TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO de Pabellón de Arteaga TEC

INGENIERIA DE SOFTWARE

01/06/2019

APRENDIENDO UML EN 24 HORAS

PARTE 2

Docente:

Eduardo Flores Gallegos

Alumna:

Leslie Guadalupe Esparza Caldera

Carrera:

Ingeniería en Tecnologías de la Información y Comunicaciones

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo:

Código fuente y binario . Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes de Ada, bibliotecas cargadas dinámicamente, etc. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente. Un diagrama de componentes representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. Un módulo de software se puede representar como componente. Algunos componentes existen en tiempo de compilación, algunos en tiempo de enlace y algunos en tiempo de ejecución, otros en varias de éstas. Un componente de sólo compilación es aquel que es significativo únicamente en tiempo de compilación. Un componente ejecutable es un programa ejecutable.

Un diagrama de componentes tiene sólo una versión con descriptores, no tiene versión con instancias. Para mostrar las instancias de los componentes se debe usar un diagrama de despliegue. Un diagrama de componentes muestra clasificadores de componentes, las clases definidas en ellos, y las relaciones entre ellas. Los clasificadores de componentes también se pueden anidar dentro de otros clasificadores de componentes para mostrar relaciones de definición.

Un diagrama que contiene clasificadores de componentes y de nodo se puede utilizar para mostrar las dependencias del compilador, que se representa como flechas con líneas discontinuas (dependencias) de un componente cliente a un componente proveedor del que depende. Los tipos de dependencias son específicos del lenguaje y se pueden representar como estereotipos de las dependencias. El diagrama también puede usarse para mostrar interfaces y las dependencias de llamada entre componentes, usando flechas con líneas discontinuas desde los componentes a las interfaces de otros componentes. El diagrama de componente hace parte de la vista física de un sistema, la cual modela la estructura de implementación de la aplicación por sí misma, su organización en componentes y su

despliegue en nodos de ejecución. Esta vista proporciona la oportunidad de establecer correspondencias entre las clases y los componentes de implementación y nodos. La vista de implementación se representa con los diagramas de componentes. Debido a que estos son más parecidos a los diagramas de casos de usos estos son utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. Se utilizan para modelar la vista estática de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. Su panorama del UML le muestra las categorías de los diagrama y a estos en cada categoría. El UML cuenta con una arquitectura de cuatro capas: capa de objetos del usuario, capa de modelado, capa metamodelado y capa metametamodelado. **Extensiones del UML**

Estereotipos: El propósito de un estereotipo es extender a un elemento del UML para que sea una instancia de una nueva metaclase, y se escribe entre dos pares de paréntesis angulares. Esto agrega una gran flexibilidad. **Dependencia:** puede tomar la cantidad de estereotipos ya creados, cada uno extiende una relación de dependencia entre un origen y un destino (extender, usar, importar, instancia, enviar). **Clasificador:** los estereotipos extienden a los clasificadores de diversas formas (tipo de autoridad, proceso, subproceso, utilería, estereotipos). **Clase:** muestra algo más específico que un clasificador. **Generalización:** es una relación entre clasificadores, con su propio pequeño conjunto de estereotipos (heredar, subclase, privado). **Paquete:** los estereotipos de los paquetes son directos (fachada, sistema, cabo). Componente: los estereotipos para los componentes son aún más directos. Puede mostrar que un componente es un documento, ejecutable, un archivo, una tabla de datos(tabla, biblioteca, documento, archivo, ejecutable). **Restricciones** Las restricciones se encuentran entre llaves. Proporcionan las condiciones para las asociaciones, extremos de vínculos, generalizaciones y peticiones. **Valores etiquetados** Un valor etiquetado se escribe entre llaves. Consiste en una etiqueta, un signo = y un valor. Existen dos tipos de metodologías: antiguas y recientes. Se entiende por

metodología a la estructura y naturaleza de los pasos en un esfuerzo de desarrollo. Pero antes de iniciar a programar los desarrolladores deben tener claridad sobre el problema.

Método antiguo

Las etapas deben suceder en lapsos definidos, una después de otra. Obsérvese el método en cascada: Este método reduce el impacto de la comprensión obtenida en el proyecto. Si el proceso no puede retroceder y volver a ver los primeros estados, es posible que las ideas desarrolladas no sean utilizadas. **Método reciente**

Tiende a la colaboración entre las fases de desarrollo esta moderna ingeniería de programas, los analistas y diseñadores hacen revisiones para desarrollar un sólido fundamento para los desarrolladores. Existe interacción entre todo el equipo de trabajo.

La ventaja es que conforme crece la comprensión, el equipo incorpora nuevas ideas y genera un sistema más confiable. Gracias a todo lo anterior nos podemos dar cuenta que es mas fácil identificar el problema y la solución al problema con diferentes métodos que te acercan mas lo que se necesita para resolverlo por medio de diagramas que cada vez te van dejando mas claras las ideas de lo que vas a realizar.

Para después poder tener en cuenta lo que en verdad necesita (necesidades); para un mejor conocimiento, con las necesidades ya ubicadas ahora tenemos que hacerlas para que dejen de ser necesidades y convertirse en realidades. Con los casos de uso nos damos cuenta que es lo que va a realizar cada uno y si es que esta completo todo lo que el cliente pidió es la manera más práctica de darte cuenta.