



UML

DANIEL SANTANA
RAUL JIMENEZ
MIKHAEL SHMATENKO
JUAN ALBERTO ACOSTA

CONTENIDO

- 01** Introducción a UML
- 02** Beneficios de uso de UML
- 03** Conceptos Fundamentales
- 04** Elementos de UML
- 05** Diagramas de casos de uso
- 06** Diagramas de clase
- 07** Diagramas de actividades

INTRODUCCION A UML

¿QUE ES UML?

UML (Lenguaje de Modelado Unificado) es un lenguaje visual estándar que sirve para especificar, construir, visualizar y documentar sistemas, especialmente de software. No es un lenguaje de programación, sino una herramienta de comunicación que usa diagramas para representar la estructura y el comportamiento de sistemas. El propósito de UML es facilitar la compresión del sistema

HISTORIA DE UML

En los años 90, para unificar diversas metodologías de diseño orientado a objetos, Rumbaugh, Booch y Jacobson crearon UML, combinando sus enfoques. UML 1.0 se publicó en 1997 y, entre 1999 y 2005 surgieron mejoras como UML 2.0, que se consolidó como estándar. Hoy sigue siendo ampliamente usado en desarrollo de software.

PARA QUE SE UTILIZA

Modelar sistemas de software desde el análisis de requisitos hasta el diseño detallado

Visualizar la estructura y comportamiento de un sistema

Comunicar ideas técnicas entre desarrolladores, analistas y clientes de manera clara

Documentar procesos y arquitecturas de sistemas

Diseñar sistemas operativos, incluyendo sistemas en tiempo real, distribuidos y empresariales

CONCEPTOS FUNDAMENTALES

MODELAR

Modelar es crear representaciones visuales de un sistema de software. Los diagramas se usan para entender, especificar, construir y documentar el sistema

La abstracción es mostrar solo lo esencial de un sistema u objeto, ocultando los detalles internos. Se aplica con interfaces o clases abstractas para definir qué hace un elemento, sin detallar cómo lo hace

ABSTRACCION

ENCAPSULAMIENTO

El encapsulamiento consiste en ocultar los detalles internos de una clase y restringir el acceso directo a sus datos. Protege la información permitiendo interactuar con ella solo a través de métodos públicos como getters y setters.

CONCEPTOS FUNDAMENTALES

VISIBILIDAD

La visibilidad controla quién puede acceder a un método o atributo

- + Público: Accesible desde cualquier parte
- - Privado: Solo accesible dentro de la clase
- # Protegido: Accesible desde la clase y sus subclases
- ~ De paquete: Accesible dentro del mismo paquete

El alcance indica si un atributo o método pertenece a la clase o a las instancias

- De instancia: Pertenece a un objeto específico
- Estático / de clase: Pertenece a la clase y se representa en UML con el nombre subrayado

ELEMENTOS ESTRUCTURALES

ATRIBUTOS

Los atributos son las propiedades o datos de los objetos de una clase, como “nombre” o “edad” en una clase “Persona”. Pueden contener Visibilidad (los signos mostrados anteriormente “+,-,#,~”), Tipo de dato que almacenan ej: String, estaticidad (si pertenecen a la clase o a cada objeto) y derivación que es cuando un atributo se calcula a partir de otro (/)

Un actor es una entidad externa que interactúa con el sistema en un caso de uso, como personas, sistemas o eventos externos. Se clasifican en:

- Actores principales: esenciales para el funcionamiento del sistema, como los usuarios que inician los casos de uso.
- Actores secundarios: no indispensables, pero apoyan la eficiencia del sistema, como otros sistemas o componentes.

ACTOR

CLASE

Una clase es una plantilla para crear objetos mediante atributos y métodos. Estas se dividen en:

- Clase concreta: Clase normal para crear objetos.
- Clase abstracta: Se usan como bases para otras clases, nombre en cursiva o con <<abstract>>.
- Interfaz: Contrato de acciones, representada con círculo o <<interface>>.

