Metodología de resolución de problemas (M.R.P.)

Problema

Es un desvió negativo sobre lo planeado (logro de objetivo), en caso de ser positivo sería un punto de análisis.

- Es un desvió entre el resultado esperado y el obtenido
- Genera un impacto negativo
- Debe poder ser resuelto

Análisis de las causas y el estudio de su comportamiento

1. Lineal: Cuando el efecto de la causa puede ser causa de otro efecto
2. Bifurcada: Cuando una causa tiene más de un efecto
3. Red: Cuando un efecto tiene más de una causa

Pasos de la metodología de resolución de problemas

1. Determinar la causas
2. Codificarlas en causas internas (controlables) o causas externas (no controlables) a la organización.
    - Las causas externas no son consideradas para la resolución del problema ya que no se pueden controlar
3. Establecer un cuadro de relación
4. Asignar pesos a cada causa según la importancia y la capacidad de solucionar el problema
    - Se ataca cada causa según él % asignado, priorizando las de mayor porcentaje.
5. Buscar posibles soluciones

- La capacitación puede causar efectos no esperados: posibilidad de que los empleados abandonen la empresa

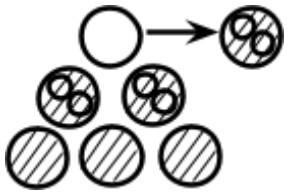
Aspectos de la metodología de resolución de sistemas

1. Objetivo
2. Alcance (desde/hasta)
3. Resultado: materialización del logro del objetivo.
4. Cosas estables: elementos de carácter permanente que influyen en el objetivo
5. Alimentación: elementos de carácter variable que intervienen en la solución del problema (ej: conocimiento, experiencia).
6. Procedimiento: descripción narrativa detallada paso por paso de lo que debe realizarse para alcanzar el objetivo o solución del problema

Política informática: Tomar una serie de decisiones relacionadas (ej.: plataforma tecnológica y metodológica).

Plataforma metodológica: Enfoque sistémico

Conceptos básicos



1. Ir de lo general a lo particular
2. Análisis por niveles progresivos (usado generalmente en análisis)
3. División del sistema en subsistemas (usado generalmente en diseño)
4. Método de prueba y error
    - 4.1. A veces es rápido
    - 4.2. Se usa generalmente en la implementación/testing.
    - 4.3. Desventajas:

        Dos entornos paralelos, no se dividen los entornos de producción y desarrollo:

        - ❖ Desarrollo: se modifica el código y cuando se prueba pasa al de producción.
        - ❖ Producción: no se modifica la documentación del análisis de sistema.

Entorno de desarrollo: Se testea el sistema y una vez que está todo bien se pasa al entorno de producción.

Aspectos a evaluar

1. Documentación: A través de los informes, se documenta mientras se avanza en las etapas. Facilita un posterior mantenimiento y es un indicador de calidad.
    - o En el tradicional debía documentarse al final y luego se puede proceder
    - o En el estructurado se documenta a medida que se va realizando (gráficos).
2. Eficiencia en cuanto a recursos (HW-SW)
3. Mantenibilidad: menor cantidad de errores con la mayor flexibilidad posible, debe ser lo más ágil que esté a nuestro alcance. Pertenece a la visibilidad externa ya que debe tener permeabilidad a los cambios que el usuario exija. Contempla el mantenimiento correctivo (soluciona errores en el sistema), el perfectivo (son mejoras, sobre algo funcional, que se realizan en el sistema, y el adaptativo (mantiene la eficacia actual ante cambios del entorno).
4. Portabilidad: Tiene en cuenta las distintas plataformas (S.O.), la aplicación puede llevarse a diferentes entornos tecnológicos. En el tradicional se aplica distinto en cada plataforma, esto trae mayores costos en desarrollo y mantenimiento, además de que se debe hacer un mayor control. Se soluciona en el estructurado y en objetos se optimiza. Está relacionado con la visibilidad interna.
5. Productividad: mayor cantidad de código en menor tiempo, lo que permite reducir tiempos y costos.

Metodología

1. Marco de referencia: define los limites
2. Define un "lenguaje": establece como se traduce el sistema.
3. Establece un estándar: indica cómo proceder dentro de los límites.

Es conjunto de métodos o técnicas formales que se siguen para lograr un objetivo de la manera más óptima. Es una manera sistemática de hacer las cosas que se fundamenta en la ciencia y utiliza determinados procedimientos a seguir. Es un método de investigación.

Tipo de enfoques

1. Top-Down: refiere a la idea de ir desde lo general hacia lo particular, desde las funciones más abarcativas a los subprocedimientos. Defino al sistema como una caja negra, sin mayor profundidad, y luego lo voy detallando cada vez más.

2. Bottom – Up: refiere a la idea de ir desde lo particular a lo general. Es usado por el enfoque orientado a objetos (y conceptualmente por el tradicional).Se busca una visión general, global. Se definen las entidad y cómo interactúan con detalle. A medida que avanzó en el análisis, las voy enlazando con otras para formar componentes más grandes, que a su vez también se enlazan, hasta llegar a formar el sistema completo.

Comparación de metodologías

Aspecto/Metodología	Tradicional	Estructurado	Orien. a Objetos
Documentación	-	+	+
Eficiencia	+	+	++
Mantenibilidad	-	+	++
Portabilidad	-	+	+
Productividad	-	+	++

- Modelizar un concepto, simplificar la realidad.
- Abstracción: proceso mental que simplifica la realidad y no considera limitaciones tecnológicas.
- Modelo lógico: es el "Que"
- Modelo físico: es el "Como"

Requisitos y requerimientos

Los requisitos y requerimientos son dos cosas distintas, por un lado:

1. Requisitos: "son funcionales", satisface necesidades del usuario.
2. Requerimientos: son "no funcionales", necesarios para que ande el sistema pero no están establecidos, contempla las necesidades internas del usuario (como backups, definir niveles de acceso, etc.).

Aspectos a mejorar

1. Mayor interacción con el usuario
2. Mejor captura de los requisitos
3. Posibilidad de “refinar” los requisitos
4. Mantener la “política de usuario”
5. Impacto de mejora en la calidad de Software (cómo evoluciona el sistema)
    - En relación a tiempos
    - Optimizar mantenimiento

Elección de la metodología

Depende de cuatro cosas:

1. Característica del problema
2. Herramientas disponibles
3. Recursos humanos
4. Presupuesto

Modelo tradicional

1. Cascada
2. No hay superposición de etapas
3. No se trabaja con modelos
4. Posee inconvenientes para documentar
5. Reciclar mas en menor tiempo
6. Esperar resultados: genera impaciencia del usuario
7. Tiene limitaciones operativas
8. Tiene problemas de costos
    - Todas las etapas tienen la misma prioridad

Modelo Estructurado

1. Componentes
    1.1. Datos + procesos
2. Énfasis
    2.1. Procesos
    2.2. Refinar RQ (Sistemas – usuario)
3. Características
    3.1. Emplea el Top-Down
    3.2. Favorece la documentación
    3.3. Hay superposición de etapas
    3.4. Se reducen tiempos y costos en relación con el modelo tradicional
    3.5. El usuario se impacienta menos
    3.6. Se implementa el concepto de modelo físico y modelo lógico
4. Herramientas
    4.1. DFD (Diagrama de flujo de datos)
    4.2. DD (Diccionario de datos)
    4.3. DER (Diagrama de Entidad / Relación)

El DFD responde al modelado de funciones mientras que el DD y el DER al modelado de datos.

- Comparado con el tradicional baja el mantenimiento y la implementación y sube el análisis y el diseño, el relevamiento queda igual. (El grafico también es válido para Orientado a Objetos)

Modelo lógico y físico

Modelo: es una simplificación de la realidad en donde se proporcionan los planos de un sistema.

- Un buen modelo incluye los elementos claves y omite los menos relevantes.

Objetivos: Construimos modelos para comprender mejor el sistema que estamos desarrollando:

1. Nos ayudan a visualizar como es o queremos que sea un sistema.
2. Nos permiten especificar la estructura o el comportamiento de un sistema.
3. Proporcionan plantillas que nos guían en la construcción del sistema
4. Documentan las decisiones que se toman.

Modelo Lógico: se piensa una solución lógica para el sistema, es decir, no tener en cuenta como se construirá, sino que debo concentrarme en determinar los eventos que ocurren y en los datos requeridos y producidos por cada evento.

Modelo Físico: se piensa una solución física para el sistema, es decir, como se va a construir el sistema, incluyendo la tecnología que se va a utilizar, los archivos y el personal. Es una descripción física del sistema.

Análisis estructurado
Modelos – Enfoque Top-Down

Análisis estructurado

Ventajas

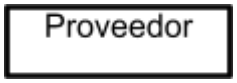
1. Mejora la interacción con el usuario y con el equipo de proyecto
2. Provee herramientas de documentación
3. Permite refinar el modelo lógico al mismo tiempo que se lo documenta
4. Utiliza niveles progresivos de análisis, de diferente profundidad
5. Se abstrae de los aspectos físicos de la implementación

Diagrama de contexto (DC o DFD Nivel 0)

1. Representa la abstracción de más alto nivel del Sistema
2. Se considera al Sistema como un único proceso, sin importar su estructura interna
3. Define las entidades externas con las que el Sistema interactúa, y sus entradas y salidas respectivas
4. Hay una gran participación del usuario que baja a medida que se particulariza el sistema.
5. Sirve para modelar funciones, requiere abstracción (permite abstraerse de los proceso internos de datos, así como los modelos físicos de los mismos (como llegan los datos)).

