

# ANÁLISIS Y DISEÑO DE SISTEMAS



# BIENVENIDA

Bienvenido(a) a la asignatura *Análisis y Diseño de Sistemas*, con la cual conocerás los conceptos de distintos paradigmas hacia el análisis y diseño de sistemas. Además, comprenderás los principios, fases, elementos y modelos para el diseño y modelado de sistemas estructurales, así como de los sistemas orientados a objetos.

Aunado a esto, comprenderás el modelado de datos y comportamiento para el diseño de sistemas estructurales. De la misma manera, nos adentraremos en la arquitectura de las vistas y los patrones para el diseño de los sistemas orientados a objetos. Finalmente, conocerás el uso de las metodologías ágiles y no ágiles para el desarrollo de sistemas, así como sus principales elementos y características.

## LIBROS RECOMENDADOS

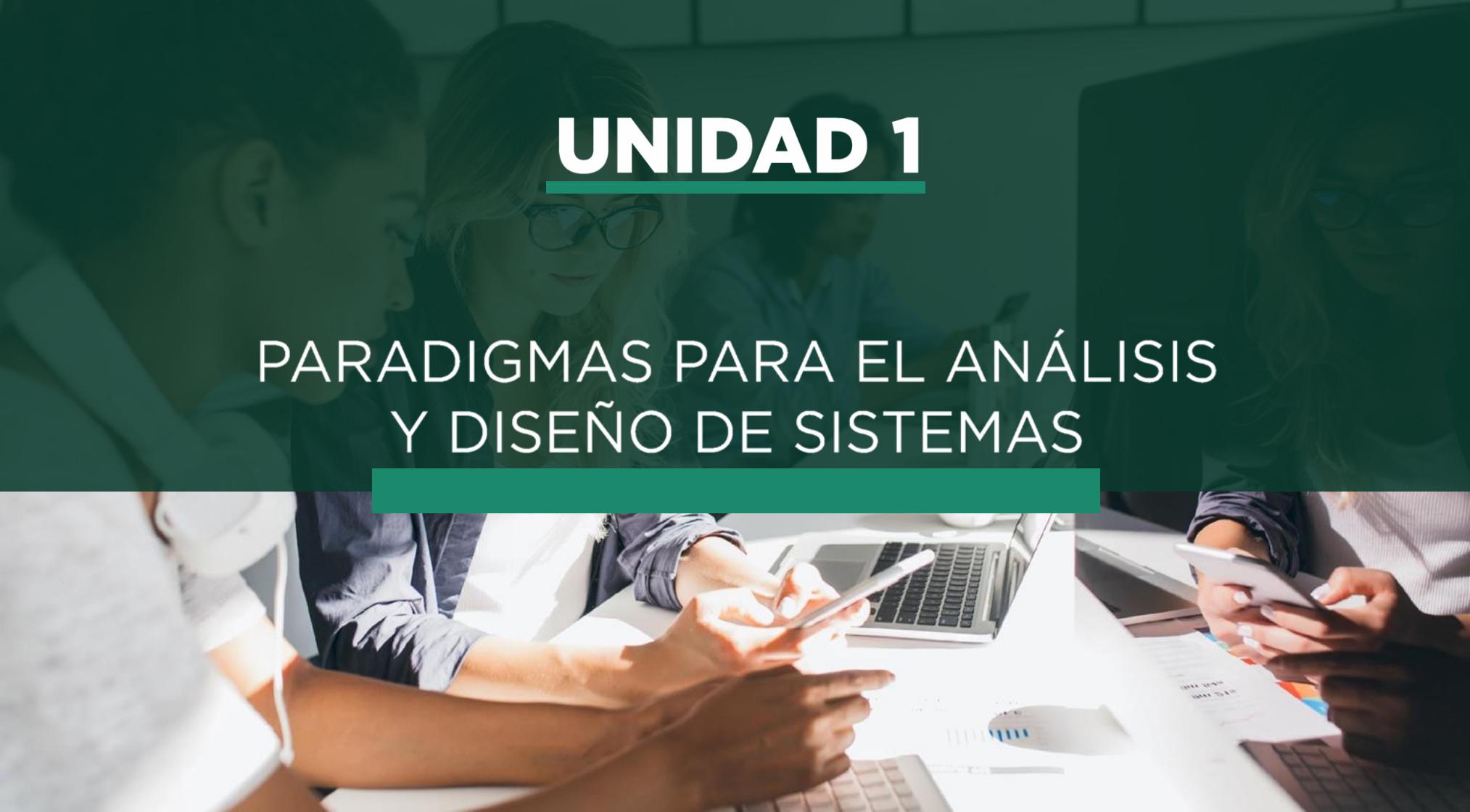
Cervantes, H. & Kazman, R. (2016) Designing software architectures: a practical approach. Addison-Wesley.

Keeling, M. (2017) Design it! From programmer to software architect. The pragmatic programmers.



# **UNIDAD 1**

## **PARADIGMAS PARA EL ANÁLISIS Y DISEÑO DE SISTEMAS**



# TEMARIO

U1

1.1



Paradigma Estructural

1.2



Paradigma Orientado a Objetos

# INTRODUCCIÓN

La asignatura *Análisis y Diseño de Sistemas* tiene como objetivo proporcionar conocimientos teóricos y prácticos que permitirán al estudiante desarrollar habilidades para seleccionar los métodos, técnicas y herramientas más competitivas para el análisis y diseño de un sistema dependiendo del contexto de intervención.

En esta primera unidad, aprenderás a relacionar los principios, fases, elementos y modelos hacia el análisis y diseño de sistemas. Esto, exponiendo el modelado de sistemas estructurales y de sistemas orientados a objetos.



# COMPETENCIAS A DESARROLLAR

- 
- 1.** El alumno será capaz de identificar la secuencia de creación de modelos en el diseño de paradigmas estructurales.
  - 2.** El alumno será capaz de conocer los principios, elementos y fases para el diseño de paradigmas orientados a objetos.

# PARADIGMA ESTRUCTURAL

El **análisis de sistemas**, básicamente, trata de determinar los objetivos y límites del sistema objeto en cuestión. De la misma manera, busca caracterizar su estructura y funcionamiento, así como marcar las directrices que permitan alcanzar los objetivos propuestos y evaluar sus consecuencias.





## PARADIGMA ESTRUCTURAL

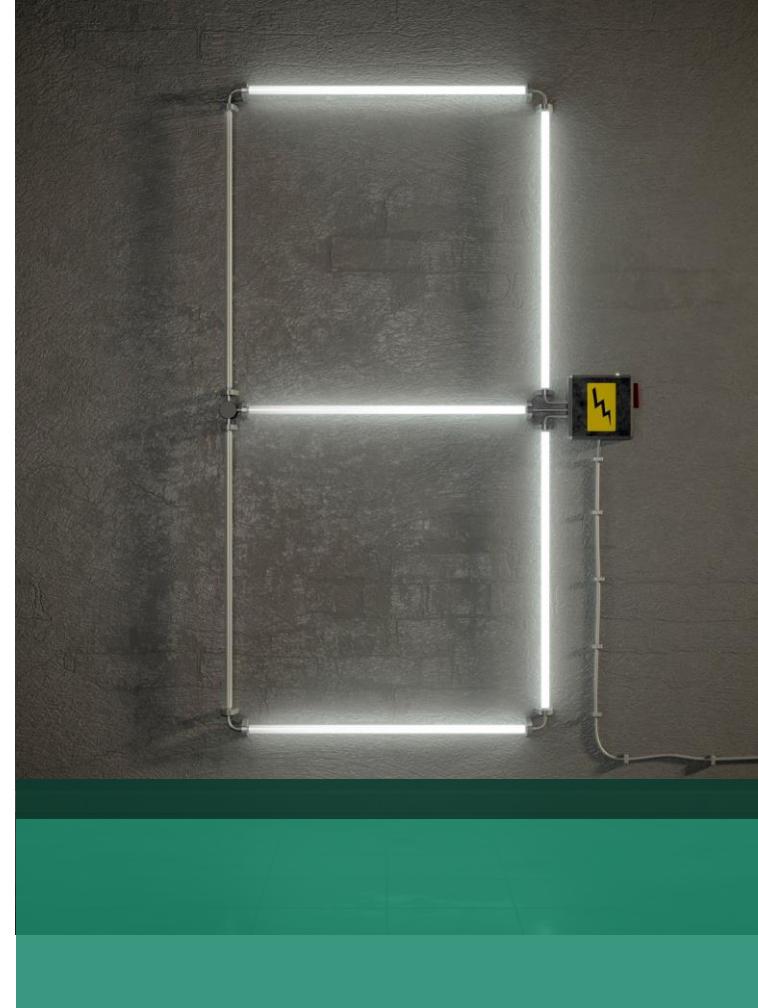
El **diseño de sistemas**, por su parte, se encarga de desarrollar las directrices propuestas durante el análisis. Esto lo realiza en términos de aquella configuración que tenga más posibilidades de satisfacer los objetivos planteados, tanto desde el punto de vista funcional como del no funcional.

El **proceso de diseño** de un sistema complejo se suele realizar de forma descendente.

## PARADIGMA ESTRUCTURAL

El Análisis y Diseño de Sistemas (SAD) se centra principalmente en:

- Sistemas,
- Procesos y
- Tecnología.





## PARADIGMA ESTRUCTURAL

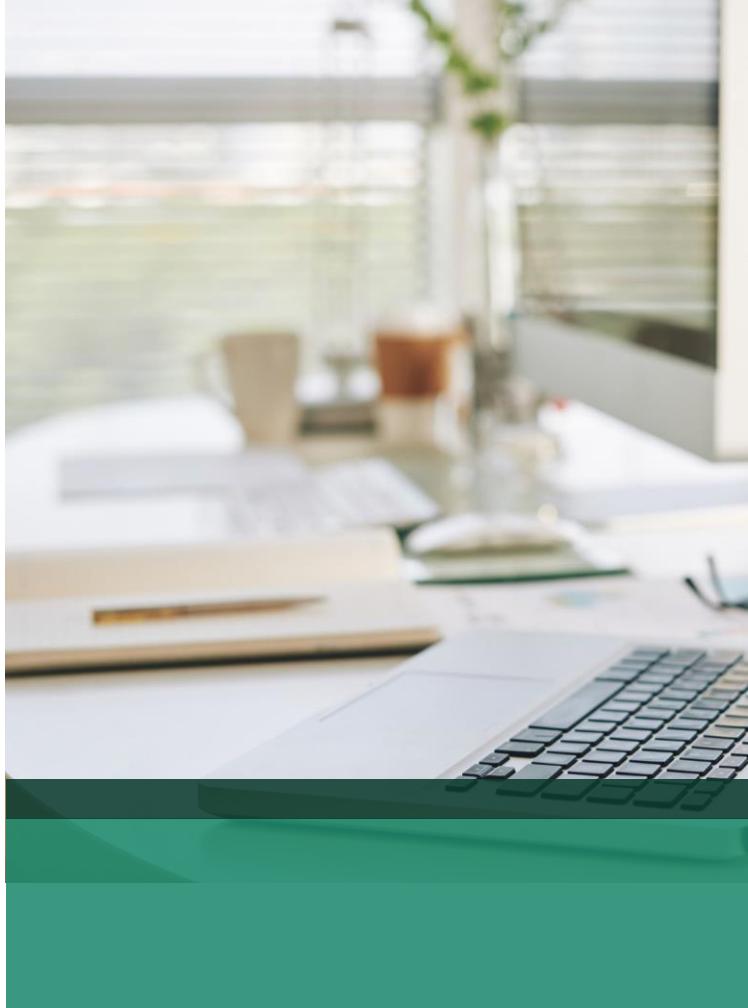
El **enfoque de sistemas** aparece para abordar el problema de la complejidad a través de una forma de pensamiento basada en la totalidad, así como en las propiedades. Este enfoque complementa el reduccionismo científico.

Una **metodología**, por su parte, genera gran cantidad de modelos que representan el aspecto estático y dinámico del sistema. Está compuesta de varias actividades.

## PARADIGMA ESTRUCTURAL

En un primer momento, se selecciona un **modelo de proceso** acorde a la naturaleza del proyecto y de la aplicación. De la misma manera, se seleccionan los métodos y las herramientas a utilizarse, así como los controles y entregas que se requieren.

Es posible combinar los paradigmas o definir uno nuevo, de acuerdo a las necesidades a resolver.





## PARADIGMA ESTRUCTURAL

Para desarrollar un sistema de información se hace uso de las etapas que conforman el ciclo de vida de los sistemas:

- Análisis
- Diseño
- Implementación
- Pruebas
- Implantación
- Mantenimiento

Sin embargo, existen diversas formas de aplicar estas etapas, las cuales son denominadas **modelos**.

## PARADIGMA ESTRUCTURAL



Es de gran importancia evitar la confusión entre los conceptos de modelo, metodología, método, ciclo de vida y proceso de desarrollo. Por esta razón, no deben utilizarse indistintamente el uno del otro.

## PARADIGMA ESTRUCTURAL

Una disciplina puede ejecutarse bajo diferentes metodologías, las cuales dependen del enfoque que se busque:

- Estructurado u
- Orientado a objetos.

El ciclo de vida de un sistema tiene uno o más modelos. A su vez, existen diversas formas de aplicar o secuenciar cada una de las disciplinas o fases.



## PARADIGMA ESTRUCTURAL

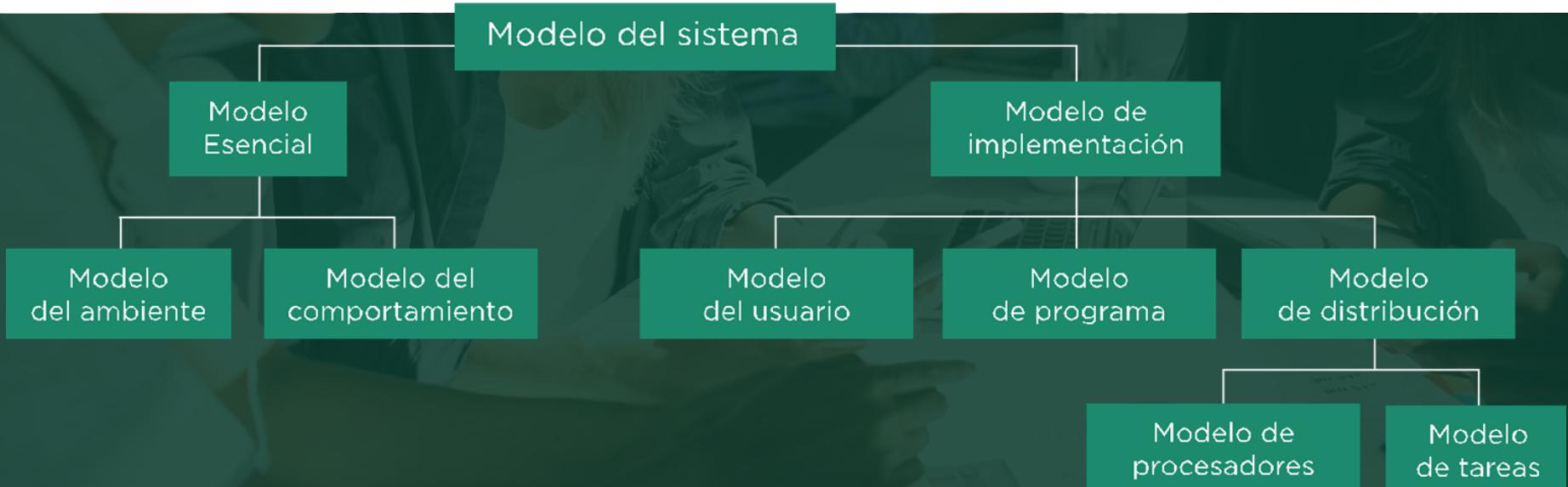
En el **enfoque estructurado**, el concepto principal es el de **función**. Esto quiere decir que el análisis de los sistemas tendrá un enfoque de descomposición funcional. Los principios de diseño estructurado son los siguientes:

- Descomposición por refinamientos sucesivos.
- Creación de una jerarquía modular.
- Elaboración de módulos independientes.



# PARADIGMA ESTRUCTURAL

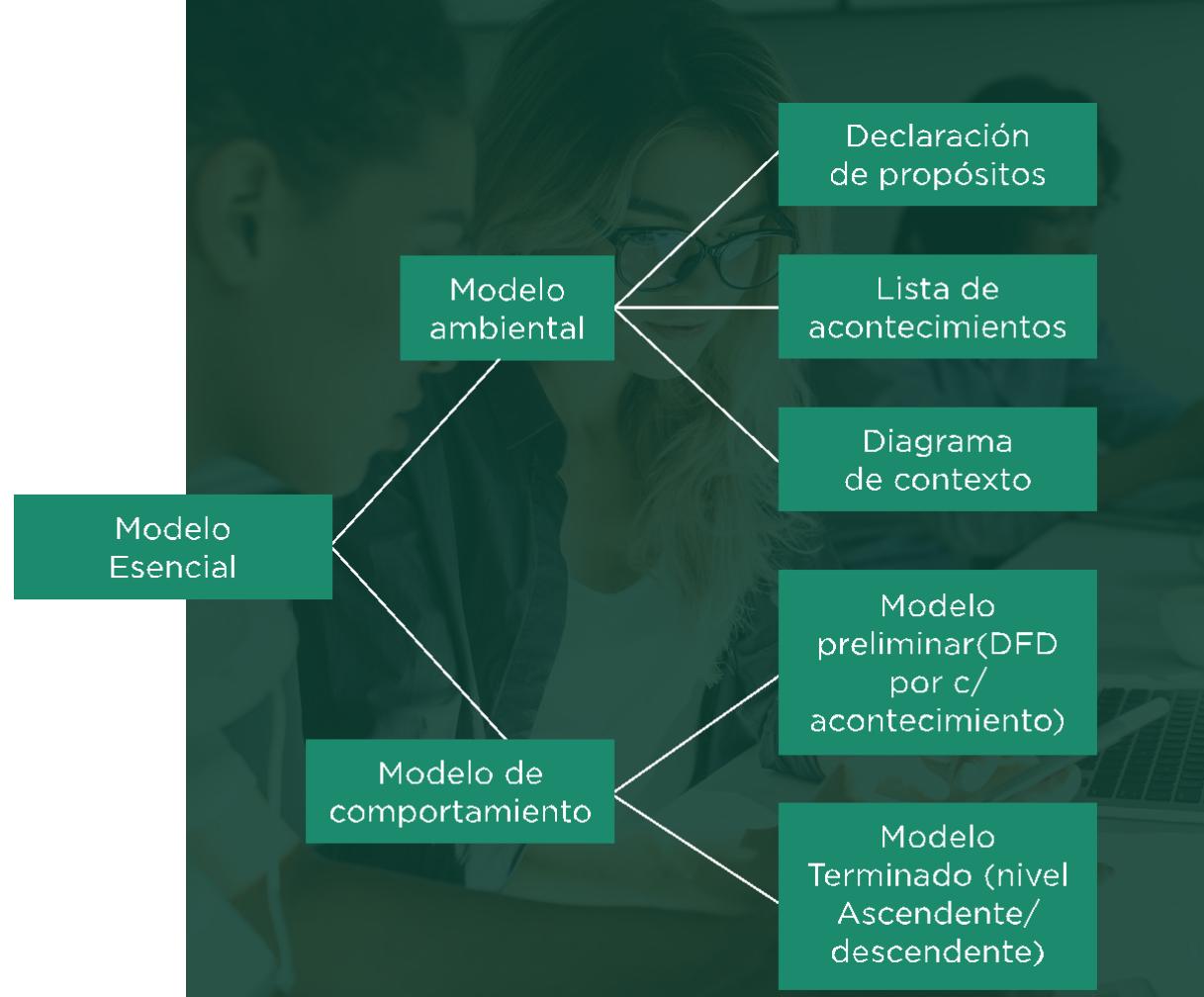
La siguiente figura describe todos los modelos desarrollados durante el ciclo de desarrollo de un sistema. Está basado en el enfoque estructurado. Abarca las actividades de análisis y diseño:



# PARADIGMA ESTRUCTURAL

La actividad de análisis construye el **modelo esencial**.

Por su parte, la actividad de diseño construye el **modelo de implementación**.





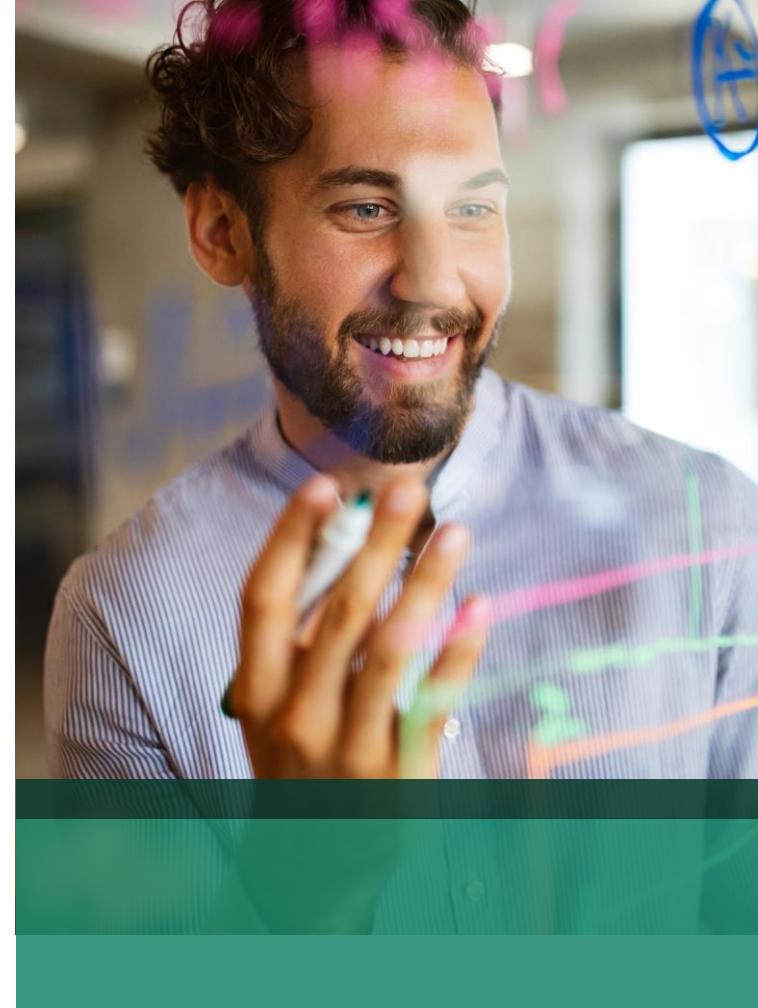
## PARADIGMA ESTRUCTURAL

El modelo esencial puede ser considerado como la aplicación de la metodología de análisis estructurado moderno de Yourdon. Se trata de un modelo que analiza lo que el sistema debe hacer para satisfacer los requerimientos del usuario.

