ESTUDIO COMPARATIVO DE METODOLOGÍAS DE DESARROLLO DE SOFTWARE

José Gaspar Arteaga Camacho

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2014

ESTUDIO COMPARATIVO DE METODOLOGÍAS DE DESARROLLO DE SOFTWARE

José Gaspar Arteaga Camacho

Trabajo de Grado presentado como requisito parcial para optar al título de Ingeniero de Sistemas

Director:
MSc. Alexander Barón Salazar

Co-Director:
Esp. Sandra Vallejo Chamorro

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2014

NOTA DE RESPONSABILIDAD

“Las ideas y conclusiones aportadas en el siguiente trabajo son responsabilidad exclusiva del autor”.

Artículo 1, acuerdo No. 324 de octubre 11 de 1966 emanado del honorable Consejo Directivo de la Universidad de Nariño.

“La Universidad de Nariño no se hace responsable de las opiniones o resultados obtenidos en el presente trabajo y para su publicación priman las normas sobre el derecho de autor”.

Artículo 13, acuerdo No. 005 de 2010 emanado del Honorable Consejo Académico.

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

Jurado:

_____

Jurado:

San Juan de Pasto, Junio de 2014

AGRADECIMIENTOS

A DIOS por regalarme cada día de mi vida permitiendo enriquecer mi conocimiento y cumplir cada una de mis metas.

A LA UNIVERSIDAD DE NARIÑO, por ser el recinto donde se adquiere y genera nuevo conocimiento, brindando la oportunidad de ser profesional idóneo y humano, con el propósito de ayudar a la sociedad.

Al Programa INGENIERÍA DE SISTEMAS, por permitirme adquirir sabiduría.

Al Magister ALEXANDER BARON, Docente y Asesor, quien con su empeño y sabiduría se hizo posible esta meta.

A Sebastián, Omar, Mateo, Mauricio, Rider, German, amigos incondicionales desde el inicio de la carrera.

A la Especialista SANDRA VALLEJO, docente y Co-Asesora, quien con su apoyo y sabiduría se hizo posible esta meta.

A MIS PROFESORES, por compartir sus conocimientos enriqueciendo mi saber y el formarme como un profesional idóneo.

DEDICATORIA

Los esfuerzos, cada meta cumplida se la dedico a DIOS por darme cada día de mi vida y poder alcanzar estos logros.

A Mixi Camacho mi madre, ejemplo de Bondad y Trabajo, por sus consejos, por el apoyo incondicional en cada uno de los momentos de mi carrera, a Francisco Arteaga mi padre, por el apoyo en mi carrera.

A Jozsue Vilamar, José Camacho, Elihud Basante, mis primos por sus palabras de aliento, consejos y el nunca rendirse a pesar de los obstáculos.

Y a todas las personas quienes hicieron de este sueño una realidad, les agradezco de todo corazón.

José Gaspar Arteaga Camacho

RESUMEN

Al momento de realizar el proceso de desarrollo de software, se ve la necesidad de seguir una metodología que se adapte de mejor manera a dicho proceso. Al no tener una investigación previa, referente al comportamiento de algunas metodologías en el proceso de desarrollo, se generan demoras en el tiempo de ejecución.

Con la existencia de esta investigación, la persona o grupo de desarrolladores podrá identificar según criterios y espacios de desarrollo, que metodología es la más adecuada para tomar como referencia, haciendo que sea ágil al momento de seleccionar la metodología e iniciar con el proceso de desarrollo.

ABSTRACT

At the time of the software development process, is the need to follow a methodology that fits better to the process. Having no prior investigation concerning the behavior of some methodologies in the development process, delays are generated in runtime.

With the existence of this research the person or group of developers may identify criteria and areas of development, which methodology is the most appropriate to make reference, making it agile when selecting the methodology and initiate the development process.

CONTENIDO

ANEXOS

Tabla de Ilustraciones

INTRODUCCIÓN

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software.

Generalmente, el proceso de desarrollo lleva asociado un marcado énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada. Dichas metodologías a la hora de abordar el desarrollo de software han demostrado ser efectivas y necesarias en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de cumplimiento en el proceso para así mantener su calidad.

Dichas metodologías, pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan alterados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software.

Por lo tanto, se podrían clasificar en dos grandes grupos:

Respecto a la clasificación de metodologías, Yanira, dice:

- ✓ Las metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas metodologías son llamadas metodologías pesadas.

- ✓ Las metodologías orientadas a la interactuación con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Estas son llamadas metodologías ligeras/ágiles[1].

[1] RUIZ, Yadira. Principales metodologías pesadas y orientadas a objetos en el desarrollo de software. 14 convención científica de ingeniería y arquitectura, México 2008. Investigación documental.

TEMA

TÍTULO DEL PROYECTO

ESTUDIO COMPARATIVO DE METODOLOGÍAS DE DESARROLLO DE SOFTWARE

LÍNEA DE INVESTIGACIÓN

La línea de investigación de este proyecto es: Software y manejo de información.

ALCANCE Y DELIMITACIÓN

La investigación se restringe al estudio comparativo de algunas metodologías de desarrollo de software importantes del momento, que serán definidas dentro del proceso de investigación, utilizando criterios de comparación que se definirán igualmente en dicho proceso y ya como producto final se entregará un reporte sobre el resultado.

MODALIDAD

Trabajo de Investigación.

DESCRIPCIÓN DEL PROBLEMA

PLANTEAMIENTO DE PROBLEMA

En la actualidad no existe localmente un estudio reconocido y vigente sobre las metodologías de desarrollo de software y específicamente del proceso de desarrollo de software, por lo tanto, los investigadores que utilizan enfoques de procesos de desarrollo de software como insumo en sus trabajos no tienen referentes para tomar una decisión acertada sobre el enfoque que más se adapta a los requisitos particulares del problema que el investigador pretende resolver.

Como también en la actualidad se cuenta con diversos tipos de metodologías para el desarrollo de software, lo que genera una gran problemática sobre cual utilizar a la hora de desarrollar un software, para esto es necesario conocer a fondo las

diversas metodologías existentes, saber cómo funciona cada una, y así poder elegir correctamente la más adecuada según la necesidad que se tenga.

FORMULACIÓN DEL PROBLEMA

¿Cómo establecer un cuadro comparativo de metodologías de desarrollo de software que oriente su uso en la academia?

SISTEMATIZACIÓN DEL PROBLEMA

- ✓ ¿Cuáles son los elementos que deben ser estudiados para establecer el estado de los procesos de desarrollo de software?
- ✓ ¿Cómo se identifican los aspectos que deben ser estudiados en cada una de las propuestas de los procesos de desarrollo de software?
- ✓ ¿Cuál es el proceso para estudiar las propuestas de desarrollo de software?
- ✓ ¿De qué manera se emplean los resultados del estudio comparativo para determinar el estado de los procesos de desarrollo de software?

OBJETIVOS

OBJETIVO GENERAL

Realizar un estudio de metodologías del proceso de desarrollo de software siguiendo unos criterios de comparación.

OBJETIVOS ESPECÍFICOS

- ✓ Identificar las propuestas de metodologías de desarrollo de software que hacen parte del estudio comparativo.

- ✓ Definir los criterios de comparación empleados para el desarrollo del estudio comparativo.

- ✓ Aplicar los criterios de comparación a cada una de las propuestas de los procesos de desarrollo de software.

- ✓ Crear el informe final del estudio comparativo de metodologías de desarrollo de software, a partir de los resultados de los criterios de comparación aplicados.

JUSTIFICACIÓN

En todo proceso de desarrollo de software y por la urgente necesidad de obtener productos de alta calidad, que satisfagan a plenitud las demandas de los clientes, se hace imprescindible el estudio a profundidad de la diversidad de metodologías de desarrollo de software que son esenciales para el éxito de los proyectos.

En tal sentido, se hace importante la elección de una metodología adecuada y ajustada al equipo de desarrollo que permita la flexibilidad necesaria para lograr los objetivos trazados al concebirse el proyecto, superando incluso las expectativas iniciales de los clientes.

Sin dudas el éxito del producto, depende en gran medida, de la acogida por el equipo de desarrollo que tenga la metodología seleccionada.

La comunidad académica se encuentra en un proceso de aprendizaje continuo, por ello, este proyecto de investigación ofrecería agilidad en otros proyectos de investigación, que utilizan ingeniería de software y metodologías de desarrollo de software como insumo para sus investigaciones.

Los procesos de desarrollo de software están estrechamente vinculados a la creación de software, que busca la construcción de nuevos sistemas informáticos a partir de componentes, especificaciones o diseños. La relación entre procesos de desarrollo de software y creación es evidente: disponer de una metodología característica de un área, hará más sencilla la creación de sistemas, módulos, componentes implicando un ahorro en el tiempo, algo muy importante el cual puede aplicarse a múltiples escenarios y en la resolución de distintos problemas.

DESARROLLO DEL PROYECTO

1. RECOPILACIÓN DE INFORMACIÓN

1.1. METODOLOGÍA DE RECOLECCIÓN DE INFORMACIÓN

La búsqueda de información se realiza con base en los elementos del problema, el planteamiento de preguntas relevantes con el fin de orientar la búsqueda de la información. Por parte del investigador y los responsables de este trabajo de grado se tiene un dominio conceptual y teórico del tema objeto de investigación.

El tipo de información recolectada es **primaria** debido a que se recogió de forma directa a través de libros y revistas científicas publicados por autores reconocidos. Encontrando información completa, descomposición y análisis detallado de la investigación, ayudando considerablemente en la valoración e interpretación de los datos.

Validez de la metodología de recolección de información

Yolanda Gallardo de Parada, Adonay Moreno Garzón.
Serie Aprender a investigar.
Módulo 3: Recolección de la información.
Instituto Colombiano para el Fomento de la Educación Superior ICFES.
ISBN: 958-9279-11-2 Obra completa
ISBN: 958-9279-14-7 Módulo 3

1.2. MODELO EN CASCADA

Como lo dice Pressman[2], al usar el modelo de cascada, se necesitaría completar un conjunto de tareas en forma de fase para después continuar con la fase próxima. El modelo en cascada trabaja perfectamente para los proyectos en los cuales los requisitos están definidos claramente y no son obligados a futuras modificaciones. Se puede monitorear fácilmente ya que asigna responsabilidades definidas.

[2] PRESSMAN , Roger S. Ingeniería del software un enfoque práctico. McGraw-Hill. México 2005. 6ª Edición. Pág. 50-51.

1.2.1. Fases del modelo

Análisis de requisitos

En esta fase se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos.

Es importante señalar que en esta etapa se debe **consensuar** todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

Diseño del sistema

Descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

Es conveniente distinguir entre diseño de alto nivel o arquitectónico y diseño detallado. El primero de ellos tiene como objetivo definir la estructura de la solución (una vez que la fase de análisis ha descrito el problema) identificando grandes módulos (conjuntos de funciones que van a estar asociadas) y sus relaciones. Con ello se define la arquitectura de la solución elegida. El segundo define los algoritmos empleados y la organización del código para comenzar la implementación.

Diseño del programa

Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario así como también los análisis necesarios para saber que herramientas usar en la etapa de Codificación.

Codificación

Es la fase en donde se implementa el código fuente, haciendo uso de prototipos así como de pruebas y ensayos para corregir errores.

Dependiendo del lenguaje de programación y su versión se crea las bibliotecas y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.

Pruebas

Los elementos, ya programados, se ensamblan para componer el sistema y se comprueba que funciona correctamente y que cumple con los requisitos, antes de ser entregado al usuario final.

Verificación

Es la fase en donde el usuario final ejecuta el sistema, para ello el o los programadores ya realizaron exhaustivas pruebas para comprobar que el sistema no falle.

En la creación de desarrollo de cascada se implementa los códigos de investigación y pruebas del mismo.

Mantenimiento

Una de las etapas más críticas, ya que se destina un 75% de los recursos, es el mantenimiento del Software ya que al utilizarlo como usuario final puede ser que no cumpla con todas las expectativas.

1.3. MODELO EN ESPIRAL

"Este modelo se basa en la necesidad continua de refinar los requerimientos para in determinado proyecto. Este modelo es eficaz cuando se utiliza para el rápido desarrollo de proyectos pequeños. Este logra el acercamiento entre el equipo de desarrollo y el cliente ya que el cliente es implicado en todas las etapas proporcionando la regeneración del proyecto y la aprobación del mismo."[3]

Este modelo no incorpora puntos de comprobación claros, de manera que el proceso de desarrollo puede ser caótico.

1.3.1. Ciclo o iteraciones

A propósito de ciclo e iteraciones Arboleya Hernan, dice:

En cada vuelta o iteración hay que tener en cuenta:

- **Los Objetivos:** qué necesidad debe cubrir el producto.

[3] VIRRUETA, Alejandra. Metodologías de desarrollo de software. Instituto Tecnológico Superior de Apatzingán (ITSA), Michoacán (México) 2010. Investigación Documental (Ingeniería en Informática). Pág. 2

- **Alternativas:** las diferentes formas de conseguir los objetivos de forma exitosa, desde diferentes puntos de vista como pueden ser:

1. Características.
2. Formas de gestión del sistema.
3. Riesgo asumido con cada alternativa.

- **Desarrollar y verificar:** Programar y probar el software.

Si el resultado no es el adecuado o se necesita implementar mejoras o funcionalidades:

- Se planificaran los siguientes pasos y se comienza un nuevo ciclo de la espiral.

