CAPÍTULO 12

METODOLOGÍAS DE ANÁLISIS Y DISEÑO

Las metodologías son propuestas teóricas para llegar al objetivo de desarrollo de sistemas que incluyen los artefactos (básicamente procesos y herramientas) para llegar al objetivo de desarrollar la aplicación.

Una metodología, para considerarse como tal, debe responder a una serie de principios dados y articular sus elementos en forma lógica, conformando un sistema de relaciones organizado según un cierto orden.

Como objetivos últimos de las metodologías podemos mencionar:

- Describir cómo hacer técnicamente para obtener el producto (proceso).
- Servir como elemento de comunicación (herramientas de documentación).

Este último elemento es fundamental tanto para el éxito del proceso de desarrollo, como para la operación (mantenimiento) posterior.

El proceso de comunicación en el momento de desarrollo es de gran trascendencia para:

- Lograr que el analista interprete las necesidades, acordando la tarea a realizar (no significa "hacer todo lo que se solicite", ya que puede haber pedidos fuera del alcance del sistema en desarrollo, pero si que ambos tengan en claro qué se hace y no se hace, sin generar falsas expectativas).
- Lograr que los constructores interpreten las especificaciones, construyendo lo diseñado.

La comunicación hacia el futuro debe atender a que quienes se ocupen posteriormente de la utilización, operación y mantenimiento del sistema, tengan la información suficiente para realizar sus tareas.

El objetivo de comunicación toma mayor relevancia en la medida que intervengan más personas, tanto en la definición de requerimientos como en el desarrollo y la implantación del sistema.

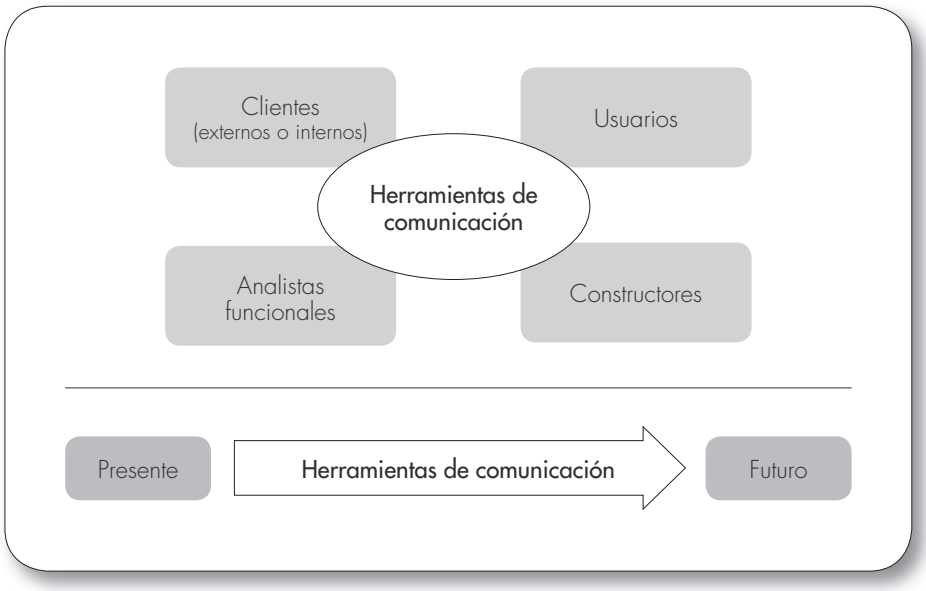


Figura 12.1
Documentación como elemento de comunicación

Prototipos como herramienta de comunicación

Los prototipos son representaciones del sistema en desarrollo para que el analista pueda representar su visión final a los ojos del usuario.

Según su profundidad pueden limitarse a la visión externa del sistema (imágenes de pantallas y listados) o simular su comportamiento incluyendo alguna funcionalidad de la aplicación, por ejemplo, simulando el ingreso de datos o incluyendo el encadenamiento de pantallas.

Al presentar el prototipo al usuario se completa el circulo de comunicación, validando los requerimientos y facilitando los ajustes que se detecten.

El uso de prototipos es una técnica que puede ser aplicada independientemente, tanto del ciclo de vida como de la metodología de desarrollo que se utilice, y puede extenderse a todo el sistema o sólo a las funciones identificadas como críticas.

Encontramos dos metodologías dominantes en la actualidad, la metodología de análisis y diseño estructurado, y la metodología de análisis y diseño orientado a objetos. A ellas se suman una serie de propuestas que toman en gran medida componentes de aquéllas.

12.1 METODOLOGÍAS DE ANÁLISIS Y DISEÑO ESTRUCTURADO

Estas metodologías[95] proponen la construcción de un "modelo lógico" del sistema mediante la descomposición gradual de los requerimientos del negocio, para llegar a funciones elementales, sobre las cuales se detallan las especificaciones para programación y se realizan los ajustes necesarios para la implementación física.

Utilizan pocas y simples herramientas para descomponer el sistema de manera tal, que facilitan tanto su comprensión por parte de usuarios y desarrolladores como su construcción y mantenimiento.

[95] Podemos mencionar, entre otras, las siguientes variaciones metodológicas menores en los siguientes trabajos: Larry Constantine y Eduard Yourdon, *Structured Design*, Yourdon Press, 1975; Tom DeMarco, *Structured Analysis and System Specification*, Prentice Hall, 1979; Chris Gane y Trish Sarson, *Structured Systems Analysis: Tools and Techniques*, Mcdonnell Douglas Information, 1977; y, últimamente, con actualización Web, Eduard Yourdon, *Just Enough Structured Analysis*, Yourdon Press, 2006.

El análisis se realiza desde dos visiones complementarias, la de los datos y la de los procesos.

- Los procesos, específicos para el sistema tratado, interactúan con los datos.
- Los datos se encuentran disponibles tanto para estos procesos, como para otros que los requieran, en un entorno de integración de múltiples aplicaciones.

Este enfoque permite, con una adecuada gestión de datos, facilitar la reutilización de datos, siendo este un objetivo perseguido para lograr una integración eficaz y efectiva de las aplicaciones.

12.1.1 Visión desde los procesos

La visión desde los procesos, basadas en los conceptos propuestos por Stevens, Myers y Constantine en 1974[96], se realiza partiendo de un diagrama general representativo del sistema y avanzando en su descomposición, desde lo general a lo particular, utilizando como herramienta el diagrama de flujo de datos (DFD).

El DFD describe el sistema como una red de "procesos" conectados, mediante "flujos de datos", entre ellos mismas, con agentes externos (usuarios u otras aplicaciones) y con almacenamientos de información.

Debido al enfoque de descomposición gradual encontramos diferentes niveles de DFD, donde:

- La graficación de más alto nivel se denomina "diagrama de contexto" o DFD de nivel 0, y se limita a exponer la interacción entre el sistema y los agentes externos que actúan como fuentes y destinos de los datos. Muestra todo el sistema como un proceso único.
- Este DFD de contexto se "explota" en el nivel 1, donde se desagregan los principales procesos del sistema modelado y su relación con los almacenamientos de información internos del sistema y los agentes externos señalados en el nivel 0.
- Sucesivamente cada proceso se "explota" en el nivel siguiente, respetando la relación con agentes externos y almacenes de información, y agregando los almacenes internos de ese nivel.
- Luego de llegar al último nivel de descomposición, el comportamieno del proceso se detalla para su codificación fuera del DFD, utilizando principalmente los siguientes artefactos específicos:
  - Lenguaje estructurado
  - Tablas de decisión
  - Árboles de decisión

12.1.2 Visión desde los datos

La visión desde los datos llega a la formulación de la estructura lógica de datos (en la cual los datos requeridos por el sistema son agrupados en entidades) requerida para soportar los procesos del sistema, utilizando la técnica de normalización y graficando el resultante en el diagrama de entidad relación, detallado en el Capítulo 10.

La técnica de normalización parte por la identificación de todos los elementos de la base, analiza las relaciones subyacentes entre ellos y permite determinar la mejor forma de organizar los datos en tablas, en función de esas relaciones. Se basa en el estudio profundo de las relaciones subyacentes entre los elementos, a la luz de los requisitos del sistema objeto y la aplicación de principios de álgebra de relaciones.

[96] "Structured design" en *IBM Systems Journal*, 1974.

La normalización fue formalizada por E. F. Codd en *A relational Model of Data for large shared data banks*, 1970.

Los estudios de Codd demostraron que todas las relaciones entre datos pueden resumirse a relaciones simples entre tablas de dos dimensiones (filas y columnas), y que la estructura así determinada es la más simple que puede establecerse para representar adecuadamente el sistema objeto, dando mayor facilidad tanto para responder preguntas, que pueden resolverse con los elementos ya contenidos, como para responder nuevos requisitos, que necesiten el agregado de elementos adicionales.

Las entidades así creadas se denominan relacionales; y esta teoría es la base sobre la cual se desarrollan los sistemas administradores o de gestión de bases de datos relacionales. Sin embargo, no es necesario contar con una base de datos relacional para reconocer la estructura lógica de datos, que puede implementarse tanto con bases relacionales como con bases jerárquicas o archivos tradicionales.

El proceso de normalización garantiza que la estructura de datos así determinada es la que mejor representa la realidad subyacente a los datos requeridos por el sistema. Como consecuencia también asegura que tanto el mantenimiento posterior como las ampliaciones y la integración con otros sistemas, que seguramente requerirían el agregado de nuevos datos y/o nuevas tablas con nuevos datos, no va a desnaturalizar la estructura lógica definida.

