República Bolivariana de Venezuela
Ministerio del Poder Popular Para la Educación Universitaria, Ciencia y Tecnología
Universidad Politécnica Territorial del Estado Mérida "Kléber Ramírez"
Ejido. Estado. Mérida

Unidad 3

Proceso de Desarrollo del Software

Elaborado por:

Jesus Gonzalez

Javier Peña

Lisbeth Angulo

PNF Informática T2t1 Sección: A.

Noviembre, 2016

Contenido

Proceso de Desarrollo de Software:

El Proceso para el desarrollo de software, también denominado ciclo de vida del desarrollo de software es una estructura aplicada al desarrollo de un producto de software. Hay varios modelos a seguir para el establecimiento de un proceso para el desarrollo de software, cada uno de los cuales describe un enfoque diferente para diferentes actividades que tienen lugar durante el proceso. Algunos autores consideran un modelo de ciclo de vida un término más general que un determinado proceso para el desarrollo de software. Por ejemplo, hay varios procesos de desarrollo de software específicos que se ajustan a un modelo de ciclo de vida de espiral.

Fundamentos del enfoque orientado a objetos

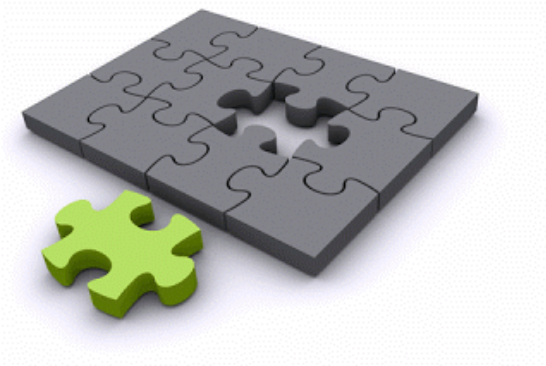
La orientación a objetos ofrece una solución que ayuda a los desarrolladores a hacer corresponder el mundo real tan cerca como sea posible al dominio de la solución. Cabe mencionar que existen muchas metodologías que permiten soluciones para problemas complejos. En la orientada a objetos se basa en modelar el mundo real y ha ganado importancia significativa en los últimos tiempos. En la orientación a objetos se trabaja con objetos en el sistema que interactúan unos con otros a través de mensajes. La orientación a objetos proporciona los recursos para ocuparse de los objetos de un sistema complejo. El análisis y diseño de un sistema desde una perspectiva orientada a objetos forma el núcleo de un sistema.

Características

- Modelado del mundo real
- Datos Abstractos
- Abstracción de datos
- Encapsulamiento
- Ocultamiento de la información
- Clase
- Objeto
- Interfaz e Implementación
- Métodos
- Mensajes
- Herencia
- Agregación
- Polimorfismo
- Tipo
- Rol
- Paquete

Componentes

Es una unidad autocontenida que encapsula el estado y el comportamiento de varios clasificadores. También se podría decir que es un tipo clasificador con la diferencia de que no tiene características propias, pero contiene las clases que definen las características. Un componente proporciona una vista encapsulada de la funcionalidad definida por las clases contenidas. Un componente es una parte física del sistema. Cada componente tiene un nombre, el cual puede ser un nombre simple o un nombre de ruta.



Tipos de componentes y características

- Componentes de despliegue o distribución (Deployment): Estos componentes se usan para formar un sistema ejecutable.

- Componentes de Producto de Trabajo: Estos componentes son parte del proceso de desarrollo que es esencial para el sistema. Algunos ejemplos de componentes de producto de trabajo son los archivos fuente, archivos de datos y tablas.

- Componentes de Ejecución: Estos componentes son el resultado de un sistema que se está ejecutando.