La espiral tiene una forma de caracola y se dice que mantiene dos dimensiones, la radial y la angular:

1. **Angular:** Indica el avance del proyecto del software dentro de un ciclo.

2. **Radial:** Indica el aumento del coste del proyecto, ya que con cada nueva iteración se pasa más tiempo desarrollando.

Este sistema es muy utilizado en proyectos grandes y complejos como puede ser, por ejemplo, la creación de un sistema operativo[4].

1.4. PROCESO UNIFICADO RATIONAL (RUP)

Referente a proceso unificado rational, Virrueta Alejandra, dice:

Un proceso que define claramente qué, quien, cómo y cuándo debe hacerse, este aporta herramientas como los casos de uso, que definen los requerimiento, además de permitir la ejecución iterativa del proyecto y del control de riesgos.

1.4.1. Características principales:

- Guiado por los casos de uso.
- Centrado en la arquitectura.
- Guiado por los riesgos.
- Iterativo.

[4] ARBOLEYA, Hernán. Propuesta de ciclo de vida y mapa de actividades para proyectos de explotación de información. Universidad Nacional de Lanus (UNLA), Remedios de Escalada (Argentina) 2013. Trabajo de Grado (Licenciado en Sistemas). pág. 9-11.

1.4.2. Prácticas asociadas al proceso de ingeniería de software

- Desarrollo iterativo.
- Manejo de los requerimientos.
- Uso de una arquitectura basada en componentes.
- Modelización visual.
- Verificación continúa de la calidad.
- Manejo de los cambios.

1.4.3. Fases de desarrollo del software

Son 4 fases en donde cada una tiene objetivos y puntos de control.

Inicio

Objetivos
- Definir el alcance del proyecto.
- Entender que se va a construir.

Puntos de Control
- Objetivo del proyecto.

Elaboración

Objetivos
- Construir una versión ejecutable de la arquitectura de la aplicación.
- Entender cómo se va a construir.

Puntos de Control
- Arquitectura de la aplicación.

Construcción

Objetivos
- Completar el esqueleto de la aplicación con la funcionalidad.
- Construir una versión Beta.

Puntos de Control
- Versión Operativa Inicial de la Aplicación.

Transición

Objetivos

- Hacer disponible la aplicación para los usuarios finales.
- Construir la versión final.

Punto de Control

- Liberación de la versión de la Aplicación[5].

1.5. PROGRAMACIÓN EXTREMA (XP)

Referente a programación extrema (XP), Zambrano Solange, dice:

La programación extrema es una metodología de desarrollo ligera (o ágil) basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas.

Este modelo de programación se basa en una serie de metodologías de desarrollo de software en donde la prioridad es hacia los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación.

1.5.1. Principios básicos

- **Retroalimentación a escala fina.**

1. El principio de pruebas: se tiene que establecer un período de pruebas de aceptación del programa llamado también *período de caja negra* donde se definirán las entradas al sistema y los resultados esperados de estas entradas. Es muy recomendable automatizar estas pruebas para poder hacer varias simulaciones del sistema en funcionamiento.

2. Proceso de planificación: en esta fase, el usuario tendrá que escribir sus necesidades, definiendo las actividades que realizará el sistema. Se creará un documento llamado *Historias del usuario.* Entre 20 y 80 historias dependiendo de la complejidad del problema, se consideran suficientes para formar el llamado *Plan de Liberación*, el cual define de forma específica los tiempos de entrega de la aplicación para recibir retroalimentación por parte del usuario. Por regla general, cada una de las Historias del usuario suelen necesitar de una a tres semanas de desarrollo.

[5] VIRRUETA, Alejandra. Metodologías de Desarrollo de Software. Instituto Tecnológico Superior de Apatzingán (ITSA), Michoacán (México) 2010. Investigación Documental (Ingeniería en Informática). pág. 5-8

3. El cliente en el sitio: se le dará poder para determinar los requerimientos, definir la funcionalidad, señalar las prioridades y responder las preguntas de los programadores. Esta fuerte interacción cara a cara con el programador disminuye el tiempo de comunicación y la cantidad de documentación, junto con los altos costes de su creación y mantenimiento. Este representante del cliente estará con el equipo de trabajo durante toda la realización del proyecto.

4. Programación en parejas: uno de los principios más radicales y en el que la mayoría de gerentes de desarrollo pone sus dudas. Requiere que todos los programadores XP escriban su código en parejas, compartiendo una sola máquina. De acuerdo con los experimentos, este principio puede producir aplicaciones más buenas, de manera consistente, a iguales o menores costes.

- **Proceso continuo en lugar de por lotes.**

1. Integración continua: permite al equipo hacer un rápido progreso implementando las nuevas características del software. En lugar de crear versiones estables de acuerdo a un cronograma establecido, los equipos de programadores XP pueden reunir su código y reconstruir el sistema varias veces al día. Esto reduce los problemas de integración comunes en proyectos largos y estilo cascada.

2. Refactorización: permite a los equipos de programadores XP mejorar el diseño del sistema a través de todo el proceso de desarrollo. Los programadores evalúan continuamente el diseño y recodifican lo necesario. La finalidad es mantener un sistema enfocado a proveer el valor de negocio mediante la minimización del código duplicado y/o ineficiente.

3. Entregas pequeñas: colocan un sistema sencillo en producción rápidamente que se actualiza de forma rápida y constante permitiendo que el verdadero valor de negocio del producto sea evaluado en un ambiente real. Estas entregas no pueden pasar las 2 o 3 semanas como máximo.

- **Entendimiento compartido.**

1. Diseño simple: se basa en la filosofía de que el mayor valor de negocio es entregado por el programa más sencillo que cumpla los requerimientos. *Simple Design* se enfoca en proporcionar un sistema que cubra las necesidades inmediatas del cliente, ni más ni menos. Este proceso permite eliminar redundancias y rejuvenecer los diseños obsoletos de forma sencilla.

2. Propiedad colectiva del código: un código con propiedad compartida. Nadie es el propietario de nada, todos son el propietario de todo. Este método difiere en mucho a los métodos tradicionales en los que un simple programador

posee un conjunto de código. Los defensores de XP argumentan que mientras haya más gente trabajando en una pieza, menos errores aparecerán.

3. Estándar de codificación: define la propiedad del código compartido así como las reglas para escribir y documentar el código y la comunicación entre diferentes piezas de código desarrolladas por diferentes equipos. Los programadores las han de seguir de tal manera que el código en el sistema se vea como si hubiera estado escrito por una sola persona.

- **Bienestar del programador.**

1. La semana de 40 horas: la programación extrema sostiene que los programadores cansados escriben código de menor calidad. Minimizar las horas extras y mantener los programadores frescos, generará código de mayor calidad. Como dice *Kent Beck*[6], está bien trabajar tiempos extra cuando es necesario, pero no se ha de hacer durante dos semanas seguidas[7].

1.6. DESARROLLO MANEJADO POR RASGOS

Referente a desarrollo manejado por rasgos, Virrueta Alejandra, dice:

Esta metodología se enfoca en iteraciones cortas que entregan funcionalidades tangibles. Para el caso de esta metodología, dichas iteraciones duran dos semanas.

1.6.1. Procesos

- ✓ Consta de 5 procesos. Los primeros tres se hacen al principio del proyecto. y los últimos dos se hacen en cada iteración.
- ✓ Cada proceso de divide en tareas y se da un criterio de comparación:

- Desarrollar un modelo global.
- Construir una lista de los rasgos.
- Planear por rasgos.
- Diseñar por rasgos.
- Construir por rasgos.

[6] BECK, Kent. Ingeniero de software, Estadounidense. Uno de los creadores de la metodología de desarrollo extrema (XP).
[7] Zambrano Solange. Programación Extrema “XP”. Universidad de Los Andes. Mérida (Venezuela) 2010. Informe (Maestría en Educación, Mención Informática y Diseño Instruccional).

1.6.2. Tipos de desarrolladores

Programadores jefe y dueños de clases.

- ✓ Programadores jefe, son los desarrolladores más experimentados. A ellos se les asignan rasgos a construir. Sin embargo, ellos no los construyen solos. Solo identifican qué clases se involucran en la implantación de un rasgo y juntan a los dueños de dichas clases para que formen un equipo para desarrollar ese rasgo.

El programador jefe actúa como el coordinador, diseñador líder y mentor mientras los dueños de clases hacen gran parte de la codificación del rasgo.

- ✓ Dueños de clase, donde cada uno construye su diagrama para el área de dominio especificada, bajo ciertas consideraciones de arquitectura, haciendo énfasis en las clases y asociaciones, luego en los métodos y finalmente en los atributos. Por lo tanto, se realiza uno o más diagramas de secuencias informales[8].

1.7. INCREMENTAL

Referente al Modelo Incremental, Virrueta Alejandra, dice:

El Modelo Incremental combina elementos del MLS (*Multiple Listing Service*), una herramienta comunicacional basada en el acceso de información entre colegas, con la filosofía interactiva de construcción de prototipos.

Aquí el mismo cliente es quien incluye o desecha elementos al final de cada incremento a fin de que el software se adapte mejor a sus necesidades reales; el proceso se repite hasta que se elabore el producto completo.

El Modelo Incremental es particularmente útil cuando no se cuenta con una dotación de personal suficiente. Los primeros pasos los pueden realizar un grupo reducido de personas y en cada incremento poder añadir personal, de ser necesario. Por otro lado, los incrementos se pueden planear para gestionar riesgos técnicos.

[8] VIRRUETA, Alejandra. Metodologías de desarrollo de software. Instituto Tecnológico Superior de Apatzingán (ITSA), Michoacán (México) 2010. Investigación Documental (Ingeniería en Informática). pág. 11.

1.7.1. Características

- ✓ Se evitan proyectos largos y se entrega “algo de valor” a los usuarios con cierta frecuencia.
- ✓ El usuario se involucra más.
- ✓ Difícil de evaluar el costo total.
- ✓ Difícil de aplicar a los sistemas transaccionales que tienden a ser integrados y a operar como un todo.
- ✓ Requiere gestores experimentados.
- ✓ Los errores en los requisitos se detectan tarde.
- ✓ El resultado puede ser muy positivo.

1.7.2. Ventajas

- ✓ Con un paradigma incremental se reduce el tiempo de desarrollo inicial, ya que se implementa la funcionalidad parcial.
- ✓ También provee un impacto ventajoso frente al cliente, que es la entrega temprana de partes operativas del software.
- ✓ El modelo proporciona todas las ventajas del modelo en cascada realimentado, reduciendo sus desventajas sólo al ámbito de cada incremento.
- ✓ Permite entregar al cliente un producto más rápido en comparación del modelo de cascada.
- ✓ Resulta más sencillo acomodar cambios al acotar el tamaño de los incrementos.
- ✓ Por su versatilidad requiere de una planeación cuidadosa tanto a nivel administrativo como técnico.

1.7.3. Desventajas

- ✓ El Modelo Incremental no es recomendable para casos de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido, y/o de alto índice de riesgos.
- ✓ Requiere de mucha planeación, tanto administrativa como técnica.

- ✓ Requiere de metas claras para conocer el estado del proyecto[9].

1.8. DESARROLLO RÁPIDO DE APLICACIONES (RAD)

Respecto a desarrollo rápido de aplicaciones (RAD), el Instituto Nacional de Tecnologías de la Comunicación, gobierno de España, dice:

Metodología que implica el desarrollo interactivo y la construcción de prototipos basada en un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información, con la ayuda de herramientas CASE.

1.8.1. Etapas

Consiste en diferentes períodos que permiten desarrollar de manera ordenada el aplicar esta metodología.

- ✓ **Planificación:** en donde se requiere que un usuario de la compañía tenga un conocimiento integral de los procesos, con el fin de determinar cuales serán las funciones del sistema.

- ✓ **Diseño:** consistente en un análisis detallado de las actividades de la compañía en relación a la solución propuesta. Desarrollando actividades con los usuarios para que se conozcan funciones y la definición de entidades con el sistema.

- ✓ **Construcción:** desarrolladores y usuarios trabajan en equipo en la finalización del diseño y la construcción de la aplicación.

- ✓ **Implementación:** implementar el nuevo producto y manejo de la transición entre el viejo y nuevo sistema.

1.8.2. Particularidades

- ✓ Equipos compuestos de seis personas de diferentes áreas del conocimiento, incluyendo desarrolladores, personas involucradas con los requisitos incluso usuarios de tiempo completo de sistema.

- ✓ Sus herramientas en el momento de la aplicación son especializadas, la utilización de instrumentos, como: desarrollo visual, creación de prototipos funcionales, calendario grupal, interfaz estándar (API), entre otras.

[9] MEZA, Daniel. Fundamentos de desarrollo de sistemas. Instituto Tecnológico de Comitán. Chiapas (México) 2010. Documentación (Ingeniería en Sistemas Computacionales). pág. 21-24.

- ✓ El desarrollo de prototipos iterativos y evolucionarios con la aplicación de: reuniones JAD (Joint Application Development), iteraciones de prototipo, depuración basada en los requisitos actuales.

1.8.3. Principios básicos

- Objetivo clave para un rápido desarrollo y entrega de alta calidad en un sistema de relativamente con un bajo costo de inversión.

- Intenta reducir el riesgo inherente del proyecto partiéndolo en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo.

- Orientación dedicada a producir sistemas de alta calidad con rapidez, principalmente mediante el uso de iteración por prototipos en cualquier etapa de desarrollo, promueve la participación de los usuarios y el uso de herramientas de desarrollo computarizadas.