La experiencia empírica demuestra que el desarrollo de aplicaciones informáticas, sin considerar un diseño de datos encuadrado en la estructura lógica subyacente, provoca, ante el agregado de nuevas funciones o cambios en las existentes, una mayor necesidad de modificaciones en las aplicaciones con los consiguientes costos y tiempos adicionales en el mantenimiento y el crecimiento de los sistemas, debido al aumento en la complejidad de las interrelaciones entre los elementos nuevos y los existentes.

Por lo tanto, la estructura lógica de datos debería ser considerada como la base sobre la cual no sólo se construirá una aplicación en particular, sino también la base sobre la cual se asentarán las modificaciones a esa aplicación, y los futuros desarrollos e integraciones con otras aplicaciones.

Como comentamos ambas visiones son complementarias. Los almacenamientos de datos referidos en el DFD corresponden a las entidades de información utilizadas en el DER, y los datos contenidos en las entidades de información del DER deben ser utilizados en los procesos del DFD.

Luego de consensuado el diseño lógico, se realiza su implementación física. Para esta implementación se tienen en cuenta las restricciones tecnológicas. En particular la estructura lógica de datos puede recibir una gran cantidad de modificaciones, las que se llaman "desnormalización", para que sea físicamente implementable debido a restricciones impuestas por el nivel tecnológico disponible (básicamente capacidad de almacenamiento y tiempo de respuesta); sin embargo, ello no obsta que se busque definir adecuadamente la estructura lógica, como marco de referencia de cualquier tarea posterior.

Como herramientas adicionales podemos mencionar las siguientes:

- Diccionario de Datos (DD)
  - Repositorio integrado de todos los datos ingresados, producidos, administrados y entregados por el sistema.
- Diagrama de Transición de Estados (DTE)
  - Modelización del comportamiento.
  - Representa el comportamiento de un sistema exponiendo los eventos que producen que el sistema cambie de estado y destaca qué acciones se llevan a cabo como consecuencia de ese cambio.

En los anexos del Punto 12.4 incluimos ejemplos de DFD y una breve descripción del proceso de normalización.

12.2 METODOLOGÍAS ORIENTADAS A OBJETOS

A diferencia de los métodos estructurados, que separan datos de procesos, el enfoque de análisis y diseño orientado a objetos (ADDO) une datos y procesos en artefactos denominados "objetos".

Mientras que el enfoque tradicional (y el estructurado) se basa en el análisis de eventos y la determinación de su equivalente lógico, el enfoque OO requiere que esos eventos pertenezcan a un "objeto" identificable. Un objeto puede ser un lugar, una persona o una cosa relevante para el sistema, por ejemplo, un objeto puede ser un cliente, una factura, un empleado, un proveedor.

Eso supone un avance en cuanto a la reutilización e integración aplicativa con relación a los métodos estructurados, donde los datos son compartidos mientras que los procesos son específicos para cada aplicación.

Las actividades de desarrollo se centran en los objetos. El software se organiza a partir de los elementos que existen en el dominio del problema.

Como en el caso de la metodología estructurada, el análisis y diseño orientado tiene por objetivo la construcción de un modelo que interprete la complejidad subyacente en el sistema objeto y la determinación de su equivalente lógico, no la aplicación de herramientas de programación orientadas a objetos.

En tal sentido, Rumbaugh[97], uno de los principales referentes de las metodologías de ADDO, nos dice:

> "La esencia del desarrollo orientado a objetos es la identificación y organización de conceptos del dominio de la aplicación, y no de su representación final en un lenguaje de programación tanto si éste es orientado a objetos como si no lo es".

La implementación física posterior dependerá de los lenguajes y bases de datos utilizados.

En los términos de ADDO un objeto es todo conjunto cohesionado (adherido fuertemente, atraído internamente), integrado por dos componentes esenciales:

- Atributos (datos organizados).
- Servicios (referentes lógicos de los procesos de transformación, operaciones, los cuales reciben y entregan información al exterior del objeto por medio de parámetros).
- Métodos (forma en que se implementan los servicios; un mismo servicio puede implementarse con diferentes métodos, dependiendo de la tecnología que se utilice, siendo esto transparente para el usuario).

El armado conjunto de atributos, servicios y métodos se denomina "encapsulado".

El "encapsulado" provoca "ocultamiento de información", haciendo visibles y accesibles los datos sólo mediante los servicios implementados, así:

- Protege los datos del uso arbitrario.
- Oculta los detalles de la implantación interna a los usuarios de un objeto, por lo que los usuarios conocen los servicios que puede solicitar del objeto, pero desconocen los detalles de cómo se llevan a cabo.

[97] Rumbaugh y otros, *Modelado y diseño orientado a objetos*, Prentice Hall, 1996.

- Al separar el comportamiento del objeto de su implantación, permite la modificación de ésta sin que se tengan que modificar las aplicaciones que lo utilizan, en la medida que se mantengan los servicios.

El "objeto" factura puede tener como servicios, entre otros, los siguientes:

- Informar nombre del cliente destinatario de la factura.
- Informar importe total.

Un objeto contiene estructuras de datos y comportamientos que lo caracterizan.

Sólo se accede a él por los servicios establecidos.

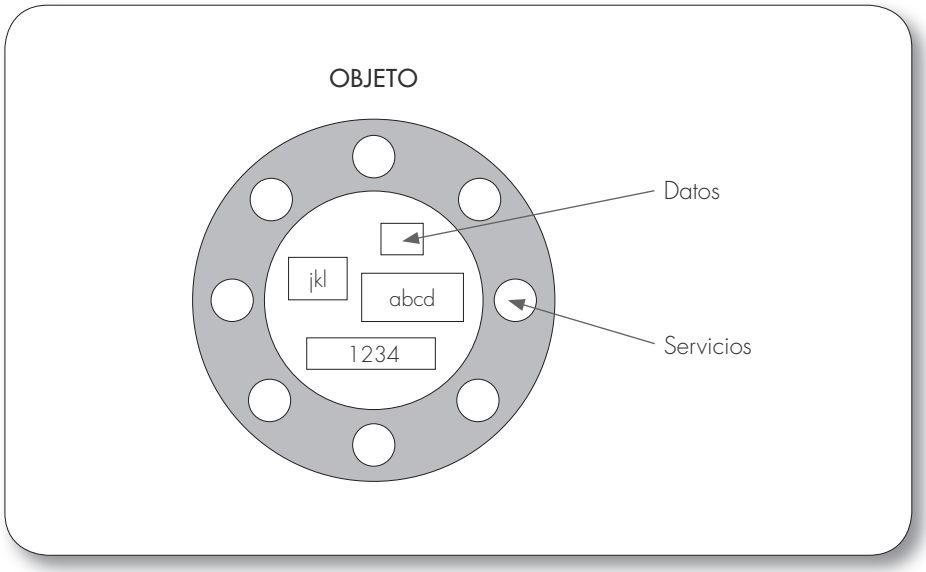


Figura 12.2
Encapsulamiiento de datos y servicios en el objeto

Los distintos objetos se comunican por "mensajes". Un mensaje solicita un servicio que ejecute el método[98] apropiado y, en su caso, realize una modificación de datos y/o produzca una respuesta. El mensaje que constituye contiene el nombre del objeto, el nombre del servicio y, según corresponada, un grupo de parámetros.

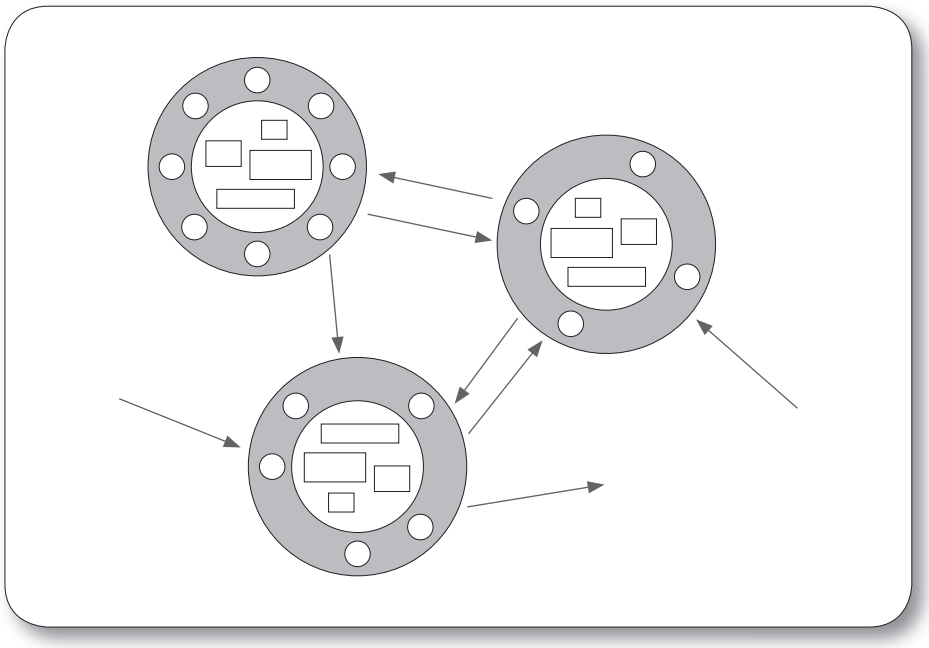
De esta forma se pueden "armar" aplicaciones nuevas combinando, mediante mensajes, objetos existentes, integrando en objetos y construyendo los objetos no existentes. Asimismo, un objeto puede estar compuesto por otros objetos, formando un objeto complejo.

