Tecnológico Nacional De México, Campus Región Sierra

Carrera:

Informática

Actividad:

Resumen

Maestro:

M.I.A. Alejandra Guadalupe Vázquez Martínez

Alumnos:

Carlos Ignacio Lorca Lopez

5to semestre

Grupo A

22 de agosto del 2024

El modelo del proceso del software

1.1 Conceptualización de tecnología orientada a objetos.

El modelo de proceso de software como la programación estructurada consta de múltiples datos y funciones globales. Todos los datos o funciones dentro de un programa son visibles y en actualidad la POO tiene un roll importante dentro de la programación, puesto que es una herramienta muy necesaria que ayuda a elaborar un programa u sistema determinando, teniendo en cuenta las diferentes características que este debe de llevar, no dejando afuera las especificaciones que deben englobar. La Tecnología Orientada a Objetos se apoya en fundamentos de la ingeniería, cuyos elementos reciben el nombre global de modelo de objetos, el cual abarca los principios de abstracción, encapsulación, modularidad, jerarquía, concurrencia y persistencia.

En la programación tradicional, también conocida como programación estructurada, un programa o aplicación consta de múltiples datos y funciones “globales”. El término “global” describe el hecho que todos los datos o funciones son “visibles” en todo el programa y, por lo tanto, pueden ser llamados desde cualquier ubicación en la aplicación. Esta forma de programación tiene sus orígenes en las primeras computadoras modernas basadas en la arquitectura Von Neuman de 1945 donde las instrucciones de un programa se guardaban en memoria creando el concepto de programa almacenado.

A continuación, hay algunas metodologías que son las más utilizadas:

- **Modelado Basado en Objetos (Object Oriented Modeling, OOM):**

Objetivo: Representar los sistemas de software en términos de objetos y sus interacciones.

Características: Enfatiza la encapsulación, herencia y polimorfismo.

Ejemplo de Uso: Usado para crear modelos que describen el comportamiento y la estructura de sistemas complejos.

- **Unified Modeling Language (UML):**

Objetivo: Proporcionar un lenguaje estándar para especificar, visualizar, construir y documentar los artefactos de software.

Características: Incluye diagramas de clases, secuencia, casos de uso, entre otros.

Ejemplo de Uso: Es ampliamente usado para diseñar y documentar sistemas orientados a objetos.

- **Rational Unified Process (RUP):**

Objetivo: Ofrecer un marco de proceso para el desarrollo de software orientado a objetos.

Características: Se basa en iteraciones, y define roles, artefactos y procesos que guían el desarrollo.

Ejemplo de Uso: Facilita una estructura detallada para gestionar proyectos de software.

Extreme Programming (XP):

Objetivo: Mejorar la calidad del software y la capacidad de respuesta al cambio.

Características: Incorpora prácticas como desarrollo iterativo, programación en pareja y pruebas continuas.

Ejemplo de Uso: Beneficioso en proyectos donde los requisitos pueden cambiar frecuentemente.

1.2. Metodologías emergentes de desarrollo de software.

Un proceso define quién hace qué, cuándo y cómo para alcanzar cierto objetivo. El éxito de las organizaciones radica en gran medida en la correcta definición y empleo de sus procesos. Los sistemas de software pueden llegar a ser muy complejos, para administrar dicha complejidad es necesario contar con modelos de procesos y tecnologías de software apropiadas. Modelo de proceso. Define cómo solucionar la problemática del desarrollo de sistemas de software. Para desarrollar software se requiere resolver ciertas fases de su proceso, las cuales se conocen como el ciclo de vida de desarrollo de software. Un modelo de proceso debe considerar aspectos como el conjunto de personas, estructuras organizacionales, reglas, políticas, actividades, componentes de software, metodologías y herramientas.

Aquí se encuentran en algunas metodologías y herramientas:

- **Desarrollo Ágil (Agile Development):** Se centra en la flexibilidad y la entrega continua de valor mediante ciclos cortos y colaboración constante con el cliente. Ejemplos incluyen Scrum y Kanban.

- **DevOps:** Integra desarrollo y operaciones para mejorar la automatización y colaboración en el ciclo de vida del software, utilizando herramientas como Jenkins y Docker.

- **Model-Driven Development (MDD):** Utiliza modelos abstractos para guiar el desarrollo de software y generar automáticamente código y otros artefactos. Herramientas como Eclipse Modeling Framework (EMF) se emplean en este enfoque.

- **Desarrollo Basado en Servicios (Service-Oriented Development):** Construye aplicaciones como un conjunto de servicios independientes que se comunican a través de interfaces estandarizadas, incluyendo arquitecturas como SOA y microservicios.

- **Design Thinking:** Se enfoca en entender profundamente a los usuarios para resolver problemas complejos mediante la empatía, definición de problemas, ideación y prototipado.

- **Domain-Driven Design (DDD):** Basado en crear un modelo conceptual del dominio del problema, facilitando un diseño que refleje con precisión la realidad del negocio, con un lenguaje común y contextos bien definidos.

