MEDAC
Instituto Oficial de Formación Profesional

ENTORNOS DE DESARROLLO

El lenguaje UML

6

ÍNDICE

/ 1. Introducción y contextualización práctica 3

/ 2. Utilización de metodologías de desarrollo orientado a objetos 4

/ 3. Qué es UML 4

/ 4. Caso práctico 1: “Estandarización del modelado” 5

/ 5. Versiones de UML y herramientas 6

/ 6. Principales diagramas en UML 6

/ 7. Caso práctico 2: “HERRAMIENTAS” 7

/ 8. Resumen y resolución del caso práctico de la unidad 8

/ 9. Bibliografía 8

OBJETIVOS

Saber qué es una metodología de desarrollo.

Conocer a diferentes metodologías orientadas objetos.

Saber identificar las herramientas para el modelado.

Conocer los principios de UML.

/ 1. Introducción y contextualización práctica

A la hora de modelar un sistema es importante disponer de las herramientas necesarias para realizarlo correctamente. Además, es imprescindible considerar el contexto en el que se desarrolla nuestro modelado. Será diferente el modelado realizado para una aplicación desarrollada mediante tecnologías no orientadas a objetos que para tecnologías de este tipo.

En la actualidad existe un gran número de metodologías para el modelo software que se adapta en mayor o en menor medida al contexto y tecnología utilizada en el desarrollo. Dentro del paradigma de orientación a objetos, que pertenece al paradigma de programación imperativa, existen diferentes metodologías para el desarrollo. A su vez, cada una de las metodologías requieren un conjunto de herramientas que permitan llevarlas a cabo. Por ejemplo, es necesario que la metodología disponga de todas las normas y reglas para poder implementarla correctamente.

Fig. 1. El modelado del software requiere un gran estudio previo

Algunas metodologías ofrecen a sus usuarios lenguajes para modelar correctamente. A la hora de elegir una metodología es importante tener en cuenta lo estandarizada que está y su penetración en el mundo del desarrollo.

Escucha el siguiente audio en el que planteamos el caso práctico que iremos resolviendo a lo largo de esta unidad.

Audio intro. ". Proceso unificado y UML"
https://bit.ly/2DSNQGQ

/ 2. Utilización de metodologías de desarrollo orientado a objetos

Existen diferentes tipos de metodologías para el desarrollo software teniendo en cuenta el enfoque de desarrollo que queramos seguir. Uno de los principales beneficios de utilizar metodologías orientadas a objetos es que éstas facilitan la reutilización de componentes software.

Al seguir un modelo basado en objetos, cuando utilizamos estas metodologías es necesario que utilicemos un lenguaje orientado a objetos. No tendría sentido utilizar una metodología orientada objetos si el lenguaje que vamos a utilizar para la implementación no lo es. Todas las metodologías orientadas a objetos deben utilizar un lenguaje de modelado que sea orientado a objetos.

Actualmente, uno de los lenguajes más utilizados para la modelación orientada objetos es el lenguaje **UML** (Lenguaje unificado de modelado, del inglés, Unified Modeling Language). Utilizando una metodología orientada objetos es posible construir un sistema a través de la unión de diferentes componentes, pues estos permiten desarrollar funcionalidades básicas que pueden ser unidas para crear una funcionalidad mayor.

El principio básico que estamos utilizando es por un lado la reutilización y por otro lado la limitación de las funcionalidades de cada módulo. Una metodología basada en objetos se centra en determinadas fases como, por ejemplo, la de análisis para concretar qué elementos se encuentran en el sistema. Existen al menos tres tipos de metodologías orientadas a objetos:

- **Técnicas de Modelado de Objetos (OMT)**. Se trata de una de las primeras metodologías orientadas a objetos que se crearon.
- **Metodología de Proceso de Objetos (OPM).** Se introdujo después de la metodología anterior y una de sus principales características es que únicamente posee un tipo de diagrama denominado diagrama de proceso de objeto que permite modelar las estructuras funciones y comportamiento del sistema.
- **Proceso Unificado de Rational (RUP).** Se trata de una metodología bastante compleja que introduce diferentes fases en el desarrollo que requieren dos interacciones.