## PARADIGMA ESTRUCTURAL

En el **modelo del ambiente** se definen los elementos que son parte del sistema y los que no lo son. De la misma manera, se definen las interfaces entre el sistema y el resto de los elementos que lo rodean. Dentro de este modelo, se debe considerar:

- Declaración de objetivos.
- Creación de un diagrama de contexto y de una lista de eventos.
- Descripción de los estímulos que recibe el sistema, así como de las respuestas generadas por estos.





## PARADIGMA ESTRUCTURAL

El **modelo de comportamiento** se define por la creación de un DFD y un ERD por cada uno de los eventos de la lista de eventos:

Los DFD por eventos se unen en un único DFD (el modelo funcional).

Los ERD por eventos se unen en un único ERD (el modelo de datos).

## PARADIGMA ESTRUCTURAL

A partir de la etapa del modelo de implementación, el modelo esencial es instanciado en una tecnología dada.

En este punto, se deben considerar las imperfecciones de la tecnología, así como determinar sus limitaciones.

Posteriormente, se diseña la solución sobre la base de esas restricciones tecnológicas.





## PARADIGMA ESTRUCTURAL

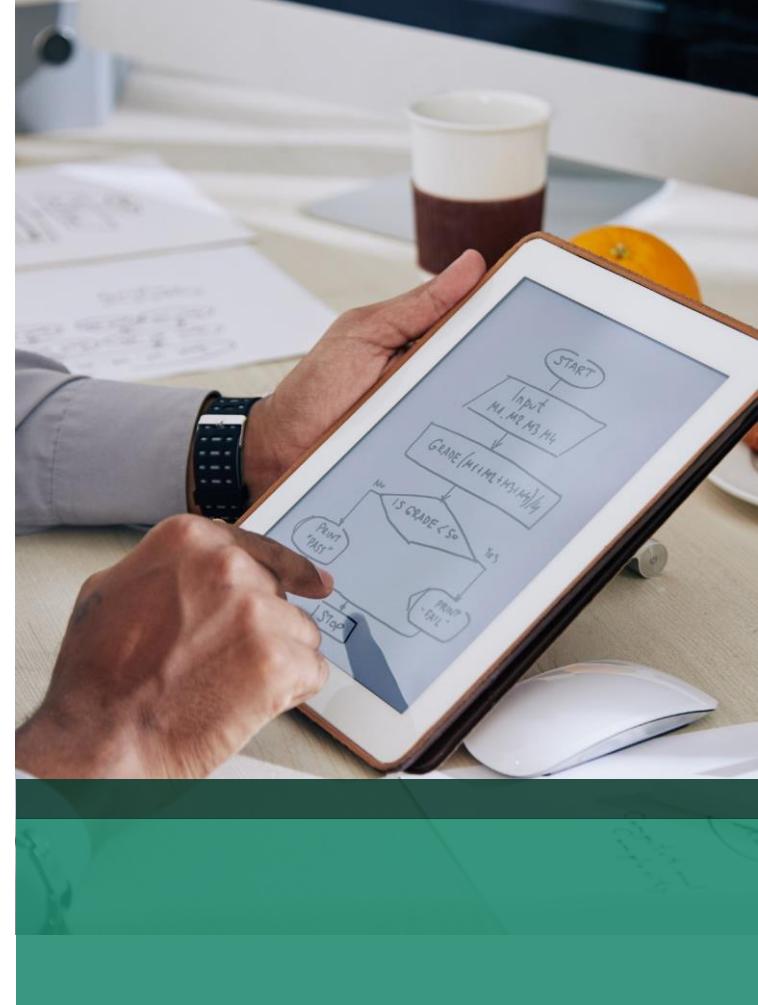
El **modelo de implantación del usuario** es el punto de inflexión entre la etapa de análisis y la etapa de diseño. Aquí se especifica un conjunto de restricciones que el usuario deberá imponer al grupo de desarrollo. Estas condicionarán al diseñador.

## PARADIGMA ESTRUCTURAL

El **modelo de distribución** describe todas las decisiones relativas a la arquitectura de hardware (modelo de procesadores) y a la estructuración general de la arquitectura de software (modelo de tareas).

Se incorporan, en los modelos creados hasta este punto, algunas distorsiones destinadas a optimizar el uso de esa tecnología.

El criterio fundamental es: minimizar todo lo posible las distorsiones agregadas.



## PARADIGMA ESTRUCTURAL



El **modelo de procesadores** asigna el modelo esencial a distintos procesadores. Además, determina la arquitectura de comunicación entre ellos. Implica la asignación de procesos y almacenes a los mismos.

## PARADIGMA ESTRUCTURAL

En el **modelo de tareas**, los modelos resultantes de la creación del modelo de procesadores son estudiados por separado. Esto sirve para determinar tareas diferentes.

Una tarea está formada por procesos (puede ser uno). A las tareas se les asignan localidades específicas de la memoria física y/o virtual de un procesador.





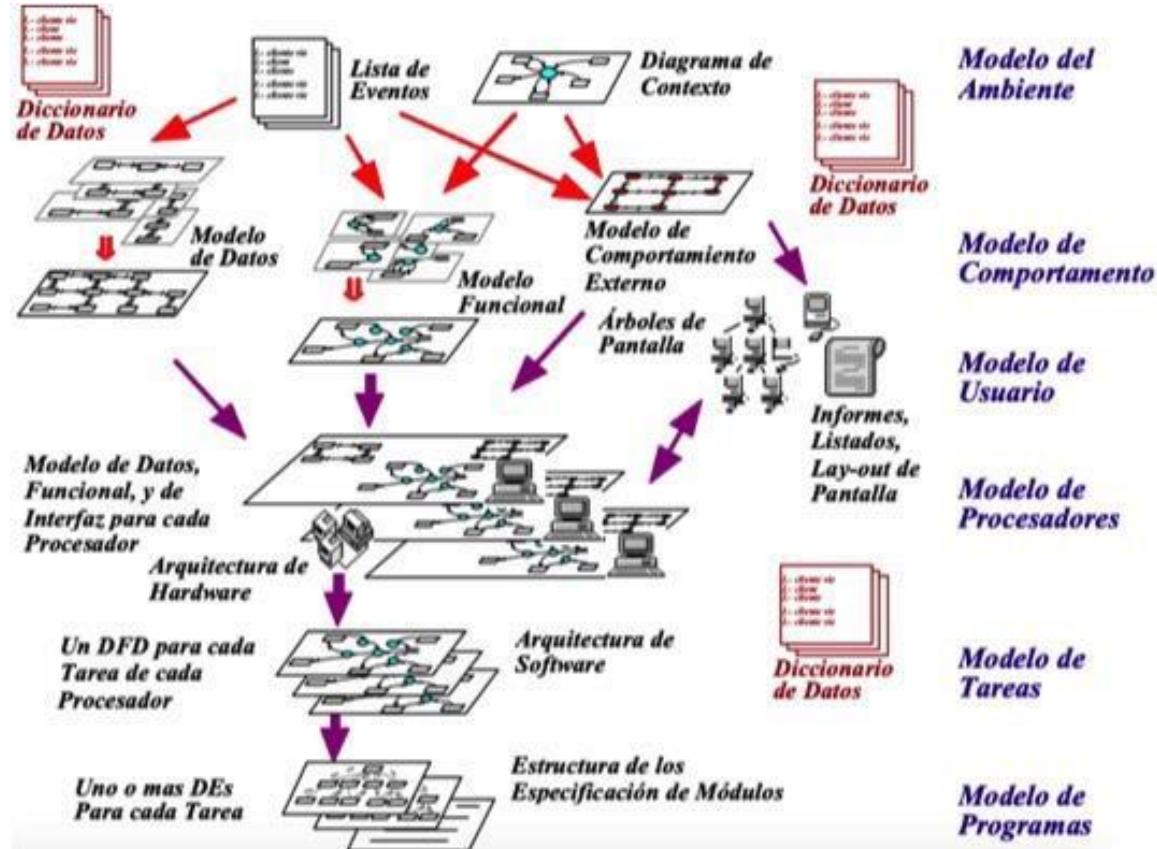
## PARADIGMA ESTRUCTURAL

El **modelo de programas** indica que para cada tarea debe desarrollarse un modelo de programa. De esto se encarga principalmente el diseño estructurado.

La estructura del programa que implementa cada una de las tareas resultantes de las etapas de modelado de procesadores y tareas es diseñada mediante la aplicación de las técnicas y estrategias descritas por el diseño estructurado.

# PARADIGMA ESTRUCTURAL

Tal como se indicó con anterioridad, el paradigma estructural cuenta con un proceso de creación de modelos. Este se representa en la figura siguiente:



# VIDEO

Te invitamos a ver el siguiente video:



# PARADIGMA ORIENTADO A OBJETOS

El paradigma orientado a objetos tiene como concepto principal al objeto. Bajo esta perspectiva, este enfoque analiza y diseña los sistemas con miras en las responsabilidades a través de objetos que están presentes en una situación real. Sean estos de carácter físico o abstracto.



## PARADIGMA ORIENTADO A OBJETOS

Elementos de objeto orientado al sistema:

- **Objetos.** Se trata de algo que existe en el dominio del problema y que puede identificarse por *data* (atributo) o comportamiento. Todas las entidades tangibles (estudiante, paciente, etc.) y algunas entidades intangibles (cuenta bancaria) se modelan como un objeto.
- **Atributos.** Describen información entrante sobre el objeto.





## PARADIGMA ORIENTADO A OBJETOS

- **Comportamiento.** Especifica lo que puede hacer el objeto. Define la operación realizada sobre los mismos.
- **Clase.** Esta se encarga de encapsular *data* y su comportamiento. Se puede ver como objetos con significado y propósito similares agrupados como en un salón de clases.

## PARADIGMA ORIENTADO A OBJETOS

- **Métodos.** Determinan el comportamiento de una clase. Tienen que ver, simplemente, con la acción que puede realizar un objeto.
- **Mensaje.** Es una llamada a una función o procedimiento de un objeto a otro. Esta información es enviada a objetos para activar métodos. Básicamente, un mensaje es una llamada a una función o procedimiento de un objeto a otro.





## PARADIGMA ORIENTADO A OBJETOS

Características de un sistema orientado a objetos:

**Encapsulación.** Es un proceso de ocultar información. Es la combinación de proceso y *data* en una sola entidad. La *data* de un objeto se oculta del resto del sistema. Esta solo está disponible a través de los servicios de la clase. Le permite mejorar o modificar los métodos utilizados por los objetos sin afectar otras partes de un sistema.

## PARADIGMA ORIENTADO A OBJETOS

Formas de encapsular:

- **Abierto.** Hace que el miembro de la clase pueda ser accedido desde el exterior de la misma. Desde cualquier parte del programa.
- **Protegido.** Solo es accesible desde la clase y las clases que heredan (a cualquier nivel).
- **Cerrado.** Solo es accesible desde las clases.



## PARADIGMA ORIENTADO A OBJETOS



**Abstracción.** Este es un proceso de tomar o seleccionar el método y los atributos necesarios para especificar el objeto. Se centra en las características esenciales de un objeto en relación con la perspectiva del usuario.

## PARADIGMA ORIENTADO A OBJETOS

**Relaciones.** Todas las clases del sistema están relacionadas entre sí. Los objetos no existen de forma aislada, existen en relación con otros objetos.



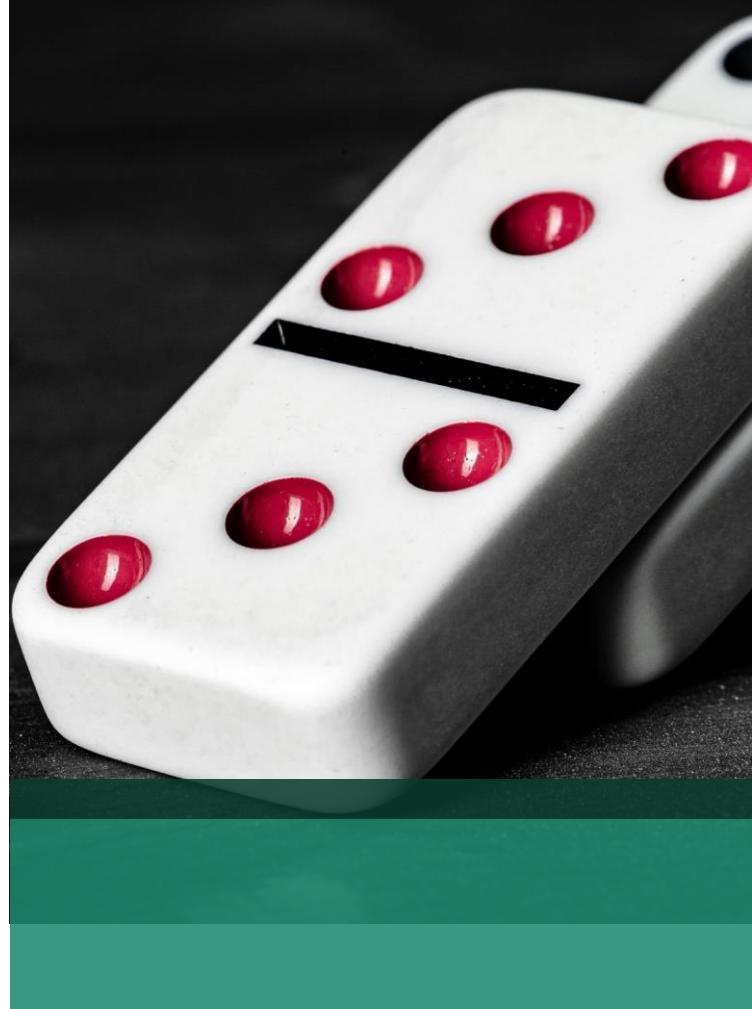
**Hay tres tipos de relaciones de objeto:**

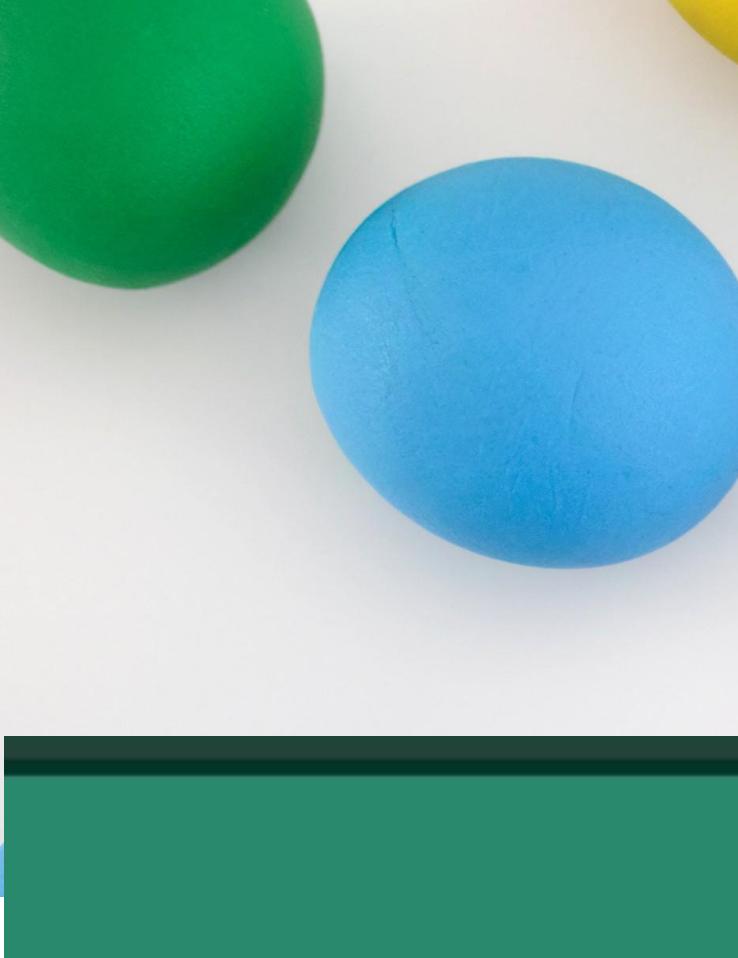
- **Agregación.** Indica la relación entre un todo y sus partes.
- **Asociación.** Dos clases están vinculadas o conectadas de tal manera que una trabaja con otra para realizar una tarea; es decir, una clase actúa sobre otra clase.
- **Generalización.** La clase secundaria se basa en la clase principal. Esto indica que dos clases son similares pero tienen algunas diferencias.

## PARADIGMA ORIENTADO A OBJETOS

**Herencia.** Se trata de una característica interesante que te permite crear subclases de una clase existente heredando sus atributos y/u operaciones de clases existentes. Existen dos tipos de herencia:

- Por especialización.
- Por generalización.





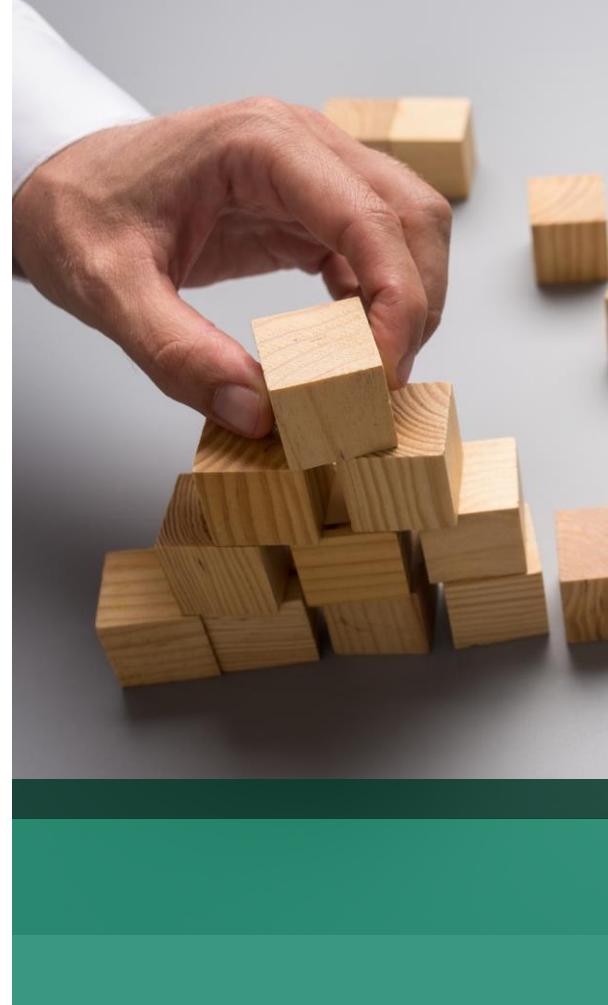
## PARADIGMA ORIENTADO A OBJETOS

**Polimorfismo y vinculación dinámica.** Se trata de la capacidad de adoptar muchas formas diferentes. Se aplica tanto a objetos como a operaciones:

- Un **objeto polimórfico** es un objeto cuyo tipo verdadero está oculto en una clase superior o principal.
- Una **operación polimórfica** se puede realizar de manera diferente por diferentes clases de objetos.

## Fases de diseño:

- **Subsistema.** Contiene una representación de cada uno de los subsistemas que le permiten al software conseguir los requisitos definidos por el cliente e implementar la infraestructura técnica que los soporta.
- **Clases y objetos.** Contiene las jerarquías de clases que permiten crear el sistema. Utiliza generalizaciones y especializaciones mejor definidas incrementalmente.





## PARADIGMA ORIENTADO A OBJETOS

- **Mensajes.** Contiene los detalles que permiten a cada objeto comunicarse con sus colaboradores. Establece las interfaces externas e internas para el sistema.
- **Responsabilidades.** Contiene las estructuras de datos y el diseño algorítmico para todos los atributos y operaciones de cada objeto.