- Puede incluir constructores de interfaz gráfica de usuario (GUI).

- Ingeniería de requisitos (CASE).

- Sistemas Gestión Base Datos (DBMS).

- Control de proyecto implica el desarrollo de prioridades y la definición de los plazos de entrega[10].

1.9. SCRUM

Como lo dice Palacio[11], metodología ágil que requiere de un trabajo arduo ya que no se basa en el seguimiento de un plan si no en la adaptación continua a las circunstancias del proyecto.

1.9.1. Características

- Es un modo de desarrollo de carácter adaptable más que predictivo.
- Orientado a las personas más que a los procesos.
- Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones.

[10] Instituto Nacional de Tecnologías de la Comunicación. Gobierno de España. Ingeniería del Software: Metodologías y Ciclos de Vida. Madrid (España) 2009. Documentación (Laboratorio Nacional de Calidad del Software). pág. 46-49

[11] PALACIO, Juan. RUATA, Claudia. SCRUM MANAGER Gestión de proyectos. Safe Creative Enero 2011.

1.9.2. Formas de control

- **Revisión de las iteraciones:** Al finalizar cada iteración normalmente 30 días se lleva a cabo una revisión con todas las personas implicadas en el proyecto.

- **Desarrollo Incremental:** Durante el proyecto, las personas implicadas no trabajan con diseños o abstracciones.

Implica que al final de cada iteración se dispone de una parte del producto operativa que se puede inspeccionar y evaluar.

- **Desarrollo evolutivo:** este tipo de modelos se emplean para trabajar en entornos de incertidumbre e inestabilidad de requisitos.

Intentar predecir en las fases iniciales cómo será el producto final como también el desarrollo del diseño y arquitectura del producto no es realista ya que las circunstancias obligaran a remodelar varias veces.

- **Auto-Organización:** Durante el desarrollo de un proyecto son muchos los factores imprescindibles que surgen en todas las áreas y niveles.

- **Colaboración:** Es de gran necesidad para que funcione la auto-organización como un control eficaz así que cada miembro del equipo debe colaborar de forma abierta con los demás.

1.9.3. Visión general del proceso

SCRUM denomina “sprint” a cada iteración de desarrollo y recomienda realizarlas con duraciones de 30 días. Por lo tanto, el “SPRINT” es el núcleo central que proporciona la base de desarrollo iterativo e incremental.

1.9.4. Elementos de desarrollo

- **Las reuniones**

  - **Planificación de Sprint:** jornada de trabajo previa al inicio de cada sprint en el que se determina cual va hacer el trabajo y los objetivos que se deben cumplir.

  - **Reunión diaria:** Breve revisión del equipo de trabajo realizado a la fecha.

- **Los elementos**

  - **Pila del producto:** lista de requisitos de usuario que se origina con la visión inicial del producto y va creciendo y evolucionando durante el desarrollo.

  - **Pila del Sprint:** Lista de los trabajos que debe realizar el equipo durante el sprint para generar el incremento previsto.

  - **Incremento:** Resultado de cada sprint.

- **Los roles:** SCRUM clasifica a todas las personas que intervienen o tienen interés en el desarrollo del proyecto en: propietario del producto, equipo, gestor de SCRUM.

2. PLANEACIÓN

2.1. ALCANCE Y OBJETIVOS DEL ESTUDIO

2.1.1. Alcance

La investigación se restringe al estudio comparativo de algunas de las más importantes metodologías de desarrollo de software del momento, que fueron definidos dentro del proceso de investigación, utilizando criterios de comparación que se definirán igualmente en dicho proceso y ya como producto final se entrega un reporte sobre el resultado.

2.1.2. Objetivo

- Definir los criterios de análisis empleados para el desarrollo del estudio comparativo.
- Aplicar los criterios de análisis a cada una de las propuestas de las metodologías de desarrollo de software.
- Crear el informe final del estudio comparativo de metodologías de desarrollo de software, a partir de los resultados de los criterios aplicados.

2.2. SELECCIÓN DE METODOLOGÍAS DE DESARROLLO DE SOFTWARE QUE HICIERON PARTE DEL ESTUDIO.

Para seleccionar qué metodologías de desarrollo de software hicieron parte del estudio comparativo, se establecieron unos criterios de selección, donde a cada metodología se le dio una calificación cuantitativa según su comportamiento.

La calificación fue en una escala de 0 a 10. Donde 0 representa la calificación más baja y 10 la calificación más alta, respecto a la documentación, experiencias prácticas y herramienta CASE.

2.2.1. Criterios de selección

Documentación disponible

Proceso de reunión, recolección de documentos sobre cada una de las metodologías y el tratamiento que permitan darles, para su posterior estudio y análisis.

Experiencias prácticas

La existencia de trabajos de grado, tesis y experiencias empresariales con el hecho de evidenciar si fue exitosa, en que caso fracasó ó características más relevantes.

Herramientas CASE de soporte

Aplicaciones software que permitan aumentar la productividad en el desarrollo de software, reduciendo el costo de las mismas en términos de tiempo y dinero.

2.2.2. Aplicación criterios de selección

Documentación disponible[12]

Para evidenciar el análisis detallado de cómo se realizó la calificación, dirigirse al anexo A: Documentación disponible, análisis y obtención del "valor" como resultado final.

Calificación al momento de aplicar el criterio, documentación disponible:

METODOLOGÍA	VALOR	OBSERVACIÓN
Modelo en Cascada	4	La cantidad de información encontrada no permitiría llevar a cabo el estudio comparativo de metodologías de desarrollo de software.
Modelo en Espiral	6	La información permite desarrollar una parte del estudio comparativo de metodologías de desarrollo de software, pero no finalizarlo adecuadamente.
Proceso Unificado Rational (RUP)	8	Información concreta y precisa para el completo desarrollo del estudio comparativo de metodologías de desarrollo de software.
Programación Extrema (XP)	8	Información completa, clara para el normal desarrollo del estudio comparativo de metodologías de desarrollo de software.
Desarrollo Manejado por Rasgos	4	Existe una cantidad mínima de información que no haría posible culminar el estudio comparativo de metodologías de desarrollo de software.
Incremental	6	Información puntual, que permite desarrollar la mayor parte del estudio comparativo de metodologías de desarrollo de software.
Desarrollo Rápido de Aplicaciones (RAD)	7	Cantidad de información suficiente que permite llegar a la meta en este estudio comparativo de metodologías de desarrollo de software.

[12] Anexo A: Documentación disponible, análisis y obtención del "valor" como resultado final.

SCRUM	8	Definición concisa en la mayoría de características, conceptos, que permiten una evolución frente al desarrollo del estudio comparativo de metodologías de desarrollo de software.

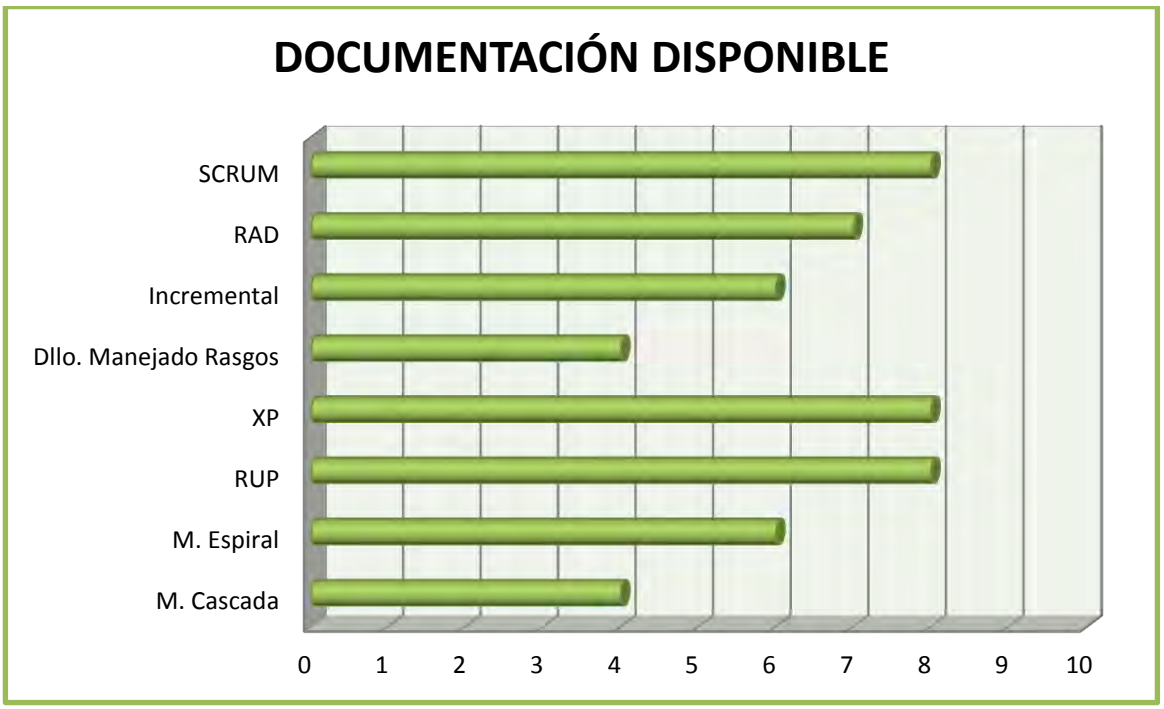

Ilustración 1. Documentación disponible

Experiencias prácticas[13]

Calificación al momento de aplicar el criterio, experiencias prácticas:

METODOLOGÍA	VALOR	OBSERVACIÓN
Modelo en Cascada	4	Cantidad mínima de trabajos de grado, por ser una de las primeras metodologías y no encontrar información suficiente para investigar.
Modelo en Espiral	3	No es posible identificar un estudio detallado aplicando esta metodología y así obtener resultados claros.
Proceso Unificado Rational (RUP)	7	Experiencias con resultados favorables, donde se observa la correcta aplicación de la metodología y sus resultados.
Programación Extrema (XP)	6	Permite identificar la aplicación de la metodología hasta el punto de observar resultados conformes.
Desarrollo Manejado por Rasgos	3	Trabajos e investigaciones que no demuestran la aplicación clara de dicha

[13] Anexo B: Experiencias prácticas, análisis y obtención del "valor" como resultado final.

		metodología, obteniendo resultados imprecisos.
Incremental	4	Igualmente existen trabajos donde se ve aplicada la metodología en un porcentaje mínimo y no es su totalidad.
Desarrollo Rápido de Aplicaciones (RAD)	6	La aplicación de la metodología es correcta y se observa resultados finales dentro de contexto.
SCRUM	7	Los resultados de las experiencias, han hecho que den mejoras a los procesos aplicados, obteniendo mejores resultados.

Ilustración 2. Experiencias prácticas

Herramientas CASE de soporte[14]

Calificación al momento de aplicar el criterio, herramientas CASE de soporte:

METODOLOGÍA	VALOR	OBSERVACIÓN
Modelo en Cascada	2	La cantidad de aplicaciones que existe frente a esta metodología son mínimas y no se observa su implementación.
Modelo en Espiral	2	Al igual que el Modelo en Cascada no existe una aplicación completa que permita observar el desarrollo de la metodología.

[14] Anexo C: Herramientas CASE de soporte, análisis y obtención del "valor" como resultado final.

Proceso Unificado Rational (RUP)	4	Aplicaciones que permiten un desarrollo normal al momento de aplicar esta metodología, con entrega de resultados positivos.
Programación Extrema (XP)	3	Herramientas que permiten evidenciar y servir de guía en la aplicación de esta metodología.
Desarrollo Manejado por Rasgos	1	Imposible identificar una herramienta que ayude evidenciar, la aplicación de esta metodología.
Incremental	2	No existe posibilidad de brindar ayuda con algún tipo de herramienta en el desarrollo de esta metodología.
Desarrollo Rápido de Aplicaciones (RAD)	3	Al igual que en la Metodología XP, permite guiarme de manera adecuada y realizar una correcta aplicación de esta metodología.
SCRUM	4	Herramientas que trazan el camino claro cómo se debe hacer para la correcta aplicación de esta metodología y así obtener resultados claros.