Audio 1. "Metodologías de desarrollo"
https://bit.ly/3afShrv

/ 3. Qué es UML

UML, es un lenguaje estándar que permite el modelado de objetos existentes en el mundo real para ser utilizado en una metodología orientada a objetos.

Se trata de un lenguaje de propósito general que se centra en los dominios de la ingeniería del software y cuya intención es generar un estándar para la visualización del diseño de un sistema. Al ser un lenguaje de propósito general no está vinculado a ningún contexto de desarrollo, lo que significa que se puede adaptar a cualquier tipo de problema siempre y cuando sea orientada objetos.

De esta forma, UML se ha convertido en un lenguaje estándar para generación de modelados que se utiliza en entornos profesionales. Además, ofrece la capacidad de visualizar diferentes etapas en el desarrollo software como la de análisis, diseño e implementación.

a. **Características principale**s

La principal característica de UML es que es un lenguaje de modelado software de propósito general y estándar.

Gracias a que es un lenguaje estándar, sus reglas están bien definidas y se pueden intercambiar diagramas y modelados entre diferentes regiones. Además, UML es capaz de reducir el coste de desarrollo de un proceso software mediante la utilización de diagramas que permiten visualizar fácilmente nuestro sistema. Esto se puede realizar de una manera más sencilla utilizando herramientas de generación de diagramas y visualización.

De esta forma podemos decir que UML permite reducir el tiempo de desarrollo y solventar errores que podrían suceder en el pasado cuando no existía ningún tipo de documentación.

Gracias a la utilización de este lenguaje tenemos una fuerte ayuda visual para la construcción del software. Esto permite que trabajar con nuevos desarrolladores sea más sencillo, pues el sistema será más fácil de entender. Además, la comunicación con clientes y desarrolladores será también más sencilla y eficiente.

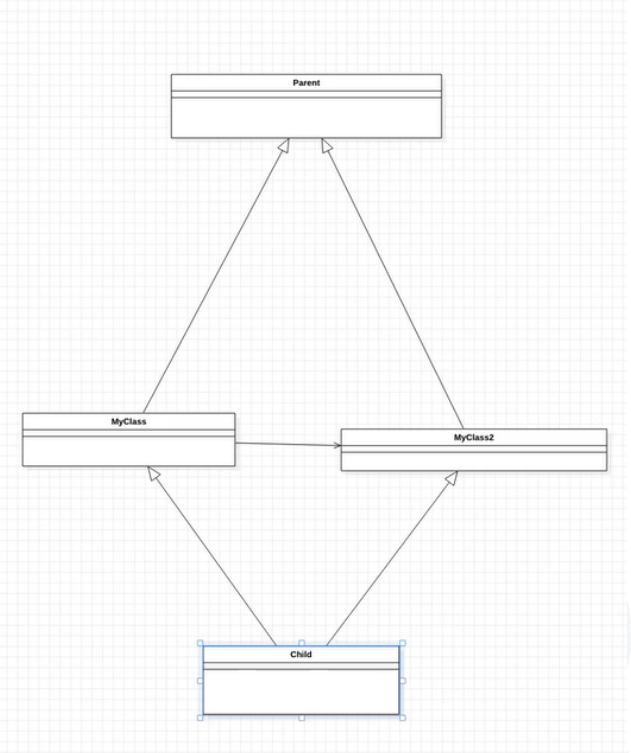

Fig 2. Ejemplo básico diagrama de clases con UML

Enlaces de interés...

En este enlace puedes encontrar una lectura recomendada para el tema sobre qué es y para que sirve UML.

/ 4. Caso práctico 1: "Estandarización del modelado"

Planteamiento. En nuestro departamento de desarrollo software se plantea la necesidad de crear un nuevo sistema de integración de aplicaciones y se nos pide que realicemos su modelado, sabiendo que va a ser desarrollado con lenguajes de orientación a objetos.

Nuestro jefe ha oído hablar de una nueva metodología de desarrollo software en la orientación objetos y aunque no se conoce demasiado, nos propone que la utilicemos, pues todas las herramientas de modelado son gratuitas.

Nudo. ¿Qué opinas de esta situación?