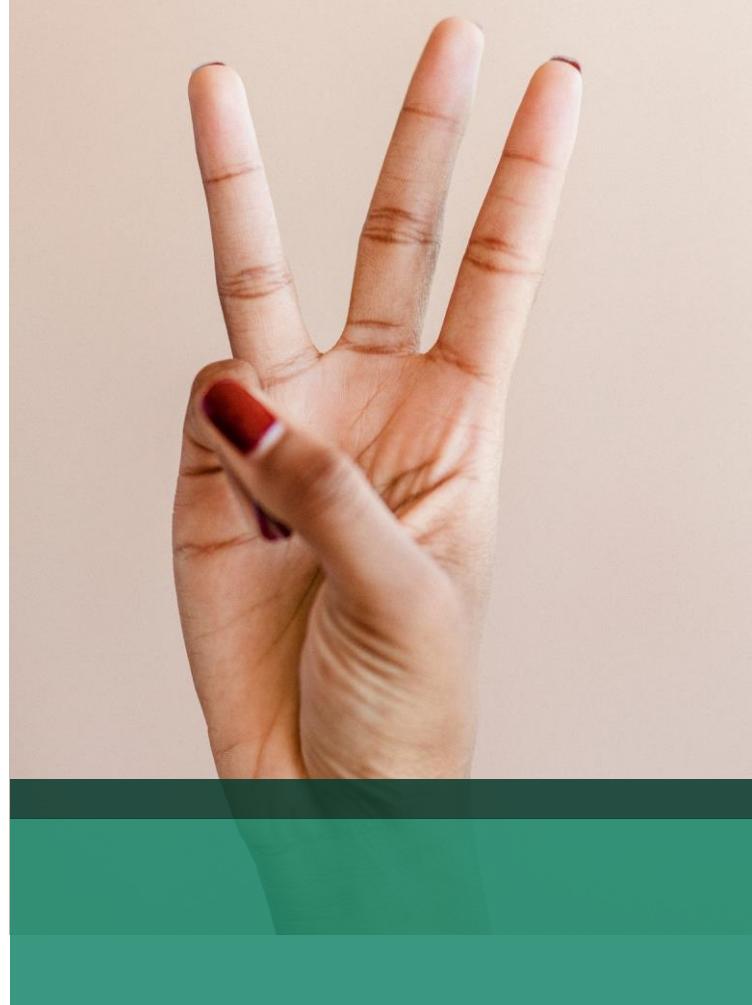
## PARADIGMA ORIENTADO A OBJETOS

El **marco de desarrollo** de una aplicación software está formado por las **tres fases tradicionales**:

- Análisis,
- Diseño e
- Implementación.

**Análisis.** Es la fase cuyo objetivo es estudiar y comprender el dominio del problema. Es decir, este se centra en responder al interrogante:

*¿Qué hacer?*





## PARADIGMA ORIENTADO A OBJETOS

**Diseño.** Dirige sus esfuerzos a desarrollar la solución a los requisitos planteados en el análisis. En este sentido, el diseño se centra en el espacio de la solución que intenta dar respuesta a la cuestión:

*¿Cómo hacerlo?*

**Implementación.** Se encarga de la traducción del diseño de la aplicación al lenguaje de programación elegido. Para ello, adapta la solución a un entorno concreto.

## PARADIGMA ORIENTADO A OBJETOS



Que este marco de desarrollo sea válido tanto para los desarrollos tradicionales (desarrollos estructurados) como para los desarrollos orientados a objetos no significa que la realización de las actividades propias de cada fase se lleve a cabo de la misma manera.



## PARADIGMA ORIENTADO A OBJETOS

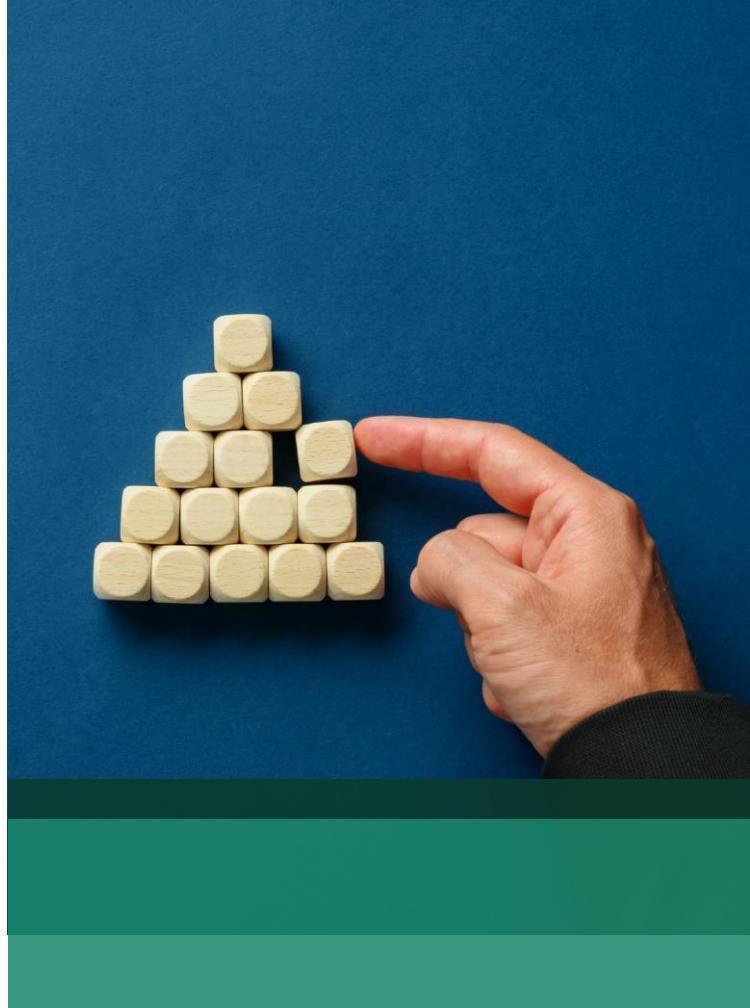
En los desarrollos estructurados existe mucha distancia entre las fases de análisis y de diseño; e incluso entre los diferentes modelos generados en una misma fase.

Esta separación se conoce con el nombre de **intervalo semántico**. Esta es la barrera que la orientación a objetos intenta eliminar al difuminar la frontera entre las diferentes fases.

## PARADIGMA ORIENTADO A OBJETOS

La transición entre las fases de análisis y diseño en la orientación al objeto es más suave, por así decirlo, que en las metodologías estructuradas. Ya que en la primera no existe tanta diferencia entre las etapas.

Esto implica que es difícil determinar dónde acaba el Análisis Orientado a Objetos (AOO) y dónde comienza el Diseño Orientado a Objetos (DOO).





## PARADIGMA ORIENTADO A OBJETOS

**Objetivo del AOO.** Modelar la semántica del problema en términos de objetos distintos, pero relacionados.

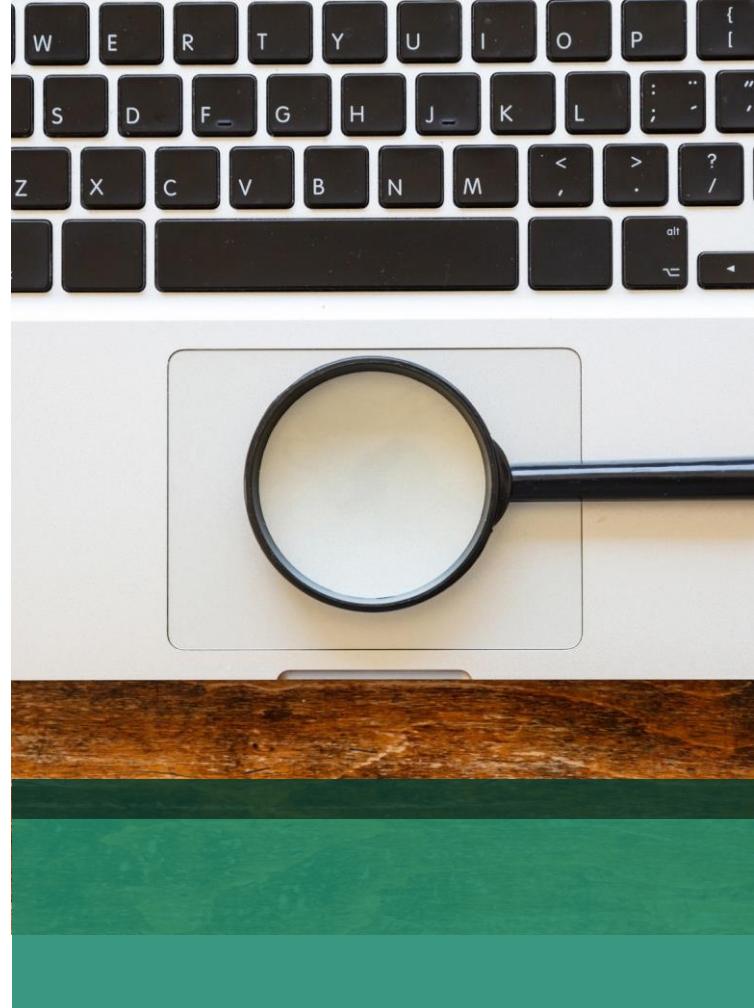
**Objetivo del DOO.** Conlleva reexaminar las clases del dominio del problema, al refinarlas, extenderlas y reorganizarlas para mejorar su reutilización y tomar ventaja de la herencia.

## PARADIGMA ORIENTADO A OBJETOS

El análisis se centra en la representación del problema; es decir, en la identificación de las abstracciones que representa el significado de las especificaciones y de los requisitos del sistema.

El énfasis del diseño se centra en la definición de la solución.

Los objetos semánticos serán refinados durante la fase de análisis y de diseño. Estos serán completados con los objetos propios del dominio de la solución.





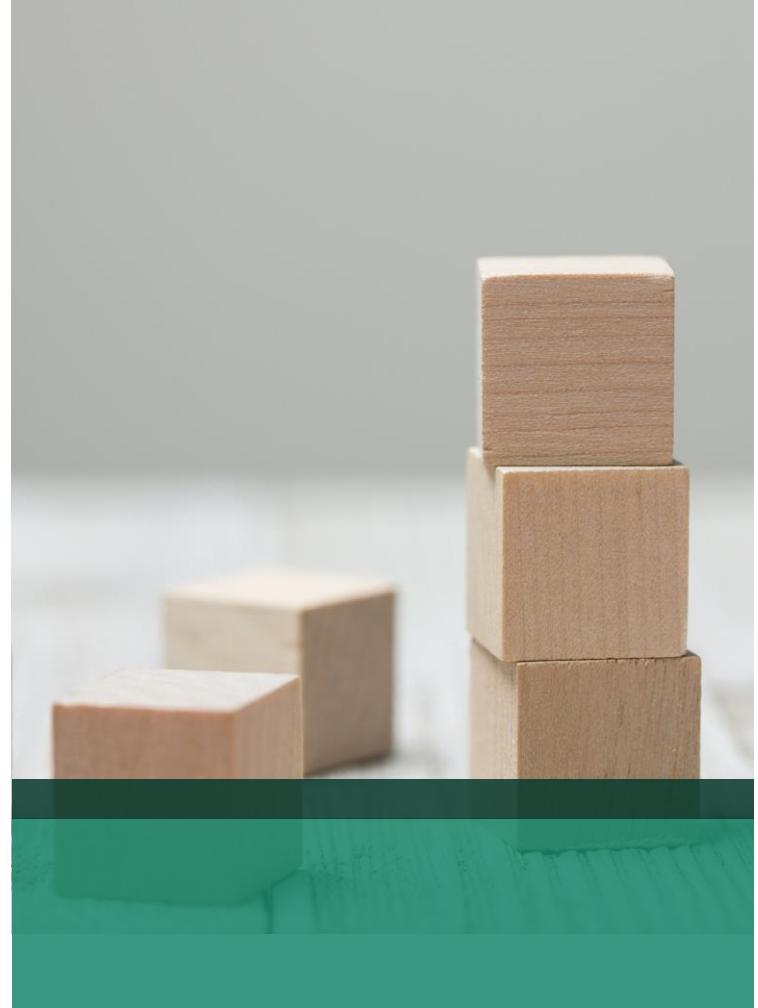
## PARADIGMA ORIENTADO A OBJETOS

Se puede definir AOO como el proceso que modela el dominio del problema identificando y especificando un conjunto de objetos semánticos que interaccionan y se comportan de acuerdo a los requisitos del sistema

Por su parte, el DOO es el proceso que modela el dominio de la solución. Esto incluye a las clases semánticas con posibles añadidos, así como a las clases de interfaz, aplicación y utilidad identificadas durante el diseño.

## Actividades del AOO:

- La identificación de clases semánticas, atributos y servicios.
- Identificación de las relaciones entre clases (generalizaciones, agregaciones y asociaciones).
- El emplazamiento de las clases, atributos y servicios.
- La especificación del comportamiento dinámico mediante paso de mensajes.





## PARADIGMA ORIENTADO A OBJETOS

Actividades del DOO:

- Añadir las clases interfaz, base y utilidad.
- Refinar las clases semánticas.

## PARADIGMA ORIENTADO A OBJETOS

Se podría afirmar que el AOO y el DOO no deben verse como fases muy separadas. Lo que se recomienda es llevarlas a cabo concurrentemente. De esta manera, el modelo de análisis no puede completarse en ausencia de un modelo de diseño, ni viceversa. Debe existir sinergia entre los dos conceptos.





# FORO 1

Entorno de trabajo.

Participa en el foro enviando imágenes que demuestren que ya tienes acceso a las siguientes herramientas en su versión de prueba:

- diagramas.net, LucidChart o StarUML

Presiona el botón para participar en el foro.



# CONCLUSIÓN



El enfoque estructurado analiza y diseña los sistemas a través de una descomposición funcional. Por su parte, el enfoque orientado objetos analiza y diseña los sistemas con una visión de responsabilidades a través de objetos que se encuentran presentes en una situación real, ya sean de carácter físico o abstracto.

El paradigma de la orientación a objetos es mucho más potente que el estructurado, ya que pone especial énfasis en la reutilización. Esto quiere decir que proporciona mecanismos efectivos que permiten reutilizar aquello que es común. Además, cuenta con el concepto de herencia, que es vital para expresar explícitamente las características comunes de una serie de objetos. Estas quedan escondidas en el análisis y diseño estructurado.

# ¡FELICIDADES!

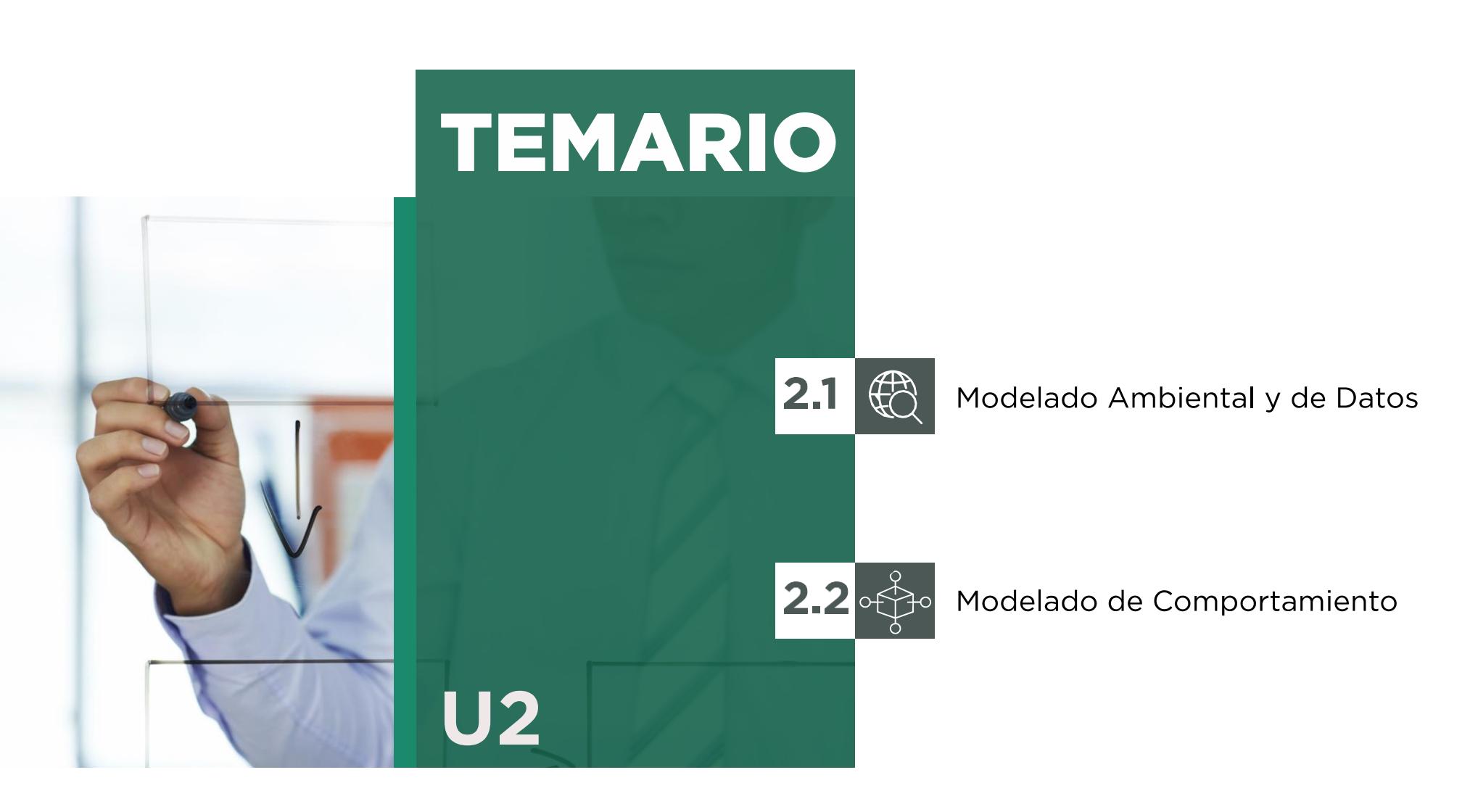


Acabas de concluir la primera unidad de tu curso *Análisis y Diseño de Sistemas*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.

# **UNIDAD 2**

---

DISEÑO Y MODELADO DE SISTEMAS  
ESTRUCTURALES



# TEMARIO

## U2

**2.1**



Modelado Ambiental y de Datos

**2.2**



Modelado de Comportamiento



# INTRODUCCIÓN

En esta segunda unidad identificarás las diferentes metodologías para ejecutar el diseño y modelado de sistemas estructurados, así como sus notaciones y simbologías relacionadas.



# COMPETENCIAS A DESARROLLAR



- 1.** El alumno será capaz de reconocer y modelar la parte dinámica del sistema conforme al modelado ambiental; y la parte estática conforme al modelado de datos.
  
- 2.** El alumno construirá eficazmente los diagramas para el modelado de comportamiento para los sistemas estructurales.

# MODELADO AMBIENTAL Y DE DATOS

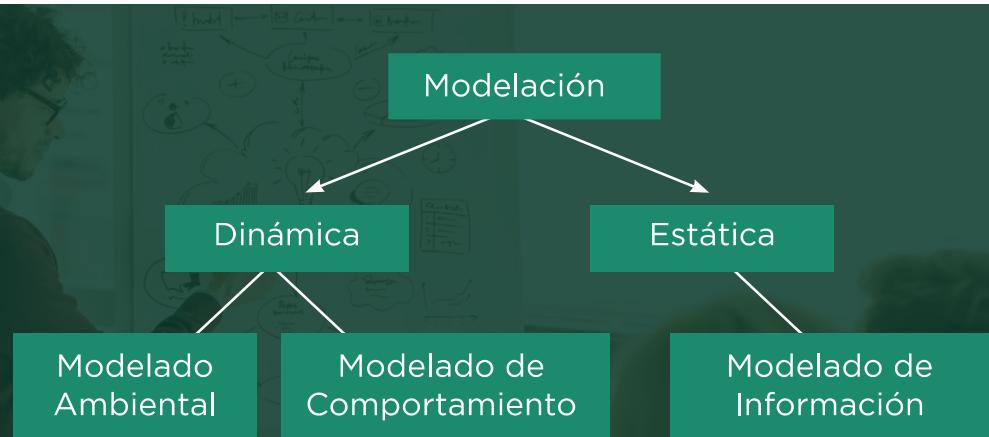
En sentido estricto, un modelo es una representación de la realidad. Conforme al análisis de sistemas y el paradigma que se esté utilizando, se emplean ciertos diagramas para especificar los modelos.



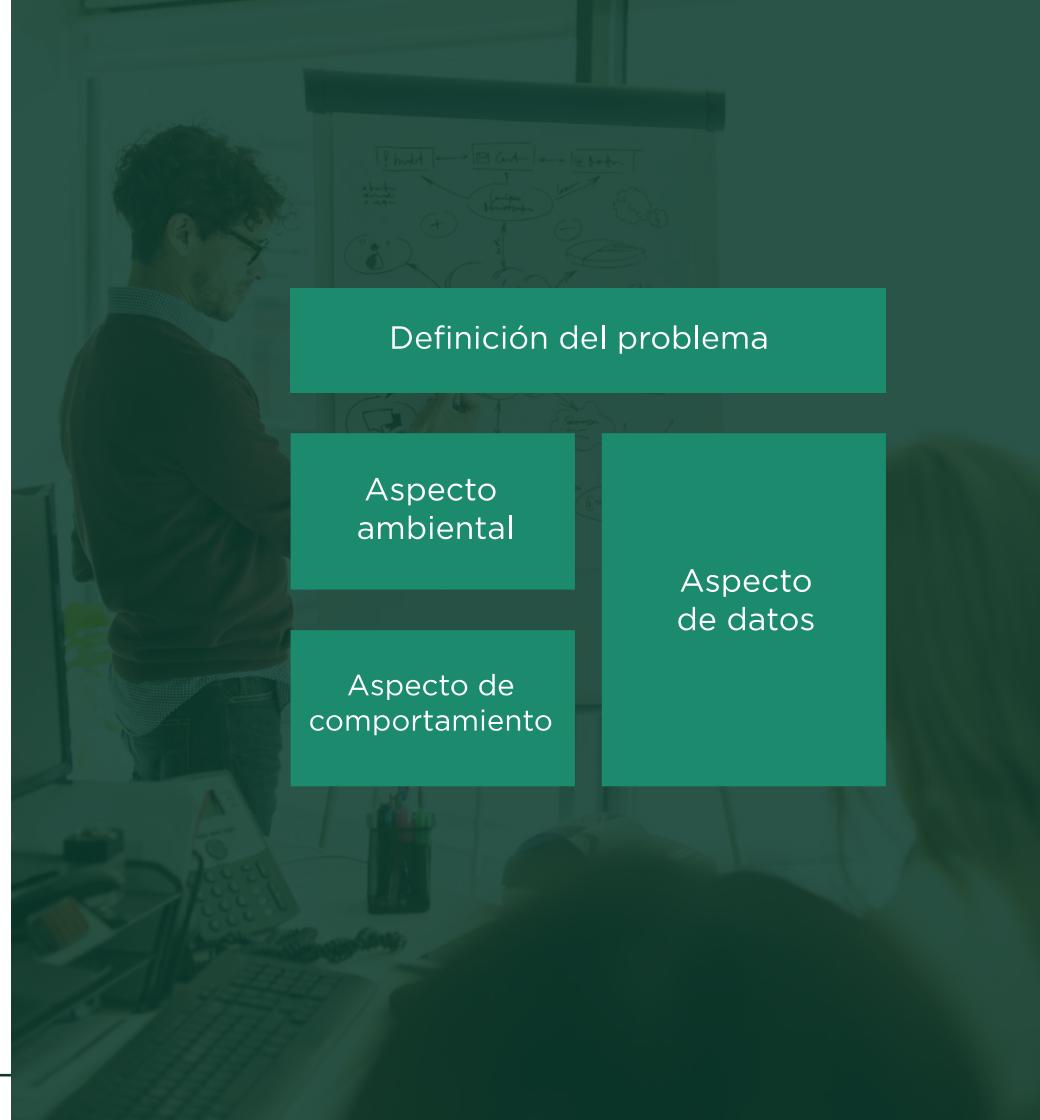
## MODELADO AMBIENTAL Y DE DATOS

Después de identificar el problema por resolver, se procede a documentar la parte **dinámica** y **estática** del sistema actual.