Estándares en el proceso de desarrollo de software

ISO Es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica. Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional. Estándares ISO existentes: ISO 9001, 9000–3, 9004–2, ISO/IEC 12207, ISO/IEC 15504 (SPICE) Algunos estándares existentes:

- ✓ Estándares para datos
- ✓ Estándares de codificación
- ✓ Estándares estructurales
- ✓ Estándares de documentación
- ✓ Estándares de proceso software
- ✓ Estándares para otras actividades

Ejemplos de estándares en ingeniería del software:

- IEEE Standards Collection Software Engineering – 1998 Edition
- IEEE Std. 610.12-1990, Glossary of Software Engineering Terminology
- IEEE Std. 829-1983, Standard for Software Test Documentation

- IEEE Std. 830-1993, Recommended Practice for Software Requirements Specifications.
- IEEE Std. 990-1987, Recommended Practice for Ada as a Program Design Language.
- IEEE Std. 1045-1992, Standard for Software Productivity Metrics
- IEEE Std. 1062-1987, Recommended Practice for Software Acquisition
- IEEE Std. 1063- 1987, Standard for Software User Documentation
- IEEE Std. 1219-1992, Standard for Software Maintenance

Documentación y Artefactos

La documentación no es más que la debilidad más frecuente en productos e instalaciones informáticos. Cabe mencionar que los actores que intervienen en el ciclo de vida del software desempeñan diversos roles. Arquitectos, diseñadores, analistas, programadores, implementadores, administradores o auditores son quienes explicitan distintos aspectos de los productos y procesos.

Un artefacto es una pieza de información que es producida o utilizada por procesos. Los artefactos son los elementos son los elementos tangibles de un proyecto, elementos que el proyecto produce o usa mientras se trabaja en busca del producto final. Éstos, pueden tomar varias formas y formatos, como por ejemplo:

- ✓ Un documento, tal como la visión o la lista de riesgos.
- ✓ Un modelo, por ejemplo un diagrama de casos de uso o el modelo de diseño.

- ✓ Un elemento dentro de un modelo, tal como una clase, un caso de uso o un subsistema.
- ✓ Ejecutables, por ejemplo el ejecutable del prototipo.
- ✓ Código fuente.

Las actividades tienen artefactos como entrada y salida. Los roles usan artefactos para ejecutar actividades y producen artefactos durante la ejecución de sus actividades. Los artefactos son la responsabilidad sencilla del rol, creando responsabilidades fáciles de identificar y entender, promoviendo la idea de que cada pieza de información producida en un proceso de desarrollo requiere un conjunto apropiado de habilidades. Aunque un rol puede ser el propietario de un artefacto, otros roles pueden hacer uso de éste, incluso podrían actualizar el artefacto si el rol que va a hacerlo, tiene permiso para hacerlo.

Metodologías para Desarrollo de Software

Un proceso de software detallado y completo suele denominarse “Metodología”. Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, etc.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc. Habitualmente se utiliza el término “método” para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo, por ejemplo, suele hablarse de métodos de análisis y/o diseño.

La comparación y/o clasificación de metodologías no es una tarea sencilla debido a la diversidad de propuestas y diferencias en el grado de detalle, información disponible y alcance de cada una de ellas. A grandes rasgos, si tomamos como criterio las notaciones utilizadas para especificar artefactos producidos en actividades de análisis y diseño, podemos clasificar las metodologías en dos grupos: Metodologías Estructuradas y Metodologías Orientadas a Objetos. Por otra parte, considerando su filosofía de desarrollo, aquellas metodologías con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías Tradicionales (o peyorativamente denominada Metodologías Pesadas, o Peso Pesado). Otras metodologías, denominadas Metodologías Ágiles, están más orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso.

Proceso Unificado de Desarrollo (UP del inglés Unified Process). Disciplinas.

Es una metodología de desarrollo de software que está basado en componentes e interfaces bien definidas, y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Es un proceso que puede especializarse para una gran variedad de sistemas de software, en diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Introducción a los procesos ágiles de desarrollo: Fundamentos de los procesos ágiles de desarrollo

Existen numerosas propuestas de metodología para desarrollar software. Tradicionalmente estas metodologías se centran en el control del proceso, estableciendo rigurosamente las actividades, herramientas y notaciones al respecto, dado estas reglas estas metodologías se caracterizan por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Algunos ejemplos de metodología ágil son

- ✓ Programación Extrema, es uno de los ejemplos más exitosos de metodología ágil.
- ✓ Scrum
- ✓ Crystal
- ✓ Evolutionary Project Management (Evo)
- ✓ Feature Driven Development (FDD)