Desenlace. Según hemos visto a largo del tema, el desarrollo software debe utilizar una metodología que lo permita llevar a cabo para ahorrar costes y aumentar la productividad de los desarrolladores.

Como vamos a trabajar con una orientación a objetos lo adecuado sería utilizar una metodología orientada a objetos. A largo del tema también hemos visto que es mejor utilizar metodologías que sean estándar para poder facilitar el intercambio de modelos y basarnos en las reglas ampliamente utilizadas y validadas por otros modelos y desarrolladores. Al utilizar metodologías que sean estándar podemos ganar en seguridad a la hora de crear nuestros modelos.

Fig. 3. La estandarización del modelado nos permite ahorrar tiempo

Cuando nuestro jefe nos pide que utilizamos una metodología que no ha sido probada suficientemente justificando la elección por el precio de las herramientas, tenemos que considerar que existen herramientas gratuitas para metodologías orientadas a objetos suficientemente probadas ya.

/ 5. Versiones de UML y herramientas

a. **Versiones de UML**

UML ha ido evolucionando a lo largo del tiempo con el fin de añadir nuevas funcionalidades y poder adaptarse a las necesidades y desarrollo de los nuevos lenguajes de programación orientados a objetos. Sabemos que la etapa de modelado es diferente a la etapa de implementación, pero es necesario que exista una unión entre el modelado y la implementación.

En la evolución de UML se puede observar que existen tres principales versiones: UML 1.0, UML 1.x y UML 2.0 (versión actual). En la versión actual se ha definido un árbol completo de las estructuras en las que se relacionan los diferentes diagramas de lenguaje para poder definir el modelado de cualquier componente software.

A pesar de las diferentes actualizaciones, el lenguaje sigue siendo cuestionado porque, a pesar de ser un estándar, las reglas no son lo suficientemente rígidas para garantizarlo. Uno de los problemas que tiene UML es la interpretación que se puede realizar de los diagramas a pesar de que éstos estén correctamente implementados.

b. **Herramientas para la elaboración de diagramas**

Existen múltiples herramientas para la generación de diagramas UML que pueden ser utilizadas en entornos profesionales. La mayor parte de estas herramientas suelen ser de pago, pues se trata de herramientas con características muy específicas. Algunas de ellas pueden ser incorporadas a los IDEs de desarrollo, permitiendo la vinculación de la generación de código a través de diagramas UML. Algunos ejemplos de herramientas para el modelado a través de UML son:

- **Rational Rose**. Se trata de una herramienta profesional de pago y que incluye todos los diagramas UML del modelo.
- **StarUML.** Es una herramienta que se puede vincular a diferentes IDEs, siendo fácil de usar y bastante rápida.
- **Papyrus UML.** Se trata del entorno de modelado estándar utilizado dentro de Eclipse, siendo gratuito y de código abierto. Para conocer más sobre él, así como para su descarga e instalación puedes acceder aquí.

Video 1. "StarUML"
https://bit.ly/347ZYgM

/ 6. Principales diagramas en UML

Como hemos visto anteriormente, UML es un lenguaje de modelado que permite representar visualmente los elementos y comportamiento de nuestro sistema. Dentro de este lenguaje podemos destacar los diferentes diagramas que lo componen y que permiten generar la visualización de todo nuestro sistema software.

Al estar compuesto por diferentes tipos de diagramas, es importante determinar qué diagrama es el adecuado para representar la información que necesitemos. Nótese que, cada tipo de diagrama es útil en una fase del desarrollo software. De esta forma, los principales diagramas que podemos diferenciar en UML (y que estudiaremos en unidades posteriores) son los siguientes:

- **Diagrama de casos de uso.** Permiten indicar con detalle el comportamiento que debe tener el sistema desde el punto de vista del usuario.
- **Diagrama de clases.** Permite mostrar las relaciones que existen entre las diferentes clases, así como el modelo de datos que contiene. También es importante destacar los paquetes a los que pertenecen las clases.
- **Diagramas de secuencia.** Se trata de un diagrama que permite representar interacciones de manera dinámica.
- **Diagramas de colaboración.** Permite representar la interacción que tiene un objeto con su alrededor.
- **Diagramas de estado.** Muestra la secuencia de estados por la que pasa un objeto a lo largo de sus interacciones y su tiempo de vida.
- **Diagramas de componentes.** Permite representar una visión de alto nivel de la estructura de paquetes del sistema.
- **Diagramas de despliegue**. Se trata de un diagrama que permite mostrar la configuración necesaria para la ejecución de todos los componentes software en un entorno real.