ELEMENTOS ESTRUCTURALES

OBJETOS

Son las instancias de una clase los cuales tiene valores concretos y puede realizar los comportamientos de esa clase

- Objeto de entidad: Representa datos clave del sistema.
- Objeto de control: Gestiona la lógica, decisiones y coordina la interacción entre objetos.
- Objeto de frontera: Representa la interfaz entre el sistema y usuarios o actores externos

METODOS

Los métodos son acciones que realizan clases u objetos. Tipos principales:

- Constructor: Crea instancias (rara vez se muestra en UML).
- Destructor: Libera recursos (poco común en UML).
- Getter/Setter: Acceden o modifican atributos.
- Abstracto: Declarado sin implementación, en cursiva en UML.
- Interfaz: Declarado en interfaces, implementado por clases.
- Sobrecargado: Mismo nombre, distintos parámetros (no se diferencia explícitamente en UML).

ELEMENTOS ESTRUCTURALES

INTERFACES

Una interfaz define métodos que una clase debe implementar, especificando qué hacer, no cómo hacerlo.

- Actúa como un contrato que las clases deben cumplir.
- Es independiente de la implementación; solo define las firmas de los métodos.
- Permite polimorfismo, ya que distintas clases pueden implementar la misma interfaz de diferentes maneras.

Representa un recurso físico donde se ejecutan componentes del sistema, como un servidor, ordenador. Estos pueden ejecutar uno o varios componentes del sistema y puede contener otros nodos anidados dentro de otro nodo

NODO

COMPONENTE

Representa una unidad lógica de software, como un módulo, biblioteca o subsistema, que encapsula cierta funcionalidad y puede ser desplegado en un nodo. Es decir que es un pieza de software que se ejecuta en un nodo

ELEMENTOS DE AGRUPACION

PAQUETE

Un paquete es un elemento estructural usado para agrupar elementos relacionados (clases, casos de uso, componentes, etc.).

Características principales:

- Organización: Agrupa elementos para mantener el sistema ordenado.
- Jerarquía: Puede contener otros paquetes (subpaquetes).
- Modularidad: Facilita la división del sistema en partes manejables.
- Visibilidad: Define relaciones y dependencias entre los elementos agrupados.

ELEMENTOS DE COMPORTAMIENTO

CASOS DE USO

Es una representación que muestra cómo interactúan los actores con el sistema, enfocándose en las funcionalidades desde la perspectiva del usuario. Tipos de actores:

- Cliente: Persona que utiliza la aplicación.
- Administrador: Persona que gestiona o configura el sistema.
- Sistema externo: Otro sistema que interactúa con el sistema principal.

Es el conjunto de valores actuales de los atributos de una clase en un momento dado.

Ejemplo:

- Para la clase Coche, atributos como marca, modelo y velocidad

ESTADO DE UN OBJETO

ACTIVIDAD

Es una representación gráfica del flujo de trabajo o comportamiento del sistema.
Muestra cómo se desarrollan las actividades y cómo se conectan las acciones dentro de un proceso o tarea

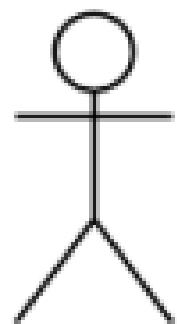
DIAGRAMA DE CASOS DE USO

Un diagrama de casos de uso en UML es un tipo de diagrama de comportamiento que se utiliza para representar como interactúan los actores con un sistema y qué funcionalidades ofrece. Esto es muy común ya que permiten ver quién usa el sistema y cómo lo hace y qué funciones están disponibles. Además, proporciona una base clara para el diseño posterior del sistema, ayudando a entender su comportamiento frente a distintos eventos o estímulos.