Simbología

1. Entidad Externa



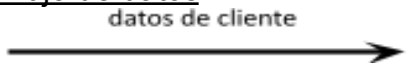
1.1. Representa el entorno del sistema
1.2. Es el origen y/o destinatario lógico de la información del sistema
1.3. Las entidades externas no se relacionan entre si

- Ejemplos de una entidad externa:
  - ✔ Una organización externa. Ej: Proveedor
  - ✔ Un área de la organización. Ej: Compras
  - ✔ Un rol de la organización. Ej: Jefe deposito
  - ✔ Otro Sistema. Ej: Sistema de liquidación de sueldos

2. Sistema

2.1. Representa la funcionalidad completa del sistema

3. Flujo de datos



3.1. Representa una relación de datos lógicos entre el entorno y el sistema con abstracción del medio físico, y con sentido unidireccional

Ejemplo

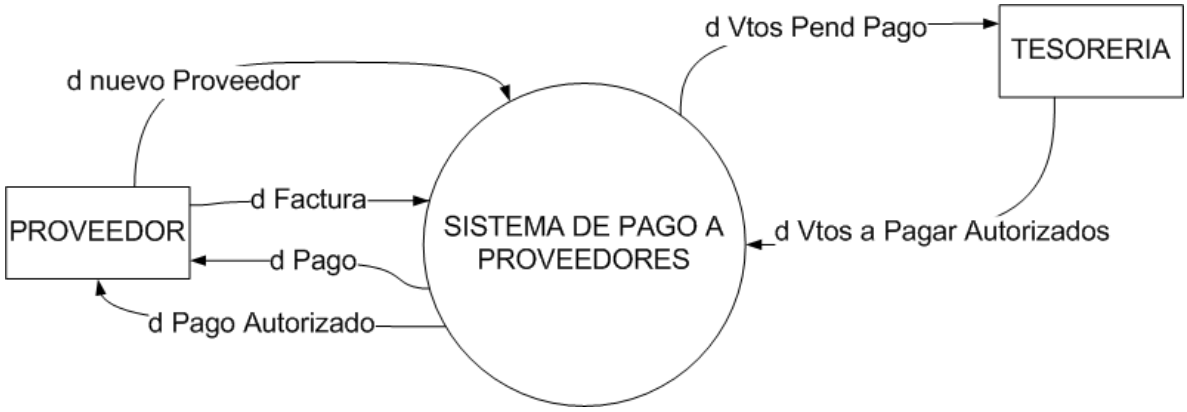


Diagrama de Flujo de Datos (DFD)

1. Representa la descripción grafica de los subsistemas lógicos que componen el sistema principal, y sus movimientos y almacenes lógicos de datos
2. En el DFD de nivel 1 los procesos no se comunican entre sí directamente
3. En el DFD de nivel 2 se controlan los errores (excepciones).
4. Proceso: aquello que permite una transformación de datos en el sistema.

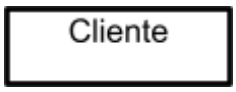
Lista de eventos: Concepto

1. Es un listado jerárquico que determina las funciones o procesos que se utilizaran en el DFD
2. Cada proceso se identifica con un Nº, el cual no implica el orden de ejecución

3. De un nivel al siguiente debe estar balanceado, mantener la entrada externa y la entrada/salida de datos

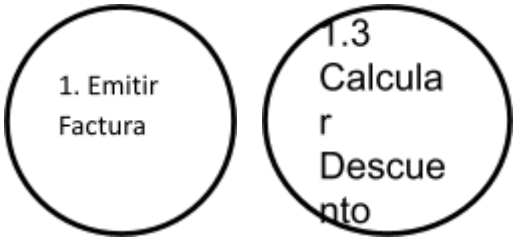
Simbología

1. Entidad Externa



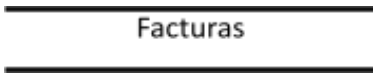
1.1. Deben ser las mismas que en el diagrama de contexto.
1.2. Define la frontera (entorno) del sistema.

2. Proceso o función



2.1. Es un subsistema que representa una función lógica interna del sistema
2.2. Se identifica con un código jerárquico definido en la lista de eventos, que indica el nivel del proceso
2.3. Representan un trabajo que realiza el sistema, provocando siempre un cambio o transformación en los datos
2.4. El nombre es una expresión que comienza con un verbo infinitivo y representa la actividad realizada por el proceso
2.5. Recibe uno o más flujos de datos de entrada
2.6. Genera uno o más flujos de datos de salida
2.7. Puede especificarse con:
- Lenguaje estructurado
- Tabla de decisión
- Árbol de decisión

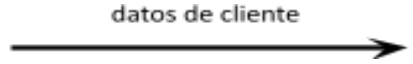
3. Almacenamiento o Demora



3.1. Representa un deposito lógico de datos en el sistema, donde los datos se almacenan temporalmente para ser utilizados en otro momento
3.2. Los flujos de datos de origen significan una consulta de los datos por un proceso
3.3. Los flujos de datos de destino significan la inserción o actualización de los datos por un proceso

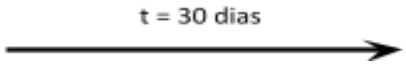
3.4. No representa ni especifica el tipo de almacenamiento físico (archivo, BD, etc.)
3.5. El nombre representa la estructura de datos que contiene, suele ser un sustantivo en plural
3.6. Son modelados en el DER y desarrollados en el DD

4. Flujo de datos



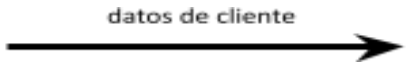
4.1. Representa una relación de datos lógicos, con abstracción del medio físico de la transferencia
4.2. El nombre representa la estructura de datos que contiene la información transferida. Usamos la expresión “datos de …” haciendo hincapié en la abstracción del soporte físico
4.3. No debería haber dos FD en el sistema con el mismo nombre (excepto FD entrante y saliente a un mismo almacenamiento)

5. Flujo temporal



5.1. Representa la activación periódica de un proceso con una frecuencia determinada
5.2. En el nombre se indica la frecuencia de activación

6. Flujo Activador

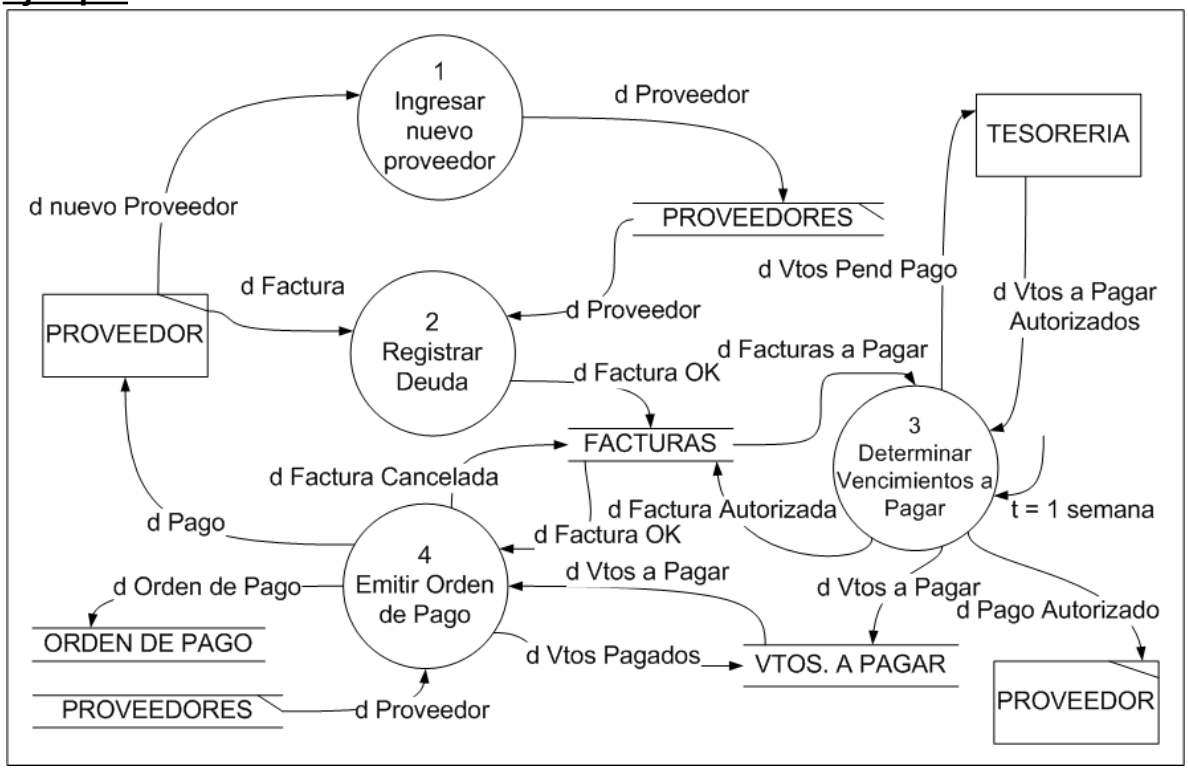


6.1. Es el flujo de datos que representa la “activación” de un proceso, provocando su ejecución
6.2. Puede ser un flujo de datos o un flujo temporal
6.3. El concepto también es aplicable al Diagrama de Contexto (flujo activador del Sistema)
6.4. Se escriben en plural

DFD Nivel 1: Listado de Eventos

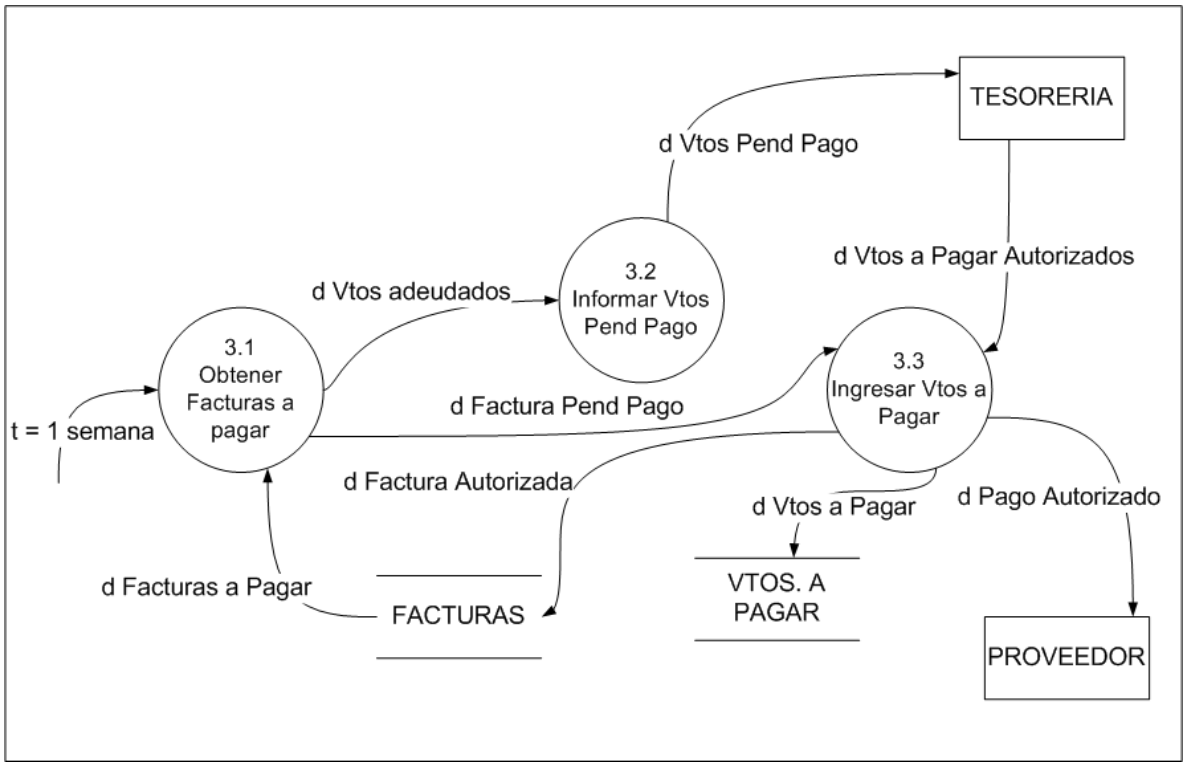
1. Gestionar proveedores
2. Registrar Deuda
3. Determinar vencimientos a Pagar
4. Emitir Orden de pago

