

---

## ANÁLISIS DE SISTEMAS 2DO CUATRIMESTRE

---

---

### ENFOQUES SISTÉMICOS (CONCEPTOS DEL 1ER CUATRIMESTRE A RECORDAR)

Características básicas:

- 1) Ir de lo general a lo particular: se visualiza el problema en general y luego se desgrana en sub-conceptos que conforman parte del todo.
- 2) Análisis por niveles progresivos: se definen niveles y se analiza de lo general a lo particular.
- 3) División del sistema en sub sistemas
- 4) Método de prueba y error

Aspectos básicos a evaluar:

- 1) **Eficiencia:** cumplir con los objetivos de manera óptima utilizando la menor cantidad de recursos (tanto HW como SW)
- 2) **Productividad:** consiste en realizar la mayor cantidad de código en el menor tiempo posible (una manera de hacer esto posible es reutilizando código ya escrito).
- 3) **Portabilidad:** consiste en realizar un sistema independientemente de la plataforma tecnológica disponible. Esto quiere decir que el modelo lógico de la solución debe ser el mismo a pesar del contexto en el que se encuentre (el equipamiento de hardware que se posee), se debe poder adaptar a los distintos escenarios. Dichos cambios en la plataforma impactan en el modelo físico y producen cambios en la interfaz, por lo que es recomendable documentar a medida de que el proyecto avance para así no realizar el sistema de cero.
- 4) **Mantenibilidad:** el mantenimiento correctivo enmienda errores que deberían de haber sido contemplados desde el inicio vs el mantenimiento perfectivo y adaptativo que dependen de la flexibilidad del sistema. El equipo designado a realizar el mantenimiento se denomina soporte.
- 5) **Documentación:** a medida que avanza el sistema hay que documentarlo. En el paradigma tradicional, a diferencia del paradigma estructurado y el de orientado a objetos, es un proceso lento debido a que hay que esperar la finalización de cada etapa para poder documentarla.

El **PARADIGMA** consiste es una manera de llevar a cabo la metodología elegida, determinando en que etapas se debe hacer más énfasis. Existen 3 paradigmas ordenados según el grado de limitaciones que poseen (de mayor a menor), y por lo tanto, en la evolución generada a través del tiempo.



### COMPARACIÓN EN METODOLOGÍAS

Se determina las pautas, un lenguaje comprensible (el cual es distinto para cada paradigma) y un estandar para cada uno.

Además se determina el tipo de enfoque a utilizar:

BOTTOM – UP: consiste en ir de lo particular a lo general, es decir, se realizan distintas soluciones que luego se consolidan en una solución general. Este es el tipo de enfoque del paradigma tradicional y el orientado a objetos.

TOP DOWN: consiste en ir de lo general a lo particular. Este es el tipo de enfoque del paradigma estructurado.

#### PARADIGMA TRADICIONAL:

✓ Posee grandes limitaciones operativas:

- En un principio, no es adaptable a la hora de realizar cambios en la plataforma tecnológica (impacta sobre el modelo lógico), es decir no se encuentra beneficiada en cuanto al aspecto de portabilidad.
- Se encuentra asociado con el ciclo de vida de cascada, lo que conlleva el hecho de que se debe esperar una gran cantidad de tiempo para esperar resultados, generando impaciencia en el usuario y castigando la productividad.
- Presenta inconvenientes al documentar.
- Al ser Bottom Up no posee mucha flexibilidad al hacer el mantenimiento.
- No se modeliza, es decir, no se realiza una abstracción del modelo lógico y físico del sistema

### TRADICIONAL

Relevamiento  
Análisis y diseño  
Implementación  
Mantenimiento

En igual cantidad de tiempo (puede ser misma metodología o no), los paradigmas evolucionaron y se enfocaron más en el análisis y diseño del sistema para que el mantenimiento sea más flexible y la implementación más rápida.

Además, esta modificación permite una mejora en la visibilidad externa. Sin embargo, lo que el usuario ve, posee un gran trasfondo (un muy buen análisis y diseño).

### ESTRUCTURADO Y O.O

Relevamiento  
Análisis y diseño  
Implementación  
Mantenimiento

Por otro lado, mientras que los paradigmas estructurado y orientado a objetos modelizan, el tradicional no lo hace.

Los paradigmas van evolucionando a través del tiempo: "lo único permanente es el cambio". Para esto se deben tener en cuenta distintos aspectos a mejorar:

- Mayor interacción con el usuario
- Mejor captura de los requerimientos.
- Posibilidad de "refinar los requerimientos": agregarle un mayor valor, profundizarlos.
- Impacto de mejora en la calidad del SW: a partir de la búsqueda de una mejor aceptación por parte del usuario (Mantener una "política de usuario", hacerlo partícipe) y del perfeccionamiento del mantenimiento del sistema.

Abstracción: simplificación de la realidad.

El modelo físico del sistema define cómo se hará y el modelo lógico determina qué se hará.

ASPECTO	TRADICIONAL	ESTR.	O.O
PRODUCTIVIDAD	-	+	'++
PORTABILIDAD	-	+	+
EFICIENCIA	+	+	++
MANTENIBILIDAD	-	+	++
DOCUMENTACIÓN	-	+	+

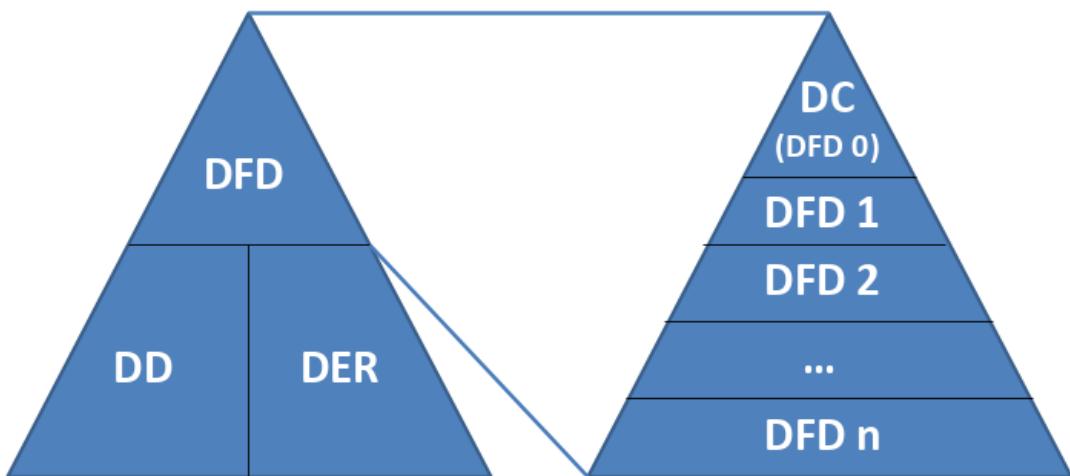
PARADIGMA ESTRUCTURADO:

- ✓ Posee como componentes DATOS + PROCESOS .
- ✓ Hace énfasis en los PROCESOS .
- ✓ Utiliza como enfoque TOP DOWN, lo cual favorece la documentación.
- ✓ Refinar RQ (Sistemas - Usuario) .
- ✓ Posee como herramientas: DFD (Diagrama de Flujo de Datos), DD (Diccionario de Datos) y DER (Diagrama de Entidad / Relación).
- ✓ Se realizar en primer lugar el modelo lógico y luego el físico.

