Unidad 4 - Herramientas actuales de modelado

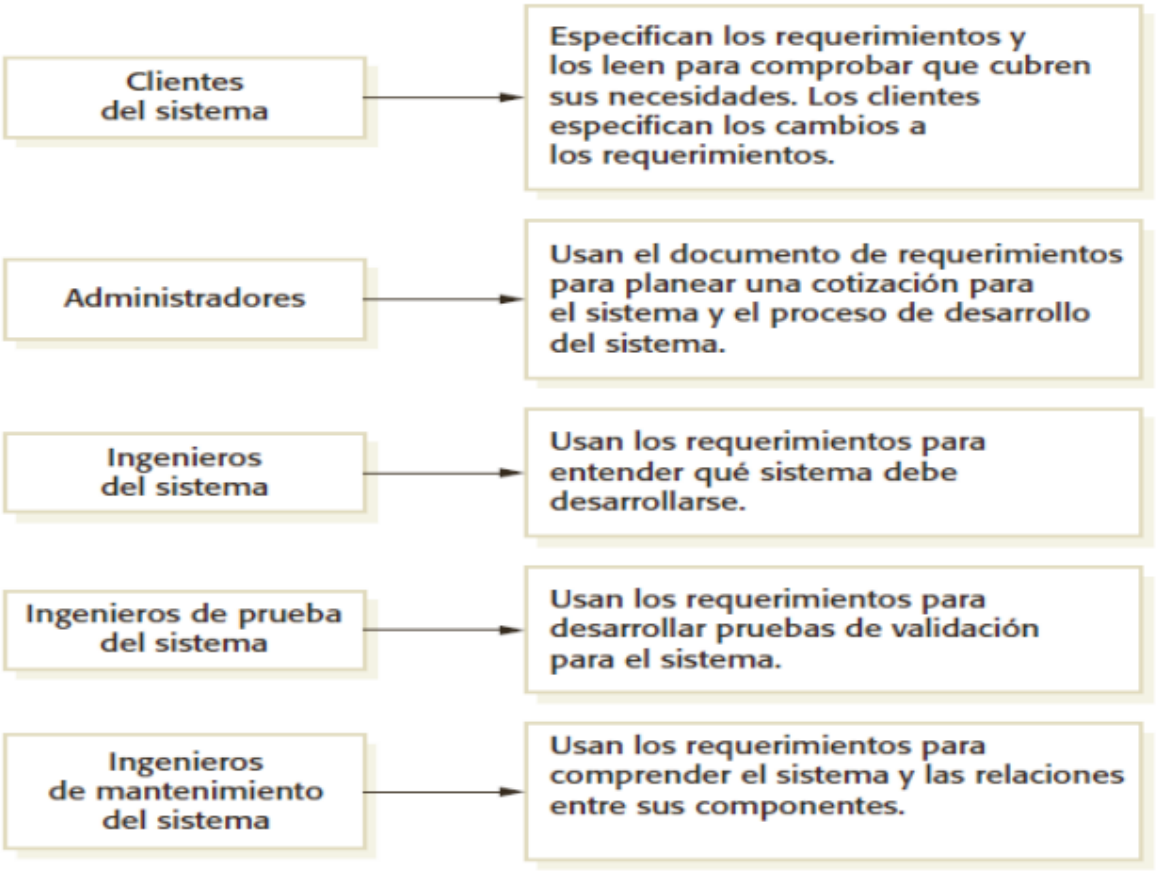
¿Qué es un requisito?

Es el flujo de trabajo fundamental cuyo propósito esencial es orientar el desarrollo hacia el sistema correcto. Esto se lleva a cabo mediante la descripción de los requisitos del sistema de forma tal que se pueda llegar a un acuerdo entre el cliente (usuario) y los desarrolladores del sistema, acerca de lo que el sistema debe hacer y lo que no.

¿Qué es un requisito de software?

Es una descripción completa del comportamiento del sistema que se va a desarrollar.

Usuarios de un documento de requisitos



Identificación de los requisitos

1. Definidos sin ambigüedad.
2. Son completos.
3. Tienen consistencia.
4. Especifica el origen.
5. Evita detalles de diseño.
6. Están enumerados.

Elicitación

Es el proceso de adquirir todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio de problema. Su **objetivo** es entender el dominio del problema particular.

Técnica de elicitación

- Entrevista de comienzo y final abierto.
- Entrevistas estructuradas.
- Brainstorming (lluvia de ideas).
- Prototipos.
- Escenarios.
- Observación.

Tipos de requisitos

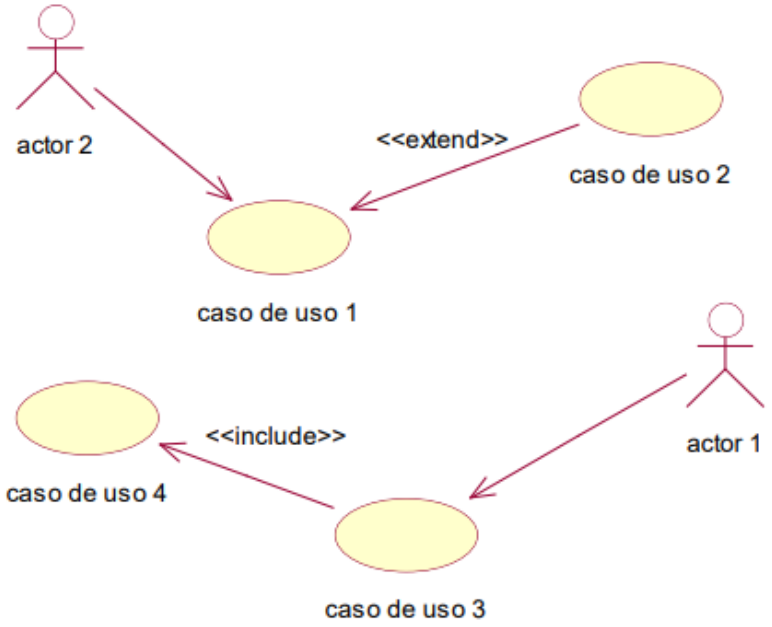
- **Requisitos de usuarios:** necesidades que los usuarios expresan verbalmente**.**
- **Requisitos del sistema:** son los componentes que el sistema debe tener para realizar determinadas tareas.
- **Requisitos funcionales:** servicios que el sistema debe proporcionar.
    - Especifica una acción que debe ser capaz de realizar el sistema, sin considerar restricciones físicas.
    - Describen la funcionalidad o los servicios que se espera proveerá el sistema.
- **Requisitos no funcionales:** restricciones que afectan al sistema.
    - Especifica restricciones físicas sobre un requisito funcional (rendimiento, plataforma, fiabilidad).
    - A nivel hardware y redes.

Requisitos de casos de uso

Los ***casos de uso*** son una técnica para la especificación de requisitos funcionales propuesta inicialmente por *Jacobson* (1992). Estos presentan una ventaja sobre la descripción textual de los requisitos funcionales**.**

Los casos de uso son una técnica que se utiliza para documentar los requerimientos funcionales de un sistema desde el punto de vista de los usuarios.
Responden a la pregunta *¿Qué se supone que el sistema deba hacer para los usuarios?*

Diagrama de casos de uso



El valor de los casos de uso está en su descripción detallada (no en el dibujo…)

Describe tanto lo que hace el actor como lo que hace el sistema cuando interactúa con él. Están acotados al uso de una determinada funcionalidad, claramente diferenciada, del sistema.

Actores

Un actor es alguien o algo que interactúa con el sistema (una persona, una organización, un programa o sistema de hardware o software) y es externo al sistema. Éste estimula al sistema con algún evento o recibe información del sistema y posee un rol definido (cliente, banco, empleado, etc).

Los actores se dividen en:

1. *Actores primarios*: utilizan las funciones principales del sistema.
2. *Actores secundarios*: efectúan tareas administrativas o de mantenimiento.

Pre y post condiciones

Pre condiciones: establece lo que siempre debe cumplirse antes de comenzar un caso de uso. No se prueban en el caso de uso, se asumen que son verdaderas.

Post condiciones: establece qué debe cumplirse cuando el caso se completa con éxito.

Introducción al paradigma OO

¿Qué es un paradigma?

Un paradigma es una forma de entender y representar la realidad: un conjunto de teorías, estándares y métodos que, juntos, representan un modo de organizar el pensamiento.

El paradigma OO

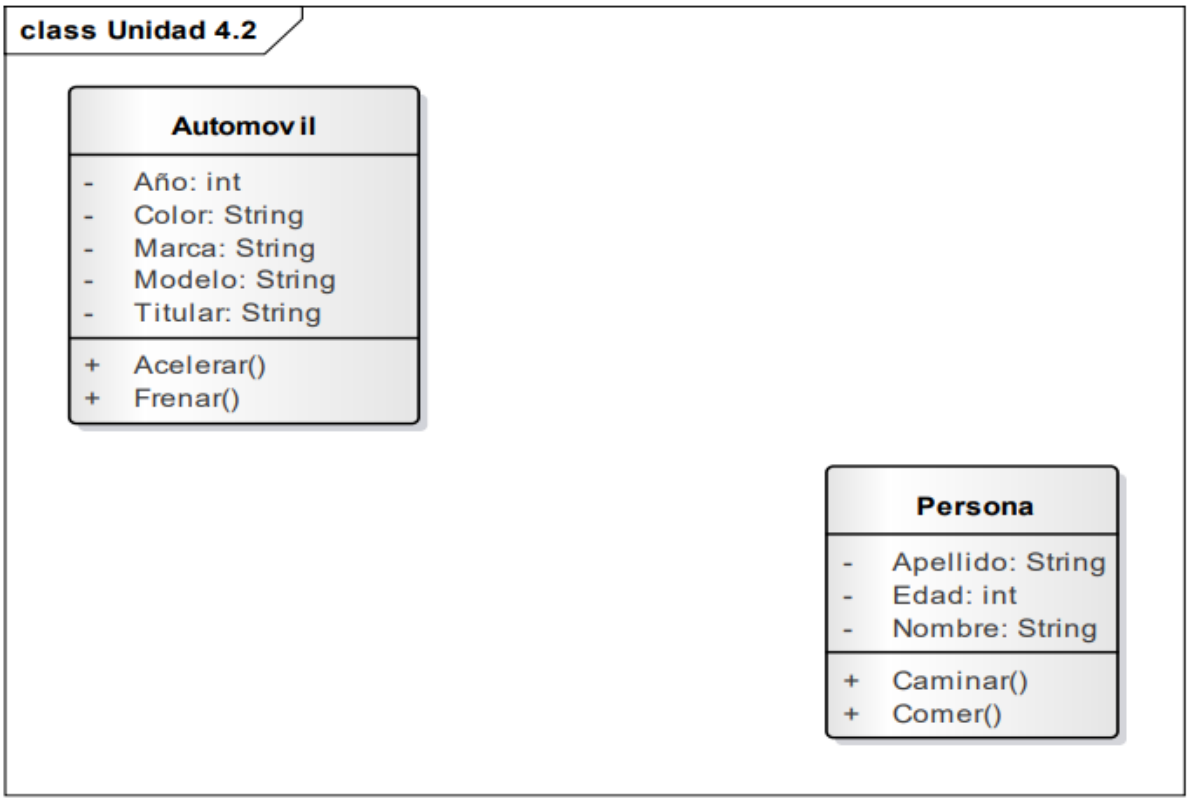
Define los programas en términos de comunidades de objetos. Los objetos con características comunes se agrupan en clases (un conjunto similar al de tipo abstracto de dato).