Ejemplo

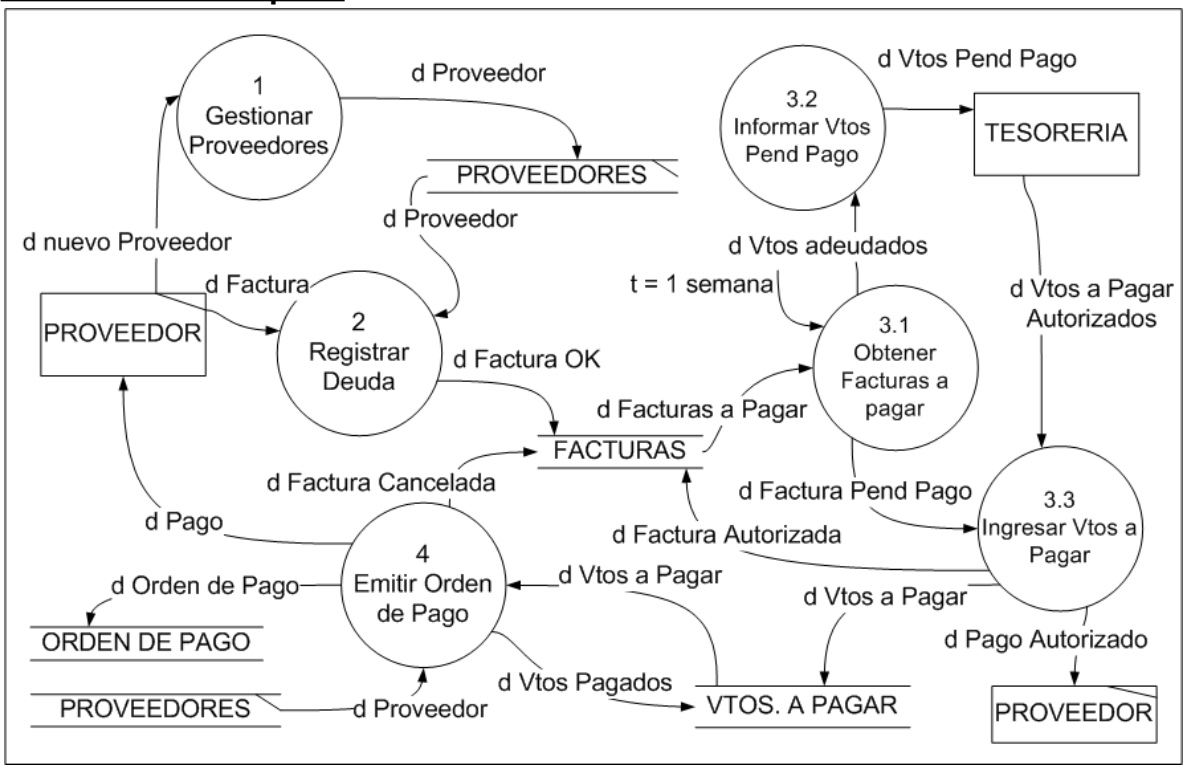


- "\" en la esquina superior derecha indica que es el mismo (si se repite entrada)

DFD Nivel 2: Proceso 3

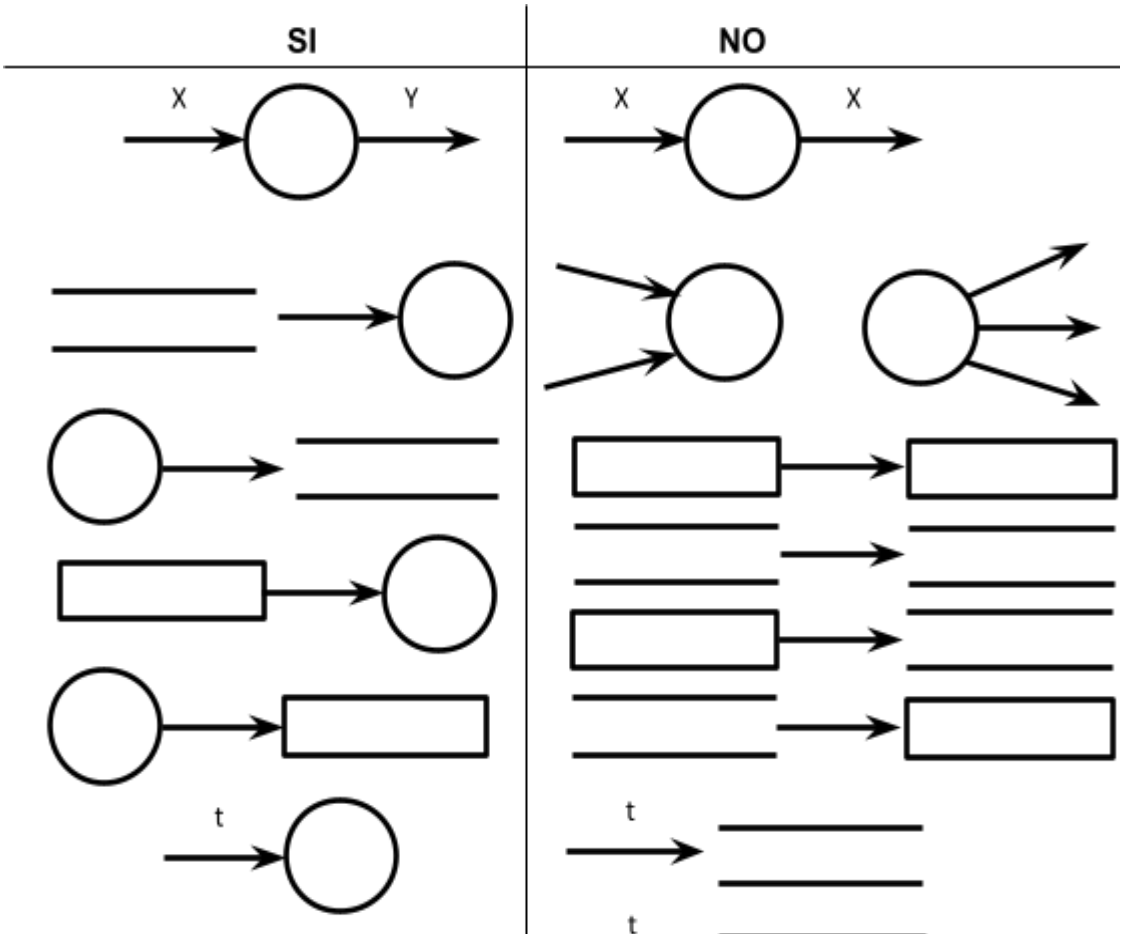


DFD Nivel 2: Completo



Observaciones sobre graficacion

1. Para que un dato vaya de una entrada a otra, debe pasar por los procesos (sino es ajeno al sistema y no se grafica)
2. Un dato no puede pasar de un proceso a otro



Observaciones Generales

1. Alcance funcional: es lo que puede hacer el sistema y puede apreciarse en el DFD1
2. Primitiva: Es el DFD de máximo nivel de detalle
3. El DFD es útil tanto para procesos informatizados como no informatizados
4. El nivel de un DFD es el de la burbuja más explotada (mayor subíndice)
5. El usuario participa hasta el DFD1
6. 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”
7. El usuario participa generalmente hasta el nivel 1 o 2 para validar requisitos
8. El detalle técnico comienza a participar a partir de los niveles 2 o 3, donde los requisitos funcionales ya están definidos y validados por el usuario.
9. El tratamiento de errores en general se comienza a evaluar del nivel 2 en adelante (resultado negativo de los controles)
10. Las Estructura de Datos (flujos y almacenamientos) se especifican en el DD (Diccionario de Datos)
11. El modelo de datos almacenado se especifica en el DER (Diagrama Entidad-Relación)

Diccionario de Datos

Concepto

1. Es un listado ordenado alfabéticamente que describe todos los datos que administra el sistema (balanceo) y valida el DC:

1.1. Flujos de Datos (Excepto Flujos Temporales)
1.2. Almacenamientos
2. Es un modelo de datos
3. Es una herramienta de Documentación y Comunicación
4. Elimina redundancias
5. Permite validar DFD
6. No es objetivo del DD la interacción con el usuario
7. Elemento de dato: Dato atómico (Ejemplo: fecha, importe).
8. Estructura de dato: Conjunto de datos relacionados, puede estar compuesto por estructuras de datos y/o elementos de datos (Ejemplo: d. factura).

Simbología

Simbología	Significado	Observaciones
=	Está compuesto de	
+	Y	
[|]	Alternativa	Se separan con PIPE (**|**)
( )	Opcional	
n{}m	Iteración	n y m: cotas para requisitos funcionales
@	Identificador	Permite extraer inequívocamente una estructura de datos dentro de un conjunto de estructuras.
* *	Comentario	Uso: *Comentario acerca de la estructura de datos
“ ”	Valor literal	Ejemplo: [“SI” | “NO”]

Ejemplo

- d_Factura_OK = @CODIGO_PROVEEDOR + @LETRA + @NUMERO + {FECHA_VTO + IMPORTE}
  + [PENDIENTE | AUTORIZADA | CANCELADA]
- 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

Diagrama Entidad Relación (DER)

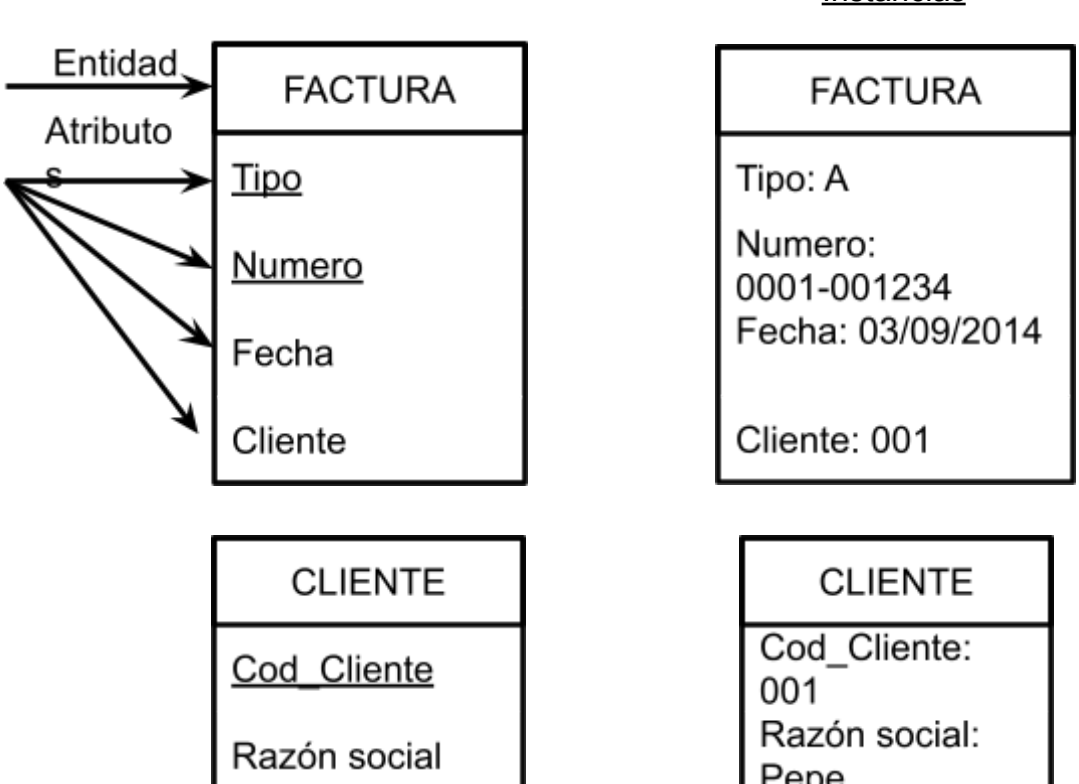
1. Representa mediante un modelo de red los datos almacenados o persistidos en el sistema, y sus relaciones
2. Modelo atado al modelo relacional de base de datos (modelo relación de resistencia)
3. Los elementos que lo componen son:
   3.1. Entidades: ideas de datos.
   3.2. Atributos: características de dichas ideas.
   3.3. Relaciones: entre entidades
   3.4. Clave: atributo que identifica una entidad

