PROCESOS DE DESARROLLO DE SOFTWARE Y MATERIALES EDUCATIVOS COMPUTARIZADOS

SOFTWARE DEVELOPMENT PROCESS AND COMPUTER EDUCATIONAL MATERIALS

Victor Esteller
vestelle@uc.edu.ve
Elsy Medina
emedina@uc.edu.ve
Universidad de Carabobo, Valencia, Edo. Carabobo, Venezuela

Recibido: 01/11/2011
Aceptado: 02/03/2012

RESUMEN

En este artículo se abordan principios teóricos relacionados con los procesos de desarrollo de software y su importancia en el diseño y elaboración de materiales educativos computarizados en contextos virtuales de aprendizaje. Se enfatiza la importancia de la documentación del software y la especificación adecuada de requerimientos del usuario como fundamento para alcanzar máximos niveles de satisfacción, obtener productos finales de calidad reutilizables, aprovechando el esfuerzo y reduciendo los tiempos de desarrollo. En este trabajo se hizo una investigación acción; es producto de experiencias y disertaciones realizadas en la cátedra de informática IV, en el Departamento de Informática de la Facultad de Educación de la Universidad de Carabobo.

Palabras clave: procesos de desarrollo de software, materiales educativos computarizados, calidad de software.

ABSTRACT

This article addresses theoretical principles related to software development processes and their importance in the design and development of computerized educational materials in virtual

learning contexts. It emphasizes the importance of software documentation and the user's requirements to achieve maximum satisfaction levels and quality end recycling products spending less effort and time. It was an action research and it is the product of different experiences and presentations made in the subject of Computer Science IV, Department of Informatics, School of Education at the University of Carabobo.

Key words: Software development processes. Educational materials. Computer, software quality.

1. Consideraciones previas

Sommerville (2007), afirma que el desarrollo de un software tiene como objetivo la elaboración de un producto que reúna los requisitos del cliente y/o usuario. Un esquema representativo puede apreciarse en la figura Nº 1. Este proceso es intelectual, determinado por la creatividad y juicio del equipo interdisciplinario que participa en la planificación y ejecución del mismo. Aunque un proyecto de desarrollo de materiales educativos computarizados se puede comparar en muchos aspectos a cualquier otro proyecto de ingeniería, en el desarrollo de software hay una serie de desafíos adicionales, relativos esencialmente a la naturaleza del producto final.

Figura 1. Proceso de desarrollo de software. Fuente Sommerville (2007)

2. Modelos de proceso software

El referido Sommerville, define modelo de proceso de software como "Una representación simplificada de un proceso de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto; un modelo de procesos de *software* es una abstracción de un proceso real."

Los modelos genéricos no son descripciones definitivas de procesos de *software;* sin embargo, son abstracciones útiles que pueden ser utilizadas para explicar diferentes enfoques del desarrollo de *software,* específicamente para la creación de materiales educativos computarizados. A continuación se esbozan algunos modelos conocidos:

2.1 Diseñar y corregir (code-and-fix)

Este es el modelo básico utilizado en los inicios del desarrollo de *software.* Contiene dos pasos:

- Codificar y diseñar.
- Corregir problemas en el diseño.

Se trata de primero implementar algo de contenido y, luego, pensar acerca de requisitos, diseño, validación, y mantenimiento.

Este modelo tiene tres problemas substanciales en la elaboración de materiales educativos computarizados:

- Después de un número de correcciones, el diseño puede tener una muy mala estructura, lo que hará que los arreglos sean muy costosos.
- Frecuentemente, aun el *software* bien diseñado, no se ajusta a las necesidades del usuario, por lo que es rechazado o su reconstrucción es muy costosa.
- El diseño es difícil de reparar por su escasa preparación en cuanto a pruebas y modificación.

2.2. Modelo en cascada

El primer modelo de desarrollo de *software* que se publicó se derivó de otros procesos de ingeniería. Éste toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y las representa como fases separadas del proceso, fundamento sustentado en Sommerville.