PARADIGMA ORIENTADO A OBJETOS:

- ✓ A diferencia con el estructurado, el modelo lógico y físico se encuentran superpuestos (lo cual mejora la productividad y la documentación).
- ✓ Además posee un enfoque BOTTOM UP.
- ✓ Se reutilizan partes de otros sistemas/ librerías prehechas (lo cual impacta en la positivamente tanto en la productividad, al generar una mayor cantidad de código con la menor cantidad de recursos, como en la mantenibilidad).
- ✓ La comunicación necesaria para realizar este tipo de paradigma influye en los costos que conlleva.
- ✓ La implementación se realiza de forma parcial.

## ANÁLISIS ESTRUCTURADO



**DFD:** Diagrama de Flujo de Datos

**DD:** Diccionario de Datos

**DER:** Diagrama de Entidad-Relación

**DC:** Diagrama de Contexto

Ventajas:

- ✓ Mejora interacción con usuario y con el equipo de proyecto.
- ✓ Provee herramientas de documentación.
- ✓ Permite refinar el modelo lógico al mismo tiempo que se lo documenta.
- ✓ Niveles progresivos de análisis, de diferente profundidad.
- ✓ Abstracción de los aspectos físicos de la implementación.

---

### TÉCNICAS DEL ANÁLISIS ESTRUCTURADO

Como se dijo previamente (en el resumen anterior) en la etapa de análisis se tiende a disminuir la participación del usuario (en el diseño y desarrollo del sistema prácticamente este no participa sino que son dos etapas totalmente técnicas), es por esto que se pretende refinar en esta etapa completamente el modelo lógico de la solución (definir completamente los requerimientos del usuario).

En cuanto a los gráficos utilizados, se determina un lenguaje comprensible sin pensar en los aspectos físicos de la implementación.

*Diagrama de Flujo de Datos (DFD)*

La construcción de la solución no es taxativa, es decir, la cantidad de niveles para llegar a ella puede variar. Sin embargo, conceptualmente la solución es la misma. Existen ciertos factores que influyen en la cantidad de niveles empleados dentro de los cuales se encuentran:

- i) La experiencia del equipo de sistemas.
- ii) La capacidad de análisis que tiene, es decir, la capacidad de abstracción (simplificación de la realidad, proceso mental en el que se determina cómo se le va a dar forma al modelo lógico).
- iii) El conocimiento que posee el equipo.

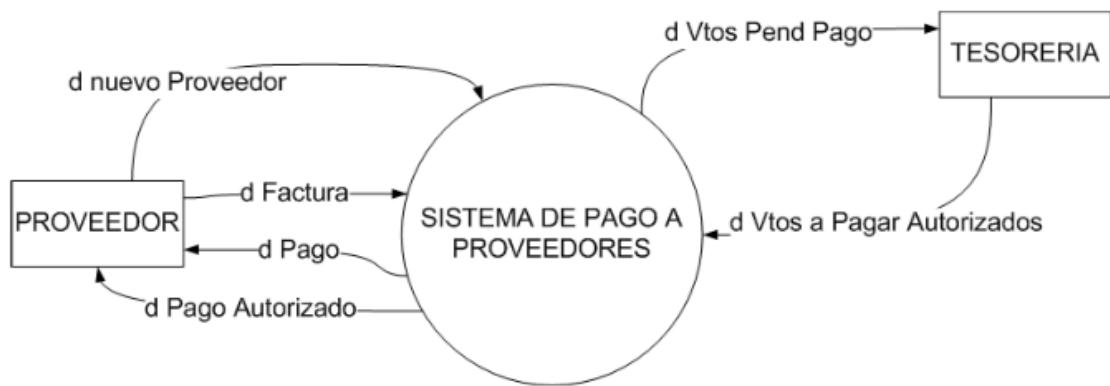
A mayor complejidad del sistema, mayor cantidad de niveles tendrá.

#### Diagrama del Contexto (DC):

- Representa la abstracción de más alto nivel del Sistema, es decir, es la visión más general de la solución del sistema.
- Se considera al Sistema como un único proceso, sin importar su estructura interna.
- Define las entidades externas con las que el Sistema interactúa (es decir su entorno), y sus entradas y salidas respectivas.
- También conocido como DFD nivel 0 (cero).

Gráficamente:

1. Se representa a la funcionalidad completa del sistema con un círculo.
2. Las entidades externas (al sistema) con las que interactua se representan con un rectángulo.
  - a. No hay flujo de información entre ellas.
  - b. Estas pueden ser: una organización externa (**PROVEEDORES**), un área de la organización (**COMPRAS**), un rol de la organización (**JEFE DE DEPÓSITO**) u otro sistema (**SISTEMA DE LIQUIDACIÓN DE SUELDOS**).
3. El flujo de datos existente que representa una relación de datos lógicos entre el entorno y el Sistema con abstracción del medio físico.
  - a. Es un arco con sentido unidireccional.
  - b. Si hay reciprocidad, se realizan dos arcos (uno de ida y otro de vuelta).
  - c. NO EXISTE UN SISTEMA SIN FLUJO DE SALIDA (debido a que no soluciona nada), NI UN SISTEMA SIN ENTRADAS (no puede no haber estímulo): TODO SISTEMA DEBE DE TENER **AL MENOS 1 ENTRADA Y 1 SALIDA**.
  - d. El nombre del flujo de datos no se repite.
  - e. No interesa el medio: si paga por computo o personalmente (da lo mismo), o si la factura es original o duplicada (también -> da el dato de la factura).

