



Buenas prácticas de software



Unidad 3. Mejores prácticas para la ingeniería de diseño.



Buenas prácticas de software



3.1 Mejores prácticas para el modelado de software.



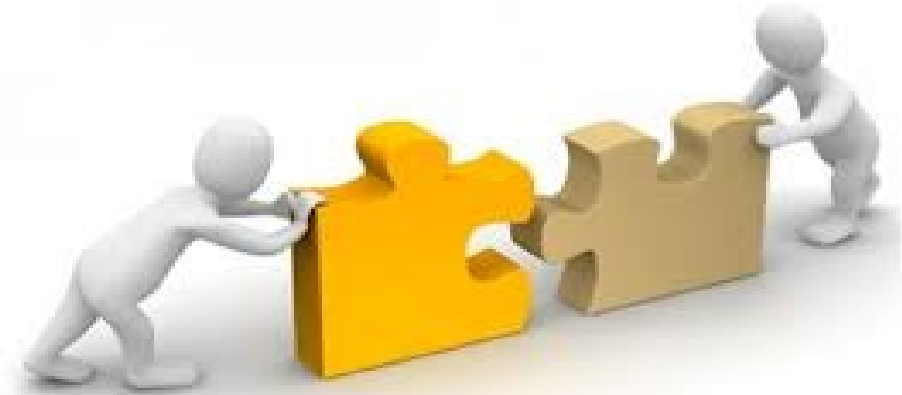
Buenas prácticas de software



Consideraciones

Una metodología de desarrollo de software consta de varias etapas, que se pueden desarrollar de manera secuencial o paralela dependiendo del tipo de estrategias que se defina.

Pero sin dudarlas dichas etapas deben llegar a la construcción del software.





Buenas prácticas de software



Análisis

Se identifique se quiere hacer, con que y como se desarrollara. Construye un modelo de los requisitos.

Diseño y desarrollo

A partir del análisis se deducen las estructuras de datos y de la información, así como el diseño lógico y físico (interfaz de usuario)



Buenas prácticas de software



3.1.1 Esquematización del modelo del software.

Sommerville define modelo de proceso de software como *“Una representación simplificada de un proceso de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto, un modelo de procesos del software es una abstracción de un proceso real ([Sommerville, 2005](#)).”*



Buenas prácticas de software



3.1.1 Esquematización del modelo del software.

El modelado del software, es una técnica que permite identificar la estructura del software a desarrollar.

La elaboración del modelo ayuda a "identificar y visualizar" cada una de las partes y al sistema en su conjunto.

Se crean modelos de un alto nivel, que se utilizan para la comunicación con el cliente y para concretar los requerimientos que serán desarrollados.



Buenas prácticas de software



3.1.1 Esquematización del modelo del software.

El modelado es vital en todo tipo de proyectos,

El modelado permite identificar la arquitectura del software.

Para el modelado del software se utiliza el lenguaje UML.

UML se usa como lenguaje gráfico para modelar el software y permite: Definir, especificar, construir y documentar los procesos o funciones del software.

UML es visual, y modela distintos aspectos del software que permiten una mejor interpretación y entendimiento de los requerimientos y funciones del software.

Con UML se esquematiza y se representan las diferentes partes de un sistema de software, para ello se utilizan varios tipos de diagrama.

Cada diagrama sirve para modelar diferentes partes o puntos de vista de un sistema de software.



Buenas prácticas de software



3.1.2 Técnicas y lenguajes de modelado.

Sommerville define modelo de proceso de software como *“Una representación simplificada de un proceso de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto, un modelo de procesos del software es una abstracción de un proceso real ([Sommerville, 2005](#)).”*



Buenas prácticas de software



3.1.2 Técnicas y lenguajes de modelado.

La herramienta Lucidchart es una herramienta online muy útil para dibujar rápidamente diagramas UML complejos.

Con la cuenta gratuita tienes acceso a las siguientes plantillas de diagramas UML:

Diagrama de clases.

Diagrama de estados.

Diagrama de actividades.

Diagrama secuencial.

Diagrama de componentes.

Diagrama de casos de uso.

Diagrama de despliegue.

La herramienta permite importar datos y diagramas con la cuenta gratuita.

Lucidchart es compatible con los formatos de diagramas nativos de Microsoft Visio, Omnigraffle, Gliffy y Draw.io.



Buenas prácticas de software



Beneficios del modelado de software.

1. Mejora la productividad del equipo de desarrollo.
2. Reduce el número de defectos en el código.
3. Facilita la comprensión de las funciones.
4. Mejora la descomposición y modularización del software.
5. Facilita el desarrollo y mantenimiento del software.
6. Mejora la reusabilidad.