Clases

Una **clase** específica una estructura de datos y las operaciones permisibles que se aplican a cada uno de sus objetos. Dicho objetos se vinculan mediante **mensajes** a **operaciones** que activan los **métodos**.

- **Mensaje**: es una solicitud para que se lleve a cabo la operación indicada y se produzca el resultado.
- **Operación**: es una función o transformación que se aplica a un objeto de una clase.
- **Método**: es la implementación de una operación.

*Un objeto es una instancia de una **clase**.*



Objetos

Los objetos son entidades que combinan un estado (es decir, datos) y un comportamiento (esto es, procedimientos o métodos). Estos objetos se comunican entre ellos para realizar tareas.

Propiedades de los objetos

- **Estado**: abarca todas las propiedades (normalmente estáticas) del mismo, más los valores actuales (normalmente dinámicos) de cada una de esas propiedades.

- **Comportamiento**: muestra cómo actúa y reacciona un objeto, en términos de sus cambios de estado y paso de mensajes.
- **Identidad**: es aquella propiedad de un objeto que lo distingue de todos los demás objetos.

Abstracción

Es la propiedad que permite representar las características esenciales de un objeto, sin preocuparse de las restantes características (no esenciales). Plantea **¿Qué hace?** en lugar de **¿Cómo lo hace?**

Desde el punto de vista **orientado a objetos**, la abstracción**:**

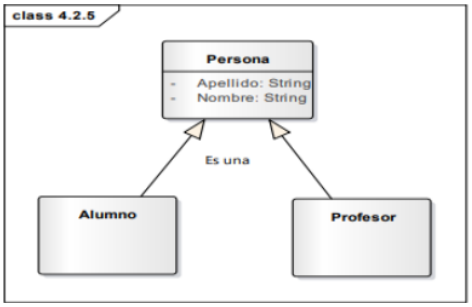
- Expresa las características esenciales de un objeto, la cual distingue al objeto de los demás.
- Provee límites conceptuales.

Encapsulamiento

Es la propiedad que permite asegurar que el contenido de la información de un objeto está oculta al mundo exterior.

Relaciones de jerarquía

- Generalización: consiste en factorizar los elementos comunes de un conjunto de clases en una clase más general llamada superclase.
- Herencia: es una técnica de los lenguajes de programación para construir una clase a partir de una o varias clases, compartiendo atributos y operaciones.

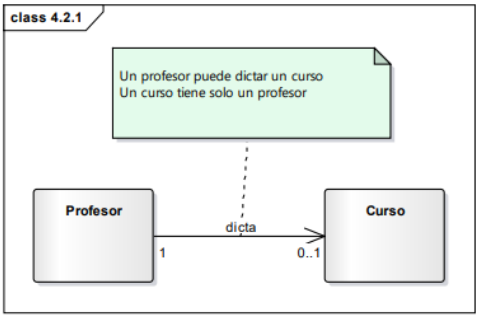


Algunas Personas son Alumnos y otras Profesores
Un Profesor es una Persona

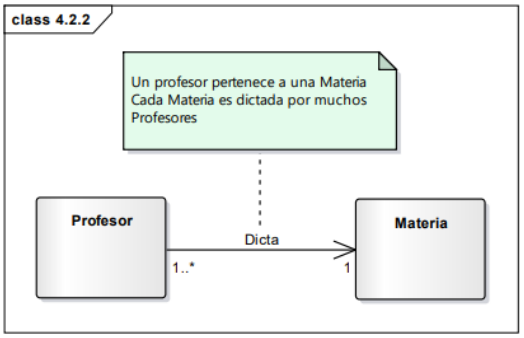
Multiplicidad de relaciones

- Determina cuantos objetos de cada tipo intervienen en una relación.
- Cuántas instancias de una **clase** que se relacionan con **una** instancia de otra clase.

Multiplicidad	Significado
1	Uno y sólo uno
0..1	Cero o uno
N..M	Desde N hasta M
*	Cero o varios
0..*	Cero o varios
1..*	Uno o varios (al menos uno)



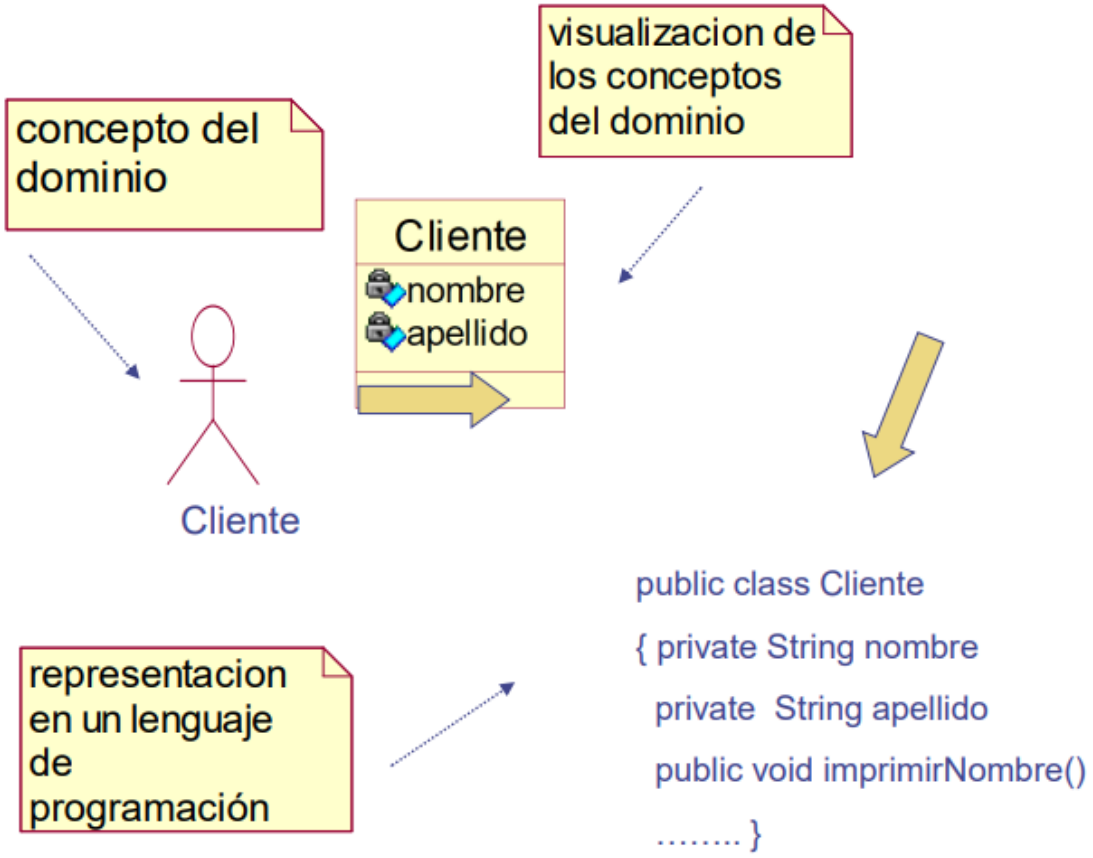
Multiplicidad	Significado
1	Uno y sólo uno
0..1	Cero o uno
N..M	Desde N hasta M
*	Cero o varios
0..*	Cero o varios
1..*	Uno o varios (al menos uno)



Análisis OO vs diseño OO

Durante el **análisis OO** se presta especial atención a encontrar y describir los objetos (conceptos) del dominio del problema.

Durante el **diseño OO** se presta atención a la definición de los objetos software y en cómo colaboran para satisfacer requisitos.



Análisis OO

La finalidad del análisis OO es crear una descripción del dominio desde una perspectiva de clasificación de objetos: identificación de conceptos, atributos e interrelaciones significativas.

Vista estática vs vista dinámica

Vista estática: se ocupa de definir la estructura del sistema en cuestión. Está compuesta por las clases y las relaciones que dan soporte para cumplir lo planteado en los RF.

- Modelo de dominio.

Vista dinámica: define las responsabilidades que tendrá el sistema.

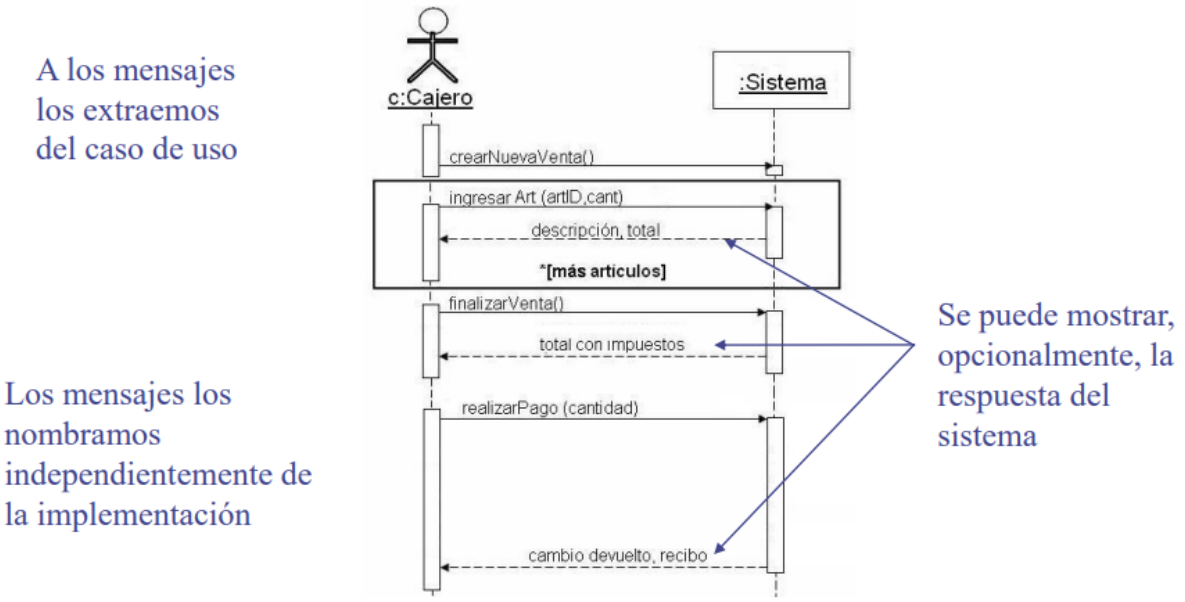
- Diagrama de secuencia del sistema.