Entidades

1. Es una idea relevante del negocio que estoy modelando
2. Es un contexto determinado del cual deseamos guardar información para utilizarla en otro momento
3. Se designa con un sustantivo en singular. Ej: Factura, Cliente
4. Posee una clave que permite identificar de forma única una ocurrencia de la entidad
5. Está compuesta por Atributos y Relaciones

Atributos

1. Son características o propiedades asociadas a la entidad
2. Toman valor en una instancia particular
3. Son valores atómicos (similar al Elemento de Dato del DD)



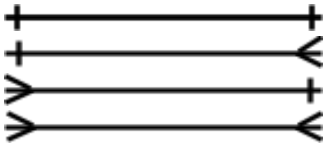
Relaciones

1. Indican cómo se relacionan las entidades
2. Se nombran con una letra R seguida de un subíndice “n” (R_n). 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:

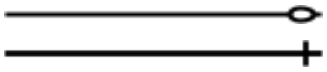
1 a 1:
1 a N:
N a 1:
N a N:



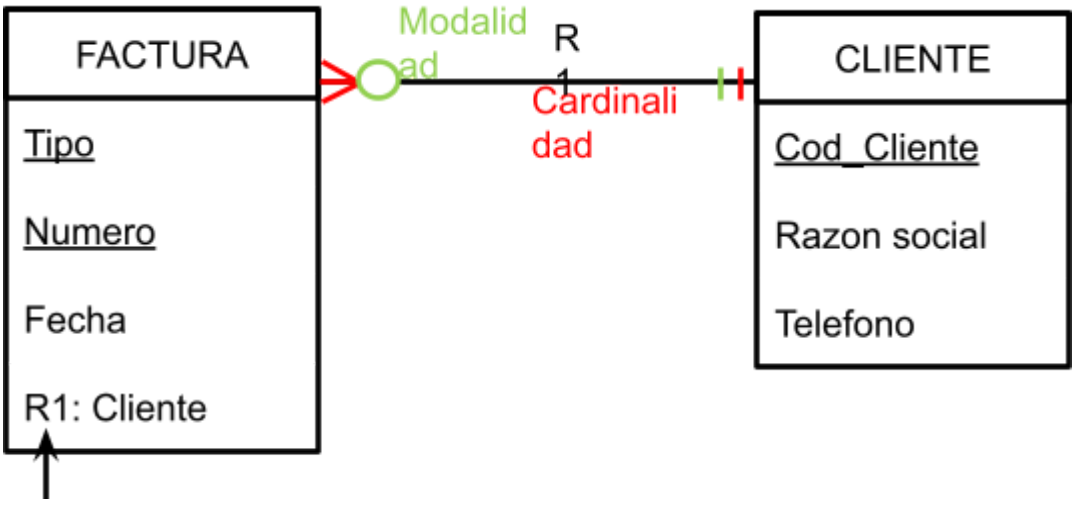
Modalidad

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

Opcional
Obligatorio



Ejemplo de relación



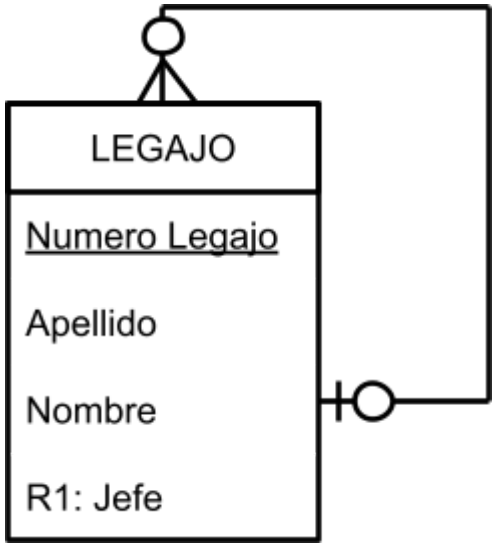
Atributo que indica la relación, siempre está en la entidad de cardinalidad “N” o “Muchos”. Se indica al lado del nombre de la relación, una descripción del significado lógico de la misma

Relación muchos a muchos

1. Es una idea lógica que no puede ser implementada físicamente
2. En el DER utilizamos una entidad asociativa para resolver este impedimento

Relación unaria

Es una relación de una entidad con otra instancia de sí misma



Clave

1. Superclave
   1.1. Conjunto de uno o más atributos (incluyendo atributos de relaciones) que permiten identificar de forma única una ocurrencia o instancia de una entidad (concepto de UNICIDAD)
2. Clave candidata o superclave mínima

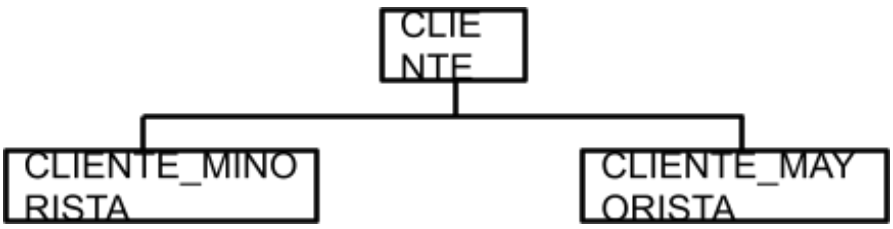
2.1. Aquellas superclaves para las cuales ningún subconjunto propio de atributos es a su vez una superclave (concepto de MINIMALIDAD)
2.2. Identifica inequívocamente una sola instancia por entidad
2.3. Cualesquiera de las claves candidatas puede ser elegida como clave principal

3. Clave principal (Primary key, PK)
   3.1. Es una de las claves candidatas que es elegida por el analista o diseñador para ser utilizada por el sistema, siguiendo su criterio de análisis para elección. Aquellas claves candidatas que no son elegidas como clave principal se denominan **clave alternativa**. Al ser clave candidata cumple con:
   - Unicidad
   - Minimalidad

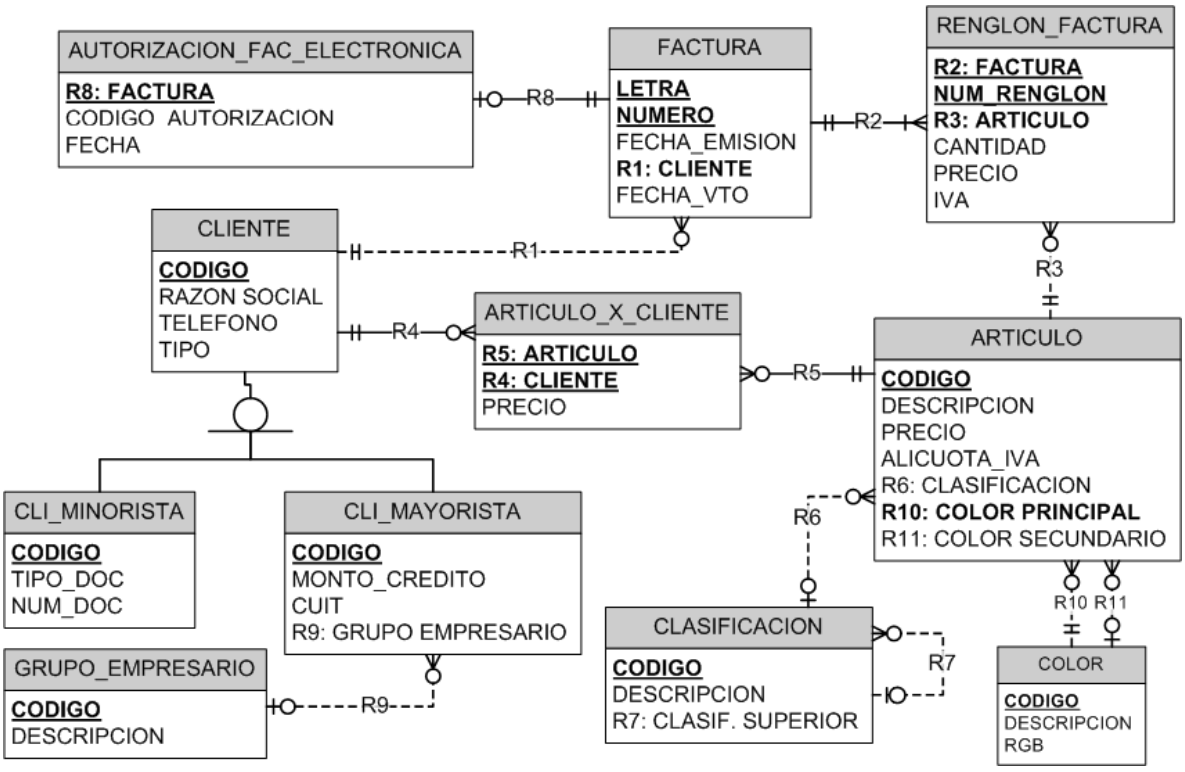
   Nota: en el diagrama los atributos que conforman la clave principal van subrayados para distinguirlos del resto
4. Clave secundaria o foranea (Foreign Key, FK)
   4.1. Es la clave de una entidad usada como atributo de otra entidad con la cual está relacionada. En el DER lógico se representa con el atributo de relación Rn. En el DER físico este es reemplazado por el atributo o conjunto de atributos que conforman la Clave Primaria de la entidad relacionada.