Por ejemplo, en un sistema de un banco, el objeto que provee (tiene el servicio) de "dar saldo de cuenta" será el mismo si el saldo se consulta desde una posición de caja, desde un cajero automático o desde internet, así como también será el mismo si se lo solicita para determinar si se paga o no un cheque contra ese saldo.

[98] Un "servicio", una vez que se avanza en la construcción, se implementa en los lenguajes de programación como uno o varios "métodos".

Figura 12.3

Comunicación entre objetos



Los objetos tienen las siguientes características:

a) Clasificación

Una clase es un grupo de objetos que tiene atributos y comportamientos similares.

b) Identidad o instanciación

Objetos con iguales atributos y servicios son distinguibles entre si, debido a que tienen una característica distintiva de "identidad".

Por ejemplo, dentro del objeto "factura", el número de documento le da "identidad".

Por lo tanto, un objeto es una "instancia" única de una clase, que posee su propio valor para cada uno de los atributos, pero comparte los nombres de atributos y las operaciones del resto de la clase.

c) Jerarquía y herencia

Las clases se encuentran relacionadas jerárquicamente, y comparten atributos y servicios tomando como base esa relación jerárquica.

Una clase puede incluir sub-clases de nivel jerárquico inferior.

Esta es una característica fundamental y trascendente, ya que permite que conociendo el comportamiento de la "clase" se sabe que la subclase tiene el mismo comportamiento, más otros comportamientos adicionales específicos de ella.

d) Polimorfismo

Un mismo servicio puede comportarse de manera diferente en distintas instancias de una misma clase, por aplicación de un método diferente.

Así un usuario no necesita conocer el método aplicado para una operación o servicio, y, a la vez, se pueden agregar instancias nuevas a una clase en la medida en que el objeto de la instancia tenga el servicio con su método.

Durante el proceso de diseño se realiza la "segmentación", esto es la asignación de responsabilidades a una clase de objetos, para lo que se requiere. Así los requeri-

mientos son cumplidos por los objetos según su pertenencia lógica, y no en función de la forma en que fueron relevados o de su implementación física.

Por ende, se "centralizan" todas las funciones que corresponden al mismo objeto, es decir, utilizan los mismos datos y realizan la misma transformación, generando economías en del desarrollo y el mantenimiento al garantizar la reusabilidad.

Encontramos una profusión de herramientas utilizadas en metodologías orientadas a objetos, algunas de ellas superpuestas. Esto hace que los artefactos correspondientes a las metodologías orientadas a objetos resulten complejas y requieran una mayor formación para su interpretación que los artefactos de diseño estructurado.

Un esfuerzo de unificación de estas herramientas se realizó con la construcción del lenguaje unificado de modelado, concido como UML por sus siglas en inglés (*Unified Modeling Language*). Este conjunto de artefactos para modelado fue diseñado en su primera versión por un trabajo conjunto de los principales autores del enfoque de diseño orientado a objetos, James Rumbaugh, Grady Booch e Ivar Jacobson. El UML en su versión actual describe trece herramientas, siendo ellas:

i) Diagramas estructurales

1. Diagrama de clases
2. Diagrama de componentes
3. Diagrama de objetos
4. Diagrama de estructura compuesta
5. Diagrama de despliegue
6. Diagrama de paquetes

ii) Diagramas de comportamiento

7. Diagrama de actividades
8. Diagrama de casos de uso
9. Diagrama de estados

iii) Diagramas de Interacción (subtipo de diagramas de comportamiento)

10. Diagrama de secuencia
11. Diagrama de comunicación
12. Diagrama de tiempos
13. Diagrama global de interacciones

A continuación, haremos una breve descripción de los artefactos que, a nuestro juicio, resultan de mayor utilidad para la modelización del sistema objeto:

a) Estructurales

a.1) Diagrama de clases

- Muestran la relación entre clases de objetos.

b) De comportamiento e interacción

b.1) Diagrama de casos de uso

- Herramientas de comunicación muy simple y efectiva, que puede utilizarse con cualquier metodología.
- Modelan el diálogo entre un actor y el sistema describiendo la funcionalidad que ofrece el sistema al actor.
- El conjunto de casos de uso del sistema constituyen todas la formas de uso definidas en el sistema.

- Documentan el comportamiento del sistema desde el punto de vista externo, detallando:
  - funciones requeridas para el sistema
  - los actores
  - la interacción entre las funciones y los actores
- Se constituyen en el medio principal para viabilizar el diálogo entre usuario y desarrollador acerca de las funcionalidades del sistema y su comportamiento, para llegar a los acuerdos correspondientes en relación al producto a entregar.
- Un caso de uso describe en lenguaje natural, la forma en que un "actor" del mundo real (persona, organización o sistema externo) interacciona con el modelo.
- Comprende un diagrama y la explicación en castellano de la forma en que el "actor" (persona, organización o sistema externo) interactúa con el modelo.
- Este modelo va agregando nivel de detalle en la medida en que se avanza en la defínción. En sus comienzos puede referirse solamente al "curso normal de los eventos" (por ejemplo, en el caso de una venta con pago con tarjeta, no considerar las acciones en caso de rechazo de crédito) y luego avanzar en los "escenarios alternativos" a ese curso normal.

b.2) Diagrama de actividades

- Exponen las actividades de un caso de uso en la forma en que se van dando, incluyendo actividades paralelas y decisiones tomadas.
- Tiene una conformación similar al tradicional cursograma.

b.3) Diagrama de secuencias

- Muestran la relación entre las diferentes funciones detalladas en los casos de uso y los objetos y servicios que esas funciones requieren.

En los anexos del Punto 12.4 incluimos ejemplos de casos de uso, diagramas de actividades y de secuencias.

Uno de los principales problemas de esta metodología es la necesidad de una total y completa definición de clases al inicio de las actividades, a los efectos de asegurar que nuevas necesidades no requieran asignaciones de responsabilidades a objetos existentes respetando la herencia entre clases y subclases.

Esto, considerando la natural fragmentación que tiene todo proceso de incorporación de software, resulta imposible en la práctica, ya que hasta ahora no se ha desarrollado un proceso tal, que, de forma análoga al proceso de normalización para el caso de estructuras de datos, garantice una distribución adecuada de servicios y datos entre clases y subclases.

En los casos en que esas nuevas responsabilidades requieran la modificación de algún servicio que no respete la herencia (ascendente o descendente), siguiendo la teoría, se debe realizar una recomposición de servicios y clases, con costos que pueden ser significativos. Ante esta situación se plantean tres alternativas:

- Asumir el costo de reorganización de clases.
- Copiar y modificar clases. Armar una nueva y específica estructura de objetos y clases, con los nuevos requisitos de herencia, con lo que se producen duplicaciones innecesarias.
- Negar la herencia. No respetar la herencia en algún punto de la cadena de clases-subclases con lo que conocer la clase no implica conocer la subclase, desnaturalizando la metodología.

Un ejemplo simple servirá para conceptualizar esta situación:

Las "cajas de ahorro" y las "cuentas corrientes" tienen muchas características comunes, teniendo como principal diferencia que las segundas permiten el manejo de descubiertos.
Supongamos que estamos desarrollando el sistema bancario de cajas de ahorro.
Definimos la "clase" "caja de ahorro", e implementamos servicios que no permiten que tenga descubierto.
Si luego queremos desarrollar el sistema de "cuentas corrientes" encontraremos que la clase "caja de ahorros" implementa la gran mayoría de los servicios requeridos, sin embargo, tiene como característica la inexistencia de descubiertos.
Esto tiene significativas implicancias, sin ir más lejos el "saldo disponible para pagos" en el caso de cuenta corriente es diferente que en el de caja de ahorros.
Desde la metodología corresponde definir una clase "cuenta" que tenga como subclases "caja de ahorro" y "cuenta corriente", donde todos los servicios comunes se encuentren implementados en la clase y los pocos servicios específicos en las subclases "caja de ahorros" y "cuenta corriente". Esto puede implicar una serie significativa de cambios.
En el caso de "copiar y modificar clases" se copiaría la estructura completa introduciendo los cambios a nivel de clase, duplicando de esta manera el código a mantener.
En el caso de "negar la herencia" se modificaría un servicio según el tipo de cuenta, con lo que los servicios de la clase no se aplican a todas las subclases.

12.3 ALGUNAS CONSIDERACIONES SOBRE LA INCORPORACIÓN DE SISTEMAS DEL MERCADO (PAQUETES DE SOFTWARE)

Si bien la incorporación de sistemas del mercado, tal se describen en la parte dos, no presenta la complejidad de un desarrollo de todas formas, es necesario comparar explícitamente la diferencia entre los requerimientos y las funciones disponibles.

Si bien estos sistemas ofrecen una funcionalidad estándar flexibilizada en forma paramétrica –es decir, modificando ciertos valores en archivos del sistema que provocan que el mismo cambie su comportamiento– es habitual que no cumplan con todos los requerimientos funcionales y de integración con el resto de sistemas de la organización.

Si fuera necesario cumplir con requerimientos no cubiertos por el paquete ya parametrizado, nos encontraríamos ante la necesidad de modificar el producto, construir agregados por afuera del producto (por ejemplo, tomando los datos que el sistema utiliza y explotándolos con otras aplicaciones) o complementar el producto con tareas manuales.