El modelo en cascada consta de las siguientes fases:

1. Definición de los requisitos: los servicios, restricciones y objetivos son establecidos con los usuarios del sistema. Se busca hacer esta definición en detalle.

2. Diseño de software: se divide el material educativo computarizado en sistemas de software o hardware. Se establece la arquitectura total del material. Se identifican y describen las abstracciones y relaciones de los componentes del sistema.
3. Implementación y pruebas unitarias: construcción de los módulos y unidades del material educativo computarizado. Se realizan pruebas de cada unidad.
4. Integración y pruebas del sistema: se integran todas las unidades, se prueban en conjunto, se entrega y aplica el conjunto probado al cliente.
5. Operación y mantenimiento: generalmente es la fase más extensa. El material educativo computarizado es puesto en marcha y se realiza la corrección de errores descubiertos. Se realizan mejoras de implementación. Se identifican nuevos requisitos.

La interacción entre fases puede observarse en la figura 2. Cada fase tiene como resultado documentos que deben ser aprobados por el usuario. Una fase no comienza hasta que termine la fase anterior y generalmente se incluye la corrección de los problemas encontrados en fases previas.

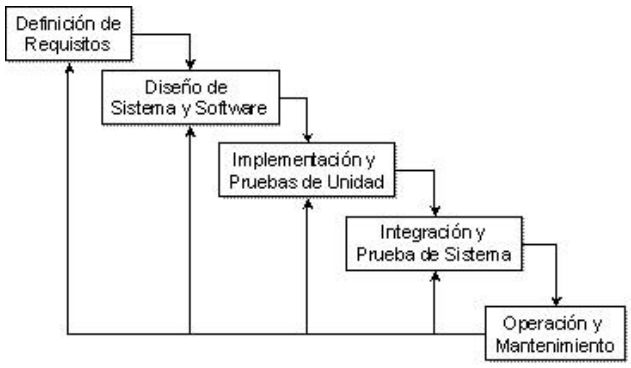

Figura 2. Modelo de desarrollo en cascada. Fuente Sommerville (2007).

En la práctica, este modelo no es lineal, e involucra varias iteraciones e interacción entre las distintas fases de desarrollo. Algunos problemas que se observan en el modelo de cascada son:

- Las iteraciones son costosas e implican rehacer trabajo debido a la producción y aprobación de documentos.
- Aunque son pocas iteraciones, es normal congelar parte del desarrollo y continuar con las siguientes fases.
- Los problemas se dejan para su posterior resolución, lo que lleva a que estos sean ignorados o corregidos de una forma poco elegante.
- Existe una alta probabilidad de que el material educativo computarizado no cumpla con los requisitos del usuario por el largo tiempo de entrega del producto.
- Es inflexible a la hora de evolucionar para incorporar nuevos requisitos. Es difícil responder a cambios en los requisitos.
- Este modelo sólo debe usarse si se entienden a plenitud los requisitos. Aún se utiliza como parte de proyectos grandes.

3. Desarrollo evolutivo

La idea detrás de este modelo es el desarrollo de una implantación del sistema inicial, exponerla a los comentarios del usuario, refinarla en versiones hasta que se desarrolle el material educativo computarizado adecuado. En la figura 3 se observa como las actividades concurrentes: especificación, desarrollo y validación, se realizan durante el desarrollo de las versiones hasta llegar al producto final, de acuerdo con el ya citado, Sommerville (2007).

Una ventaja de este modelo es que se obtiene una rápida realimentación del usuario, ya que las actividades de especificación, desarrollo y pruebas se ejecutan en cada iteración.