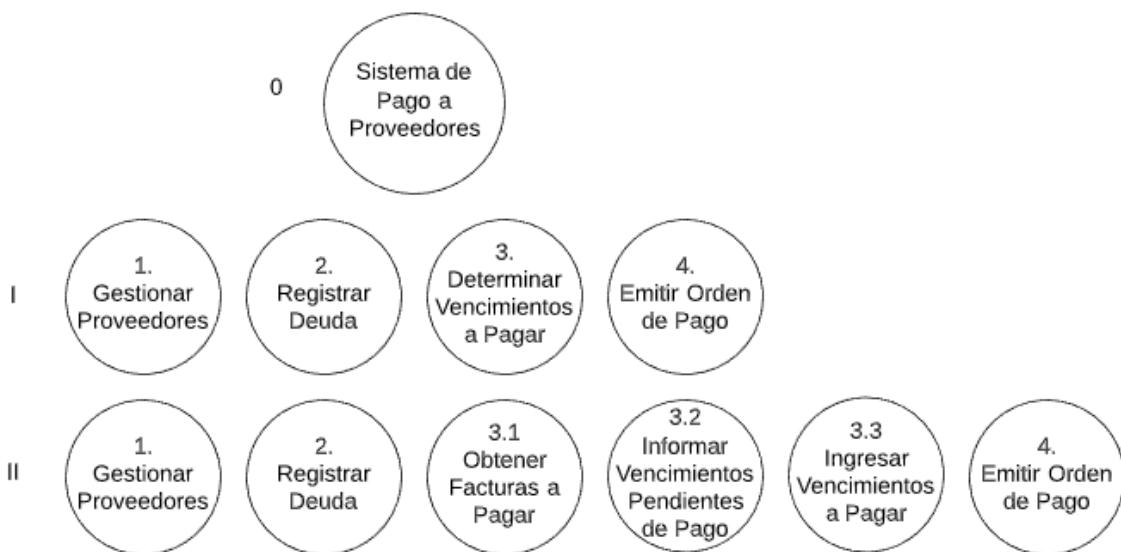


#### **Lista de Eventos:**

- Es un listado jerárquico que determina las funciones o procesos que se utilizarán en el DFD.
- Se validan las entidades externas determinadas en el diagrama de contexto (se repiten).
- Cada proceso que cumple con el objetivo se identifica con un N°, el cual no implica el orden de ejecución sino que determina el código jerárquico de la lista de eventos (para qué nivel es). El orden de ejecución se encuentra definido por la disponibilidad de datos necesarios para que un proceso se efectue.

Ejemplo:

1. Gestionar Proveedores.
2. Registrar Deuda.
3. Determinar Vencimientos a Pagar
  - 3.1. Obtener Facturas a Pagar.
  - 3.2. Informar Vencimientos Pendientes de Pago.
  - 3.3. Ingresar Vencimientos a Pagar.
4. Emitir Orden de Pago.



### Diagrama de Flujo de Datos (DFD):

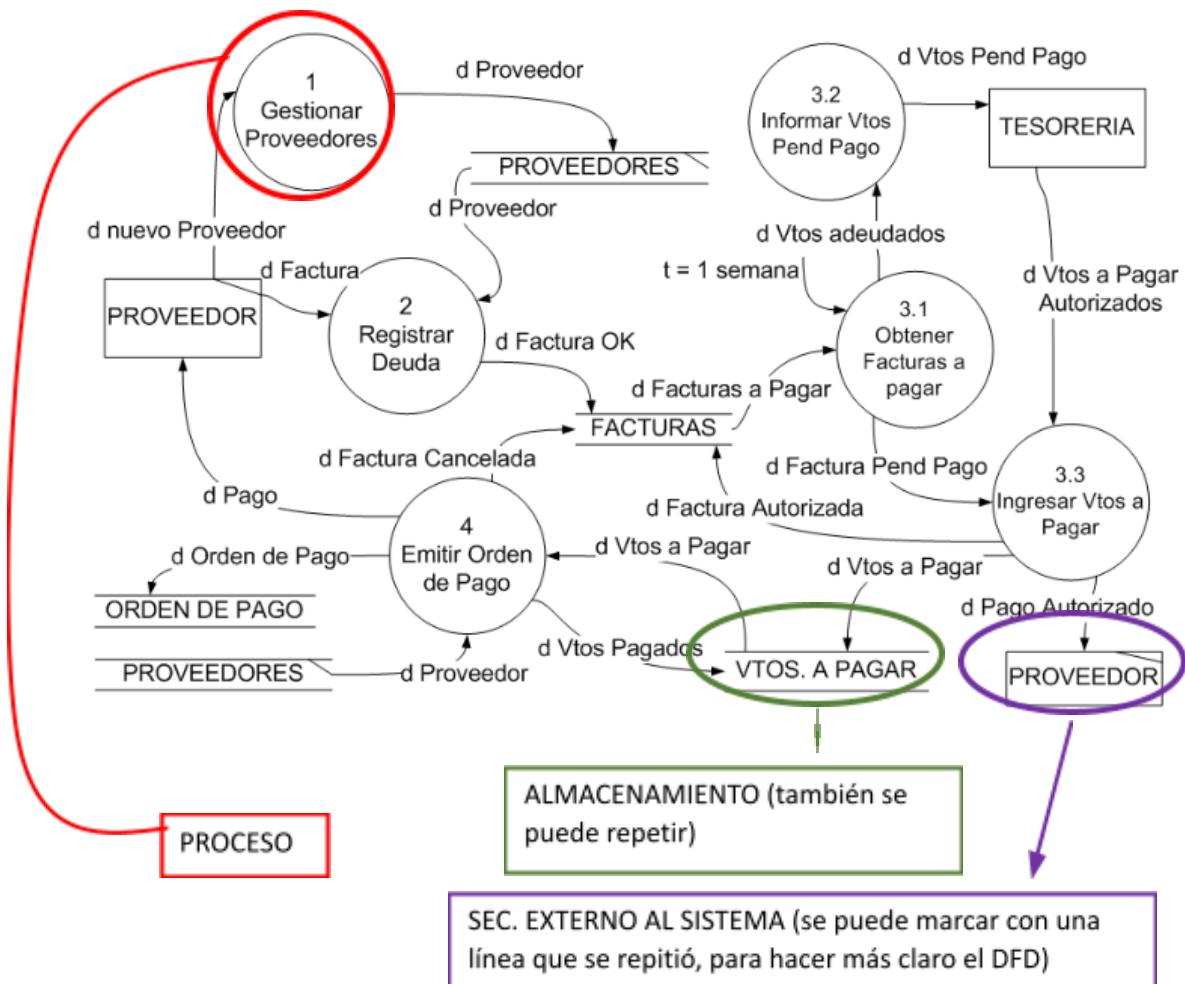
Representa la descripción gráfica de los subsistemas lógicos que componen el sistema principal, y sus movimientos y almacenes lógicos de datos.