Los aspectos que se encargan de modelar la parte dinámica del sistema son el ambiental y el de comportamiento.



Por otra parte, el aspecto que se encarga de modelar la parte estática del sistema es el de información. Este aspecto va a generar un modelo. En consecuencia, al final tendremos un modelo ambiental, un modelo de comportamiento y un modelo de datos.



# MODELADO AMBIENTAL Y DE DATOS

Si tenemos un mejor conocimiento de los procesos de la organización, es recomendable empezar por los modelos ambiental y de comportamiento, para luego pasar al modelo de información o de datos.

Modelado  
Ambiental

Modelado de  
Comportamiento

Modelado de  
datos

Sin embargo, si tenemos un mejor conocimiento sobre los datos e información que genera la empresa, es recomendable empezar por el modelo de información, para posteriormente pasar a los modelos ambiental y de comportamiento.

A estos aspectos se les denomina **elementos del modelo de análisis de sistemas**.



## Modelo ambiental

El objetivo del aspecto ambiental es modelar el aspecto dinámico externo del sistema. Es decir, establecer tanto el alcance del sistema como sus límites.

Para modelar este aspecto, se utiliza el **diagrama de contexto**. Este se utiliza tanto para el sistema actual como para el sistema propuesto.



**Diagrama de contexto (nivel 0).** Es el nivel más alto. Delimita la frontera entre el sistema y el mundo exterior (interfaz externa). Se representa por:

- **Burbuja (sistema).** Un único proceso que representa una caja negra del sistema completo.
- **Rectángulo (entidad).** Un conjunto de entidades que representan la procedencia y el destino de la información.
- **Flechas (flujos).** El flujo de información que se intercambia entre el sistema y las entidades externas.





### Modelado de datos

El objetivo de la información es modelar el aspecto estático del sistema. Es decir, las estructuras de persistencia de datos y las relaciones que existen entre ellas.

Para modelar el aspecto de información se utiliza el **Diagrama de Entidad-Relación (DER)**. Cabe mencionar que este diagrama se utiliza tanto para el sistema actual como para el sistema propuesto.

## Diagrama Entidad-Relación (DER).

Las entidades son los elementos más significativos en el DER. Cada entidad está compuesta por atributos.

En un futuro, cuando el DER se convierta en una base de datos, cada atributo contendrá un tipo de dato, a saber: rectángulo (entidad) y relación.

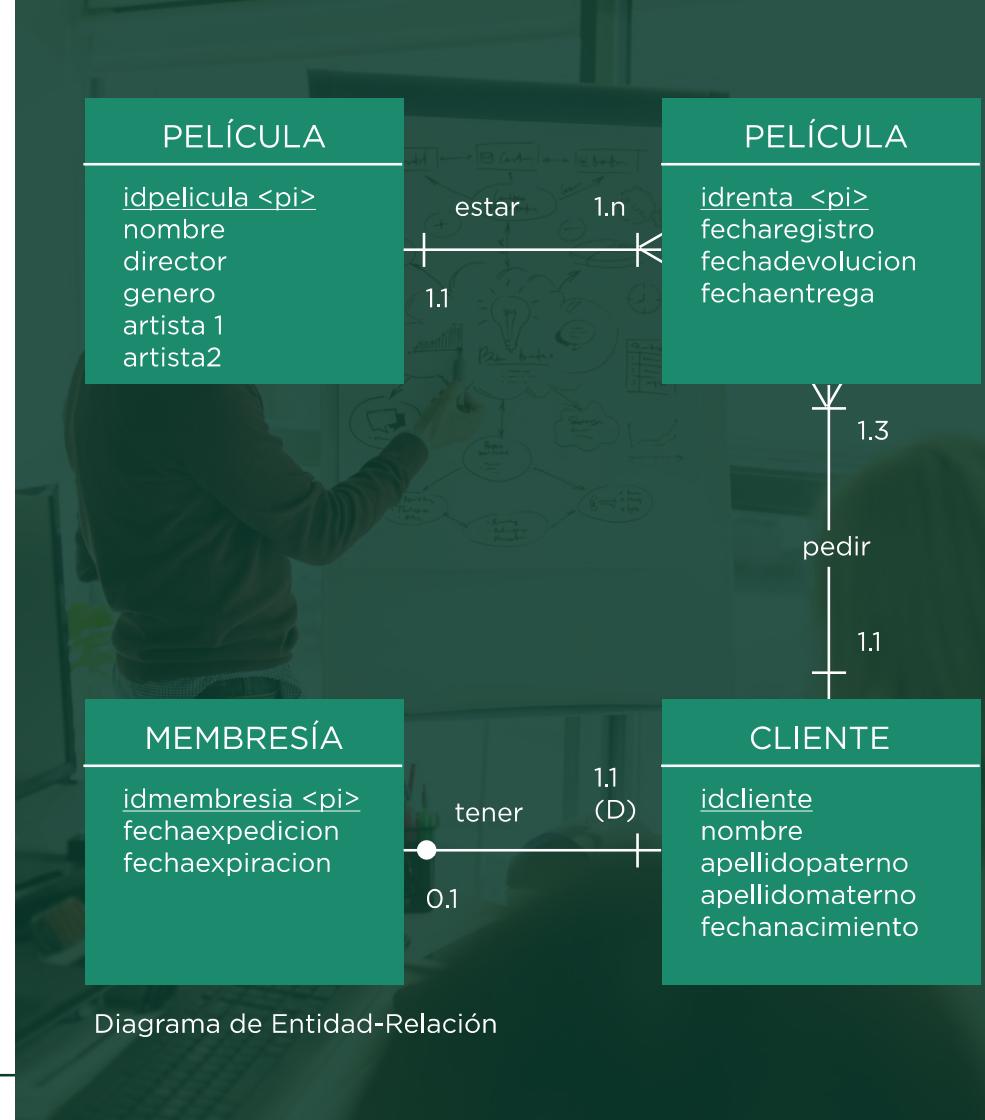


Diagrama de Entidad-Relación

Modelo de datos semántico que se basa en la representación de la información en tres categorías:

- **Entidades.** Objetos que se van a modelar.



- **Atributos.** Propiedades de esos objetos.

- **Relaciones.** Asociaciones entre las entidades.

## Componentes

- **Rectángulos.** Representan un conjunto de entidades. Se dividen en:
  - Sencillos: conjunto de entidades fuertes.
  - Dobles: conjunto de entidades débiles.
  
- **Elipses.** Representan atributos. Se dividen en:
  - Dobles: atributos multivaluados.
  - Discontinua: atributos derivados.

**Nota.** Los atributos que son miembros de la clave primaria se subrayan.



## MODELADO AMBIENTAL Y DE DATOS



- **Rombos.** Representan conjuntos de relaciones.
- **Líneas.** Conectan los atributos con los conjuntos de entidades y estos con los conjuntos de relaciones.
- **Líneas dobles.** Participación total (si cada entidad en un conjunto de entidades participa en al menos una relación en un conjunto de relaciones) de una entidad en un conjunto de relaciones.
- **Flechas.** Expresan la cardinalidad de asignación.

## Cardinalidad y Modalidad

Existen tres tipos de cardinalidad:

- De uno a uno.
- De uno a muchos.
- De muchos a muchos.

Símbolo	Descripción
 estar      1..n	Relación de uno a muchos
 estar      n..n	Relación de muchos a muchos
 estar      1..1	Relación de uno a uno

Símbolos para representar la Cardinalidad

Por su parte, existen cuatro tipos de modalidad:

Modalidad	Descripción
0,1	Establece una interrelación en donde la entidad puede tener/contener ningún o un elemento de la otra entidad.
1,1	Establece una interrelación en donde la entidad debe tener/contener un elemento de la otra entidad.
0,n	Establece una interrelación en donde la entidad puede tener/contener ningún o más de un elemento de la otra entidad.
1,n	Establece una interrelación en donde la entidad puede tener/contener uno o más de un elemento de la otra entidad.

**El diccionario de datos** es una herramienta que define una gramática para describir el contenido de los elementos de información. Esta gramática casi formal sirve para describir el contenido de los objetos definidos durante el análisis estructurado.

Carácter(es)	Descripción
=	está compuesto de
+	y
( )	optativo (puede estar presente o ausente)
{ }	iteración
[ ]	seleccionar una de varias alternativas
* *	comentario
@	identificador (campo clave) para un almacén
	separa opciones alternativas en la construcción

Símbolos del Diccionario de Datos

Este diccionario representa características de cada objeto de datos y elementos de control. El diccionario de datos contiene los siguientes elementos:

- Nombre del elemento de datos.
- Alias para el nombre principal.
- Dónde se usa / cómo se usa.
- Descripción del contenido.
- Información adicional.

El diccionario de datos debe contener las definiciones de todos los datos mencionados en el DFD. Existen dos tipos de datos:

- **Los datos compuestos** (datos que pueden ser divididos, como el nombre completo de una persona) se definen en términos de sus componentes.
- **Los datos elementales** (datos que no pueden ser divididos) se definen en términos del significado de cada uno de los valores que puede asumir.



# VIDEO



Te invitamos a ver el siguiente video:





## MODELADO DE COMPORTAMIENTO

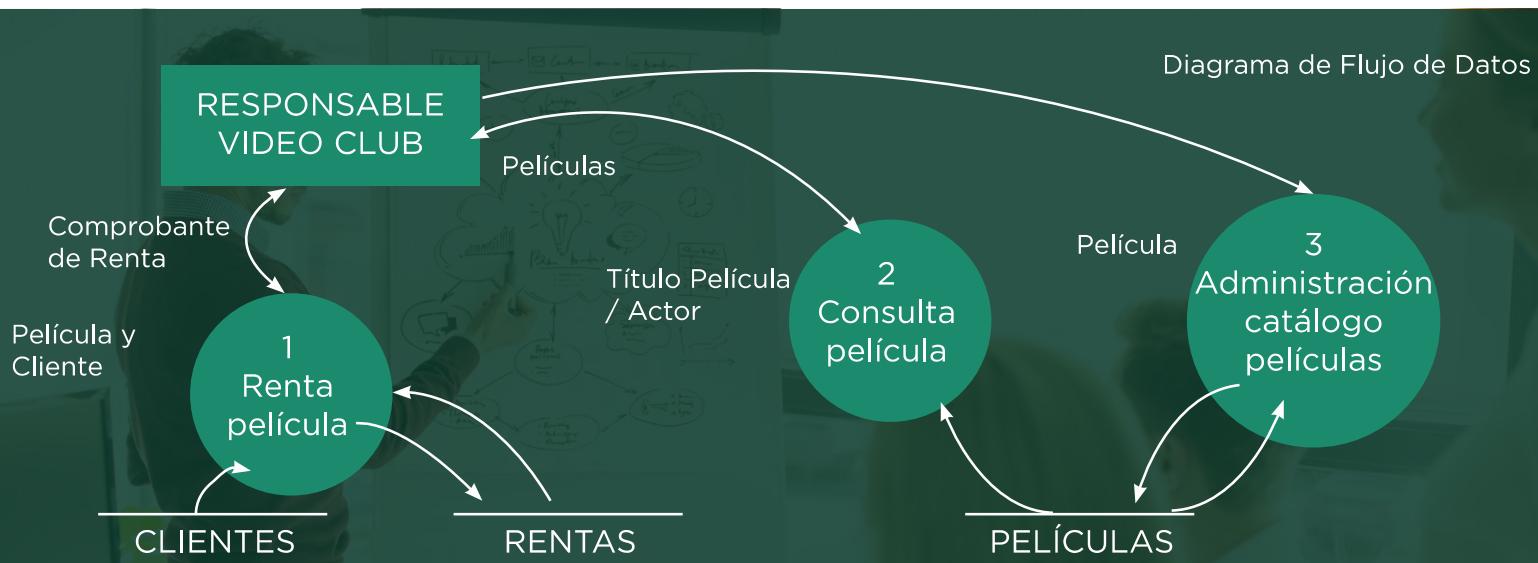
El objetivo del comportamiento es modelar el aspecto dinámico interno del sistema.

Para modelar este aspecto, se utilizan los **diagramas de flujo de datos**. Estos sirven para modelar el aspecto dinámico, tanto de un sistema manual como de un sistema automatizado. De la misma manera que lo hace un diagrama de contexto.

## MODELADO DE COMPORTAMIENTO

En los Diagramas de Flujo de Datos (DFD) se emplean los símbolos de una burbuja. Estas representan al proceso en sí.

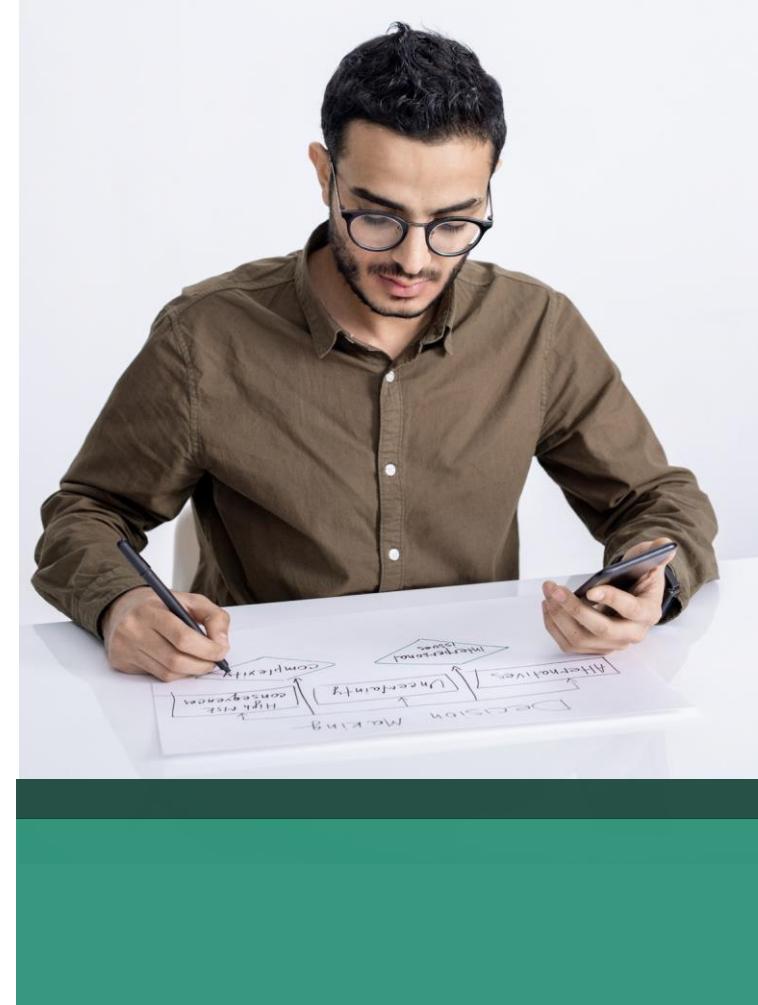
Por su parte, un rectángulo representa un agente externo; y una flecha representa al flujo.



En estos diagramas se incluye un nuevo símbolo, el cual está compuesto por dos líneas horizontales paralelas. Este se conoce como representación (Yourdon).

Las líneas horizontales paralelas se utilizan para modelar un **almacén** (los almacenes son repositorios de datos).

### LIBROS



## Tipos de flujo de datos

- **Flujo de actualización.** Indica la modificación de la información de un almacén, bien sea para crear o borrar ocurrencias de una entidad o relación del almacén o para modificar el valor de un atributo.



- **Flujo de diálogo.** Representa un flujo de consulta y de actualización que no tienen relación directa entre sí.



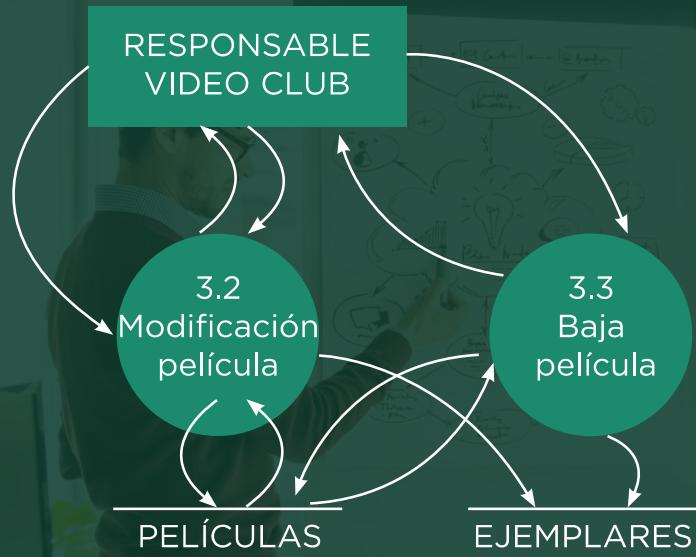
Este flujo puede aparecer para resaltar la relación existente entre dos flujos de datos (**par de diálogo**).



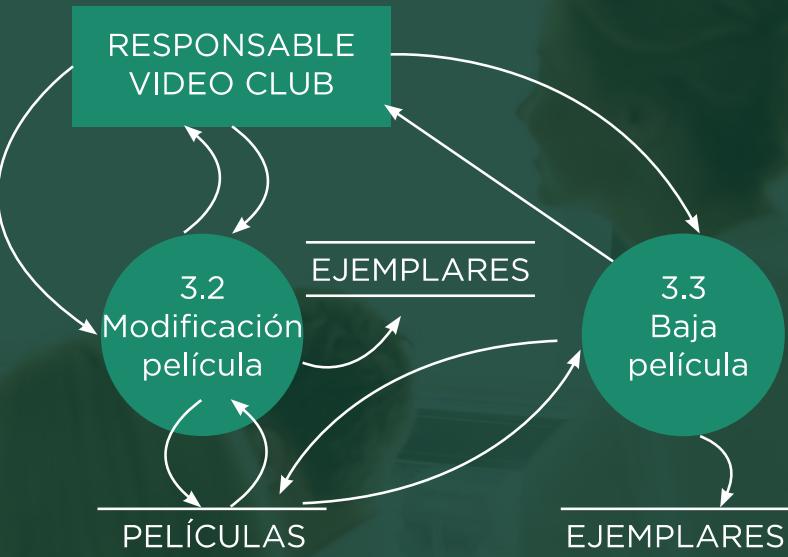
## MODELADO DE COMPORTAMIENTO

En algunas situaciones puede presentarse un flujo que atraviesa otros flujos. Esto no es recomendable, ya que dificulta la lectura del diagrama y este puede llegar a convertirse en una *telaraña*, por así decirlo.

**Incorrecto**



**Correcto**



## MODELADO DE COMPORTAMIENTO

Representación gráfica de los componentes de un DFD para distintas metodologías:

	Yourdon, DeMarco	Gane y Sarson	SSADM MÉTRICA
Flujo de Datos			
Procesos			
Almacén de Datos			
Entidades Externas			

Existe una extensión para los diagramas de flujo de datos denominada **diagramas de flujo de control (DFC)**.

Estos diagramas se utilizan cuando se requiere modelar un sistema en tiempo real. Por ejemplo, un software para controlar el tráfico de un sistema de vuelo o para monitorear el ritmo cardiaco.

### Un DFC tiene dos propósitos:

- Indica cómo se transforman los datos a medida que se avanza en el sistema.
- Representa las funciones que transforman el flujo de datos.

En la especificación de proceso se encuentra un descripción de cada función representada en el DFD.



## MODELADO DE COMPORTAMIENTO



En los DFC se utilizan dos símbolos adicionales a los DFD:

- Burbuja punteada.
- Flecha punteada.

■ **Burbuja punteada.** Representa un proceso de control que tiene como propósito coordinar a otros procesos que no son de control. Lo único que puede entrar o salir de un proceso de control son flujos de control.

■ **Flecha punteada.** Representa flujos de control que tienen como propósito indicarle a otro proceso algún estado o situación que requiera conocer. Se dividen, a su vez, en:

- Flechas de entrada.
- Flechas de salida.



