Ciclos de Vida de Sistemas y Tipos de Proyecto.

TABLA DE CONTENIDO

1. Historial de cambios.

Rev.	Pág.	Sección	Resumen del cambio	Responsable del cambio	Aprobó	Fecha
PA1			Primer borrador	David Adame	Oscar Mesta	13/10/2008
PA2			Conclusión del primer borrador	David Adame	Oscar Mesta	15/10/2008
A			Aprobado para su liberación e implementación	David Adame	Oscar Mesta	16/10/2008

2. Introducción.

2.1 Propósito del Documento.

El documento tiene como propósito presentar los ciclos de vida de sistemas para el desarrollo de software utilizados por tipo de proyecto en la Universidad Autónoma de Coahuila.

3. Tipos de Proyectos.

3.1 Proyectos de Corto Alcance.

Son los proyectos de desarrollo de software a la medida cuya característica particular es que toda la funcionalidad será entregada al cliente en un solo incremento. Generalmente su duración no es mayor a 6 meses y la funcionalidad incluida en los requerimientos no puede ser dividida en más de un incremento por lo que debe ser desarrollado en una sola iteración. Adicionalmente los requerimientos tienen la característica de ser estables desde el inicio del proyecto.

3.2 Proyectos de Largo Alcance.

Son los proyectos de desarrollo de software a la medida cuya característica particular es que la funcionalidad requerida será entregada al cliente en dos o más incremento. Generalmente su duración es mayor a 6 meses y la funcionalidad incluida en los requerimientos debe ser dividida en dos o más incrementos por lo que debe ser desarrollado en varias iteraciones.

4. Ciclos de Vida de Sistemas.

4.1 Introducción.

El Modelo de Ciclo de Vida utilizado para el desarrollo de software define el orden de las tareas o actividades involucradas, también definen la coordinación entre ellas, enlace y realimentación entre las mencionadas etapas. Entre los más conocidos se puede mencionar: El Modelo en Cascada o secuencial, El Modelo Espiral, o El Modelo Iterativo Incremental.

4.2 Proceso de Desarrollo de Software.

El proceso de desarrollo involucra numerosas y variadas tareas, desde lo administrativo hasta lo técnico. Sin embargo siempre se cumplen ciertas etapas mínimas; las que se pueden resumir como sigue:

- Análisis de requerimientos
- Diseño
- Implementación
- Pruebas
- Instalación
- Mantenimiento

4.3 Etapas en el Desarrollo de Software

4.3.1 Análisis de requerimientos

Esta es la primera fase que se realiza, y, según el modelo de proceso adoptado, puede casi terminar para pasar a la próxima etapa (caso de Modelo Cascada) o puede hacerse parcialmente para luego retomarla (caso Modelo Iterativo Incremental).

En simple palabras y básicamente, durante esta fase, se adquieren, reúnen y especifican las características funcionales y no funcionales que deberá cumplir el futuro sistema a desarrollar.

Las bondades de las características, tanto del sistema a desarrollar, como de su entorno, parámetros no funcionales y arquitectura dependen enormemente de lo bien lograda que esté esta etapa. Esta es, probablemente, la de mayor importancia y una de las fases más difíciles de lograr certeramente, pues no es automatizable, no es muy técnica y depende en gran medida de la habilidad y experiencia del analista que la realice.

Involucra fuertemente al usuario o cliente del sistema, por tanto tiene matices muy subjetivos y es difícil de modelar con certeza y/o aplicar una técnica que sea "la más cercana a la adecuada" (de hecho no existe "la estrictamente adecuada"). Si bien se han ideado varias metodologías, incluso software de apoyo, para levantamiento y registro de requerimientos, no existe una forma infalible o absolutamente confiable, y deben aplicarse conjuntamente buenos criterios y mucho sentido común por parte del o los analistas encargados de la tarea; es fundamental también lograr una fluida y adecuada comunicación y comprensión con el usuario final o cliente del sistema.