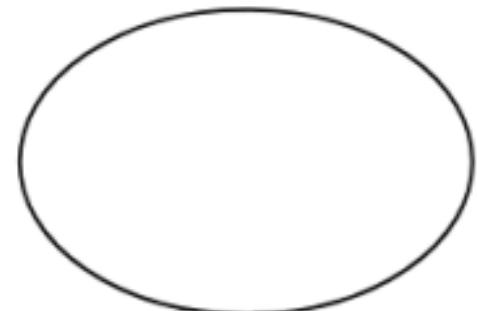
ELEMENTOS DE LOS DIAGRAMAS DE CASOS DE USO

ACTORES

Representan a las personas, otros sistemas, o entidades externas que interactúan con el sistema. Se dibujan como muñecos (stick figures)



Actor



Son acciones o funciones que el sistema pueden realizar. Se representan como óvalos.

CASOS DE USO

RELACIONES

Conectan actores con casos de usos o casos de usos entre si.

RELACIONES EN LOS DIAGRAMAS DE CASOS DE USO

RELACIONES ENTRE ACTOR Y CASOS DE USO

LA RELACION ENTRE UN ACTOR Y UN CASO DE USO SE DENOMINA RELACION DE COMUNICACCION EN ESTA EL ACTOR PUEDE SER:

ACTIVO

Cuando el actor inicia es uso de un caso de uso. Si eso usa un flecha como relación esta debe apuntar al caso de uso, si no hay flecha el actor es activo de por si

PASIVO

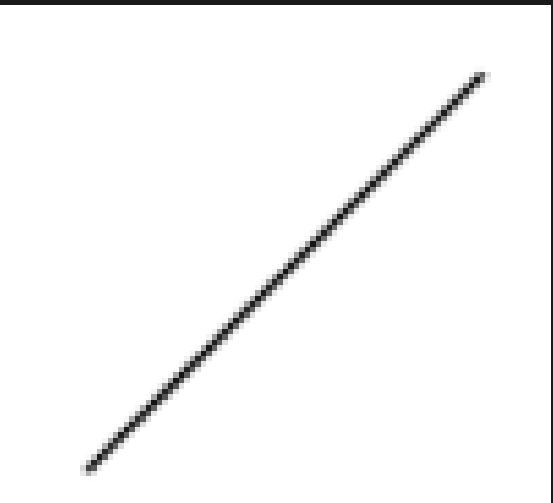
Cuando el caso de uso es iniciado por el software y no por el actor. La flecha debe apuntar al actor

RELACIONES EN LOS DIAGRAMAS DE CASOS DE USO

RELACIONES ENTRE CASOS DE USO Y CASOS DE USO

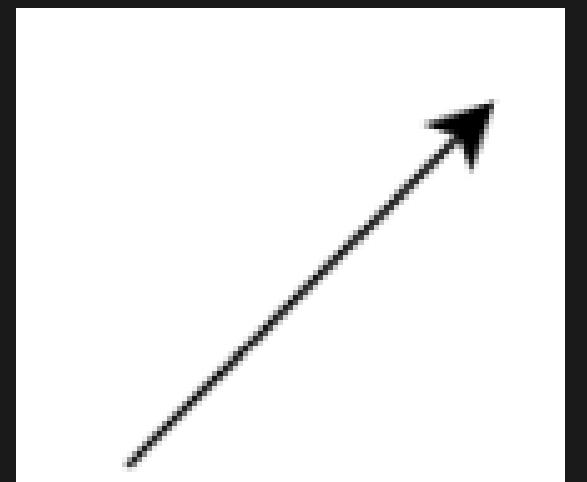
ASOCIACION

Esta relación se representa mediante un línea y se usa para representar la comunicación entre un actor y un caso de uso, entre casos de uso o entre actores.



GENERALIZACION

La generalización en UML permite que varios casos de uso o actores compartan un comportamiento común. Se representa con una flecha y solo se aplica entre casos de uso o entre actores. Un ejemplo es “Realizar pedido” como caso general para pedidos en tienda, online o por teléfono.

