TECNOLOGICO NACIONAL CEICOM

CARRERA DE SISTEMAS INFORMATICO

F

Practica02

METODOLOGIA Y PARADIGMA

Materia: Análisis y Diseño de Sistemas

Elaborado por: Fabrica Plaz Estevan

Docente: Ing. Baltazar Llusco Ever Jaime

Fecha: 18/06/2018

Cochabamba-Bolivia

METODOLOGIA Y PARADIGMA

Metodología.- Es el elemento que nos enuncia cuales son las herramientas que se van usar, quienes realizaran las tareas, en que momento y cuales son consideradas tareas críticas, entre otros. Existen innumerables mitologías y cada uno posee sus ventajas y desventajas, lo que hace necesario tomar la decisión de cual utilizar para cada desarrollo en particular.

Al referirse a **paradigma**, damos como opciones el desarrollo o el orientado a objetos. Estos no son otra cosa que la forma de desarrollo o ver el problema. No definen el ciclo de vida ni las tareas concretas a realizarse ni el momento en que se efectúan.

El proceso de software.- El proceso de software es un conjunto **definido**, **limitado** y **coherente** de actividades que permiten el desarrollo de software. El software posee ciertas características de proceso:

- Requerimientos
- Construcción
- Validación
- Evolución

Estas etapas el ciclo de vida de un proyecto y pueden ser ordenadas de acuerdo a un modelo. Un modelo es una representación simplificada de la realidad y, en este caso, es el marco de los pasos que hay que seguir en el desarrollo alcance buen término en el proceso de software.

- Define las tareas necesarias para correcto funcionamiento por etapa y en general.
- Describe y delimita las fases.
- Permite la ubicación en proyecto.
- Facilita la estimación de tiempo y costo.
- Ubica a los recursos en un momento dado y en una tarea determinada.

A continuación veremos la división de los modelos en tres tipos:

Lineales: representados en cascada, cada etapa está determinada y se sigue una secuencia definida esperando resultados anteriores.

Evolutivos: son de diversa índole, aunque se destacan el exploratorio y el desechable, haciendo en el resultado rápido y su confrontación con los deseos del cliente.

Componentes: desarrollo de introducción posterior en el mundo del software aunque muy conocido en otras industrias. Se apoya en el concepto de utilizar partes estandarizadas para lograr la construcción final.

Modelo en cascada.- Este modelo es desde hace más de 35 años y sigue siendo el pilar para tener noción sobre cuáles son las etapas que se requiere cumplir antes de tener un entregable de mínima funcionalidad.

Especificación.- Esta etapa es comúnmente denominada recolección de requisitos o especificación de requisitos, aunque en algunos entornos se la llama ingeniería de sistemas, en esta fase necesitamos definir el dominio de software que vamos a desarrollar, comprender sus fines, utilidad y limitaciones.

Análisis.- En esta etapa se determinan los objetivos y limites a partir de la información adquirida anteriormente, antes de pasar al diseño. Se debe determinar las tareas que realizara el sistema y sus estructuras, procesos críticos y secundarios. Por sus propiedades, las tareas de análisis son:

- Conceptualización: el objetivo es obtener una vista de alto nivel de abstracción.
- Análisis funcional: se trata de identificar y describir los procesos internos del sistema una vez establecido se realiza su refinamiento obteniendo los flujos de datos ingresantes y salientes.
- Análisis de condiciones: especificación de restricciones de nuestro sistema.
- Aplicar la metodología: luego de las etapas anteriores aplicaremos el proceso de diagramación. Esta creación para poder modelar el análisis realizado incorpora toda la información recopilada anteriormente.
- Validación: si nuestro análisis es correcto debemos pasar a la etapa siguiente.

Diseño.- En el diseño se aplica prácticas de ingeniería para de4finir de forma técnica los procesos, elementos y datos del sistema. El diseño profundiza y completa la visión del software, además le agrega los componentes técnicos, el diseño será la base para la construcción del software. Según las definiciones el diseño se materializa los requerimientos del cliente en general podemos identificar tres etapas:

- Diseño de estructuras: el diseño arquitectónico es el eje de la construcción.
- Diseño de estructura de datos: se definen y se documentan todos los datos que serán utilizados por el sistema. Se especifican todas las entradas, salidas y almacenajes.
- Diseño de interfaz: se decide desde un punto de vista técnica formal el tipo de interfaz que utilizaremos.

Codificación.- En esta etapa debemos destinar recursos para que el sistema que ya ha sido analizado y diseñado puede comenzar a ser una realidad. Las tareas de codificación son amplias y se realizan siguiendo cronogramas o patrones de trabajo establecidos.

Prueba.- La prueba del software es un conjunto de prácticas que se llevan a cabo para localizar, identificar y eliminar errores y mejorar el producto.

Mantenimiento.- luego de que el sistema ha sido probado e implementado y su funcionamiento es correcto, se debe realizar tareas que permitan prolongar su vida útil. Las operaciones del mantenimiento proponen lograr que el sistema mantenga el nivel de ejecución satisfactorio que obtuvo en las pruebas. Dentro del mantenimiento podemos observar:

- Monitoreo: es el conjunto de procesos que evalúan la calidad del control en el tiempo y permiten al sistema reaccionar en forma dinámica.
- Mantenimiento correctivo: es la preparación de errores en el momento en que estos aparecen.
- Mantenimiento preventivo: también denominado mantenimiento planificado consiste en la corrección de fallas de forma programada.