El artefacto más importante resultado de la culminación de esta etapa es la Especificación de Requerimientos.

4.3.2 Diseño

Con los principales requerimientos establecidos, hay un proceso continuo e iterativo de análisis, síntesis, evaluación, y optimización del diseño. Este proceso comienza con la definición en términos generales de una configuración del sistema durante la fase de diseño conceptual y continúa hasta que el sistema y sus componentes estén perfectamente definidos y listos para entrar en la fase de producción y/o construcción.

Dentro de los componentes del sistema podemos encontrar el modelo de clases, las interfaces, los modelos de bases de datos, sistemas operativos, componentes de reuso (propios o adquiridos), aplicativos, etc.

El artefacto más representativo de esta fase es el documento que contiene la arquitectura del sistema.

4.3.3 Implementación.

Durante esta la etapa se realizan las tareas que comúnmente se conocen como programación o desarrollo de código; que consiste, esencialmente, en llevar a código fuente, en el lenguaje de programación elegido, todo lo diseñado en la fase anterior. Esta tarea la realiza el programador, siguiendo por completo los lineamientos impuestos en el diseño y en consideración siempre a los requisitos funcionales y no funcionales especificados en la primera etapa.

En esta etapa, el programador es responsable de ejecutar las pruebas unitarias, que consisten en probar piezas de software pequeñas; a nivel de secciones, procedimientos, funciones y módulos; aquellas que tengan funcionalidades específicas. Dichas pruebas se utilizan para asegurar el correcto funcionamiento de secciones de código, mucho más reducidas que el conjunto, y que tienen funciones concretas con cierto grado de independencia.

4.3.4 Pruebas.

Las pruebas de función e integración se realizan una vez que las pruebas unitarias fueron concluidas *exitosamente*; con éstas se intenta asegurar que el sistema completo, incluso los subsistemas que componen las piezas individuales grandes del software funcionen correctamente al operar en conjunto.

Las pruebas normalmente se efectúan por un equipo de personas diferente al equipo de desarrolladores, y se utilizan datos de prueba, que es un conjunto seleccionado de datos típicos a los que puede verse sometido el sistema y/o módulos y/o bloques de código. También se escogen: Datos que llevan a condiciones límites al software a fin de probar su tolerancia y robustez; datos de utilidad para mediciones de rendimiento; datos que provocan condiciones eventuales o particulares poco comunes y a las que el software normalmente no estará sometido pero pueden ocurrir; etc. Los "datos de prueba" no necesariamente son ficticios o "creados", pero normalmente si lo son los de poca probabilidad de ocurrencia.

4.3.5 Instalación.

La instalación del software es el proceso mediante el cual los programas desarrollados son transferidos apropiadamente al ambiente de producción destino, inicializados, y, eventualmente, configurados; todo ello con el propósito de ser ya utilizados por el usuario final. Constituye la etapa final en el desarrollo del software. Luego de ésta el producto entrará en la fase de funcionamiento y producción, para el que fue diseñado.

4.3.6 Mantenimiento.

El mantenimiento de software es el proceso de control, mejora y optimización del software ya desarrollado e instalado, que también incluye depuración de errores y defectos que puedan haberse filtrado de la fase de pruebas. Esta fase es la última (antes de iterar, según el modelo empleado) que se aplica al ciclo de vida del desarrollo de software. La fase de mantenimiento es la que viene después de que el software está operativo y en producción.