Buenas prácticas de software



3.1.3 Nomenclatura para el modelado de diagramas UML.

El Lenguaje de Modelado Unificado (UML), es propio del modelo orientado a objetos.

Permite visualizar la estructura de un software o sistema.

Es una herramienta práctica que utilizan los desarrolladores para el modelado lógico de un sistema o software.



Buenas prácticas de software



Modelo orientada a objetos.



Se basa en un conjunto de objetos que inter actúan entre sí.

Se basa en componentes, lo que facilita la reutilización de código.

Esta orientada a objetos y no a procesos.

Esta orientada a métodos y no a procedimientos o funciones.



Buenas prácticas de software



La metodología Orientada a Objetos consta de los siguientes elementos:

Objetos.

Consta en una estructura de datos, los cuales son sus atributos.



Clases.

Son los nombres de todos los tipos de datos y almacén de datos junto con sus definiciones que comparten atributos.

Diagrama de casos de uso.

Modelan los requisitos funcionales (componentes del software).

La secuencia que representan la interacción de los elementos externos con el sistema, indica que hace el sistema, hace referencia a un tipo de caso a las que se ajustan las diferentes instancias (sucesos y comportamientos) de dichos casos.



Buenas prácticas de software



Un caso de uso.

Permite identificar que hace el sistema.

Por lo que son un conjunto de secuencias internas y externas que desarrolla el sistema. Gráficamente se denota con un elipse.

Los escenarios.

Son la secuencia de acciones que desarrolla el caso de uso.

Gráficamente es un conjunto de elipses.

El actor:

Modelan los requisitos funcionales

Representan entidades externas al sistema.

Puede ser una persona, sistema o procesos. Su representación es un muñequito.

Actor.

Interaccionan con el sistema mediante mensajes cortos que se colocan dentro de los casos de uso (elipse) e indican la acción que realiza el sistema o usuario.





Buenas prácticas de software



Relaciones: Se representan mediante una línea y denotan la asociación del actor con el caso de uso.

Hay tres tipos de relaciones

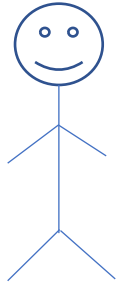
La Conexión, se representa con una línea continua y representa la asociación sencilla del actor y un caso de uso.

La Inclusión, se representa por una flecha punteada y en la parte superior se coloca la palabra incluye. Permite incorporar un caso de uso al de base.

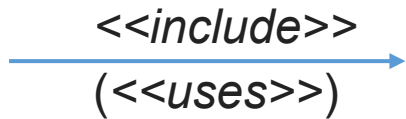
La Extensión, incluye otro caso de uso e indica un comportamiento adicional del caso de uso. Se representa con una línea punteada y en la parte superior se coloca la palabra extend

La Generalización, se representa con una flecha punteada y es un elemento hijo del caso de uso hereda el comportamiento de otro caso de uso.

3.1.3 Nomenclatura para el modelado de diagramas UML.



Actor

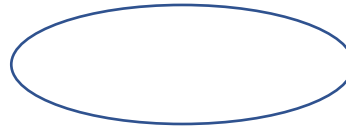


Include

o

Inclusión

Indica la relación entre dos casos de uso, en donde hay dependencia, actividades similares o iguales.

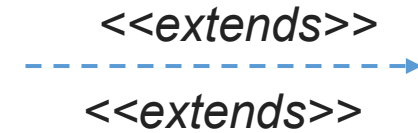


Caso de Uso



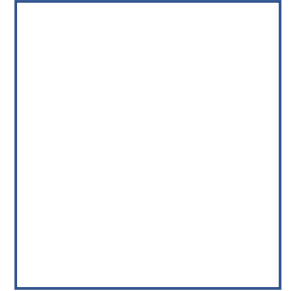
Comunica

Simplemente conecta un actor con un caso de uso, indica la participación del actor con el caso de uso




Extends o Extender

Cuando un caso de uso requiere de otra acción o función para finalizar. La dependencia de los casos de uso



Límite del sistema



Generaliza, la punta de la flecha indica hacia el caso de uso general

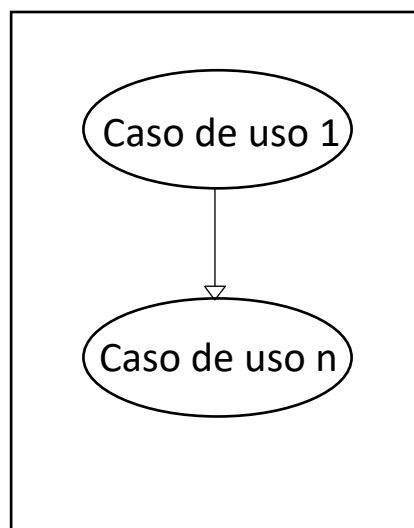


Buenas prácticas de software

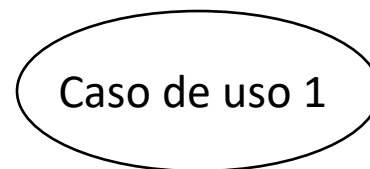


Diagrama de casos de uso

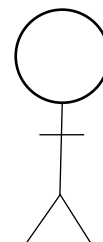
- Sistema
El rectángulo representa los límites del sistema que contiene los *casos de uso*. Los *actores* se ubican fuera de los límites del sistema.