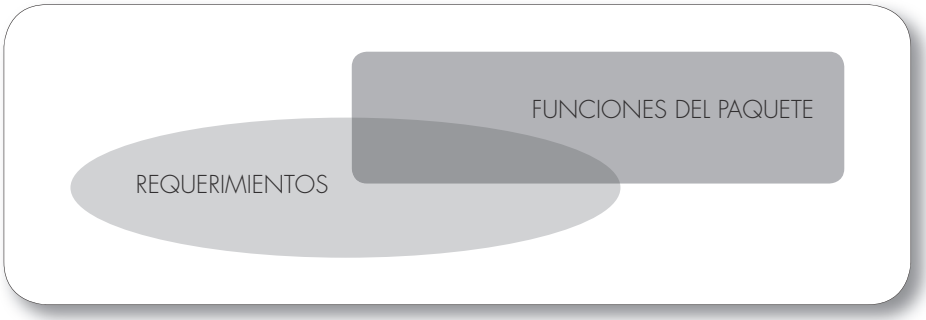


Figura 12.4
Diferencias en alcance deseado y alcance implementado en el paquete

Por lo tanto, los requerimientos no cubiertos por el paquete pueden dar lugar a:

- Incorporar el paquete, sin esos requerimientos.
- Realizar adecuaciones y desarrollos complementarios para cumplir con esos requerimientos.
- Alguna situación intermedia.

La detección y explicitación de la brecha entre requerimientos y disponibilidades debe realizarse en forma temprana, antes de la adquisición misma del paquete, ya que la cobertura de ella puede generar costos tales que, de haberlos conocido anteriormente, podrían haber cambiado la decisión tomada, tanto hacia la adquisición de otro paquete (inicialmente más caro, pero con menor necesidad de adaptación, resultando en un menor costo total) como, incluso, un desarrollo a medida.

Tanto las adecuaciones como la implementación del paquete en sí mismo, requiere un enfoque de ciclo de vida y metodológico.

12.4 ANEXOS

12.4.1 Ejemplos de diagrama de flujo de datos (DFD)

Figura 12.5
Componentes de un DFD

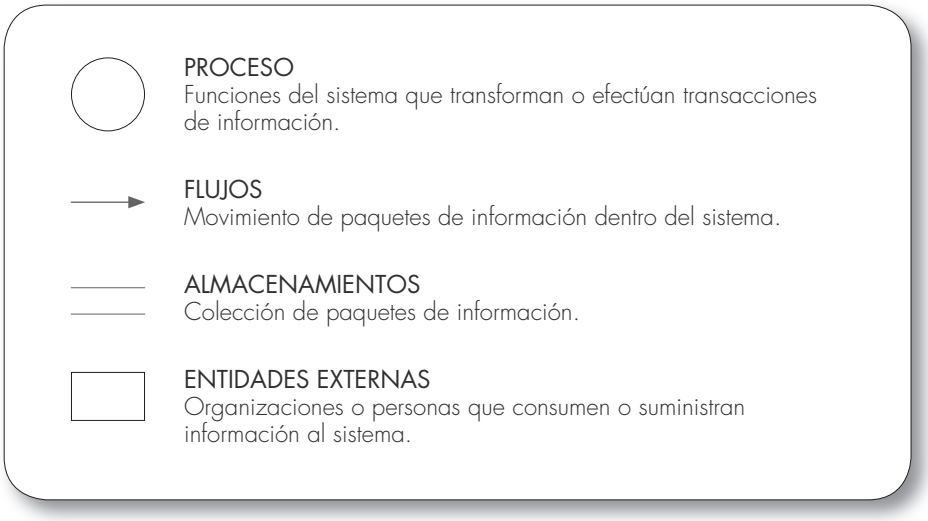
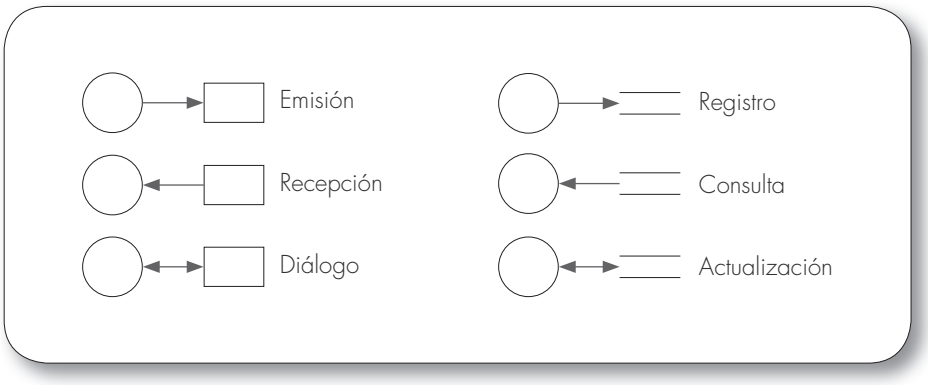


Figura 12.6
Interacción entre componentes de un DFD



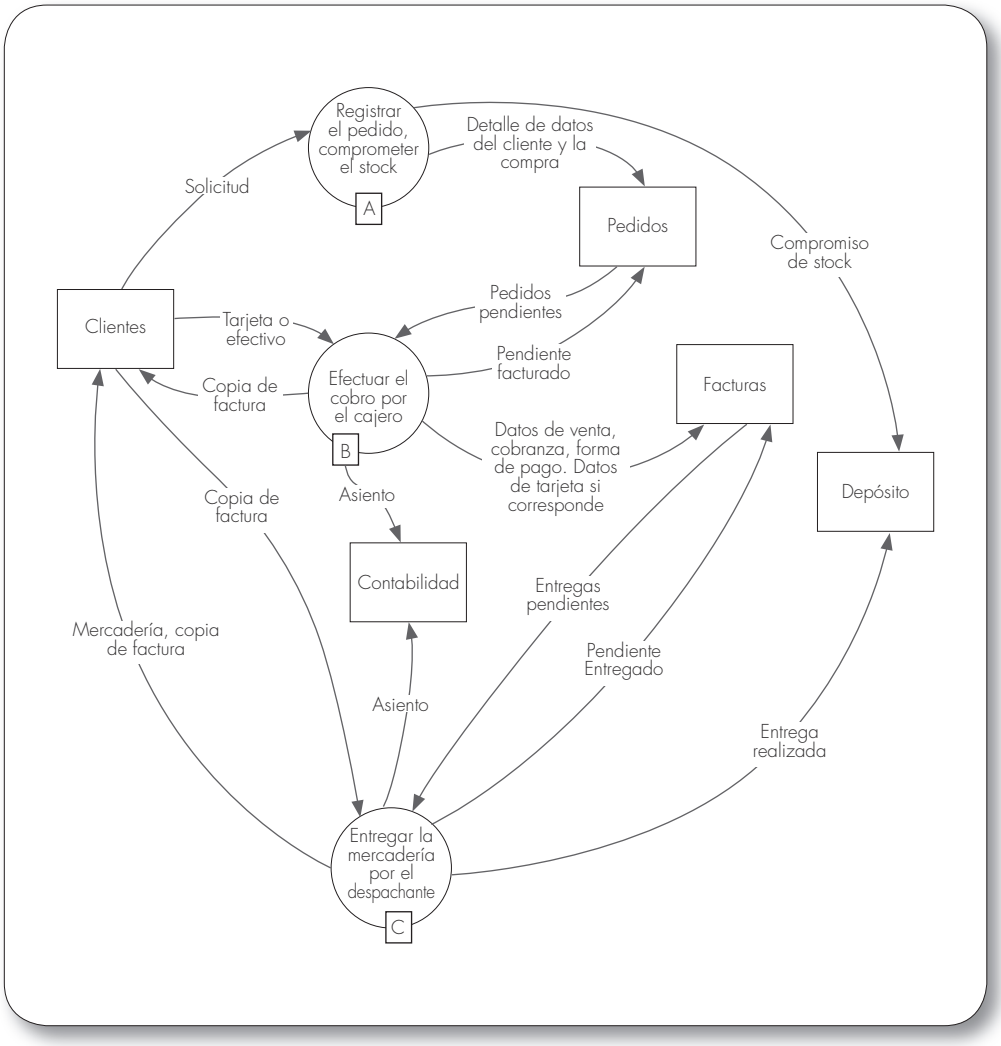


Figura 12.7

Ejemplo de DFD de nivel 1. Venta en local (simplificado)

12.4.2 Proceso de Normalización. Síntesis conceptual de la técnica alcanzando las tres primeras formas normales

a) Identificación de los elementos intervinientes

Se trata de realizar un inventario exhaustivo de los elementos que intervienen y el origen de ellos.

En esta etapa deben individualizarse y excluirse de la base de datos aquellos que pueden ser generados en función de cálculos aplicados sobre otros. Por ejemplo, la nota de un trimestre es un elemento calculable considerando la cantidad de notas del trimestre y la suma de ellas.

Tomemos como ejemplo para este desarrollo la búsqueda de la estructura de datos de un sistema simple en el que se cuente con los siguientes documentos:

- Pedido
- Factura
- Informe diario de totales facturados
- Detalle de facturación del día

Veamos su representación en las siguientes figuras:

Figura 12.8
Pedido

PEDIDO

Número: .....
Fecha: ..... / ..... / .....

Sres.: .....
Domicilio: .....
CUIT: .....

Artículo	Cantidad

Figura 12.9
Factura

FACTURA

A

EMPRESA XX:
CUIT 99-99999999-9
Corresspondiente al pedido N° .....

Número: .....
Fecha: ..... / ..... / .....

Sres.: .....
Domicilio: .....
CUIT: .....