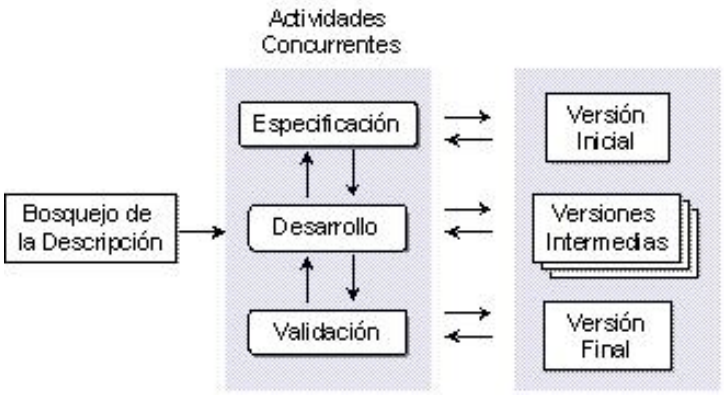

Figura 3. Modelo de desarrollo evolutivo. Fuente Sommerville (2007).

Algunas características favorables de este modelo:

- La especificación puede desarrollarse de forma creciente.
- Los usuarios y desarrolladores logran un mejor entendimiento del sistema. Esto se refleja en una mejora de la calidad del material educativo computarizado.
- Es más efectivo que el modelo de cascada, ya que cumple con las necesidades inmediatas del cliente.
- Desde una perspectiva de ingeniería y didáctica se identifican los siguientes problemas:
    - Proceso no visible: los profesores necesitan entregas para medir el progreso. Si el sistema se necesita desarrollar rápido, no es efectivo producir documentos que reflejen cada versión del sistema.
    - Material educativo computarizado débilmente estructurado: los cambios continuos pueden ser perjudiciales para la estructura del software lo que generaría grandes costos en cuanto a mantenimiento se refiere.
    - Se requieren técnicas y herramientas: para el rápido desarrollo se necesitan herramientas que podrían ser incompatibles con otras o que pocos usuarios tengan la competencia de utilizar.

Este modelo es efectivo en proyectos pequeños o medianos con poco tiempo para su desarrollo y sin generar documentación para cada versión.

Para proyectos largos es mejor combinar lo mejor del modelo de cascada y evolutivo: se puede hacer un prototipo global del sistema y posteriormente reimplementarlo con un acercamiento más estructurado. Los subsistemas con requisitos bien definidos y estables se pueden programar utilizando cascada y la interfaz de usuario se puede especificar utilizando un enfoque exploratorio.

4. Desarrollo basado en reutilización

Como su nombre lo indica, es un modelo fuertemente orientado a la reutilización. Este modelo consta de 4 fases ilustradas en la figura 4, Pressman (2002). A continuación se describe cada fase:

1. Análisis de componentes: se determina qué componentes pueden ser utilizados para el material educativo computarizado en cuestión. Casi siempre hay que hacer ajustes para adecuarlos.
2. Modificación de requisitos: se adaptan (en lo posible) los requisitos para concordar con los componentes de la etapa anterior. Si no se puede realizar modificaciones en los requisitos, hay que seguir buscando componentes más adecuados (fase 1).
3. Diseño del sistema con reutilización: se diseña o reutiliza el marco de trabajo para el material educativo computarizado. Se debe tener en cuenta los componentes localizados en la fase 2 para diseñar o determinar este marco.
4. Desarrollo e integración: el material educativo computarizado que no puede comprarse, se desarrolla. Se integran los componentes y subsistemas. La integración es parte del desarrollo en lugar de una actividad separada.

Entre las ventajas de este modelo se pueden mencionar:

- Disminuye el costo y esfuerzo de desarrollo.
- Acorta el tiempo de entrega.
- Reduce los riesgos durante el desarrollo.