Clasificación de entidades

1. Fundamentales o fuertes
   Su clave está formada únicamente por atributos propios, si elimino las relaciones las entidades siguen existiendo.
2. Dependientes o débiles
   Su clave contiene al menos una clave foránea (relación)
   2.1. Asociativa: clave principal sin atributos propios
   2.2. Atributiva: clave principal con relación y atributos propios
3. Subtipo – Supertipo
   Tipos de entidades de una o más subcategorías, conectadas por una relación. La entidad Supertipo tiene un atributo que permite discernir de que Subtipo es una instancia determinada



Ejemplo completo



Balanceo con DFD y DD

1. Balanceo con DFD:
   - 1.1. Cada almacenamiento del DFD debe estar representado por una o más entidades del DER
   - 1.2. Cada entidad del DER debe estar representado por uno o más almacenamientos del DFD
2. Balanceo con DD:
   - 2.1. Cada definición en DD de una estructura de datos que corresponda a un almacenamiento debe corresponder a una o varias entidades del DER relacionadas, donde los elementos de datos del DD se visualicen como atributos del DER.

Normalización

1. Proceso mediante el cual se aplican una serie de reglas (Formas Normales) a las entidades y relaciones de un modelo con el objetivo de:
   - 1.1. Eliminar redundancias de datos
   - 1.2. Evitar problemas de actualización
   - 1.3. Proteger la integridad del modelo
2. Formas normales

2.1. No hay campos calculados. Toda entidad tiene clave y no pueden ser campos nulos. Todo campo no clave, depende de la misma.
2.2. Los campos no claves dependen completamente de la clave.
2.3. No existen dependencias transitivas entre atributos no clave.

Resumen

1. Refleja los datos, no lo que se hace con ellos
2. Se incluyen todos los datos almacenados del sistema en estudio
3. Es independiente de la tecnología de base de datos, pero ésta debe responder al modelo relacional
4. DER Lógico no tiene restricciones de espacio de almacenamiento ni performance
5. Su proceso de construcción es flexible

Modelo orientado a objetos

Origen del Análisis orientado a objetos

En 1994, G. Booch (Metodo Booch), J. Rumbaugh (OMT) y I. Jacobson (Proceso Objectory), deciden unificar sus métodos en el Unified Modeling Language (UML) y así surge el paradigma orientado a objetos, dicho proceso de estandarización fue promovido por OMG (organismo de estandarización) en 1997.
UML es una metodología de trabajo que permite usar modelos

Conceptos

1. Solapa el modelo lógico con el modelo físico. El solapamiento (o superposición) reduce tiempos, mejora la interacción con el usuario y refina requisitos.
2. Para facilitar el mantenimiento se utiliza el prototipado (“cascara del sistema”), esto mejora la política del usuario

Elementos

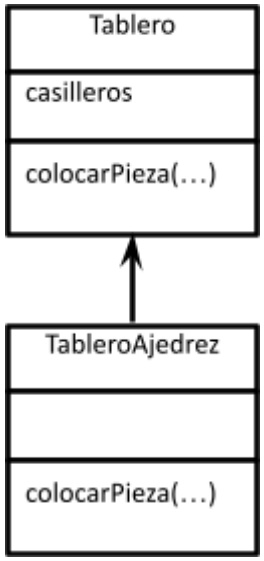
1. Componentes
   1.1. Datos + Procesos

2. Énfasis
   2.1. Datos (encapsulamiento)
   2.2. Refinar RQ (Sistemas – usuario)
3. Características
   3.1. Es Bottom-UP: pueden ser componentes nuevos, viejos que se reutilizan, o viejos que se reutilizan en parte. La reutilización es la diferencia con los otros paradigmas y tiene como ventajas principales la productividad y el mantenimiento.
   3.2. Robustece la documentación: esto se debe a que, a medida que desarrollo, todo queda automáticamente documentado. Tiene más módulos que el estructurado (14).
   3.3. Modularidad: permite separar el todo en partes (módulos) lo cual reduce tiempos pero requiere una buena coordinación.
   3.4. Polimorfismo: hace referencia a la posibilidad de que dos métodos implementen distintas acciones, aun teniendo el mismo nombre, dependiendo del objeto que lo ejecuta o los parámetros que recibe. Ayuda a la reutilización a nivel de procedimientos.
   3.5. Jerarquía: definición de conjuntos atachados a otros. Se refiere a la relación de herencia entre objetos o clases. Una superclase tiene mayor jerarquía que una subclase de ella.
   3.6. Herencia: el mecanismo mediante el cual se puede crear una nueva clase partiendo de una existente, donde la nueva clase hereda las características de la existente aunque pudiendo añadir o modificar las que tiene. Ayuda a la reutilización a nivel de datos y a reducir la redundancia.
   3.7. Orientado a sistemas específicos o complejos: en caso de que el sistema no sea específico o complejo se utiliza el modelo estructurado.
   3.8. Facilita el crecimiento de SW: disminuye el mantenimiento correctivo y facilita el adaptativo y el perfectivo.
   3.9. Optimiza la Captura y Validación de RQ:
      - Captura: entender que es lo que el usuario quiere
      - Validación de RQ: verifica si lo capturado es correcto
   3.10. Optimiza la Productividad: se mejora por la reutilización, mayor código en menor tiempo
   3.11. Modelos ajustados al Dominio (reglas de negocio):
      - Dominio: conocimiento específico y detallado del problema a mejorar.
      - Las reglas de negocio son los requisitos del sistema (“lo que necesita el usuario”).
      - Time to market: tiempo de respuesta rápida para satisfacer los nuevos requisitos que van apareciendo, y ganarle a la competencia.
   3.12. Uso de herramientas Case: facilitan el prototipado.
   3.13. Generación de Prototipos
   3.14. Catálogo de Objetos: objetos predefinidos, programados, listos para usarse (reutilización)

3.15. Ingeniería inversa: se parte del objeto y se llega a la fuente, gracias a la documentación. La ingeniería inversa tiene como gran ventaja poder completar documentación cuando no la tengo.
3.16. Módulos que permiten incrementar la potencia del paradigma
3.17. Costos asociados

Herencia simple

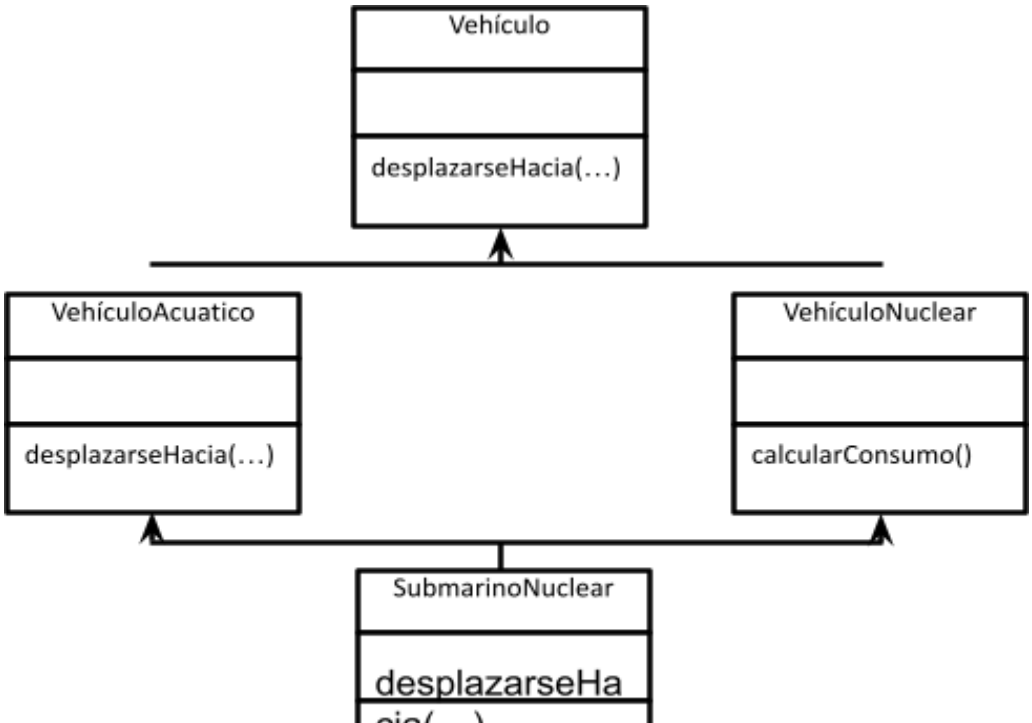
1. Es una forma de reutilización
2. Es un criterio para definir jerarquías



- Si un método se repite significa que el hijo lo modifica (parcial o totalmente).

Herencia múltiple

1. Es difícil de resolver
2. Es fundamental la potencia de la herramienta



Pasos a seguir en el enfoque orientado a objetos

1. Identificar clases y objetos, definir el lenguaje de identificación
2. Identificar semántica de clases y objetos (diccionario)
3. Identificar relaciones entre objetos
   2.1. Definir la frontera del sistema (alcance)
   2.2. Definir la arquitectura del sistema

Aspectos de elección entre estructurado y orientado a objetos

1. Si el sistema es simple se utiliza estructurado, si es complejo o. o.
2. experiencia persona.
3. Conocimiento (por parte del grupo de trabajo) de las metodologías y herramientas asociadas al paradigma
4. Disponibilidad de herramientas.

UML: Unified Modeling Language

Conceptos generales

1. Es un lenguaje estándar para la construcción, documentación, especificación y visualización de sistemas de software.
   1.1. Construir: Generación de código (herram. CASE) + Ingeniería Inversa
   1.2. Documentar: Producción de artefactos para entregables
   1.3. Especificar: Construir modelos precisos, no ambiguos y completos, desde análisis hasta implementación
   1.4. Visualizar: Proporciona un lenguaje común para los diferentes roles del equipo de proyecto. Permite trascender la dificultad de aspectos que sean de difícil apreciación al ser modelados en texto o código
2. Es un conjunto de modelos que respetan una simbológica común, permite unificar, lo que facilita la comprensión de proyectos.
3. Sus diagramas son aplicables a todo el ciclo de vida.
4. Es un complemento para las metodologías orientadas a objetos.
5. Sirve para todos los lenguajes, pero se explota con objetos.

6. Es Estándar: permite unicidad y estandarización a la hora de hacer las notaciones, en cualquier parte del mundo tiene la misma interpretación.
7. No prescriptivo: no proporciona un método de desarrollo. Es independiente del proceso.
8. Cada modelo enfatiza un determinado aspecto del sistema
9. Herramientas CASE
   9.1. UML -> Código
10. Ingeniería inversa
   10.1. Código -> UML

Simbología UML

Elementos UML

1. Estructurales
   1.1. Constituyen la parte “estática” del UML.
   1.2. Definen estructuras
   1.3. Son: clase, clase activa, interfaz, caso de uso, colaboración, componente, artefacto, nodo.
      1.3.1. Componente: Parte reemplazable de un sistema que conforma y proporciona la implementación de un conjunto de interfaces
2. De Comportamiento
   2.1. Constituyen la parte “dinámica” del UML.
   2.2. Relacionan a los elementos estructurales
   2.3. Son: interacción (mensaje), máquina de estados, estado, transición, evento, actividad, acción.

