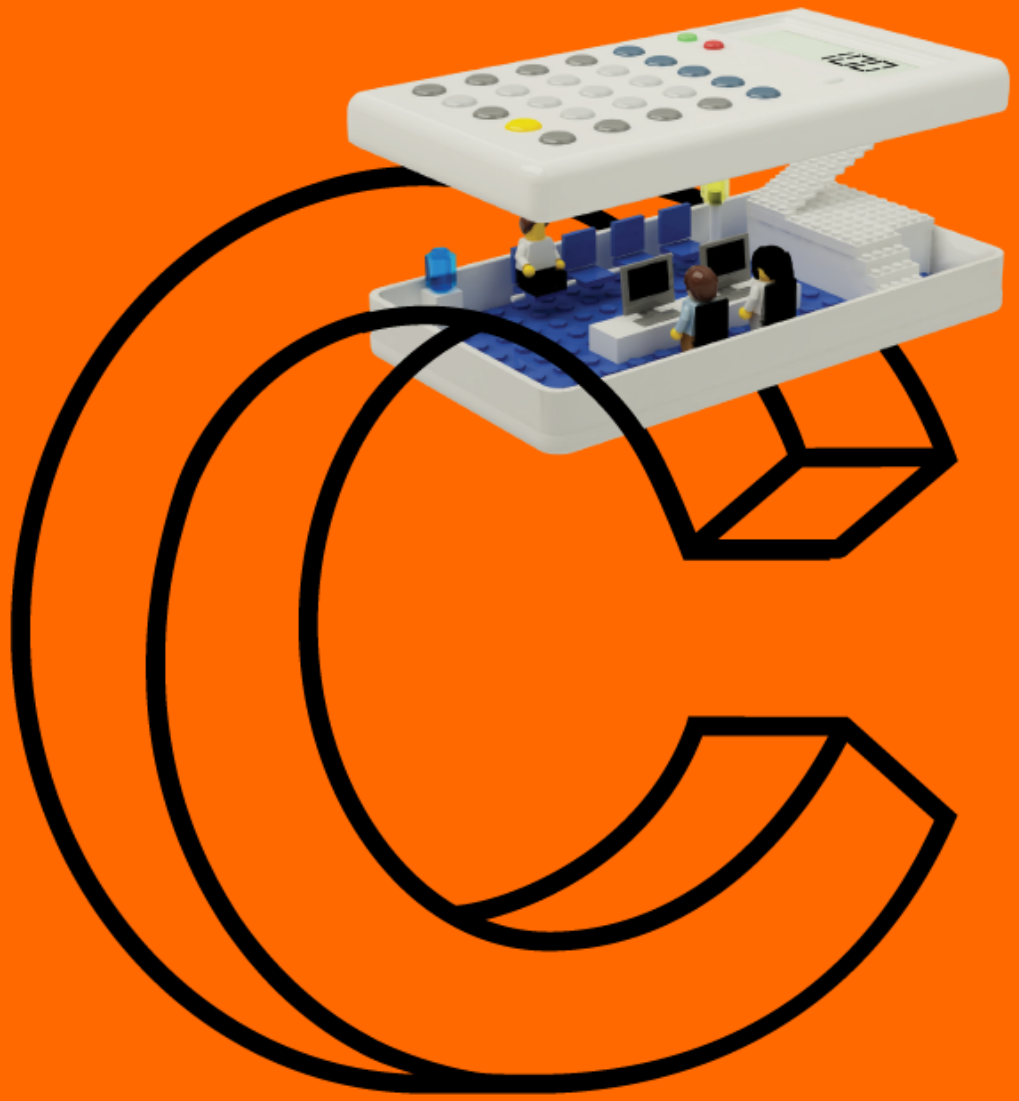
Análisis y Diseño de Sistemas Orientado a Objetos

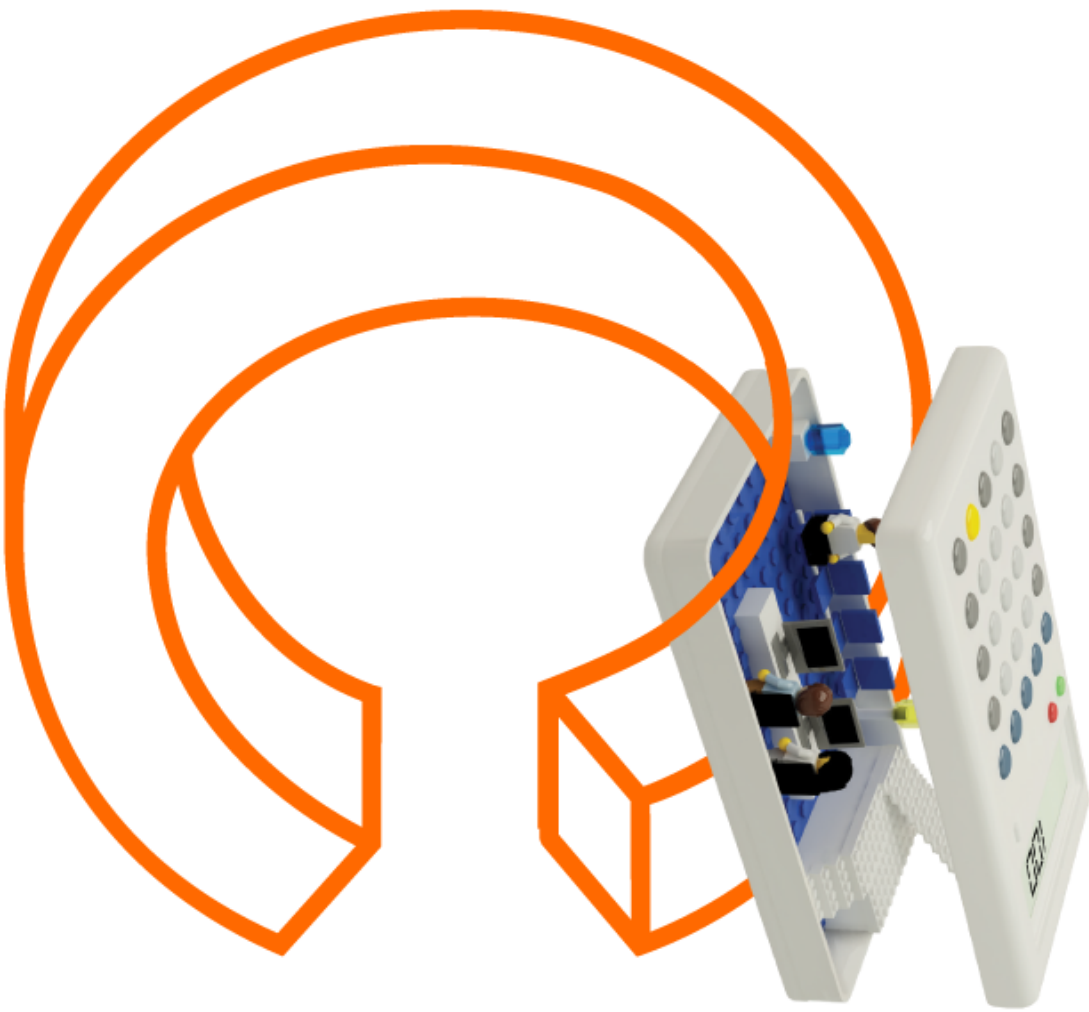


idat

Mg. Daniel Arias, **PMP™** y **Scaled Scrum Master™**

Tecnología Orientada a Objetos

Semana 1



idat

Semana 1: Tecnología Orientada a Objetos

Logro de aprendizaje

- Definición
- Lenguajes de Programación Orientados a Objetos
- Metodología Proceso Unificado Rational – RUP
- Lenguaje de Modelado Unificado – UML

Actividades

Actividad 01:

- Elabora y presenta un Cuadro Comparativo de los Lenguajes de Programación Orientado a Objetos

- Guía ADSOO-01- Diagramas UML

Nos conocemos

Daniel Arias

- Ingeniero Informático de la Universidad Ricardo Palma
- Magister en Dirección de Sistemas y Tecnología de la Información de la UPC
- Máster Ejecutivo en Dirección de Sistemas y Tecnologías de Información de la Universidad Politécnica de Cataluña
- Project Manager Professional (PMP)
- Scaled Scrum Master (SSM)

- Jefe de Proyectos Informáticos en la SUNAT
- Profesor a tiempo parcial en IDAT
- Ex Gerente de Proyectos en Arkin Software Technologies
- Ex Jefe de Operaciones de la Bolsa de Valores de Lima
- Ex Jefe de Desarrollo de Aplicaciones de la Bolsa de Valores de Lima

Sistema de Evaluación

Sesión	Evaluación	%
02 y 03	Participación en clase (máx 1 puntos)	
04	**Primera Evaluación Continua**	**4%**
05 y 06	Participación en clase (máx 1 puntos)	
07	**Segunda Evaluación Continua**	**12%**
08 y 09	Participación en clase (máx 1 puntos)	
10	**Tercera Evaluación Continua**	**24%**
11, 12 y 13	Participación en clase (máx 1 puntos)	
14	**Evaluación Final**	**60%**

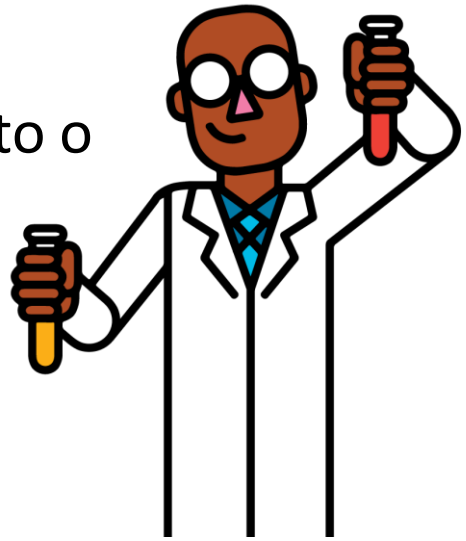
http://www.menti.com

Logro del curso

El alumno, es capaz de realizar el análisis y diseño durante el desarrollo de software utilizando el paradigma Orientado a Objetos, incluyendo técnicas y flujos de trabajo.