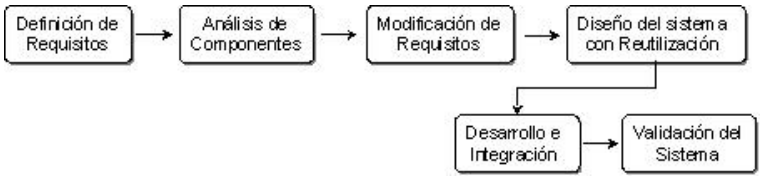

Figura 4. Desarrollo basado en reutilización de componentes. Fuente Pressman (2002).

Desventajas de este modelo:

- Los "compromisos" en los requisitos son inevitables, por lo que el material educativo computarizado puede que no cumpla con las expectativas del cliente.
- Las actualizaciones de los componentes adquiridos no están en poder de los desarrolladores del sistema.

5. Procesos iterativos

A continuación, se expondrán dos enfoques híbridos, especialmente diseñados para el soporte de las iteraciones:

- Desarrollo incremental.
- Desarrollo en espiral.

5.1. Desarrollo incremental

Mills (1980), sugirió el enfoque incremental de desarrollo como una forma de reducir la repetición del trabajo en el proceso de desarrollo y dar oportunidad de retrasar la toma de decisiones en los requisitos hasta adquirir experiencia con el sistema (ver figura 5). Es una combinación del modelo de cascada y modelo evolutivo. Reduce el rehacer trabajo durante el proceso de desarrollo y da oportunidad para retrasar las decisiones hasta tener experiencia con el material educativo computarizado. Durante el desarrollo de cada incremento se puede utilizar el modelo de cascada o evolutivo, dependiendo del conocimiento que se tenga sobre los requisitos a implementar. Si se tiene un óptimo conocimiento, se puede optar por cascada, si es dudoso, evolutivo.

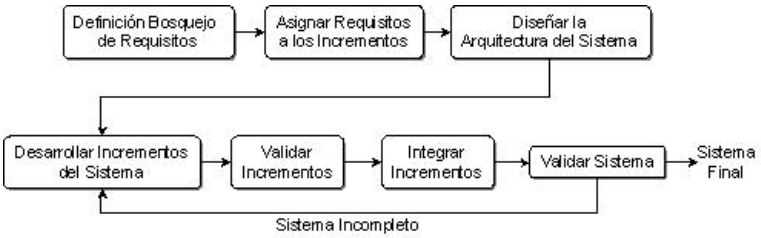

Figura 5. Modelo de desarrollo iterativo incremental. Fuente Mills (1980)

Entre las ventajas del modelo incremental se encuentran:

- Los clientes no esperan hasta el fin del desarrollo para utilizar el material educativo computarizado. Pueden empezar a usarlo desde el primer incremento.
- Los clientes pueden aclarar los requisitos que no tengan definido conforme se manifiestan las entregas del material educativo computarizado.
- Se disminuye el riesgo de fracaso de todo el proyecto, ya que se puede distribuir en cada incremento.
- Las partes más importantes del sistema son entregadas primero, por lo cual se realizan más pruebas en estos módulos y se disminuye el riesgo de fallos.

Algunas de las desventajas identificadas para este modelo son:

- Cada incremento debe ser pequeño para limitar el riesgo.
- Cada incremento debe aumentar la funcionalidad.
- Es difícil establecer las correspondencias de los requisitos contra los incrementos.
- Es difícil detectar las unidades o servicios genéricos para todo el material educativo computarizado.

5.2. Desarrollo en espiral

Sommerville, en su modelo de desarrollo en espiral (ver figura 6) es considerado actualmente uno de los más conocidos. El ciclo de desarrollo se representa como una espiral, en lugar de una serie de actividades sucesivas con retrospectiva de una actividad a otra.