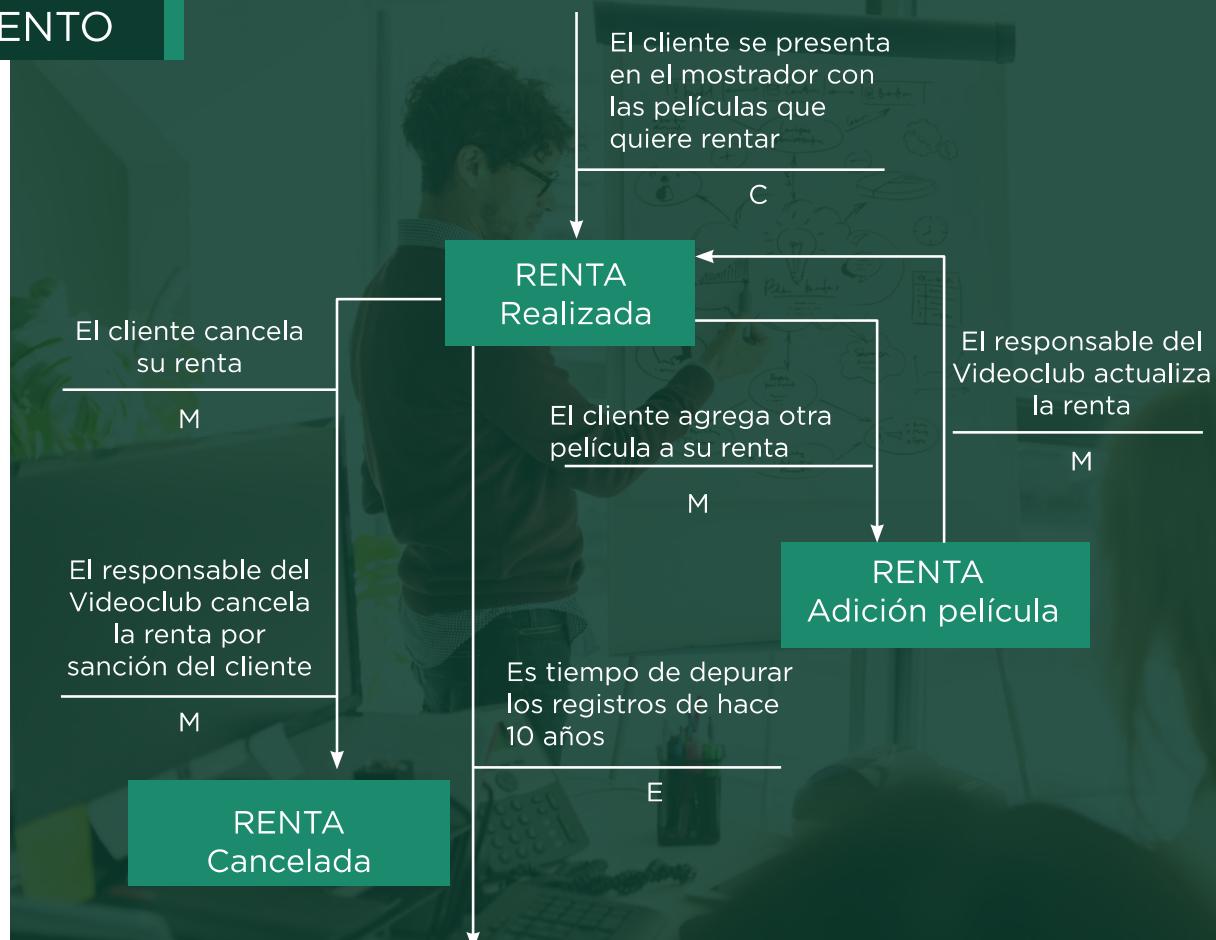
### Creación de un modelo de flujo de control (DFC):

- Se toma como base el DFD.
- Se eliminan flechas de flujo de datos.
- Se añaden sucesos y elementos de control (flechas con líneas discontinuas) y ventanas (barras verticales) a las especificaciones de control.

# MODELADO DE COMPORTAMIENTO

Otro diagrama que se utiliza para modelar el aspecto dinámico interno del sistema es el **Diagrama de Transición de Estados (DTE)**.

En este diagrama se busca representar los diferentes estados que puede tener una entidad del Diagrama de Entidad-Relación.



## Símbolos del Diagrama de Transición de Estados

Nombre	Figura
Rectángulo (estado)	
Rectángulo vacío (estado final)	
Flechas evento de transición	

**Nota.** Cabe mencionar que los estados de las entidades están conformados por los valores que tienen las entidades en un momento dado.

El DTE representa el comportamiento de un sistema que muestra los estados y los sucesos que hacen que el sistema cambie su estado.

Otros tipos de representación del comportamiento son los llamados diagramas de secuencia.





# ACTIVIDAD 1

Te invitamos a realizar la siguiente actividad:

Presiona el botón para descargar la actividad:



Presiona el botón para entregar la actividad:



# CONCLUSIÓN



Un modelo es una representación de la realidad. Durante el proceso del diseño del sistema estructurado, sus partes se dividen en dinámica, con los modelos ambiental y de comportamiento; y en estática, con el modelado de datos o de información. No se define un proceso lineal para el inicio del modelado, ya que este se realizará en función de los datos que se tienen a disposición, así como de su comprensión.

El modelado ambiental se representa mediante los Diagramas de Contexto (DC). El modelado de datos, por su parte, está representado por los Diagramas Entidad-Relación (DER) y el diccionario de datos. En cuanto al modelado de comportamiento, este se figura mediante un Diagrama de Flujo de Datos (DFD), un Diagrama de Flujo de Control (DFC) o un Diagrama de Transición de Estados (DTE)

# ¡FELICIDADES!



Acabas de concluir la segunda unidad de tu curso *Análisis y Diseño de Sistemas*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.



# UNIDAD 3

## DISEÑO DE SISTEMAS ORIENTADOS A OBJETOS





# U3

# TEMARIO

**3.1**



Arquitectura de Sistema  
Orientado a Objetos

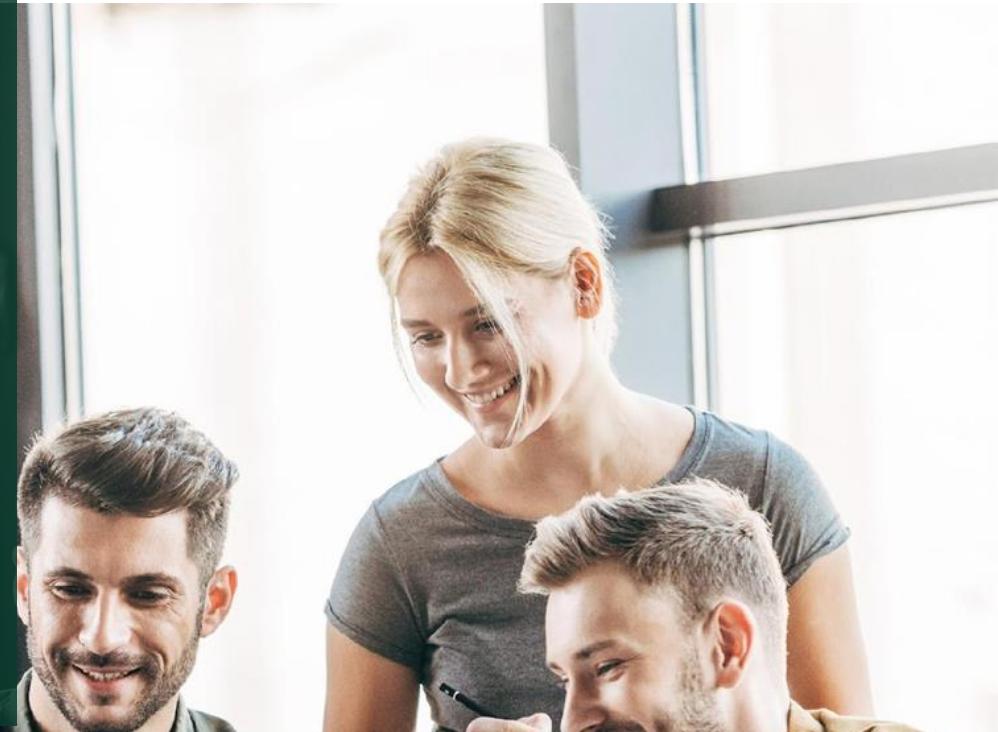
**3.2**



Patrones de Diseño  
Orientado a Objetos

# INTRODUCCIÓN

En esta tercera unidad identificarás conceptos generales sobre la arquitectura de un sistema a partir de su representación en vistas. En la segunda parte conoceremos los patrones de diseño de acuerdo a los sistemas orientados a objetos.



# COMPETENCIAS A DESARROLLAR



1. El alumno será capaz de reconocer el proceso de una arquitectura a partir de su representación en vistas, de acuerdo a los sistemas orientados a objetos.

2. El alumno será capaz de identificar la clasificación de los patrones de diseño y relacionar su uso de acuerdo a los sistemas orientados a objetos.

# ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

Existen numerosos métodos de diseño orientado a objetos: Booch, Yourdon-Coad, Martín, Shlaer & Mellor, Rumbaugh, por citar algunos.

Un proyecto de software orientado a objetos se compone de las siguientes etapas:

- Análisis Orientado a Objetos (AOO).
- Diseño Orientado a Objetos (DOO).
- Programación Orientada a Objetos (POO).





## ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

El diseño orientado a objetos se define como un diseño de sistemas que utiliza objetos autocontenidos y clases de objetos.

Difiere considerablemente del diseño estructurado, ya que en el DOO no se realiza un problema en términos de tareas (subrutinas) ni en términos de datos, sino que se analiza el problema como un sistema de objetos que interactúan entre sí.



Dentro del paradigma de la orientación a objetos el DOO es, por mucho, más complejo que el diseño estructurado clásico. En este, lo que se busca es crear un diseño genérico y abierto; no cerrado y concreto.

## ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

Por su parte, el objetivo del AOO es modelar la semántica del problema en términos de objetos distintos, pero relacionados.

El objetivo del DOO conlleva reexaminar. Es decir, refinar, extender y reorganizar las clases del dominio del problema para mejorar su reutilización y tomar ventaja de la herencia.





## ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

Características principales del diseño orientado a objetos:

- Los objetos son abstracciones del mundo real o entidades del sistema que se administran entre ellas mismas.
- Los objetos son independientes, y encapsulan el estado y la representación de información.

# ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

- La funcionalidad del sistema se expresa en términos de servicios de los objetos.
- Las áreas de datos compartidas son eliminadas. Los objetos se comunican mediante paso de parámetros.
- Los objetos pueden estar distribuidos y pueden ejecutarse en forma secuencial o en paralelo.



### Ventajas del diseño orientado a objetos:

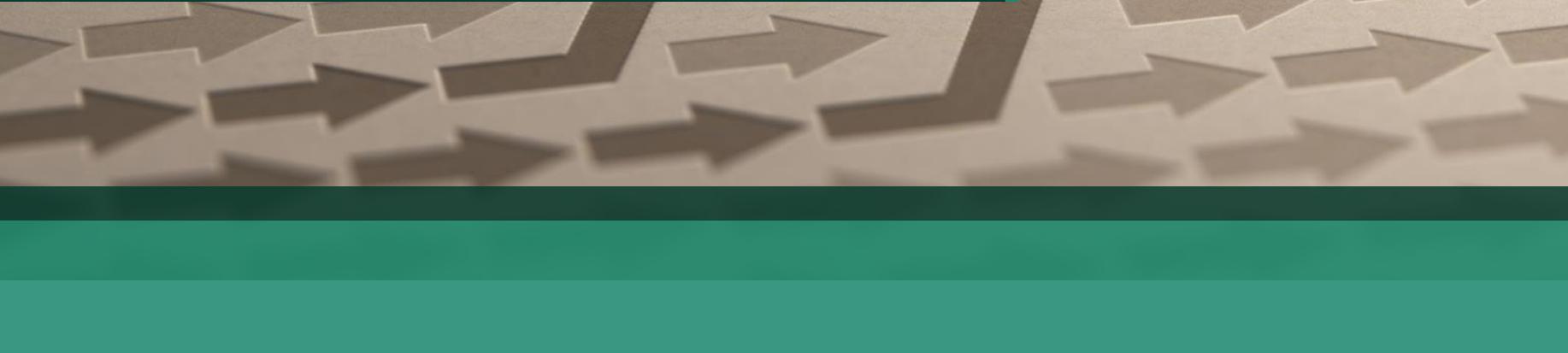
- Fácil de mantener. Los objetos representan entidades autocontenidas.
- Los objetos son componentes reutilizables.
- Para algunos sistemas, puede haber un mapeo obvio entre las entidades del mundo real y los objetos del sistema.

## Componentes del diseño orientado a objetos:

- La identificación de objetos, sus atributos y servicios.
- La organización de objetos dentro de una jerarquía.
- La construcción de descripciones dinámicas de objetos que muestran cómo se usan los servicios.
- La especificación de interfaces de objetos.



## ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

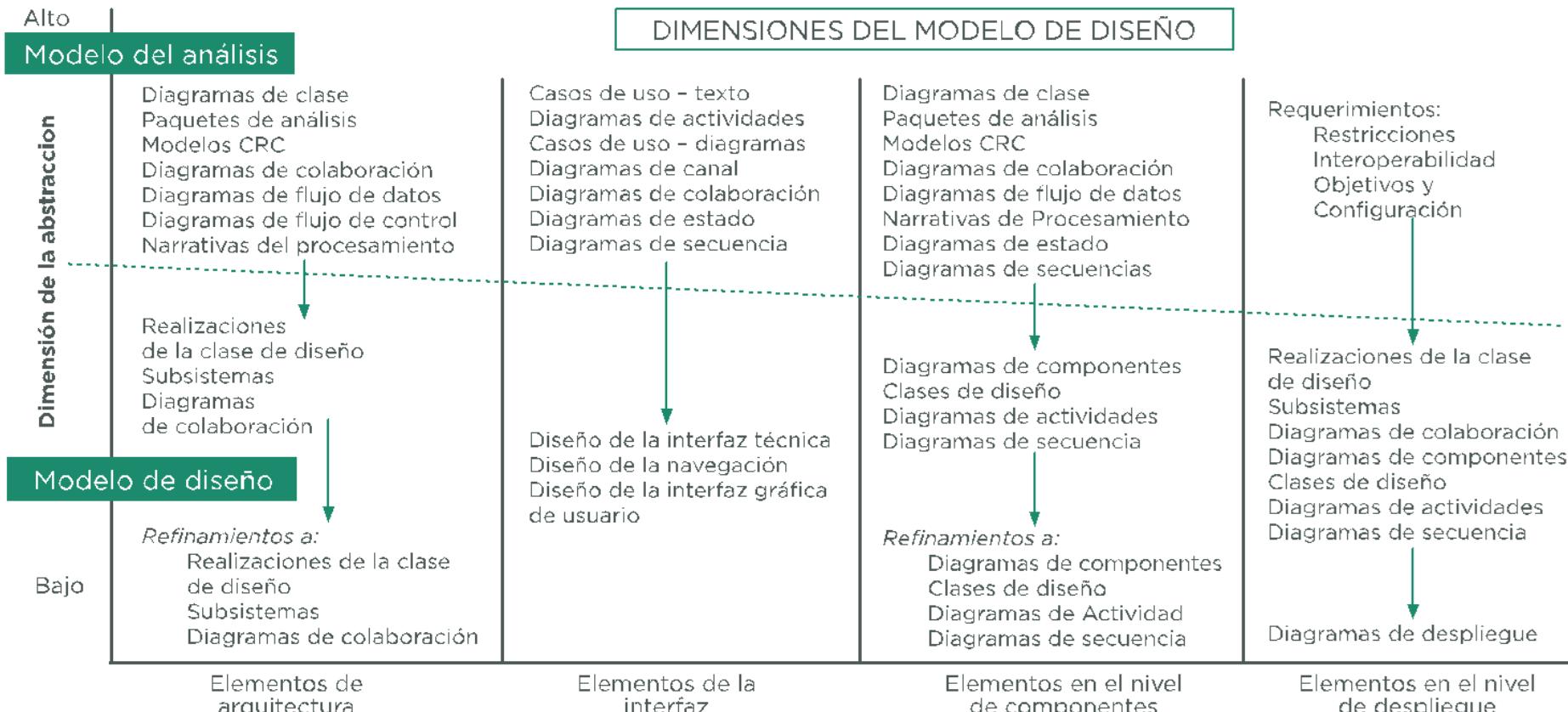


En la siguiente figura, la línea punteada indica la frontera entre los modelos de análisis y de diseño.

En ciertos casos, es posible hacer una distinción clara entre ambos modelos.

En otros, el modelo de análisis se mezcla poco a poco con el de diseño, y la distinción es menos obvia.

# ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS





## ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

Los modelos del diseño pueden ser estáticos y dinámicos:

- **Estáticos.** Estructura de subsistemas y/o clases, y sus relaciones. El modelo de objetos contiene diagramas de objetos.
- **Dinámicos.** Se describen las estructuras que muestran la interacción entre objetos. Ejemplos de UML: diagramas de secuencia, diagramas de estado.

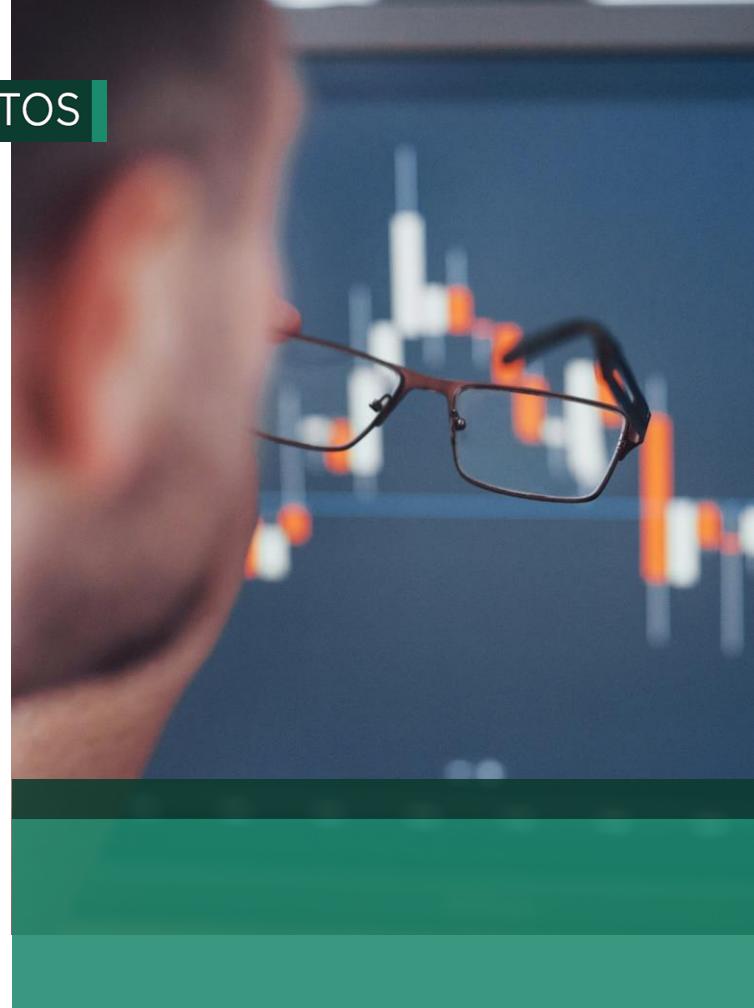
## ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

La arquitectura de software es una abstracción que muestra un marco de referencia que sirve de guía en la construcción de un software. Esta se construye con diferentes diagramas que ayudan a mostrar diferentes vistas del sistema.

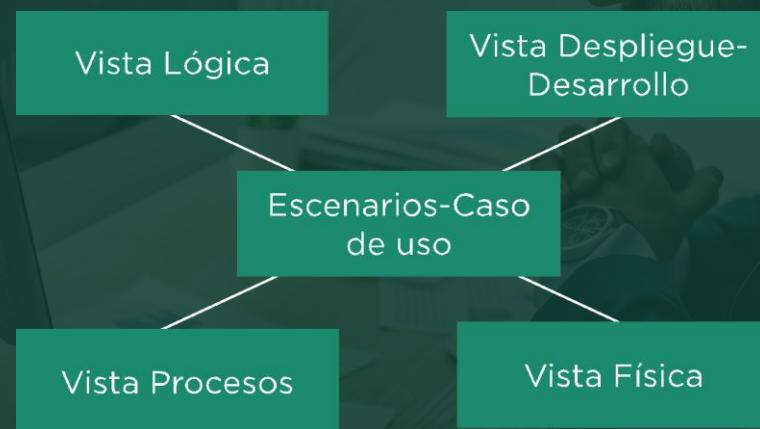


Las **vistas** se utilizan para representar la arquitectura del software en un solo diagrama.

Una vista se define como la forma de presentar un modelo que describe un sistema de manera completa desde un cierto punto de vista.



# ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS



Para describir una arquitectura de software, utilizamos un modelo compuesto por múltiples vistas.

**Arquitectura 4+1.** Describe la arquitectura de un sistema a partir de su representación en vistas. El modelo que se propone se compone de cinco principales tipos de vistas:

## Vista Lógica

- Dirigida a usuarios finales.
- Factores. Implica el modelo de objetos del diseño. Representa la funcionalidad y estructura del sistema y lo que debe hacer.
- Se pueden incluir los diagramas de clases, de comunicación o de secuencia.

Usuario Final  
■ Funcionalidad

Vista Lógica

- Diagrama de clases
- Diagrama de comunicación
- Diagrama de secuencias

# ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

El estilo que usamos para la arquitectura lógica es un estilo orientado a objetos.