Aprendemos - Tecnología Orientada a Objetos

Las organizaciones han reconocido la importancia de administrar recursos clave como las personas y la materia prima. Los responsables de la toma de decisiones por fin comprenden que la información no sólo deriva de las operaciones comerciales, sino que provee impulso a las empresas y puede constituir el factor decisivo para determinar el éxito o el fracaso de un negocio.



Para maximizar la utilidad de la información, una empresa debe administrarla en forma apropiada, de la misma manera en que administra los demás recursos y que existen costos asociados con la producción, distribución, seguridad, el almacenamiento y la recuperación de toda información. Aunque la información está a nuestro alrededor, no es gratuita.

Aprendemos - Tecnología Orientada a Objetos

¿Qué es la tecnología orientada a objetos?

Un conjunto de principios (abstracción, encapsulación polimorfismo) que guían la construcción de software, junto con lenguajes, bases de datos y otras herramientas que soportan estos principios.

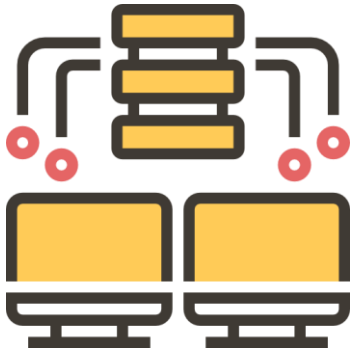
Fortalezas de la tecnología de objetos:

- Reflejar un paradigma de modelaje sencillo
- Facilitar la reutilización de elementos arquitectónicos y código
- Crear modelos más apegados a la realidad
- Mantener estabilidad
- Adaptarse a cambios en el dominio del problema

Aprendemos - Tecnología Orientada a Objetos

¿Dónde se utiliza la tecnología Orientado a Objetos?

- **Sistemas cliente-servidor y desarrollos WEB.**
  La tecnología de objetos permite a las empresas encapsular información de negocio en objetos reutilizables que pueden distribuirse a través de un ambiente de red-Internet para mejorar el procesamiento de transacciones.

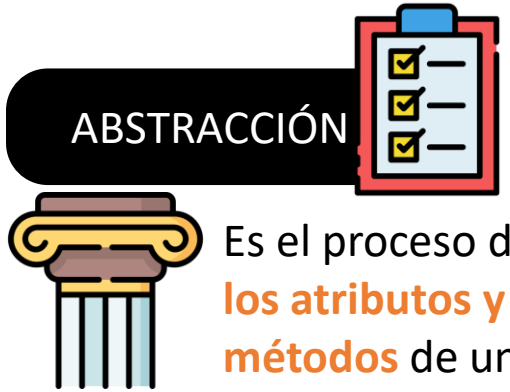


- **Sistemas de tiempo real**
  La tecnología de objetos permite desarrollar sistemas de tiempo real con alta calidad y flexibilidad

Aprendemos - Tecnología Orientada a Objetos

Objetivos

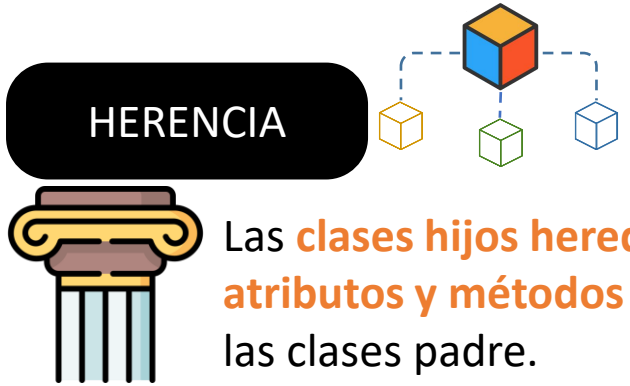
- Describir abstracción, encapsulación, polimorfismo y herencia.



Es el proceso de **definir los atributos y los métodos** de una clase.

Da la misma orden a varios objetos para que respondan de diferentes maneras.

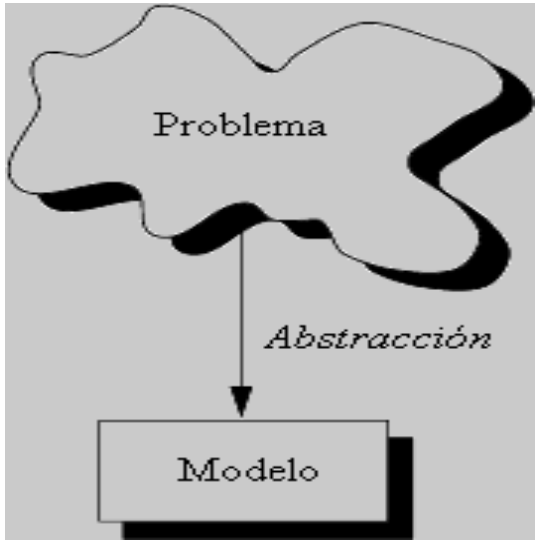
Protege la información de manipulaciones no autorizadas.

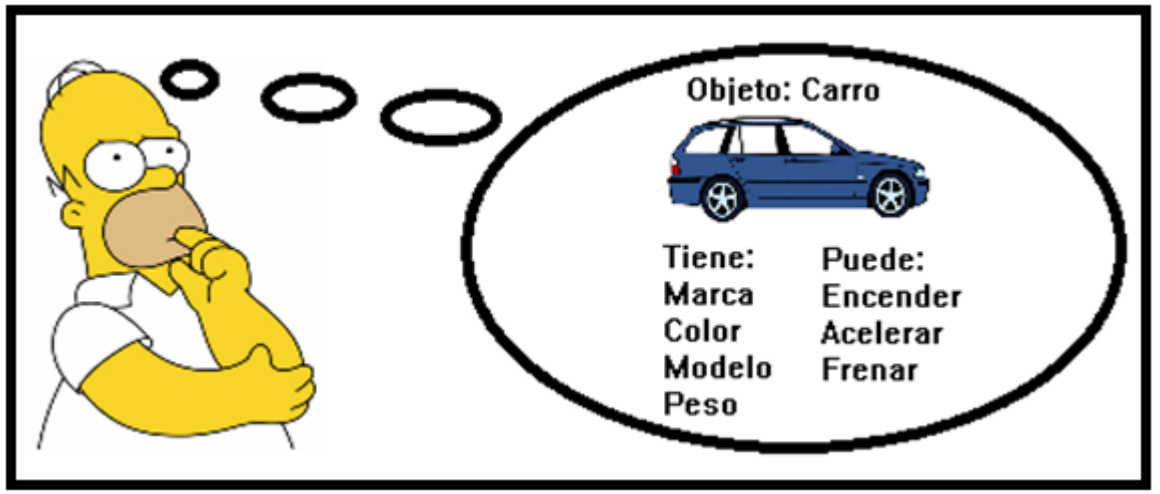


Las **clases hijos heredan atributos y métodos** de las clases padre.

Abstracción

- El modelo define una perspectiva abstracta del problema
  - Los datos que son afectados
  - Las operaciones que se aplican sobre los datos