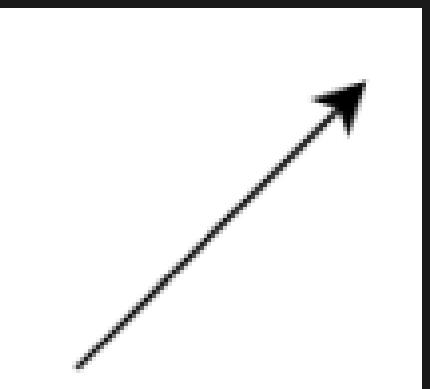


RELACIONES EN LOS DIAGRAMAS DE CASOS DE USO

RELACIONES ENTRE CASOS DE USO Y CASOS DE USO

INCLUDE

Esta relación se usa cuando varios casos de uso comparten pasos obligatorios. Se representa con una flecha punteada desde el caso base hacia el caso incluido y solo se da entre casos de uso. Por ejemplo, “Seleccionar cliente” puede incluirse en “Ingresar pedido”, “Facturar” y “Registrar pago” si todos requieren ese paso.



EXTEND

Esta relación se usa para representar comportamientos opcionales que amplían un caso de uso. La flecha punteada va desde el caso extendido hacia el caso base, y solo se usa entre casos de uso. Por ejemplo, en un sistema bancario, el “Sistema de autenticación alternativo” podría extender el caso de uso “Autenticación” si el usuario está en otro país.

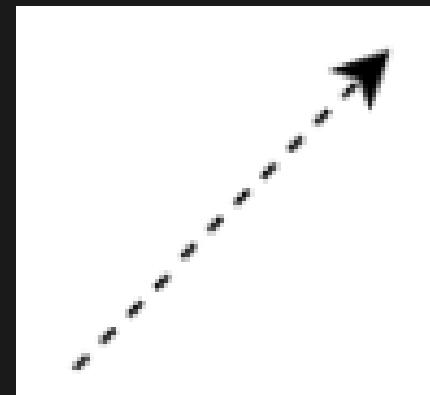
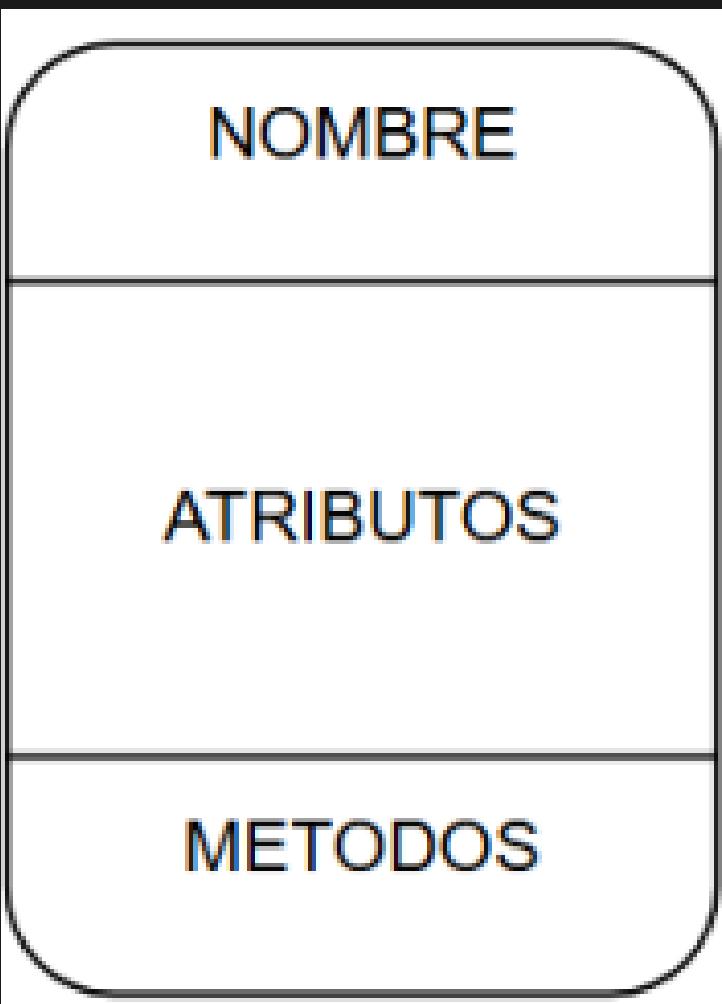