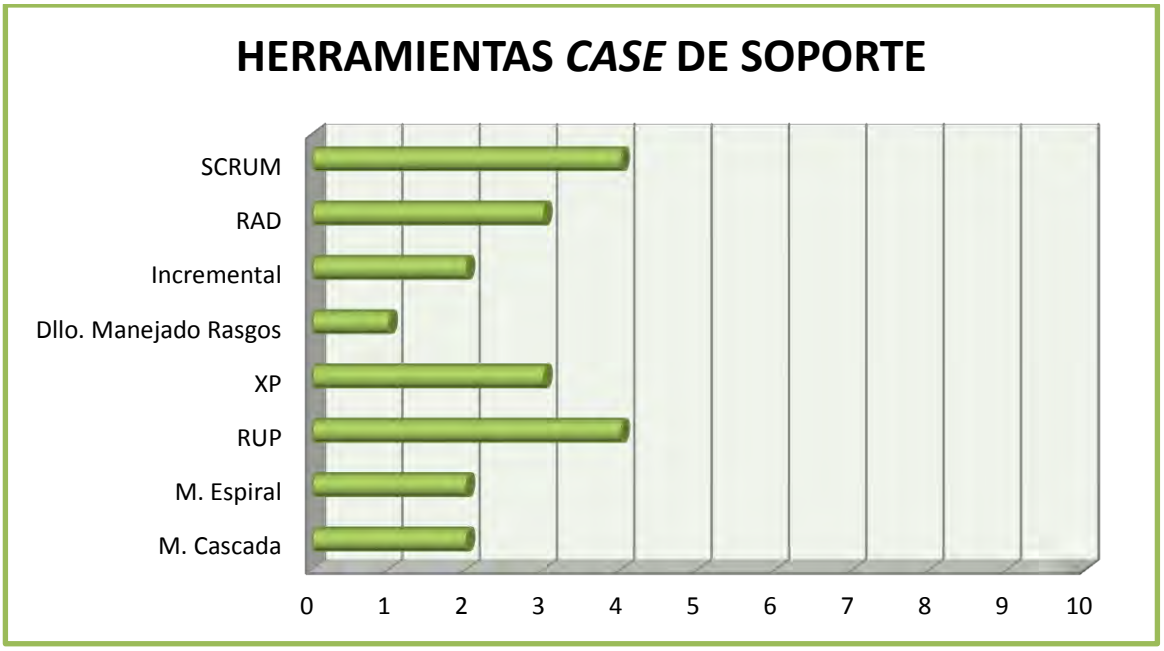

Ilustración 3. Herramientas CASE de soporte

Resultado general

METODOLOGÍA	Criterio 1	Criterio 2	Criterio 3	TOTAL
Proceso Unificado Rational (RUP)	8	7	4	19
SCRUM	8	7	4	19
Programación Extrema (XP)	8	6	3	17
Desarrollo Rápido de Aplicaciones (RAD)	7	6	3	16
Incremental	6	4	2	12
Modelo en Cascada	4	4	2	10
Modelo en Espiral	6	3	2	11
Desarrollo Manejado por Rasgos	4	3	1	8

Criterio 1: Documentación disponible
Criterio 2: Experiencias prácticas
Criterio 3: Herramientas CASE de soporte

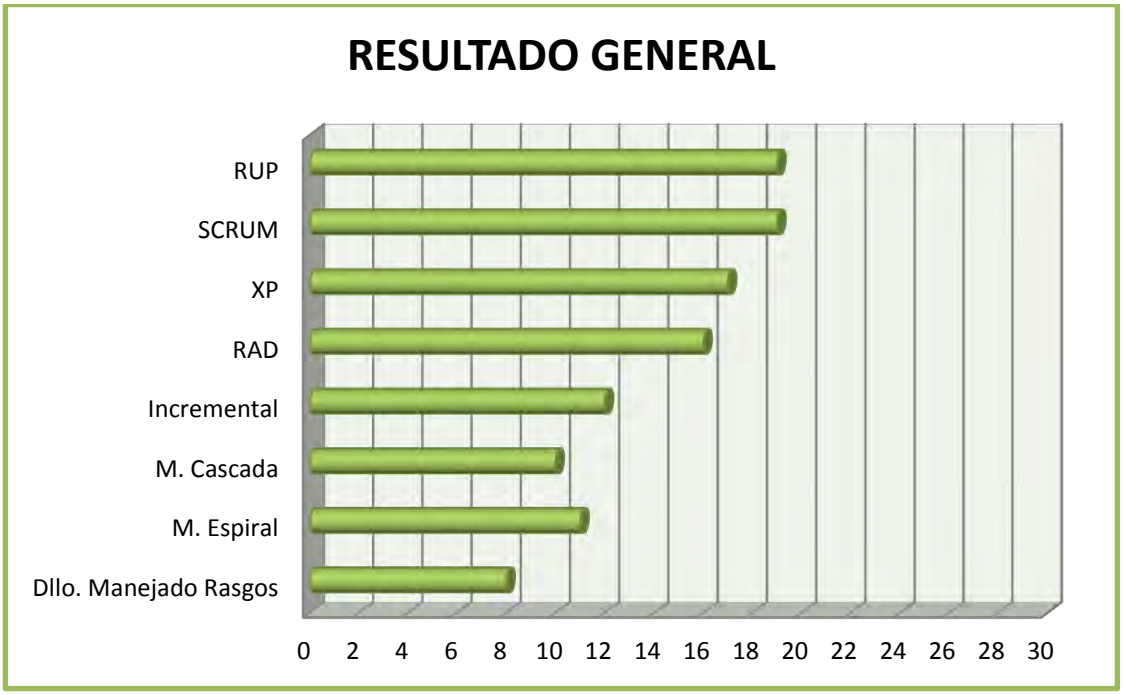

Ilustración 4. Resultado general

Una vez terminada la aplicación de los criterios de selección a cada una de las metodologías y obtenidos dichos resultados, se evidencia que para la realización del estudio comparativo de metodologías de desarrollo de software y en dicho proceso no tener inconvenientes en la investigación por falta de información, es conveniente tomar las siguientes metodologías de desarrollo de software:

- **Proceso Unificado Rational (RUP).**
- **SCRUM.**
- **Programación Extrema (XP).**
- **Desarrollo Rápido de Aplicaciones (RAD).**

2.3. DESCRIPCIÓN DE LAS METODOLOGÍAS TOMADAS COMO CASO DE ESTUDIO.

2.3.1. Proceso Unificado Rational (RUP)

Representación gráfica

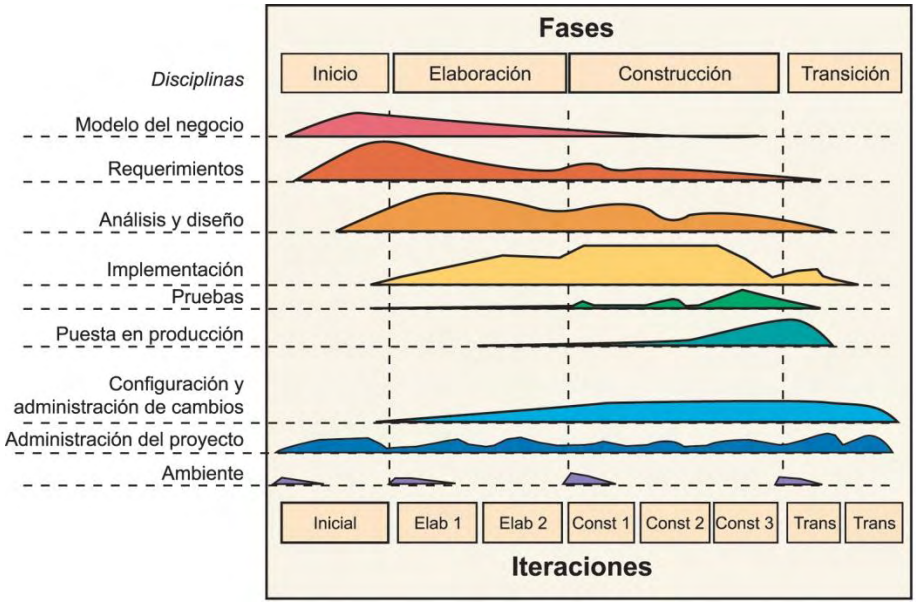

Ilustración 5. Esquema RUP
Fuente: http://ingsoftwarefinal1.files.wordpress.com/2010/11/modelo3.jpg

Descripción

A propósito de Proceso Unificado Rational, Villalba, dice:

Rational Unified Process (RUP) es una metodología de desarrollo de software orientado a objetos que establece las bases, plantillas y ejemplos para todos los aspectos y fases de desarrollo del software. RUP es una herramienta de la ingeniería de software que combinan los aspectos del proceso de desarrollo (fases definidas, técnicas y prácticas) con otros componentes de desarrollo (documentos, modelos, manuales, código fuente) dentro de un framework unificado. RUP establece cuatro fases de desarrollo cada una de las cuales está organizada en varias iteraciones separadas que deben satisfacer criterios definidos antes de emprender la próxima fase.

Tiene tres características esenciales: está dirigido por los *Casos de Uso,* está centrado en la *arquitectura* y es *Iterativo e incremental.*

Los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que

sería bueno contemplar. Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema.

RUP presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

Existe una interacción entre los casos de uso y la arquitectura, los casos de uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos, actualmente y en el futuro. Esto provoca que tanto arquitectura como casos de uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software.

RUP propone tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos, permitiendo generar un equilibrio entra casos de uso y arquitectura. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

Una iteración puede realizarse por medio de una cascada como se muestra en la siguiente figura (6). La cual pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

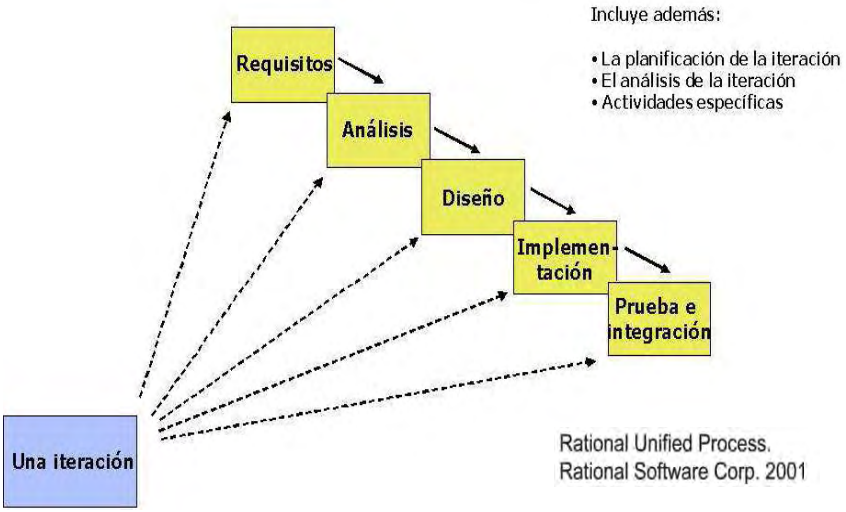

Ilustración 6. Iteraciones en RUP
Fuente: http://cocolito.comoj.com/rup.html

RUP identifica las seis mejores prácticas con las que define una forma efectiva de *trabajar para los* equipos de desarrollo de software.

- ✓ La administración de requerimientos.
- ✓ El desarrollo iterativo.
- ✓ La arquitectura basada en componentes.
- ✓ El modelo visual
- ✓ La verificación continua de la calidad
- ✓ La administración del cambio

Estas seis prácticas orientan el modelo y con ellas se pretenden solucionar muchos de los problemas asociados al software. Adicionalmente, hay muchos aspectos de diseño que son bien conocidos, pero que en realidad han sido muy poco implementados en los proyectos de software; estos son: facilidad de uso, modularidad, encapsulamiento y facilidad de mantenimiento. Es necesario entonces definir una arquitectura sólida basada en componentes, para construir mejores y más flexibles soluciones de software para las necesidades organizacionales.

RUP brinda una guía para encontrar, organizar, documentar y seguir los cambios de los requisitos funcionales y restricciones. Utiliza una notación de Caso de Uso para representar los requisitos. Con la finalidad de especificar el comportamiento deseado del sistema (objetivos del usuario), así como describir qué debe de hacer, pero no especifica cómo lo hace y da lugar a un conjunto de posibles escenarios.

Un rol define el comportamiento y responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. Una persona puede desempeñar diversos roles, así como un mismo rol puede ser representado por varias personas.

RUP define grupos de roles, agrupados por participación en actividades relacionadas. Estos grupos, son:

- ✓ Analistas.
- ✓ Desarrolladores.
- ✓ Gestores.
- ✓ Apoyo.
- ✓ Especialistas en pruebas.
- ✓ Revisor.
- ✓ Coordinación de revisiones.
- ✓ Revisor técnico[15].

[15] VILLALBA Erika, RAMÓN Eder. Desarrollo de sistemas con metodología RUP (RATIONAL UNIFIED PROCES). Universidad Nacional Autónoma de México. México D.F. Trabajo de grado (Ingeniero en Computación). Pág. 99-106.

Ventajas

Como lo dice, Cardinales[16].

- ✓ Forma disciplinada de asignar tareas y responsabilidades dentro de una organización de desarrollo.
- ✓ Mejora la productividad del equipo.
- ✓ Estima tareas y horarios del plan midiendo la velocidad de iteraciones concernientes a sus estimaciones originales.
- ✓ Basada en las mejores prácticas que se han intentado y se han probado en el campo.
- ✓ Realiza un levantamiento exhaustivo de requerimientos.

Desventajas

- ✓ Requiere un grupo grande de programadores para trabajar con esta metodología.
- ✓ Es más apropiada para proyectos grandes.
- ✓ Los miembros del equipo deben ser expertos en su campo para desarrollar un software.
- ✓ El proceso de desarrollo es demasiado complejo.
- ✓ La evaluación de riesgos es compleja.