Artículo	Descripción	% IVA	Cant.	P.U.	Importe

Subtotal mercaderías	
IVA	
TOTL FACTURA	

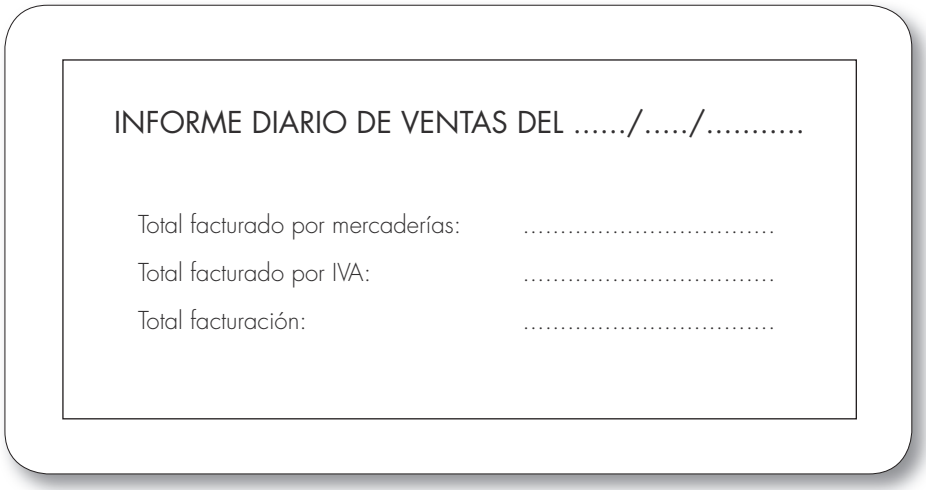


Figura 12.10
Informe diario de totales facturados

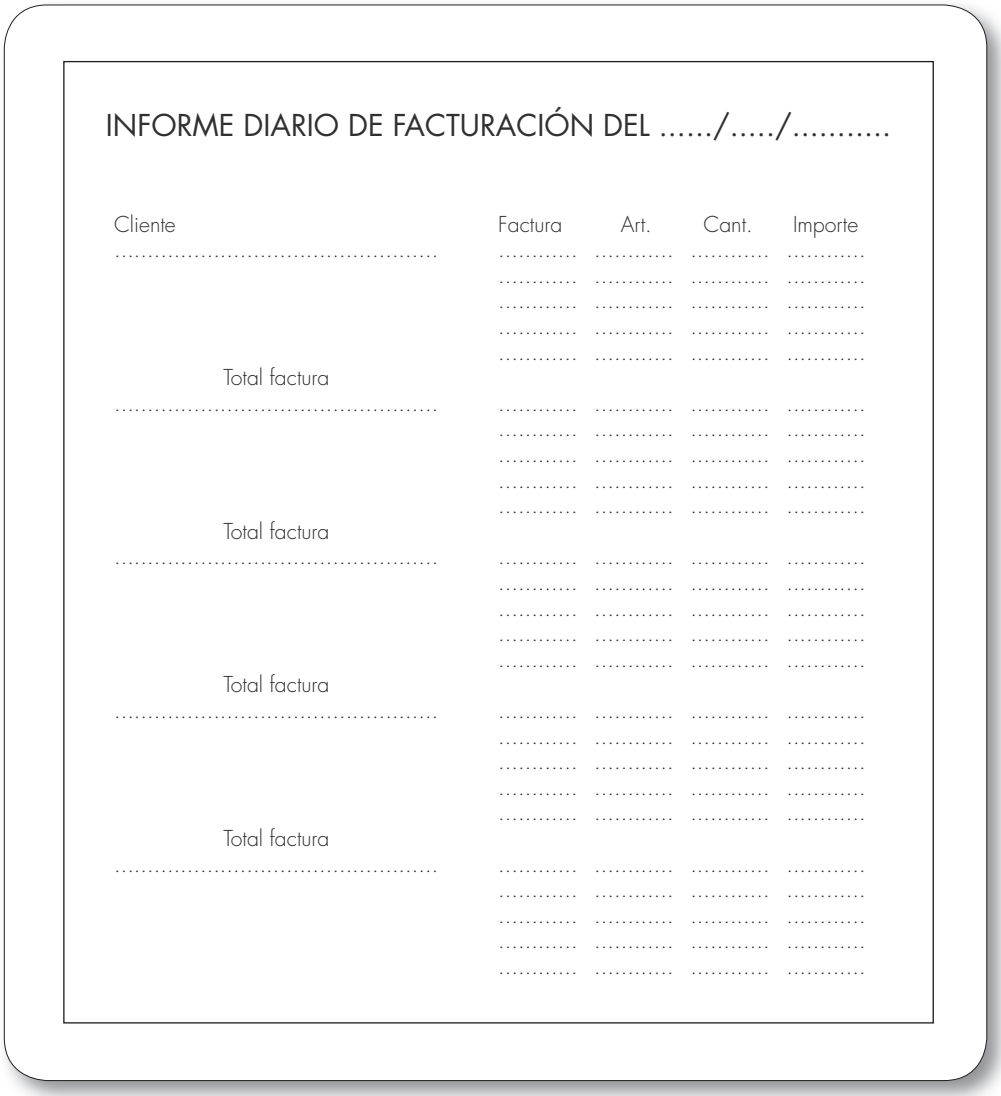


Figura 12.11
Detalle de facturación del día

Por lo tanto, identificamos los siguientes elementos:

Tabla 12.1

Elemento	Pedido	Factura	Informe diario	Detalle diario
Número de pedido	X	X		
Fecha pedido	X			
Nombre cliente	X	X		X
Dirección cliente	X	X		
Datos impositivos cliente	X	X		
Código artículo	X	X		X
Cantidad	X	X		X
Número factura		X		X
Fecha factura		X		
Nombre artículo		X		
Porcentaje IVA aplicable		X		
Precio unitario artículo		X		
Importe total artículo		X		X
Subtotal mercaderías		X		X
Importe IVA		X		
Total factura		X		
Total diario mercaderías			X	
Total diario IVA			X	
Total diario facturación			X	

Incorporaremos ahora una nueva columna, la que denominaremos Columna Origen de Datos (COD) en la que identificaremos con "C" (de calculable) aquellos elementos que pueden ser determinados, desde un estricto punto de vista lógico, mediante la elaboración de datos de otros elementos. Estos datos son "redundantes", no necesitan ser almacenados ya que pueden ser elaborados cuando sea necesario.

Tabla 12.2

Elemento	Pedido	Factura	Informe diario	Detalle diario	COD
Número de pedido	X	X			
Fecha pedido	X				
Nombre cliente	X	X		X	
Dirección cliente	X	X			
Datos impositivos cliente	X	X			
Código artículo	X	X		X	
Cantidad	X	X		X	
Número factura		X		X	
Fecha factura		X			
Nombre artículo		X			
Porcentaje IVA aplicable		X			
Precio unitario artículo		X			
Importe total artículo		X		X	C
Subtotal mercaderías		X		X	C
Importe IVA		X			C
Total factura		X			C
Total diario mercaderías			X		C
Total diario IVA			X		C
Total diario facturación			X		C

En este proceso debemos explicitar y documentar claramente los procesos necesarios para "reconstruir" los valores que excluimos de la estructura lógica, siendo estos:

Tabla 12.3

Elemento eliminado	Proceso para reconstruirlo
Importe total artículo	Cantidad x precio unitario artículo
Subtotal mercaderías	Sumatoria de importe total artículo de todas las líneas de factura
Importe IVA	Sumatoria para todas las líneas de la factura del resultado del siguiente cálculo: Importe total artículo x porcentaje IVA aplicable
Total factura	Subtotal mercaderías + Importe IVA
Total diario mercaderías	Sumatoria del subtotal mercaderías de todas las facturas del día
Total diario IVA	Sumatoria del importe IVA de todas las facturas del día
Total diario facturación	Sumatoria del total factura de todas las facturas del día

b) Agrupación genérica de datos. Formación preliminar de entidades

Se agrupan los elementos en "entidades preliminares" (registros tentativos), en función de sus características obvias.

Se determina cual es el "identificador" de cada entidad, siendo este el elemento que identifica unívocamente a todos los elementos del registro.

En nuestro caso:

Tabla 12.4

Entidad	Identificador	Elementos
PEDIDO	Número de pedido	Fecha pedido, nombre cliente, dirección cliente, datos impositivos cliente, (y por cada línea del pedido) código artículo, cantidad
FACTURA	Número de factura	Número de pedido, fecha factura, nombre cliente, dirección Cliente, datos impositivos cliente, (y por cada línea de la factura) código artículo, cantidad, nombre artículo, porcentaje IVA aplicable, precio unitario artículo

Sobre estas entidades se aplicará el análisis de relaciones.

c) Normalización según la primera forma. Eliminación de grupos repetitivos

Se identifican aquellos grupos de elementos que se repiten para un mismo registro, formando con ellos un registro independiente, vinculado con el originante por medio de su identificador.

Siguiendo con nuestro ejemplo, en una factura las líneas de la ésta representan un grupo repetitivo, todas ellas contienen como elementos "artículo", "cantidad" y "precio", por lo tanto se forma un nuevo registro, llamado "línea de factura", con los elementos "artículo", "cantidad" y "precio" (los que se excluyen de la entidad "factura") más el "número de factura", para permitir el relacionamiento de estas líneas de factura con la factura a la cual pertenecen.

Se define como identificador de la nueva entidad el conjunto de elementos "número de factura" y "artículo".