DIAGRAMA DE CLASES

Un diagrama de clases en UML es un tipo de diagrama estructural que representa las clases de un sistema, sus atributos, métodos y relaciones. Es fundamental en el diseño orientado a objetos, ya que permite visualizar la estructura estática del sistema. Estos diagramas están vinculados con otros como los de casos de uso, objetos y comunicación, y sirven para analizar cómo interactúan las clases entre sí. Además, son la base para los diagramas de componentes y despliegue, ayudando a comprender tanto el comportamiento como la organización del software.

ELEMENTOS DE LOS DIAGRAMAS DE CLASES

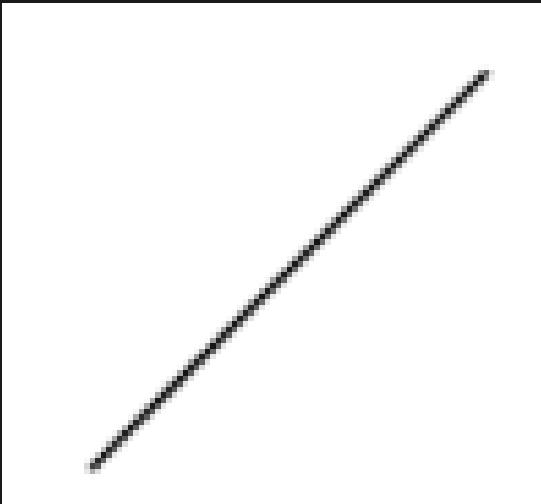
Los diagramas de clases muestran las clases del sistema y sus interacciones, representadas con rectángulos que incluyen el nombre, atributos y métodos. Las relaciones entre clases se indican con líneas y flechas, cada una con un significado específico (como herencia o dependencia). Además, las clases pueden agruparse en paquetes según su funcionalidad.



RELACIONES EN LOS DIAGRAMAS DE CLASE

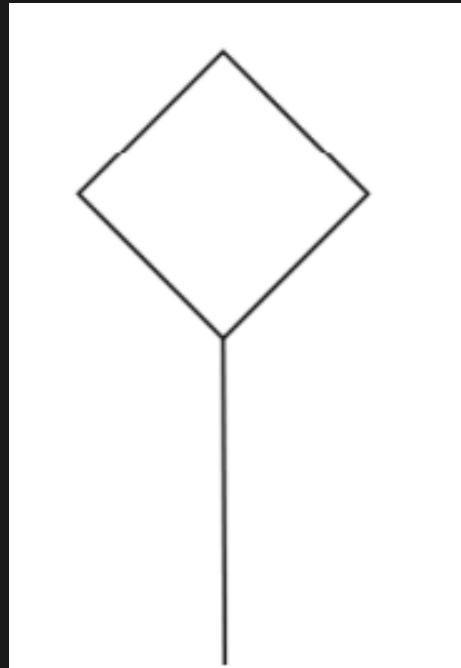
ASOCIACION

Esta relación se representa mediante un línea y se usa para representar la comunicación entre un actor y un caso de uso, entre casos de uso o entre actores.



AGREGACION

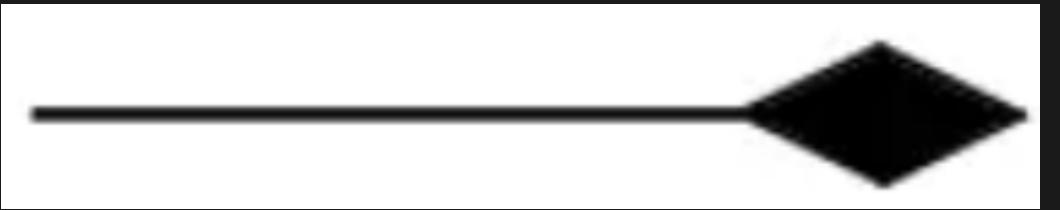
La agregación es una relación entre clases que indica que una clase contiene a otras, pero los objetos contenidos pueden existir de forma independiente. Se representa con una línea y un rombo blanco en el extremo de la clase contenedora.