- ✓ Adaptive Software Developmen(ASD)Lean Development (LD) y Lean Software Development (LSD)

Procesos Ágiles de Desarrollo

El desarrollo ágil de software son métodos de ingeniería del software basado en el desarrollo iterativo e incremental, donde los requerimientos y soluciones evolucionan mediante la colaboración de grupos auto organizado y multidisciplinarios. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en cortos lapsos de tiempo. El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. Cada iteración del ciclo de vida incluye: planificación, análisis de requerimientos, diseño, codificación, revisión y documentación. Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, pero la meta es tener una «demo» (sin errores) al final de cada iteración. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto

Introducción al Modelado

Características de los Lenguajes de Modelado

La herramienta UML debe apoyar todos los diagramas de los nueve que componen UML. La herramienta debe soportar la diagramación de casos de uso, permitir definir la visión estática con diagramas de clases y diagramas de objeto, permitir la definición de la visión dinámica, tales como los diagramas de secuencia, la actividad, de los estados, de colaboración y el despliegue de componentes que forman el sistema.

Lo fundamental de una herramienta UML es la capacidad de diagramación, y los diferentes tipos de diagramas que soporta la herramienta. Sus esquemas de apoyo de diseño, documentación, construcción e implantación de sistema. Así mismo, su flexibilidad para admitir cambios no previstos durante el diseño o el rediseño. En resumen, la herramienta ideal, es aquella que admite diseño desde inicio a fin, diseño inverso (o

rediseño) y diseño vise-versa, con esquemas amplios para documentar detalladamente los procesos.

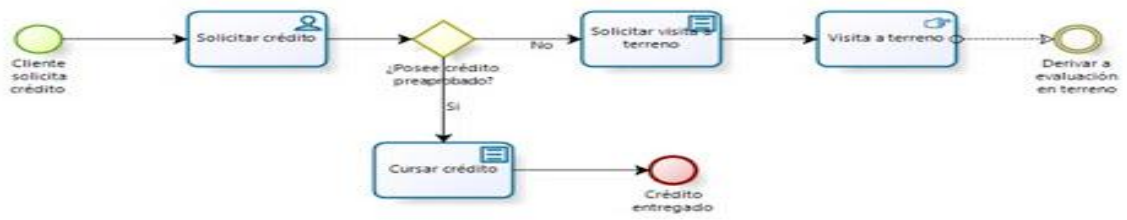
Diagramas, Símbolos y Notación

Un diagrama es una representación gráfica de una colección de elementos de modelado, a menudo dibujada como un grafo conexo de arcos (relaciones) y vértices (otros elementos del modelo). Un diagrama no es un elemento semántico, un diagrama muestra representaciones de elementos semánticos del modelo, pero su significado no se ve afectado por la forma en que son representados. Un diagrama está contenido dentro de un paquete.

La mayoría de los diagramas de UML y algunos símbolos complejos son grafos que contienen formas conectadas por rutas. La información está sobre todo en la topología, no en el tamaño o la colocación de los símbolos (hay algunas excepciones como el diagrama de secuencia con un eje métrico de tiempo). Hay tres clases importantes de relaciones visuales: conexión (generalmente de líneas a formas de dos dimensiones), contención (de símbolos por formas cerradas de dos dimensiones), y adhesión visual (un símbolo que está "cerca" de otro en un diagrama). Estas relaciones geométricas se reasignan a conexiones entre nodos en un gráfico en la forma analizada de la notación.

La notación de UML está pensada para ser dibujada en superficies bidimensionales. Algunas formas bidimensionales son proyecciones de formas tridimensionales tales como cubos, pero todavía se representan como íconos en una superficie bidimensional.