Cada ciclo de desarrollo se divide en cuatro fases:

- Definición de objetivos: se definen los objetivos. Se definen las restricciones del proceso y del producto. Se realiza un diseño detallado del plan administrativo. Se identifican los riesgos y se elaboran estrategias alternativas dependiendo de estos.
- Evaluación y reducción de riesgos: se realiza un análisis detallado de cada riesgo identificado. Pueden desarrollarse prototipos para disminuir el riesgo de requisitos dudosos. Se llevan a cabo los pasos para reducir los riesgos.
- Desarrollo y validación: se escoge el modelo de desarrollo después de la evaluación del riesgo. El modelo que se utilizará (cascada, evolutivo, etc.) Depende del riesgo identificado para esa fase.
- Planificación: se determina si continuar con otro ciclo. Se planea la siguiente fase del proyecto.

Este modelo a diferencia de los otros toma en consideración explícitamente el riesgo, esta es una actividad importante en la administración del proyecto.

El ciclo de vida inicia con la definición de los objetivos. De acuerdo a las restricciones se determinan distintas alternativas. Se identifican los riesgos al sopesar los objetivos contra las alternativas. Se evalúan los riesgos con actividades como análisis detallado, simulación, prototipos, entre otros. Se desarrolla un poco el material educativo computarizado. Se planifica la siguiente fase.

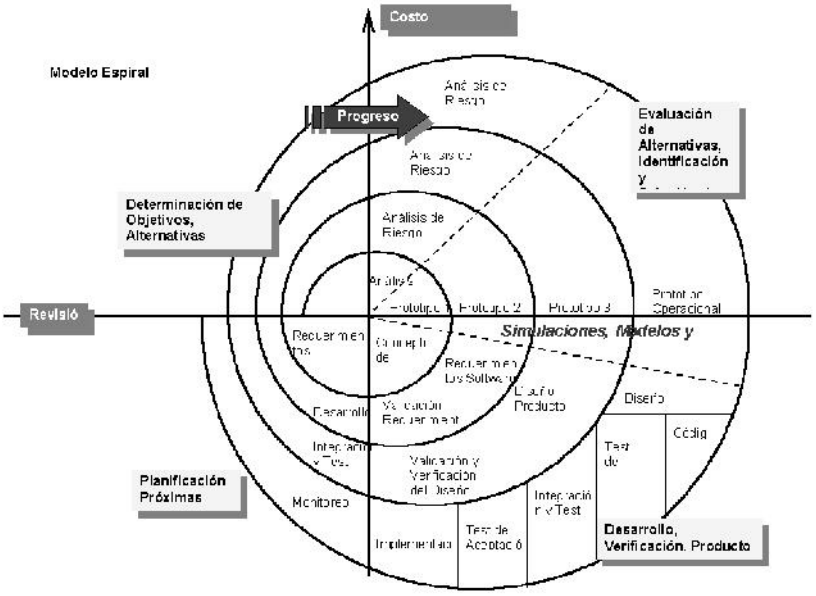

Figura 6: Modelo de desarrollo en espiral. Fuente Sommerville (2007).

6. Proceso Unificado de Desarrollo de Software

Es un proceso de *software* genérico que puede ser utilizado para una gran cantidad de tipos de sistemas de software, en este caso particular los materiales educativos computarizados. Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible.