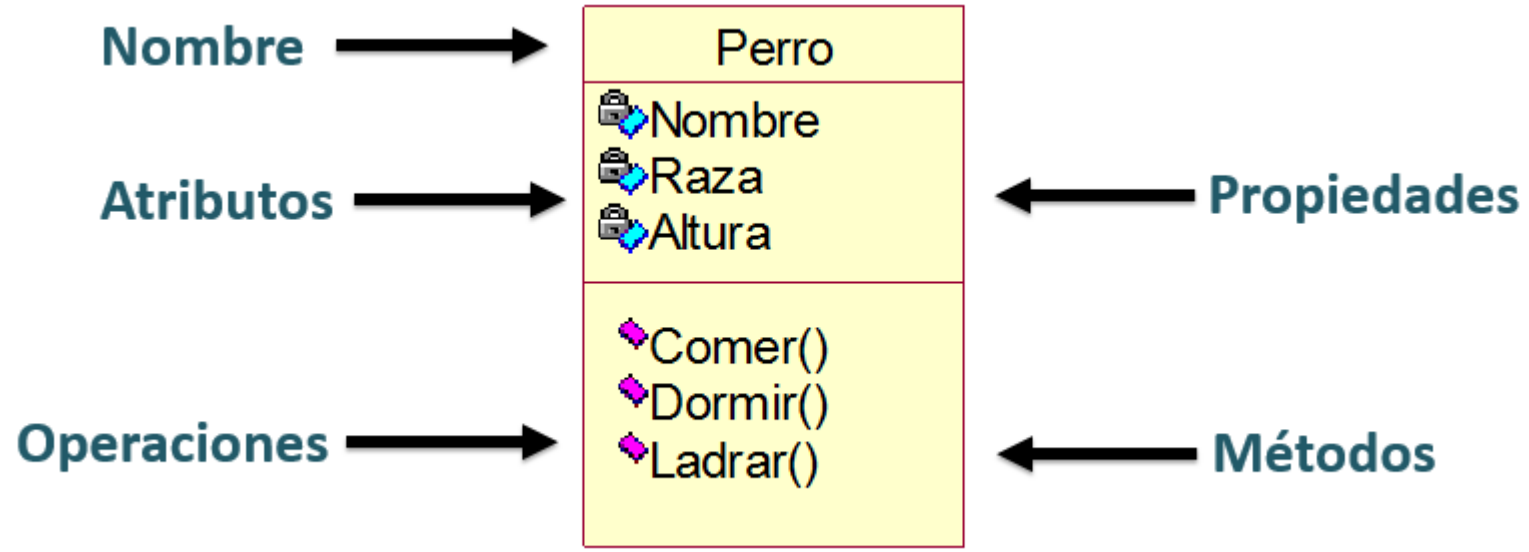




Aprendemos - Tecnología Orientada a Objetos

Objetivos

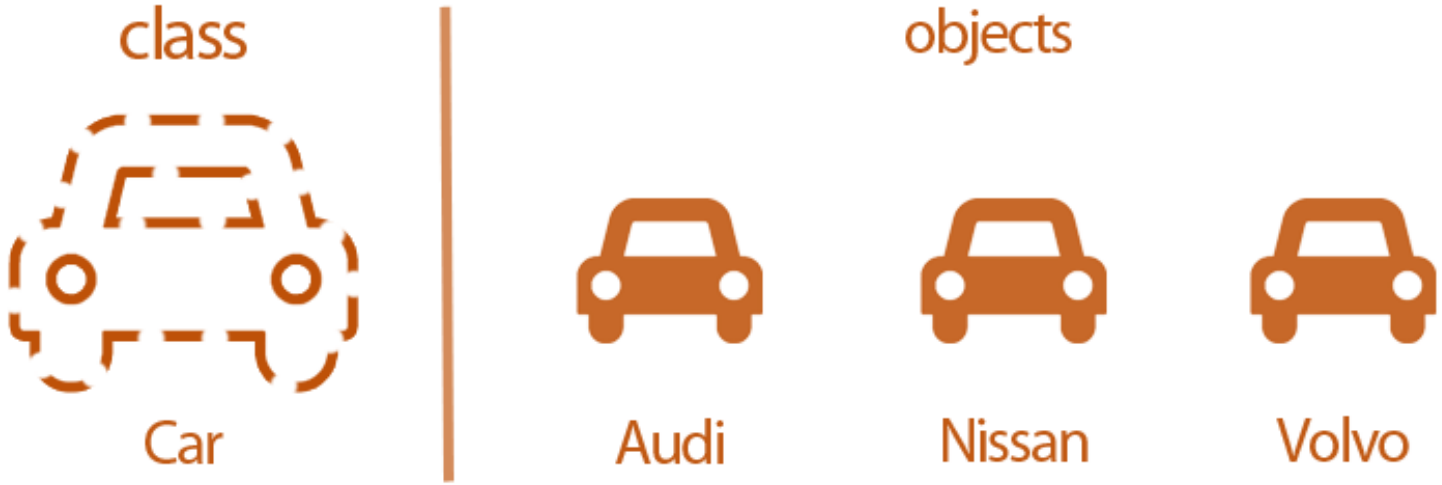
- Describir la estructura física de una clase.
  - Una clase describe un grupo de objetos que comparten propiedades y métodos comunes
  - Una clase es una plantilla que define qué forma tienen los objetos de la clase
  - Una clase se compone de:
    - Información: campos (atributos, propiedades)
    - Comportamiento: métodos (operaciones, funciones)



Aprendemos - Tecnología Orientada a Objetos

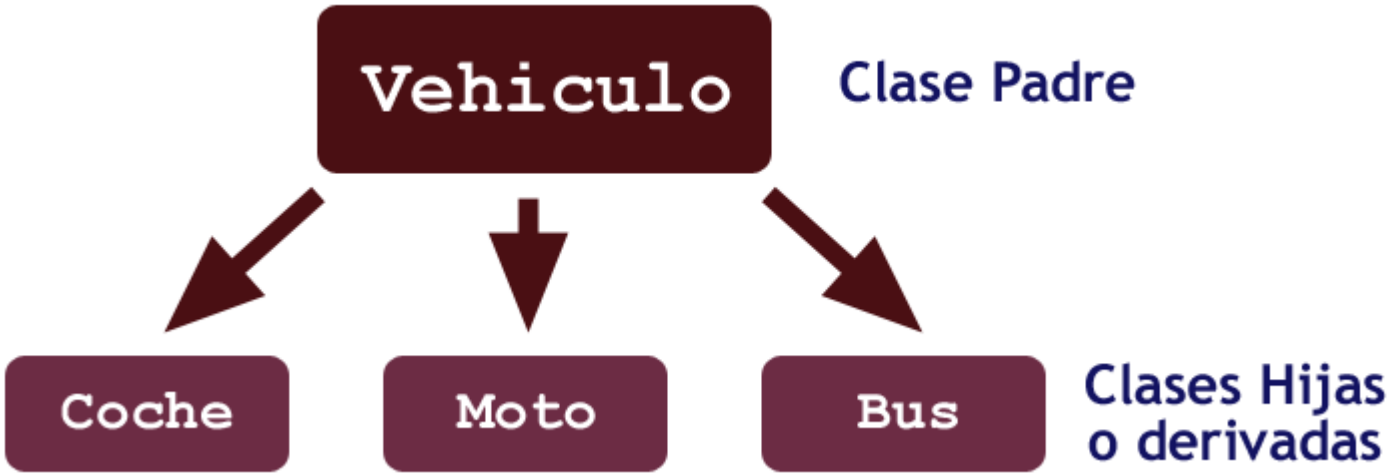
Objetivos

- Describir la relación entre una clase y un objeto.
  - Un objeto es una instancia de una clase



Objetivos

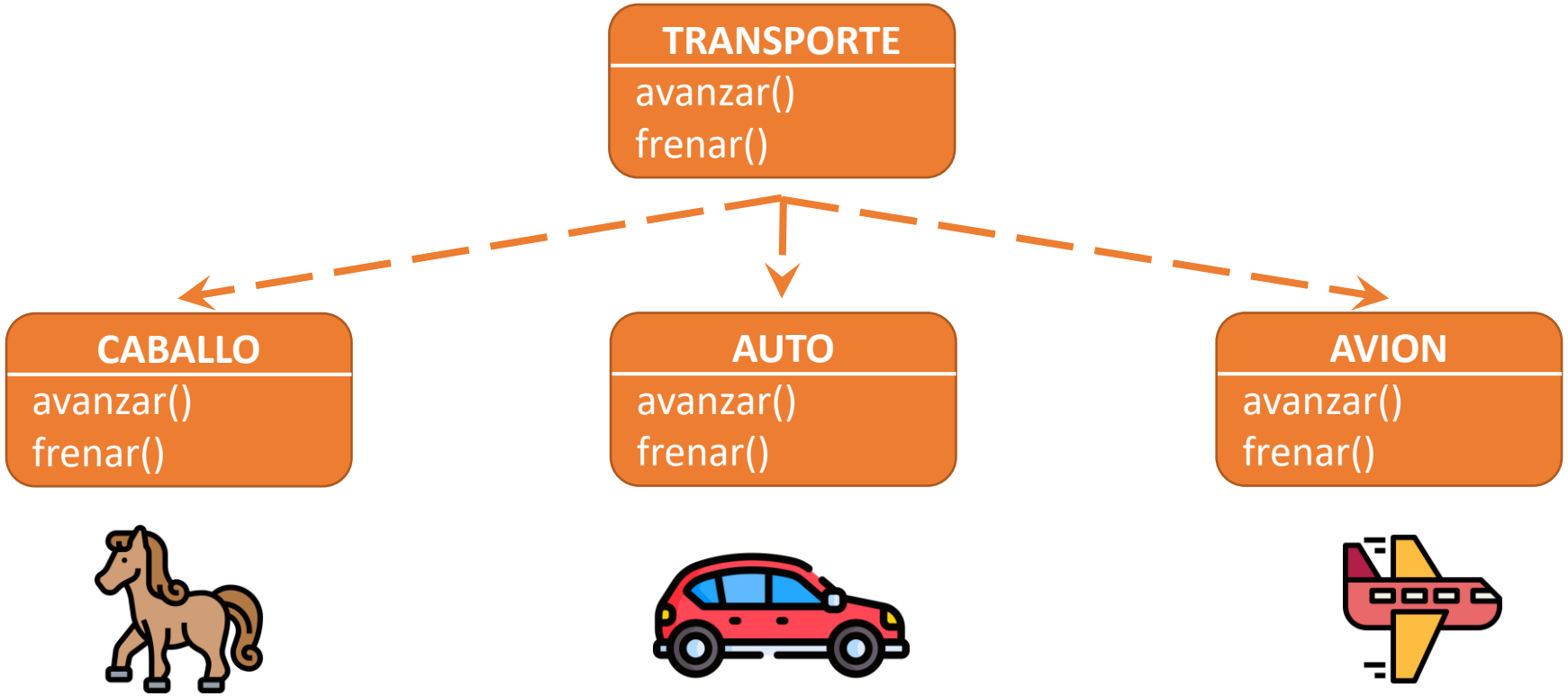
- Definir polimorfismo y generalización.
  - Herencia: es una relación entre clases donde una comparte la estructura o el comportamiento definido en otra(s) clase(s).



Aprendemos - Tecnología Orientada a Objetos

Objetivos

- Definir polimorfismo y generalización.



Aprendemos - Tecnología Orientada a Objetos

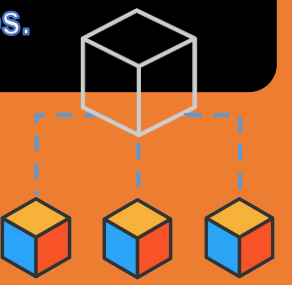
Realidades de la TOO:

- Mejor capacidad de modelamiento:
  - Códigos + datos: Entidades abstractas existentes.
  - Punteros-relación entre entidades (mensajes).
- Menor complejidad:
  - Ocultamiento de información: Simplicidad de uso y de cambio.
  - Polimorfismo: Interfaz común para muchos objetos.
- Facilidad de desarrollo:
  - Representación natural de la interfaz.
  - Ocultamiento de las APIs del sistema.

Observamos y respondemos:

¿QUÉ ES LA PROGRAMACIÓN ORIENTADA A OBJETOS?

Es un paradigma de programación que organiza las funciones en entidades llamadas objetos.

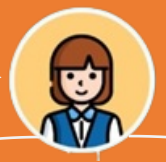


- **Los objetos se crean a partir de una plantilla llamada clase.**
  Cada objeto es una Instancia de su clase.