Modelo del dominio

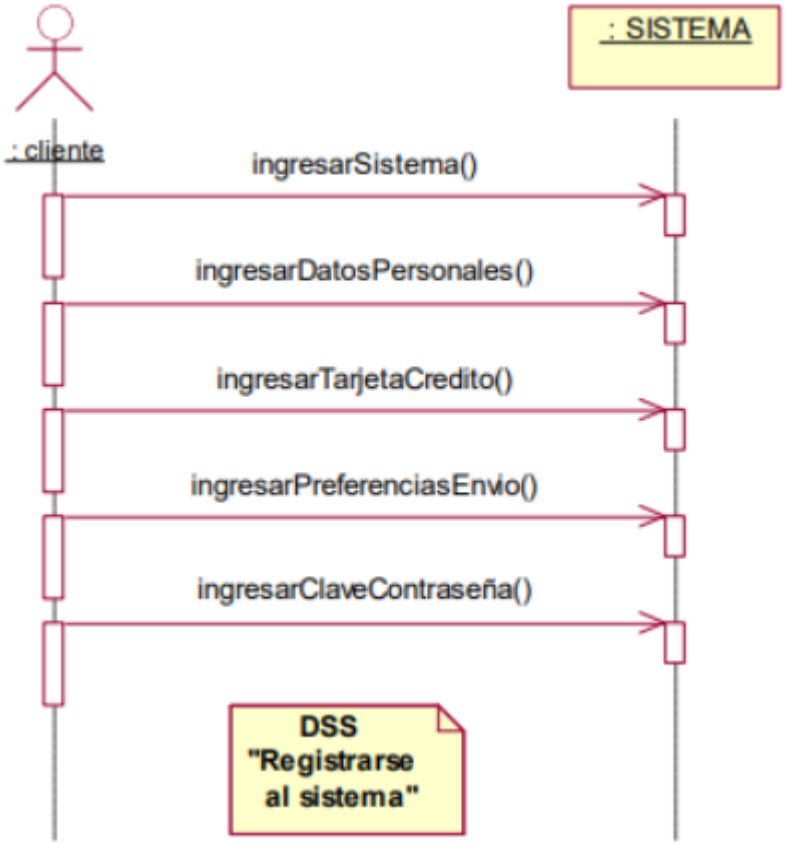
El modelo del dominio es una representación visual de las clases conceptuales del mundo real.

Diagrama de secuencia del sistema

Los diagramas de secuencia del sistema se utilizan en la etapa de análisis para documentar casos de uso.



Ejemplo:



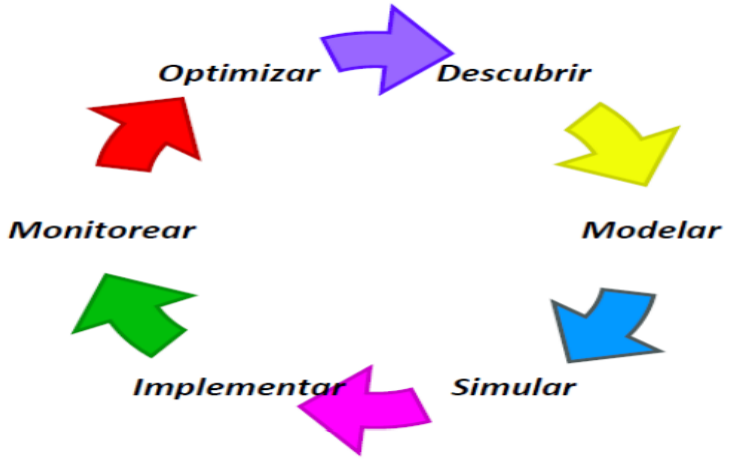
Unidad 5 - Desarrollo de Software

Proceso de Negocio

¿Qué es un proceso de negocio?

“Los procesos de negocios se refieren a la manera de organizar, coordinar y enfocar el trabajo para elaborar un producto o servicio”

Ciclo de vida de un proceso de negocio



Flujo: simplificarlo, de modo que no haya redundancia de trabajo, demoras, pasadas por un mismo lugar, etc.

Costo: minimizar el número de personas que participan y racionalizar el uso de los recursos que intervienen.

Tiempo: reducirlo, que tienda a tiempo real o bien lo antes que sea posible (la salida debe estar disponible en el momento que se requiera).

Calidad: mejorarla de modo de conseguir el 100% de lo indicado en las normas correspondientes.

Espacio: desde el punto de vista de la información, reducir el espacio que media entre una estación de trabajo y otra, de modo que en vez que viaje el medio que lo contiene, se haga viajar sólo los datos que son propios de esa entrada o salida.

Servicio: satisfacer a plenitud al cliente que recibe la salida de una acción, de modo de lograr fidelización.

Diagramas de flujo

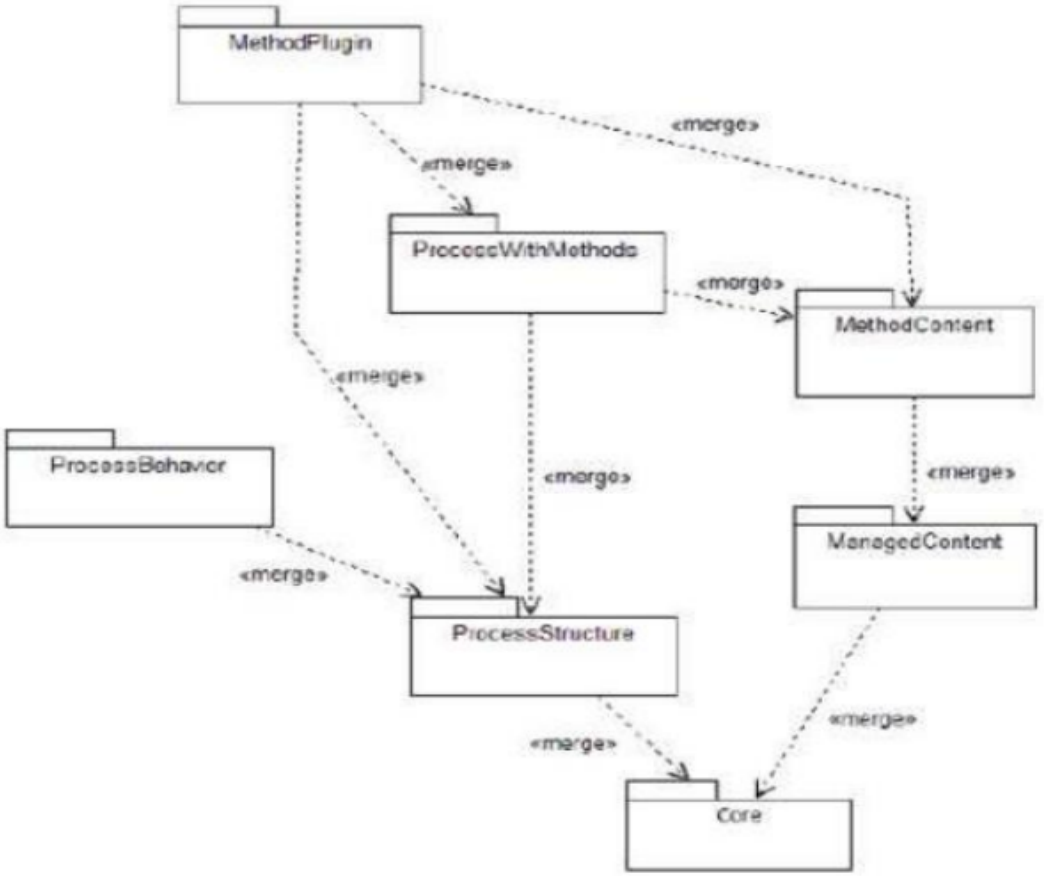
Representación abstracta (gráfica) de los procesos de una organización, que muestran principalmente **cómo** y por **quién** son llevadas a cabo las actividades que generan valor para la organización.

*Ventajas y desventajas***:**

- Mejor entendimiento del sistema y la empresa.
- Proporcionar mejores soluciones a la empresa.
- Más rápido y entendible.
- Mejores resultados.

SPEM (software process engineering metamodel)

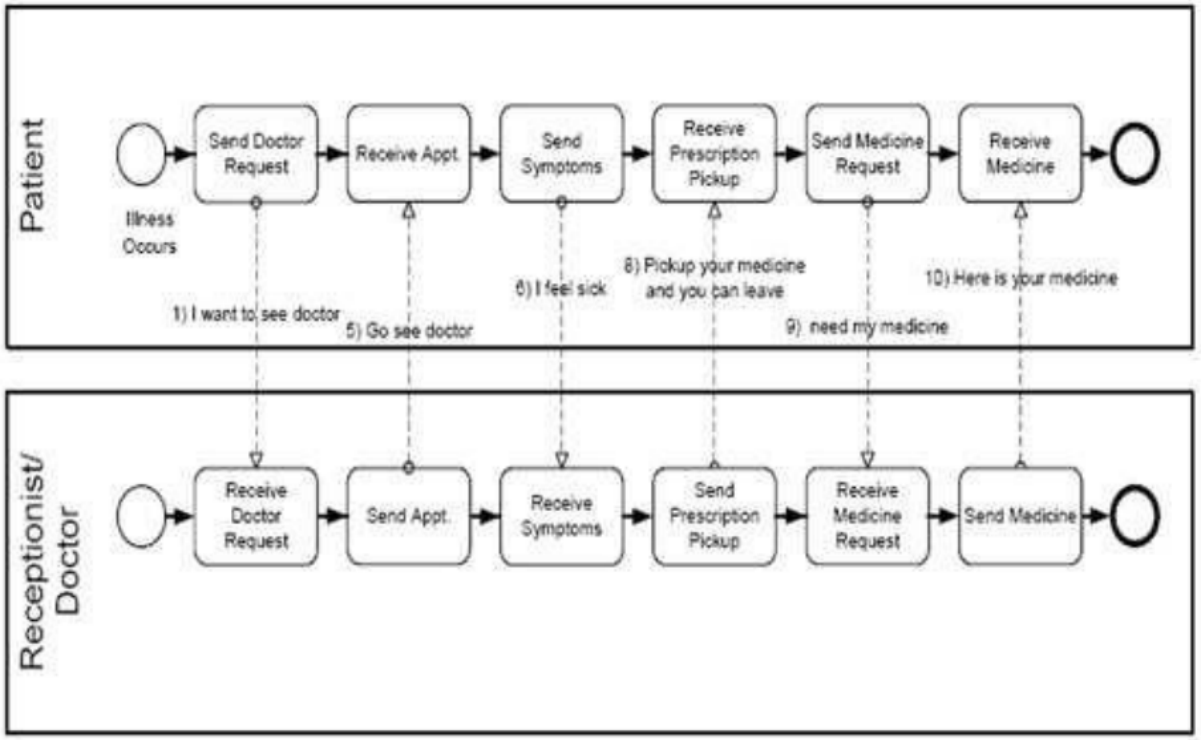
Es un estándar de la OMG cuyo objetivo principal es proporcionar un marco formal para la definición de procesos de desarrollo de sistemas y de software así como para la definición y descripción de todos los elementos que los componen.



BPMN (business process modelling notation)

Es un estándar de la **BPMI** (Business Process Management Initiative), cuyo principal **objetivo** es: “proporcionar una notación fácilmente comprensible por todos los usuarios del negocio, desde los analistas, los desarrolladores técnicos hasta aquellos que monitorizarán y gestionarán los procesos“.

- Crear puentes entre el diseño de los procesos de negocio y la implementación del proceso.



BMM (business modeling method)

Método de Modelado de Negocios orientado al desarrollo de sistemas de información empresarial. Integra los aspectos o elementos más importantes de un negocio y delimita el proceso de modelado.

Divide el sistema de negocios en 3 niveles:

1. Objetivos.
2. Procesos.
3. Sistemas.

Diagramas de actividad

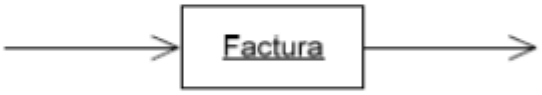
Es un diagrama **UML** que se utiliza para la representación del comportamiento dinámico en un sistema. Se centra en la secuencia de actividades que se llevan a cabo.