#### Simbología:

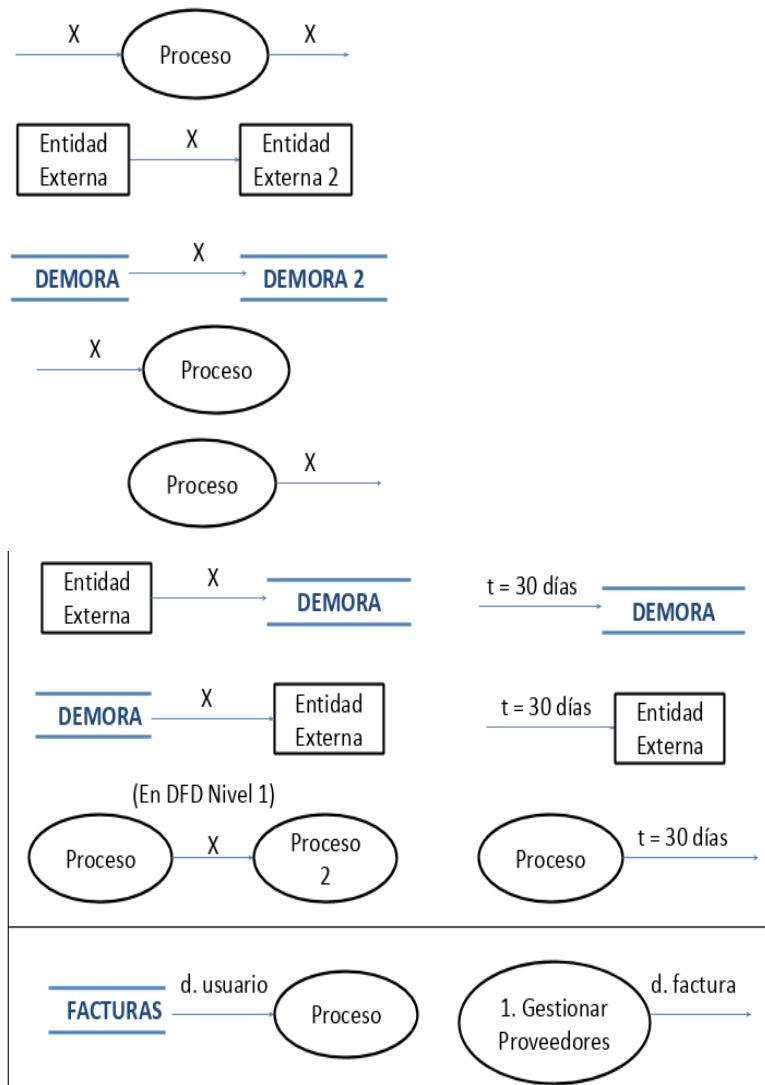
- A. Entidades externas: deben ser las mismas que en el DC.
- B. Proceso o función: es un subsistema que representa una función lógica interna del sistema.
  - a. Se identifica con un código jerárquico definido en la Lista de Eventos, que indica el nivel del Proceso.
  - b. Representan un trabajo que realiza el sistema, provocando siempre un cambio o transformación en los datos.
  - c. El nombre es una expresión que comienza con un verbo infinitivo y representa la actividad realizada por el proceso.
  - d. En el primer nivel, los procesos se comunican a través de los almacenamientos. A partir del segundo nivel, los procesos pueden comunicarse entre sí.
- C. Almacenamiento o Demora: representa un depósito lógico de datos en el sistema, donde los datos se almacenan temporalmente para ser utilizados en otro momento.
  - a. Los flujos de datos de origen significan una consulta de los datos por un proceso.
  - b. Los flujos de datos de destino significan la inserción, actualización o eliminación de los datos por un proceso.
  - c. No representa ni especifica el tipo de almacenamiento físico (archivo, BD, etc).
  - d. El nombre representa la estructura de datos que contiene, suele ser un sustantivo en plural.
- D. Flujo de Datos: representa una relación de datos lógicos, con abstracción del medio físico de la transferencia.
  - a. El nombre representa la Estructura de Datos que contiene la información transferida. Usamos la expresión “datos de ...” haciendo hincapié en la abstracción del soporte físico.
  - b. No debería haber dos FD en el sistema con el mismo nombre (excepto FD entrante y saliente a un mismo Almacenamiento).
- E. Flujo Temporal: representa la activación periódica de un proceso con una frecuencia determinada.
  - a. Origen: No posee
  - b. Destino: Proceso
  - c. En el nombre se indica la frecuencia de activación
- F. Flujo Activador: es el flujo de datos que representa la “activación” de un proceso, provocando su ejecución:
  - a. Puede ser un flujo de datos o un flujo temporal.
  - b. En general hay un solo flujo activador por cada proceso pero existen excepciones en donde hay mas de 1.
  - c. El concepto también es aplicable al Diagrama de Contexto (flujo activador del Sistema): TODO SISTEMA TIENE 1 FLUJO ACTIVADOR (QUE LO “DISPARA”).

#### Detalles generales:

- La cantidad de niveles depende de la experiencia en el dominio del problema, y deben hacerse tantos niveles necesarios hasta que el DFD sea “técnicamente correcto”.
- El usuario participa generalmente hasta el nivel 1 o 2 para validar requisitos.
- El detalle técnico comienza a participar a partir de niveles 2 o 3, donde los requisitos funcionales ya están definidos y validados por el usuario.
- El tratamiento de errores en general se comienza a evaluar del nivel 2 en adelante (resultado negativo de los controles).
- Primitiva: Es el DFD de máximo nivel de detalle.
- Las Estructura de Datos (flujos y almacenamientos) se especifican en DD (Diccionario de Datos).
- El modelo de datos almacenados se especifica en DER (Diagrama Entidad-Relación).



NO SE PUEDE HACER:



### Diccionario de Datos (DD)

- Es un listado ordenado alfabéticamente que describe todos los datos que administra el sistema:

- Flujos de Datos (Excepto Flujos Temporales)
- Almacenamientos

Estos deben de estar balanceados, es decir, que todo flujo de datos y almacenamiento perteneciente al DD debe estar en el DFD, y viceversa.