Fig. 4. Cada diagrama aporta su grano de arena al desarrollo software

/ 7. Caso práctico 2: "HERRAMIENTAS"

Planteamiento. A la hora de realizar el modelado de un sistema se nos plantea la necesidad de utilizar lenguajes como UML. Dado que nuestra experiencia no es muy amplia en este campo decidimos utilizar herramientas de dibujo tradicional o suites de ofimática para realizar dicho el modelado, ya que nos sentimos más cómodos con éstas.

Nudo. ¿Crees que es una aproximación correcta?

Desenlace. Como hemos visto en el tema, existen herramientas que permiten realizar el modelado software basándose en los requisitos de alguna de las metodologías más utilizadas. Por ejemplo, en el caso de la metodología UML existen herramientas que ya permiten realizar el modelado de una manera más sencilla que las herramientas ofimáticas o las herramientas de dibujo. De esta forma, es más sencillo utilizar el lenguaje UML, pues gracias a las herramientas de modelado ya existentes y su vinculación con los entornos de desarrollo, es posible generar código de los modelos o bien, generar el modelo desde el código (esta característica es muy importante en el mundo profesional y la veremos en más profundidad en la siguiente unidad).

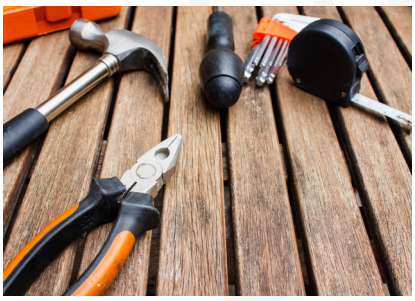

Fig 5. Cada problema requiere las herramientas necesarias

Al utilizar herramientas que están diseñadas para trabajar en entornos ofimáticos y no en entornos de desarrollo, perderemos la capacidad de poder modelar y generar código o generar el modelo a partir del código.

/ 8. Resumen y resolución del caso práctico de la unidad

A largo de este tema hemos presentado **diferentes metodologías** para el desarrollo software utilizando un enfoque orientado objetos. Para poder apoyar estas metodologías se ha introducido un lenguaje de modelado que permite crear fácilmente una representación visual del sistema que queramos implementar. Para ello, hemos conocido **UML** como lenguaje de modelado, considerado un estándar y el más utilizado en la actualidad.

Existen **diferentes versiones de UML** que han ido incorporando nuevas capacidades al modelado. Dentro de UML hemos visto que existen **diferentes tipos de diagramas.**

Como hemos podido comprobar, destaca el diagrama de clases frente a otros. Además, hemos presentado diferentes tipos de herramientas que permiten el modelado a través de UML.

Resolución del caso práctico de la unidad

Independientemente de la metodología que estemos utilizando para el desarrollo software, es importante tener en cuenta que los diagramas que realicemos se realicen con un lenguaje estándar. En caso contrario, tendremos problemas a la hora de intercambiar e interpretar dichos diagramas y la documentación en sí misma.

A largo de este tema hemos visto que podemos utilizar UML para poder crear el diseño y modelado de un sistema de manera estándar. Gracias a este lenguaje, podemos incorporar gran cantidad de diagramas que pueden ofrecer diferentes perspectivas del sistema, mostrando grandes detalles.

En el caso se nos pide que utilicemos herramientas genéricas de propósito general para generar los diagramas que van a documentar nuestro software. Esta aproximación no sería una buena práctica al no tratarse de soluciones estándar para la documentación del software.

Fig 6. Filosofía SMART para tus objetivos

/ 9. Bibliografía

Juan, J. & Resusta, L. (2014). UML práctico: aprende UML paso a paso. España: editor no identificado.

Jacobson, I., Booch, G. & Rumbaugh, J. (2000). El proceso unificado de desarrollo de software. Madrid: Addison Wesley.

Larman, C. (2002). UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado. Madrid: Prentice Hall.