Elementos:

Actividad:

Flujo:

Flujo de Dato:

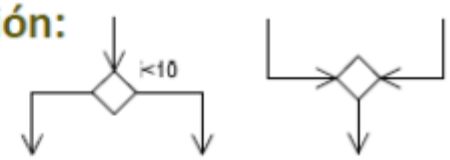


Nodo de Inicio:

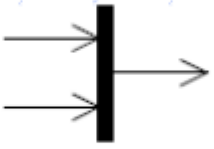
Nodo de Fin:

Partición:

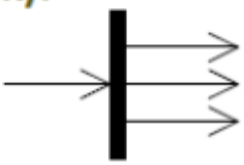
Nodo Decisión y Combinación:



Nodo Unión (Join):



Nodo Bifurcación (Fork):



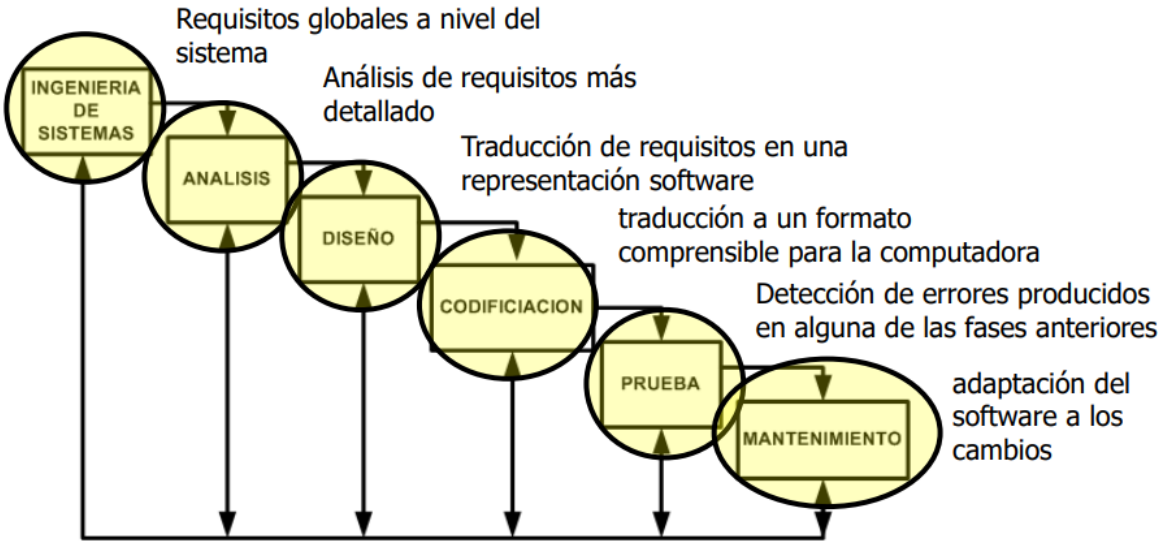
- **Nodo Decisión**: Guían el flujo en una u otra dirección.. Poseen un único flujo de entrada y varios flujos de salida.
- **Nodo Combinación**: Tiene la misma representación que el nodo anterior pero a diferencia de este tienen múltiples flujos entrada pero un único flujo de salida.
- **Nodo Fork**: Divide un único flujo de entrada en varios flujos de salida que se ejecutarán de manera concurrente.
- **Nodo Join**: Para sincronizar múltiples flujos. Varios flujos de entrada y un único flujo de salida.

Proceso de Desarrollo de Software

Ciclo de vida

El ciclo de vida de desarrollo de sistemas es una sucesión de etapas por las que atraviesa el software desde que comienza un nuevo proyecto hasta que éste se deja de utilizar.

Ciclo de vida en cascada



Herramientas 4G

Es un conjunto muy diverso de métodos y herramientas que tiene por objetivo facilitar el desarrollo de software. Como por ejemplo:

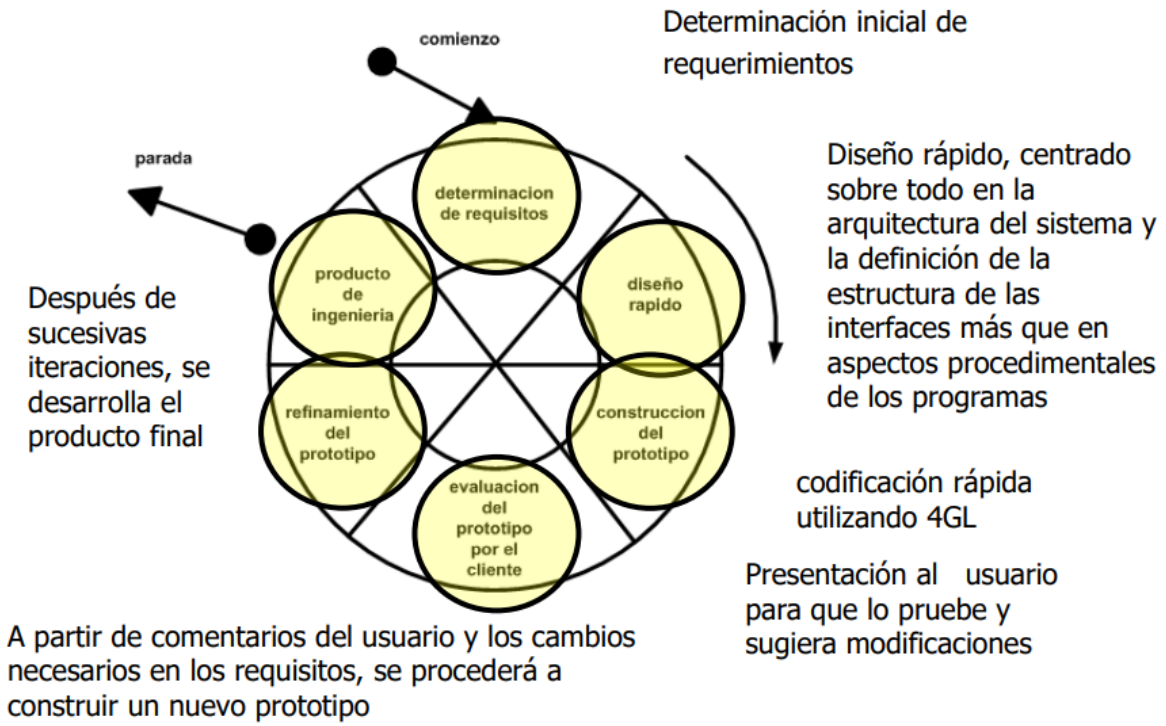
- Generación de código.
- Generación de pantallas y de informes.
- Gestión de entornos gráficos.
- Herramientas de acceso a bases de datos.

La **ventaja principal** de estas herramientas es la generación automática de código a partir de especificaciones de alto nivel de abstracción.

Prototipos

La construcción de un prototipo comienza con la realización de un modelo del sistema, a partir de los requisitos que se conocen. No es necesario realizar una definición inicial completa de los requisitos del usuario.

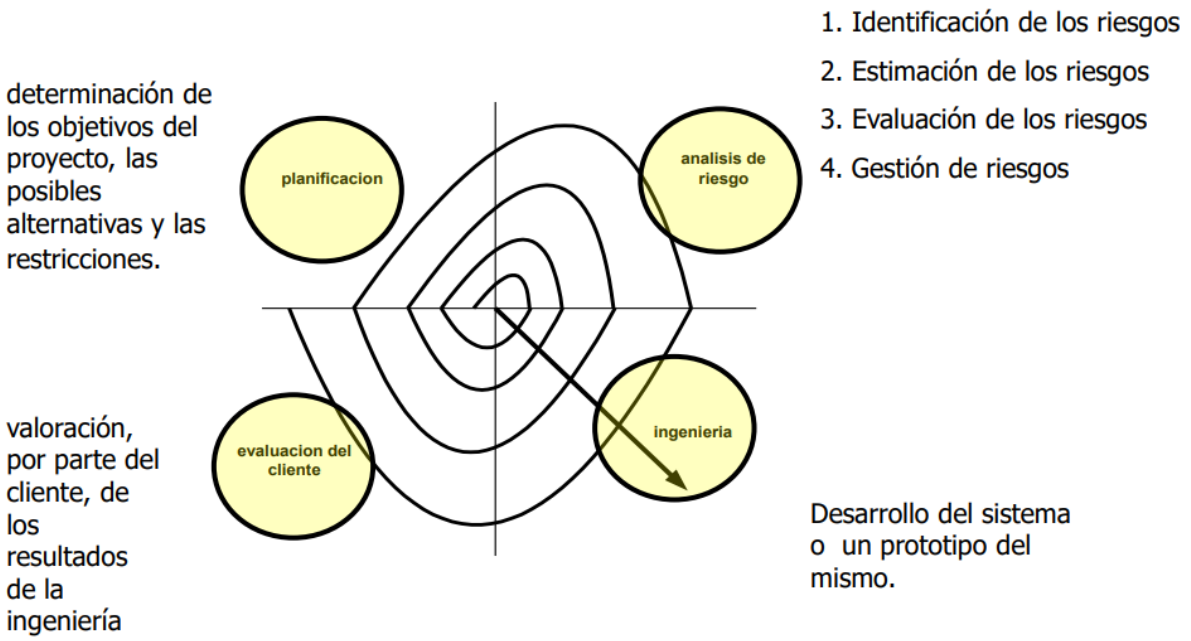
Es un proceso iterativo e incremental.

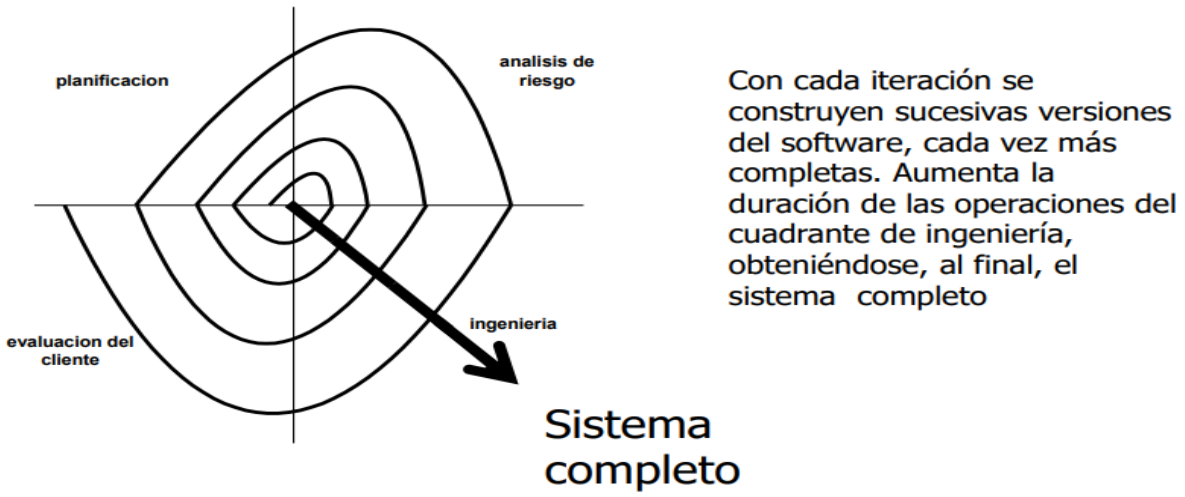


Uno de los principales **problemas** con este método es que, con demasiada frecuencia, el prototipo pasa a ser parte del sistema final.