4.4 Modelo de Cascada.

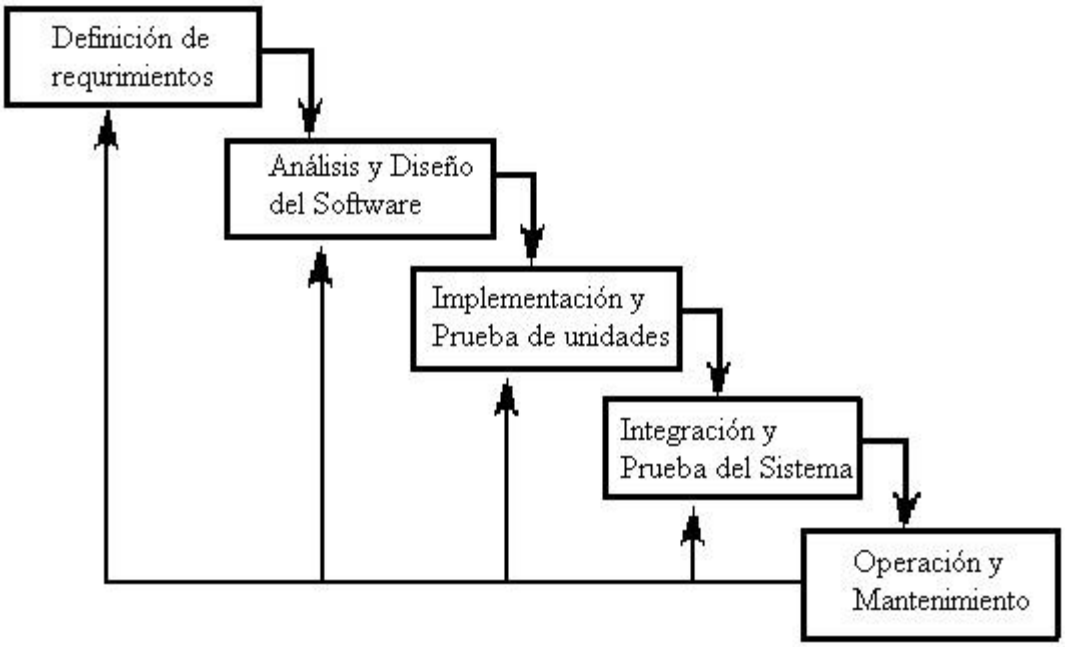
Este, aunque es más comúnmente conocido como Modelo en cascada es también llamado "Modelo Clásico", "Modelo Tradicional" o "Modelo Lineal Secuencial".

El Modelo en cascada implica un previo y absoluto conocimiento de los requisitos, la no volatilidad de los mismos y etapas subsiguientes libres de errores; ello sólo podría ser aplicable a escasos y pequeños desarrollos de sistemas. En estas circunstancias, el paso de una etapa a otra de las mencionadas sería sin retorno, por ejemplo pasar del Diseño a la Codificación implicaría un diseño exacto y sin errores ni probable modificación o evolución: "codifique lo diseñado que no habrán en absoluto variantes ni errores". Esto es utópico; ya que intrínsecamente el software es de carácter evolutivo, cambiante y difícilmente libre de errores, tanto durante su desarrollo como durante su vida operativa.

Algún cambio durante la ejecución de una cualquiera de las etapas en este modelo secuencial implicaría reiniciar desde el principio todo el ciclo completo, lo cual redundaría en altos costos de tiempo y desarrollo. La figura muestra un posible esquema del modelo en cuestión.

Sin embargo, el modelo de cascada en algunas de sus variantes es uno de los actualmente más utilizados, por su eficacia y simplicidad, más que nada en software de pequeño y algunos de mediano porte; pero nunca se usa en su forma pura, como se dijo anteriormente. En lugar de ello, siempre se produce alguna realimentación entre etapas, que no es

completamente predecible ni rígida; esto da oportunidad al desarrollo de productos software en los cuales hay ciertas dudas, cambios o evoluciones durante el ciclo de vida. Así por ejemplo, una vez capturados y especificados los requerimientos (primera etapa) se puede pasar al diseño del sistema, pero durante esta última fase lo más probable es que se deban realizar ajustes en los requerimientos (aunque sean mínimos), ya sea por fallas detectadas, ambigüedades o bien por qué los propios requerimientos han cambiado o evolucionado; con lo cual se debe retornar a la primera o previa etapa, hacer los pertinentes reajustes y luego continuar nuevamente con el diseño; esto último se conoce como retroalimentación. Lo normal en el modelo cascada será entonces la aplicación del mismo con sus etapas realimentadas de alguna forma, permitiendo retroceder de una a la anterior (e incluso poder saltar a varias anteriores) si es requerido.