El diccionario de Datos...

... elimina redundancias

... permite validar DFD

... no tiene por objetivo la interacción con el usuario: es una herramienta de documentación y comunicación INTERNA (más técnico) debido a que los requerimientos ya están definidos.

... está compuesto por:

- Elemento de dato: Dato atómico
  - Estructura de dato: Conjunto de datos que tienen una relación lógica.
- ... su simbología es:

Símbolo	Significado	Observaciones
=	Está compuesto de	
+	Y	
[ ]	Alternativas	Se separan con PIPE (   )
( )	Opcional	
n{ }m	Iteración	n y m: cotas para requisitos funcionales
@	Identificador	
* *	Comentarios	Uso: * Comentarios acerca de la estructura de datos *
“ ”	Valor literal	Ejemplo: [ “SI”   “NO” ]

Un claro ejemplo es:

- d\_Factura\_OK = @CODIGO\_PROVEEDOR + @LETRA + @NUMERO + FECHA\_EMISION + {FECHA\_VTO + IMPORTE} + [“PENDIENTE” | “AUTORIZADA” | “CANCELADA”] \* FECHA\_VTO es mayor o igual a FECHA\_EMISION \*
- d\_nuevo\_Proveedor = RAZON\_SOCIAL + DIRECCION + EMAIL
- d\_Proveedor = @CODIGO + d\_nuevo\_Proveedor
- DIRECCION = CALLE + NUMERO + (PISO) + (DEPTO) + COD\_POSTAL + PROVINCIA + PAIS
- FACTURAS = d\_Factura\_OK
- PROVEEDORES = d\_Proveedor

#### +INFO

{ } sin cotas -> no posee límites.

pueden meterse estructuras dentro de otras como en d\_Proveedor que posee d\_nuevo\_Proveedor

**EXISTEN CLAVES:** naturales (permiten al usuario un mayor entendimiento -> @) y subrogadas

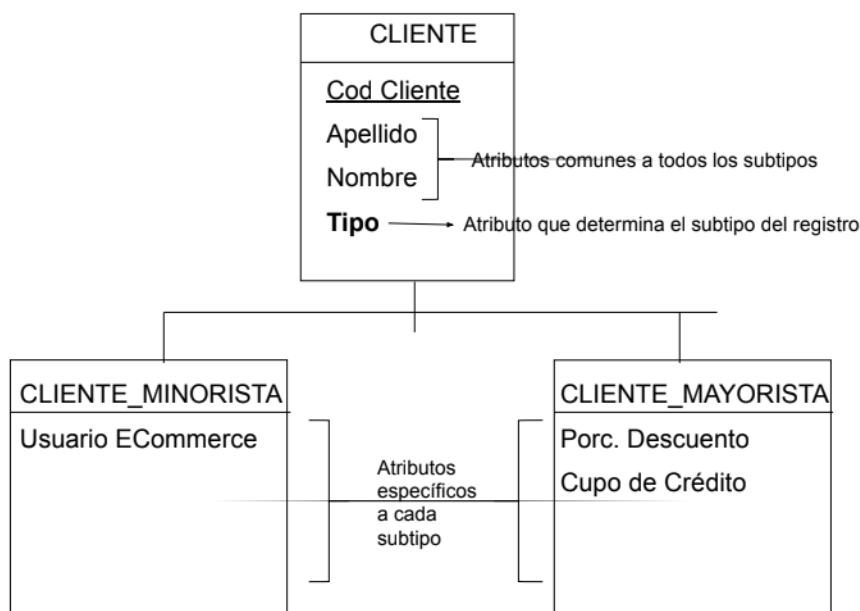
#### *Diagrama de Entidad Relación (DER)*

Representa mediante un modelo de red los datos almacenados o persistidos en el sistema, y sus relaciones.

- Los elementos que lo componen son:
  - Entidades:
  - Es una idea relevante del negocio que estoy modelando.

- Es un contexto determinado del cual deseamos guardar información para utilizarla en otro momento
- Se designa con un sustantivo en singular. Ej: Factura, Cliente
- Posee una clave que permite identificar de forma única una ocurrencia de la entidad
- Está compuesta por Atributos y Relaciones
- Se clasifican en:
  - a) Fundamentales o Fuertes: su clave está formada únicamente por atributos propios.
  - b) Dependientes o Débiles: su clave contiene al menos una clave foránea (relación).
    1. Asociativa: clave principal sin atributos propios.
    2. Atributiva: clave principal con relación y atributos propios.
  - c) Subtipo – Supertipo: tipos de entidades de una o más subcategorías, conectadas por una relación. La entidad Supertipo tiene un atributo que permite discernir de qué subtipo es una instancia determinada. Ejemplo:

### SUPERTIPO – SUBTIPO: ATRIBUTOS



#### - Atributos:

- Son características o propiedades asociadas a la entidad.
- Toman valor en una instancia particular.
- Son valores atómicos.

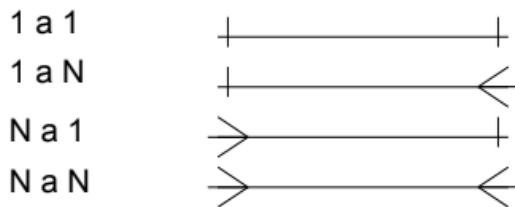
#### - Relaciones:

- Indican cómo se relacionan las entidades.

- Se nombran con una letra R seguida de un subíndice “n” (R<sub>n</sub>). El “n” es único y no indica orden.

- CARDINALIDAD:

- Número máximo de instancias con las que puede relacionarse una entidad de un extremo de la relación con la del otro extremo de la relación.

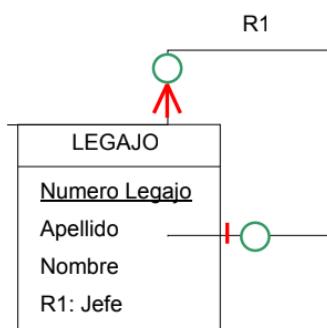


- MODALIDAD

- Número mínimo de instancias con las que puede relacionarse una entidad de un extremo de la relación con la del otro extremo de la relación.



Existe la relación UNARIA: es la relación de una entidad con otra instancia de sí misma.