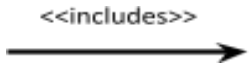
3. De Agrupación
   3.1. Constituyen la parte “organizativa” del UML.
   3.2. Agrupan elementos estructurales
   3.3. Es: Paquete.

4. De Anotación
   4.1. Constituyen la parte “explicativa” del UML.
   4.2. Aclaraciones extras
   4.3. Es: Nota.

Relaciones UML

Establecen vínculos entre distintos elementos

1. **Dependencia**: un cambio a un elemento puede afectar a otro
   1.1. Inclusión: dependencia obligatoria entre casos de uso



   1.2. Extensión: dependencia condicional entre casos de uso

2. **Asociación**: una clase posee una relación estructural con otra

2.1. Agregación: relación de tipo Todo-Parte, el objeto “Parte” forma parte de uno o varios objetos “Todo”.

2.2. Composición: asociación donde un objeto “parte” existe si y solo si forma parte de un objeto “todo”.

3. **Generalización**: La especificación del hijo se basa en la del padre, heredando su estructura y comportamiento

4. **Realización**: indica el cumplimiento de un contrato entre:
   4.1. Interfaz/Clase y componentes que las realizan.
   4.2. Caso de uso y colaboraciones que las realizan.

Otras definiciones

- **Estereotipo:** permite definir un nuevo tipo de elemento UML
  - Ej: <<includes>> (para relación de dependencia)
  - Ej: <<servidor_backup>> (para un nodo)
- **Valor etiquetado:** permite añadir propiedades a un estereotipo. Se colocan en notas asociadas al elemento afectado
  - Ej: Version = 3.1.00 (para un componente)
- **Restricción:** permite añadir nueva semántica o modificar las existentes
  - Ej: {Cant. máxima de instancias = 5} (para una clase)
  - Ej: {Cant. máxima de instancias = 5} (para una llamada remota)

Diagramas UML

1. Diagramas de estructura (“Aspectos Estáticos”): enfatiza los elementos estáticos del sistema. Los elementos estáticos son aquellos que componen el sistema y sus relaciones
   1.1. Paquetes
   1.2. Clases
   1.3. Objetos
2. Diagramas de comportamiento (“Aspectos Dinámicos”): resaltan los elementos dinámicos del sistema. Los elementos dinámicos componen el comportamiento del sistema.
   2.1. Actividad
   2.2. Casos de uso
   2.3. Estados
   2.4. Interacción
   - **Secuencia**

Diagrama de paquetes

Paquete: agrupación lógica de elementos UML que permite dividir y ordenar el modelo en diferentes contenedores

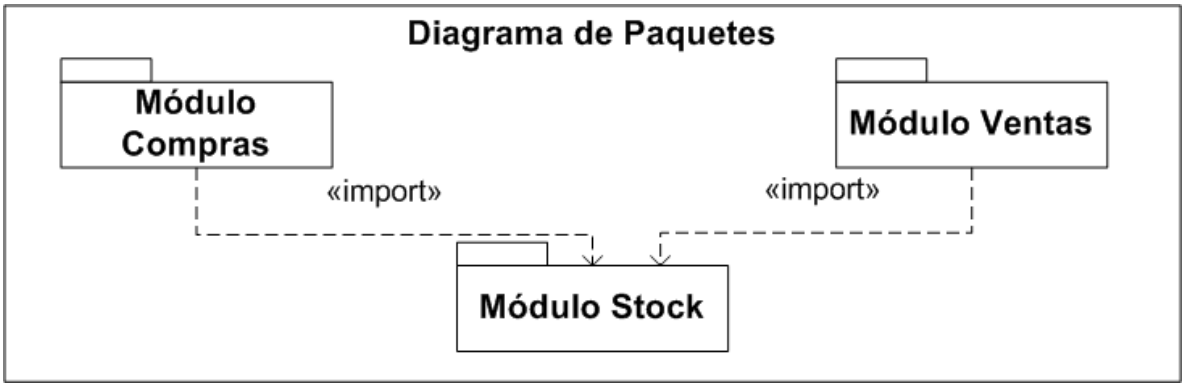


Diagrama de clases

Clase: abstracción lógica de un concepto que forma parte del vocabulario del problema. Posee un nombre que representa la especificación de un conjunto de objetos que comparten:

1. Atributos: Abstracción de los datos de los objetos. Caracterizan al objeto adquiriendo un valor determinado.
2. Operaciones: Abstracción del comportamiento del objeto. Ofrece un servicio a ser ejecutado por el objeto ante un mensaje.
3. Relaciones: Asociaciones estructurales con otros objetos
4. Semántica**:** "Responsabilidad" o "contrato": Expresión de cuál es la misión del objeto en el contexto del sistema.

Interfaz: definición de un conjunto de operaciones que especifican un servicio proporcionado o solicitado por una clase o componente.

Relaciones

1. Dependencia: relaciones de uso
2. Generalización (herencia de clases)
3. Asociación: relación estructural
4. Realización: determina que una clase implementa una interfaz. Conjunto de funcionalidades

Visibilidad: define el nivel de cada atributo/método para ser accedido por otro objeto

1. + public
2. # protected (público para los hijos)
3. - private
4. ~ package (público solo para el paquete que contiene a la clase)

Alcance: define si cada instancia tendrá su propio atributo/método, o si este es compartido por todas las instancias de la clase

1. De instancia
2. De clase

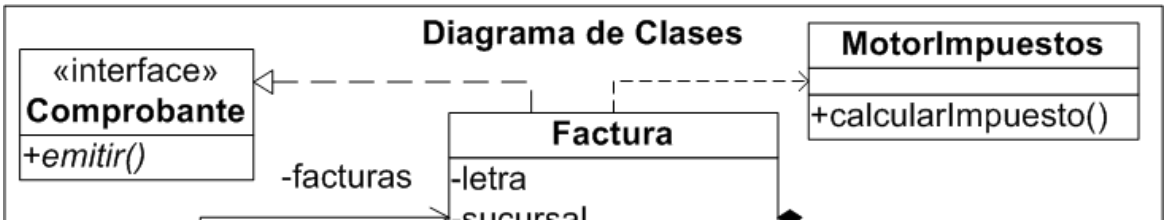


Diagrama de objetos

1. Objeto**:** “Instancia” o manifestación concreta de una clase
2. Estado: Almacena en sus atributos los datos propios del objeto
3. Su comportamiento se representa por los métodos definidos en la clase a la que pertenece
4. Otros conceptos:
   - 4.1. Mensajes: llamado a una operación
   - 4.2. Ocultamiento: los demás objetos no pueden ver lo que contiene
   - 4.3. Encapsulamiento: los otros objetos no pueden ver cómo trabaja
   - 4.4. Polimorfismo: capacidad de poder enviarle un mensaje a un objeto sin saber cómo es, permite generalizar, manda el mismo mensaje a objetos que trabajan diferente

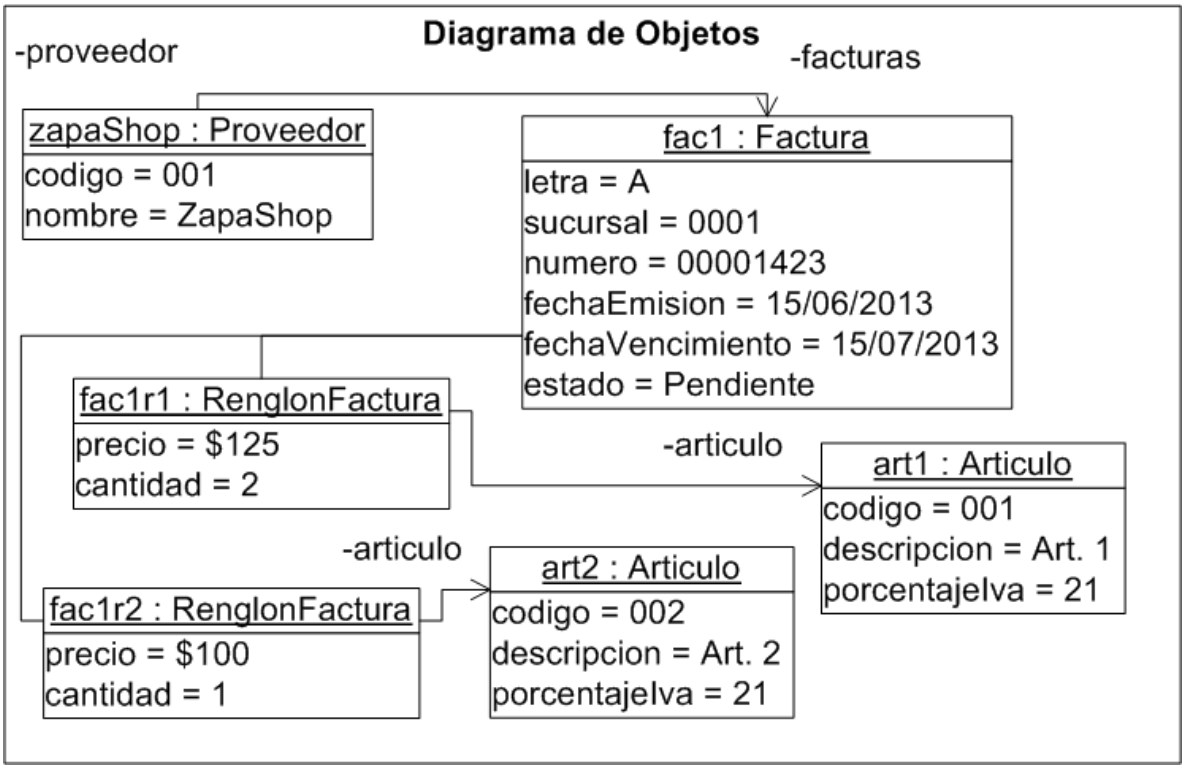


Diagrama de secuencia

1. Es un Diagrama de Interacción. Modela el aspecto dinámico de un sistema.
2. Describe la colaboración realizada entre distintos objetos y sus relaciones, mediante el intercambio de mensajes, para responder a la solicitud de una operación.
3. Se pueden utilizar para describir las colaboraciones necesarias para cumplir la realización de un Caso de Uso.

