Rafael Osuna Ventura

# PRÁCTICA 1. Modelos del proceso del software

## MODELO LINEAL SECUENCIAL (CASCADA)

El Modelo Lineal Secuencial sugiere un enfoque sistemático o más bien secuencial del desarrollo de software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento. Descripción de las etapas:

- Análisis de los requerimientos del software: Es la fase en la cual se reúnen todos los requisitos que debe cumplir el software. En esta etapa es fundamental la presencia del cliente que documenta y repasa dichos requisitos.
- Diseño: Es una etapa dirigida hacia la estructura de datos, la arquitectura del software, las representaciones de la interfaz y el detalle procedimental (algoritmo). En forma general se hace un esbozo de lo solicitado y se documenta haciéndose parte del software.
- Generación del código: Es la etapa en la cual se traduce el diseño para que sea comprensible por la máquina. Esta etapa va a depender estrechamente de lo detallado del diseño.
- Pruebas: Esta etapa se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en la detección de errores.
- Mantenimiento: Debido a que el programa puede tener errores, puede no ser del completo agrado del cliente o puede necesitar, eventualmente acoplarse a los cambios en su entorno. Esto quiere decir que no se rehace el programa, sino que sobre la base de uno ya existente se realizan algunos cambios.

Ventajas:

- Se debe tener en cuenta que fue el primer modelo empleado, y por lo tanto es mejor que ninguno.
- Facilita la gestión del desarrollo.

Desventajas:

- En general, establecer todos los requisitos al principio del proceso de desarrollo es un mito inalcanzable, Los usuarios no pueden imaginarse lo que quieren hasta que no ven un sistema funcionando.
- Los requisitos no se pueden congelar mientras dura el desarrollo. El mercado cambia, todo cambia.
- El usuario debe esperar mucho tiempo hasta ver los resultados
- Los errores de análisis y diseño son costosos de eliminar, y se propagan a las fases siguientes con un efecto conocido como bola de nieve.

- Se genera mucho mantenimiento inicial debido al período de congelación de requisitos y éste recae, en su mayor parte

## Modelo de construcción de prototipos

Este modelo no secuencial, basado en la construcción de simulaciones o modelos ejecutables de aplicaciones más extensos. Descripción de las etapas:

- Comunicación: tener una interacción con el cliente para evaluar la petición del software y determinar si el programa a desarrollar es un buen candidato para construir un prototipo. Debido a que el cliente debe interaccionar con el prototipo para determinar el refinamiento del proyecto.
- Plan rápido: cuando se tienen que los resultados de un proyecto son aceptables, se procede a desarrollar una representación abreviada de los requerimientos. Antes de que pueda comenzar la construcción de un prototipo, en este se debe representar los dominios funcionales y de información del programa. La aplicación de estos principios de análisis fundamentales, pueden realizarse mediante los métodos de análisis de requerimientos.
- Modelado diseño rápido: Después de que se haya revisado la representación de los requerimientos, se crea un conjunto de especificaciones de diseño abreviadas para el prototipo. El diseño debe ocurrir antes de que comience la construcción del prototipo. Sin embargo, el diseño de un prototipo se enfoca normalmente hacia la arquitectura a nivel superior y a los aspectos de diseño de datos.
- Construcción de prototipo: El software del prototipo se crea, prueba y se corrigen Idealmente todos los posibles errores, los bloques de construcción de software preexisten se utilizan para crear el prototipo de una forma rápida y se determina si un prototipo es funcional o no. Para las aplicaciones interactivas con el hombre, es posible frecuentemente crear un prototipo en papel que describa la interacción hombre-máquina.
- Desarrollo y entrega: Una vez que el prototipo ha sido probado, se presenta al cliente, el cual "conduce la prueba" de la aplicación y sugiere modificaciones. Este paso es el núcleo del método de construcción de prototipo. Es aquí donde el cliente puede examinar una representación implementada de los requerimientos del programa, sugerir modificaciones que harán al programa cumplir mejor las necesidades reales.
- construcción de prototipos: este modelo básicamente se comienza elaborando un prototipo del producto final: qué aspecto tendrá, cómo funcionará además de los aspectos ya mencionados el prototipo debe ser construido en poco tiempo, usando los programas adecuados y no se debe utilizar mucho dinero pues a partir de que este sea aprobado nosotros podemos iniciar el verdadero desarrollo del software. Pero eso si al construir el prototipo nos asegura que

nuestro software sea de mejor calidad, además de que su interfaz sea de agrado para el usuario.