Tabla 12.5

Entidad	Identificador	Elementos
PEDIDO	Número de pedido	Fecha pedido, nombre cliente, dirección cliente, datos impositivos cliente
PEDIDO-ITEM	Número de pedido, código artículo	Cantidad
FACTURA	Número de factura	Número de pedido, nombre cliente, dirección cliente, datos impositivos cliente, fecha factura
FACTURA-ITEM	Número de factura, código artículo	Cantidad, nombre artículo, porcentaje IVA aplicable, precio unitario artículo

Una tabla está en primera forma normal (1FN) si no contiene grupos repetitivos de elementos para el mismo registro.

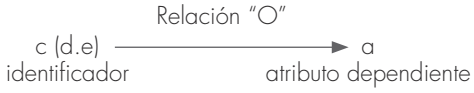
d) Normalización según la segunda forma. Eliminación de dependencias funcionales parciales con el identificador

Este paso consiste en abrir nuevas entidades para los atributos que no tengan dependencia funcional con el identificador completo.

Recordemos los conceptos de dependencia funcional simple y compleja:

Se dice que un dato "a" es funcionalmente dependiente de otro "b" en la relación "R" si para cada valor de "b" hay un y sólo un valor de "a" que cumpla con la relación "R".
Por lo tanto, dado un valor determinado de "b", podremos inferir el valor único que toma "a", determinado por la relación "R".
Gráficamente:

Asimismo se dice que un dato "a" es funcionalmente dependiente de otro dato más complejo "c" (formado por la unión de más de un dato no complejo) en la relación "O" si para cada valor de "c" hay un solo valor de "a" que satisface la relación "O".
Luego, conociendo el valor de los elementos componentes del complejo "c" podemos inferir el valor de "a" tal que cumpla la relación "O".
Gráficamente:



Por lo tanto, la tarea consiste en determinar cuales son los elementos que tienen una relación con el identificador parcializado, formando nuevas entidades con ellos, de forma tal que se pueda, en todo momento, reconstruir la información original.

Siguiendo con nuestro ejemplo encontramos, entre otros, que el nombre del artículo depende del código artículo, no del conjunto número de factura, código artículo.

Aplicando este análisis y realizando la normalización según las 2FN la nueva estructura de datos es:

Tabla 12.6

Entidad	Identificador	Elementos
PEDIDO	Número de pedido	Fecha pedido, nombre cliente, dirección cliente, datos impositivos cliente
PEDIDO-ÍTEM	Número de pedido, código artículo	Cantidad
FACTURA	Número de factura	Número de pedido, nombre cliente, dirección cliente, datos impositivos cliente, fecha factura
FACTURA-ITEM	Número de factura, código artículo	Cantidad
ARTÍCULO	Código artículo	Nombre artículo, porcentaje IVA aplicable, precio unitario artículo

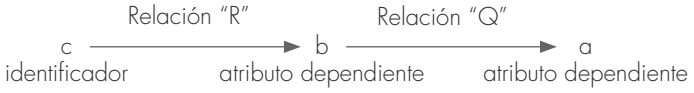
Nótese que la tabla ARTÍCULO es aplicable por igual para la reconstrucción de la información de las tablas PEDIDO-ITEM y FACTURA-ÍTEM.

Una tabla está en segunda forma normal (2FN) si todos sus elementos presentan dependencia funcional con el identificador completo.

e) Normalización según la tercera forma. Eliminación de dependencias funcionales transitivas con el identificador

Este paso consiste en abrir nuevas entidades para los elementos que no tengan dependencia funcional directa con el identificador, es decir, que tengan dependencia funcional con otro elemento y éste con el identificador.

Recordemos los conceptos de dependencia funcional transitiva:

> Siendo "a", "b" y "c" elementos de un conjunto dado, si "a" es funcionalmente dependiente de "b" en la relación "R" y "b" es funcionalmente dependiente de "c" en la relación "Q", se dice que "a" es transitivamente dependiente de "c" en las relaciones "R" y "Q".
> Por lo tanto, dado un valor determinado de "a", podremos inferir el valor único que toma "c", determinado por la relación compuesta "R" y "Q".
> Gráficamente:
>
> 
>

Por lo tanto, la tarea consiste en determinar cuales son los elementos que tienen una relación funcional indirecta con el identificador, formando nuevas entidades con ellos, de forma tal que se pueda, en todo momento, reconstruir la información original.

Siguiendo con nuestro ejemplo encontramos, entre otros, que la "dirección cliente" depende del "nombre del cliente" y este del "número de pedido" (para cualquier pedido de ese cliente la dirección será la misma, dependiendo del cliente, por lo tanto, la relación de dependencia de la dirección es con el cliente).

Aplicando este análisis y realizando la normalización según las 3FN la nueva estructura de datos es:

Tabla 12.7

Entidad	Identificador	Elementos
PEDIDO	Número de pedido	Fecha pedido, nombre cliente
PEDIDO-ITEM	Número de pedido, código artículo	Cantidad
FACTURA	Número de factura	Número de pedido, nombre cliente, fecha factura
FACTURA-ITEM	Número de factura, código artículo	Cantidad
ARTICULO	Código artículo	Nombre artículo, porcentaje IVA aplicable, precio unitario artículo
CLIENTE	Nombre cliente[99]	Dirección cliente, datos impositivos cliente

Nótese que la tabla CLIENTE es aplicable por igual para la reconstrucción de la información de las tablas PEDIDO-ÍTEM y FACTURA-ÍTEM.

[99] En la realidad nos encontraremos con Códigos de Cliente (que puede ser algún documento de uso generalizado como el CUIT/CUIL o un número ad hoc) siendo el nombre un elemento dependiente del Código.

Una tabla está en tercera forma normal (3FN) si todos sus elementos presentan dependencia funcional directa con el identificador completo.

Como mencionamos anteriormente, podemos graficar la relación entre las entidades determinadas con el diagrama de entidad relación (DER). En nuestro caso, un DER simple del ejemplo dado sería:

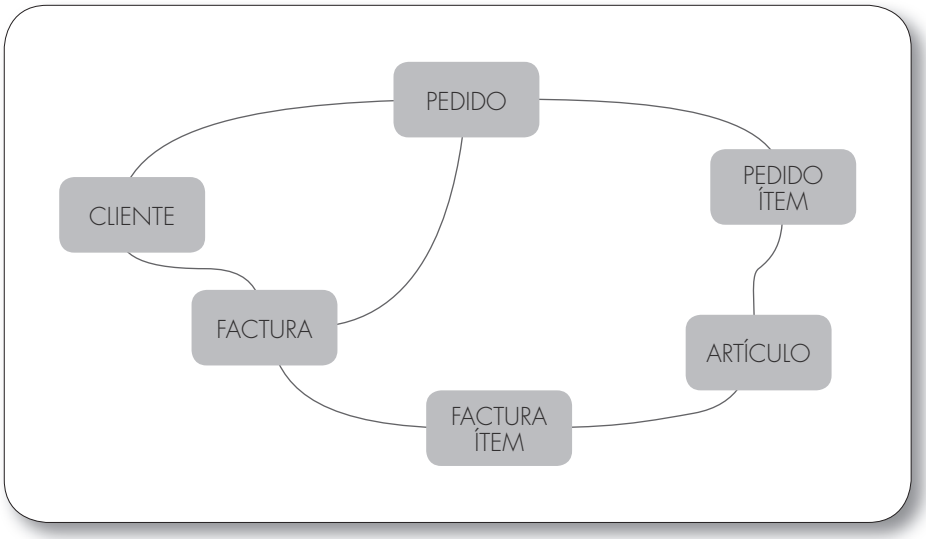


Figura 12.12
Ejemplo de un DER

12.4.3 Componentes y ejemplos de diagrama de transición de estados (DTE)

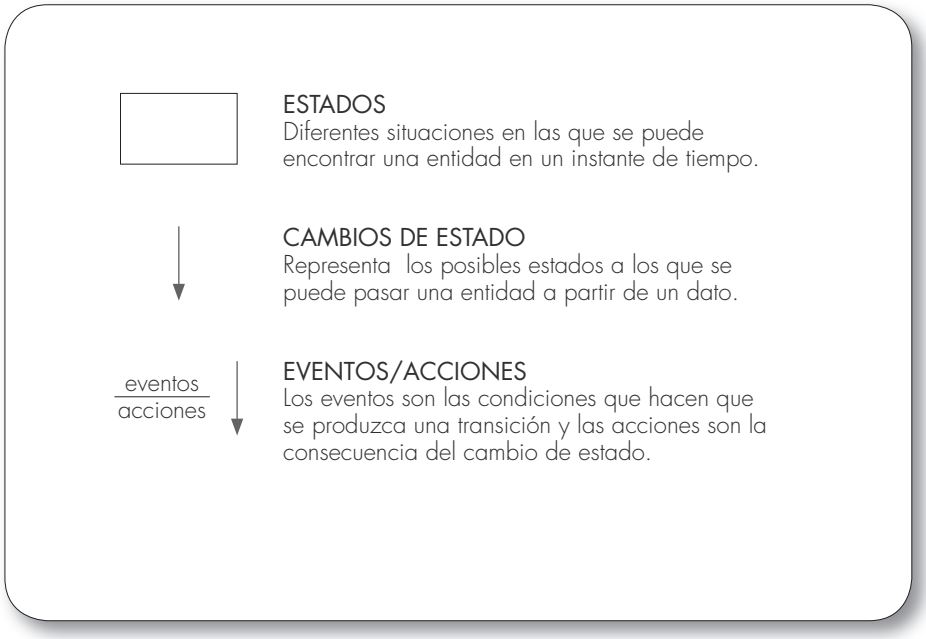
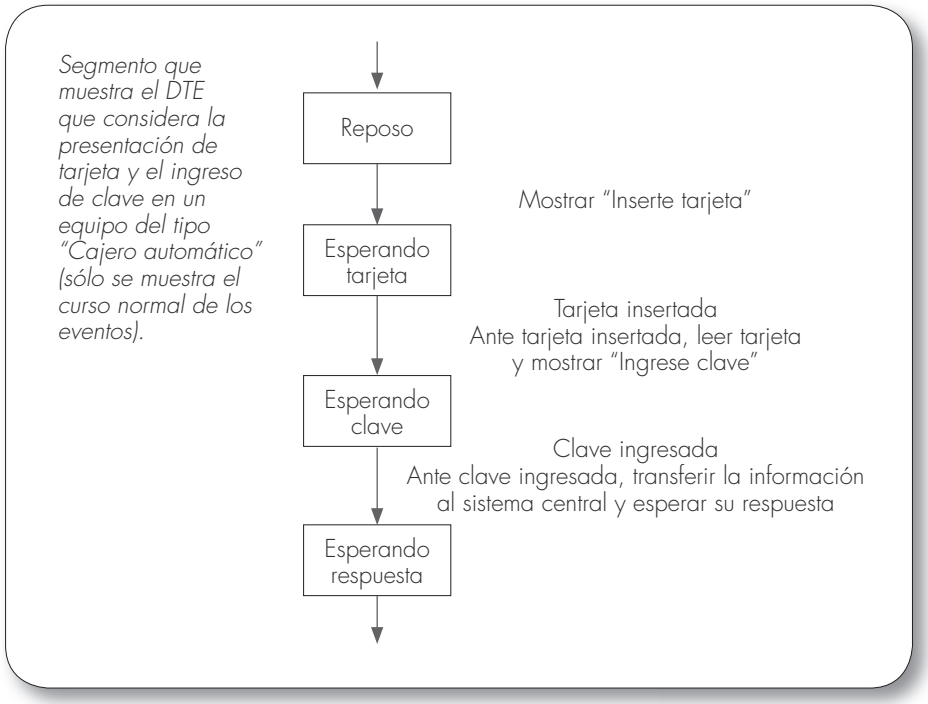


Figura 12.13
Componentes de un diagrama de transición de estados

Figura 12.14

Ejemplo de componentes de un diagrama de transición de estados



12.4.4 Componentes y ejemplos de diagramas casos de uso

Figura 12.15

Componentes de los diagramas de casos de uso

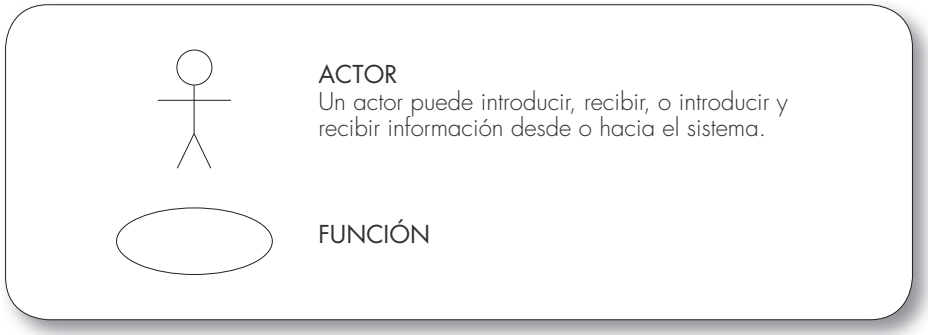
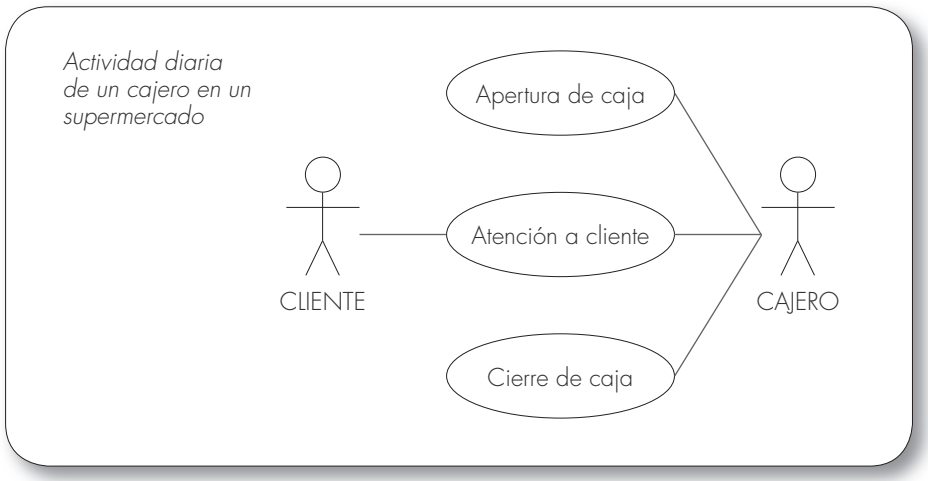


Figura 12.16

Ejemplo 1 de diagrama de casos de uso (simplificado)



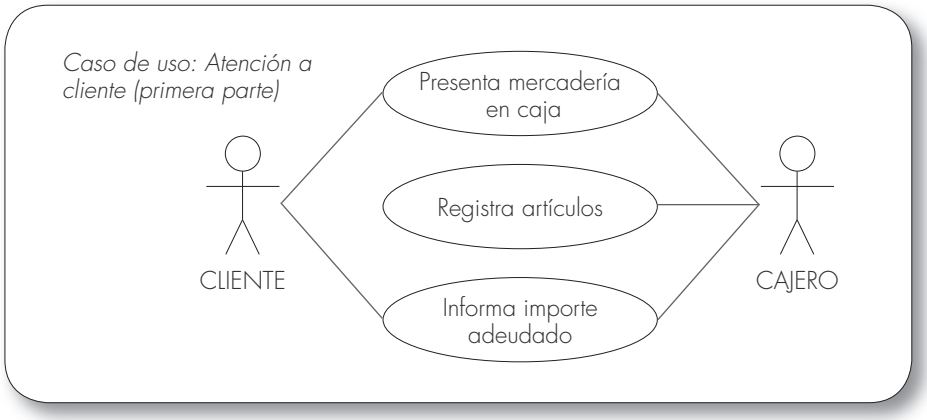


Figura 12.17

Ejemplo 2 de diagrama de casos de uso (simplificado)