- Casos de Uso
Se representan con óvalos. La etiqueta en el óvalo indica la función del sistema.



- Actores
Los *actores* son los usuarios de un sistema.





Buenas prácticas de software



Relaciones

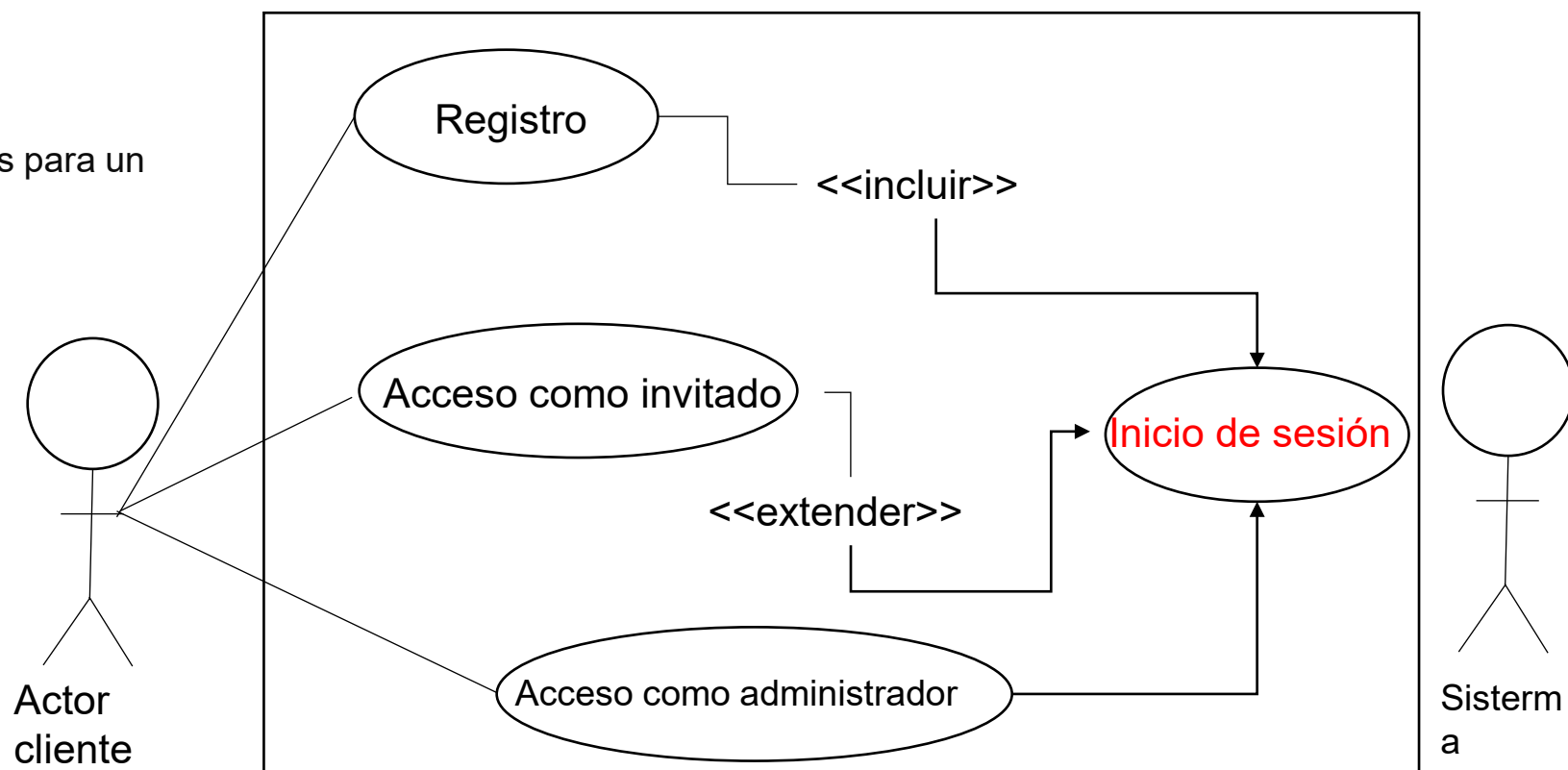
Las relaciones entre un *actor* y un *caso de uso*, se dibujan con una línea simple.

