DIAGRAMAS DE SECUENCIA DEL SISTEMA, CONTRATOS DE LAS OPERACIONES DEL SISTEMA, GLOSARIO Y PAQUETES

Extraído de:
UML y Patrones. 2ª Edición.
Craig Larman.
Prentice Hall. 2003

1. Diagramas de Secuencia del Sistema

Volvamos nuestra atención a los casos de usos y al análisis del modelado del dominio. Antes de empezar el trabajo de diseño, resultará útil realizar un estudio adicional del dominio del problema. Parte de este estudio comprende la aclaración de los eventos del sistema de entrada y salida relacionados con nuestro sistema, que puede representarse en diagramas de secuencia UML.

Un **diagrama de secuencia del sistema** es un artefacto que muestra los eventos de entrada y salida relacionados con el sistema que se está estudiando. UML incluye la notación de los diagramas de secuencia para representar eventos que parten de los actores externos hacia el sistema.

Antes de continuar con el diseño lógico de cómo funcionará la aplicación software, conveniente estudiar y definir su comportamiento como una "caja negra". El **comportamiento del sistema** es una descripción de *qué* hace el sistema, sin explicar cómo hace. Una parte de esa descripción es un diagrama de secuencia del sistema. Otras partes comprenden los casos de uso y los contratos del sistema (que se presentarán después).

Los casos de uso describen cómo interactúan los actores externos con el sistema software que estamos interesados en crear. Durante esta interacción, un actor genera eventos sobre un sistema, normalmente solicitando alguna operación como respuesta. Por ejemplo, cuando un cajero inserta el ID de un artículo está solicitando al sistema PDV que registre la venta de ese artículo. Ese evento de solicitud inicia una operación sobre el sistema.

Es deseable aislar e ilustrar las operaciones que un actor externo solicita a un sistema, porque constituyen una parte importante de la comprensión del comportamiento del sistema. UML incluye los **diagramas de secuencia** como notación que puede representar las interacciones de los actores y las operaciones que inician.

Un **diagrama de secuencia del sistema** (DSS) es un modelo que muestra, para un escenario específico de un caso de uso, los eventos que generan los actores externos, el orden y los eventos entre los sistemas. Todos los sistemas se tratan como cajas negras; los diagramas

destacan los eventos que cruzan los límites del sistema desde los actores a los sistemas.

En la práctica real, debería hacerse un DSS para el escenario principal de éxito del caso de uso, y los escenarios alternativos complejos o frecuentes. Aunque a efectos de la asignatura se realizarán DSS para todos los escenarios alternativos.

UML no define nada denominado diagrama de secuencia "del sistema", sino simplemente diagrama de secuencia. La calificación se utiliza para subrayar su aplicación para representar sistemas como cajas negras. Posteriormente, se utilizarán los diagramas: de secuencia en otro contexto -para ilustrar el diseño de la interacción entre objetos: software para completar un trabajo.

Un DSS muestra, para un curso de eventos específico en un caso de uso, los actores externos que interaccionan directamente con el sistema, el sistema (como una caja negra) y los eventos del sistema que genera el actor (ver Figura 1). El tiempo avanza hacia abajo, y la ordenación de los eventos debería seguir su orden en el caso de uso. Los eventos del sistema podrían contener parámetros. Este ejemplo muestra el escenario principal de éxito del caso de uso Procesar Venta. Se indica que el cajero genera los eventos del sistema *crearNuevaVenta*, *introducirArticulo*, *finalizarVenta*, y *realizarPago*.