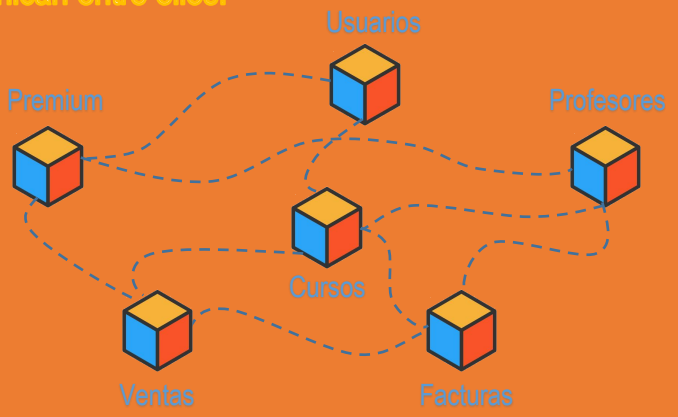
CLASE

INSTANCIACIÓN

OBJETO



- En una aplicación los objetos están separados, **pero se comunican entre ellos.**



ATRIBUTOS

- Nombres
- Apellidos
- Correo
- Contraseña
- Premium

METODOS

- Editar perfil
- Iniciar sesión
- Cerrar sesión
- Cambiar contraseña
- Pasar a premium

- **Los objetos tienen** datos (atributos) **y funcionalidades (métodos).**

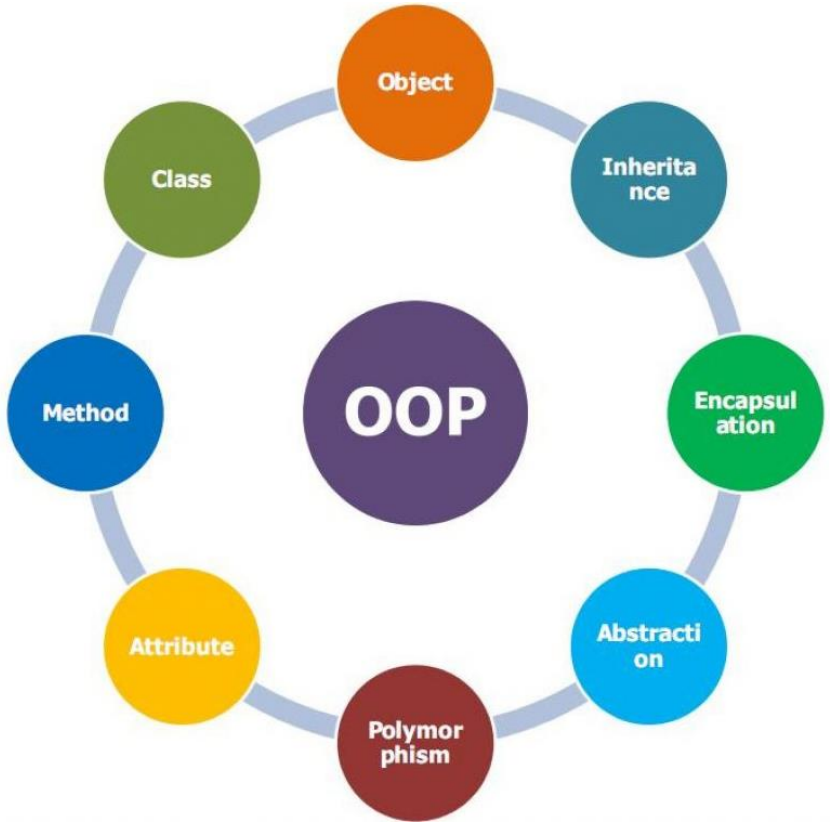
Puedes programar con este paradigma en la mayoría de lenguajes.

Observamos y respondemos:

OOP

- En Object-Oriented Programming (OOP), o Programación Orientada a Objetos (POO), un programa está hecho de clases, con sus campos y métodos

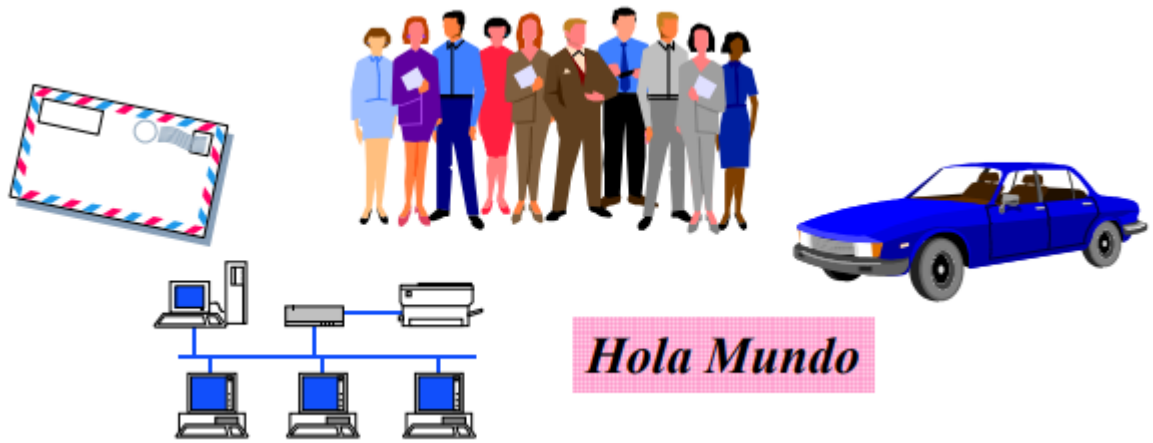
- La creación de un programa involucra ensamblar objetos y hacerlos interactuar entre ellos



Aprendemos - Tecnología Orientada a Objetos

Objetos

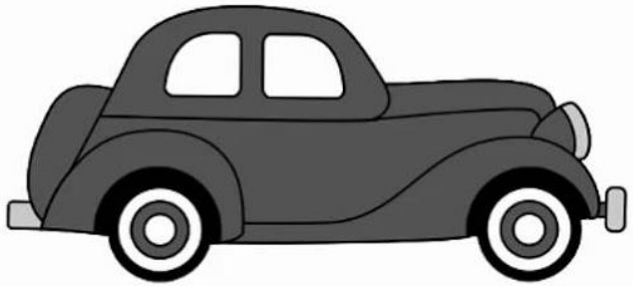
- Los objetos son/representan cosas
- Los objetos pueden ser simples o complejos
- Los objetos pueden ser reales o imaginarios



Atributos

- Valores o características de los objetos
- Permiten definir el estado del objeto u otras cualidades

Properties
Make
Model
Color
Year
Price



Events
On_Start
On_Parked
On_Brake

Methods
Start
Drive
Park

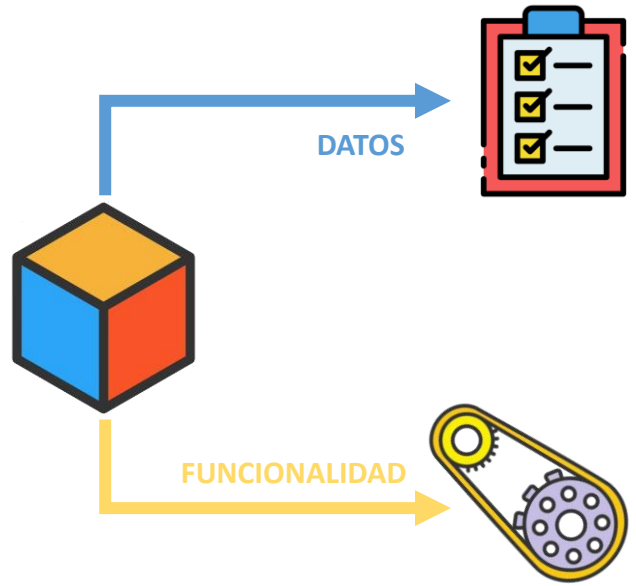
Aprendemos - Tecnología Orientada a Objetos

¿Que tienen que ver los objetos con la programación?

Es el paradigma más usado en el mundo. Cada uno de los elementos de la interfaz es un objeto y cada objeto tiene datos y funcionalidad.

Los zapatos son un objeto y por lo tanto tiene estas dos propiedades:

- Datos: Nombre, marca, color, talla, etc.
- Funcionalidad: Pueden ser agregados al carrito y pueden ser comprados.



Los usuarios también son un objeto:

- Datos: Nombre, número de tarjeta de crédito, etc.
- Funcionalidad: Pueden comprar zapatos.

Aprendemos - Tecnología Orientada a Objetos

¿Que tienen que ver los objetos con la programación?

Con la POO pasamos de tener un código de arriba hacia abajo en el que las funcionalidades están mezcladas y son difíciles de separar o escalar, a un sistema en el que tenemos los elementos (Objetos) separados y se comunican entre ellos:

- El usuario se comunica con el producto para comprarlo.
- El producto se comunica con el carrito para ser agregado.
- El carrito se comunica con la pasarela de pago y con el usuario.

De esta manera es más fácil manejar y mantener un sistema, si necesitáramos una nueva funcionalidad podríamos sin problemas agregar un nuevo objeto o añadir datos y funcionalidades a los objetos que ya existen.