Para relaciones entre *casos de uso*, se utilizan flechas etiquetadas "incluir" o "extender."

Una relación "incluir" indica que un *caso de uso* es necesitado por otro para poder cumplir una tarea.

Una relación "extender" indica opciones alternativas para un cierto *caso de uso*.

Recuperar datos de registro





Buenas prácticas de software



Diagrama de Clases

Los *diagramas de clases* describen la estructura estática de un sistema.

Una *clase* es una categoría o grupo de cosas que tienen *atributos* (propiedades) y *acciones* similares.



Buenas prácticas de software



Un ejemplo puede ser la *clase* “Auto” que tiene *atributos* como el “modelo”, “el motor”, “la velocidad” y “la capacidad”.

Entre las acciones de esta clase se encuentran: “acelerar”, “moverce”, “dar vuelta”, “parar”, “desacelerar”.

Un rectángulo es el símbolo que representa a la *clase*, y se divide en tres áreas.

Nombre de la clase
Atributo: Tipo/atributo deriado
OPERACIÓN ()

Auto
Modelo
Motor
Velocidad
Capacidad
Acelerar ()
Moverse ()
Dar vuerta ()
Parar ()
Desacelerar ()



Buenas prácticas de software



3.1.4 Herramientas y marcos de trabajo código libre para el modelado.

¿Qué son las herramientas para el modelado de código libre?

Una licencia de software otorga al usuario el derecho legal a utilizar un software, existen básicamente dos tipos de licencias: Licencia de software libre y Licencias de software comercial.

Según la Fundación de Software Libre (Free Software Foundation) «el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software», se refiere a cuatro libertades para los usuarios del software: la libertad de usar el programa, con cualquier propósito; de estudiar el funcionamiento del programa, y adaptarlo a las necesidades; de distribuir copias, con lo que puede ayudar a otros; de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad informática se beneficie.



Buenas prácticas de software



3.1.5 Herramientas y marcos de trabajo para el modelado.

¿Qué son las herramientas para modelado comerciales?

En las licencias de software comercial una compañía o corporación, posee los derechos de autor sobre un software, negando o no otorgando los derechos de usar el programa con cualquier propósito, de estudiar cómo funciona el programa y adaptarlo a las propias necesidades y además el acceso al código fuente es una condición previa, se pueden distribuir copias, o mejorar el programa y hacer públicas las mejoras, para poder realizar los cambios, el acceso al código fuente es un previo requisito.

Así, un software sigue siendo no libre aún si el código fuente es hecho público, cuando se mantiene la reserva de derechos sobre el uso, modificación o distribución.



Buenas prácticas de software



Comparación de herramientas para modelado UML.

- Compatibilidad con sistemas operativos.

Herramienta	Windows	Linux	MacOS
ArgoUML	x	x	
BOUML	x	x	x
Lucidchart	Web		
GModeler	x	x	
MagicDraw	x	x	x
MonoUML		x	
StarUML	x		
Umbrello	x	x	
Netbeans UML	x	x	x

- Libre o comercial.

Herramienta	Libre	Costo
ArgoUML	x	
BOUML	x	
Lucidchart		x
GModeler	x	
MagicDraw		x
MonoUML	x	
StarUML		x
Umbrello	x	
Netbeans UML	x	



Buenas prácticas de software



Comparación de herramientas para modelado UML.

Diagramas que realizan.

Herramienta	CU	CL	S	CO	A	O	P	E	ET
ArgoUML	x	x	X	X	x			x	
BOUML	x	X	X	x		x			
Lucidchart	x	x	x		x	x	x	x	x
GModeler		x					x		
MagicDraw	x	x	x	x	x			x	
MonoUML	x	x							
StarUML	x	x	x	x	x				x
Umbrello	x	x	x	x	x			e	
Netbeans UML	x	x	x	x	x	x	x	x	x

CU: casos de uso.
CL: clases.
S: secuencia.
CO: colaboración.
A: actividades.
O: objetos.
P: paquetes.
E: estados.
ET: estructura.



Buenas prácticas de software



Comparación de herramientas para modelado UML.

Generan código con base en el diagrama.

Herramienta	Java	C++	C#	PHP	Python	Ruby
ArgoUML	x	x	x	x	x	
BOUML	x	x		x	x	
MagicDraw	x	x	x			
MonoUML			x			
StarUML	x	x	x			
Umbrello	x	x	x			x
Netbeans UML	x	x	x			