Ciclo de vida en espiral

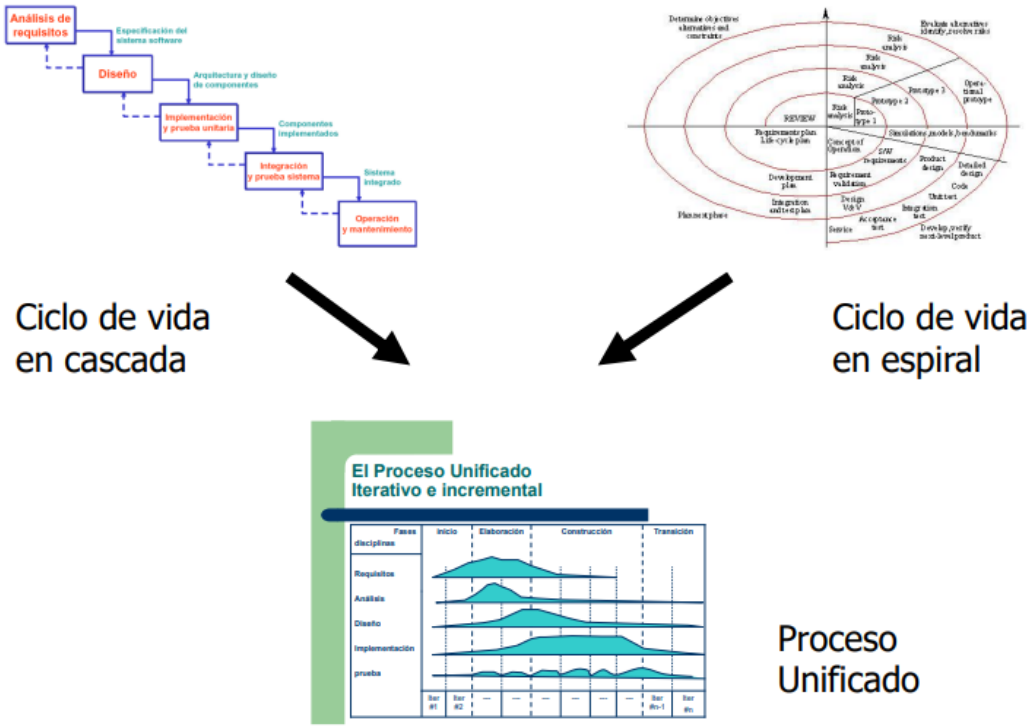
El modelo en espiral proporciona un modelo evolutivo para el desarrollo de sistemas de software complejos.





Proceso unificado de desarrollo

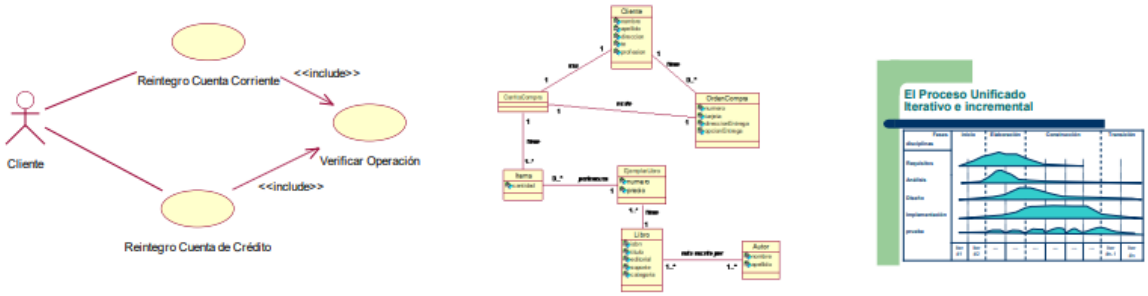
UML no es un proceso, es una notación. Por lo tanto precisa de un proceso de desarrollo (ciclo de vida) que especifique la secuencia de actividades que deben realizarse.



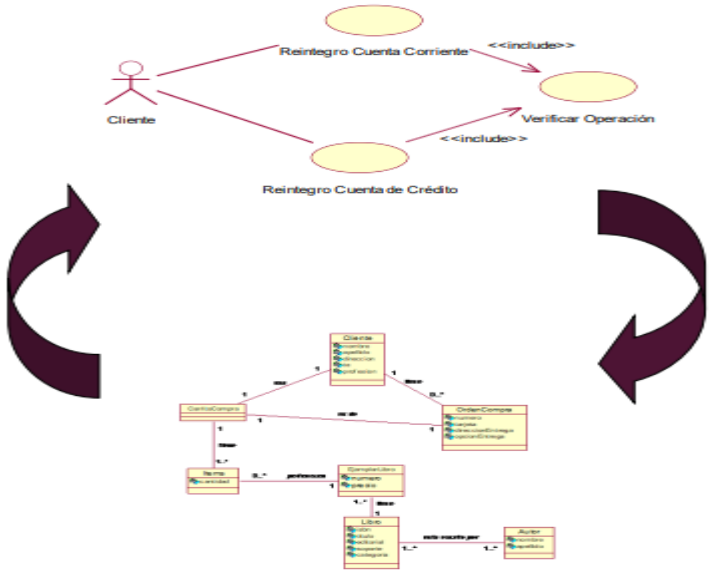
Proceso unificado

El proceso de desarrollo de software es el conjunto de actividades para transformar los requisitos del usuario en un sistema de software.

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

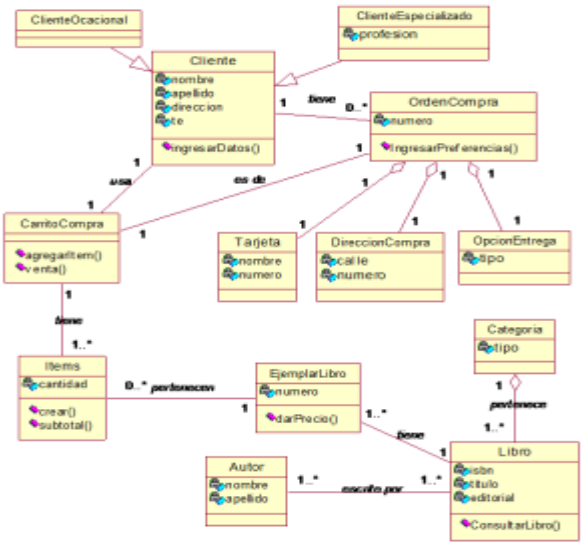


Los **casos de uso** representan los requisitos funcionales y guían el diseño, la implementación y la prueba. Basándose en los casos de uso los desarrolladores crean modelos de diseño e implementación que llevan a cabo los casos de uso.



La **arquitectura** es una vista de diseño con las características más importantes, dejando los detalles de lado. Describe diferentes vistas del sistema. Constituyen la “forma del sistema”, incluye aspectos estáticos y dinámicos.

La arquitectura y los casos de uso evolucionan en paralelo.



Desarrollo iterativo

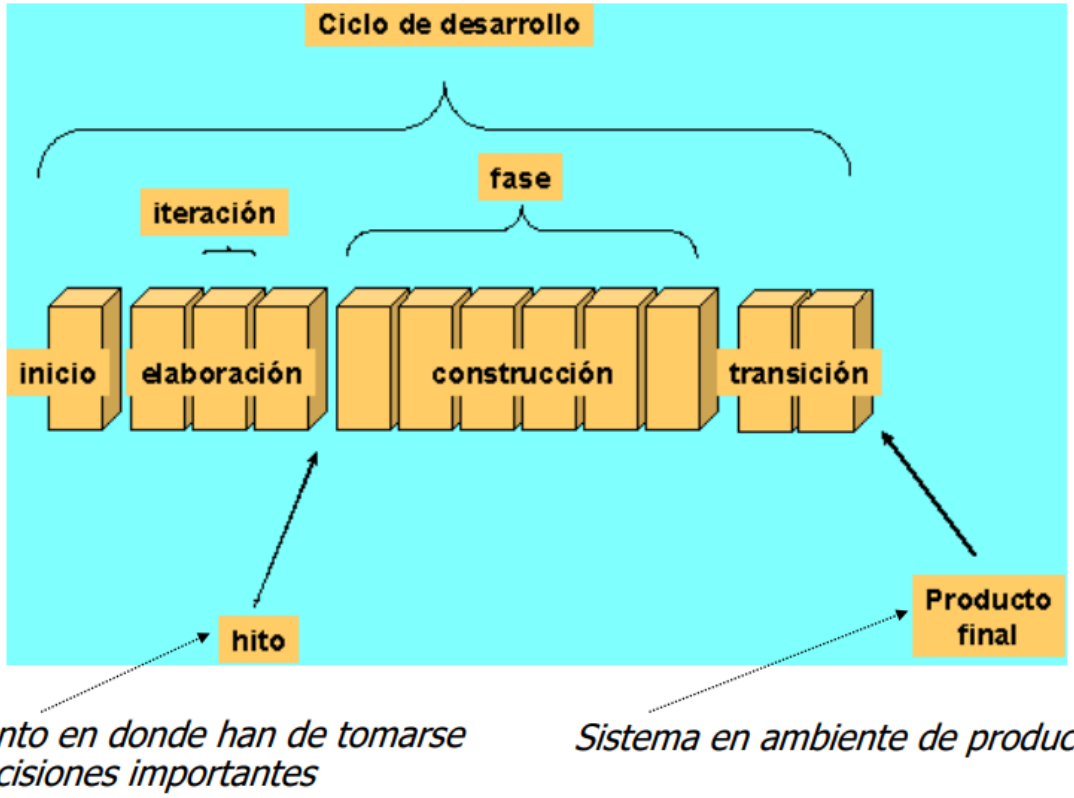
El desarrollo se organiza en una serie de mini proyectos de **duración fija**, llamados **iteraciones** (2 a 6 semanas).

El resultado de cada uno es un sistema que puede ser **probado, integrado y ejecutado**.

Cada iteración incluye sus propias actividades de: análisis de requisitos, diseño, implementación y prueba.

El sistema crece de forma **incremental a lo largo** del tiempo, iteración tras iteración.

Ciclo de desarrollo



Desarrollo guiado por test

Este es el ciclo de vida del desarrollo guiado por test, Rojo-Verde-Refactor y se debería repetir esto una y otra vez hasta que tengamos las características funcionales y completas.

Características:

- La obtención de un buen resultado depende de la calidad de las pruebas.
- No solo se basa en pruebas, depende también del refactor del código (mejor código, limpio, mantenible).
- El refactor solo se aplica si es necesario.
- El test es escrito por el desarrollador.
- El foco principal de TDD es testar las funcionalidad de bajo nivel.

Ventajas:

- Se condicionan las pruebas a lo implementado: pueden obviarse casos de prueba.
- Se realiza un profundo análisis funcional y se refinan los requisitos.

Planificación y programación de proyectos

Un **proyecto** se puede considerar como la sucesión de un conjunto de tareas interrelacionadas que deben ejecutarse en un orden determinado y con el fin de alcanzar un objetivo.