2.3.2. Programación Extrema (XP)

Representación gráfica

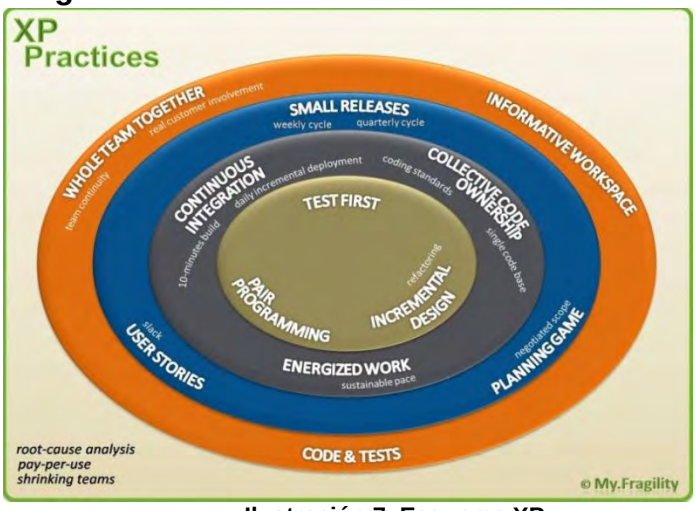

Ilustración 7. Esquema XP
Fuente: http://ullizee.files.wordpress.com/2009/11/grafx-xp-practices.jpg

[16] CARDINALES, Cristina. Cuadro comparativo de las metodologías. República Bolivariana de Venezuela 2012. Documentación. Ministerio del Poder para la Educación Universitaria, Fundación Misión Sucre. Pág. 3.

Descripción

Como bien lo dice, Zambrano[17], denominada extrema porque lleva a límites extremos algunos elementos y actividades comunes de la forma tradicional de programar.

XP es una metodología ágil para pequeños a medianos equipos, que modela el proceso de desarrollo de software cuando los requerimientos son ambiguos o rápidamente cambiantes.

Este modelo de programación, que plantea una forma liviana y adaptable del proceso, surgió en respuesta a una serie de problemas y necesidades que se evidencian al llevar a cabo un proyecto de desarrollo de software son la finalidad de minimizar las dificultades y errores que se presentan específicamente al abordar cierto tipo de proyectos.

La utilidad de esta metodología de trabajo fue creada para que se utilice:

- ✓ Cuando los clientes no tienen idea clara de los requerimientos y los van cambiando.
- ✓ Para proyectos de riesgo: fecha fija de entrega, algo nunca hecho por el grupo, algo nunca hecho por la comunidad de desarrolladores.
- ✓ Entre 2 y 10 programadores. NO es apto para proyectos con mucho personal.
- ✓ Proyectos en que los requisitos tienen altas probabilidades de cambiar con el tiempo (por ejemplo, porque el cliente no tiene claro lo que quiere, o porque el cambio de requisitos está ligado al dominio del problema a resolver).
- ✓ El objetivo es entregar el software tal cual se necesita y en el momento en que se necesita. Incidentalmente, los proyectos XP muestran mayor productividad.

Características

- ✓ **Enfoque práctico:** ya que se adapta y obedece a las realidades que ocurren al desarrollar un software como la posibilidad de los cambios y la ocurrencia de errores.
- ✓ **Conocer sus ingredientes desde el principio:** ya que parte de la definición clara de las necesidades y requerimientos del cliente manteniéndolas presentes durante todo el desarrollo del proyecto.

[17] ZAMBRANO Solange, LEÓN Carlos, GÓMEZ Laura. Programación Extrema “XP”. Universidad de Los Andes. Mérida (Venezuela) 2010. Informe (Maestría en Educación, Mención Informática y Diseño Instruccional).

- ✓ **Retroalimentación (*Feedback*) temprana, continua y correcta:** gracias a los ciclos extremadamente cortos de desarrollo.

- ✓ **Enfoque de planificación incremental,** que consigue obtener un plan global rápido, del que se espera que evolucionará a lo largo de la vida del programa.

- ✓ **Su funcionalidad de forma extremadamente flexible,** en respuesta a los cambios en las necesidades del negocio.

- ✓ **Uso de test automatizados,** escritos por programadores y cliente para monitorizar el progreso del desarrollo, para permitir que evolucione el sistema y para atrapar errores de forma temprana.

- ✓ **Comunicación oral, test y el código fuente,** para comunicar la estructura del sistema y las intenciones del código.

- ✓ **Metodología de colaboradores entre programadores,** con habilidades normales.

- ✓ **Adopción de prácticas realistas,** que trabajan a favor de los instintos a corto plazo de los programadores y los intereses a largo plazo del proyecto.

- ✓ **Integración de gerentes y clientes** a la formulación de preguntas, la negociación de cronograma y alcances, la creación de las pruebas.

- ✓ **Automatiza las pruebas,** es posible en casi todos los dominios.

Objetivos

- ✓ **La satisfacción del cliente.** Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, debemos responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación.

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

Todo proyecto de desarrollo de software bajo esta metodología, debe considerar cuatro variables esenciales, en donde sólo tres de ellas podrán ser fijadas o indicadas por el cliente o jefe del proyecto, mientras que una variable quedará libre.

Variables esenciales:

- ✓ **Coste,** del proyecto se incrementa cuando se necesita máquinas más rápidas, más especialistas técnicos en determinadas áreas o mejores oficinas para el equipo de desarrollo.

  En Extreme Programming el costo del cambio maneja un papel muy importante, porque comparado con otras metodologías para implementar software, es mucho más barato, debido a que las pruebas se van haciendo según las versiones liberadas, no es como una metodología normal, que primero se realiza análisis, después el diseño, implementación, pruebas y finalmente producción, mientras que en la Extreme Programming siempre se está implementando, probando y produciendo.

- ✓ **Tiempo,** en el que se planifica y en que realmente se lleva a cabo el proyecto. Debe tomarse en cuenta que los cambios aumentarán el tiempo de realización mientras que la optimización y la inversión pueden acortarlo.

- ✓ **Calidad,** puede representar un cambio extraño, debido a que a mayor calidad menor tiempo de realización del proyecto. Por lo tanto, el equipo de desarrolladores está encargado de la tarea de hacer las pruebas con los mejores resultados posibles para así tener una idea de cuál es el problema y como lo van a resolver de una manera simple y eficiente, para que la calidad del proyecto se mantenga al 100% y tener una facilidad de adaptarse a los cambios del código lo que hace proceso más rápido.

- ✓ **Ámbito,** es la que se encuentra libre es el alcance del proyecto, en la cual el equipo determina: la estimación de las tareas a realizar, que es lo que el cliente quiere, la implementación de los requisitos más importantes de manera que sea siempre funcional.

Además, la metodología que propone la programación extrema requiere una serie de valores que darán consistencia y solidez al equipo de trabajo.

- ✓ **La comunicación prevalece en todas las prácticas de Extreme Programming.** Comunicación cara a cara es la mejor forma de comunicación, entre los desarrolladores y el cliente. Método muy ágil. Gracias a esto el equipo puede realizar cambios que al cliente no le gustaron.

- ✓ **La sencillez.** Ayuda a que los desarrolladores de software encuentren soluciones más simples a problemas, según el cliente lo estipula. Los desarrolladores también crean características en el diseño que pudieron ayudar a resolver problemas en el futuro.

- ✓ **La retroalimentación continua** del cliente permite a los desarrolladores llevar y dirigir el proyecto en una dirección correcta hacia donde el cliente quiera.

- ✓ **El valor o coraje** requiere que los desarrolladores vayan a la par con el cambio, porque sabemos que este cambio es inevitable, pero el estar preparado con una metodología ayuda a ese cambio.

Ventajas

Como lo dice, Abril[18].

- ✓ Planificación más transparente para los clientes, donde se conocen las fechas de entrega de las funcionalidades algo vital para el negocio.

- ✓ Menor taza de errores.

- ✓ Permite tener realimentación de los usuarios muy útil.

- ✓ Satisfacción del Programador

- ✓ La presión está a lo largo del proyecto y no en la entrega de un producto final.

- ✓ Estar preparados para el cambio, significa reducir su coste.

Desventajas

- ✓ Es factible en proyectos a corto plazo

- ✓ El desarrollo de software es riesgoso y difícil de controlar

- ✓ Se rediseñara todo el tiempo (refactoring), dejando el código siempre en el estado más simple e incrementándose el tiempo de desarrollo.

- ✓ Delimitar el alcance del proyecto con el cliente.

- ✓ La falta de documentación que no se hace, debido a que todos saben del proyecto, no permite recoger la experiencia para próximos proyectos.

- ✓ No se requiere un nivel alto del tipo de programadores.

[18] ABRIL, David. Parte I Metodología de desarrollo de software. Especialización tecnológica en desarrollo de aplicaciones para dispositivos Móviles 2011. Análisis documental. pág. 3

2.3.3. SCRUM

A propósito de SCRUM, Juan Palacio, dice:

Representación gráfica

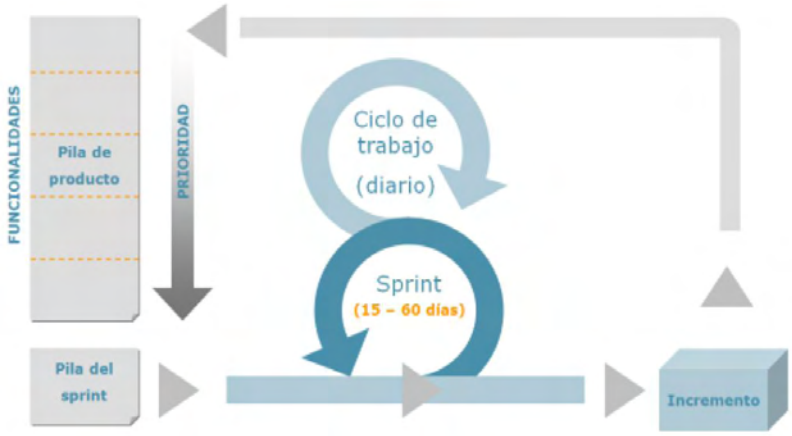

Ilustración 8. Esquema SCRUM
Fuente: http://www.scrummanager.net

Descripción

SCRUM es una metodología de desarrollo muy simple, que requiere trabajo duro, por la adaptación continua a las circunstancias de la evolución del proyecto.

Como método ágil:

- ✓ Es un modelo de desarrollo adaptable, antes que predictivo.
- ✓ Orientado a las personas, más que a los procesos.
- ✓ Emplea el modelo de construcción incremental basado en iteraciones y revisiones.

Comparte los principios estructurales del desarrollo ágil: a partir del concepto o visión de la necesidad del cliente, construye el producto de forma incremental a través de iteraciones breves que comprenden fases de especulación, exploración y revisión. Estas iteraciones (en SCRUM llamadas Sprint) se repiten de forma continua hasta que el cliente da por cerrado el producto.

Estas iteraciones son la base del desarrollo ágil y SCRUM gestiona su evolución en reuniones breves diarias donde todo el equipo revisa el trabajo realizado el día anterior y el previsto para el siguiente.

SCRUM controla de forma empírica y adaptable la evolución del proyecto, a través de las siguientes prácticas de la gestión ágil:

- ✓ **Revisión de las Iteraciones:** al finalizar cada iteración (sprint) se lleva a cabo una revisión con todas las personas implicadas en el proyecto. Es por tanto la duración del sprint, el periodo máximo que se tarda en reconducir una desviación en el proyecto o en las circunstancias del producto.

- ✓ **Desarrollo incremental:** las personas implicadas no trabajan con diseños o abstracciones. El desarrollo incremental implica que al final de cada iteración se dispone de una parte de producto operativa, que se puede inspeccionar y evaluar.

- ✓ **Desarrollo evolutivo:** los modelos de gestión ágil se emplean para trabajar en entornos de incertidumbre e inestabilidad de requisitos.
  Intentar predecir en las fases iniciales cómo será el resultado final, y sobre dicha predicción desarrollar el diseño y la arquitectura del producto no es realista, porque las circunstancias obligarán a remodelarlo muchas veces.

  ¿Para qué predecir los estados finales de la arquitectura o del diseño si van a estar cambiando? SCRUM considera a la inestabilidad como una premisa, y se adopta técnicas de trabajo para permitir la evolución sin degradar la calidad de la arquitectura que también evoluciona durante el desarrollo.

- ✓ **Auto-Organización:** en la ejecución de un proyecto son muchos los factores impredecibles en todas las áreas y niveles. La gestión predictiva confía la responsabilidad de su resolución al gestor de proyectos.

En SCRUM los equipos son auto-organizados (no auto-dirigidos), con márgenes de decisión suficiente para tomar las decisiones que consideren oportunas.

- ✓ **Colaboración:** las prácticas y el entorno de trabajo ágiles facilitan la colaboración del equipo. Ésta es necesaria, porque para que funcione la auto-organización como un control eficaz cada miembro del equipo debe colaborar de forma abierta con los demás, según sus capacidades y no según su rol o su puesto.

- ✓ **Visión general del proceso:** SCRUM denomina “sprint” a cada iteración de desarrollo y según las características del proyecto y las circunstancias del sprint puede determinarse una duración desde uno hasta dos meses, aunque no suele ser recomendable hacerlos de más de un mes.
  El sprint es el núcleo central que proporciona la base de desarrollo iterativo e incremental.

Los elementos que conforman el desarrollo SCRUM son:

- **Las reuniones**

Ilustración 9. Reuniones SCRUM
Fuente: http://www.scrummanager.net

- **Planificación del Sprint:** jornada de trabajo previa al inicio de cada sprint en la que se determina cuál va a ser el trabajo y los objetivos que se deben conseguir en la iteración.
- **Seguimiento del Sprint:** breve revisión diaria, en la que cada miembro describe tres cuestiones:

  1. El trabajo se realizó el día anterior.

  2. El que tiene previsto realizar.

  3. Cosas que puede necesitar o impedimentos que debe suprimirse para realizar el trabajo.

✓ **Revisión del Sprint:** análisis y revisión del incremento generado.