Mensajes

- Los objetos se comunican e interaccionen entre sí por medio de mensajes
- Si un objeto desea que otro objeto haga algo le envía un mensaje que puede tener información adicional en forma de parámetros
- Cuando un objeto recibe un mensaje ejecutará un método u operación
- Componentes de un mensaje
    - Objeto destinatario del mensaje (miAuto)
    - Método que se debe ejecutar como respuesta (cambiar marcha)
    - Parámetros necesarios del método (segunda)

Clase: auto

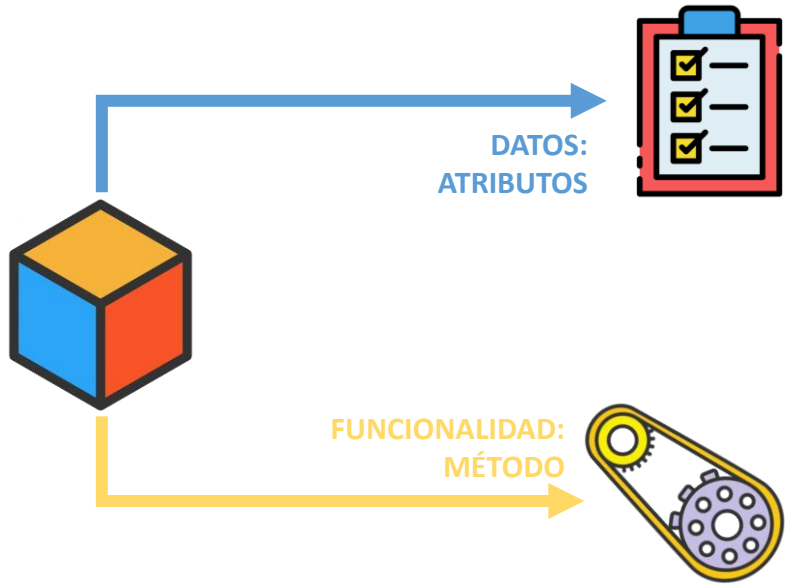
Objeto: auto_amarillo

Elementos de la Programación Orientada a Objetos

Atributos y métodos

Los objetos tienen datos y funcionalidad y en la POO se les llama de esta manera:

Datos → Atributos
Funcionalidad → Método

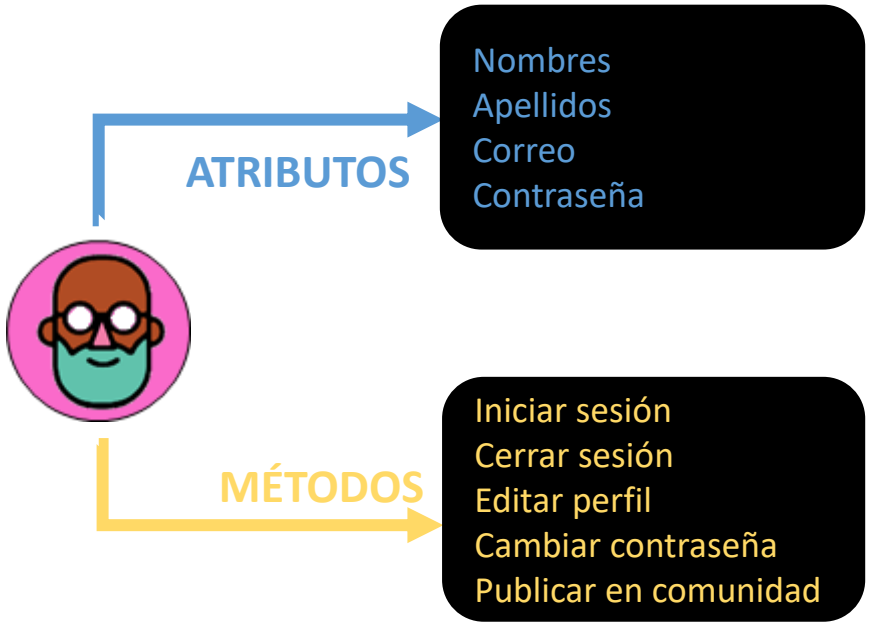


Elementos de la Programación Orientada a Objetos

Abstracción

En palabras simples la abstracción es pensar ¿Cuáles son los atributos y métodos que necesita un objeto? Imagina que estamos programando la aplicación de cursos y queremos crear el objeto usuarios, ¿Que atributos y métodos necesitamos?

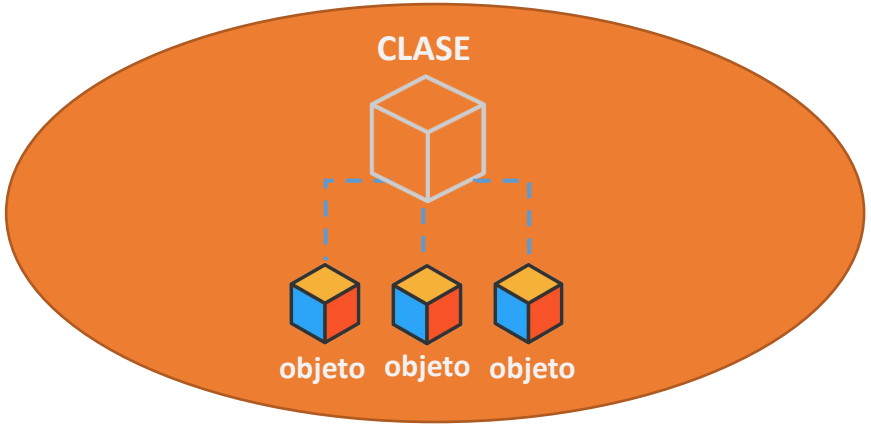
Este es el resultado de nuestro proceso de abstracción



Elementos de la Programación Orientada a Objetos

Clase

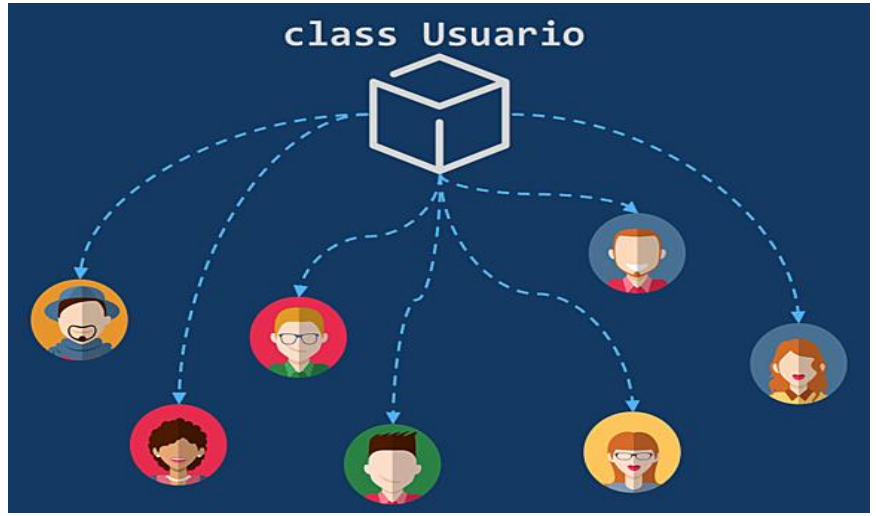
Las clases son plantillas que contienen la estructura básica de un objeto (Atributos y Métodos). En el ejemplo anterior donde realizamos el proceso de abstracción no creamos un objeto, en realidad creamos la plantilla en la que se basaran los objetos. Cada vez que alguien se registra usará la clase (plantilla) para crear el objeto usuario.



Elementos de la Programación Orientada a Objetos

Instancia

El proceso de crear un objeto a partir de una clase se llama Instanciar y cada objeto creado también se le puede llamar Instancia. Con una sola clase podemos crear cientos de usuarios (Instancias) sin tener que escribir código para cada uno de ellos.



Aprendemos - Metodología Proceso Unificado Rational – RUP

Definición

El Proceso Unificado es un proceso de desarrollo de software: “conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema de software”.

RUP es un marco genérico que puede especializarse para una variedad de tipos de sistemas, diferentes áreas de aplicación, tipos de organizaciones, niveles de aptitud y diferentes tamaños de proyectos.

RUP está basado en componentes. El sw está formado por componentes software interconectados a través de interfaces.

RUP está dirigido por **casos de uso**, **centrado en la arquitectura**, y es **iterativo e incremental**.

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo a través del UML, y trabajo de muchas metodologías utilizadas por los clientes

Aprendemos - Metodología Proceso Unificado Rational – RUP

Características esenciales de RUP

- Proceso dirigido por los CU: El proceso utiliza casos de uso para manejar el proceso de desarrollo desde la Incepción hasta el Despliegue.
- Proceso Iterativo e Incremental: El proceso reconoce que es práctico dividir grandes proyectos en proyectos más pequeños o mini proyectos. Cada mini proyecto comprende una iteración que resulta en un incremento. Una iteración puede abarcar la totalidad de los flujos del proceso. Las iteraciones son planificadas con base a los casos de uso.
- Proceso Centrado en la Arquitectura: El proceso busca entender los aspectos estáticos y dinámicos más significativos en términos de arquitectura de software. La arquitectura se define en función de las necesidades de los usuarios y se determina a partir de los Casos de Uso base del negocio.

Aprendemos - Metodología Proceso Unificado Rational – RUP

Fases del Ciclo de Vida

El ciclo de vida consiste en una serie de ciclos, cada una de las cuales produce una nueva versión del producto. Cada ciclo está compuesto por fases y cada una de estas fases está compuesta por un número de iteraciones

Aprendemos - Metodología Proceso Unificado Rational – RUP

1ra Fase: Iniciación o Estudio de oportunidad

Se concibe la idea central del producto, se arma el documento de visión. En esta fase, se **revisa** y **confirma** nuestro entendimiento sobre los **objetivos centrales del negocio**. Queremos entender los argumentos comerciales en favor de porqué el proyecto debe intentarse. La fase de incepción establece la **viabilidad del producto** y delimita el alcance del proyecto:

- Define el ámbito y objetivos del proyecto
- Se define la funcionalidad y capacidades del producto

Durante la fase de inicio se desarrolla una descripción del producto final y se presenta el análisis del negocio. Esta fase responde las siguientes preguntas:

- ¿Cuáles son las principales funciones del sistema para los usuarios más importantes?
- ¿Cómo podría ser la mejor arquitectura del sistema?
- ¿Cuál es el plan del proyecto y cuánto costará desarrollar el producto?

Aprendemos - Metodología Proceso Unificado Rational – RUP

2da Fase: Elaboración

Durante la fase de elaboración la mayoría de los CU son especificados en detalle y la arquitectura del sistema es diseñada. Esta fase se focaliza en las “habilidades” del proyecto. Se identifican los riesgos significativos y se preparan el calendario, el equipo de trabajo y el costo del proyecto.