El modelo en cascada la gran ventaja de delimitar las tareas y ofrecer un macro de trabajo claro y simple que puede ser utilizado en proyectos de distinta tamaño, costos y recursos. También posee numerosas desventajas, entre las cuales se destacan el costo y el alta carga de esfuerzo para hacer la documentación. Entonces el modelo en cascada define un macro de trabajo que establece, que una etapa debe finalizar antes de comenzar el siguiente.

Modelos evolutivos.- Suponen la codificación de lo desarrollado con los deseos de los clientes. Obteniendo la respuesta del cliente más las mejoras que aportan el equipo de desarrollo.

Prototipos e incremental.- Es el que establece interacciones cortas de forma tal de mostrarle los avances al cliente.

El modelos incremental.- Presenta parecido que modelo de prototipo.se busca satisfacer al cliente desarrollando un conjunto de todos los requisitos que él solicita al equipo de trabajo.

Modelo en espiral.- Es un modelo mixto propuesto por Barry Boehm es un modelo con el cual se desarrollan versiones de software con mayor funcionalidad por iteración, y se definen cuatro actividades:

- Planificación: se entrevista al cliente y se obtiene la información importante para el resto de iteración.
- Análisis de riesgo: se evalúa en base a la composición del equipo de desarrollo, del proyecto y los nuevos requisitos, las posibles amenazas y debilidades del proyecto.
- Construcción: fase propiamente dicha de desarrollo, construcción de sistema en base al análisis.
- Evaluación: el cliente realiza un relevamiento crítico del entregable y en base a esto se inicia una nueva iteración.

Modelo de desarrollo de componentes.- Es un intento de mejora sobre el modelo de construcción en espiral. Se puede materializar en la práctica gracias a los paradigmas de análisis, diseño y programación orientados a objetos.

Con sus características esenciales, el lenguaje orientado a objetos permite generar componentes denominados clases. El proceso de creación de clases consta de tres etapas:

- Identificación del componente necesario.
- Búsqueda del componente en nuestro biblioteca de clases.
- Creación o uso de componentes.

Paradigma de desarrollo.- El desarrollo estructurado y el orientado a objetos son paradigmas o modelos de desarrollo.

Desarrollo orientado a objetos.- El software presenta características particulares que hacen que su complejidad alcance niveles importantes. A través de los años, los investigadores, ingenieros y desarrolladores han intentado minimizar esa dificultad incorporando distintas ideas. Entre los paradigmas de desarrollo actuales más destacados encontramos el **orientado a objetos y el orientado a estructura.**

El modelo de objetos.- el modelo de objetos no es solo una forma de programar si no que nos brinda los cimientos para poder desarrollar todo tipo de soluciones bajo sus principios. La principal

razón que hace a este modelo es la capacidad que tiene para combatir la complejidad inherente y disminuir los riesgos en el desarrollo.

El modelo orientado a objetos presenta algunos elementos fundamentales:

- Jerarquía
- Abstracción
- Modularidad
- Encapsulamiento

Si cualquiera de estos elementos está ausente, no podemos considerar al paradigma como modelo de objetos. Además existen otros elementos opcionales:

- Tipos (tipificación)
- Concurrencia
- Persistencia

Estos elementos brindan mayor robustez al modelo sin que su ausencia sea definitoria. A continuación definiremos a cada uno de ellos:

Jerarquía.- La jerarquía no es más que la posibilidad de realizar un ordenamiento en niveles de lo que deseamos representar, de manera práctica la jerarquía se ve presentada por la herencia, que puede ser de distintos tipos. Entre las más frecuentes la simple, la múltiple y la restrictiva.

Abstracción.- Es un proceso intelectual humana por el cual somos capaces de concentrarnos particularmente en las características que nos interesan para la solución de una situación.

Modularidad.- Su objetivo final de este es la división de un problema complejo en unidades más pequeñas y casi siempre más sencillas.

Encapsulamiento.- Es el ocultamiento de información de forma tal que solo esté disponible para interactuar con un objeto sin la necesidad de conocer cómo se comporta internamente.

Tipos (tipificación).- Permiten representar las abstracciones de forma adecuada.

Concurrencia.-Supone que en la solución de un problema se manejan toda clase de eventos y muchas veces algunos interactúan de manera asincrónica, pero también pueden hacerlo simultáneamente.

Persistencia.- la persistencia es lo que posibilita que trabajemos guardando información durante el tiempo que necesitamos operar con ella. La persistencia nos permite mantener el estado del objeto.

El objetivo como base.- Lógicamente, la orientación a objetos presenta muchas características comunes a otro tipo de prácticas. Lo que la distingue es que pone el énfasis en definir y caracterizar de forma clara los componentes del sistema, dotándolos de sus capacidades.

Programación orientado a objetos.- La programación orientado a objetos es un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos,

cada uno delos cuales representan una instancia de alguna clase, y cuyas clases son, miembros de una jerarquía de clases unidas mediante relaciones de herencia.