- Clave: identifica inequívocamente una sola instancia por entidad

**SUPERCLAVE**: conjunto de uno o más atributos (incluyendo relaciones) que permiten identificar de forma única una ocurrencia o instancia de una entidad (concepto de UNICIDAD)

**CLAVE CANDIDATA o SUPERCLAVE MÍNIMA**: aquellas Superclaves para las cuales ningún subconjunto propio de atributos es a su vez una Superclave (concepto de MINIMALIDAD)

**CLAVE PRINCIPAL (Primary Key, PK)**: conjunto de atributos que nos permite identificar inequivocadamente a la entidad. Es una de las Claves Candidatas por lo que cumple con unicidad y minimalidad. Al conjunto de Claves Candidatas no elegidas como clave principal se las denominan **Clave Alternativa**.

Nota: en el diagrama los atributos que conforman la Clave Principal van subrayados para distinguirlos del resto.

**CLAVE FORANEA o SECUNDARIA (Foreign Key, FK)**: clave de una entidad formada por un atributo de otra entidad con la cual ésta se relaciona.

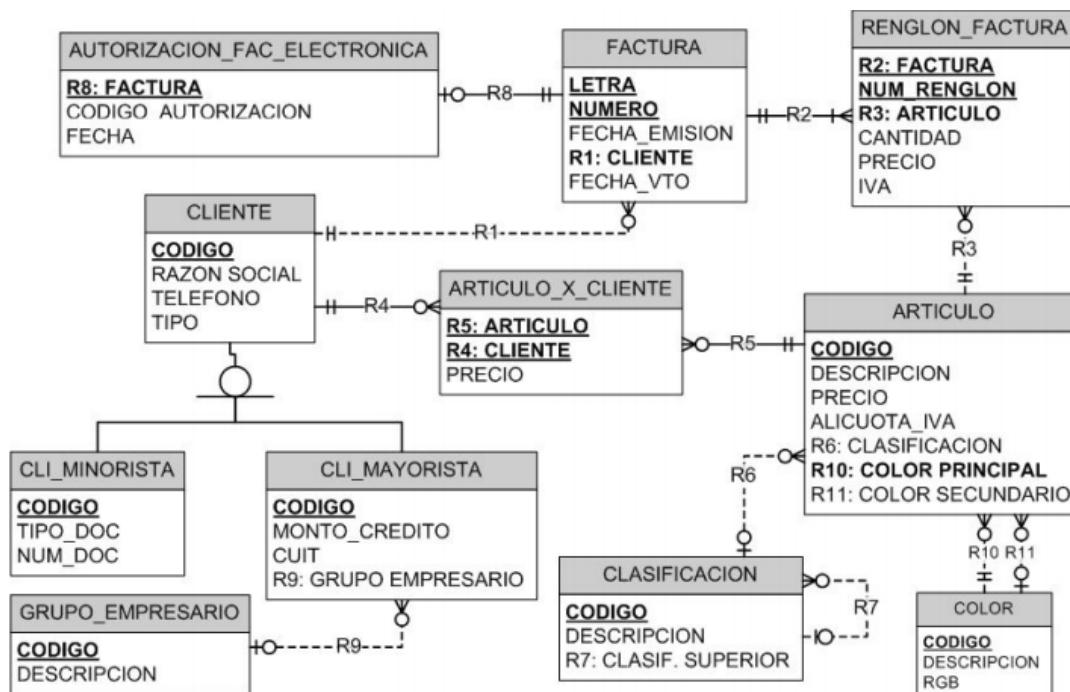
## NORMALIZACIÓN:

Proceso mediante el cual se aplican una serie de reglas (Formas Normales) a las entidades y relaciones de un modelo con el objetivo de: eliminar redundancias de datos, evitar problemas de actualización y proteger la integridad del modelo.

Para ello:

- 1) Se eliminan los atributos calculados (para una mayor mantenibilidad al tratar de evitar las contradicciones de información); toda entidad tiene clave (no puede poseer atributos); todo campo no clave, depende de la misma; se eliminan “grupos repetitivos”; cada atributo contiene datos “atómicos”.
- 2) Los campos no claves dependen totalmente de la clave.
- 3) No existen dependencias transitivas entre atributos no clave.

Ejemplo:



Por último, el DER...

... refleja los datos, no lo que se hace con ellos.

... es independiente de la tecnología de base de datos, pero ésta debe responder al modelo relacional.

... lógico no tiene restricciones de espacio de almacenamiento ni performance.

... su proceso de construcción es flexible.

Es un lenguaje estándar de **modelado** (no para programación), es decir, es utilizado para la visualización (visión externa), especificación, construcción (diseño) y documentación de sistemas de software.

Los diagramas que genera son aplicables a todo ciclo de vida y es considerado como un complemento para metodologías Orientadas a Objetos (para el cual el modeo lógico y el físico se encuentran superpuestos).

No es prescriptivo, lo cual significa que no proporciona un método de desarrollo sino que es independiente del proceso.

Además, cada modelo enfatiza una determinada etapa del sistema.

Las herramientas CASE, prácticamente propias del paradigma Orientado a Objetos, permiten potenciar el desarrollo del sistema (mejoras, optimizaciones a nivel de software engineering): habilitan el pasaje de los diagramas generados en el análisis al código. Así mismo permiten realizar la ingeniería inversa, lo cual consiste en generar el/los diagrama/s (análisis) a partir del código.



Herramientas CASE:

ANÁLISIS CÓDIGO

Ingeniería Inversa:

CÓDIGO ANÁLISIS

Ventajas de CASE: en caso de perder el análisis hecho (los diagramas), al poder realizar la ingeniería inversa se pueden recuperar por lo que permite re-asegurar la inversión del proceso ANÁLISIS-CÓDIGO.

Desventajas de CASE: la herramienta que brinda soporte a la ingeniería inversa es muy cara.

- |  |  |
|--|--|
| <ul style="list-style-type: none"><li>▪ <b>Diagramas de Estructura</b><br/>“Aspectos Estáticos”<ul style="list-style-type: none"><li>▫ Paquetes</li><li>▫ Clases</li><li>▫ Objetos</li><li>▫ Componentes</li><li>▫ Estructura Compuesta</li><li>▫ Despliegue</li></ul></li></ul> | <ul style="list-style-type: none"><li>▪ <b>Diagramas de Comportamiento</b><br/>“Aspectos Dinámicos”<ul style="list-style-type: none"><li>▫ Actividad</li><li>▫ Casos de Uso</li><li>▫ Estados</li><li>▫ Interacción<ul style="list-style-type: none"><li>▫ Colaboración/Comunicación</li><li>▫ Secuencia</li><li>▫ Tiempos</li><li>▫ Vista-Interacción</li></ul></li></ul></li></ul> |
|--|--|

Los diagramas de Estructura permiten determinar la funcionalidad del sistema, mientras que los diagramas de Comportamiento determinan las disfuncionabilidades.