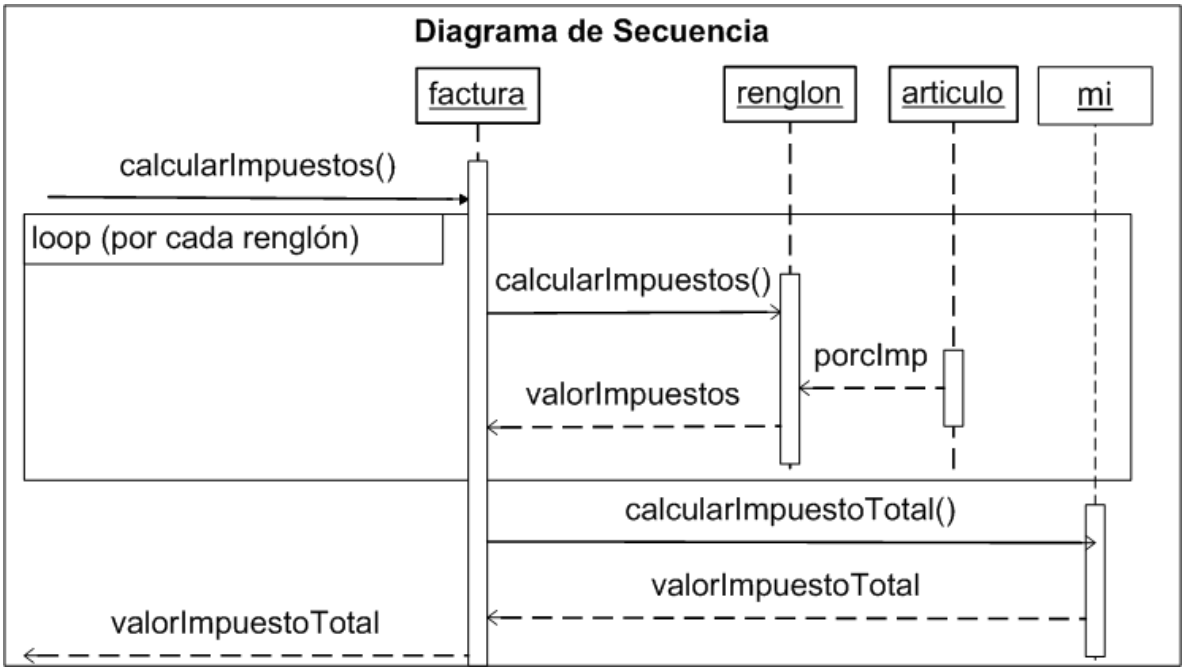


Diagrama de actividad

1. Describe el comportamiento de un elemento UML, por ejemplo:
   1.1. Clase
   1.2. Componente
   1.3. Caso de Uso
2. Permite modelar:
   2.1. Un flujo de trabajo: Procesos de negocio, desde un alto nivel de análisis. Puede incluir flujos de objetos.
   2.2. Una operación: Procesos computacionales iniciados por la llamada a un método de un objeto, ejecutados por uno o más objetos.
3. Símbolos

[ ] : Estado en el momento

: aparece elemento nuevo

: Operación

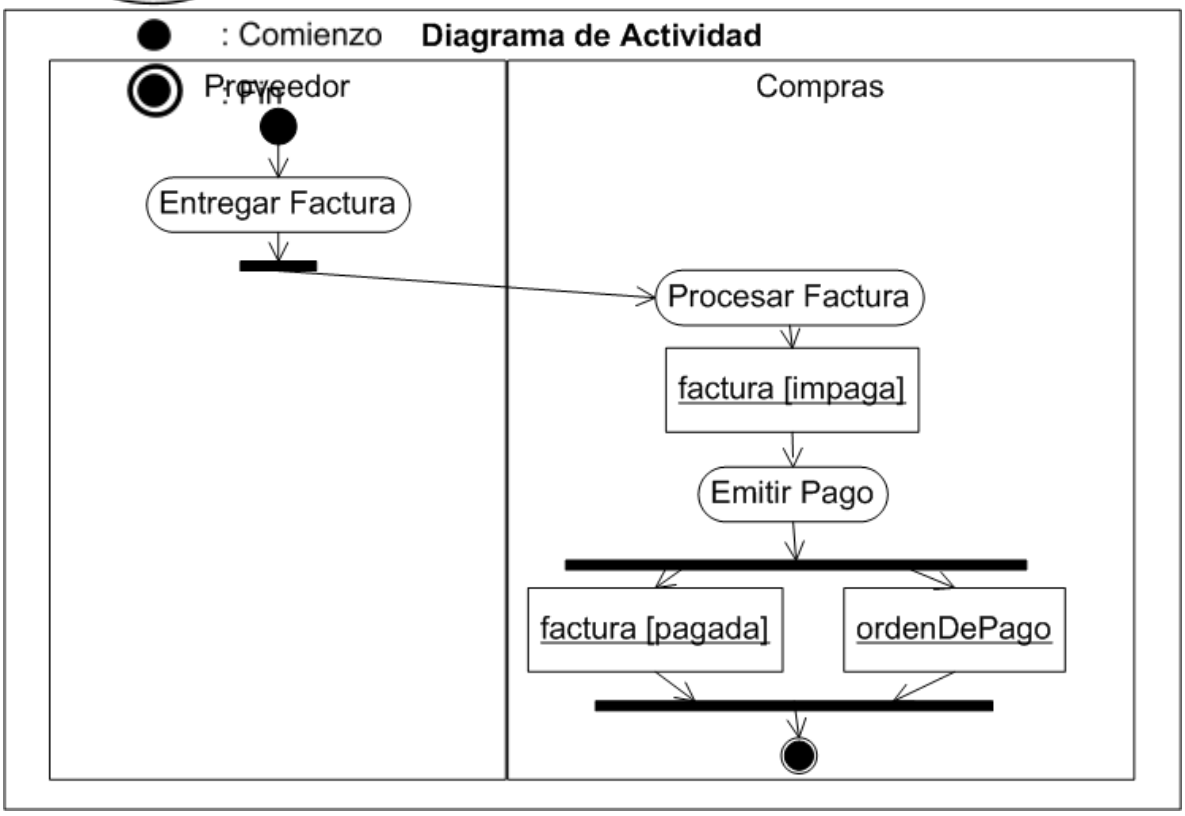
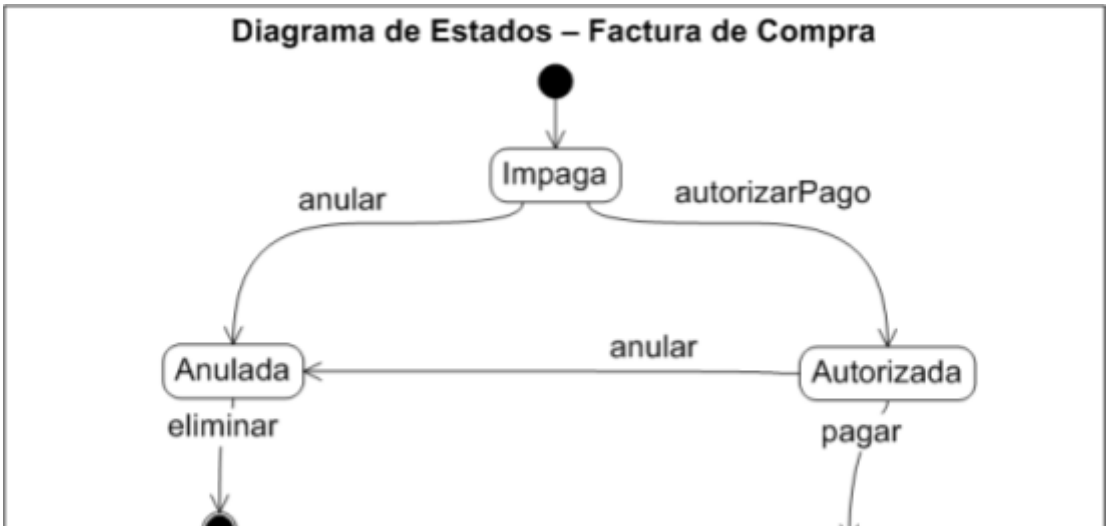


Diagrama de estados

1. Definiciones
   1.1. Estado: Situación en la vida de un objeto durante la cual se cumple una condición, realiza una actividad y/o espera un evento
   1.2. Evento: Acontecimiento significativo o estímulo que provoca una transición de estados
   1.3. Transición: Relación entre 2 estados que indica la acción por la cual un objeto pasará de un estado inicial a un estado final
   1.4. Acción: Ejecución atómica que, ante un evento, produce una transición entre dos estados de un objeto
2. Permite modelar una “Máquina de Estados”: Los posibles cambios de estado de un objeto durante su vida en el sistema.



Casos de usos

Concepto

1. Unidad atómica de comportamiento del sistema
2. Descripción de una secuencia de acciones, y sus variantes
3. Otorga resultado de valor para un Actor
4. Actor: Rol que se relaciona directamente con el sistema
5. Punto de vista del Actor
6. Iniciado por un Actor
7. “Dialoga” con uno o más Actores intercambiando datos para lograr un objetivo
8. Tipos
    8.1. Caso de Uso de Negocio
    8.2. Caso de Uso de Sistema