- Tanto la funcionalidad como el dominio del problema se estudian en profundidad
- Se define una arquitectura básica
- Se planifica el proyecto considerando recursos disponibles

Las iteraciones en la fase de elaboración:

- Establecen una firme comprensión del problema a solucionar.
- Establece la fundación arquitectural para el software.
- Establece un plan detallado para las siguientes iteraciones.
- Elimina los mayores riesgos.

Aprendemos - Metodología Proceso Unificado Rational – RUP

3ra Fase: Desarrollo o Construcción

Durante la fase de construcción, el foco del producto se mueve de la arquitectura de base a un sistema lo suficientemente completo como para llevarlo al usuario. La línea base de arquitectura crece en complejidad y se convierte en un sistema completo, de la misma manera, se refina el diseño para llevarlo a código fuente.

- El producto se desarrolla a través de iteraciones donde cada iteración involucra tareas de análisis, diseño e implementación.
- Las fases de estudio y análisis sólo dieron una arquitectura básica que es aquí refinada de manera incremental conforme se construye (se permiten cambios en la estructura).
- Gran parte del trabajo es programación y pruebas.
- Se documenta tanto el sistema construido como el manejo del mismo.
- Esta fase proporciona un producto construido junto con la documentación.

Aprendemos - Metodología Proceso Unificado Rational – RUP

4ta Fase: Transición

En la fase de transición el objetivo es **garantizar que los requisitos se han cumplido**, con la satisfacción de las partes interesadas. Esta fase a menudo se inicia con una versión beta de la aplicación. Otras actividades incluyen la preparación del ambiente, se completan, se identifican y corrigen defectos. La fase de transición termina con un cierre dedicado al aprendizaje de lecciones, las cuales quedan para futuros ciclos.

- Se libera el producto y se entrega al usuario para un uso real.
- Se incluyen tareas de marketing, empaquetado atractivo, instalación, configuración, entrenamiento, soporte, mantenimiento, etc.
- Los manuales de usuario se completan y refinan con la información anterior.
- Estas tareas se realizan también en iteraciones.

La fase de transición cubre el período durante el cual el producto se convierte en la versión beta.

Aprendemos - Metodología Proceso Unificado Rational – RUP

Componentes del RUP

- Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

Aprendemos - Metodología Proceso Unificado Rational – RUP

Flujos de trabajo:

- Modelamiento del negocio: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- Requerimientos: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- Análisis y diseño: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- Implementación: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

Aprendemos - Metodología Proceso Unificado Rational – RUP

Flujos de trabajo:

- Prueba (Testeo): Busca los defectos a lo largo del ciclo de vida.
- Instalación: Realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- Administración del proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Administración de configuración y cambios: Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a la utilización/actualización concurrente de elementos, control de versiones, etc.
- Ambiente: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto, así como el procedimiento para implementar el proceso en una organización.

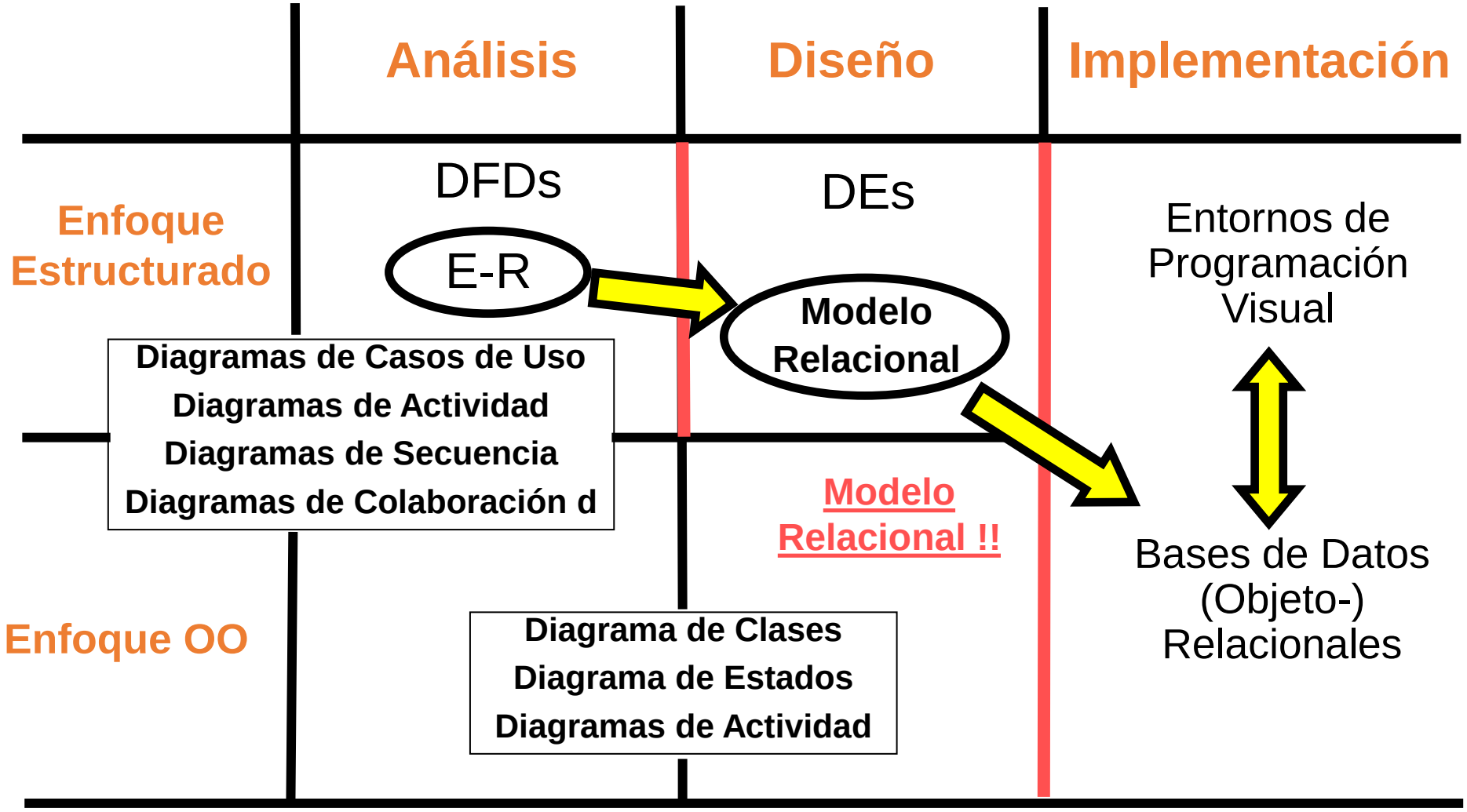
Aprendemos - Metodología Proceso Unificado Rational – RUP

Artefactos de Gestión y técnicos

RUP en cada una de sus fases (pertenecientes a la estructura estática) realiza una serie de artefactos que sirven para comprender mejor tanto el análisis como el diseño del sistema, estos artefactos son los siguientes:

Inicio	Elaboración	Construcción
• Documento Visión • Especificación de Requerimientos	• Diagramas de CU	Vista Lógica • Diagrama de clases • Modelo E-R (si el sistema así lo requiere) Vista de implementación • Diagrama de secuencia • Diagrama de estados • Diagrama de colaboración Vista conceptual • Modelo de dominio Vista Física • Mapa de comportamiento a nivel de hardware.

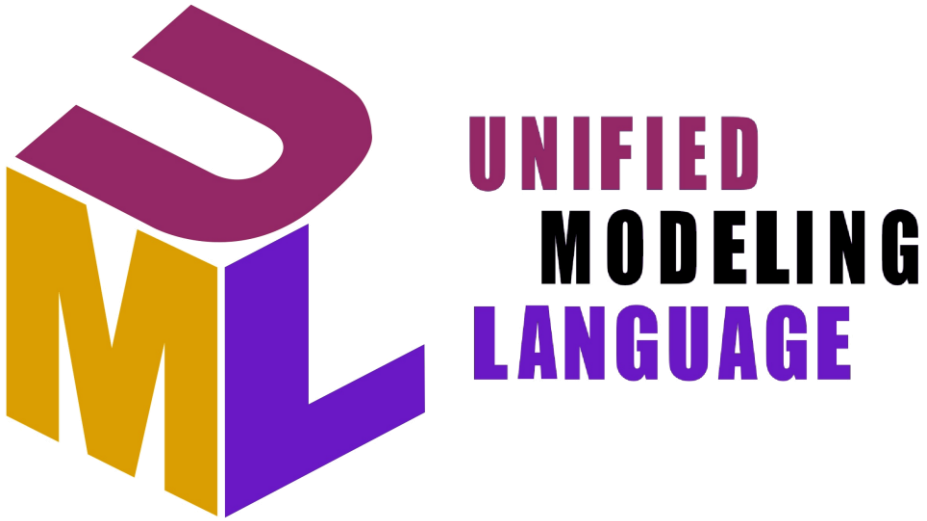
Aprendemos - Metodología Proceso Unificado Rational – RUP



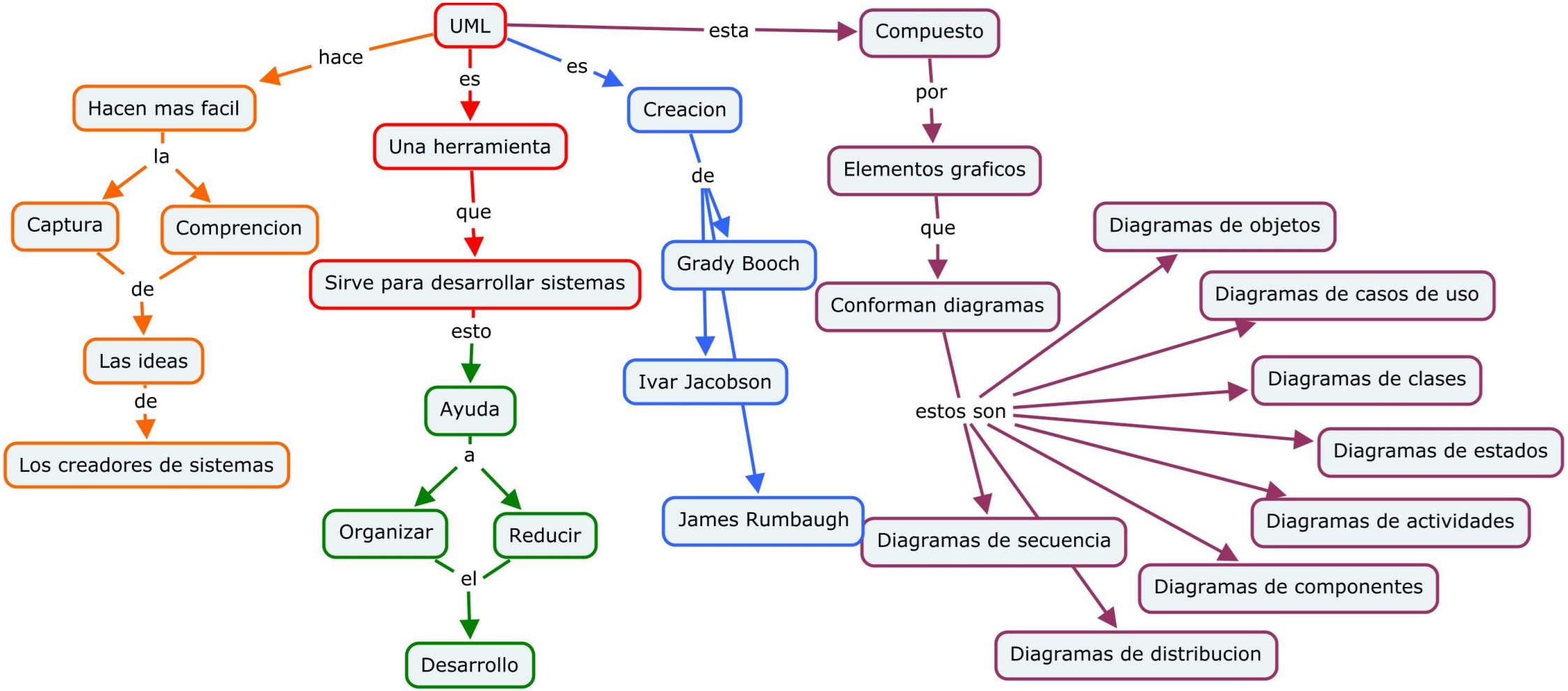
Aprendemos - UML

¿Qué es UML?

- UML = Unified Modeling Language
- Un lenguaje de propósito general para el modelado orientado a objetos
- Documento “OMG Unified Modeling Language Specification”
- UML combina notaciones provenientes desde:
    - Modelado Orientado a Objetos
    - Modelado de Datos
    - Modelado de Componentes
    - Modelado de Flujos de Trabajo (Workflows)



Aprendemos - UML



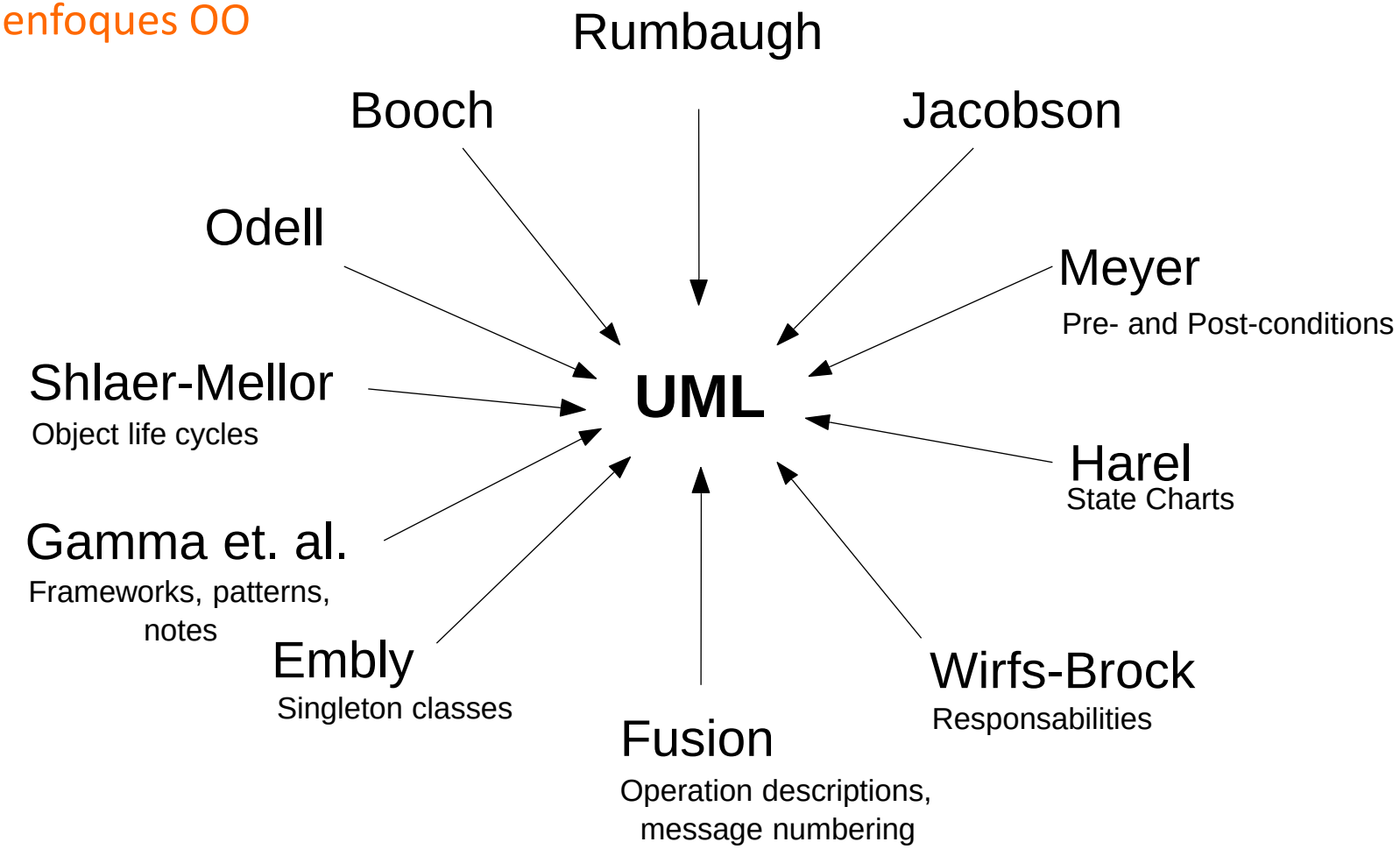
Aprendemos - UML

Motivación

- Diversos métodos y técnicas OO, con muchos aspectos en común, pero utilizando distintas notaciones
- Inconvenientes para el aprendizaje, aplicación, construcción y uso de herramientas, etc.
- Pugna entre distintos enfoques (y correspondientes gurús)
- Establecer una notación estándar

Aprendemos - UML

UML “aglutina” enfoques OO



Aprendemos - UML

La importancia de Modelar

- Ayuda a visualizar un sistema como deseamos que sea diseñado. Un modelo ayuda al equipo de trabajo a comunicar la visión del sistema que se está construyendo. Es muy difícil compartir una misma visión si solo se cuenta con especificaciones textuales.
- Permite especificar la estructura y conducta del sistema. Un modelo permite documentar la estructura y conducta de un sistema antes de que sea codificado.
- Brinda una plantilla que guía el proceso de construcción. Un modelo es una herramienta invaluable durante la construcción, sirve como guía para el programador. ¿Alguna vez ha tenido la experiencia de que un programador codifique la funcionalidad incorrecta debido a que confundió la descripción textual de un requerimiento? El modelado ayuda a aliviar esta situación.
- Documenta las decisiones que hemos tomado. Los modelos son herramientas que apoyan al proyecto a largo plazo, ya que documentan las decisiones de diseño tomadas y ya no solo se depende de la memoria.

Aprendemos - UML

Visión general de UML

- UML es un lenguaje para visualizar
- Tradicionalmente, las organizaciones han desarrollado sus propios lenguajes para modelar sistemas, haciendo difícil la comunicación a miembros externos o a nuevos empleados.
- La comunicación de estos modelos conceptuales a otros es propensa a errores a menos que todos los involucrados compartan un lenguaje común. UML ofrece un conjunto de símbolos estándares con semántica bien definida para tal propósito. Un desarrollador puede escribir un modelo en UML y otros podrán interpretarlo sin ambigüedades.
- Hay elementos de un sistema de SW que no son posibles de entender exclusivamente a través del análisis del código de la aplicación, es necesario elaborar modelos.
- Si los desarrolladores que han escrito el código nunca escriben modelos puede haber información que se pierda para siempre. Un modelo explícito además de facilitar la comunicación garantiza que los detalles de diseño no se pierdan.

Aprendemos - UML

UML es un lenguaje para especificar:

Es este contexto, especificar significa construir modelos que sean precisos, no ambiguos y completos. En particular, UML permite la especificación de los aspectos más importantes de análisis, diseño e implantación, que son definidos a lo largo del desarrollo de un sistema de software.