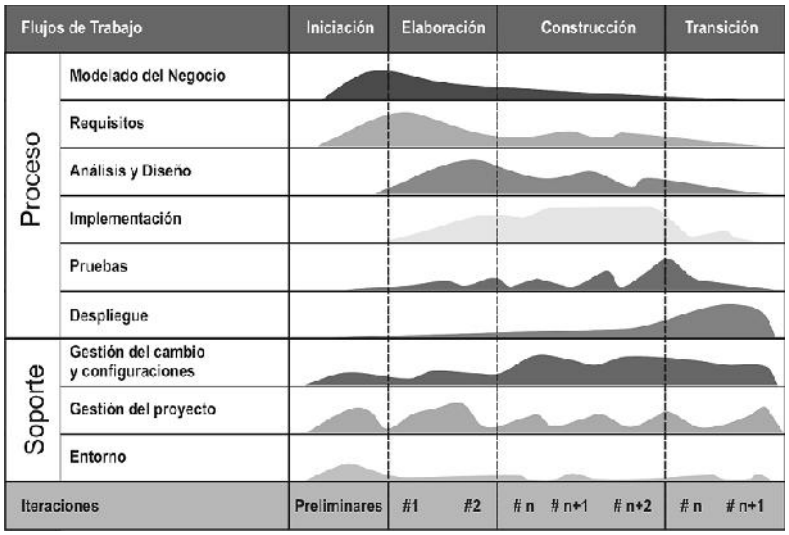

Figura 7. Proceso unificado. Fuente Weitzenfield (2005).

Iterativo e incremental.- Compuesto de cuatro fases: Inicio, elaboración, construcción y transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio sólo consta de varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del material educativo computarizado en desarrollo. Cada una de estas iteraciones se dividen en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: análisis de requisitos, diseño, implementación y prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto. Este proceso ayuda al diseñador a enfocarse en las metas correctas, tales como claridad (entendimiento), flexibilidad en los cambios futuros (mantenibilidad) y re-uso.

Enfocado en los riesgos.- El proceso unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de

cada iteración, en especial los de la fase de elaboración, deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

7. A manera de cierre

¿Cuál es el modelo de proceso más adecuado?

Cada proyecto de software para elaborar materiales educativos computarizados requiere de una forma particular de abordar el problema. Las propuestas comerciales y académicas actuales promueven procesos iterativos, donde cada iteración puede utilizar uno u otro modelo de proceso, considerando un conjunto de criterios (por ejemplo: grado de definición de requisitos, tamaño del proyecto, riesgos identificados, entre otros).

7.1 Ubicación de los modelos en relación al nivel de formalidad requerido para elaborar materiales educativos computarizados

Se puede observar en la figura 8 la disposición que surge de colocar al Modelo en Cascada, el RUP, las metodologías ágiles, el modelo en espiral, el modelo por prototipos, y el modelo iterativo en la recta correspondiente.

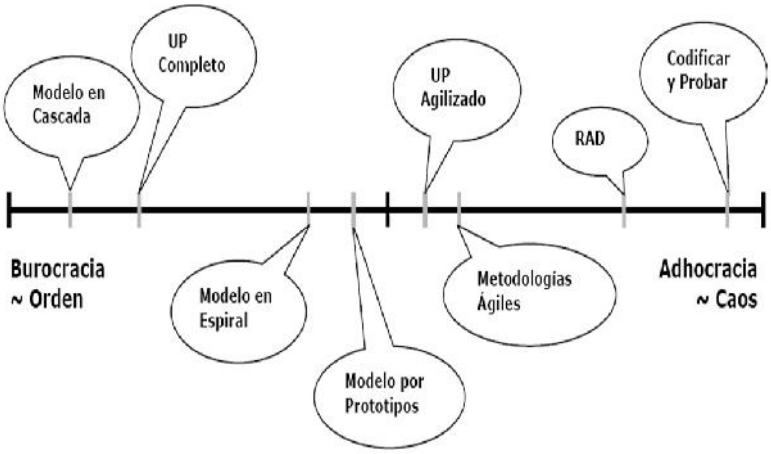

Figura 8. Ubicación de las metodologías en relación al nivel de formalidad requerido para elaborar materiales educativos computarizados. Fuente Autor (2011)