La directriz principal para el diseño de la vista lógica es tratar de mantener un único modelo. Debe existir un objeto coherente en todo el sistema para evitar la prematura especialización de las clases y los mecanismos por sitio o por procesador.

La notación para la vista lógica se deriva de la **notación de Booch**.



## Vista de Procesos

- Dirigida a integradores.
- Factores. Requerimientos no funcionales. Se incluyen los aspectos arquitectónicos de sincronización, tiempo de respuesta y seguridad en el envío de solicitudes. Representa el flujo del trabajo del sistema en tiempo de ejecución.
- Se representa con los **diagramas de actividad**.

Integradores

- Rendimientos
- Escalabilidad

Vista Procesos

- Diagrama de actividad

## ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

En la arquitectura de procesos se tienen en cuenta algunos requisitos no funcionales, tales como el rendimiento y disponibilidad. Se abordan aspectos de concurrencia y distribución, de la integridad del sistema, de tolerancia a fallos, entre otras.

La notación para la vista de procesos se expande de la notación de Booch. Esta se centra solamente en los elementos arquitectónicamente relevantes.



## Vista Física

- Dirigida a ingenieros o consultores de infraestructura.
- Factor. Describe la asignación del software y hardware, así como sus aspectos distribuidos, la topología, dependiendo de las necesidades y de los atributos de calidad.
- Se representa con los **diagramas de despliegue**.

Ingenieros  
de sistemas

- Topología

Vista Física

- Diagrama  
de despliegue

# ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

**Notación para la arquitectura física.** Los diagramas físicos pueden tornarse muy confusos en grandes sistemas y, por lo tanto, toman diversas formas, con o sin el mapeo de la vista de procesos.



## Vista Despliegue-Desarrollo

- Dirigida a desarrolladores.
- Describe la organización estática del software en el medio ambiente de su desarrollo. Se ocupa de la gestión del software.
- Se pueden incluir los diagramas de componentes y de paquetes.
- Es común utilizar un representación en capas.

Programadores

- Gestión del Software

Vista de desarrollo

- Diagrama componentes
- Diagrama de paquetes

# ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

La vista de desarrollo de un sistema se representa en diagramas de módulos o subsistemas que muestran las relaciones.

**Notación.** Tal como se muestra en la siguiente figura, se usa una variante de la notación de Booch. Esta limita a aquellos ítems relevantes para la arquitectura.



## Vista de Escenario

- Dirigida a todos los interesados del sistema.
- Factores. Evidenciar qué funcionalidades, a grandes rasgos, poseerá y serán implementadas dentro del sistema. Tiene la función de unir y relacionar las otras cuatro vistas.
- Esta vista va a ser representada por **diagramas de casos de uso de UML**.

Todos los  
Stakeholders

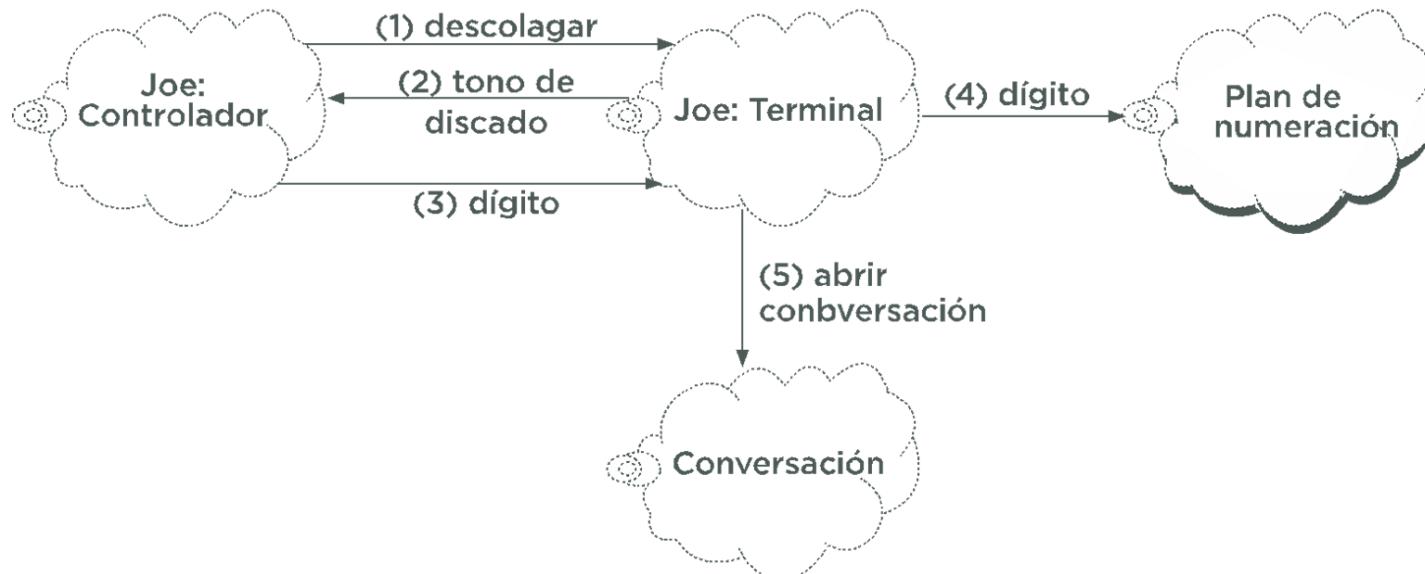
Vista de Escenario

■ Diagrama de caso de uso

## ARQUITECTURA DE SISTEMA ORIENTADO A OBJETOS

La notación es muy similar a la vista lógica para los componentes, pero usa los conectores de la vista de procesos para la interacción entre objetos.

Las instancias de objetos se denotan con **líneas sólidas**.



# VIDEO



Te invitamos a ver el siguiente video:



# PATRONES DE DISEÑO ORIENTADO A OBJETOS

Un **patrón** describe un problema que ocurre de forma iterativa en nuestro entorno. También describe la solución a dicho problema, de tal forma que podemos usar la misma solución todas las veces que sea necesario. Estas soluciones deberán estar basadas en la experiencia. Es decir, que se ha demostrado que funcionan.

La solución a algunos problemas del desarrollo viene en forma de bloques de código y de esqueletos de una solución. A estos bloques de código se les conoce como patrones.



### Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables.
- Evitar la reiteración en la búsqueda de soluciones a problemas.
- Formalizar un vocabulario común.
- Estandarizar el modo de diseño.
- Facilitar el aprendizaje.

### Un patrón tiene cuatro elementos esenciales:

- **Nombre.** Describe en una o dos palabras un problema de diseño.
- **Problema.** Explica el problema y su contexto. A veces el problema incluye una serie de condiciones.
- **Solución.** Describe los elementos que constituyen el diseño, sus relaciones, responsabilidades y colaboraciones.
- **Consecuencias.** Son los resultados. Son las ventajas y los inconvenientes de aplicar el patrón. Son fundamentales para evaluar.

# PATRONES DE DISEÑO ORIENTADO A OBJETOS

Un patrón de diseño puede considerarse como un documento que define una estructura de clases que aborda una situación particular.

En la actualidad, existen muchos patrones de diseño. Por este motivo, suelen agruparse según su propósito:

## Clasificación de Patrones

Patrón de Creación

Patrón de Estructura

Patrón de Comportamiento

### Patrón de Creación

Concierne al proceso de creación de objetos. Facilita la tarea de creación de nuevos objetos, de tal forma que el proceso de creación pueda ser desacoplado de la implementación del resto del sistema.

Los patrones creacionales facilitan la tarea de creación de nuevos objetos, al encapsular el proceso.

**Los patrones creacionales están basados en dos conceptos:**

- Encapsular el conocimiento acerca de los tipos concretos que nuestro sistema utiliza.
- Ocultar cómo estas implementaciones concretas necesitan ser creadas y cómo se combinan entre sí.





**Los patrones creacionales más conocidos son:**

- *Abstract Factory*
- *Factory Method*
- *Builder*
- *Singleton*
- *Prototype*
- *Lazy initialization*

## Patrón de Estructura

Este patrón trata la composición de clases y/o objetos. Su objetivo es desacoplar las interfaces e implementar clases y objetos.

Los patrones estructurales especifican la forma en que unas clases se relacionan con otras.

**Los patrones de estructura más conocidos son:**

- *Adapter*
- *Bridge*
- *Composite*
- *Decorator*
- *Facade*
- *Flyweight*
- *Proxy*



## Patrón de Comportamiento

Los patrones de comportamiento gestionan algoritmos, relaciones y responsabilidades entre objetos.



**Los patrones de comportamiento más conocidos son:**

- *Chain of responsibility*
- *Command*
- *Iterator*
- *Mediator*
- *Memento*
- *Observer*
- *State*
- *Strategy*
- *Template Method*
- *Visitor*



# PATRONES DE DISEÑO ORIENTADO A OBJETOS

Antes de seleccionar un patrón de diseño, se debe asegurar que este sea adecuado para el caso particular.

Además, se debe tener en cuenta que, si se deben hacer solo unos cambios mínimos, puede ser señal de que esto no sea lo más adecuado para su desarrollo.





## PATRONES DE DISEÑO ORIENTADO A OBJETOS

Otro concepto es el de **antipatrones**, que hace referencia a los errores que comúnmente suelen ocurrir al intentar solucionar problemas conocidos.

No es necesario memorizar cómo funcionan todos los patrones, pero sí es importante saber de su existencia para así recurrir a ellos en caso necesario.



# ACTIVIDAD 2

Te invitamos a realizar la siguiente actividad:

Presiona el botón para descargar la actividad:



Presiona el botón para entregar la actividad:



# CONCLUSIÓN



La importancia del modelo de arquitectura por vistas 4+1 es que, a la hora de desarrollarlo, se hace énfasis en proporcionar una mayor información del mismo a través de diagramas que nos ayuden a tener una mejor descripción. Juntar sus cinco vistas implica una mejor comunicación para, en consecuencia, tener una mejor comprensión del diseño.

Por otro lado, se cuenta con una lista muy variada de patrones que nos dan la oportunidad de crear un sistema con estructuras probadas y funcionales, de acuerdo a la clasificación. Entre los patrones reconocidos, se encuentran el patrón de estructura, de diseño y de comportamiento. Su mayor complejidad radica en saber cuándo utilizarlos. Cuestión que solo la práctica permitirá.

# ¡FELICIDADES!

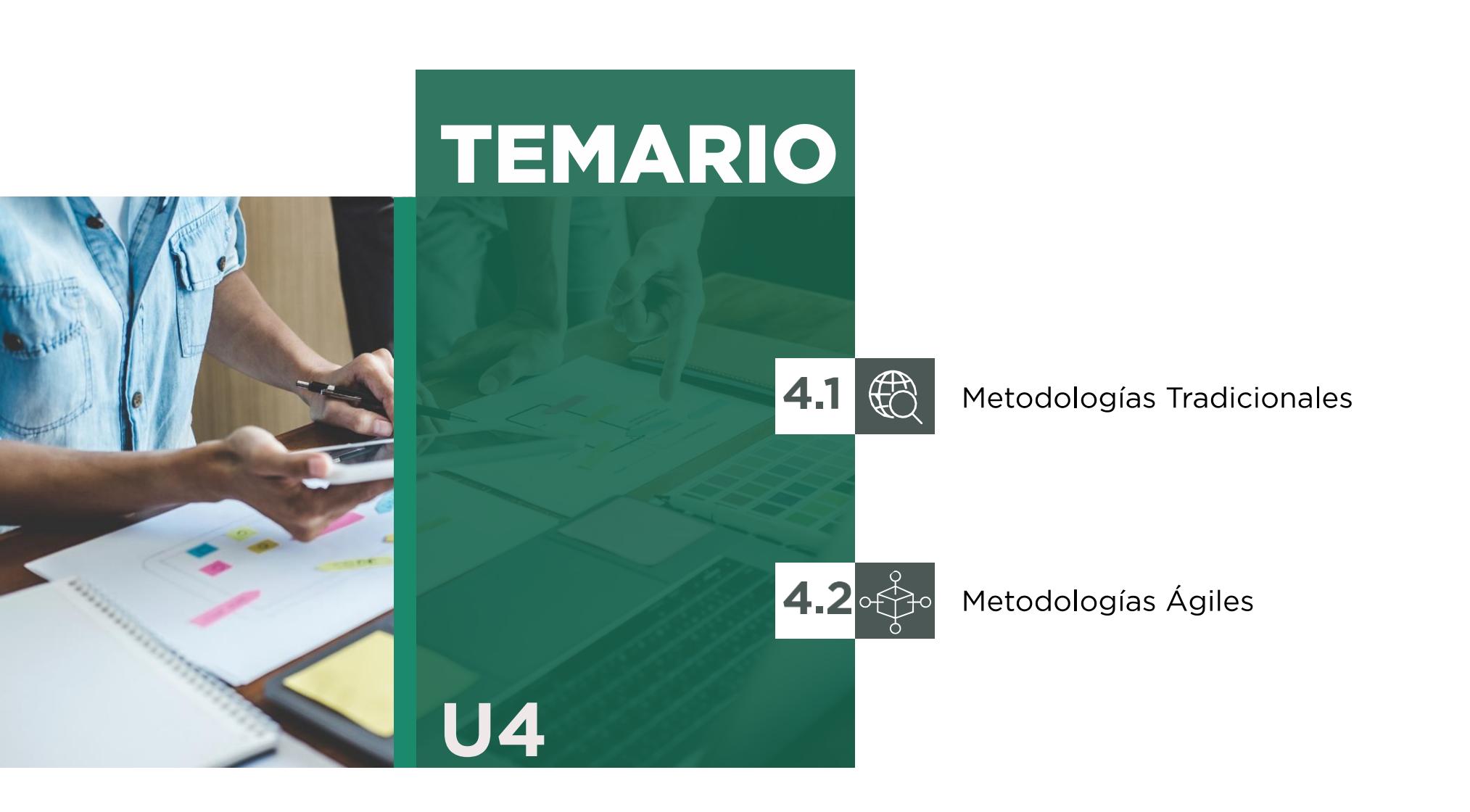


Acabas de concluir la tercera unidad de tu curso *Análisis y Diseño de Sistemas*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.

# UNIDAD 4

## METODOLOGÍAS DE DESARROLLO DE SISTEMAS





# TEMARIO

## U4

**4.1**



Metodologías Tradicionales

**4.2**



Metodologías Ágiles

# INTRODUCCIÓN

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo.

En esta cuarta unidad conocerás metodologías para el desarrollo de software, separadas en dos grandes grupos: las metodologías tradicionales, pesadas o no ágiles y las metodologías ágiles.



# COMPETENCIAS A DESARROLLAR



**1.**

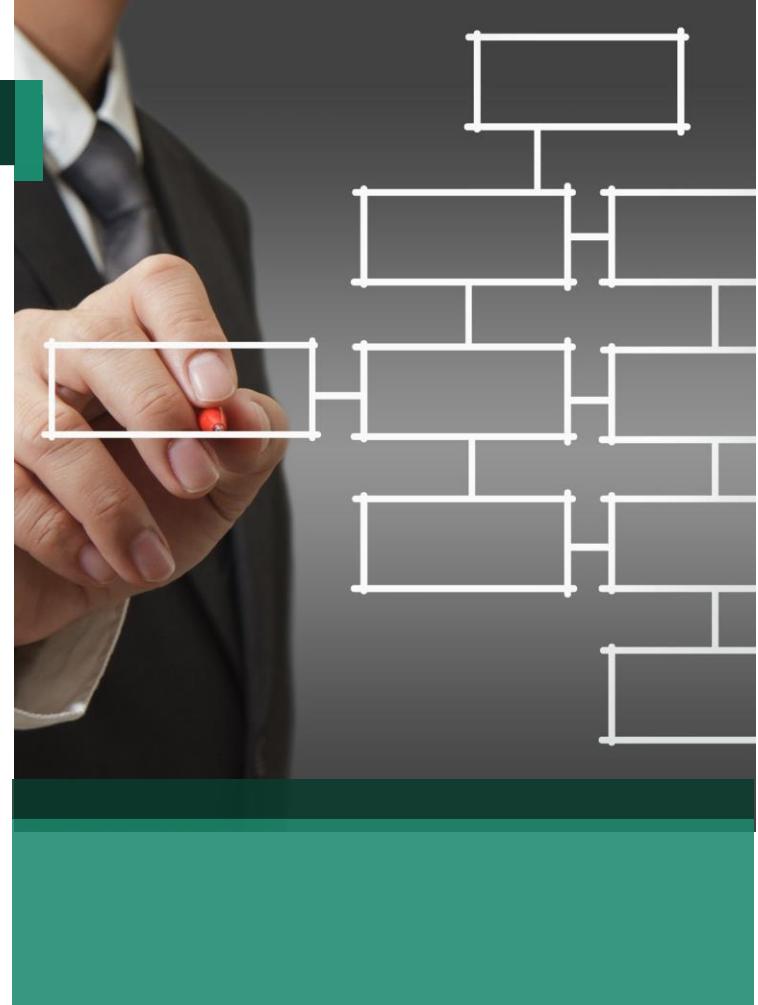
El alumno identificará las funciones y propuestas metodológicas tradicionales para el desarrollo de sistemas durante el proceso de desarrollo.

**2.**

El alumno conocerá las ventajas y los tipos de metodologías ágiles para el desarrollo del sistema.

# METODOLOGÍAS TRADICIONALES

La metodología es una de las etapas específicas de un trabajo o proyecto que parte de una posición teórica y conlleva a una selección de técnicas concretas o métodos acerca del procedimiento para el cumplimiento de los objetivos.





## METODOLOGÍAS TRADICIONALES

El objetivo general de la puesta en práctica de una metodología de software es **construir un producto de alta calidad de una manera oportuna**.

La selección de una metodología adecuada implica un conjunto de principios fundamentales que se deben seguir y cumplir.

## METODOLOGÍAS TRADICIONALES

Para conseguir el objetivo de construir productos de alta calidad dentro de la planificación, las metodologías, en general, emplean una serie de prácticas para:

- Entender el problema.
- Diseñar una solución.
- Implementar la solución correctamente.
- Probar la solución.
- Gestionar las actividades anteriores para conseguir alta calidad.



## METODOLOGÍAS TRADICIONALES

La utilización de la metodología adecuada representa un proceso formal que incorpora una serie de métodos bien definidos para el análisis, diseño, implementación y pruebas de software y sistemas.

Además, abarca una amplia colección de métodos y técnicas de gestión de proyectos para el aseguramiento de la calidad y la gestión de la configuración del software.





## METODOLOGÍAS TRADICIONALES

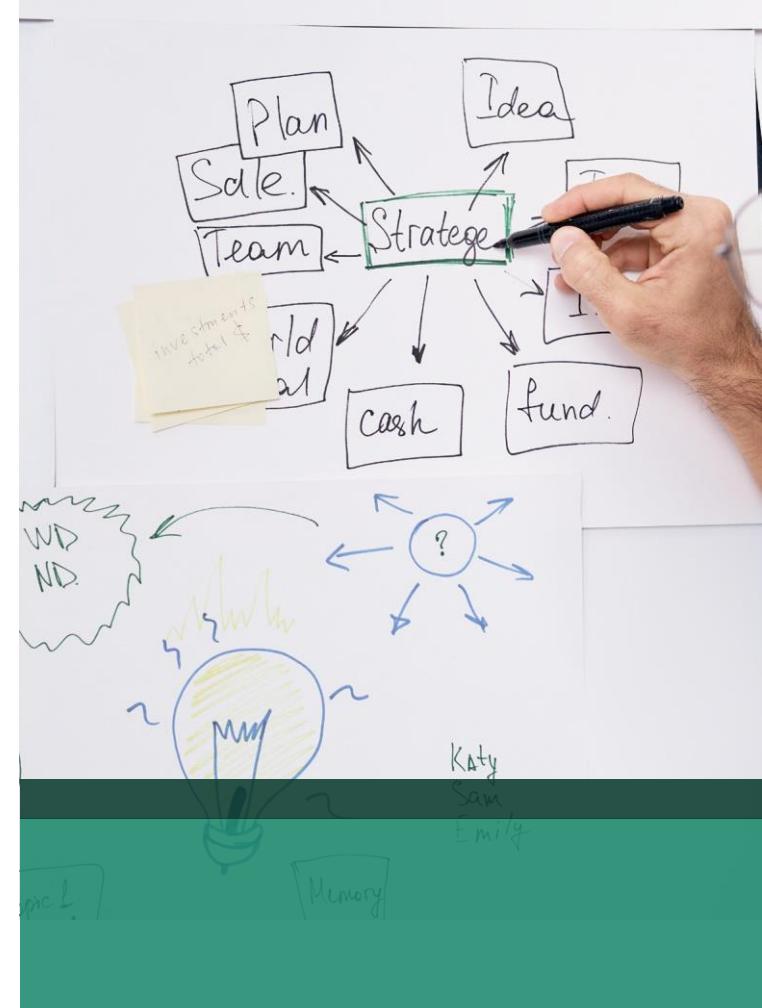
Entre los **elementos** que forman parte de una metodología, se pueden destacar:

- **Fases.** Tareas a realizar en cada fase o etapa.
- **Productos.** E/S de cada fase, documentos.
- **Procedimientos y herramientas.** Apoyo a la realización de cada tarea.
- **Criterios de evaluación del proceso y del producto.** Saber si se han logrado los objetivos.

## METODOLOGÍAS TRADICIONALES

Las **metodologías tradicionales** son denominadas, a veces, de forma despectiva, como *metodologías pesadas*.

Centran su atención en llevar una documentación exhaustiva de todo el proyecto, así como su planificación y control del mismo, en especificaciones precisas de requisitos y modelado. Además, requieren cumplir con un plan de trabajo definido. Todo esto se busca en la fase inicial del desarrollo del proyecto.