RELACIONES EN LOS DIAGRAMAS DE CLASE

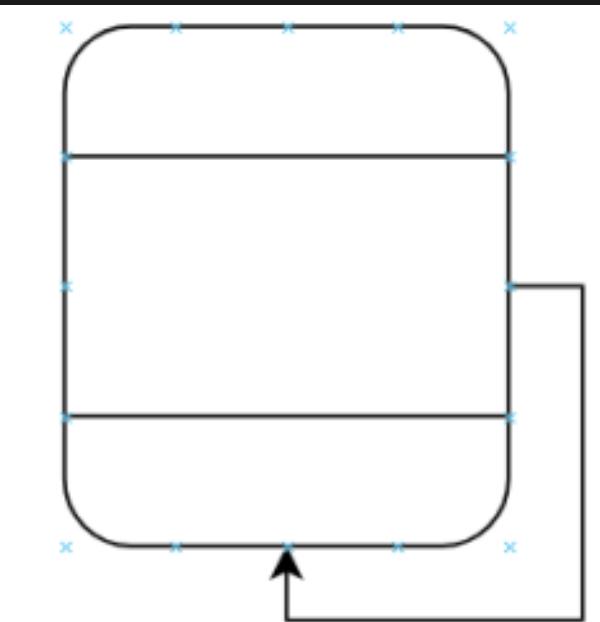
COMPOSICION

La composición es una relación fuerte entre clases, donde una clase forma parte esencial de otra y no puede existir por separado. Se representa con una línea y un rombo negro (relleno) en la clase que contiene. Un ejemplo común es la relación entre Coche y Motor.



ASOCIACION REFLEXIVA

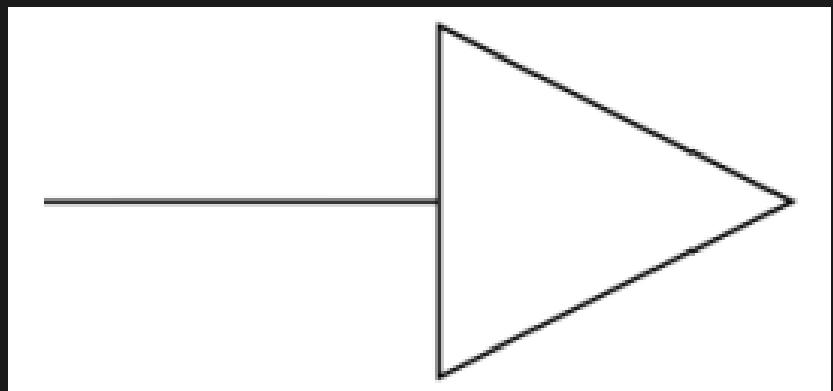
La asociación reflexiva en UML representa una relación entre instancias de una misma clase. Se dibuja como una línea en forma de bucle que sale y vuelve a la misma clase. Es útil cuando un objeto necesita relacionarse con otro objeto del mismo tipo.