Cuando se emprende la realización de un proyecto se reconoce en su desarrollo tres etapas bien diferenciadas:

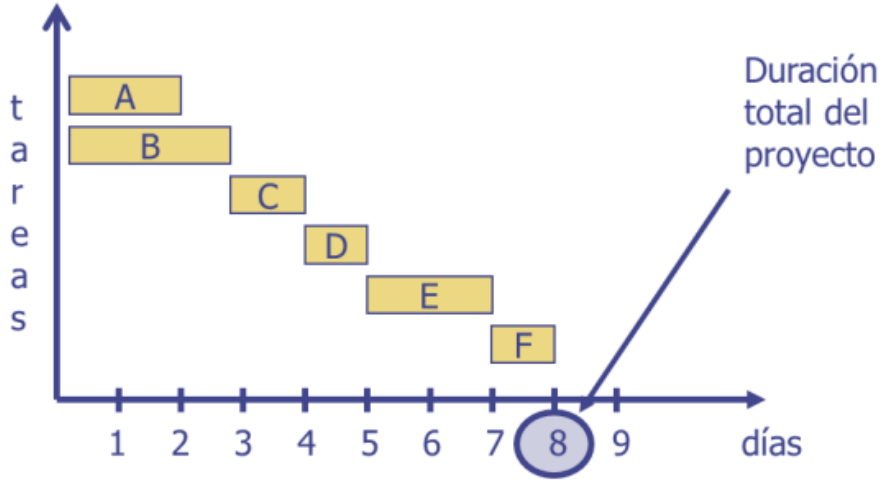
- El **planeamiento** establece qué debe hacerse y en qué secuencia.
- La **programación** determina cuándo debe hacerse, esto es, acota en el tiempo lo planeado.
- El **control** se encarga de verificar si se cumple con lo planeado y lo programado anteriormente.

Diagrama de Gantt

Es una de las técnicas más simples utilizadas en la administración de proyectos. Consiste en representar las tareas por medio de barras, cuyas longitudes son proporcionales a la duración de las tareas.

- Se destaca la sencillez y facilidad de comprensión al integrar gráficamente la planeación, la programación y el progreso del proyecto.
- Permite visualizar rápidamente los elementos principales, su programación en el tiempo y, además, el progreso en cada uno de ellos.

Actividad	Duración	Precedencia
A	2	-----
B	3	----
C	1	B
D	1	A, C
E	2	D
F	1	E



Metodologías ágiles

Las metodologías ágiles de desarrollo están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes.

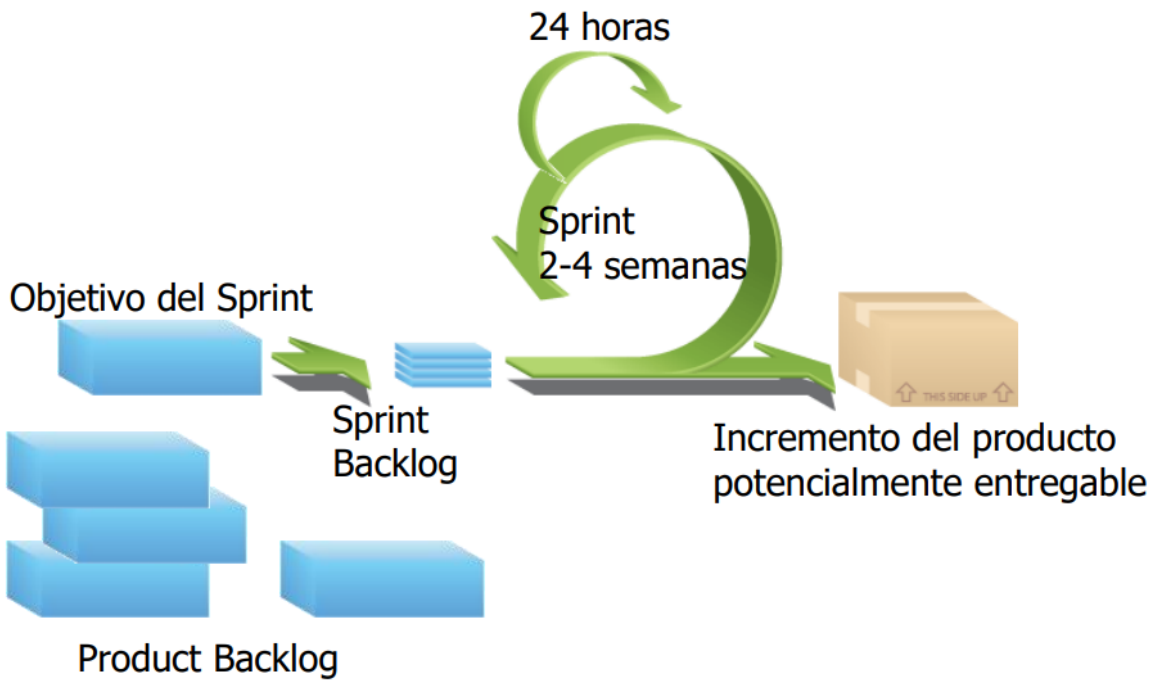
Ventajas

- Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo.
- Entrega continua y en plazos breves de software funcional.
- Trabajo conjunto entre el cliente y el equipo de desarrollo.

Scrum

Scrum es un proceso ágil que nos permite centrarnos en ofrecer el más alto valor de negocio en el menor tiempo. Nos permite rápidamente y en repetidas ocasiones inspeccionar software real de trabajo (cada dos semanas o un mes).

Los equipos se auto organizan a fin de determinar la mejor manera de entregar las funcionalidades de más alta prioridad. Cada dos semanas o un mes, cualquiera puede ver el software real funcionando y decidir si liberarlo o seguir mejorándolo en otro sprint.



Sprints

En Scrum los proyectos avanzan en una serie de “Sprints”.

- La duración típica es 2–4 semanas o a lo sumo un mes calendario.
- La duración constante conduce a un mejor ritmo.
- El producto es diseñado, codificado y testeado durante el Sprint.

Roles

1) Product owner:

- Define las funcionalidades del producto.
- Decide sobre las fechas y contenidos de los releases.
- Es responsable por la rentabilidad del producto (ROI).
- Prioriza funcionalidades de acuerdo al valor del mercado/negocio.
- Ajusta funcionalidades y prioridades en cada iteración si es necesario.
- Acepta o rechaza los resultados del trabajo del equipo.

2) Scrum master:

- Representa a la gestión del proyecto.
- Responsable de promover los valores y prácticas de Scrum.
- Remueve impedimentos.
- Se asegura de que el equipo es completamente funcional y productivo.
- Permite la estrecha cooperación en todos los roles y funciones.
- Escudo del equipo de interferencias externas.

3) Team:

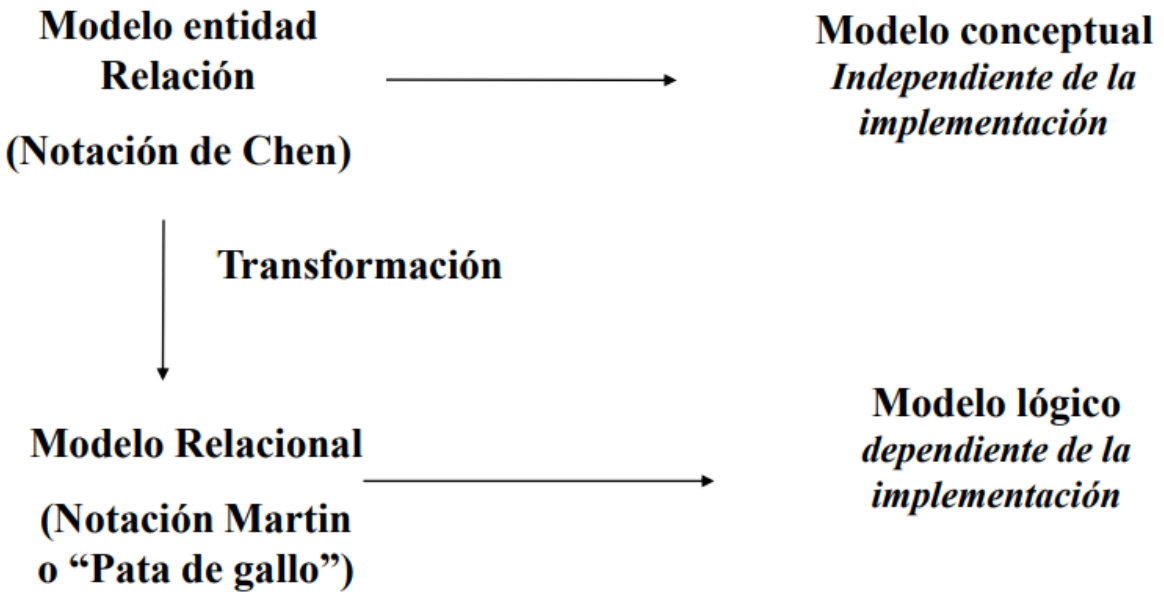
- Típicamente de 5 a 9 personas
- Multi-funcional: Programadores, testers, analistas, diseñadores, etc.
- Los miembros deben ser full-time: puede haber excepciones (Ej.: Infraestructura, SCM, etc.)
- Los equipos son auto-organizativos: Idealmente, no existen títulos pero a veces se utilizan de acuerdo a la organización
- Solo puede haber cambio de miembros entre los sprints.

Reuniones

- **Daily scrum:**
  - Diario.
  - Dura 15 minutos.
  - ***No*** es dar un status report al Scrum Master.

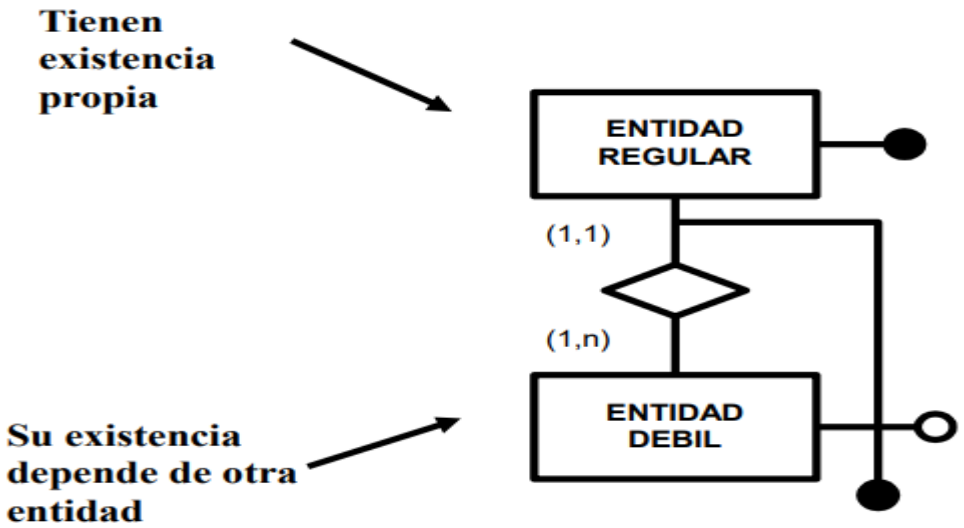
Unidad 6 - Modelado y Persistencia de Datos

Modelo Entidad-Relación



Entidades fuertes y débiles

Una ***entidad fuerte*** es la que tiene existencia propia. En cambio, una ***entidad débil*** es la que su existencia depende de otra entidad.