Herramientas.

Desarrollo Ágil:

JIRA: Gestión de proyectos y seguimiento de tareas.

Trello: Organización de tareas con tableros Kanban.

DevOps:

Jenkins: Automatización de integración y entrega continua (CI/CD).

Docker: Gestión de contenedores.

Kubernetes: Orquestación de contenedores.

Model-Driven Development (MDD):

Eclipse Modeling Framework (EMF): Creación y gestión de modelos.

MagicDraw: Modelado UML y generación de código.

Desarrollo Basado en Servicios (Service-Oriented Development):

Apache ServiceMix: Integración de servicios SOA.

Spring Boot: Desarrollo de microservicios.

Design Thinking:

Miro: Colaboración visual y prototipos.

InVision: Diseño de prototipos interactivos.

1.3. Métodos de desarrollo de software orientado a objetos.

En el desarrollo de software, una metodología hace cierto énfasis al entorno en el cual se plantea y estructura el desarrollo de un sistema. Como lo mencioné al principio, existen una gran cantidad de metodologías de la programación que se han utilizado desde los tiempos atrás y que con el paso del tiempo han ido evolucionando. Por esta razón, es importante que dependiendo del tipo de software que se vaya a desarrollar, se identifique la metodología para el diseño de software idónea cada metodología de desarrollo de software tiene más o menos su propio enfoque para el desarrollo de software. Estos son los enfoques más generales, que se desarrollan en varias metodologías específicas.

Su metodología de desarrollo de software consiste principalmente en hacer uso de diversas herramientas, técnicas, métodos y modelos para el desarrollo. Regularmente este tipo de metodología, tienen la necesidad de venir documentadas, para que los programadores que estarán dentro de la planeación del proyecto comprendan perfectamente la metodología y en algunos casos el ciclo de vida del software que se pretende seguir.

Como por ejemplo el modelo espiral que consiste en un modelo de proceso de software evolutivo, el cual enlaza la naturaleza iterativa de la construcción de prototipos, pero conservado aquellas propiedades del modelo en cascada.

1.4. El proceso de desarrollo unificado – RUP.

El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software 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.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización es un marco de trabajo para el desarrollo de software que proporciona una metodología estructurada y flexible. Fue desarrollado por Rational Software (ahora parte de IBM) y se basa en una serie de prácticas y principios diseñados para mejorar la calidad y eficiencia del desarrollo de software.

También se conoce por este nombre al software desarrollado por Rational, hoy propiedad de IBM, el cual incluye información entrelazada de diversos artefactos y descripciones de las diversas actividades. Está incluido en el Rational Method Composer (RMC), que permite la personalización de acuerdo con las necesidades y originalmente se diseñó un proceso genérico y de dominio público, el Proceso Unificado, y una especificación más detallada, el Rational Unified Process, que se vendiera como producto independiente.

El RUP está basado en 5 principios clave que son:

Adaptar el proceso

El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño de este, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.

Equilibrar prioridades

Los requerimientos de los diversos participantes pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un equilibrio que satisfaga los deseos de todos. Gracias a este equilibrio se podrán corregir desacuerdos que surjan en el futuro.

Demostrar valor interactivamente

Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto, así como también los riesgos involucrados

Colaboración entre equipos

El desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requerimientos, desarrollo, evaluaciones, planes, resultados.

Elevar el nivel de abstracción

Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o marcos de referencia (frameworks) por nombrar algunos.

1.5. El lenguaje de modelado unificado – UML

El Lenguaje de Modelado Unificado (UML) es un lenguaje estándar para especificar, visualizar, construir y documentar los artefactos de un sistema de software. Es referido por algunos como la lengua franca entre los lenguajes de modelados utiliza el diseño y análisis de sistemas para representar gráficos y diagramas que describen la estructura y el comportamiento del sistema.

Hay cuatro categorías de modelos para la resolución de problemas: lenguajes imperativos, funcionales, declarativos y orientados a objetos (OOP). En los lenguajes orientados a objetos, los algoritmos se expresan definiendo 'objetos' y haciendo que los objetos interactúen entre sí. Esos objetos son cosas que deben ser manipuladas y existen en el mundo real. Pueden ser edificios, artefactos sobre un escritorio o seres humanos.

Los lenguajes orientados a objetos dominan el mundo de la programación porque modelan los objetos del mundo real. UML es una combinación de varias notaciones orientadas a objetos: diseño orientado a objetos, técnica de modelado de objetos e ingeniería de software orientada a objetos.

Características y métodos clave de UML:

Estándar: UML es un lenguaje estandarizado por la Object Management Group (OMG) y es ampliamente utilizado en la industria del software.

Diagramas: UML ofrece varios tipos de diagramas, cada uno con un propósito específico.

Diagramas Estructurales: Representan la estructura estática del sistema.

Diagrama de Clases: Muestra las clases del sistema y sus relaciones.

Diagrama de Componentes: Representa la organización y la dependencia de los componentes del sistema.

Diagrama de Paquetes: Organiza los elementos del sistema en paquetes.