RELACIONES EN LOS DIAGRAMAS DE CLASE

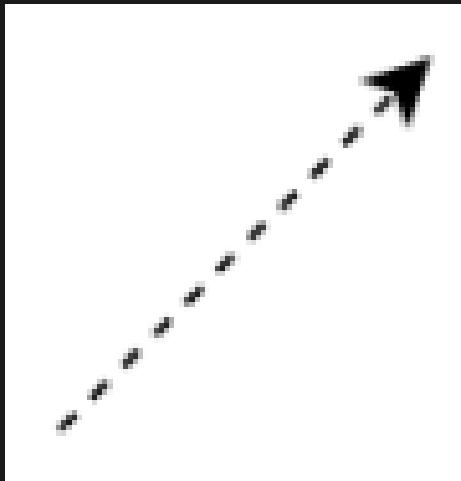
HERENCIA Y GENERALIZACION

La herencia o generalización en UML representa cómo una clase hereda atributos y métodos de otra. Se dibuja con una línea recta y un triángulo hueco apuntando hacia la clase padre o general.



DEPENDENCIA

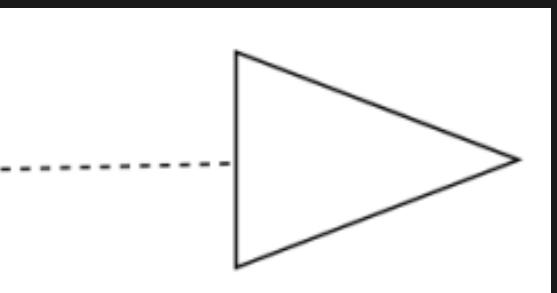
La dependencia en UML se representa con una línea discontinua (escalonada) y una flecha desde la clase dependiente hacia la clase de la que depende. Indica que un cambio en la clase de la que depende puede afectar a la clase dependiente.



RELACIONES EN LOS DIAGRAMAS DE CLASE

REALIZACION

La realización en UML indica que una clase (subclase) implementa el comportamiento definido por otra (superclase o interfaz). Se representa con una línea discontinua (escalonada) y un triángulo hueco apuntando hacia la clase o interfaz que se implementa.



MULTIPLICIDAD

La multiplicidad en UML indica cuántas instancias de una clase pueden estar relacionadas con una instancia de otra clase. Se muestra como un conjunto de números junto a las líneas de relación, especificando el rango o número exacto de objetos vinculados.

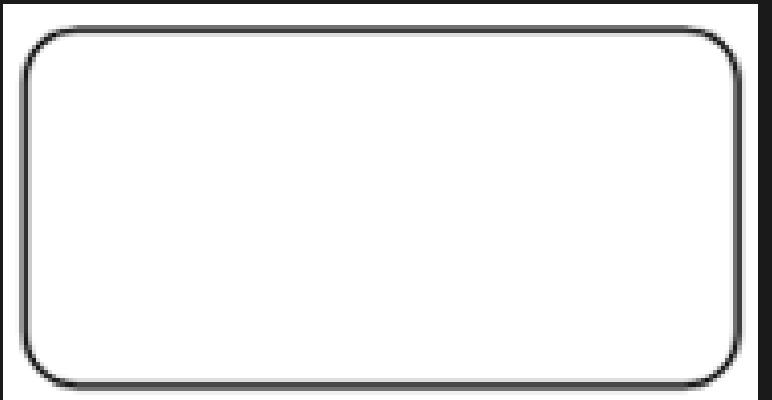
DIAGRAMA DE ACTIVIDADES

Un diagrama de actividades en UML muestra el flujo dinámico de acciones dentro de un sistema o proceso, ayudando a visualizar su comportamiento y lógica. Son útiles para modelar algoritmos, casos de uso, procesos de negocio o flujos de trabajo, y ayudan a planificar o mejorar proyectos identificando ineficiencias. Aunque describen el flujo entre sistemas o actividades, generalmente no incluyen a los usuarios o actores, a diferencia de los diagramas de casos de uso, que sí se centran en ellos.