## METODOLOGÍAS TRADICIONALES

Estas metodologías tradicionales imponen una disciplina rigurosa de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente.

Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar. Una vez que se encuentra todo detallado, comienza el ciclo de desarrollo del producto software.





### Características de las metodologías tradicionales:

- No se adaptan adecuadamente a los cambios.
- Sus costes son altos al implementar un cambio en el presupuesto.
- Falta de flexibilidad en proyectos donde el entorno es volátil.
- Enfoque predictivo, donde se sigue un proceso secuencial en una sola dirección y sin marcha atrás.

# METODOLOGÍAS TRADICIONALES

## Modelo en Cascada

Es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software. De tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior.

Bajo este modelo, los resultados no se pueden ver hasta muy avanzado el proyecto. En este sentido, cualquier cambio debido a un error puede suponer un gran retraso, además de un alto coste de desarrollo.



### Desventajas

- Los proyectos reales raramente siguen el flujo secuencial. Siempre hay iteraciones y se crean problemas en la aplicación del paradigma.
- Es difícil para el cliente establecer explícitamente al principio todos los requisitos.
- El cliente debe tener paciencia. Hasta llegar a las etapas finales del proyecto, el producto no estará disponible.



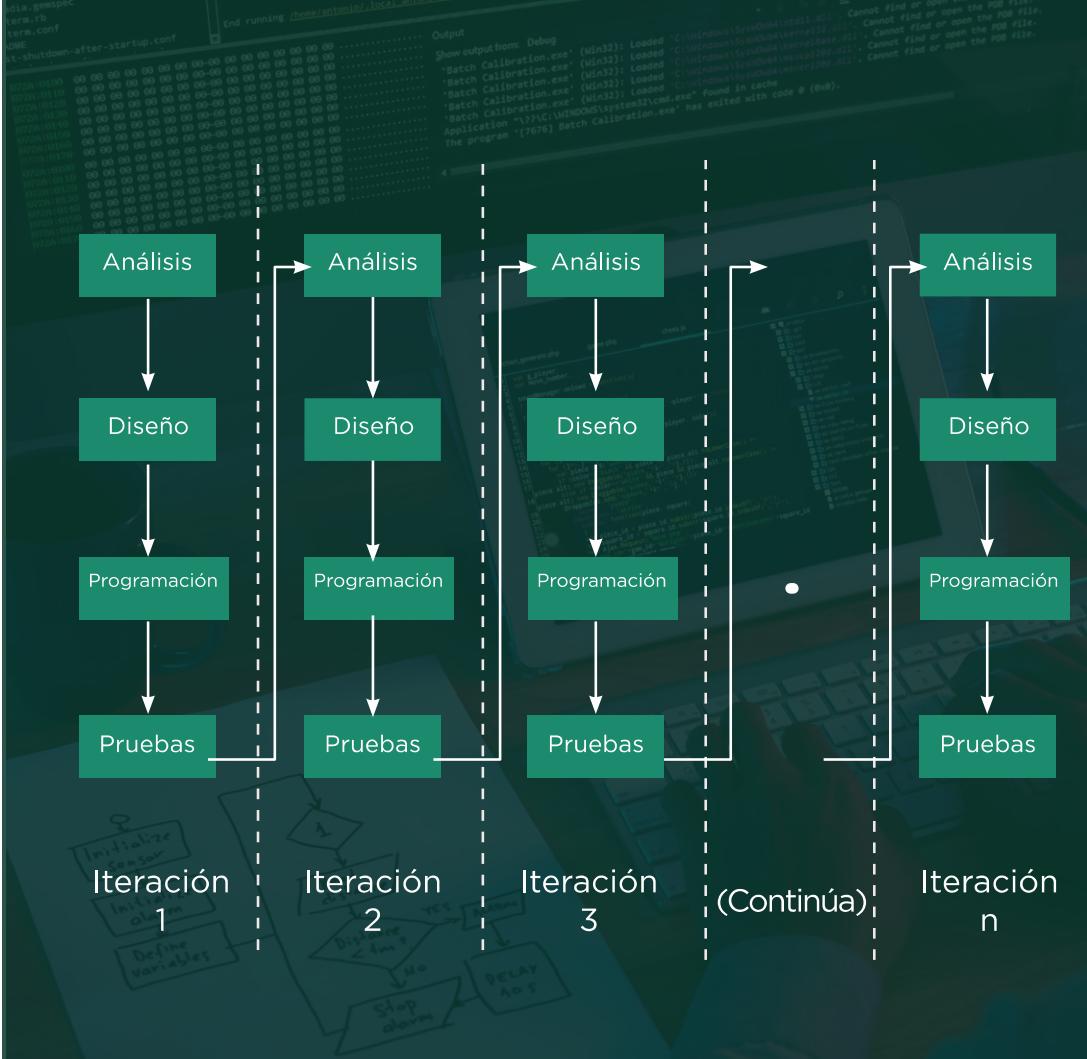
# METODOLOGÍAS TRADICIONALES

## Modelo Incremental o Iterativo y Creciente

Permite construir el proyecto en etapas incrementales, en donde cada una de estas agrega funcionalidad.

Estas etapas consisten en requerimientos, diseño, codificación, pruebas y entrega.

Permite entregar más rápidamente al cliente un producto, en comparación con el modelo en cascada. Además, reduce los riegos.



## Ventajas

- La solución se va mejorando de forma progresiva.
- Los clientes no esperan hasta el fin del desarrollo para utilizar el sistema.
- Los clientes pueden aclarar los requisitos que no tengan claros.
- Se disminuye el riesgo de fracaso de todo el proyecto.
- Las partes más importantes del sistema son entregadas primero.





## METODOLOGÍAS TRADICIONALES

- El progreso se puede medir en períodos cortos de tiempo.
- Resulta más sencillo acomodar cambios al acortar el tamaño de los incrementos.
- Se puede planear con base en la funcionalidad qué se quiere entregar primero.

### Desventajas

- Requiere de mucha planeación, tanto administrativa como técnica.
- Requiere de metas claras para conocer el estado del proyecto.
- Es un proceso de desarrollo de software creado en respuesta a las debilidades del modelo tradicional de cascada.

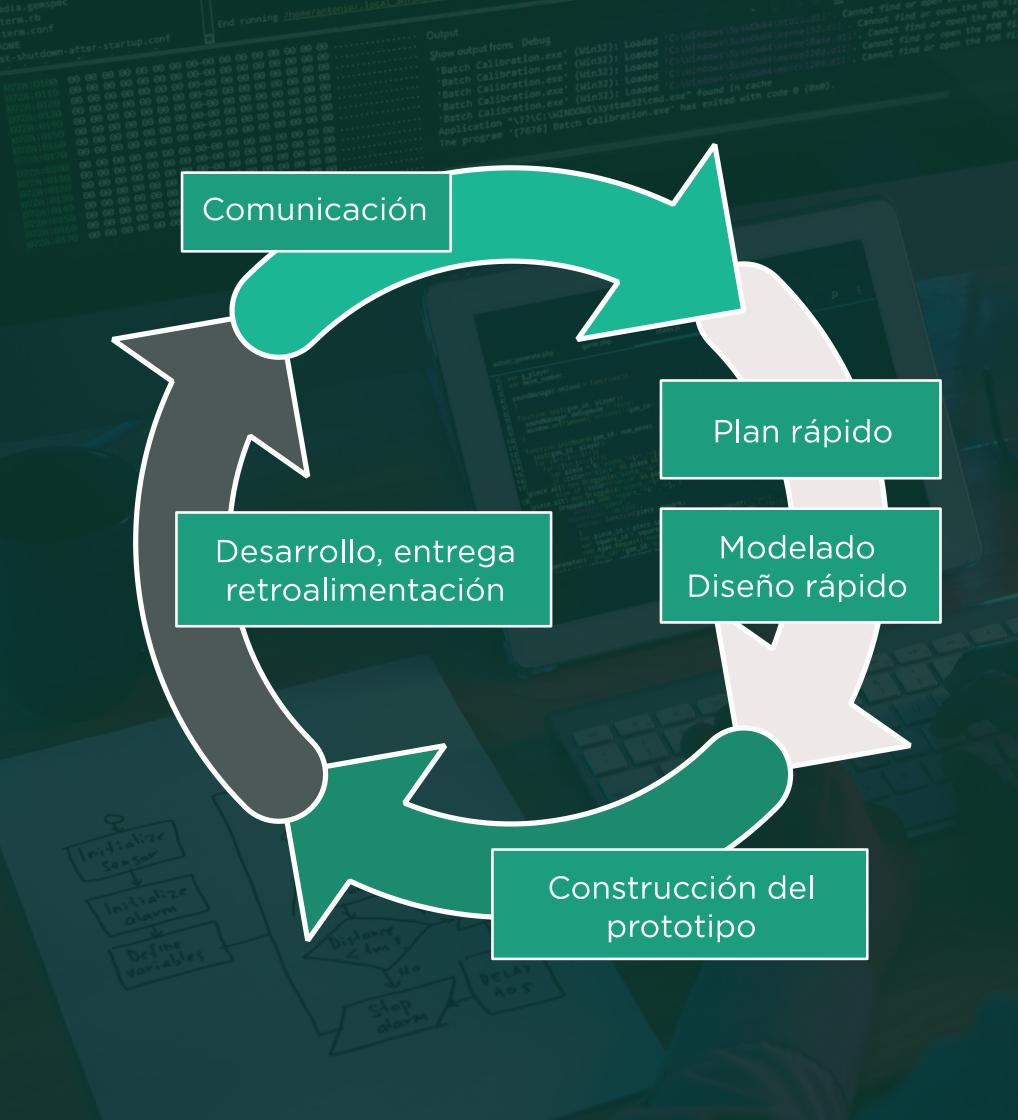


# METODOLOGÍAS TRADICIONALES

## Modelo Prototipado

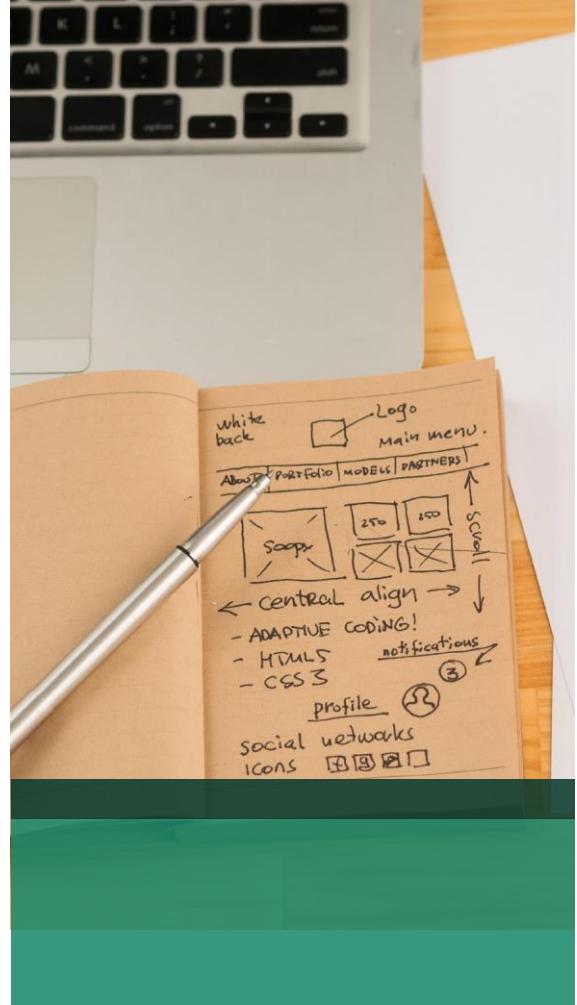
Un prototipo es una versión preliminar de un sistema de información con fines de demostración o evaluación.

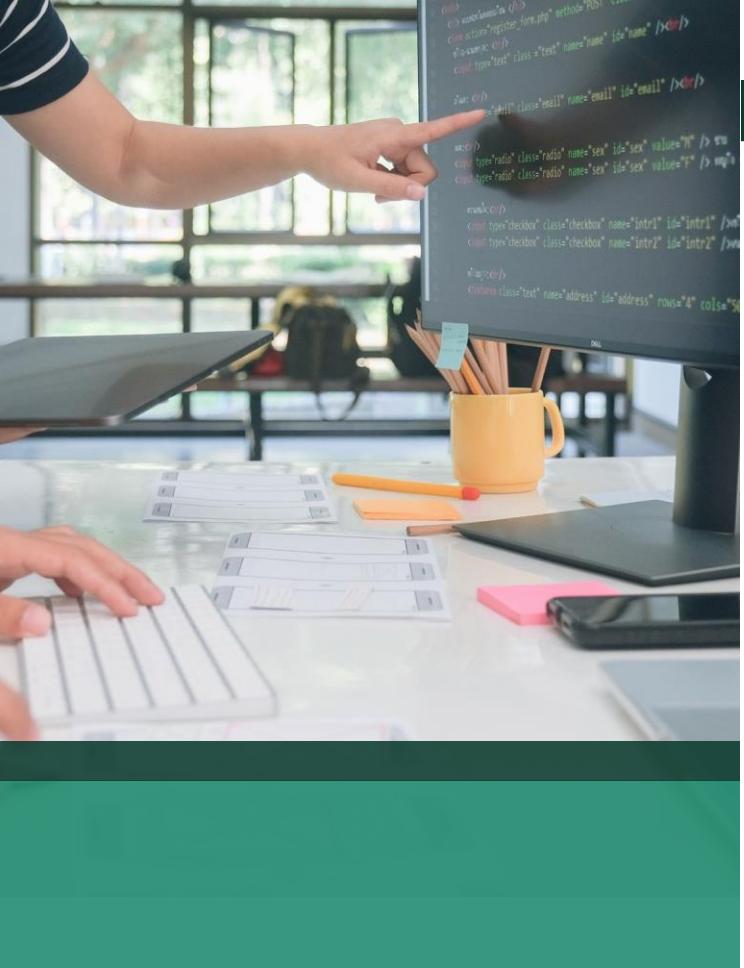
El prototipo de requerimientos es la creación de una implementación parcial de un sistema para el propósito explícito de aprender sobre los requerimientos del mismo. Un prototipo es construido de una manera rápida.



## Ventajas

- Útil cuando los requerimientos son cambiantes.
- Eficiente cuando no se conoce bien la aplicación.
- Útil cuando el usuario no se quiere comprometer con los requerimientos.
- De gran utilidad cuando se quiere probar una arquitectura o tecnología.
- Útil cuando se requiere rapidez en el desarrollo.
- Más eficiente que un método menos formal de desarrollo.





## METODOLOGÍAS TRADICIONALES

### Desventajas

- No se conoce cuándo se tendrá un producto aceptable.
- No se sabe cuántas iteraciones serán necesarias.
- Da una falsa ilusión al usuario sobre la velocidad del desarrollo.
- Se puede devolver el producto aun cuando no esté con los estándares requeridos.

# METODOLOGÍAS TRADICIONALES

## Modelo de Desarrollo Evolutivo (espiral)

Toma las ventajas del modelo de desarrollo en cascada y el de prototipos, pero añade el concepto de **análisis de riesgo**.

En este modelo, se definen cuatro actividades:

- Planificación.
- Análisis de riesgo.
- Ingeniería.
- Evaluación del cliente.





## METODOLOGÍAS TRADICIONALES

El esquema del ciclo de vida, para estos casos, puede representarse por un bucle en espiral. En esta, los cuadrantes son, habitualmente, fases de especificación, diseño, realización y evaluación (o conceptos y términos análogos).

En cada fase, el producto gana *madurez* (aproximación al final deseado). Hasta que en una iteración se logre el objetivo deseado y el producto sea aprobado, se terminan las iteraciones.

# VIDEO



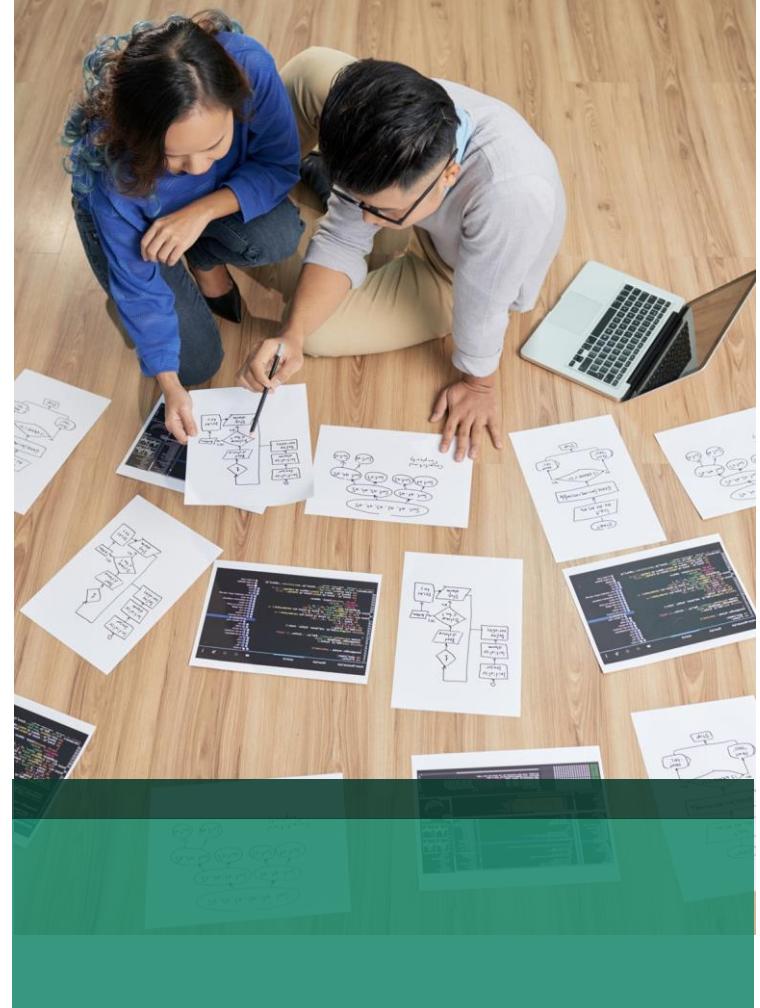
Te invitamos a ver el siguiente video:



# METODOLOGÍAS ÁGILES

Dentro de otro contexto tenemos a las metodologías ágiles. Estas surgen como alternativa a las tradicionales. Nos ayudan a reducir la probabilidad de fracaso por medio de dos aspectos fundamentales: retrasar las decisiones y la planificación adaptativa.

Basan su fundamento en la adaptabilidad de los procesos de desarrollo.



## METODOLOGÍAS ÁGILES

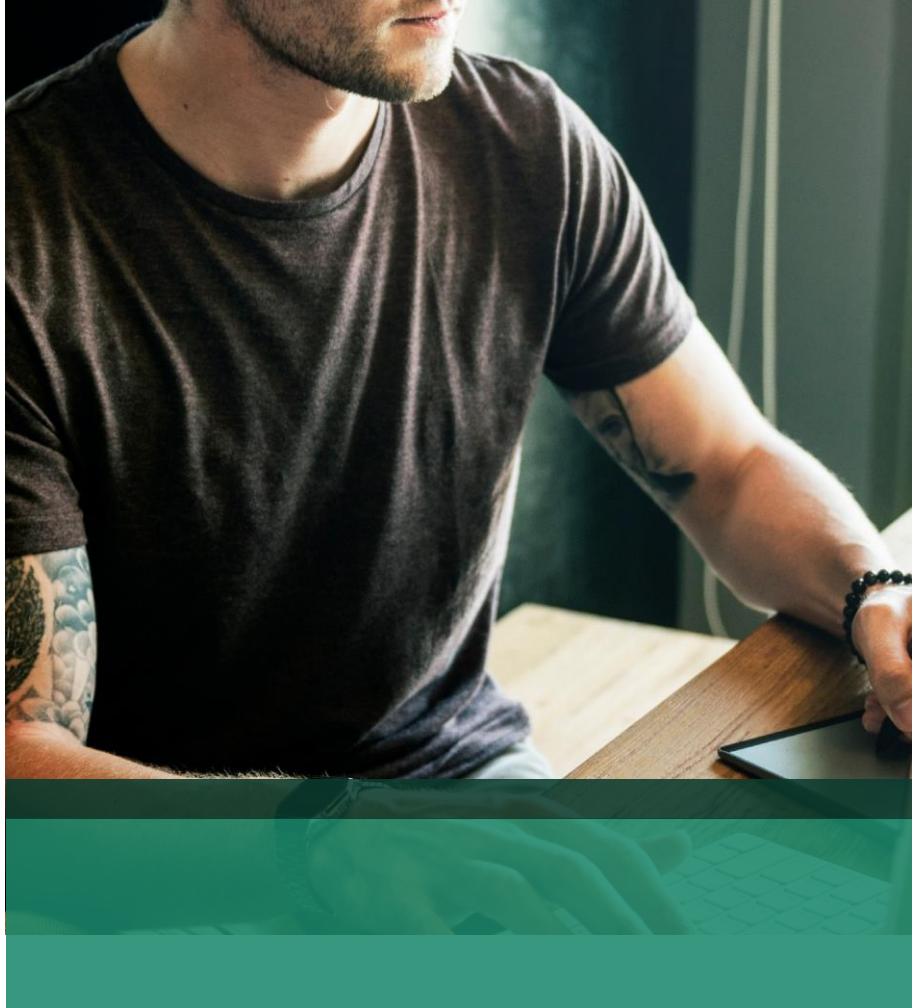


Un modelo de desarrollo ágil, generalmente, es un proceso:

- **Incremental.** Entregas frecuentes con ciclos rápidos.
- **Cooperativo.** Clientes y desarrolladores trabajan constantemente con una comunicación muy fina y constante.
- **Sencillo.** El método es fácil de aprender y modificar para el equipo.
- **Adaptativo.** Capaz de permitir cambios de último momento.