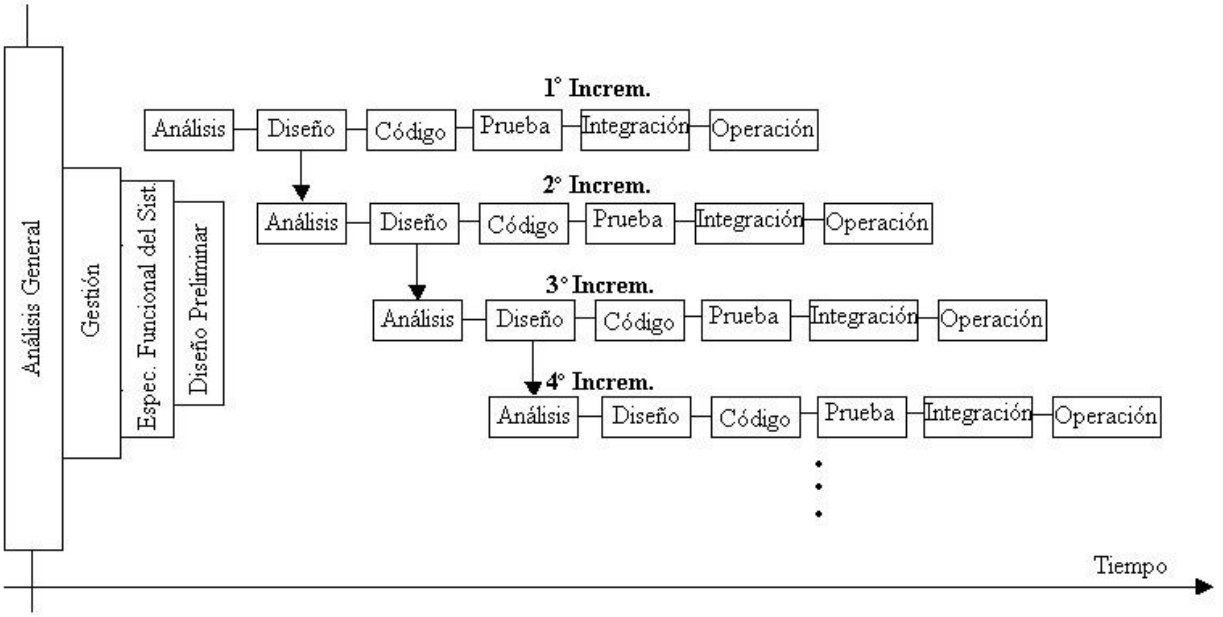


Ciclo de Vida de Cascada

Algunas desventajas del Modelo Cascada son:

- Los cambios introducidos durante el desarrollo pueden confundir al equipo en las etapas tempranas del proyecto. Si los cambios se producen en etapa madura (codificación o prueba) pueden ser catastróficos para un proyecto grande.
- No es frecuente que el cliente o usuario final explique clara y completamente los requisitos (etapa de inicio); y el modelo lineal lo requiere. La incertidumbre natural en los comienzos es luego difícil de acomodar.
- El cliente debe tener paciencia ya que el software no estará disponible hasta muy avanzado el proyecto. Un error detectado por el cliente (en fase de operación) puede ser desastroso, implicando reinicio del proyecto con altos costos.

