Aprendiz: Rafael Dario Escalante Sandoval

Instructor: Andrés Rubiano Cucarian

Centro De Servicios Financieros

Bases conceptuales acerca del lenguaje unificado de modelado (UML) y patrones de diseño
GA4-220501095-AA2-EV03.

ADSO (2977466)

Contents

Introducción

Para desarrollar el proyecto de software de cafetería se debe emplear herramientas que aclaren el cómo están dadas las características que componen el software a realizar, por lo tanto, permite representar gráficamente distintos aspectos de un sistema, facilitando el diseño, la especificación y la documentación de sistemas complejos, particularmente en el ámbito de la programación orientada a objetos.

Que es UML

UML (lenguaje unificado de modelado) sirve para estructurar, modelar y resumir de manera grafica las características del software a diseñar, contando con variedades o tipos de diagramas, así como diferentes programas para crearlos, Simplificar algo tan complejo como un aluvión de código es extremadamente útil para ingenieros y partes interesadas no técnicas. Les permite mantenerse al tanto de los proyectos en los que trabajan y evitar perderse en las interminables complejidades inherentes a la programación de software. Los diagramas UML también dividen los componentes y subcomponentes esenciales para construir un programa informático, sirve también para los principiantes entender y familiarizarse con el proyecto a construir.

Glosario de términos UML

- Compatibilidad con sintaxis abstracta

Los usuarios pueden mover modelos a través de diferentes herramientas, incluso si usan diferentes notaciones.

- Metamodelo de almacén común (CWM)

Interfaces estándares que se usan para permitir el intercambio de metadatos de almacén e inteligencia de negocios entre herramientas de almacén, plataformas de almacén y repositorios de metadatos de almacén en entornos heterogéneos distribuidos.

- **control de acceso**

la **visibilidad** define la accesibilidad para los atributos o métodos de la clase.

- Compatibilidad con sintaxis concreta

Los usuarios pueden continuar usando una notación con la que estén familiarizados a través de diferentes herramientas.

- Navegabilidad

en las relaciones de asociación se puede establecer si el vínculo es unidireccional o bidireccional. Gráficamente se representan con puntas de flecha.

- Herencia

En una relación de herencia las subclases heredan las características (atributos) y los comportamientos (métodos) de las superclases.

- Agregación

La agregación es un tipo de relación especial, donde se representa que un conjunto de clases conforma un todo, es decir, existen clases agregadas que representan el todo y se constituyen a partir de un conjunto de clases componentes

- Multiplicidad

representan el número de instancias de la clase que pueden ser partícipes en el proceso de asociación con respecto a una instancia particular de la otra clase vinculada en la relación. (Schmuller, 2001)

- Núcleo

En el contexto de UML, el núcleo comúnmente se refiere al "paquete central", que es un metamodelo completo particularmente diseñado para una alta reutilización.

- Unidad de lenguaje

Consiste en una colección de conceptos de modelado estrechamente vinculados que proporciona a los usuarios la capacidad de representar aspectos del sistema en estudio según un paradigma o formalismo en particular.

- Nivel 0 (L0)

Nivel de cumplimiento inferior para la infraestructura UML - una sola unidad de lenguaje que hace posible el modelado de tipos de estructuras basadas en clases que se encuentran en los lenguajes más populares de programación orientados a objetos.

- Meta Object Facility (MOF)

Una especificación de modelado de OMG que brinda la base para las definiciones de metamodelos en la familia de lenguajes MDA de OMG.

- **Composición**

Es un tipo especial de relación de agregación en el que las clases componentes no pueden formar parte de otra relación de agregación, es decir, son exclusivos de la composición establecida.

- Metamodelo

Define el lenguaje y los procesos a partir de los cuales formar un modelo.

- Construcciones de metamodelos (LM)

Segundo nivel de cumplimiento en la infraestructura UML - una unidad adicional de lenguaje para estructuras más avanzadas basadas en clases, usadas para construir metamodelos (por medio de CMOF), tales como el UML mismo. UML solo tiene dos niveles de cumplimiento.

- Arquitectura dirigida por modelos (MDA)

Un enfoque y un plan para lograr un conjunto coherente de especificaciones de tecnología dirigida por modelos.

- Lenguaje de restricciones para objetos (OCL)

Un lenguaje declarativo para describir reglas que se aplican al Lenguaje Unificado de Modelado. OCL complementa a UML proporcionando términos y símbolos de diagramas de flujo que son más precisos que el lenguaje natural, pero menos difíciles de dominar que las matemáticas.

- Object Management Group (OMG)

Es un consorcio sin fines de lucro de especificaciones para la industria de la computación, cuyos miembros definen y mantienen la especificación UML.

- UML 1

Primera versión del Lenguaje Unificado de Modelado.

- Lenguaje Unificado de Modelado (UML)

Un lenguaje visual para especificar, construir y documentar los artefactos de los sistemas.