Ventajas:

- Permite la retroalimentación por parte del usuario.
- Desarrollo rápido.
- El usuario se siente parte del grupo
- También ofrece un mejor enfoque cuando el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humano-máquina.
- No modifica el flujo del ciclo de vida.
- Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios.
- Reduce costos y aumenta la probabilidad de éxito.
- Exige disponer de las herramientas adecuadas.
- No presenta calidad ni robustez.

Desventajas:

- El desarrollador debe dar forma prematuramente a un sistema, incluso antes de comprender de manera básica el problema y su funcionamiento.
- El usuario puede creer que un prototipo es un software final.
- Debe ser un sistema con el que se pueda experimentar
- Debe desarrollarse rápidamente
- Énfasis en la interfaz de usuario
- Equipo de desarrollo reducido
- Herramientas y lenguajes adecuadas

# MODELO DE DESARROLLO RAPIDO DE APLICACIONES(DRA)

Este es un modelo de proceso de desarrollo del software lineal, secuencias que enfatiza un ciclo de desarrollo extremadamente corto. Descripción de las etapas:

- Modelado de Gestión: El flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas: ¿Qué información conduce el proceso de gestión?, ¿Qué información se genera?, ¿Quién la genera?, ¿A dónde va la información?, ¿Quién la procesa?
- 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 (llamadas atributos) 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.
- Generación de Aplicaciones: El modelo asume la utilización de técnicas de cuarta generación. En lugar de crear software con lenguajes de programación de tercera generación, el proceso DRA trabaja para volver a utilizar componentes de programas ya existentes (cuando es posible) o a crear componentes reutilizables (cuando sea necesario). En todos los casos se utilizan herramientas automáticas para facilitar la construcción del software.
- Pruebas y Entrega: Como el proceso DRA enfatiza la reutilización, ya se han comprobado muchos de los componentes de los programas. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y se deben ejercitar todas las interfaces a fondo.

Ventajas:

- Es muy rápido.
- Permite trabajar en él a varias personas a la vez.
- Enfatiza ciclos de desarrollo extremadamente cortos.
- Se asegura de que el producto entregado cumple las necesidades del cliente.

Desventajas:

- El enfoque DRA tiene inconvenientes para proyectos grandes, aunque por escalas, necesita suficientes recursos humanos para crear el numero correcto de equipos.
- Si los desarrolladores y clientes no se comprenden con las actividades necesarias para completar el sistema, los proyectos fallarán.

- El DRA sería inapropiado cuando los riesgos técnicos son altos.
- No todos los tipos de aplicaciones son apropiados para DRA. Si un sistema no se puede modulizar adecuadamente. La construcción de los componentes necesarios para DRA será problemático. Si está en juego el alto rendimiento, y se va a conseguir el rendimiento convirtiendo interfaces en componentes de sistema, el enfoque DRA puede que no funcione.

## Modelos de métodos formales

El modelo de métodos formales acompaña a un conjunto de actividades que conducen a la especificación matemática del software de computadora. Los métodos formales permiten que un ingeniero del software especifique, desarrolle y verifique un sistema basado en computadora aplicando una notación rigurosa y matemática.

La ambigüedad, lo incompleto y la inconsistencia se descubren y se corrigen más fácilmente, no mediante una revisión a propósito para el caso, sino mediante la aplicación del análisis matemático. Cuando se utilizan métodos formales durante el diseño, sirven como base para la verificación de programas y por consiguiente permiten que el ingeniero del software descubra y corrija errores que no se pudieron detectar de otra manera.

Aunque todavía no hay un enfoque establecido, los modelos de métodos formales ofrecen la promesa de un software libre de defectos. Sin embargo, se ha hablado de una gran preocupación sobre su aplicabilidad en un entorno de gestión.

Ventajas e Inconvenientes

Si bien los métodos formales constituyen un acercamiento alternativo a las metodologías de desarrollo de software, existen un numero de diferencias significativas que deben de ser consideradas

Ofrece un fundamento para entornos de especificación que dan lugar a modelos de análisis más completos, consistentes y carentes de ambigüedad, que aquellos que se producen empleando métodos convencionales u orientados a objetos.

al pensar en instaurar los métodos formales en un equipo de desarrollo de software