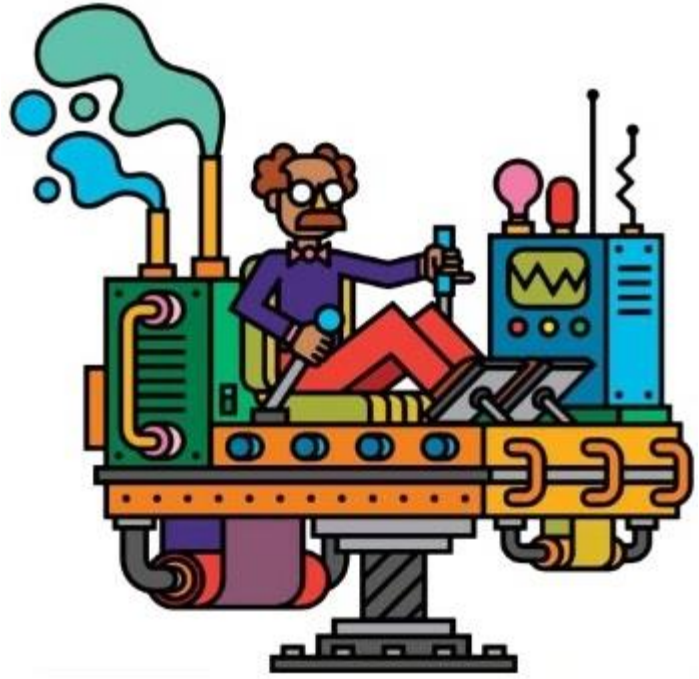
Aprendemos - UML

UML es un lenguaje para construir:

UML no es un lenguaje de programación visual, sin embargo, los modelos generados con esta notación se pueden conectar directamente a una variedad de lenguajes de programación, haciendo posible mapear los elementos de UML a estructuras de un lenguaje de programación o base de datos.

Si un elemento se puede expresar mejor de manera gráfica debemos emplear UML, si se puede expresar mejor de manera textual deberíamos de usar un lenguaje de programación.

Este tipo de mapeo permite la ejecución de ingeniería hacia delante y en reversa.



Aprendemos - UML

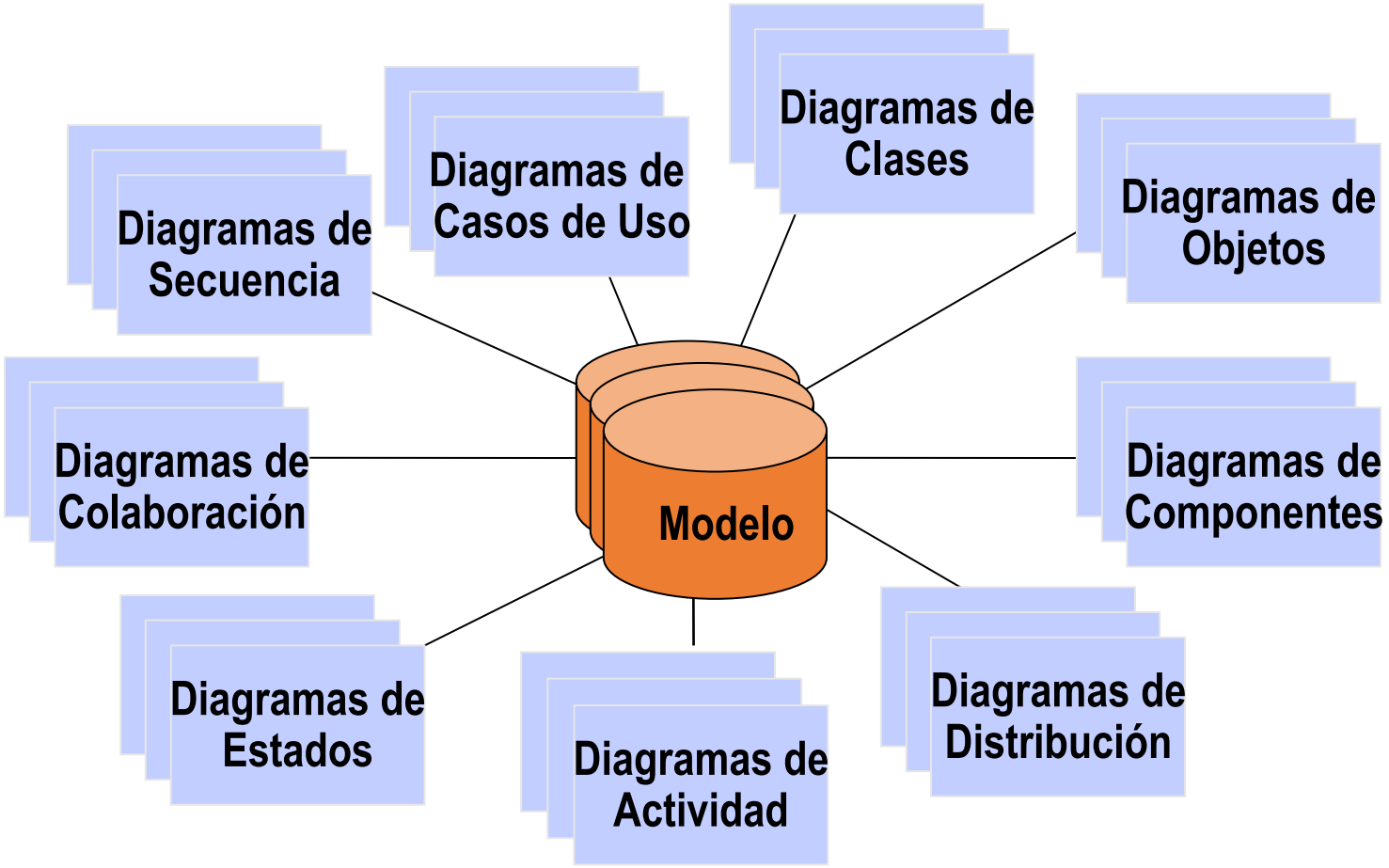
UML es un lenguaje para documentar

Los artefactos de un proyecto son críticos para poder controlar, medir y comunicar las decisiones durante el desarrollo del sistema e incluso después de su instalación.

UML permite la documentación de la arquitectura de un sistema y todos sus detalles. UML también proporciona un lenguaje para expresar requerimientos y pruebas.

Aprendemos - UML

Diagramas de UML



Verificamos lo aprendido:

Kahoot!

http://www.kahoot.it/

Aprendemos:

Tecnología Orientada a Objetos

¿Dudas?

¿Miedos?

¿Temores?

Notas

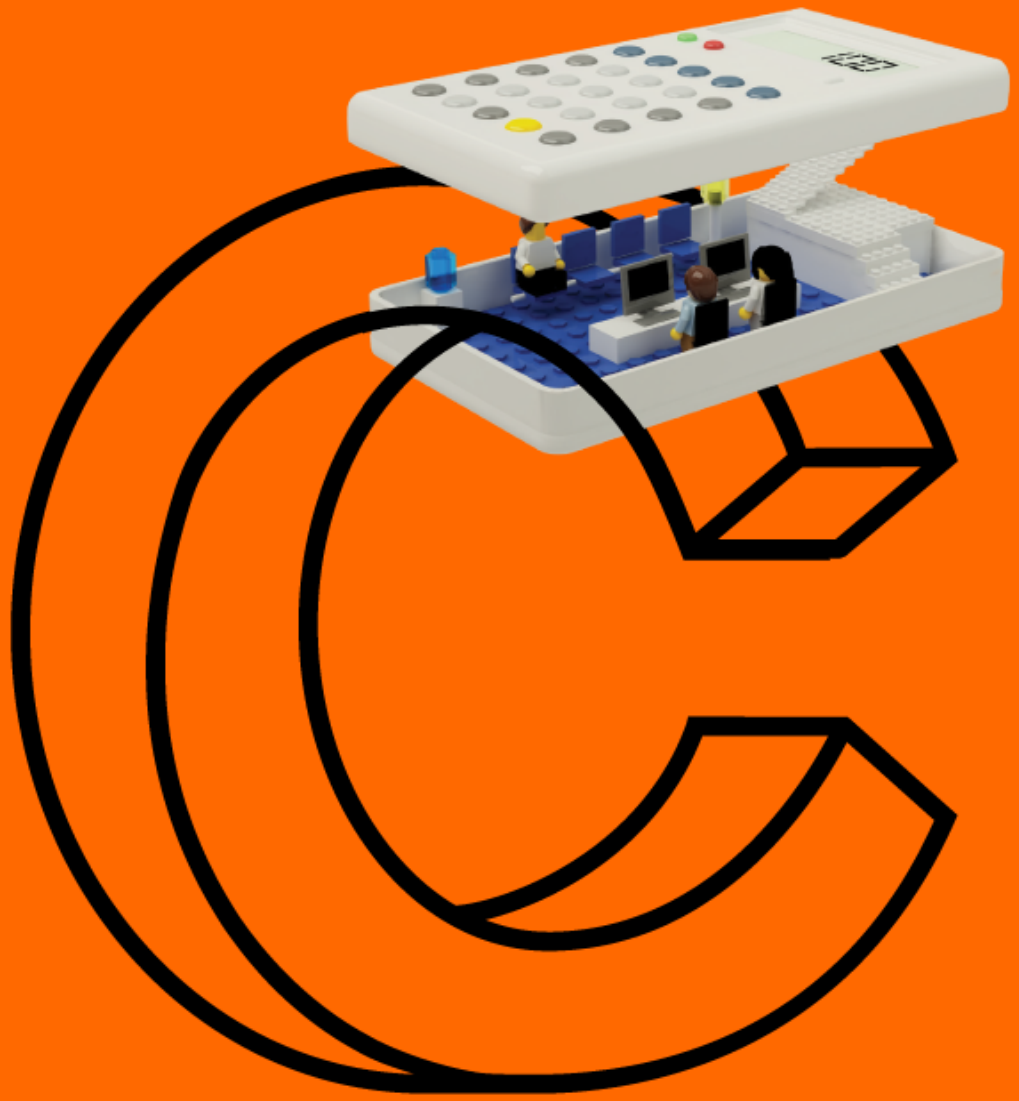
Verificamos lo aprendido:

- Absolución de Preguntas.
- Conclusiones y Recomendaciones.

Aplicamos lo aprendido: Asignación para la clase siguiente

- Comentar en el Foro

Gracias



idat