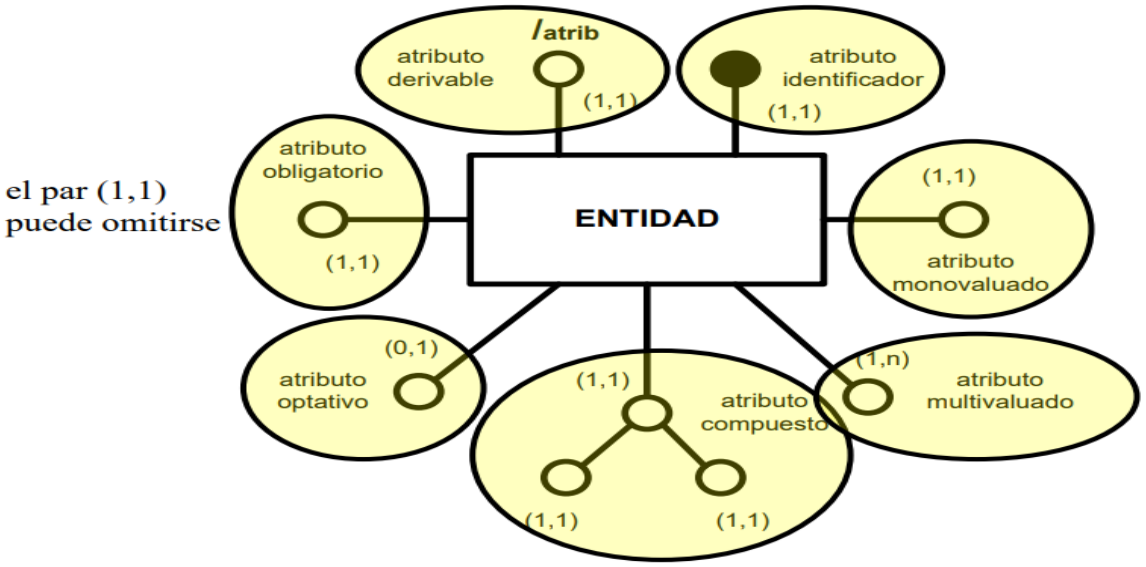


Atributos

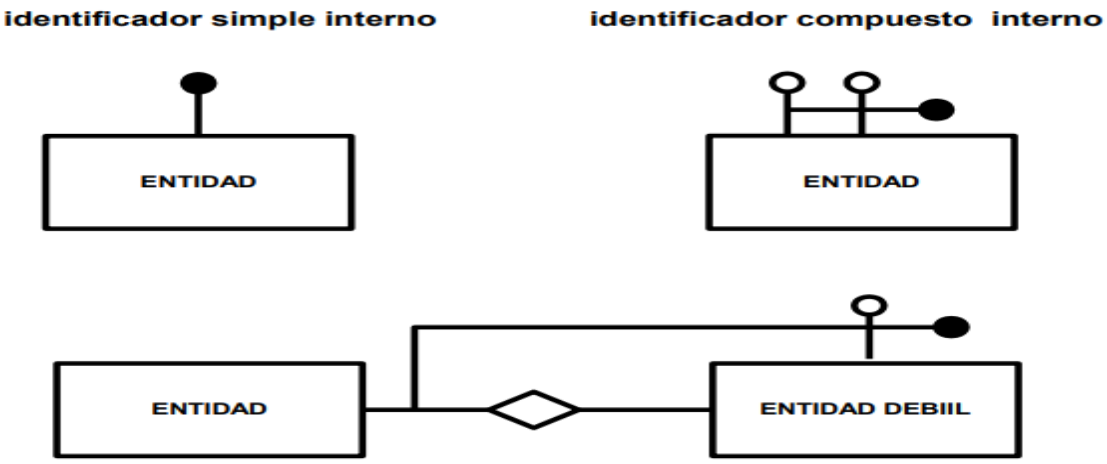
Las entidades poseen atributos que las describen o identifican. Pueden ser:

- Descriptivos
- Derivables
- Simples / Compuestos

Cada atributo está asociado a un dominio particular (tipo de dato).



Identificadores



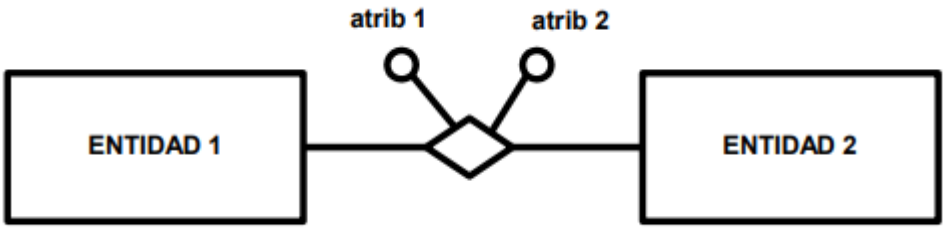
identificador externo, compuesto y mixto

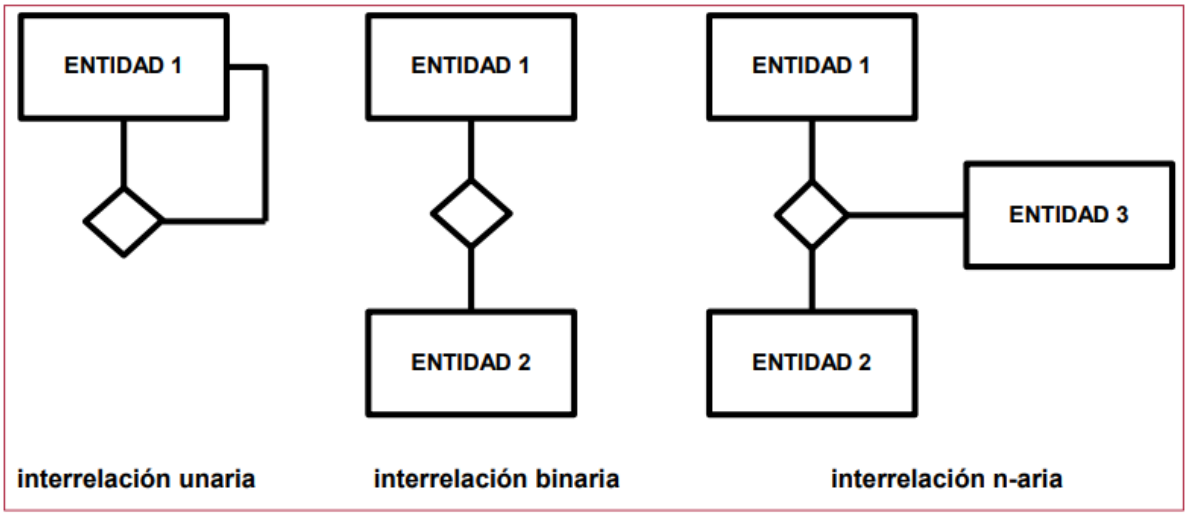
Propiedades de los identificadores:

- Minimalidad.
- Unicidad.

Interrelaciones

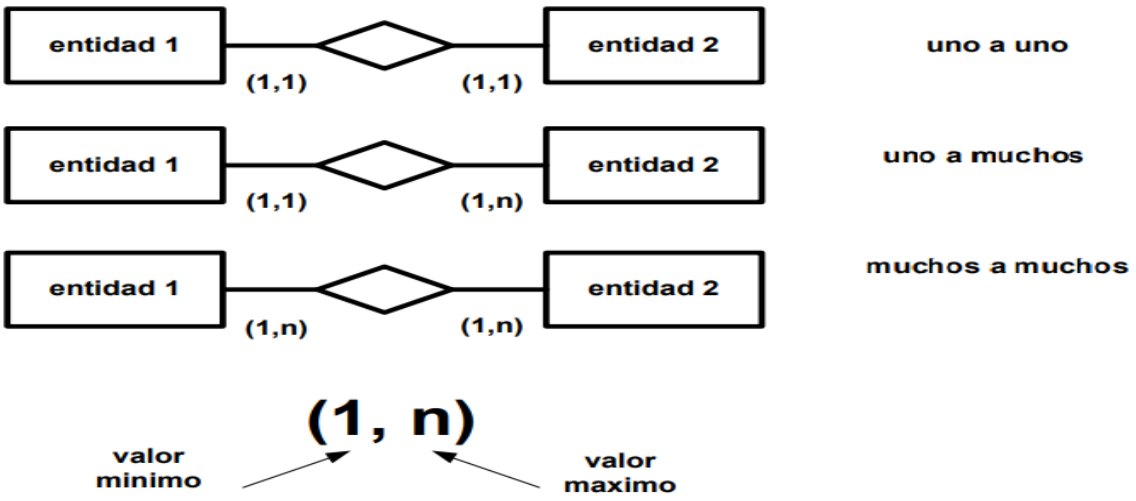
Las interrelaciones pueden tener atributos los cuales se denominan **atributos descriptivos**.





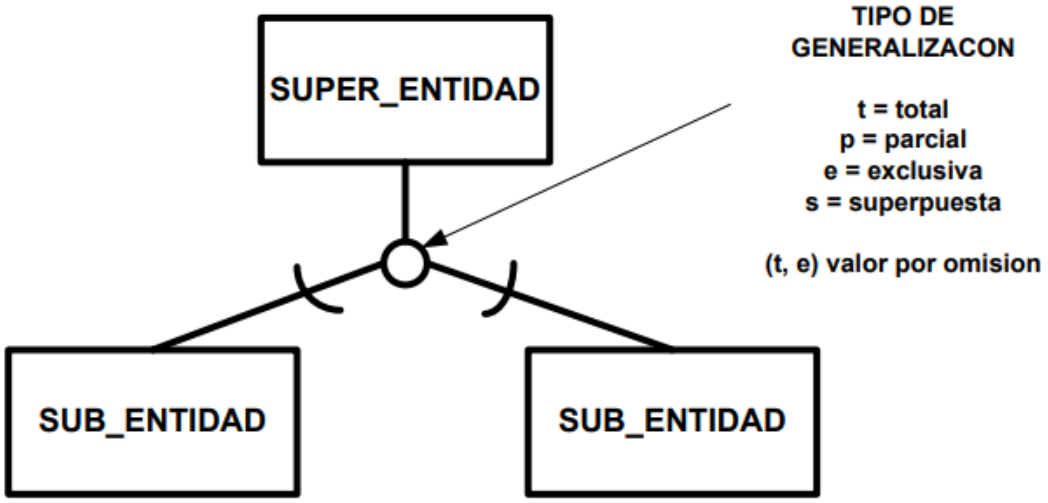
Multiplicidad de relaciones

Para analizar la multiplicidad, parto de un objeto de la entidad y pregunto con cuántos objetos de la otra puede conectarse.



Relaciones de generalización

Las sub entidades heredan atributos, interrelaciones y generalizaciones de la super entidad.



Transformación al modelo lógico

- Las **entidades** se transforman en **tablas**.
- Los **atributos** se transforman en **campos** de la tabla.
- Los **identificadores** se transforman en **claves primarias** de la tabla.
- La **interrelación binaria** (y n-arias) **M:N** se transforman en **Tabla**, el identificador es la unión de los identificadores de la entidades intervinientes, cada uno de ellos es **clave**

foránea con referencia a la entidad (tabla) respectiva. Los atributos descriptivos pasan a ser campos a la tabla generada.

- Las **interrelaciones 1:N, 1:1** se transforman en un atributo identificador del lado “1” pasa como **clave foránea** al lado “n”.
- La **generalización** posee dos opciones:
    1) **Sub entidades** y **sub entidades** se transforman en **tablas**, los atributos de ambas pasan a las tablas respectivas como campos, el identificador de la super entidad será el identificador de la tabla, el atributo identificador en las sub entidades será, además, clave foránea con referencia a la super entidad.
    2) **Desaparece** la **super entidad** y todos los atributos de ésta pasan a ser atributos en las sub entidades (incluyendo el identificador).