- **Los elementos**

✓ **Pila del producto:** (*product backlog*) lista de requisitos de usuario que a partir de la visión inicial del producto crece y evoluciona durante el desarrollo.

✓ **Pila del Sprint:** (sprint *backlog*) lista de los trabajos que debe realizar el equipo durante el sprint para generar el incremento previsto.

✓ **Incremento:** resultado de cada sprint

- **Los Roles:** todas las personas que intervienen, o tienen relación directa o indirecta con el proyecto, se clasifican en dos grupos: comprometidos e implicados.

- ✓ **Comprometidos:** propietario del producto, equipo.
- ✓ **Implicados:** otros interesados, Dirección General, Dirección Comercial, Marketing, Usuarios[19].

Ventajas

Como lo dice, Raul Jimenez[20].

- ✓ Fácil de aprender.
- ✓ Permite abarcar proyectos donde los requisitos de negocio están incompletos.
- ✓ Permite el desarrollo, testeo y correcciones rápidas.
- ✓ Mediante las reuniones diarias se ven claramente los avances y problemas.
- ✓ Por ser metodología ágil, obtiene muchos *feedback* del cliente.
- ✓ Facilita la entrega de productos de calidad a tiempo.

Desventajas

- ✓ Si no se define una fecha de fin, los clientes pedirán nuevas funcionalidades.
- ✓ Si una tarea no está bien definida puede incrementar costes y tiempos.
- ✓ Si el equipo no se compromete hay muchas probabilidades de fracasar.
- ✓ Solo funciona bien en equipos pequeños y ágiles.
- ✓ Se requieren miembros del equipo experimentados.
- ✓ Solo funciona cuando el *SCRUM Manager* confía en su equipo.
- ✓ Que un miembro abandone el equipo durante el desarrollo puede conllevar grandes problemas.

[19] PALACIO, Juan. RUATA, Claudia. SCRUM MANAGER Gestión de proyectos. Safe Creative Enero 2011.
[20] JIMÉNEZ, Raúl. Introducción a SCRUM. Asociación Web masters de Granada. Madrid (España).

2.3.4. Desarrollo Rápido de Aplicaciones (RAD)

A propósito de Desarrollo de Aplicaciones, Rosangélica Rodriguez, dice:

Representación gráfica

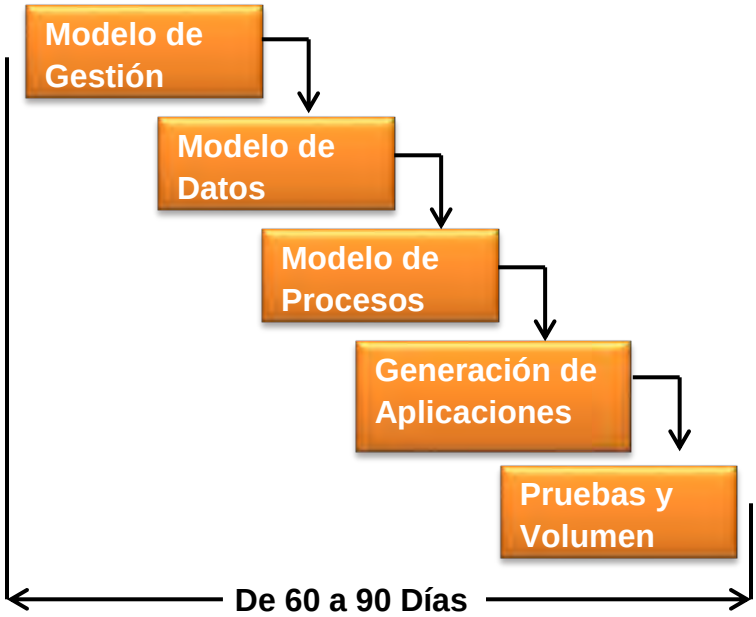

Ilustración 10 Esquema RAD
Fuente: http://www.41jaiio.org.ar

Descripción

El desarrollo rápido de aplicaciones RAD es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. El método comprende el desarrollo interactivo, la construcción de prototipos y el uso de utilidades CASE. Tradicionalmente, el desarrollo rápido de aplicaciones tiende a englobar también la usabilidad, utilidad y la rapidez de ejecución.

Ciclo de vida

- ✓ Etapa de planificación de los requisitos. Esta etapa requiere que usuarios con un vasto conocimiento de los procesos de la compañía determinen cuáles serán las funciones del sistema. Debe darse una discusión estructurada sobre los problemas de la compañía que necesitan solución.

- ✓ Etapa de diseño. Esta etapa consiste en un análisis detallado de las actividades de la compañía en relación al sistema propuesto. Los usuarios participan activamente en talleres bajo la tutela de profesionales de la informática. En ellos descomponen funciones y definen entidades asociadas con el sistema. Una vez

se completa el análisis, se crean los diagramas que definen las alteraciones entre los procesos y los datos.

- ✓ Etapa de construcción. En la etapa de construcción el equipo de desarrolladores trabaja de cerca con los usuarios finalizando el diseño y la construcción de la aplicación, consistente en una serie de pasos donde los usuarios tienen la oportunidad de afirmar los requisitos y repasar los resultados.

- ✓ Etapa de implementación. Esta etapa envuelve la implementación del nuevo producto y el manejo del cambio del viejo al nuevo sistema. Se hacen pruebas comprensivas y se guía a los usuarios.

Fases

- ✓ **Modelado de gestión:** el flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas:

  1. ¿Qué información conduce el proceso de gestión?
  2. ¿Qué información se genera?
  3. ¿Quién la genera?
  4. ¿A dónde va la información?
  5. ¿Quién lo procesó?

- ✓ **Modelado de datos:** el flujo de información definido como parte de la fase de modelado de gestión se refina como un conjunto de objetos de datos necesarios para apoyar la empresa. Se definen las características de cada uno de los objetos y las relaciones entre estos objetos.

- ✓ **Modelado de proceso:** los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir o recuperar un objeto de datos. Es la comunicación entre los objetos.

- ✓ **Generación de aplicaciones:** RAD usa técnicas de cuarta generación, en lugar de crear software con lenguajes de programación de tercera generación, el proceso RAD trabaja para volver a utilizar componentes de programas ya existentes (cuando es posible) o crear componentes de programas reutilizables (cuando sea necesario). En todos los casos se utilizan herramientas automáticas para facilitar la construcción del software.

- ✓ **Pruebas de entrega:** RAD enfatiza la reutilización de los componentes de los programas ya comprobados. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y ejercitar todas las interfaces a fondo.

Principios Básicos

- ✓ Objetivo clave para un rápido desarrollo y entrega de una alta calidad en un sistema relativamente bajo de inversión.

- ✓ Intenta reducir el riesgo inherente del proyecto partiéndolo en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo.

- ✓ Orientación dedicada a producir sistemas de alta calidad con rapidez, principalmente mediante el uso de iteraciones por prototipos en cualquier etapa de desarrollo, promueve la participación de los usuarios y el uso de herramientas de desarrollo computarizadas. Estas herramientas pueden incluir constructores de interfaz gráfica de usuario, herramientas CASE, los sistemas de gestión de bases de datos, lenguajes de programación de cuarta generación, generadores de código y técnicas orientadas a objetos.

- ✓ Hace especial hincapié en el cumplimiento de la necesidad comercial, mientras que la ingeniería tecnológica o la excelencia es de menor importancia.

- ✓ Control de proyecto implica el desarrollo de prioridades y la definición de los plazos de entrega. Si el proyecto empieza a aplazarse, se hace hincapié en la reducción de requisitos para el ajuste, no en el aumento de la fecha límite.

- ✓ La participación activa de los usuarios es imprescindible.

- ✓ Iterativamente realiza la producción de software, en lugar de enfocarse en un prototipo.

- ✓ Produce la documentación necesaria para facilitar el futuro desarrollo y mantenimiento.

Características

- ✓ **Equipos Híbridos:** equipos compuestos alrededor de seis personas, incluyendo desarrolladores y usuarios de tiempo completo del sistema así como aquellas personas involucradas con los requisitos. Los desarrolladores de RAD deben ser renacentistas, es decir, analistas, diseñadores y programadores en uno.

- ✓ **Herramientas especializadas:**

  - Desarrollo visual.
  - Creación de prototipos falsos (simulación pura).
  - Creación de prototipos funcionales.
  - Múltiples lenguajes.

- Calendario grupal.
- Herramientas colaborativas y de trabajo en equipo.
- Componentes reusables.
- Interfaces estándares (API).

✓ **Timeboxing:** las funciones secundarias son eliminadas como sea necesario para cumplir con el calendario.

✓ **Prototipos iterativos y evolucionarios:**

- Reunión JAD (Joint Application Development):

  - Se reúnen los usuarios finales y los desarrolladores.

  - Lluvia de ideas para obtener un borrador de requisitos.

- Iterar para acabar:

  - Los desarrolladores construyen y depuran el prototipo basado en los requisitos actuales.

  - Los diseñadores revisan el prototipo.

  - Los clientes pruebas el prototipo, depuran los requisitos.

  - Los clientes y desarrolladores se reúnen para revisar juntos el producto, refinar los requisitos y generar solicitudes de cambios.

- Notas:

  - Cada iteración dura entre un día y tres semanas.

  - Reunión de 2 horas con facilitador que mantiene enfocado al grupo. El facilitador tiene claras las metas sobre la información que se necesita recabar, prepara una agenda de asuntos antes de la reunión, asegura que la discusión adecuada cubra cada asunto y por ultimo escribe un reporte al final de la reunión[21].

[21] RODRIGUEZ Rosangélica, GONZÁLEZ Henyeliz, ROSAS Victor. Metodología de desarrollo rápido de aplicación. Universidad Nacional Experimental Politécnica de la Fuerza Armada Bolivariana. Núcleo Sucre-Carúpano (Venezuela) 2013. Documentación (Ingeniería de Sistemas).

Ventajas

- ✓ Los entregables pueden ser fácilmente trasladados a otra plataforma.
- ✓ El desarrollo se realiza a un nivel de abstracción mayor.
- ✓ Visibilidad temprana.
- ✓ Mayor flexibilidad.
- ✓ Menor codificación manual.
- ✓ Mayor involucración de los usuarios.
- ✓ Ciclos de desarrollo más pequeños.
- ✓ Interfaz gráfica estándar.

Desventajas

- ✓ Costo de herramientas integradas y equipo necesario.
- ✓ Progreso más difícil de medir.
- ✓ Menos eficiente.
- ✓ Riesgo de revertirse a las prácticas sin control de antaño.
- ✓ Dependencia en componentes de terceros: funcionalidad de más o de menos, problemas legales.
- ✓ Requisitos que no convergen.
- ✓ Interfaz gráfica estándar.

3. ANÁLISIS COMPARATIVO ENTRE METODOLOGÍAS DE DESARROLLO DE SOFTWARE TOMADAS COMO CASO DE ESTUDIO

3.1. CRITERIOS DE COMPARACIÓN

Para evidenciar el análisis detallado de cómo se obtuvo los criterios de comparación, dirigirse al anexo D: Selección criterios de comparación.

Las metodologías para el proceso de desarrollo de software son numerosas, dando lugar a un panorama confuso al momento de escoger una metodología en el desarrollo de software. Para la selección adecuada de la metodología, se realizó un análisis comparativo entre ellas, que en este caso son RUP, XP, SCRUM y RAD, donde se aplicó determinados criterios de comparación.

Criterios de comparación, que permitieron caracterizar bajo elementos comunes cada una de las metodologías de desarrollo de software. Sirviendo como guía para la selección de que metodología de desarrollo de software se aplicará a la hora de desarrollar un software.

3.2. DESCRIPCIÓN DE CRITERIOS DE COMPARACIÓN

3.2.1. Tipo de FrameWork

¿Qué es?

"Es un esquema (esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación, con componentes personalizables e intercambiables."[22]

¿Para qué sirve?

- ✓ Plantea una estructura global de la metodología, haciendo que el desarrollador se guía bajo el *framework* de cada metodología.

- ✓ Facilita la colaboración. Cualquiera que haya desarrollado o estudiado la metodología sabrá lo difícil que es entenderla. Por lo tanto todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrolladores colaborativos.

[22] Wikipedia-la enciclopedia libre: http://es.wikipedia.org/wiki/Framework.

¿Por qué se eligió?

Cada metodología de desarrollo de software maneja una serie de conceptos, técnicas, enfoques y fases personalizables es el caso de la metodología XP que me permite adaptación por el hecho de ser flexible.

3.2.2. Tipo de Revisión

¿Qué es?

Cada metodología propone examinar el trabajo, al igual de cómo se realiza dicho trabajo y en qué momento se ejecuta, cuyo objetivo principal es la búsqueda de errores a tiempo, con el fin de proponer y evaluar las soluciones alternativas o las mejoras en la aplicación de la metodología.

¿Para qué sirve?

- ✓ Técnicas utilizadas por RUP, XP, SCRUM y RAD, consistentes en que una vez finalizada una etapa de desarrollo, hay que aplicar pasos de revisión con el fin de corregir y mejorar la aplicación de la metodología.

- ✓ Validar la completitud y corrección de los entregables de cada fase o etapa, previniendo en forma temprana sobre potenciales problemas y riesgos que puedan derivarse en etapas posteriores de desarrollo: inconsistencias, ambigüedades, no cumplimiento de normas.

¿Por qué se eligió?