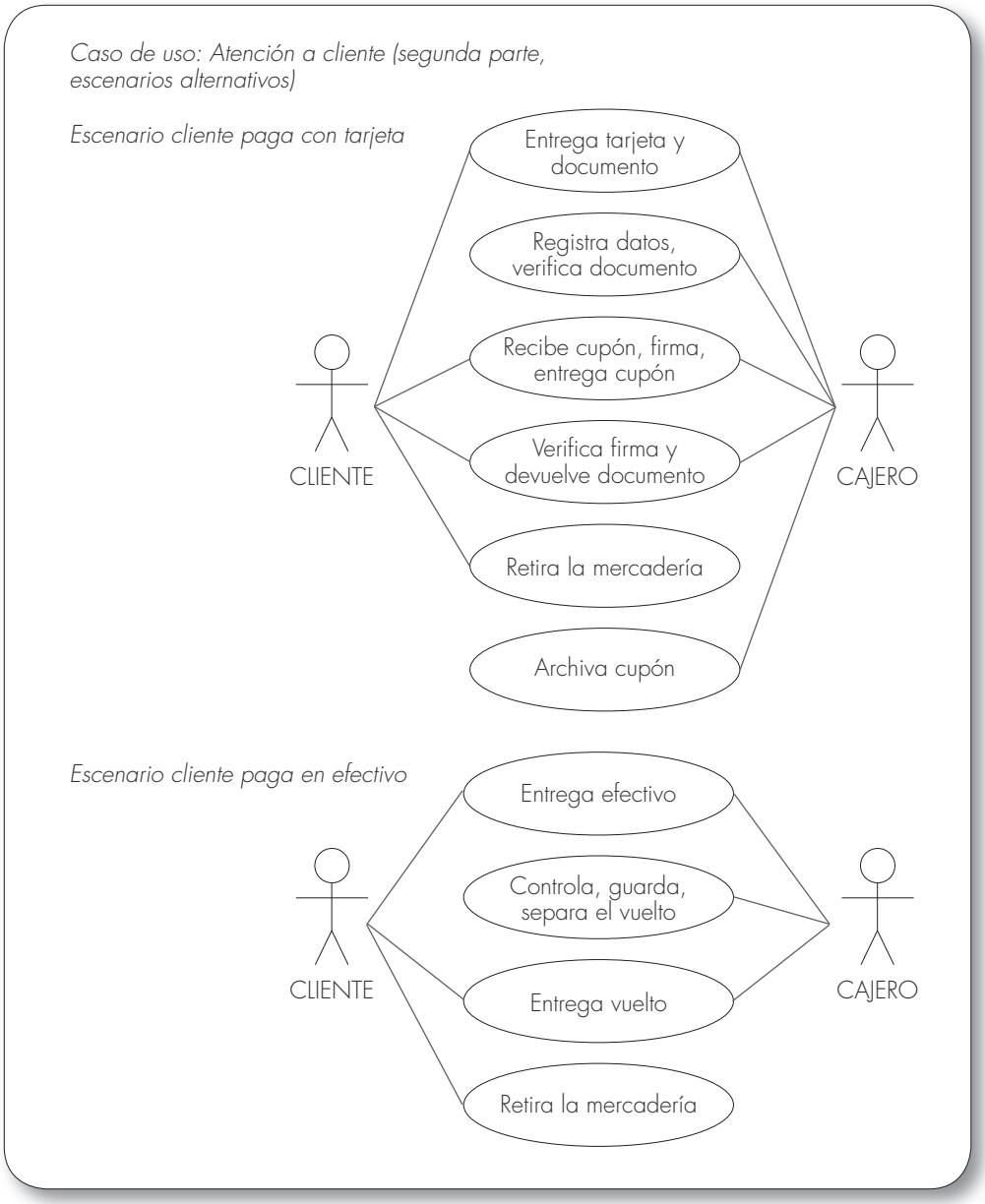


Figura 12.18

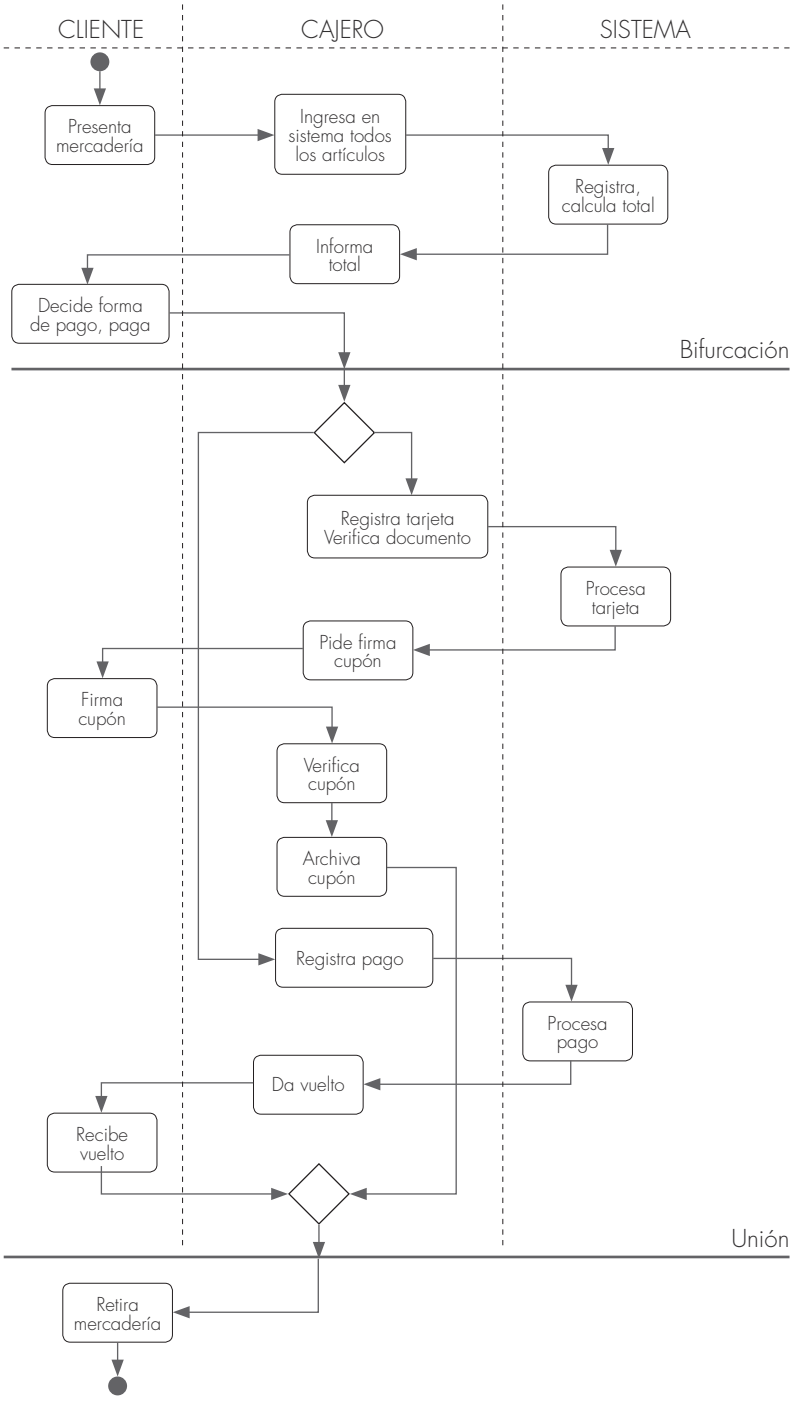
Ejemplo 3 de diagrama de casos de uso (simplificado)

12.4.5 Ejemplos de diagramas de actividades y de diagrama de secuencias

Figura 12.19
Ejemplo de diagrama de actividades (simplificado)

Exponen las actividades de un caso de uso en la forma en que se van dando, incluyendo actividades paralelas y decisiones tomadas.

Tiene una conformación similar al cursograma.



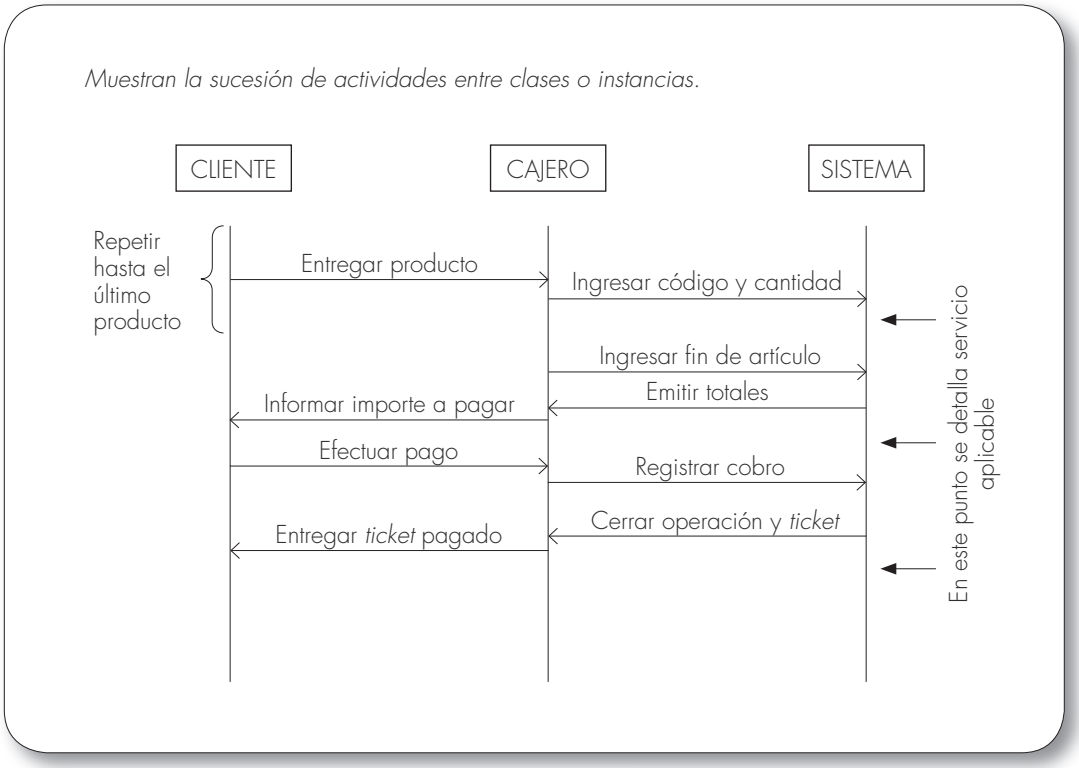


Figura 12.20
Ejemplo de diagrama de secuencia (simplificado)

CAPÍTULO 13

ESTRATEGIA DE **SISTEMAS** Y **TECNOLOGÍAS** DE LA **INFORMACIÓN**

CAPÍTULO 14

ADMINISTRACIÓN DE **PROYECTOS** Y **RECURSOS INFORMÁTICOS**

CAPÍTULO 15

DECISIONES DE **INVERSIÓN** EN **TICs**: **IMPACTO ECONÓMICO** Y DE **NEGOCIOS**

ALCANCE

En este parte se presentan las cuestiones relativas a la elaboración e implementación de la estrategia de sistemas y tecnologías de la información.

En el Capítulo 13 se discute sobre el valor de las tecnologías de la información en las organizaciones y la interrelación de la estrategia específica con la estrategia general, para luego presentar los diferentes componentes en los que se desgrana la visión estratégica de SI/TI; destacando las decisiones a tomar en cada uno de los diferentes ámbitos de decisión para, luego, presentar los planes tácticos que establecen las acciones a llevar a cabo para desarrollar la visión estratégica.

En el Capítulo 14 se tratan los dos grandes tipos de actividades de gestión que se llevan a cabo para implementar la estrategia, la gestión de operaciones y la administración de proyectos. Con relación a este último punto se presentan las diferentes etapas y las principales herramientas utilizadas para la gestión de proyectos.

Por último, en el Capítulo 15 se abordan distintos aspectos económicos de la decisión de inversión en TI. También se presentan consideraciones generales sobre la evolución económica de proyectos y, en particular, los referidos a TI, resaltando la metodología de evolución costo-beneficio, y los principales costos y beneficios relacionados con proyectos de la temática tratada. Finalmente, se presentan los métodos de Costo Total de Propiedad (CTP) y Valor Total de Propiedad (VTP), sus características más relevantes, y sus ventajas y desventajas.