Base de Datos

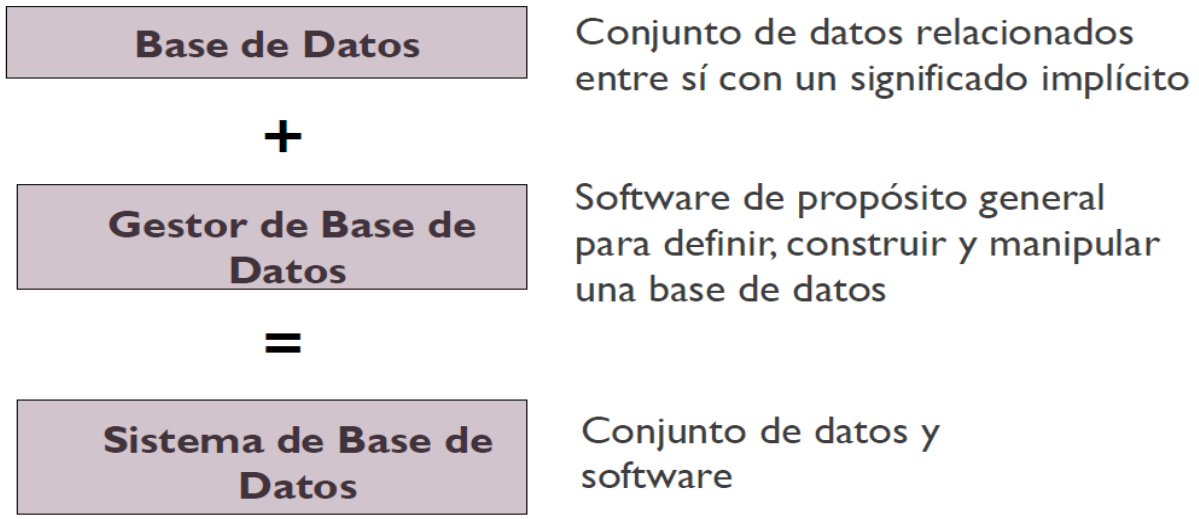
Conjunto de datos relacionados entre sí con un significado implícito.

- Representa un aspecto del mundo real.
- No es una colección aleatoria.
- Se diseña para un propósito específico.

Sistema Gestor de Base de Datos (SGBD)

Software de propósito general para definir, construir y manipular una base de datos, permite:

1. **Definir**: especificar estructuras de datos, tipos y restricciones
2. **Construir**: guardar datos en algún medio de almacenamiento controlado por el SGBD
3. **Manipular**: realizar consultas y actualizaciones



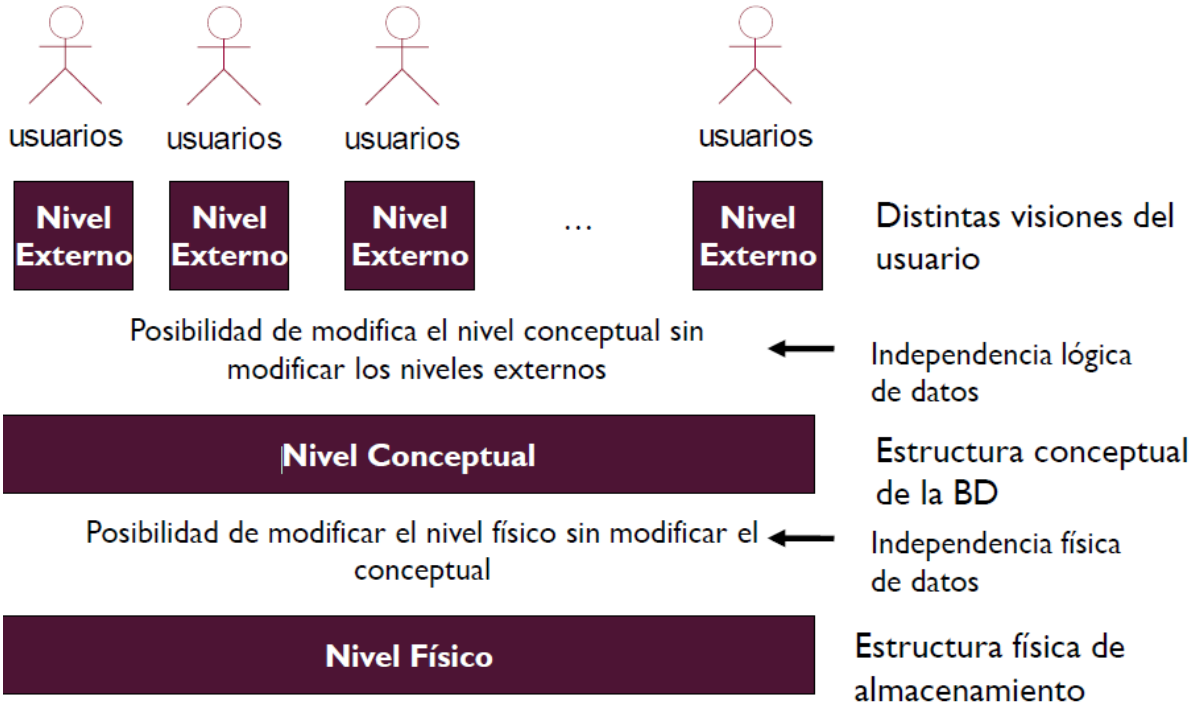
Características de las BD

- **Abstracción de datos**: ofrece una representación conceptual, no incluyen detalles de almacenamiento.
- **Manejo de múltiples vistas de usuario**: cada usuario puede tener una vista diferente de la base de datos.
- **Transacciones multiusuarios**: varios usuarios tienen acceso simultáneo a la base de datos.
- **Múltiples interfaces para diferentes usuarios**.
- **Restricciones de integridad**.
- **Respaldo y recuperación de datos**.

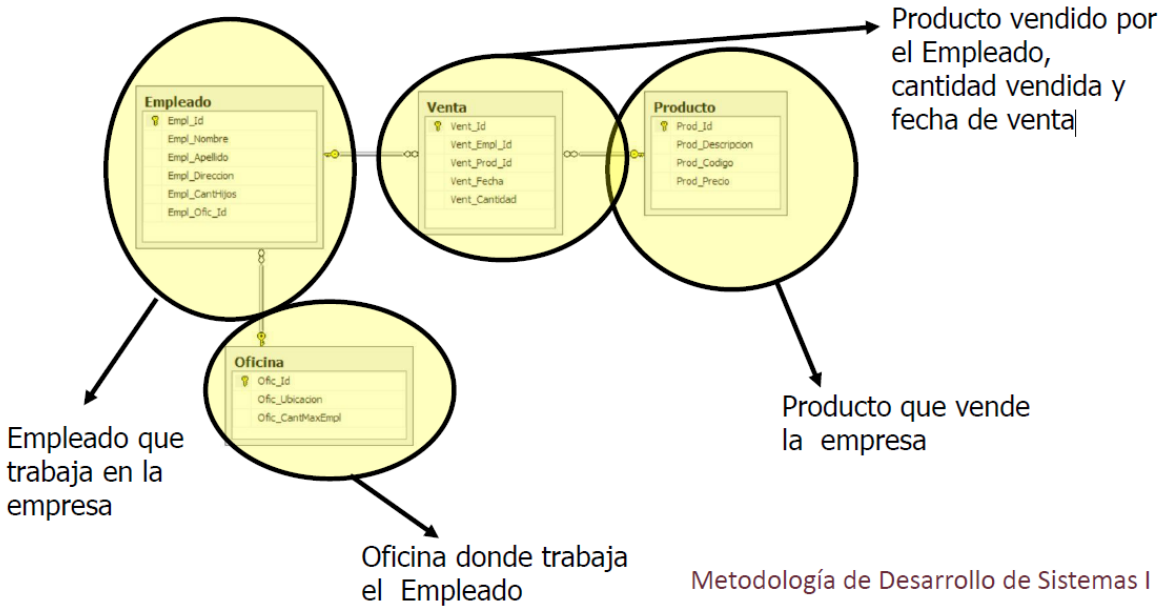
Actores BD

1. **Administrador de la BD**: supervisa y controla los recursos.
2. **Diseñador de la BD**
3. **Usuario Final**

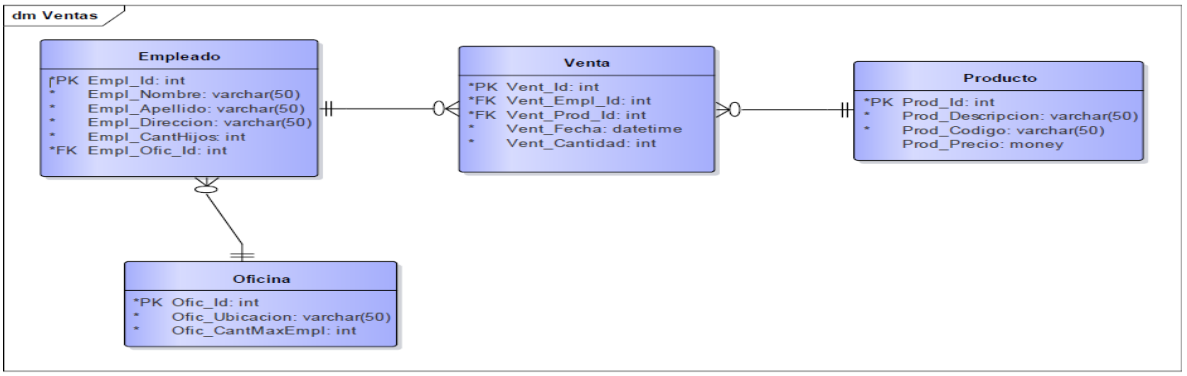
Arquitectura BD

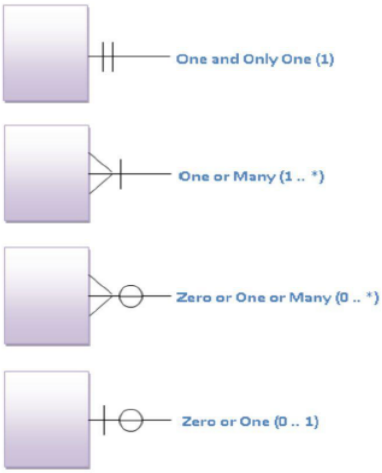


SQL (lenguaje de consulta estructurado)



Notación Martin / Pata de Gallo





Ventajas de notación Martin (Pata de gallo)

- Ampliamente usada
- Muestra al mismo tiempo opcionalidad y cardinalidad máxima.

LDD (**lenguaje de definición de datos**): permite crear la estructuras, modificarlas y borrarlas

LMD (**lenguaje de manipulación de datos**): permite hacer consultas, altas, bajas y modificaciones del contenido de las tablas.

SQL - Sintaxis básica

SELECT campo/s

FROM tabla/s

WHERE <condición>

GROUP BY tabla/s

HAVING <condición>

ORDER BY campo/s

Ejemplo Consultas Básicas

Selecciones todos los campos de la tabla Empleado

```
SELECT *
FROM Empleado;
```

Seleccione el nombre y el apellido del empleado

```
SELECT nombre, apellido
FROM Empleado;
```

Seleccione el nombre y el apellido de los empleados que vivan en Morón

```
SELECT nombre, apellido
FROM Empleado
WHERE direccion ='Moron'
```