Con el estudio de las metodologías RUP, XP, SCRUM y RAD, se observó que cada una de ellas por lo general en cada etapa o al finalizar la aplicación de la metodología. Se desarrollan actividades o pasos para la evaluación hasta ese momento y posteriormente no tener errores.

3.2.3. Objetivos

¿Qué es?

Meta o finalidad que la metodología tiene la intención de alcanzar a nivel de desarrollo. En general, la consecución de una determinada etapa lleva implícita la superación de obstáculos y dificultades que posteriormente pueden hacer naufragar el proyecto o, al menos dilatar su finalización.

¿Para qué sirve?

Identificar el objetivo al cual llegar con el uso de la determinada metodología, cuáles son sus propósitos y que finalidad se desea alcanzar, con la utilización de métodos, actividades y contenidos, manejados por RUP, XP, SCRUM y RAD.

¿Por qué se eligió?

El elegir una metodología de desarrollo de software para la ejecución de un proyecto implica tomar una decisión, lo cual es muy difícil hacerlo. Es por ello que resulta de manera importante y principal conocer a que final me lleva optar por la metodología RUP, XP, SCRUM o RAD, porque la decisión que se tome servirá de guía hasta finalizar.

3.2.4. Tipos de desarrollo

¿Qué es?

Constituye una de las formas de trabajo que aplican cada una de las metodologías, donde se observa su respectivo comportamiento a medida que elevan su complejidad. Dicha situación permite un desarrollo escalonado en algún tipo de metodologías o de forma plana en otros casos.

¿Para qué sirve?

Conocer si el tipo de desarrollo que maneja la metodología, es por etapas o continúa.

¿Por qué se eligió?

RUP, XP, SCRUM y RAD. Cada una de ellas propone un tipo de desarrollo al momento de hacer su aplicación, es por eso que resulta importante conocer cómo se comporta la metodología escogida.

3.2.5. Facilidad de uso

¿Qué es?

Facilidad con que las personas pueden utilizar una metodología, con el fin de alcanzar un objetivo en concreto, refiriéndose a la claridad en el manejo entre metodología y desarrollador.

¿Para qué sirve?

Identificar la capacidad de una metodología de ser comprendida, aprendida, usada y si es atractiva para el usuario, en condiciones específicas de uso.

¿Por qué se eligió?

RUP, XP, SCRUM y RAD Cada una de ellas para su comprensión y posterior utilización, presentan esquemas o periodos en el normal desarrollo así como otras metodologías donde hay que irse adecuando a lo largo del camino. Siendo así de vital importancia conocer con anterioridad su comportamiento.

4. CUADRO COMPARATIVO ENTRE RUP - XP – SCRUM – RAD

	METODOLOGÍAS DESARROLLO DE SOFTWARE			
Criterio Compara.	**RUP**	**XP**	**SCRUM**	**RAD**
Tipo de FrameWork	Análisis, diseño, implementación y documentación de sistemas orientados a objetos.	Basado en la adaptabilidad, mayor flexibilidad, dinámico y funcional.	Gestión y desarrollo de software, basado en un proceso iterativo e incremental.	Desarrollo interactivo, construcción de prototipos y el uso de utilidades CASE.
Tipo de Revisión	En cada fase se realiza una o más iteraciones, perfeccionando así los objetivos. Si no se termina una fase no se continúa con la siguiente.	Se debe integrar como mínimo una vez al día, y realizar las pruebas sobre la totalidad del proceso.	Breve revisión diaria, donde se describen 3 cuestiones: **1.** Trabajo realizado el día anterior. **2.** Trabajo previsto a realizar. **3.** cosas que puede realizar o impedimentos.	Sus pruebas se realizan al finalizar el proceso, enfatizado en la reutilización de los componentes de los programas ya comprobados.
Objetivos	Orientado a objetos que establece las bases, plantillas y ejemplos para todos los aspectos y fases de desarrollo de software.	Basada en dar prioridad a trabajos con resultado directo. • Satisfacción cliente. • Trabajo en grupo. • Actuar sobre variables: Coste, Tiempo, Calidad y Alcance.	Indicado para proyectos en entornos complejos: • Obtener resultados prontos • Requisitos cambiantes. • Innovación y competitividad fundamentales.	• Crear o redefinir modelos existentes. • Alto rendimiento, ahorro de tiempo. • Reducción costos de desarrollo. • Mantener la calidad en todo el desarrollo.
Tipos de Desarrollo	Proceso iterativo incremental por fases: • Inicio • Elaboración • Construcción • Transición	Liviana y adaptable. Desarrollo por fases: • Planificación del proyecto. • Diseño • Codificación. • Pruebas.	• Desarrollo simple, que requiere trabajo duro. • Control de forma empírica y adaptable a la evolución del proyecto.	Desarrollo interactivo por fases: • Modelo de gestión. • Modelo de datos. • Modelo de procesos • Generación de aplicaciones.

Facilidad de uso	• Dirigido por Casos de Uso. • Establecimiento temprano de una buena arquitectura. • Iterativo e incremental. • Incremental, el trabajo se divide en mini proyectos.	• Orientada para pequeños o medianos equipos. • Para proyectos de riesgo: fecha de entrega • No apto para mucho personal. • Requisitos con probabilidad de cambiar	• No se basa en el seguimiento de un plan • Modelo Adaptable. • Construcción incremental basada en iteraciones. • No existe trabajo con diseños o abstracciones.	• Usado para estructurar, planificar y controlar • Participación activa en talleres con profesionales. • Trabajar cerca con los usuarios.

Ilustración 11. Cuadro comparativo RUP, XP, SCRUM y RAD

5. ANÁLISIS DEL CUADRO COMPARATIVO ENTRE RUP - XP – SCRUM – RAD

No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto (recursos técnicos y humanos, tiempos de desarrollo, tipo de sistemas, etc). Las metodologías tradicionales y en este caso RUP donde el proceso de desarrollo lleva asociado un marcado énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado e información detallada. Este esquema “tradicional” para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos) donde por lo general se exige un alto grado de calidad en el proceso.

Sin embargo, para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo un alta calidad. Es difícil utilizar metodologías tradicionales, con esta restricción de tiempo y flexibilidad, en respuesta a eso surgen las metodologías ágiles en este caso XP, SCRUM y RAD. Por estar especialmente orientadas a proyectos pequeños, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.

Extreme Programing (XP). Su aplicación en proyectos medianos y en las que las especificaciones no se puedan obtener hasta luego de comenzar el proyecto, aplicado a equipos relativamente pequeños. Si bien no hay un consenso en el número máximo de desarrolladores, todo parece coincidir en números no mayor a 20.

La participación e involucramiento del cliente en el proceso es fundamental, el cliente debe conocer y aprobar la metodología y destinar los recursos necesarios. De otra manera el proyecto no podrá ser llevado a cabo.

SCRUM. Permite abordar proyectos complejos desarrollados en entornos dinámicos y cambiantes de un modo flexible. Opción de gestión ideal para acometer proyectos en entornos complejos que exigen rapidez en los resultados y en los que la flexibilidad es un requisito imprescindible

Es normal que el cliente tenga una idea del producto final pero no esté muy claro cómo llegar hasta ella. Por muy bueno que sea el análisis inicial, entendemos que durante el proceso de desarrollo se puede cambiar de opinión, enfocar el proyecto de otra manera o que se puedan ocurrir nuevas ideas que complementen al concepto inicial. Es así como SCRUM permite que se adapte al cliente durante

todo el proceso de desarrollo, ya que en todo momento el cliente es participe en el proyecto.

Rapid Application Development (RAD). Hoy en día se suele utilizar para el desarrollo rápido de interfaces graficas de usuarios, tales como Glade, construcción de prototipos y el uso de utilidades CASE. Esta metodología permite prevenir presupuestos rebasados (RAD necesita un equipo disciplinado en manejo de costos) al igual permite prevenir incumplimiento de fechas (RAD involucra un equipo ordenado en el manejo de tiempos).

El grupo de trabajo se compone por una cantidad mínima de personas, incluyendo desarrolladores y usuarios de tiempo completo del sistema, así como aquellas personas involucradas con los requisitos. Los desarrolladores de RAD deben ser analistas, diseñadores y programadores.

RAD al integrar herramientas CASE es con el propósito de integrar diagramas para representar la información y crear modelos del sistema. Se crean diseños y estructuras bien detalladas, haciendo que los diagramas ayuden a visualizar los conceptos.

6. TRABAJOS FUTUROS

La comunidad académica se encuentra en un proceso de aprendizaje continuo, por ello este proyecto de investigación ofrece agilidad para investigaciones que traten temas de ingeniería de software y metodologías de desarrollo de software. Tomando como referencia y observando el comportamiento de las metodologías RUP, XP, SCRUM y RAD frente a la aplicación de unos criterios de comparación.

Los procesos de desarrollo de software están estrechamente vinculados a la creación de software, que busca la construcción de nuevos sistemas informáticos a partir de componentes, especificaciones o diseños. La relación entre procesos de desarrollo de software y metodología de desarrollo de software es evidente: disponer de la caracterización de una metodología, hará más sencilla la elección de esta, involucrando ahorro de tiempo. Concepto importante en el proceso de desarrollo de software.

CONCLUSIONES

- ✓ La utilización de metodologías de desarrollo de software es imposible omitirla, debido a la gran necesidad de control que conlleva el mismo desarrollo y para una elaboración ordenada de software, por lo tanto, seguir metodologías nos llevan a estar en competitividad en todo momento.

- ✓ Es de suma importancia conocer el modo como se interrelacionan metodologías y herramientas siguiendo un único propósito, el cual consiste en la elaboración de software de manera eficiente, ordenada y con el menor número de defectos.

- ✓ RUP, XP, SCRUM y RAD nos proporcionan fases en las cuales se encuentran guías permitiendo documentar e implementar de una manera fácil y eficiente, todo esto dentro de las respectivas etapas con que cuenta cada metodología.

- ✓ La metodología utilizada en el desarrollo de software nos proporcionan guías para poder conocer todo el camino a recorrer desde antes de empezar la implementación, con lo cual se asegura la calidad del producto final, así como también el cumplimiento en la entrega del mismo en un tiempo estipulado.

- ✓ Es de suma importancia elegir la metodología adecuada, así como las herramientas de implementación acordes, es por ello que cada una de las metodologías nos proporciona las bases para llevar al éxito el desarrollo del software.

- ✓ En el ámbito educativo universitario de pregrado, debido a los espacios cortos de tiempo y por la superficialidad de la asignatura. Al momento de desarrollar un producto software de baja complejidad, el estudiante involuntariamente asume como metodología de desarrollo de software Rapid Application Development (RAD) por la necesidad de construir rápido una aplicación y a la vez que la misma persona hace de analista, diseñador y programador.

RECOMENDACIONES

- ✓ Seguir investigando con el fin de enriquecer el Cuadro Comparativo Entre RUP, XP, SCRUM y RAD, incluir nuevas herramientas de soporte (CASE), actualizaciones de las mejoras que se realicen a cada metodología y así brindar mayor soporte a la investigación.

- ✓ Tener en cuenta esta investigación tanto docentes y estudiantes de la Universidad de Nariño al igual que personas externas enfocadas a la rama de ingeniería de software, sirviéndoles de ayuda en temas referentes a metodologías de desarrollo de software.

BIBLIOGRAFÍA

- ✓ ABRIL, David. Parte I Metodología de Desarrollo de Software. Especialización Tecnológica en Desarrollo de Aplicaciones para Dispositivos Móviles 2011. Análisis Documental. Pag. 3

- ✓ ARBOLEYA, Hernán. Propuesta de Ciclo de Vida y Mapa de Actividades para Proyectos de Explotación de Información. Universidad Nacional de Lanus (UNLA), Remedios de Escalada (Argentina) 2013. Trabajo de Grado (Licenciado en Sistemas). Pag. 9-11.

- ✓ CARDINALES, Cristina. Cuadro Comparativo de las Metodologías. República Bolivariana de Venezuela 2012. Documentación. Ministerio del Poder para la Educación Universitaria, Fundación Misión Sucre. Pag 3.

- ✓ CELIS Zaida. Facultad de filosofía y letras, Universidad Nacional Autónoma de México, XI Congreso Nacional de Investigación Educativa, Mexico DF.

- ✓ GALICIA Bic. Centro Europeo de Empresas e Innovación. Documentación. Madrid 2008.

- ✓ GIL, Gustavo. Herramienta para implementar LEL y escenarios (TILS), ingeniería de requerimientos. Universidad Nacional de la Plata, Argentina 2002.

- ✓ Instituto Nacional de Tecnologías de la comunicación. Gobierno de España. Ingeniería del Software: Metodologías y Ciclos de Vida. Madrid (España) 2009. Documentación (Laboratorio Nacional de Calidad del Software). Pag. 46-49

- ✓ Introducción ingeniería de software, procesos iterativos e incrementales. http://esalas334.blogspot.es/1193761920/procesos-iterativos-e-incrementales/

- ✓ MEZA, Daniel. Fundamentos de Desarrollo de Sistemas. Instituto Tecnológico de Comitán. Chiapas (México) 2010. Documentación (Ingeniería en Sistemas Computacionales). Pag. 21-24.

- ✓ PEDREIRA, Oscar. Revista española de innovación-Calidad del software. España. 2007

- ✓ PRESSMAN , Roger S. Ingeniería del Software Un Enfoque Práctico. McGraw-Hill. México 2005.

- ✓ RODRIGUEZ Rosangélica, GONZÁLEZ Henyeliz, ROSAS Victor. Metodología de Desarrollo Rápido de Aplicación. Universidad Nacional Experimental Politécnica de la Fuerza Armada Bolivariana. Núcleo Sucre-Carúpano (Venezuela) 2013. Documentación (Ingeniería de Sistemas).

- ✓ RUIZ, Yadira. Principales metodologías pesadas y orientadas a objetos en el desarrollo de software. 14 convención científica de ingeniería y arquitectura, México 2008. Investigación documental.

- ✓ VILLALBA Erika, RAMÓN Eder. Desarrollo de Sistemas Con Metodología RUP (RATIONAL UNIFIED PROCES). Universidad Nacional Autónoma de México. México D.F. Trabajo de Grado (Ingeniero en Computación). Pag. 99-106.

- ✓ VIRRUETA, Alejandra. Metodologías de Desarrollo de Software. Instituto Tecnológico Superior de Apatzingán (ITSA), Michoacán (México) 2010. Investigación Documental (Ingeniería en Informática). Pag. 2

- ✓ VIRRUETA, Alejandra. Metodologías de Desarrollo de Software. Instituto Tecnológico Superior de Apatzingán (ITSA), Michoacán (México) 2010. Investigación Documental (Ingeniería en Informática). Pag. 5-8

- ✓ WIKIPEDIA-la enciclopedia libre, 1994, http://es.wikipedia.org/wiki/Investigaci%C3%B3n#cite_note-1

- ✓ WIKIPEDIA-la enciclopedia libre, 2003, http://es.wikipedia.org/wiki/Revista_cient% C3%ADfica

- ✓ WIKIPEDIA-la enciclopedia libre, 2000. http://es.wikipedia.org/wiki/Herramienta_CASE

- ✓ ZAMBRANO Solange, LEÓN Carlos, GÓMEZ Laura. Programación Extrema “XP”. Universidad de Los Andes. Mérida (Venezuela) 2010. Informe (Maestría en Educación, Mención Informática y Diseño Instruccional).

ANEXOS

ANEXO A: DOCUMENTACIÓN DISPONIBLE

Los valores consignados en la siguiente tabla, proveen información sobre el número de documentos pertinentes y que tienen como fuente de origen ya sea biblioteca o la web. El valor 0 representa que no existe ningún tipo de documentación, el valor 10 como mayor calificación que permitirá culminar con el desarrollo de la investigación.

Se realizó el análisis teniendo como base el sistema de ponderación, que según este estudio permiten clasificar la documentación de la siguiente manera:

Guía 20%: "Considerado como una herramienta analítica que facilita información para iniciar la aplicación de la metodología"[23]. Con el fin de observar cómo se inicializa la aplicación de la metodología.

Libro 45%: "Material curricular para facilitar el proceso de enseñanza-aprendizaje. Factor fundamental en la formación de los sujetos y, por tanto, un elemento indispensable para la transmisión del conocimiento"[24]. Contenidos completos y detallados, que permiten hacer un análisis exhaustivo para la investigación.

Revista Científica 35%: "Publicación periódica en la que se intenta recoger el progreso de la ciencia, entre otras cosas incluyendo informes sobre las nuevas investigaciones"[25]. Actividad netamente investigadora, las publicaciones representadas contenidos de calidad y previamente verificados.

El consolidado llamado "Valor", a partir de la ponderación es tomado redondeando al entero más próximo.

	Tipo de Documentación			
Metodología	**Guía 20%**	**Libro 45%**	**Revista Científica 35%**	**Valor**
Modelo en Cascada.	7	4	3	4
Modelo en Espiral.	8	3	9	6
Proceso Unificado Rational (RUP)	7	8	9	8
Programación Extrema (XP)	10	8	7	8
Desarrollo Manejado por Rasgos.	6	2	5	4
Incremental.	9	3	7	6
Desarrollo Rápido de Aplicaciones (RAD)	10	5	7	7
SCRUM	10	7	8	8

[23] GALICIA Bic. Centro Europeo de Empresas e Innovación. Documentación. Madrid 2008
[24] CELIS Zaida. Facultad de filosofía y letras, Universidad Nacional Autónoma de México, XI Congreso Nacional de Investigación Educativa, Mexico DF.
[25] WIKIPEDIA-la enciclopedia libre: http://es.wikipedia.org/wiki/Revista_cient% C3%ADfica

ANEXO B: EXPERIENCIAS PRÁCTICAS

Los valores consignados en la siguiente tabla, proveen información sobre el número de experiencias prácticas que tienen como fuente de origen empresas, trabajo de grado de pregrado y trabajo de grado de postgrado. El valor 0 representa que no existe ningún tipo de experiencias prácticas, el valor 10 como mayor calificación que me permitirá finalizar con éxito la investigación.

Se realizó el análisis teniendo como base el sistema de ponderación, que según este estudio permiten clasificar las experiencias prácticas de la siguiente manera:

Trabajo de grado pregrado 30%: "Actividad humana orientada a la obtención de nuevos conocimientos y su aplicación para la solución a problemas"[26]. Trabajo de investigación en donde se vierten los conocimientos propios del estudiante de pregrado, primer producto publicado.

Trabajo de grado postgrado 50%: "Proceso más formal, sistémico e intensivo de llevar a cabo el método científico de análisis. Comprende una estructura de investigación más sistémica"[27]. Trabajo de investigación, donde el contenido es conocimiento nuevo.

Experiencias empresariales 20%: Identificar el éxito o fracaso en la aplicación de una metodología. Experiencias incompletas por políticas de privacidad de las empresas.

El consolidado llamado "Valor", a partir de la ponderación es tomado redondeando al entero más próximo.

	Experiencias Prácticas			
Metodología	**Trabajo de grado pregrado 30%**	**Trabajo de grado postgrado 50%**	**Experiencias empresariales 20%**	**Valor**
Modelo en Cascada.	5	3	4	4
Modelo en Espiral.	5	2	2	3
Proceso Unificado Ratioanal (RUP)	8	7	5	7
Programación Extrema (XP)	4	8	3	6
Desarrollo Manejado por Rasgos.	6	2	2	3
Incremental.	4	3	4	4
Desarrollo Rápido de Aplicaciones (RAD)	5	7	4	6
SCRUM	7	8	4	7

[26] WIKIPEDIA-la enciclopedia libre: http://es.wikipedia.org/wiki/Investigaci%C3%B3n#cite_note-1
[27] Best, J,W. cit por EGG, E. s/f

ANEXO C: HERRAMIENTAS CASE DE SOPORTE

Los valores consignados en la siguiente tabla, proveen información sobre el número de herramientas CASE de soporte que tienen como fuente de origen la web. El valor 0 representa que no existe ningún tipo de herramienta CASE, el valor 10 como mayor calificación.

Se realizó el análisis teniendo como base el sistema de ponderación, que según este estudio permiten clasificar las experiencias prácticas de la siguiente manera:

Tipos de Herramientas CASE[28]:

Upper CASE (U-CASE) 40%: Herramientas que ayudan en las fases de planificación, análisis de requisitos y estrategia del desarrollo, usando, entre otros diagramas UML.

Middle CASE (M-CASE) 25%: Herramientas para automatizar tareas en el análisis y diseño de la aplicación.

Lower CASE (L-CASE) 35%: Herramientas que semi-automatizan la generación de código. Aquí pueden incluirse las herramientas de desarrollo rápido de aplicaciones.

El consolidado llamado "Valor", a partir de la ponderación es tomado redondeando al entero más próximo.

	Tipo de Documentación			
Metodología	**Upper CASE (U-CASE) 40%**	**Middle CASE (M-CASE) 25%**	**Lower CASE (L-CASE) 35%**	**Valor**
Modelo en Cascada.	3	2	0	2
Modelo en Espiral.	2	3	1	2
Proceso Unificado Ratioanal (RUP)	6	3	2	4
Programación Extrema (XP)	4	3	1	3
Desarrollo Manejado por Rasgos.	1	2	0	1
Incremental.	2	2	1	2
Desarrollo Rápido de Aplicaciones (RAD)	2	2	4	3
SCRUM	5	5	2	4

[28] WIKIPEDIA-la enciclopedia libre: http://es.wikipedia.org/wiki/Herramienta_CASE

ANEXO D: SELECCIÓN CRITERIOS DE COMPARACIÓN

La selección de criterios de comparación se basó en el análisis de los autores expertos en ingeniería de software entre los cuales tenemos:

- Pressman: “La garantía de calidad del software es un diseño planificado y sistémico de acciones que se requieren para asegurar la calidad del software”.

- IEEE Computer Society – SWEBOK: “El ingeniero de software, ante todo, necesita determinar el Objetivo verdadero del software. En cuanto a esto, es de capital importancia tener presente los requerimientos del cliente y aquellos que estos incluyen como requerimientos de calidad, no únicamente los requerimientos funcionales”

- Walter A. Shewart: ”El desarrollo de proyectos tiene dos aspectos. El primero tiene que ver con la consideración de la calidad de una cosa como una realidad objetiva, la otra tiene que ver con lo que pensamos, sentimos o creemos como resultado de la realidad objetiva”.

- Palacio Juan – SCRUMMANAGER: “La gestión de proyectos predictiva o clásica es una disciplina formal de gestión, basada en la planificación ejecución y seguimiento a través de procesos sistémicos y repetibles”.

Dichos criterios se obtuvieron al realizar una lectura detallada de cada una de las metodologías seleccionadas. En donde se identificaron y reconocieron los siguientes:

Proceso iterativo: “consiste en la entrega de versiones, donde la primera versión debe contener los requerimientos del usuario. En posteriores entregas se da solución a problemas encontrados”[29].

Tipo de Framework: “Es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación, con componentes personalizables e intercambiables”[30].

[29] Introducción ingeniería de software, procesos iterativos e incrementales. http://esalas334.blogspot.es/1193761920/procesos-iterativos-e-incrementales/
[30] Wikipedia-la enciclopedia libre: http://es.wikipedia.org/wiki/Framework.

Adaptación en el proceso: "La adaptación del proceso software (en inglés, software process tailoring) consiste en adaptar y particularizar la descripción general del proceso para obtener un nuevo proceso adaptado, aplicable en un entorno alternativo y probablemente menos general"[31].

Tipo de revisión: Cada metodología propone examinar el trabajo, al igual de cómo se realiza dicho trabajo y en qué momento se ejecuta, cuyo objetivo principal es la búsqueda de errores a tiempo, con el fin de proponer y evaluar las soluciones alternativas o las mejoras en la aplicación de la metodología.

Objetivos: Meta o finalidad que la metodología tiene la intención de alcanzar a nivel de desarrollo. En general, la consecución de una determinada etapa lleva implícita la superación de obstáculos y dificultades que posteriormente pueden hacer naufragar el proyecto o, al menos dilatar su finalización.

Proceso interactivo: "ejecución de un proceso en un *procesador* que requiere la interacción con el usuario. Por oposición, se denomina Procesamiento por lotes (Batch) a la ejecución de un proceso que no requiere la intervención del mismo"[32].

Tipo de desarrollo: Constituye una de las formas de trabajo que aplican cada una de las metodologías, donde se observa su respectivo comportamiento a medida que elevan su complejidad. Dicha situación permite un desarrollo escalonado en algún tipo de metodologías o de forma plana en otros casos.

Definición previa de necesidades y requerimientos: "el proceso de recopilar, analizar y verificar las necesidades para el desarrollo de un proyecto, es llamado ingeniería de requerimientos. La meta es entregar una especificación de requerimientos de software correcta y completa"[33].

Facilidad de uso: Facilidad con que las personas pueden utilizar una metodología, con el fin de alcanzar un objetivo en concreto. Refiriéndose a la claridad en el manejo entre metodología y desarrollador.

---

[31] PEDREIRA, Oscar. Revista española de innovación-Calidad del software. España. 2007
[32] Wikipedia-la enciclopedia libre: http://es.wikipedia.org/wiki/Procesamiento_interactivo
[33] GIL, Gustavo. Herramienta para implementar LEL y escenarios (TILS), ingeniería de requerimientos. Universidad Nacional de la Plata, Argentina 2002.

Terminado la descripción de las características principales de las metodologías RUP, XP, SCRUM y RAD. Se procede a verificar si dicha particularidad es aplicable en las cuatro metodologías, con el fin de decidir si es aceptada o no y así poder realizar la comparación bajo igualdad de condiciones.

Criterio de comparación.	**RUP.**	**XP.**	**SCRUM.**	**RAD.**	**Estado del criterio de comparación.**
Proceso iterativo	✓	x	✓	x	No aceptado
Tipo de Framework.	✓	✓	✓	✓	**Aceptado**
Adaptación en el proceso.	x	✓	✓	x	No aceptado
Tipo de revisión.	✓	✓	✓	✓	**Aceptado**
Objetivos.	✓	✓	✓	✓	**Aceptado**
Proceso interactivo	x	x	x	✓	No aceptado
Tipo de desarrollo.	✓	✓	✓	✓	**Aceptado**
Definición previa de necesidades y requerimientos.	✓	✓	x	x	No aceptado
Facilidad de uso	✓	✓	✓	✓	**Aceptado**

Como resultado final y con estado “aceptado” se tiene: tipo de FrameWork, tipo de revisión, objetivos y tipo de desarrollo, características comunes que posteriormente son llamados criterios de comparación, los cuales son evidenciables en las cuatro metodologías a estudiar.