**Las características principales de las metodologías ágiles son:**

- Comunicación
- Cohesión
- Funcionalidad
- Conocimientos



### Principales ideas de la metodología ágil:

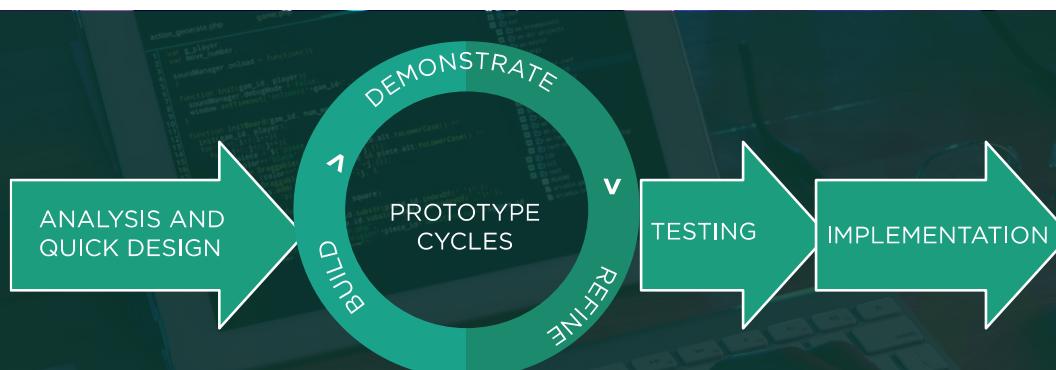
- Se encarga de valorar al individuo y las iteraciones del equipo, más que a las herramientas o los procesos utilizados.
- Se hace mucho más importante crear un producto software que funcione que escribir mucha documentación.
- El cliente está en todo momento colaborando en el proyecto.
- Es más importante la capacidad de respuesta ante un cambio realizado que el seguimiento estricto de un plan.

## Modelo RAD Rápido de Aplicaciones

Es un ciclo de desarrollo diseñado para crear aplicaciones de computadoras de alta calidad.

El método comprende el desarrollo interactivo, la construcción de prototipos y el uso de utilidades CASE.

Tradicionalmente, el desarrollo rápido de aplicaciones tiende a englobar también la usabilidad, utilidad y la rapidez de ejecución.



## METODOLOGÍAS ÁGILES



Hoy en día se suele utilizar para referirnos al desarrollo rápido de interfaces gráficas de usuario tales como Glade, o entornos de desarrollo integrado completos.

Algunas de las plataformas más conocidas son: *Visual Studio*, *Lazarus*, *Gambas*, *Delphi*, *Foxpro*, *Anjuta*, *Game Maker*, *Velneo* o *Clarion*.

### Fases del RAD:

- Modelado de gestión.
- Modelado de datos.
- Modelado de procesos.
- Generación de aplicaciones.
- Pruebas de entrega.

## Ventajas

- Comprar puede ahorrar dinero en comparación con construir.
- Los entregables pueden ser trasladados.
- El desarrollo se realiza a un nivel de abstracción mayor.
- Visibilidad temprana.
- Mayor flexibilidad.
- Menor codificación manual.
- Mayor involucramiento de los usuarios.
- Posiblemente menos fallas y menor costo.
- Ciclos de desarrollo más pequeños.
- Interfaz gráfica estándar.

### Desventajas

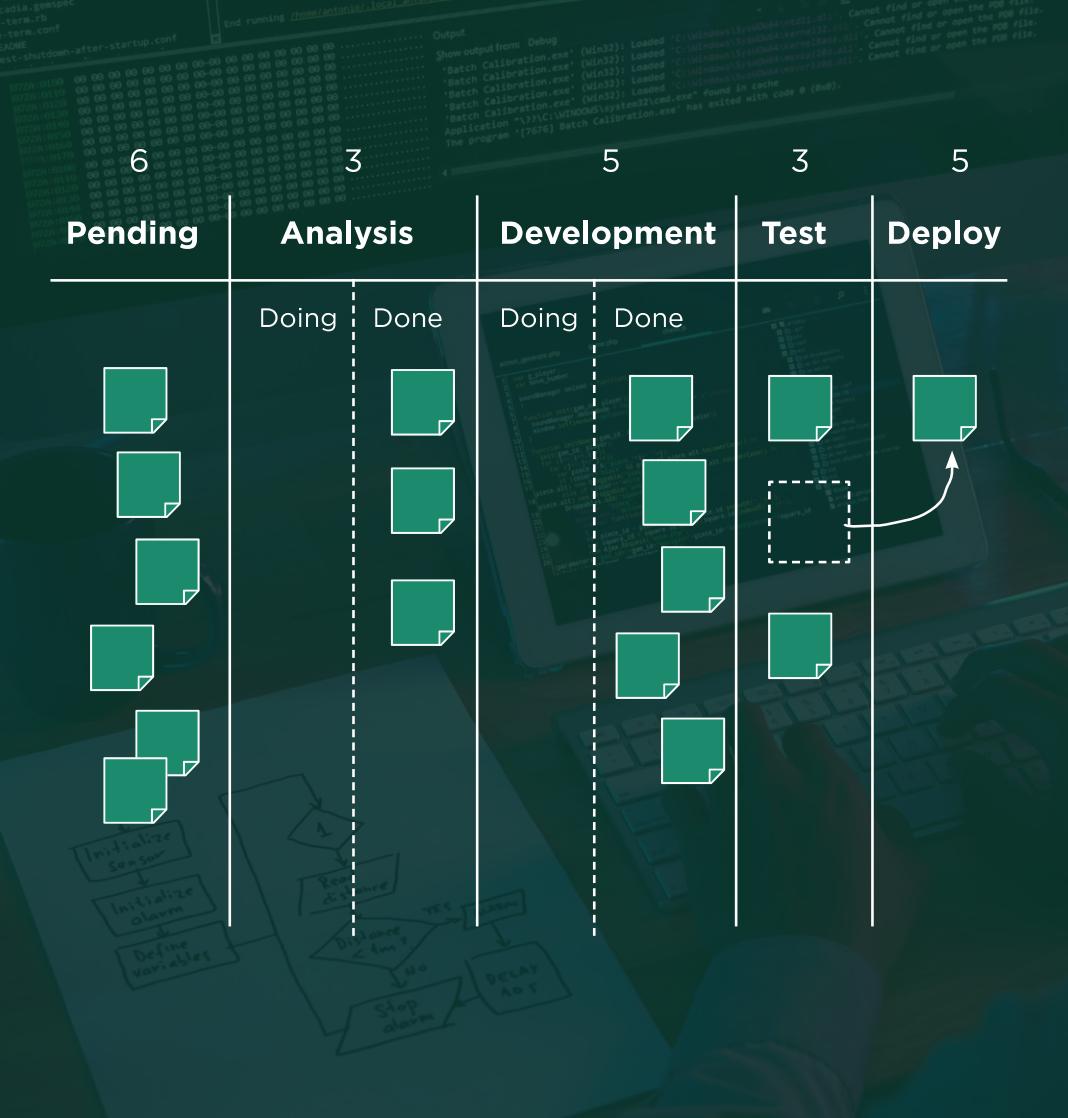
- Costo de herramientas integradas y equipo necesario.
- Progreso más difícil de medir.
- Menos eficiente.
- Menor precisión científica.
- Riesgo de revertirse a las prácticas sin control de antaño.
- Más fallas por producir rápido.
- Prototipos pueden no escalar.
- Funciones reducidas.
- Dependencia en componentes de terceros.

# METODOLOGÍAS ÁGILES

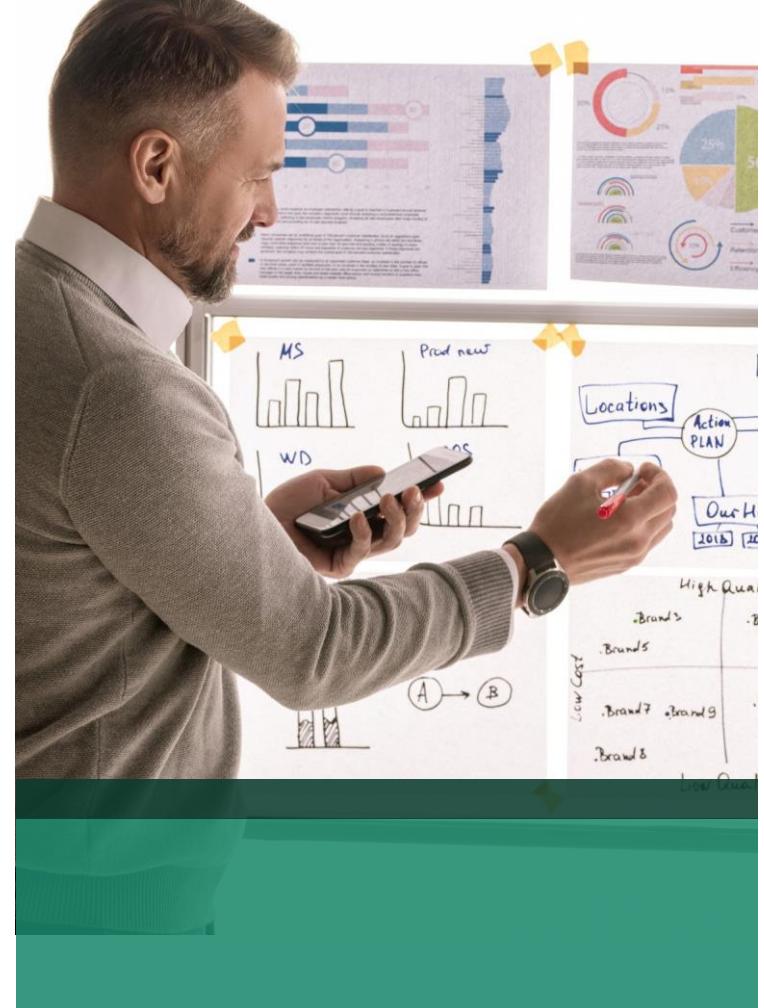
## Kanban

Esta metodología tiene como origen la aplicación de los procesos de producción JIT (*Just in Time*).

Kanban se basa en la idea de que el trabajo en curso debería limitarse. Además, solo debería empezar con algo nuevo cuando un bloque de trabajo anterior haya sido entregado o ha pasado a otra función posterior de la cadena.



A través de la implementación de tarjetas visuales, proporciona transparencia al proceso, ya que su flujo de trabajo expone los cuellos de botella, colas, variabilidad y desperdicios a lo largo del tiempo, así como todo lo que impacta al rendimiento de la organización en términos de la cantidad de trabajo entregado y el ciclo de tiempo requerido para entregarlo.





### Principios del método Kanban:

- Calidad garantizada.
- Reducción del desperdicio.
- Mejora continua.
- Flexibilidad.

### Las tres reglas de Kanban:

1. Mostrar el proceso.
2. Limitar el trabajo en curso.
3. Optimizar el flujo de trabajo.

## Scrum

Scrum es un proceso en el que se aplica, de manera regular, un conjunto de buenas prácticas para trabajar colaborativamente (en equipo) y obtener el mejor resultado posible de un proyecto.

En Scrum, se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto.



Las partes implicadas dan *feedback*

Incorporando  
el *feedback*



Recogida de  
requisitos



Gestión del  
*backlog*



Planificación  
del *sprint*



Ejecutar el  
*sprint*



Petición de  
*feedback*



Entrega a  
las partes  
implicadas

Scrum también se utiliza para resolver situaciones en las cuales no se está entregando al cliente lo que necesita. Cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable.





## METODOLOGÍAS ÁGILES

Datos importantes para la implantación de una gestión ágil de proyectos como Scrum:

- Cultura de empresa basada en trabajo en equipo, delegación, creatividad y mejora continua.
- Compromiso del cliente en la dirección de los resultados.
- Compromiso de la Dirección.
- Facilidad para realizar cambios en el proyecto.
- Dedicación del equipo a tiempo completo.

## Ventajas

- Entregas parciales a corto plazo.
- Gestión regular de las expectativas del cliente y basada en resultados tangibles.
- Resultados anticipados.
- Flexibilidad y adaptación.
- Gestión sistemática del Retorno de Inversión (ROI).
- Mitigación sistemática de los riesgos del proyecto.
- Productividad y calidad.
- Alineamiento entre el cliente y el equipo de desarrollo.
- Equipo motivado.



### Lean

La filosofía Lean consiste en tener un equipo muy preparado, altamente motivado y muy unido.

Los activos más importantes a tener en cuenta cuando se está desarrollando un proyecto bajo *Lean Development* no son el tiempo o el dinero que se está invirtiendo, sino el grado de compromiso y, sobre todo, cuánto está aprendiendo el equipo

## Principios Lean:

- Eliminar los desperdicios.
- Ampliar el aprendizaje.
- Reaccionar tan rápido como sea posible.
- Potenciar el equipo.
- Crear la integridad.
- Ver todo el conjunto.



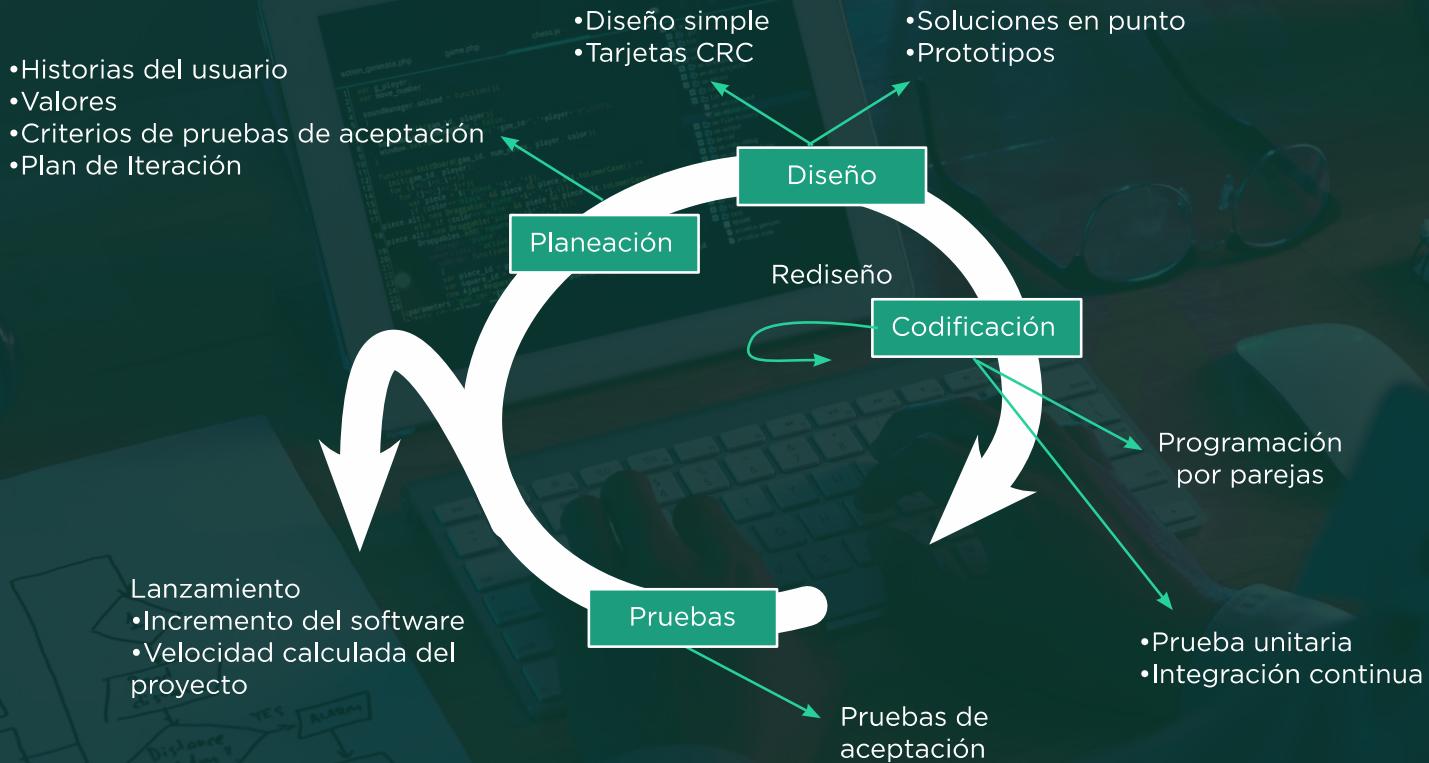
### Programación Extrema (XP)

Centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software.

XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

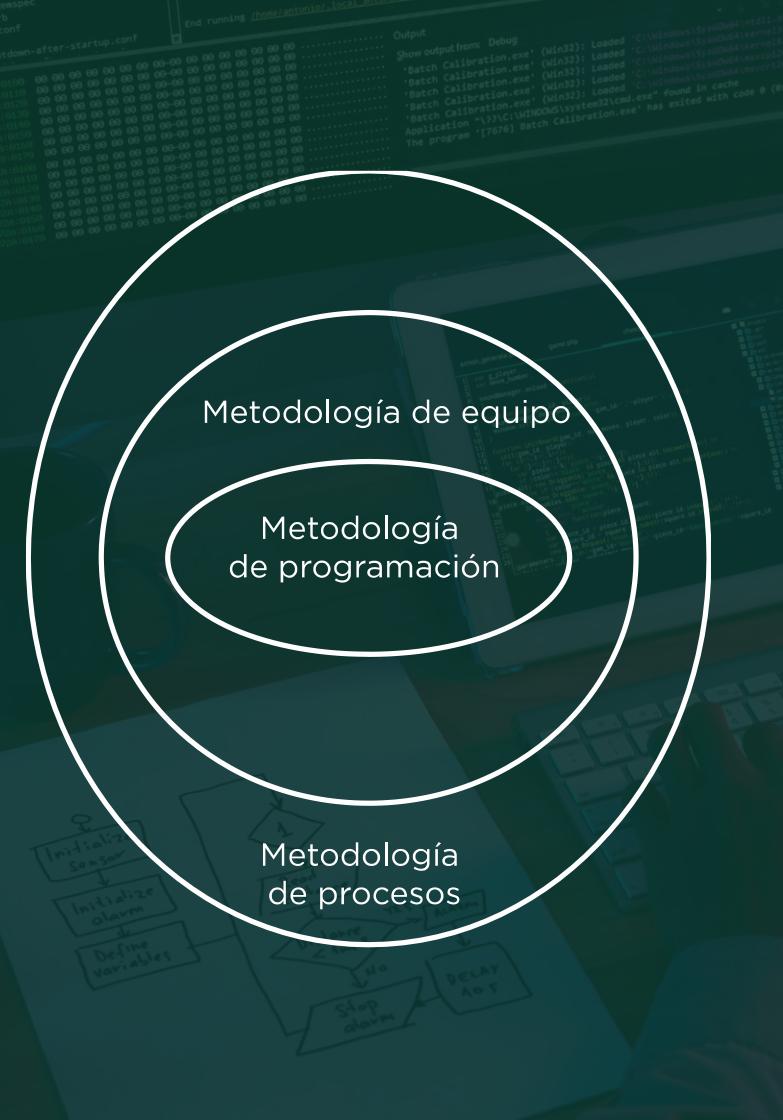


# METODOLOGÍAS ÁGILES



Estructura por capas que agrupa las doce prácticas básicas de XP:

- **Metodología de programación.** Diseño sencillo, *testing*, refactorización y codificación con estándares.
- **Metodología de equipo.** Propiedad colectiva del código, programación en parejas, integración continua, entregas semanales e integridad con el cliente.
- **Metodología de procesos.** Cliente *in situ*, entregas frecuentes y planificación.





## METODOLOGÍAS ÁGILES

El punto de partida de la metodología XP son las **variables** que utiliza para cada proyecto:

- **Coste.** La inversión económica y en recursos.
- **Tiempo.** El tiempo empleado, determinado por la fecha de entrega final.
- **Calidad.** Tanto del código como del aplicativo desarrollado.
- **Alcance.** Conjunto de funcionalidades.

Hoy en día, el marco de trabajo ágil más utilizado y conocido es Scrum.

Desde luego, existe la posibilidad de aunar las ventajas de ambas metodologías, a saber, las tradicionales y las ágiles.

De los factores ágiles se incorpora el **cómo gestionan la incertidumbre**.

Por su parte, las tradicionales aportan la **facilidad de predecir los costes y necesidades del proyecto**.



# CONCLUSIÓN



La selección y aplicación de una metodología particular para el desarrollo de software se centra en el uso de un enfoque sistemático de pasos y etapas a seguir para el cumplimiento de los objetivos en común. Por un lado, las metodologías tradicionales se basan en las buenas prácticas dentro de la ingeniería del software, siguiendo un marco de disciplina estricto y un riguroso proceso de aplicación.

Las metodologías ágiles, en cambio, representan una solución a los problemas que requieren una respuesta rápida en un ambiente flexible y con cambios constantes. Esto se debe, principalmente, a que hacen caso omiso de la documentación rigurosa y los métodos formales.

En este sentido, el objetivo es conocer las necesidades del proyecto que se busca desarrollar. Además, se deberá comprender la diferencia entre un método estructurado y un método ágil para así poder identificar cuál se adapta de manera más eficiente a este proyecto determinado.

# ¡FELICIDADES!



Acabas de concluir la cuarta unidad de tu curso *Análisis y Diseño de Sistemas*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.



# PROYECTO FINAL





# PROYECTO FINAL

Te invitamos a realizar el proyecto final:

Presiona el botón para descargar el proyecto final:



Presiona el botón para entregar el proyecto final:

