en tareas de desarrollo. Cada tarea de desarrollo corresponderá a un periodo ideal de uno a tres días de desarrollo.

Es necesario mantener vigiladas la velocidad del proyecto y el movimiento de historias de usuario. Puede ser necesario volver a calcular las historias de usuario y negociar el plan de entrega cada de tres a cinco iteraciones. Como estaremos siempre implementando las historias de usuario más importantes para el cliente, estaremos haciendo lo máximo posible por nuestro cliente y la dirección.

En cada iteración iremos alcanzado unas metas:

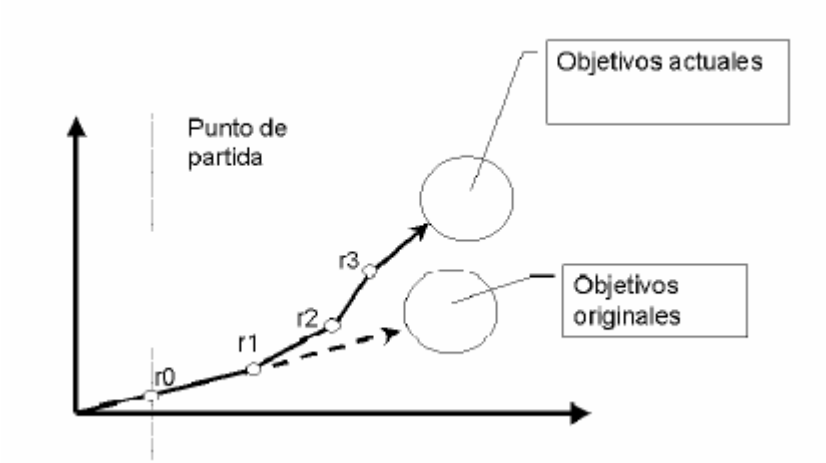


Figura 10: Representación de las metas alcanzadas en cada iteración

Rotación del personal

Si sólo una persona del equipo de desarrollo es capaz de trabajar en un área concreta, existe un riesgo enorme si esa persona decide por cualquier circunstancia dejar el proyecto. De esta forma, las rotaciones permiten que todo el mundo conozca cómo funciona el sistema en general y ayudan a realizar un reparto más equitativo del trabajo.

Además el hecho de que se asignen por parejas permite entrenar a un nuevo miembro, simplemente dividiendo el grupo original en dos.

Reuniones de seguimiento

La comunicación entre las diferentes partes que intervienen en un proyecto resulta fundamental para su desarrollo. Esto se consigue gracias a las reuniones. En ellas se analizan los problemas y las soluciones y se recomienda que sean frecuentes, de poca duración y a ser posible delante de la pantalla del ordenador. Según la metodología XP, se recomienda que sean diarias. La reunión de seguimiento de cada mañana debe usarse para sacar a la luz los problemas, las soluciones y centrar el objetivo del equipo.

Corrección de la metodología

Se debe corregir el proceso cuando éste falle. Al comenzar con un proyecto, se sigue la metodología XP, pero se debe cambiar aquello que no funcione. Además los cambios que se realicen deben ser comunicados al resto del equipo, todo el mundo debe estar al corriente de los cambios. Esto no significa que se cambie lo que a algún miembro del equipo no le guste, sino que se cambie aquello que funciona con el problema en particular.(ESCRIBANO. 2002)

Ejemplo de como se trabaja con XP.

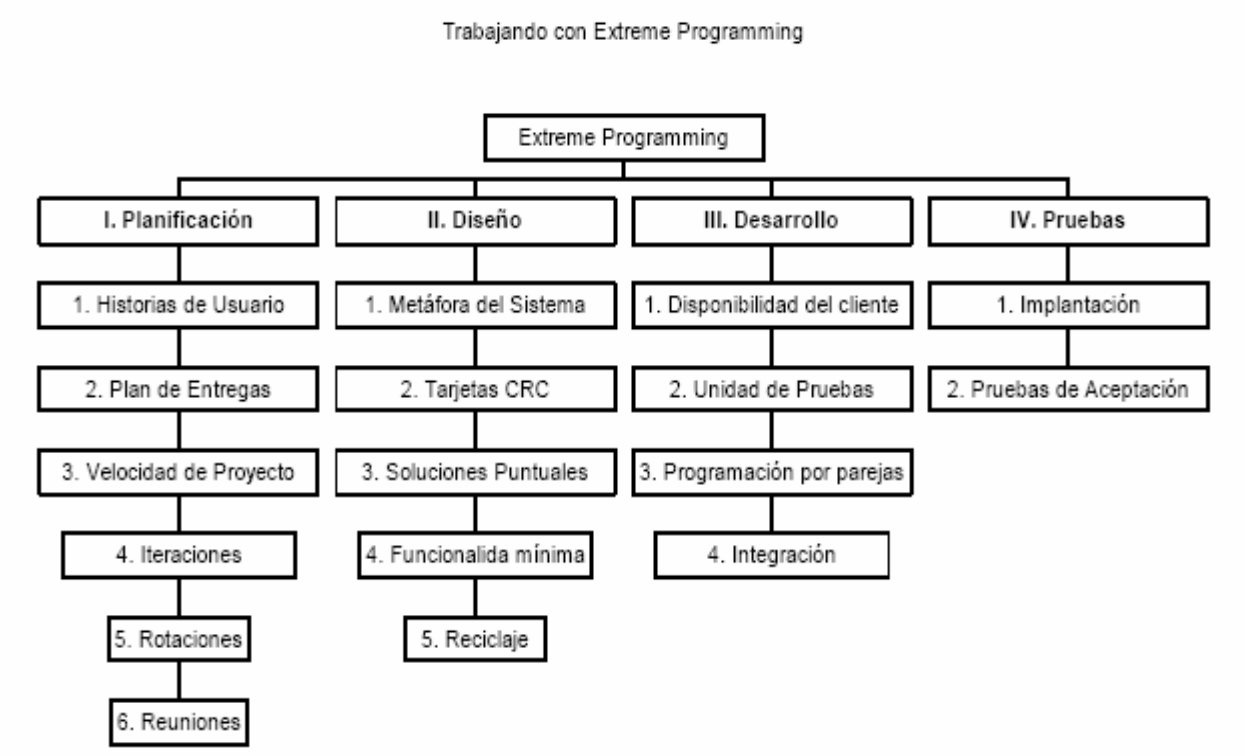


Figura 11: Representación de las fases de la Metodología XP

2.5.2 Diseño.

XP establece unas recomendaciones o premisas a la hora de abordar esta etapa.

Simplicidad

La simplicidad es la llave

Siempre cuesta menos tiempo de implementar un diseño sencillo que uno complejo. Por lo que, hay que tratar siempre de realizar las cosas de la manera más sencilla posible. Si alguna parte de la implementación resulta especialmente compleja, debe replantearse (divide y vencerás). Así, cualquier cambio y modificación será mucho más sencillo. En ocasiones, realizar un diseño sencillo puede resultar una tarea especialmente difícil.

Metáfora para el sistema

Una metáfora para el sistema es una historia que todo el mundo puede contar a cerca de cómo el sistema funciona.(BECK. 1999) La tarea de elegir una metáfora para el sistema permite mantener la coherencia de nombres de todo aquello que se va a implementar. El nombre de los objetos o partes de nuestro sistema es muy importante. La tarea de “poner nombre”, sencilla a simple vista, no lo es tanto. Se debe elegir un sistema de nombres que permita que cualquiera que lo vea adivine la relación entre el objeto y aquello que representa.

Tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración)

Para poder diseñar el sistema como un equipo se debe cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas CRC permiten que el equipo completo contribuya en la tarea del diseño.

Una tarjeta CRC representa un objeto. El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican

en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente.

Soluciones Puntuales

Un programa *Spike*, es un programa muy simple que explora una posible solución al problema. A partir de estos pequeños programas se construye la solución del problema. El sistema se pone por primera vez en producción en, a lo sumo, unos pocos meses, antes de estar completamente terminado. Las sucesivas versiones serán más frecuentes.

No se añadirá funcionalidad en las primeras etapas.

Se debe evitar añadir funcionalidades que en ese momento no se necesiten, aun incluso cuando se sabe exactamente cómo implementar. Es decir, hay que centrarse en la tarea que se ha fijado para ese momento y hacerla de la mejor forma posible. Hay que programar lo que se ha fijado, y no perder el tiempo en desarrollar código que no se sabe si será utilizado.

Reaprovechar cuando sea posible

Cuando se elimina redundancia, se excluye funcionalidad inútil, y se fortalecen antiguos diseños, de esta forma se está reciclando código. El reciclaje, dentro del ciclo de vida de un proyecto, ahorra tiempo e incrementa la calidad. El reciclaje implica mantener el código limpio y fácil de comprender, modificar y ampliar. Esto puede resultar un poco costoso al principio, pero resulta fundamental a la hora de realizar diseños futuros.(ESCRIBANO. 2002)

2.5.3 Desarrollo.

Ciclo de vida de XP para esta etapa

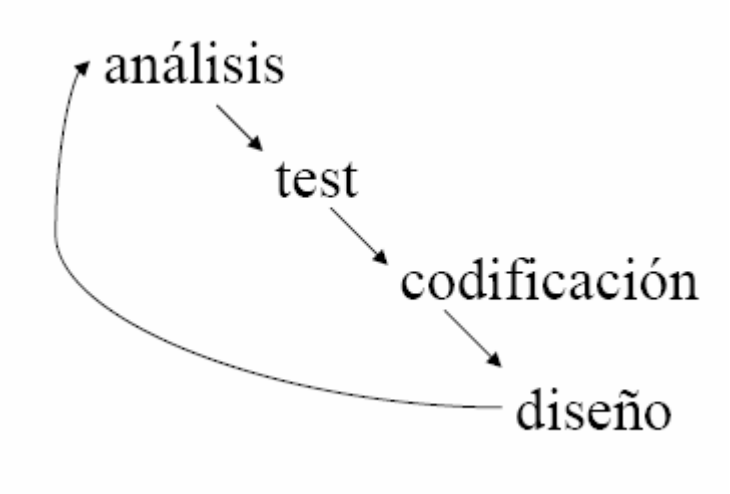


Figura 12: Ciclo de vida de XP para la etapa de desarrollo

Esta etapa debe reunir las siguientes características o cualidades:

Disponibilidad del cliente

Una de las pocas condiciones que impone la metodología XP es tener al usuario siempre disponible. No sólo para ayudar al equipo de desarrollo, sino formando parte de él. Todas las fases que se realizan en un proyecto XP requieren de comunicación con el usuario, preferiblemente cara a cara, en persona, sin intermediarios.

Durante la reunión del plan de entregas, el usuario propondrá qué historia de usuario se incluye en cada plan. También se negociarán los plazos de entrega. El usuario o cliente tomará las decisiones que le afecten para alcanzar los objetivos de su negocio.

También es necesario que el cliente colabore en la realización de los test. Estos test comprobarán que el sistema está listo para pasar a la fase de producción. El usuario

comprobará los resultados obtenidos y tomará decisiones en cuanto a la utilización o no del sistema realizado.

Estándares de implementación.

El código a de ser desarrollado siguiendo los estándares de desarrollo para facilitar su lectura y modificación por cualquier miembro del equipo de desarrollo.

Es decisiva, para poder plantear con éxito la propiedad colectiva del código. Ésta sería impensable sin una codificación basada en estándares que haga que todo el mundo se sienta cómodo con el código escrito por cualquier otro miembro del equipo.

Desarrollar unidad de pruebas.

Cuando los test son creados antes que el código, la implementación del código será mucho más rápida. El tiempo empleado en desarrollar un test y algo de código para probarlo es aproximadamente el mismo tiempo que se emplea en crear exclusivamente dicho código. La creación de las unidades de test ayuda al programador a tener una visión a cerca del cómo, en definitiva, del comportamiento del programa. Además, aún se está a tiempo de dar marcha atrás, ya que el programador no a concluido la implementación. Esta manera de trabajar resulta especialmente beneficiosa en el diseño de complicados sistemas software.

Cualquier característica de un programa para la que no haya un test automatizado, simplemente no existe.

Programación en Parejas

Todo el código que forma parte del plan, será desarrollado por dos personas que trabajarán de forma conjunta en un ordenador. De esta manera, se incrementa la calidad del software desarrollado sin afectar al tiempo de entrega. Se parte de la idea de que este equipo de dos personas posee unos conocimientos similares en cuanto a la tarea

que van a realizar, es decir, están aproximadamente al mismo nivel. Mientras uno de ellos se encarga de pensar la táctica con la que se va a abordar el problema, el otro se encarga de pensar las estrategias que permiten llevar dichas tácticas a su máximo exponente. Ambos roles son intercambiables.

Integración del Código

En esta etapa pueden aparecer problemas debidos a la integración de los módulos que se han desarrollado y no han sido testeados todavía. Las unidades de test se encargarán de verificar la corrección de dichos módulos. Estos test deberán ser completos, en el sentido de que, un fallo el los mismo podría derivar que determinados errores pasaran inadvertidos.

Código Colectivo

Esta filosofía nos permite que cualquiera contribuya al desarrollo de cualquier parte del proyecto. Cualquier programador podrá cambiar una línea de código para añadir funcionalidad o eliminar algún fallo. Cualquiera puede realizar un cambió en el mismo siempre y cuando beneficie a la arquitectura del mismo. La responsabilidad del funcionamiento recaerá sobre el equipo al completo.

40 horas semanales

Trabajar horas extras absorbe el espíritu y la motivación del equipo. Aquellos proyectos que requieren horas extras para acabarse a tiempo pueden convertirse en un problema. En lugar de esto, se utilizan las conocidas reuniones de plan de entregas para cambiar los objetivos del proyecto. También es una mala idea incorporar nueva gente al proyecto, una vez que este ya ha comenzado. Se trabajará un máximo de 40 horas semanales. Nadie es capaz de trabajar 60 horas a la semana y hacerlo con calidad.(ESCRIBANO. 2002)

2.5.4 Prueba.

Pruebas del sistema

¿Dónde encajan las pruebas del sistema en XP?

Uno de los pilares de la eXtreme Programming es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

Las pruebas del sistema tienen como objetivo verificar la funcionalidad del sistema a través de sus interfaces externas comprobando que dicha funcionalidad sea la esperada en función de los requisitos del sistema. Generalmente las pruebas del sistema son desarrolladas por los programadores para verificar que su sistema se comporta de la manera esperada, por lo que podrían encajar dentro de la definición de pruebas unitarias que propone XP.

Sin embargo, las pruebas del sistema tienen como objetivo verificar que el sistema cumple los requisitos establecidos por el usuario por lo que también pueden encajar dentro de la categoría de pruebas de aceptación.

Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente por esto, el cliente es la persona adecuada para diseñar las pruebas de aceptación.

Las pruebas de aceptación.

El objetivo de estas pruebas es verificar los requisitos, por este motivo, los propios requisitos del sistema son la principal fuente de información a la hora de construir las pruebas de aceptación.

Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a testear cuando una historia de usuario ha sido correctamente implementada.

Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento y no se considera completa hasta que no supera sus pruebas de aceptación. Esto significa que debe desarrollarse un nuevo test de aceptación para cada iteración o se considerará que el equipo de desarrollo no realiza ningún progreso.

Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones a cerca de las mismas.

La garantía de calidad es una parte esencial en el proceso de XP. La realización de este tipo de pruebas y la publicación de los resultados debe ser los más rápido posibles, para que los desarrolladores puedan realizar con la mayor rapidez los cambios que sean necesarios.

Código y unidad de pruebas

Las unidades de test o pruebas constituyen unos de los pilares básicos de la Extreme Programming (XP).

Uno de los errores que se suele cometer es pensar que podemos dejar la construcción de los test para los últimos meses en la realización de un proyecto. Descubrir todos los

errores que pueden aparecer lleva tiempo, y más si se deja la depuración de todos para el final.

Las unidades de test están directamente relacionadas con el concepto de posesión del código. En cierta manera, una parte del código no será reemplazado si no supera los test que existen para ese código.

Después de cada modificación, se pueden emplear los test para verificar que un cambio en la estructura no introduce un cambio en la funcionalidad. Sin embargo, si se añaden nuevas capacidades al código, se tiene que rediseñar la unidad de test, para adaptarse a la nueva funcionalidad, de esta manera, la probabilidad de que exista un fallo en ambos (test y código) es menor.

Si se crean los test después de la creación del código habría que hacerlos utilizando ese código como un generador, se reubican los fallos del uno al otro. De aquí la importancia de la creación de las unidades de prueba antes que el código, para que sea independiente de este

Las pruebas se convierten en una herramienta de desarrollo, no un paso de verificación que puede despreciarse si a uno le parece que el código está bien.

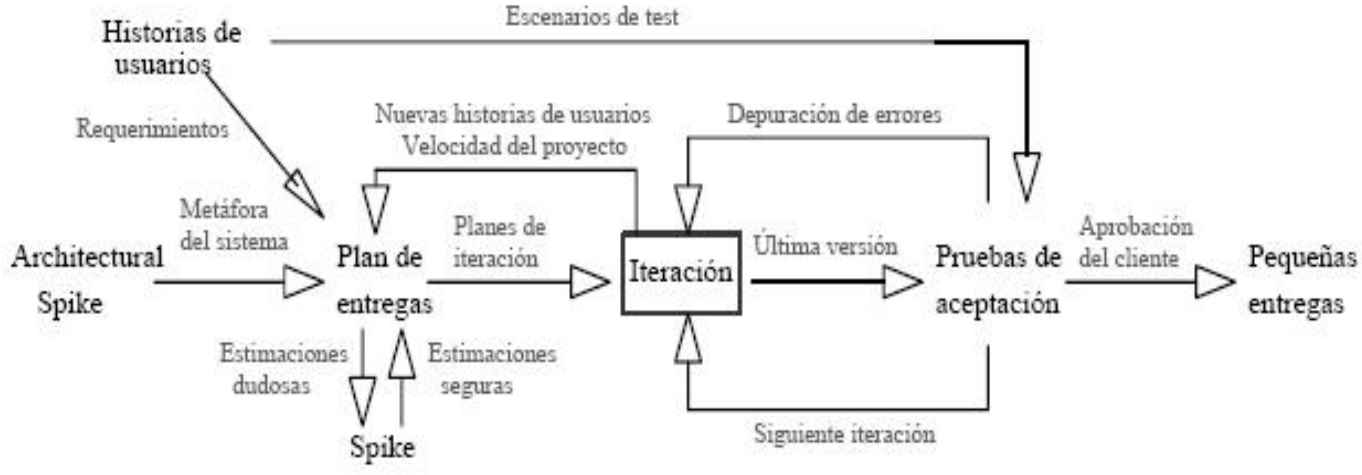


Figura 13: Representación de la metodología completa

Conclusiones:

La Metodología XP ha levantado gran revuelo en la comunidad de ingeniería del software. Extreme Programming surgió como contrapartida a las metodologías clásicas o tradicionales, añadiendo una nueva forma de hacer las cosas.

Las metodologías tradicionales para el desarrollo del software imponen un proceso disciplinado con el objetivo de hacer el trabajo más predecible, eficiente y planificado. El único inconveniente que se le ha atribuido es de ser “orientadas a documentos”, es decir, demasiado burocráticas. Las metodologías livianas o ágiles, como XP, están orientadas a las personas más que a los procesos.

Hay que tener en cuanta que en el desarrollo del software, el diseño es la actividad predominante (en XP el diseño es una actividad diaria). XP supone:

• Las personas son claves en los procesos de desarrollo de software.

• Los programadores son profesionales responsables, no requieren de supervisión.

• Los procesos se aceptan y acuerdan, no se imponen.

• Desarrolladores y gerentes comparten el liderazgo del proyecto.

• El trabajo de los desarrolladores con las personas que tienen la experiencia en el negocio es regular, no eventual.

Conviene recordar que ninguna metodología hará el trabajo por ti, porque ninguna metodología trabaja sola.

CAPÍTULO 3

RESPUESTA A LA SOLUCION PROPUESTA

En este capítulo se aplica la Metodología XP a un caso de estudio, que corresponde a una Hipermedia que tendrá un carácter educativo y se utilizará como herramienta de apoyo en la enseñanza de la asignatura de Topografía de la Carrera de Ingeniería Civil en la Facultad de Ingeniería Civil del Centro Universitario José Antonio Echeverría (CUJAE).

Caso de estudio.

En la Facultad de Ingeniería Civil del Centro Universitario José Antonio Echeverría (CUJAE) tiene entre sus asignaturas fundamentales la Topografía.

Esta misma ha presentado problemas con el nivel de asimilación del estudiante en las diferentes ramas de esta asignatura así como el uso de los diferentes equipos de medición y nivelación del terreno, por lo que se nos ha dado la tarea de realizar un software educativo para el apoyo en la enseñanza de esta asignatura.

Se pretende realizar una Multimedia donde se abarca la información referente a la carrera de Ingeniería Civil, el Programa de disciplina de Topografía que se imparte, los contenidos referentes a la asignatura de Topografía y los métodos de comprobación, ejercicios, etc. También toda la bibliografía perteneciente a los contenidos de la asignatura y a los documentos que complementen el estudia de la misma.

Aplicación de la Metodología XP al caso de estudio.

La metodología cuenta de cuatro etapas:

1-Planificación.

2-Diseño.

3- Desarrollo.

4- Prueba.

Se desarrollara la etapa de Planificación, siendo esta la más importante en el desarrollo de la metodología, ya que cuenta con una serie de actividades que dan como resultado una primera versión funcional del producto.

Esta etapa se apoya en un conjunto de recomendaciones y premisas que se establecen en las restantes etapas para la obtención de un producto final que cubra todas las expectativas esperadas por parte del cliente.

A lo largo del proyecto cada integrante del equipo de trabajo debe llevar un Registro de Tiempo para cada actividad que realice, para esto XP propone la creación de la siguiente plantilla:

Proyecto:
Equipo :

Fecha (dd/mm)	Rol	Hora Inicio	Hora Fin	Tiempo Interrupciones (en horas)	Actividad Realizada*

Tabla 3: Representación del Registro de Tiempo

Esta actividad permite llevar un control del tiempo que se toma realizar cada actividad desarrollada por un rol en específico, de esta forma la organización dentro del equipo de trabajo fluye con mayor facilidad.

3.1 Planificación.

Historias de Usuario.

1- Realizar introducción a la Multimedia.

2- Seleccionar tema.

3- Seleccionar Clase.

4- Mostrar información de la clase.

5- Obtener materiales de la clase.

6- Resolver ejercicios.

Historia de Usuario	
Número:1	**Nombre Historia de Usuario:** Realizar introducción a la multimedia
Modificación de Historia de Usuario Número:	
Usuario: Estudiante	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:**
Riesgo en Desarrollo: (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: Inicia cuando el usuario accede a la Multimedia, de ahí comienza la presentación particular de la misma en la cual se darán los consejos de cómo navegar por el producto y además le da la opción al estudiante de conocer la información referente a la Carrera de Ingeniería Civil y el Programa de disciplina de Topografía que se imparte. En esta primera pantalla el estudiante tiene la posibilidad de elegir entre los módulos principales: Planimetría y Altimetría, los cuales constituyen la base fundamental de la Multimedia Topografía.	

Tabla 4: Representación de la Historia de Usuario # 1

Historia de Usuario	
Número:2	**Nombre Historia de Usuario:** Seleccionar tema.
Modificación de Historia de Usuario Número:	
Usuario: Estudiante	**Iteración Asignada:** 2
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:**
Riesgo en Desarrollo: (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: Inicia cuando el estudiante escoge la opción Seleccionar tema, mostrándose una pantalla con los mismos, el usuario selecciona el tema que desea estudiar.	

Tabla 5: Representación de la Historia de Usuario # 2

Historia de Usuario	
Número:3	**Nombre Historia de Usuario:** Seleccionar clase.
Modificación de Historia de Usuario Número:	
Usuario: Estudiante	**Iteración Asignada:** 2
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:**
Riesgo en Desarrollo: (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: Inicia cuando el estudiante luego de haber accedido al tema que desea, se muestra una pantalla con una serie de clases y decide ver una clase especifica.	

Tabla 6: Representación de la Historia de Usuario # 3

Historia de Usuario	
Número:4	**Nombre Historia de Usuario:** Mostrar Información de la clase
Modificación de Historia de Usuario Número:	
Usuario: Estudiante	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:**
Riesgo en Desarrollo: (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: Inicia cuando el estudiante accede a la clase que desea estudiar donde le aparece en pantalla las distintas opciones que le brinda dicha clase: Orientaciones de la clase, Ejercicios a resolver y Materiales Complementarios	

Tabla 7: Representación de la Historia de Usuario # 4

Historia de Usuario	
Número:5	**Nombre Historia de Usuario:** Obtener materiales de la clase.
Modificación de Historia de Usuario Número:	
Usuario: Estudiante	**Iteración Asignada:** 3
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:**
Riesgo en Desarrollo: (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: Inicia cuando el estudiante decide obtener materiales de la clase que haya escogido ya sea un Documento Word o un PPT. , donde se muestra una pantalla con ambos materiales para que el usuario pueda descargarlos**.**	

Tabla 8: Representación de la Historia de Usuario # 5

Historia de Usuario	
Número:6	**Nombre Historia de Usuario:** Resolver Ejercicios
Modificación de Historia de Usuario Número:	
Usuario: Estudiante	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:**
Riesgo en Desarrollo: (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: Inicia cuando el estudiante selecciona el temario de ejercicio a resolver y le aparece en pantalla el contenido de dicho temario, el estudiante da respuesta a las preguntas hasta terminar la ultima, obteniendo una calificación de B, R o M, según la cantidad de respuestas correctas que sea capaz de responder.	

Tabla 9: Representación de la Historia de Usuario # 6

Plan de Entrega

Una vez que se concluye la tarea por parte del cliente de elaborar las distintas historias de usuario, se comienza con la creación del Plan de Entregas.

El mismo se hace con la intención de que los programadores obtengan una estimación de dicha historia en cuanto al nivel de detalle, o sea, para fijar el periodo de tiempo que se puede tardar en la implementación de una historia, o para determinar si el grado de dificultad es mínimo, en este caso la historia pasa a formar parte de otra.

El Plan de Entrega posibilita la obtención de una clasificación teniendo en cuenta el riesgo que se corre a la hora de implementar la historia.

Estos datos se almacenan en campos que permanecen vacíos en la historia de usuario, el responsable de llenar estos datos es únicamente el programador una vez que haya hecho el análisis requerido de los mismos.

Plan de Iteración

Historias de Usuarios divididas en tareas.

Cada una de estas historias de usuario se transformará en tareas que serán desarrolladas por programadores, dentro del equipo de desarrollo, aplicando la práctica de la Programación en parejas. Cada tarea de desarrollo corresponderá a un periodo de uno a tres días de desarrollo.

Historia de Usuario	Tareas
Realizar Introducción a la Multimedia	- Crear template de diseño. - Crear pantalla de presentación. - Crear pantallas iniciales. - Escribir el contenido de las pantallas. - Crear y actualizar los vínculos entre las pantallas.
Mostrar Información de la clase.	- Mostrar orientación de la clase. - Permitir descarga de Materiales Complementarios. - Mostrar ejercicios de la clase. - Crear y actualizar los vínculos entre las pantallas.
Resolver Ejercicios.	- Mostrar preguntas. - Permitir introducción de las respuestas. - Evaluar respuestas. - Asignar calificación. - Implementar un sistema de mensajes reflexivos. - Crear y actualizar los vínculos entre las pantallas.

Tabla 10: Distribución de las tareas por cada Historia de Usuarios

Tareas detalladas.

Tarea de Ingeniería	
Número Tarea: 1	**Historia de Usuario (Nro.1):** Realizar introducción a la multimedia.
Nombre Tarea: Crear template de diseño.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora)	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción: Esta tarea facilita la creación de una plantilla estándar para todas las pantallas de la Multimedia. De la misma se podrá acceder a las pantallas: Como navegar por la Multimedia, Plan de Disciplina de la asignatura Topografía e Información referente a la carrera de Ingeniería Civil.	

Tabla 11: Descripción de la Tarea de Ingeniería

Tarea de Ingeniería	
Número Tarea: 3	**Historia de Usuario (Nro.1):**Realizar introducción a la multimedia
Nombre Tarea: Crear pantallas iniciales.	
Tipo de Tarea : Desarrollo. Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción: Una vez que accede a la multimedia, el estudiante se encuentra con un menú donde se muestran los dos módulos principales de la multimedia: Altimetría y Planimetría, dándole la posibilidad de acceder al que le interese.	

Tabla 12: Descripción de la Tarea de Ingeniería

Tarea de Ingeniería	
Número Tarea: 1	**Historia de Usuario (Nro.4):** Mostrar información de la clase.
Nombre Tarea: Mostrar orientación de la clase.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora)	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción: Una vez que el estudiante seleccionó la clase, en pantalla aparecerá la opción de mostrar al estudiante las orientaciones de la clase, con el objetivo de que el mismo conozca en resumen de que trata esa clase, además de cuáles son los objetivos de la misma.	

Tabla 13: Descripción de la Tarea de Ingeniería

Tarea de Ingeniería	
Número Tarea: 2	**Historia de Usuario (Nro.4):** Mostrar información de la clase.
Nombre Tarea: Permitir descarga de Materiales Complementarios.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora)	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción: Una vez que el estudiante seleccionó la clase, en pantalla aparecerá una opción que le permite acceder a los materiales complementarios de la clase, con el objetivo de que el mismo pueda llevarse el contenido de la clase en un Documento Word o en un PPT y de esta forma , ejercitar sus conocimientos	

Tabla 14: Descripción de la Tarea de Ingeniería

Tarea de Ingeniería	
Número Tarea: 3	**Historia de Usuario (Nro.4):** Mostrar información de la clase.
Nombre Tarea: Mostrar ejercicios de la clase.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora)	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción: Esta tarea permite la creación de un sistema de evaluación para los estudiantes. En dependencia de los ejercicios que el estudiante pueda contestar en el cuestionario de ejercicios interactivos, se le asigna una evaluación de B, R o M, en dependencia de la cantidad que estén bien, regular o mal. De esta forma el estudiante puede medir sus conocimientos y sacar sus propias conclusiones.	

Tabla 15: Descripción de la Tarea de Ingeniería

Tarea de Ingeniería	
Número Tarea:2	**Historia de Usuario (Nro.6):** Resolver Ejercicios**.**
Nombre Tarea: Permitir introducción de respuestas.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora)	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción: Esta tarea da la posibilidad al estudiante de dar respuesta a los ejercicios propuestos, a través de la existencia de campos vacíos designados para ello.	

Tabla 16: Descripción de la Tarea de Ingeniería

Tarea de Ingeniería	
Número Tarea:3	**Historia de Usuario (Nro.6):** Resolver Ejercicios**.**
Nombre Tarea: Evaluar ejercicios	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora)	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción: Se establece una clave, donde, en dependencia de la cantidad de preguntas que tenga el cuestionario y la cantidad de preguntas que el estudiante responda bien , regular o mal , se evalúa el ejercicio de B, R o M	

Tabla 17: Descripción de la Tarea de Ingeniería

Tarea de Ingeniería	
Número Tarea:4	**Historia de Usuario (Nro.6):** Resolver Ejercicios**.**
Nombre Tarea: Asignar calificación.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora)	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción: Una vez que el estudiante da respuesta a todas las preguntas , se muestra en pantalla la calificación asignada por el sistema	

Tabla 18: Descripción de la Tarea de Ingeniería

Tarea de Ingeniería	
Número Tarea:5	**Historia de Usuario (Nro.6):** Resolver Ejercicios.
Nombre Tarea: Implementar un sistema de mensajes reflexivos.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora**)**	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción: Esta tarea es un sistema que se implementa para mostrar en pantalla mensajes que animen a los estudiantes en la realización del cuestionario interactivo.	

Tabla 19: Descripción de la Tarea de Ingeniería

Plan de la Iteración 1

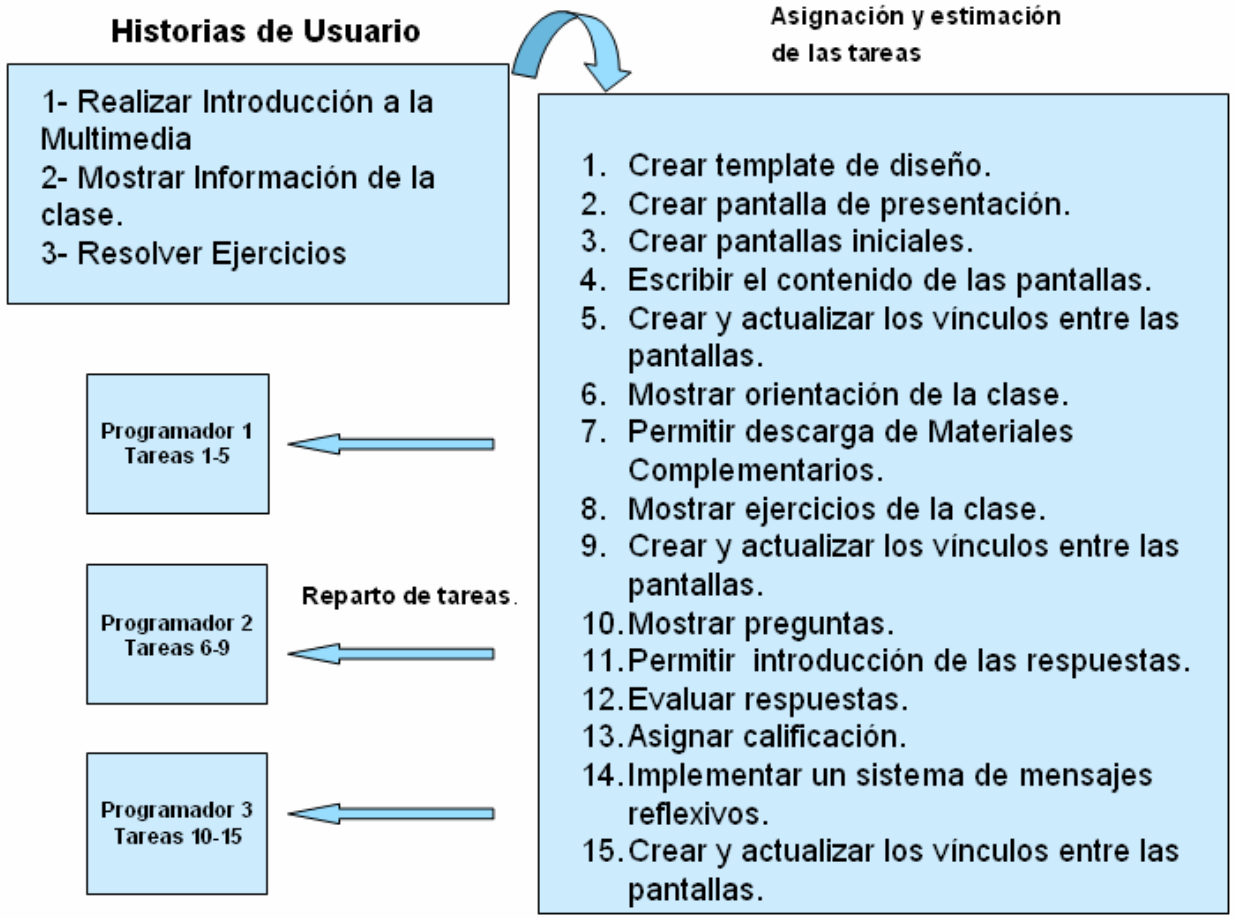


Figura 14: Descripción del Plan de Iteración # 1

Tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración).

Las tarjetas determinan el comportamiento de cada actividad. En la Multimedia cada pantalla se comporta como un objeto independiente, de esta forma nuestra multimedia esta formada por las siguientes clases:

1-Presentación.
2-Ayuda.
3-Módulos.
4-Tema
5-Clase.
6-Ejercicios

Clase: Ayuda

Responsabilidades	Clases relacionadas
Mostrar información auxiliar al estudiante	Presentación
Mostrar consejos de navegación.	Presentación

Tabla 20: Descripción de la CRC Ayuda

Clase: Módulo

Responsabilidades	Clases relacionadas
Mostrar pantalla especifica de cada módulo.	Presentación
Acceder a materiales complementarios.	Presentación

Tabla 21: Descripción de la CRC Módulo

Clase: Clase

Responsabilidades	Clases relacionadas
Mostrar orientaciones de la clase.	Tema, Módulo
Acceder a materiales complementarios de la clase.	Tema, Módulo
Realización de un cuestionario interactivo de ejercicios de la clase.	Tema, Módulo

Tabla 22: Descripción de la CRC Clase

Clase: Ejercicios

Responsabilidades	Clases relacionadas
Visualizar contenido del ejercicio.	Clase, Tema, Módulo
Dar por terminado el ejercicio, después de dar respuesta a todas las preguntas.	Clase, Tema, Módulo
Evaluar respuestas del ejercicio.	Clase, Tema, Módulo
Permitir sólo una oportunidad para la respuesta a las preguntas de los ejercicios.	Clase, Tema, Módulo
Permitir resolver otras preguntas, solo después de haber terminado la que se está resolviendo.	Clase, Tema, Módulo
Mostrar mensajes reflexivos que motiven al estudiante durante la realización del ejercicio.	Clase, Tema, Módulo

Tabla 23: Descripción de la CRC Ejercicios

3.2 Desarrollo.

Se crea el prototipo de interfaz de usuario, para ello la multimedia va a estar formada por una serie de pantallas:

1- Pantalla: Bienvenidos a la Multimedia , donde se muestran los botones de Como Navegar, Módulos Principales, Que es Ingeniería Civil y Programa de Disciplina.

2- Pantalla: Altimetría, se muestran los botones Temas y Materiales Complementarios.

3- Pantalla Planimetría: contiene lo mismo que la pantalla Altimetría.

4- Pantalla Clases: muestra los botones Orientaciones de la Clase, Materiales Complementarios y Ejercicios.

3.3 Prueba.

Cada historia de usuario está asociada a una prueba de aceptación, conocidas también como pruebas funcionales, las mismas se realizan en esta etapa del proyecto y en ellas se describen las posibles formas de utilización del software.

En estos documentos de prueba se indican las posibles respuestas que tiene el software en la utilización de cada funcionalidad, así como los posibles mensajes de error, información o de aceptación que emite el software cuando se utiliza dicha funcionalidad.

Pasos para escribir pruebas de aceptación.

1-Identificar todas las acciones en la historia.
2-Para cada acción escribir dos pruebas.
3- Para algunos datos, remplazar las entradas que hacen que la acción ocurra y llenar en la casilla Resultados esperados los resultados obtenidos.
4-Para otros datos, remplazar las entradas que hacen que la acción falle, y registrar los resultados.

Se propone redactar tablas donde cada línea es una prueba con tres columnas: la acción a probar, los datos de prueba y el resultado esperado.

	Acción a probar	**Datos de prueba**	**Resultados esperados**
Prueba			

Tabla 24: Representación de una Prueba de Aceptación

3.4 Resultados Obtenidos

Después de realizar un estudio profundo de la metodología propuesta, se han obtenidos resultados favorables que avalan la utilización de la misma en el proceso de desarrollo de software educativo.

La metodología XP proporciona una mejor organización dentro del equipo de trabajo garantizando un rendimiento máximo por parte de cada integrante, rompe con la suposición establecida en la industria del software de que el coste de los cambios en un software crece exponencialmente con el tiempo, planteando que con una buena aplicación de las prácticas, el coste del cambio decrece. Permite además la creación de un software de alta calidad en un tiempo fiable y responde en todo momento a los intereses del cliente, logrando que se cumplan todas sus expectativas.

Luego de haber estudiado las características para lograr una buena comunicación entre el equipo de desarrollo, se comprendió que un inconveniente de la Metodología podría ser la no aplicación de manera eficiente de uno de los valores que propone la misma, la comunicación, es decir, sino se logra que en el equipo de trabajo la comunicación fluya como lo requiere la metodología, pues esto se verá como una desventaja que incidirá de manera negativa en el proceso de desarrollo del software.

Se puede ver la disponibilidad del cliente como otro inconveniente de la metodología, pues el mismo tendrá que estar disponible como otro miembro del equipo de trabajo durante el desarrollo de todo el proyecto. Lo que a primera vista puede ser un tabú, se convierte en una de las grandes ventajas que propicia el uso de XP, garantiza a toda costa el cumplimiento del principal objetivo de la metodología, satisfacer todas las necesidades del cliente

CONCLUSIONES GENERALES

En el presente trabajo se ha propuesto la utilización de la Metodología XP en el proceso de producción de una multimedia educativa para el Instituto Superior Politécnico José Antonio Echeverría.

Se logró aplicar en el proceso de desarrollo del software educativo Topografía las etapas que plantea dicha metodología, obteniendo resultados favorables fundamentalmente en la etapa de la Planificación.

La propuesta mejora en organización, rendimiento, administra tiempos de desarrollo y colabora en el establecimiento de un modelo para proyectos multimedia, así como encamina la producción en la búsqueda de la certificación de software.

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 educativo que sean de corto plazo, con un equipo de desarrollo de 10 a 12 integrantes.
- Realizar un estudio y análisis mas completo de las fases Implementación y Prueba, las cuales no fueron profundamente abordadas 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 educativos 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

BECK., K. "Extreme Programming Explained. Embrace Change.,." 1999.

ESCRIBANO., G. F. "eXtreme Programming / Programación Extrema." 2002.

FERRER, G. R. Y. J. "Programación eXtrema y Software Libre." 2002.

GRAELLS., D. P. M. .MULTIMEDIA EDUCATIVO: CLASIFICACIÓN, FUNCIONES, VENTAJAS E INCONVENIENTES., 1999.

HERNÁN., S. M. Tesis de Grado en Ingeniería en Informática., 2004.

JACOBSON, I. "Applying UML in The Unified Process. Rational Software." 1998.

MADDISON, R. N. "Information System methodologies." 1983.

MARQUÈS, P. "METODOLOGÍA PARA LA ELABORACIÓN DE SOFTWARE EDUCATIVO ". 1995.

PIATTINI, M. "Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión." 1996.

PRESSMAN. "Software Engineering Resources." 2006.

BIBLIOGRAFÍA

[Beck 2000] Kent Beck. *Extreme Programming Explained: Embrace Chang*e. Addison Wesley Longman, 2000. *http://www.extremeprogramming.org/*

[Beck 1999] Kent Beck. *Embracing Change with Extreme Programmin*g. Computer (revista de la *IEEE Computer Societ*y).Vol. 32, No. 10. Octubre1999, pp. 70-77

Manuel Calero.*Una explicación de la programación extrema XP. Apolo Software*

José H. Canós, Patricio Letelier yMª Carmen Penadés. *Métodologías Ágiles en el Desarrollo de Software. DSIC -Universidad Politécnica de Valencia*
Camino de Vera s/n, 46022 Valencia.

Gerardo Fernández Escribano. *Introducción a Extreme Programming. Ingeniería del Software II. 9-12-2002.*

Patricio Letelier y Mª Carmen Penadés. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)***.** *Universidad Politécnica de Valencia Camino de Vera s/n, 46022 Valencia.*

Manuel Calero Solís. *Una explicación de la programación extrema (XP). V Encuentro usuarios xBase 2003 MADRID .http://www.apolosoftware.com.*

Gregorio Robles, Jorge Ferrer. *Programación eXtrema y Software Libre. Universidad Rey Juan Carlos, Universidad Politécnica de Madrid. 12 de octubre de2002.*

Martin Fowler. *Chief Scientist. La Nueva Metodología. abril de 2003. http://www.martinfowler.com/articles/designDead.html.*

Jose Mª Cubel Navarro .*(2003.) eXtreme Programming. Laboratorio de Sistemas de Información Facultad de Informática. Universidad Politécnica de Valencia.*

Patricio Letelier. *Introducción al Proceso de Desarrollo de Software. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia*

Patricio Letelier. *Metodologías Ágiles y XP. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. http://www.dsic.upv.es/-letelier/pub.*

Taylor, R. *(1980). "The computer in the school: tutor, tool, tutel."*
Sánchez, J. *(1999). "Construyendo y Aprendiendo con el Computador."*
Pressman. (*2006). "Software Engineering Resources."*

Piattini, M. *(1996). "Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión."*

Marquès,P.*(1995). "METODOLOGÍA PARA LA ELABORACIÓN DE SOFTWARE EDUCATIVO "***.**

Maddison, R. N. *(1983). "Information System methodologies."*

Jacobson, I. *(1998). "Applying UML in The Unified Process. Rational Software."*

Juzgado, J. *(1996). "Procesos de construcción del software y ciclos de vida."*

Gros, B. *(1997). "Diseños y programas educativos."*

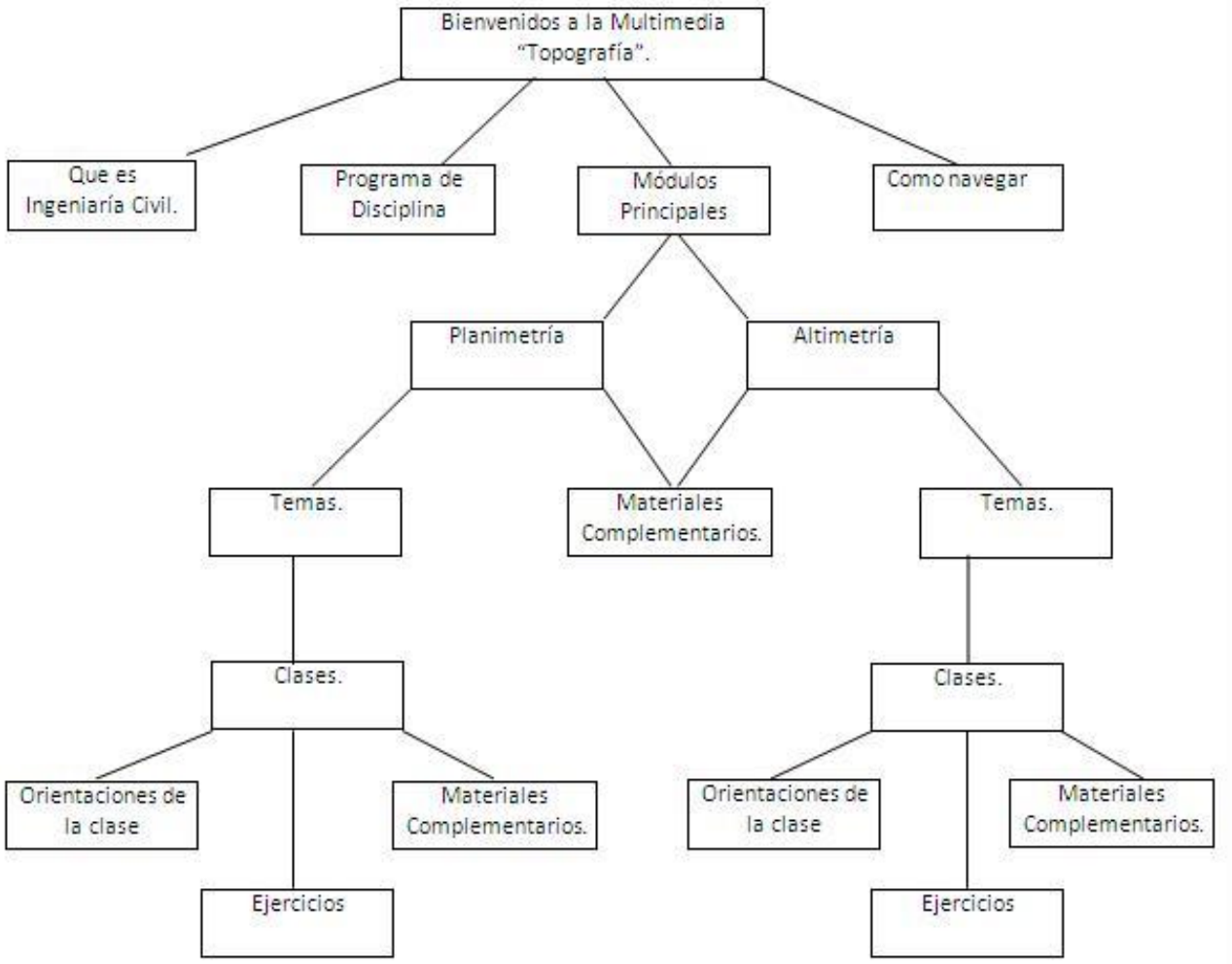
Kemmis, S. *(1976). "The Educational Potential of Computer Assisted learning: Qualitative Evidence about Student Learning."*

Dr. Perez Marquès Graells. *1999 (última revisión: 22/12/04) .MULTIMEDIA EDUCATIVO: CLASIFICACIÓN, FUNCIONES, VENTAJAS E INCONVENIENTES. Departamento de Pedagogía Aplicada, Facultad de Educación*

Schenone Marcelo Hernán. *- 2004 -.Tesis de Grado en Ingeniería en Informática. Facultad de Ingeniería.Universidad de Buenos Aires.*

ANEXOS

Diagrama de navegación.



Un día de trabajo con XP.

Como funciona la programación en parejas.

Libros sobre Metodologías Ágiles de Desarrollo, específicamente XP.

Entrevista

En la Universidad actualmente se encuentra trabajando el proyecto Paseo Virtual, el cual utiliza para el desarrollo de una maqueta virtual la metodología XP.

En una entrevista realizada a Frank Puig Placeres, actual líder de proyecto, pudimos comprobar la funcionalidad de la metodología e investigar como ha influido en el colectivo de trabajadores la aplicación de esta nueva metodología.

La experiencia para el equipo ha sido muy positiva, XP les ha brindado la posibilidad de ser ágiles, ligeros, mucho mas rápidos en cada tarea que realizan, siempre manteniendo los estándares de calidad muy por encima del nivel requerido.

Se comprobó que la Metodología XP es fiable y además muy flexible a la hora de utilizarla, pues hay quien no planifica absolutamente nada y se centra en lo que es obtener versiones funcionales de un producto en el menor tiempo posible (es el caso de este proyecto) y están los que por otra parte mantienen esa filosofía, pero llevando a cabo el juego de la planificación.

Nuestro trabajo investigativo ha servido para documentar al líder del proyecto Paseo Virtual , ya que el mismo en su trabajo aplica la metodología eficazmente obteniendo resultados satisfactorios, sin embargo , no deja de reconocer que las buenas practicas planteadas en el capitulo 2 son un pilar fundamental que hace de XP una metodología mucho mas atractiva.

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 hacerlo.

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 .

Software Educativo (SWE.): Se define como un programa automatizado diseñado con una intencionalidad educativa para ser utilizado en el proceso de aprendizaje, utiliza procedimientos para que el estudiante aprenda, se fomenta el análisis de problemas, facilita el trabajo en grupo, provee soporte en actividades docentes y en el sentido más amplio, mejora las habilidades del pensamiento y la resolución de problemas.

Pantalla: Es la agrupación visual de elementos de medias contenidas en una vista determinada.

Multimedia. Es un sistema que utiliza más de un medio de comunicación al mismo tiempo en la presentación de la información, como texto, imagen, animación, vídeo y sonido.

Hipertexto: Un hipertexto es un documento digital o no, que se puede leer de manera no secuencial. Un hipertexto tiene los siguientes elementos: secciones, enlaces o hipervínculos y anclajes.

Herramienta CASE: Conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software.

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.

TIC: Tecnologías de la Información y las Comunicaciones.

IEEE: Corresponde a las siglas del Instituto de Ingenieros Eléctricos y Electrónicos,. Es una asociación estadounidense dedicada a la estandarización internacional sin fines de lucro formada por profesionales de las nuevas tecnologías.

ISO-9000: Es un conjunto de normas de calidad establecidas por la Organización Internacional para la Estandarización (ISO) que se pueden aplicar en cualquier tipo de organización .

CMM: El Modelo de Capacidad y Madurez o CMM (Capability Maturity Model), es un método de definir y gestionar los procesos a realizar por una organización.

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 guiado por los casos de uso. Incluye artefactos y roles

Programacion 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.

Feature Driven Development (FDD): Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software.

Adaptive Software Development (ASD): Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de

vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente.

Feature-Driven Development (FDD) : Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas. Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software.

Lean Development (LD): En LD, los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios.

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.