- XMI

Una especificación basada en XML de formatos de intercambio de modelos correspondientes.

Patrones de Diseño

Los **patrones de diseño** son soluciones probadas y reutilizables para problemas comunes en el desarrollo de software. Se agrupan en tres categorías principales: **creacionales**, **estructurales** y **de comportamiento**, y cada una aborda problemas específicos en la creación, organización e interacción de objetos en un sistema.

1. Patrones Creacionales

Estos patrones ayudan a controlar la creación de objetos, promoviendo la flexibilidad y evitando el uso directo de los constructores de clases. Facilitan la creación de objetos sin especificar las clases exactas de los mismos.

Singleton: Asegura que una clase solo tenga una instancia y proporciona un punto de acceso global a ella.

Factory Method: Define una interfaz para crear objetos, pero permite que las subclases decidan qué clase instanciar.

Abstract Factory: Permite crear familias de objetos relacionados o dependientes sin especificar sus clases concretas.

Builder: Descompone el proceso de creación de un objeto complejo en varios pasos o etapas, lo que permite construir el objeto paso a paso.

Prototype: Crea nuevos objetos copiando una instancia prototípica, en lugar de crear instancias a través de un constructor.

2. Patrones Estructurales

Estos patrones definen cómo estructurar clases y objetos para formar estructuras más complejas, optimizando la organización y las relaciones entre ellos.

Adapter: Permite que dos clases con interfaces incompatibles trabajen juntas convirtiendo la interfaz de una clase en otra que el cliente espera.

Decorator: Añade dinámicamente responsabilidades a un objeto, permitiendo extender su funcionalidad sin alterar su estructura original.

Facade: Proporciona una interfaz simplificada a un conjunto de interfaces en un subsistema complejo.

Composite: Permite tratar objetos individuales y grupos de objetos de la misma forma. Se usa para estructuras jerárquicas.

Proxy: Proporciona un objeto sustituto que controla el acceso a otro objeto, generalmente para optimizar el rendimiento o la seguridad.

Bridge: Separa una abstracción de su implementación, lo que permite variarlas de forma independiente.

3. Patrones de Comportamiento

Estos patrones se enfocan en la comunicación entre objetos y en la delegación de responsabilidades, ayudando a gestionar cómo los objetos interactúan y colaboran.

Observer: Define una relación de dependencia entre objetos de modo que cuando uno cambia su estado, notifica automáticamente a sus dependientes.

Strategy: Permite definir una familia de algoritmos y hacerlos intercambiables dentro de un objeto, lo que permite modificar su comportamiento.

Command: Encapsula una solicitud como un objeto, lo que permite parametrizar las acciones y organizar las operaciones.

State: Permite a un objeto cambiar su comportamiento cuando cambia su estado interno.

Chain of Responsibility: Permite que varios objetos tengan la oportunidad de procesar una solicitud al pasarla a lo largo de una cadena de manejadores.

Template Method: Define el esqueleto de un algoritmo en una operación, delegando algunos pasos a las subclases.

Mediator: Facilita la comunicación entre objetos, promoviendo la interacción a través de un mediador central.

Memento: Permite capturar y restaurar el estado interno de un objeto sin violar su encapsulación.

Interpreter: Define una representación para la gramática de un lenguaje y un intérprete para interpretar sentencias en el lenguaje.

Visitor: Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.

Ejemplos de Uso en un Proyecto de Ventas

En nuestro proyecto de ventas de productos de comida y bebidas, podrías utilizar algunos de estos patrones para facilitar el diseño del sistema:

Singleton: Podrías usarlo para gestionar una única instancia de la base de datos o de un controlador de inventario, asegurando que todos los pedidos accedan a la misma fuente de datos.

Factory Method: Para crear diferentes tipos de productos (café, postres, bebidas), donde cada subclase de producto se crea a través de una fábrica específica.

Observer: En la funcionalidad de notificaciones, donde clientes o empleados pueden ser notificados cuando un producto específico esté disponible o haya cambios en los pedidos.

Facade: Para simplificar las operaciones complejas (como consultar inventario, procesar pagos o gestionar pedidos), proporcionando una interfaz sencilla al cliente o usuario final.

Conclusiones

Conocer, analizar, componer, estructurar y caracterizar por medio de esta herramienta, el proyecto de software a realizar, teniendo en cuenta la importancia de modelar de manera sencilla y clara para que después puedan codificar de manera clara la Programación orientada a objetos para el equipo de proyecto y resultar con buenas prácticas.

Fuentes

https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml

https://incual.educacion.gob.es/documents/20195/1873855/IFC080_3+-+A_GL_Documento+publicado/ef0485d3-d87f-4c6b-9ddb-728ef9d336ac

https://zajuna.sena.edu.co/Repositorio/Titulada/institution/SENA/Tecnologia/228118/Contenido/OVA/CF17/index.html#/introduccion