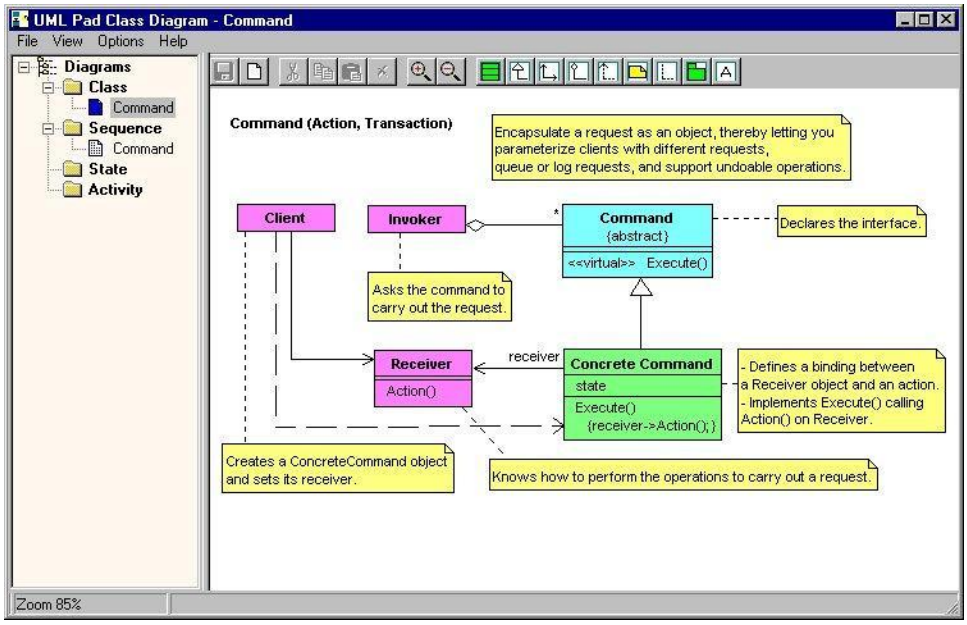
Hay cuatro clases de construcciones gráficas que se usan en la notación de UML : íconos, símbolos bidimensionales, rutas y cadenas.



Herramientas CASE (Computer Aided Software Engineering)

Ingeniería de Software Asistida por Computadoras. Son diversas Aplicaciones informáticas destinadas a aumentar la productividad en el Desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el diseño de proyectos, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, Compilación automática, documentación o detección de errores entre otras.

Es un sistema de software que intenta proporcionar ayuda automatizada a las actividades del proceso de desarrollo de software. Los sistemas CASE a menudo se utilizan como apoyo al método. La primera herramienta CASE como hoy la conocemos fue Excelerator en 1984, era para PC. Actualmente la oferta de herramientas CASE es muy amplia y tenemos por ejemplo el EASYCASE o WINPROJECT.



Tecnología de las herramientas CASE

La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información a la hora de construir software se plantean los siguientes objetivos:

- ✓ Permitir la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta conseguimos agilizar el trabajo.
- ✓ Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- ✓ Simplificar el mantenimiento de los programas.
- ✓ Mejorar y estandarizar la documentación.
- ✓ Aumentar la portabilidad de las aplicaciones.
- ✓ Facilitar la reutilización de componentes software.
- ✓ Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos.

Estructura general de una herramienta CASE

La estructura CASE se basa en la siguiente terminología:

• CASE de alto nivel: son aquellas herramientas que automatizan o apoyan las fases finales o superiores del ciclo de vida del desarrollo de sistemas como la planificación de sistemas, el análisis de sistemas y el diseño de sistemas.

• CASE de bajo nivel: son aquellas herramientas que automatizan o apoyan las fases finales o inferiores del ciclo de vida como el diseño detallado de sistemas, la implantación de sistemas y el soporte de sistemas.

• CASE cruzado de ciclo de vida: se aplica a aquellas herramientas que apoyan actividades que tienen lugar a lo largo de todo el ciclo de vida, se incluyen actividades como la gestión de proyectos y la estimación.

Herramientas Case más utilizadas

- ✓ ERwin es una herramienta de diseño de base de datos. Brinda productividad en diseño, generación, y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la base de datos diseñada, ERwin permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la base de datos. Genera automáticamente las tablas y miles de líneas de stored procedure y triggers para los principales tipos de base de datos.

- ✓ EasyCASE Profesional, el centro de productos para procesos, modelamiento de datos y eventos, e Ingeniería de Base de Datos, es un producto para la generación de esquemas de base de datos e ingeniería reversa, trabaja para proveer una solución comprensible para el diseño, consistencia y documentación del sistema en conjunto.