4.5 Modelo Iterativo Incremental.



En términos generales, podemos distinguir en la figura, los pasos generales que sigue el proceso de desarrollo de un producto software. En el modelo de ciclo de vida seleccionado, se identifican claramente dichos pasos. La Descripción del Sistema es esencial para especificar y confeccionar los distintos incrementos hasta llegar al Producto global y final. Las actividades concurrentes (Especificación, Desarrollo y Validación) sintetizan el desarrollo pormenorizado de los incrementos, que se hará posteriormente.

El funcionamiento de un Ciclo Iterativo Incremental permite la entrega de versiones parciales a medida que se va construyendo el producto final. Cada versión emitida incorpora a los anteriores incrementos las funcionalidades y requisitos que fueron analizados como necesarios.

El Incremental es un modelo de tipo evolutivo que está basado en varios ciclos de Cascada retroalimentados aplicados repetidamente, con una filosofía iterativa.

En la figura se muestra un esquema del Modelo de ciclo de vida Iterativo Incremental, con sus actividades genéricas asociadas. Aquí se observa claramente cada ciclo cascada que es aplicado para la obtención de un incremento; estos últimos se van integrando para obtener el producto final completo. Cada incremento es un ciclo Cascada Realimentado, aunque, por simplicidad, en la figura se muestra como secuencial puro.

Se observa que existen actividades de desarrollo (para cada incremento) que son realizadas en paralelo o concurrentemente, así por ejemplo, en la figura, mientras se realiza el diseño detalle del primer incremento ya se está realizando en análisis del segundo. La figura es sólo esquemática, un incremento no necesariamente se iniciará durante la fase de diseño del anterior, puede ser posterior (incluso antes), en cualquier tiempo de la etapa previa. Cada incremento concluye con la actividad de “Operación y Mantenimiento” (indicada "Operación" en la figura), que es donde se produce la entrega del producto parcial al cliente. El momento de inicio de cada incremento es dependiente de varios factores: tipo de sistema; independencia o dependencia entre incrementos (dos de ellos totalmente independientes pueden ser fácilmente

iniciados al mismo tiempo si se dispone de personal suficiente); capacidad y cantidad de profesionales involucrados en el desarrollo; etc.

Bajo este modelo se entrega software “por partes funcionales más pequeñas”, pero reutilizables, llamadas incrementos. En general cada incremento se construye sobre aquel que ya fue entregado.

Como se muestra en la figura, se aplican secuencias Cascada en forma escalonada, mientras progresa el tiempo calendario. Cada secuencia lineal o Cascada produce un incremento y a menudo el primer incremento es un sistema básico, con muchas funciones suplementarias (conocidas o no) sin entregar.

El cliente utiliza inicialmente ese sistema básico intertanto, el resultado de su uso y evaluación puede aportar al plan para el desarrollo del/los siguientes incrementos (o versiones). Además también aportan a ese plan otros factores, como lo es la priorización (mayor o menor urgencia en la necesidad de cada incremento) y la dependencia entre incrementos (o independencia).

Luego de cada integración se entrega un producto con mayor funcionalidad que el previo. El proceso se repite hasta alcanzar el software final completo.

Siendo iterativo, *con el modelo Incremental se entrega un producto parcial pero completamente operacional en cada incremento*, y no una parte que sea usada para reajustar los requerimientos (como si ocurre en el Modelo de Construcción de Prototipos).

El enfoque Incremental resulta muy útil con baja dotación de personal para el desarrollo; también si no hay disponible fecha límite del proyecto por lo que se entregan versiones incompletas pero que proporcionan al usuario funcionalidad básica (y cada vez mayor). También es un modelo útil a los fines de evaluación.

5. Tipos de Proyecto vs. Ciclo de Vida.

La siguiente tabla muestra el criterio para seleccionar el ciclo de vida para cada tipo de proyecto.

Tipo de Proyecto	Ciclo de Vida
Proyecto de Corto Alcance	Modelo de Cascada
Proyecto de Largo Alcance	Modelo Iterativo Incremental