Por otro lado, los diagramas se asocian a distintas etapas del ciclo de vida, por ejemplo:

ANÁLISIS

DISEÑO

IMPLEMENTACIÓN

D. Caso de Uso	D. Secuencia	D. Colaboración
D. Estados	D. Objetos	D. Tiempos
	D. Clases	D. Despliegue

### *Casos de Uso*

- Unidad atómica de comportamiento del sistema (representada con un óvalo)
- Descripción de una secuencia de acciones (de forma narrativa), y sus variantes
- Interactua con uno o más Actores (rol que se relaciona directamente con el sistema: una persona puede ser distintos actores, **dependiendo de su funcionalidad**) intercambiando datos para lograr un objetivo.
- El actor es el entorno con el que se relaciona el sistema (puede ser alguien externo a la organización, un sector de la organización, otro sistema o un rol de la organización) conceptualmente se lo puede relacionar con una entidad externa.
- En los casos de uso se tienen en cuenta únicamente los requisitos funcionales del sistema.

¿Para qué sirven?

- Modelar el contexto de un sistema.
- Identificar y organizar los Actores
- Medio para capturar los requisitos funcionales desde el punto de vista del usuario.
- Documentar los requisitos de un sistema, sus funciones y los roles de los actores intervenientes.
- Acordar con el cliente los requisitos (contrato).
- Generar documentación de usuario y pruebas funcionales en paralelo con el desarrollo

#### ▪ Tipos

##### ▫ Caso de Uso de Negocio:

- Representa la secuencia de acciones de un circuito administrativo de la organización desde un alto nivel de análisis. Representa un proceso o función del negocio.
- En otras palabras consiste en la representación del dominio del problema desde el punto de vista administrativo (un esquema del circuito administrativo de la organización).
- Intervienen Actores de Negocio: Alguien que se relaciona con la organización e interviene en forma directa en el CUN (Caso de Uso de Negocio). Ej: Cliente, Jefe de Ventas, AFIP.
- Describe el comportamiento del negocio como una interacción entre el mismo (la organización) y los actores de negocio, y su resultado de interés para estos últimos.
- Se refieren al concepto de “Sistemas de Información” pero pueden (o no) estar informatizados por software, o estarlo en parte.

##### ▫ Caso de Uso de Sistema:

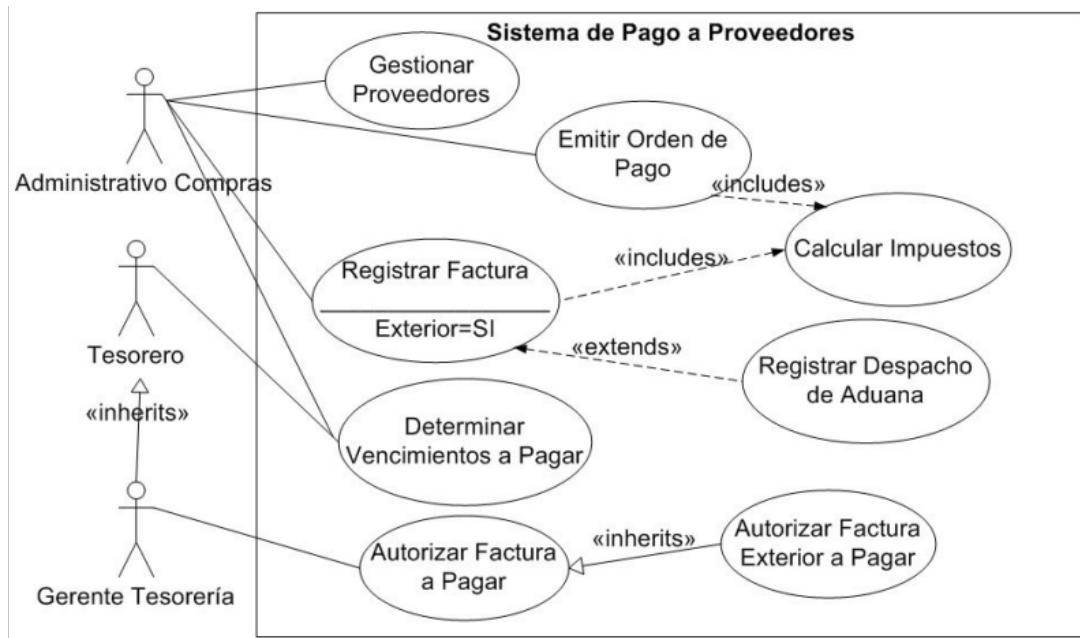
- Técnica para capturar, documentar, comunicar y validar requerimientos de un “Sistema Software” (SI O SI INFORMATIZADO).
- Describe el comportamiento esperado de un sistema, como una interacción entre éste y el usuario, para dar al usuario un resultado de valor.

- El conjunto de Casos de Uso permite definir el alcance funcional del sistema, es por eso que los Actores del Sistema que intervienen (roles de interacción directa con el sistema) se encuentran definidos por su función (independiente de la persona física).
- No especifica la forma en que quedará implementado. Aborda el “qué” y no el “cómo”.
- No necesariamente ligado a enfoque Orientado a Objetos.
- Modelo de Casos de Uso: Se compone de:
  - Diagrama de CU (UML): Ilustra la relación entre actores y CU del Sistema
  - Especificación de CU

### **Simbología del Diagrama de Caso de Uso**

- Actor: representa un rol de interacción directa con el sistema. Puede ser un usuario u otro sistema. Un usuario particular puede estar representado por distintos actores en diferentes momentos de su interacción con el sistema (ya que cada actor se encuentra definido por su función y es independiente de la persona física). Además, un actor puede **heredar** de otro sus funcionalidades.
- Caso de Uso: describe las acciones que el sistema ejecuta para proporcionar un resultado de valor para el actor vinculado. El nombre del caso de uso es un verbo en forma infinitiva.
- Asociación / Comunicación: establece un vínculo entre un actor y un caso de uso. El actor interviene directamente en las acciones definidas por el Caso de Uso. La línea que representa dicho vínculo no posee orientación con flecha.
  - Relación de inclusión (includes): es una asociación con sentido en la que el CU base incorpora el comportamiento de otro (CU incluído). Cada ejecución del CU base implica siempre la ejecución del CU incluido. El objetivo es darle flexibilidad al sistema, agregarle una mayor mantenibilidad al no modificar el caso de uso base y mejorar la productividad al reutilizar el comportamiento del CU incluído desde varios CU base.
  - Relación de Extensión (extends): es una asociación con sentido en la que el CU base incorpora de manera condicional el comportamiento del CU extendido. Modela comportamiento opcional del sistema, (controlado por decisiones del usuario, el estado del sistema o condiciones del contexto).
  - Relación de Generalización (inherits): es una asociación con sentido que consiste en que un CU particular hereda el comportamiento y el significado de otro CU general. El CU particular puede agregar comportamiento o reemplazar el comportamiento del CU general.