ELEMENTOS DE LOS DIAGRAMAS DE ACTIVIDADES

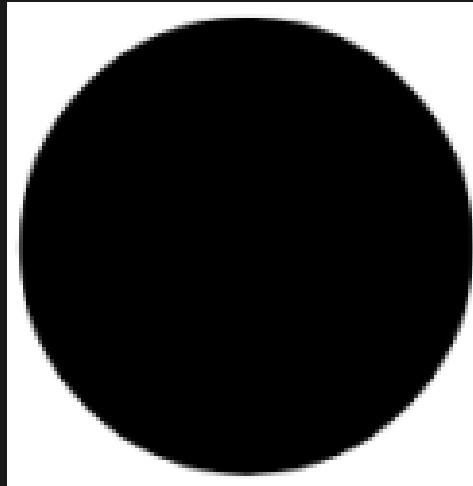
ACCION

Este elemento representa un paso dentro de una actividad. Las acciones suelen representarse con rectángulos con esquinas redondeadas.



NODO DE INICIO

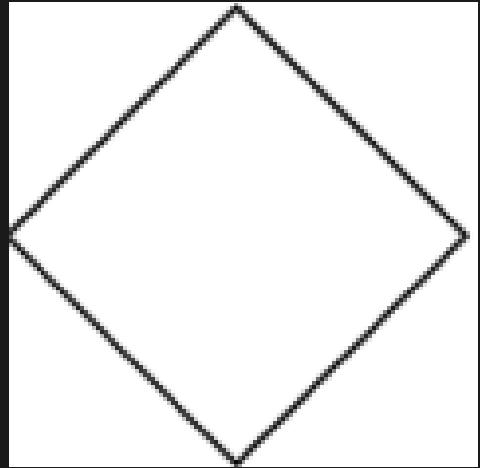
Esto es el punto de partida o el evento desencadenante de una actividad. Este se representa por un círculo negro sólido.



ELEMENTOS DE LOS DIAGRAMAS DE ACTIVIDADES

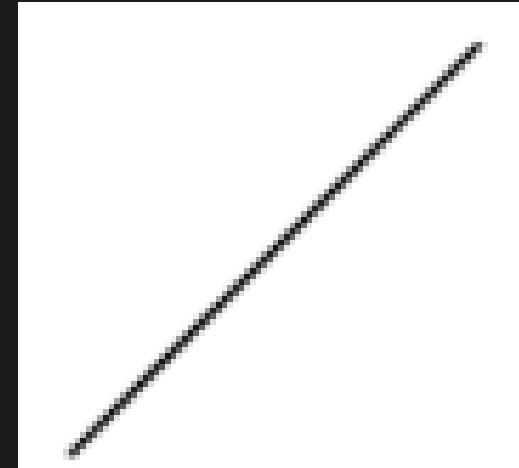
NODO DE CONTROL

Este elemento controla el flujo entre otros nodos. Se representa mediante un diamante con un flujo de entrada y dos o mas flujos de salida



FLUJOS DE CONTROL

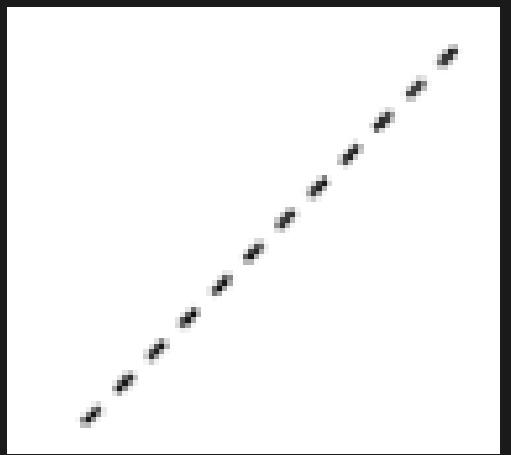
Estos también se conocen como bordes de control. Representa el flujo de control de un elemento a otro, con una línea sólida.



ELEMENTOS DE LOS DIAGRAMAS DE ACTIVIDADES

FLUJOS DE OBJETOS

También se conocen como bordes de objeto. Estos representan el flujo dirigido de objetos de un elemento a otro y se representan mediante una línea punteada.



PARTICION DE ACTIVIDAD

Es una columna o fila que se utiliza para mostrar áreas de responsabilidad para diferentes actores. Estos también se conocen como diagramas de carriles



FIN