En la figura se observa que el *modelo en cascada* es el que más contribuye a un régimen burocrático en el que los proyectos son planificados en base a tareas a realizar por los recursos; fomenta el orden de la organización mediante la clara división en etapas con sus inputs/outputs correspondientes. Asimismo, otro de los procesos más utilizados en la actualidad como es el *proceso unificado,* en una forma completa, construyendo todos los artefactos especificados y realizando todas las actividades planteadas. En otras palabras, estos procesos pretenden definir todas las actividades, roles, entregables, y demás aspectos relacionados en un proyecto de *software* para elaborar materiales educativos computarizados y generar resultados predecibles a lo largo del tiempo. Mediante la clara definición de aquellas cuestiones involucradas y la posibilidad de controlar el proceso para ajustarlo ante posibles desvíos, se llega a la burocracia. El *modelo en cascada* ejemplifica este enfoque al ser un modelo orientado a documentos, en donde el progreso se mide sobre la base de un conjunto de fases bien definidas y al grado de avance de la documentación correspondiente. Las personas son parte irremplazables en este esquema, y como tales alcanzan con definir los roles y las actividades que estas realizan para completar el marco de desarrollo. A medida que se aleja de la burocracia se encuentran los modelos con un enfoque principal hacia los entregables y hacia el rápido feedback de los clientes. El UP Agilizado no es otra cosa que una versión del UP personalizada para grupos pequeños y con feedback frecuente de los interesados.

Todas estas metodologías se encuentran hacia la izquierda del punto medio entre Burocracia y Adhocracia. Si se mueve desde ese punto hacia el Caos se muestra con el tema central de trabajo, las metodologías y modelos ágiles. Estos modelos reconocen las características inherentes de complejidad del *software,* y el carácter empírico que debe tener un proceso de desarrollo del mismo. En estos, las personas son de primera magnitud y se prefiere sacrificar el proceso y darles libertad a las personas. Sin embargo, esto no significa que se termina en un proceso de codificar y probar, liderado por programadores rebeldes que se pasan el día generando código. Por esta razón, el punto en que se ubican las mismas está bastante alejado del Caos en que la aleatoriedad es la norma. Otra

característica que diferencia estos procesos de los anteriores es el cambio existente entre un proceso empírico y un proceso determinista o definido. Los procesos más burocráticos tienden a tener definidos todos los aspectos del desarrollo. Esto se contradice con la naturaleza compleja del software, para lo cual se adaptaría con mayor eficiencia un proceso empírico.

Estos modelos surgen de la observación y la experiencia obtenida a lo largo de los años, sin basarse en leyes de la naturaleza o principios teóricos fundamentados. Es por esto que son asociados a un enfoque pragmático para desarrollar software.

8. Referencias

Debrauwer, L. y Van der Heyde, F. (2009). UML 2, Iniciación, ejemplos y ejercicios corregidos. Segunda Edición. Ediciones ENI. España.

Esteller, V. y Medina E. (2009). Evaluación de cuatro modelos instruccionales para la aplicación de una estrategia didáctica en el contexto de la tecnología. Revista de Tecnología de Información y Comunicación en Educación. Universidad de Carabobo. Volumen 3 No I. ISSN: I856-7576. Depósito Legal pp200702CA2520.

Esteller, V. (2009). Técnicas y herramientas para el modelado de software. Isbn: 978-980-12-3894-2. Depósito Legal: IF041220090042918. Universidad de Carabobo.

Esteller, V. (2009). Evaluación de cuatro modelos instruccionales para la aplicación de una estrategia didáctica en el contexto de la tecnología. Revista de Tecnología de Información y Comunicación en Educación. Universidad de Carabobo. Volumen 3 No I. ISSN: I856-7576. Depósito Legal pp200702CA2520.

ISO-FCD-25010, (2009). Software engineering, Software product quiality, Requiriments and Evaluation (SQeaRE) Quality mode.

Joyanes, L. (2003). Fundamentos de programación. Libro de problemas (2ª ed.): algoritmos, estructuras de datos y objetos. Mcgraw-hill. Interamericana de España, s.a.

Pressman, R. (2002). Ingeniería del Software: Un enfoque práctico, McGraw Hill.

Sommerville, Ian. (2007). Ingeniería de Software. Prentice Hall.

Weitzenfield, A. (2005). Ingeniería de software orientada a objetos con UML, JAVA e INTERNET. Editorial Thomson.