Ejemplo:



#### Especificación de los Casos de Uso:

1. Código y Nombre del Caso de Uso (**SI**)
2. Historial de Revisiones
3. Actores involucrados (**SI**)
4. Descripción (**SI**)
5. Disparador o Trigger (**SI**)
6. Precondiciones (**SI**)
7. Camino básico (**SI**)
8. Postcondiciones (**SI**)
9. Caminos alternativos (**SI**)
10. Excepciones (**SI**)
11. Casos de uso vinculados y puntos de extensión
12. Requisitos complementarios
13. Comentarios adicionales (**SI**)

Ejemplo:

1. CU 01 - Registrar Factura
2. Versión 1.0 - 01/08/2013 - Autor: Juan Pérez
3. Actores involucrados: AC: Administrativo Compras
4. Descripción: Utilizado para registrar Facturas de proveedores locales o del exterior
5. Disparador o Trigger: Ante la llegada de una nueva Factura de un Proveedor
6. Precondiciones:
  1. El proveedor debe estar previamente ingresado en el sistema

## **7. Camino básico**

1. El AC ingresa a la opción “Registrar Factura”
2. El Sistema muestra un listado de proveedores habilitados
3. El AC selecciona un proveedor de mercado local
4. El AC ingresa los datos de cabecera de la factura (Ver “Datos Factura”)
5. Por cada artículo comprado:
  1. El AC ingresa los datos del artículo solicitado (Ver “Datos Artículo”)
6. Por cada concepto adicional de factura:
  1. El AC ingresa los datos del concepto adicional de factura (Ver “Datos Conceptos Adicionales”)
7. El AC selecciona opción de fin de operación
8. El Sistema calcula el total de impuestos de la factura mediante “CU 02 - Calcular Impuestos”
9. El Sistema calcula el importe total de la Factura y pide confirmación
10. El AC confirma el ingreso de la Factura

## **8. Postcondiciones**

1. Queda afectado el Stock Pendiente de Ingreso de la mercadería solicitada
2. La factura queda registrada en el sistema.

## **9. Caminos alternativos**

- 3.1. Proveedor del exterior
  - 3.1.1. El AC selecciona un proveedor de mercado exterior
  - 3.1.2. El AC ingresa los datos del despacho de aduana mediante “CU 03 - Registrar Despacho de Aduana”

## **10. Excepciones**

3. El proveedor no se encuentra registrado en el Sistema.
3. El proveedor se encuentra en estado “Inhabilitado”

## **10. Excepciones**

11. Casos de uso vinculados y puntos de extensión
  - Puntos de Inclusión: 8. “CU 02 - Calcular Impuestos”
  - Puntos de Extensión: 3.1.2. “CU 03 - Registrar Despacho de Aduana”

## **12. Requisitos complementarios**

1. Rendimiento: La ejecución debe realizarse en un tiempo no mayor a 5 minutos

## **13. Comentarios adicionales**

1. “Datos Factura”
  1. Letra y Número
  2. Fecha de Emisión
  3. Observaciones (opcional)
2. “Datos Artículos”
  1. Código
  2. Cantidad
  3. Precio
  4. % IVA

Un Escenario de un Caso de Uso representa un camino completo que podría tomar una instancia del CU, desde el comienzo hasta el fin del mismo. Comienza por el primer paso del Camino Básico. No presenta condiciones o caminos alternativos, ya que refleja un único camino concreto. Pueden ser casos exitosos, o casos de excepción. Sirven para definir casos de prueba

de nivel usuario. El “camino básico” de un caso de uso suele corresponder con el escenario de mayor simplicidad.

Otros modelos de casos de Uso:

- Diagrama de Paquetes: los CU pueden agruparse en paquetes (por ej. para estructurarlos lógicamente en módulos del sistema).
- Diagrama de Clases: derivación de clases a partir de Especificación CU. La instanciación de un CU comienza a partir de la llamada a un método de una clase.
- Diagrama de Objetos: representación del estado de los objetos utilizados en un escenario, en un momento dado de su ejecución (por ejemplo, uno para representar sus pre-condiciones y otro para sus post-condiciones).
- Diagramas de Secuencia/Colaboración: representa la secuenciación de mensajes entre los objetos que colaboran para resolver un CU.
  - o Secuencia: Un diagrama por escenario.
  - o Colaboración: Un diagrama para todos los escenarios.
- Diagrama de Actividad: puede utilizarse para representar la secuencia de acciones de un CU (flujo principal, alternativos, excepciones).
- Diagrama de Estados: puede utilizarse para representar los cambios de estado de un objeto durante la ejecución de un CU.

Por último, los Casos de Uso que determinan el alcance de las funcionalidades de un determinado sistema se encuentran divididos en varias historias. Estas User Stories son una descripción breve de la captura de un requisito del sistema, es decir, pequeñas funcionalidades que son parte del alcance total del sistema.

Sus características son:

- a. Independientes: empiezan y terminan en sí mismas.
- b. Negociables con el usuario.
- c. Valiosas: poseen un resultado de valor.
- d. Estimable: predecible.
- e. Pequeña: son una pequeña porción del alcance.
- f. Testeable: deben de tener acciones y resultados claros.

En un principio se realizan las distintas User Stories para cumplir con el camino básico más simple (y por lo tanto más importante) y después darle soporte a los demás (más complejos).

Dentro de cada historia existe un ciclo de vida de cascada:



Scoped: definir el alcance.

Recapitulando...

... En el DFD importa quien nos porta la información, no quien la carga en el sistema. Mientras que en el Caso de Uso el actor es quien interactúa con el sistema