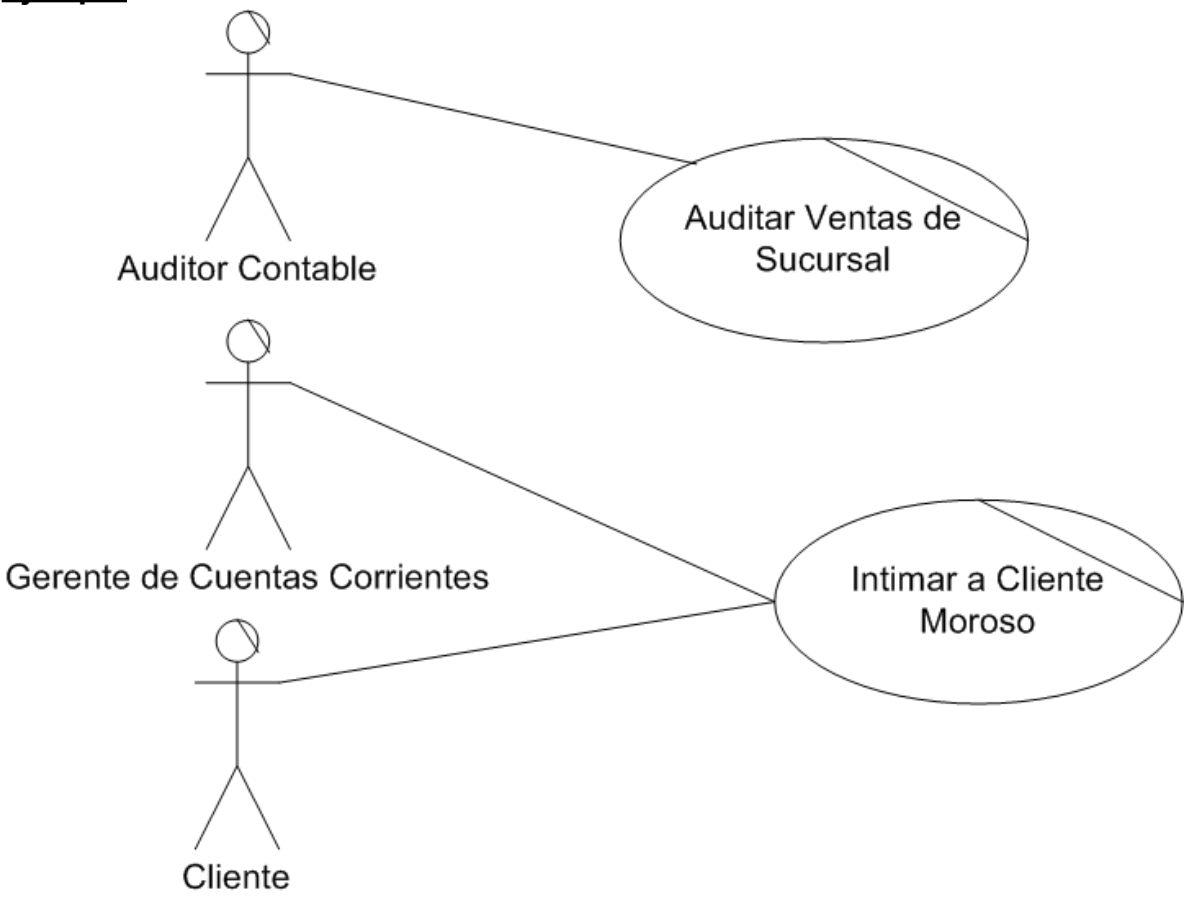
Utilidad

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

Casos de uso de negocio

1. 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.
2. Intervienen Actores de Negocio: Alguien que se relaciona con la organización e interviene en forma directa en el CUN. Ej: Cliente, Jefe de Ventas, AFIP
3. 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.
4. Se refieren al concepto de “Sistemas de Información”: Pueden (o no) estar informatizados por software, o estarlo en parte.

Ejemplo



- La línea en la cabeza de la persona refiere a que es un CU de negocio

Casos de uso de sistema

1. Técnica para capturar, documentar, comunicar y validar requerimientos de un "Sistema Software".
2. Describe el comportamiento esperado de un sistema, como una interacción entre éste y el usuario, para dar al usuario un resultado de valor.
3. Intervienen Actores de Sistema: Roles de interacción directa con el sistema: Un rol de usuario u otro sistema.
4. No especifica la forma en que quedará implementado. Aborda el "qué" y no el "cómo"
5. Define las interfaces entre el sistema Software y su entorno.
6. El conjunto de Casos de Uso permite definir el alcance funcional del sistema.
7. Punto de vista del usuario (terminología del dominio)
8. Procesos de desarrollo de Software "conducidos por Casos de Uso" (por ejemplo PU – Proceso Unificado)

9. Modelo de Casos de Uso: Se compone de:
   9.1. Diagrama de Casos de uso (UML): ilustra la relación entre actores y Casos de uso del Sistema
   9.2. Especificación de Casos de uso

Diagrama de Casos de Uso (UML 2.0)

1. Actor

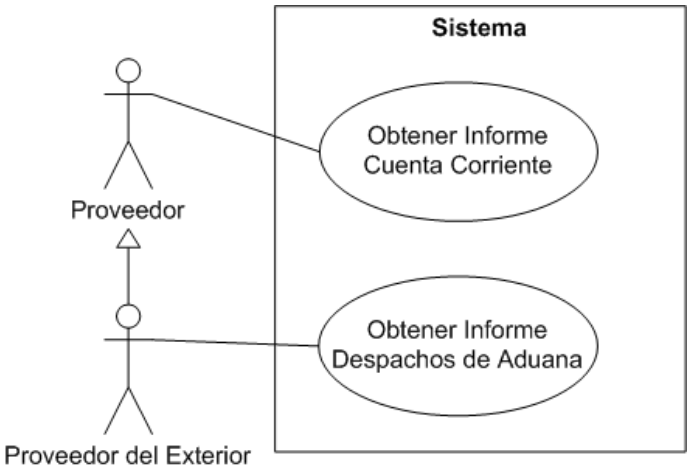
1.1. Representa un rol de interacción directa con el sistema. Puede ser un usuario u otro sistema.

1.2. Un usuario particular puede estar representado por distintos actores en diferentes momentos de su interacción con el sistema, es decir, toma distintos roles

1.3. Un actor es externo al sistema

2. Actor (Herencia)

2.1. Un actor particular puede heredar la definición de otro más general



3. **Caso de Uso**

3.1. Describe las acciones que el sistema ejecuta para proporcionar un resultado de valor para el actor vinculado

3.2. UML 2.0: Nombre con verbo en forma infinitiva.

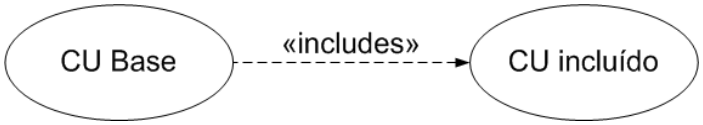
3.3. Posee una secuencia principal y puede tener secuencias alternativas

3.4. Una “instancia” de un CU es una secuencia particular de ejecución en un contexto determinado

4. Asociación

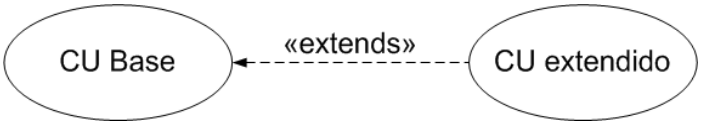
4.1. Establece un vínculo entre un actor y un caso de uso
4.2. El actor interviene directamente en las acciones definidas por el Caso de Uso
4.3. UML 2.0: La línea no posee orientación con flecha (en UML 1.0 sí tenía)
4.4. También denominada “Comunicación” por algunos autores

5. Relación de inclusión



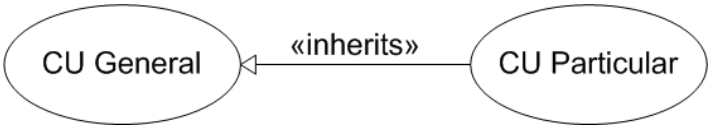
5.1. El CU **base** incorpora el comportamiento de otro (CU **incluído**)
5.2. Cada instanciación del CU **base** implica siempre la instanciación del CU **incluido**.
5.3. Objetivo: Reutilizar el comportamiento del CU incluido desde varios CU base.
5.4. UML 2.0: Reemplaza al «uses» de UML 1.0

6. Relación de extensión



6.1. El CU **base** incorpora de manera condicional el comportamiento del CU **extendido**
6.2. Modela el comportamiento opcional del sistema, (controlado por decisiones del usuario, el estado del sistema o condiciones del contexto)

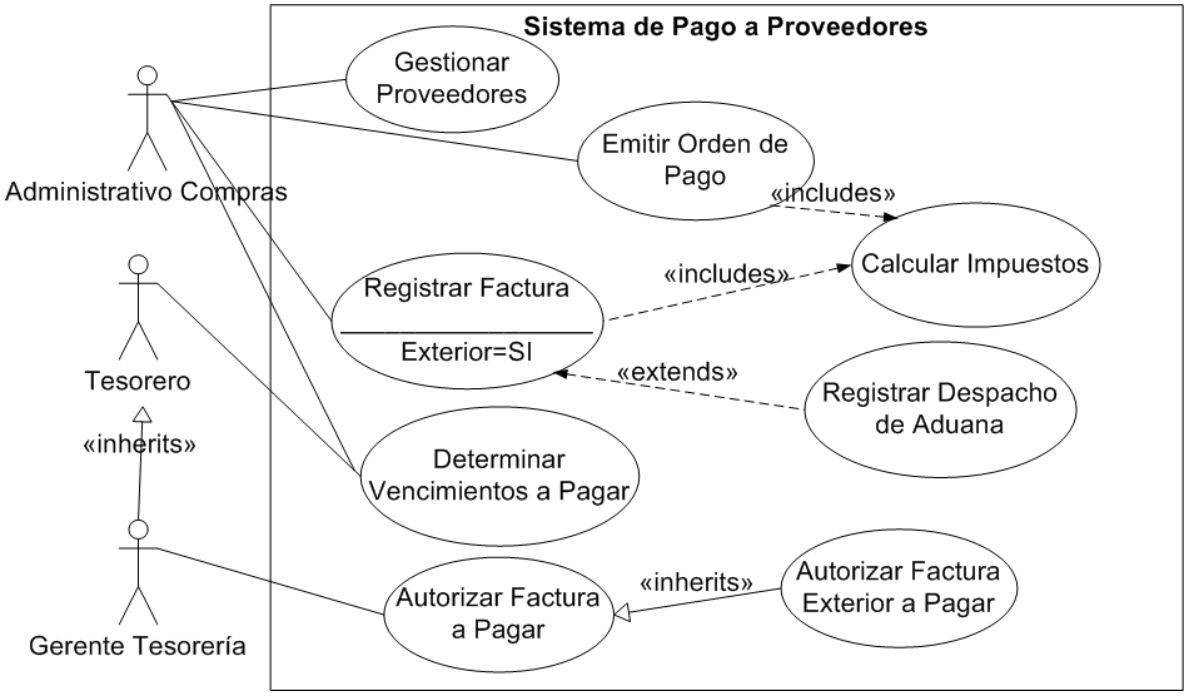
7. Relación de generalización



7.1. Un CU **particular** hereda el comportamiento y el significado de otro CU **general**.

7.2. El CU **particular** puede agregar comportamiento o reemplazar el comportamiento del CU **general**

Ejemplo



Especificación de Casos de Uso

Plantilla de especificación

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

Ejemplo

1. Código y Nombre del Caso de Uso: CU 01 – Registrar Factura
2. Historial de Revisiones: 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
    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
    11. El Sistema afecta el stock pendiente de ingreso de la mercadería solicitada
    12. El Sistema registra la Factura. Si el importe es <$1000 queda en estado “Pendiente de Pago”. En caso contrario, queda en estado “Pendiente de Autorización”
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. El AC no confirma la operación

11. Casos de uso vinculados y puntos de extensión
    - Puntos de Inclusión: “CU 02 – Calcular Impuestos”
    - Puntos de Extensión: “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 Artículos”
        1. Código
        2. Cantidad
        3. Precio
        4. % IVA
    2. “Datos Conceptos Adicionales”
        1. Descripción
        2. Precio
        3. % IVA

Escenario

1. 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.
2. Comienza por el primer paso del Camino Básico. No presenta condiciones o caminos alternativos, ya que refleja un único camino concreto.
3. Pueden ser casos exitosos, o casos de excepción
4. Sirven para definir casos de prueba de nivel usuario
5. El “camino básico” de un caso de uso suele corresponder con el escenario de mayor simplicidad.

Otros conceptos

Acción: Representa un requisito funcional.
Efecto de lado: modifica el sistema.

User Story

1. Forma de las metodologías agiles de documentar requisitos
2. Es el quien necesita, qué y para que
    - Quien: rol del sistema (actor).
    - Que: requisito funcional, los analiza con casos de uso
    - Para que: relación con el negocio.