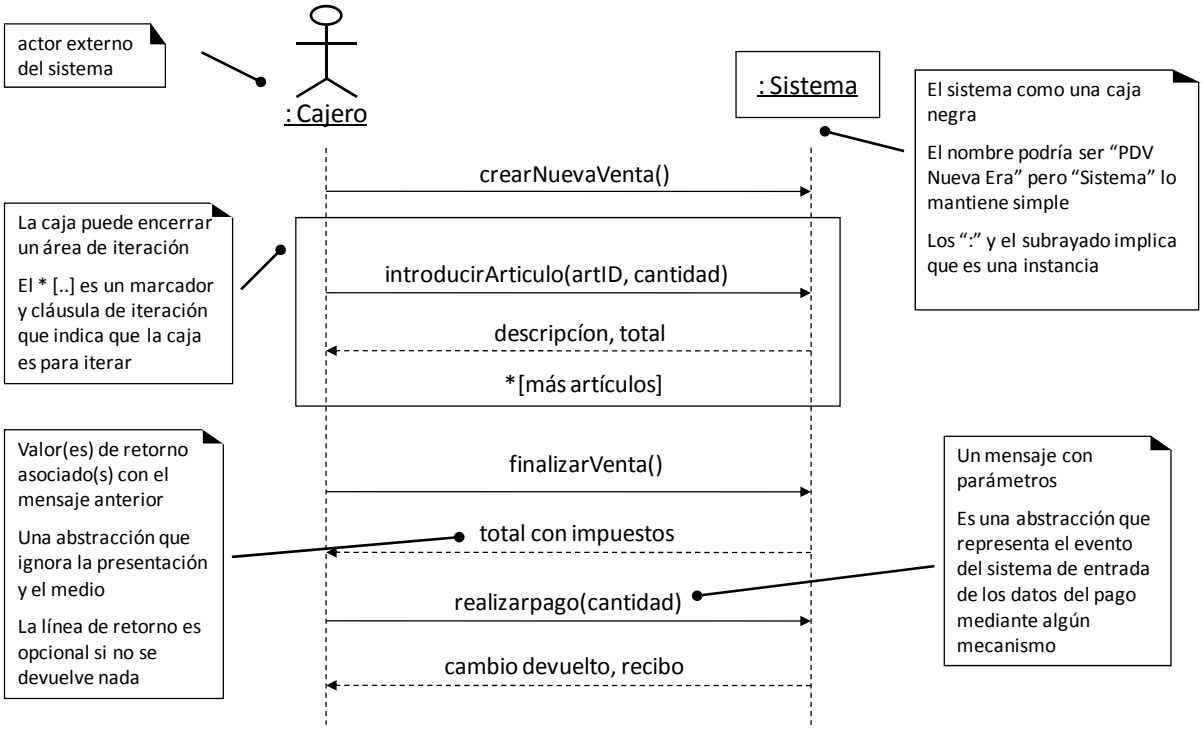

Figura 1. DSS para un escenario de Procesar Venta.

1.1. DSS entre Sistemas

Los DSS también pueden utilizarse para ilustrar las colaboraciones entre sistemas, como entre el PDV NuevaEra y el sistema externo que autoriza pagos a crédito.

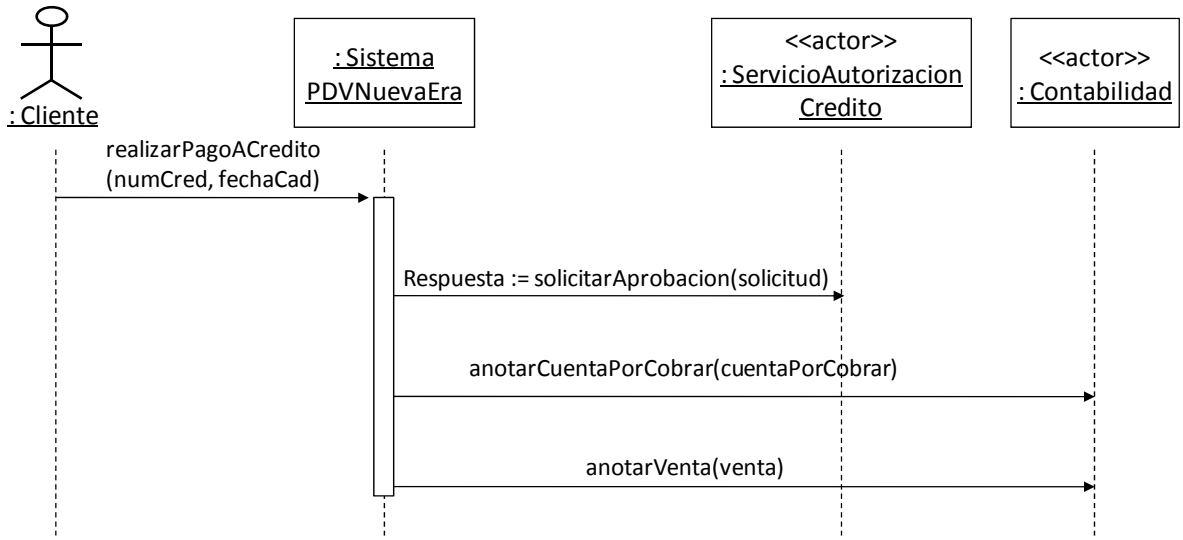

Figura 2. DSS del pago a crédito.

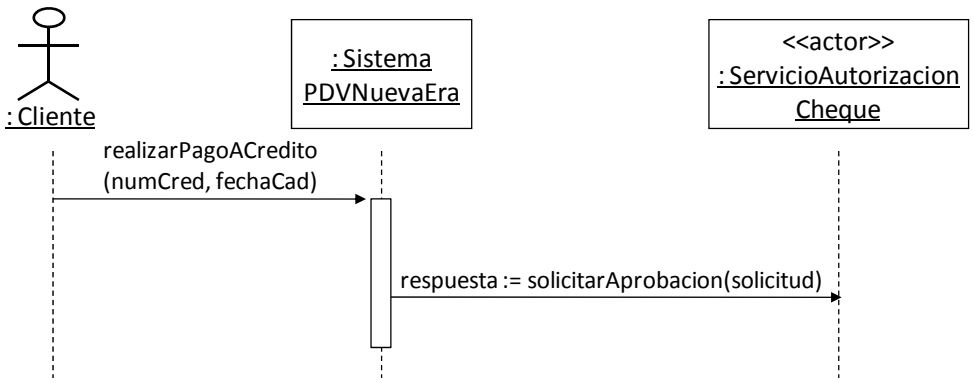

Figura3. DSS del pago con cheque.

1.2. DSS y los Casos de Uso

Un DSS muestra los eventos del sistema para un escenario de un caso de uso, por tanto, se genera para el estudio de un caso de uso (ver Figura 4).

1.3. Eventos del Sistema y los Límites del Sistema

Para identificar los eventos del sistema, es necesario tener claros los límites del sistema, como se presentó en el capítulo sobre los casos de uso. Por lo que toca al desarrollo de software, el límite del sistema normalmente se elige para que sea el propio sistema software (y posiblemente hardware); en este contexto, un evento del sistema es un evento externo que lanza un estímulo directamente al software (ver Figura 5).

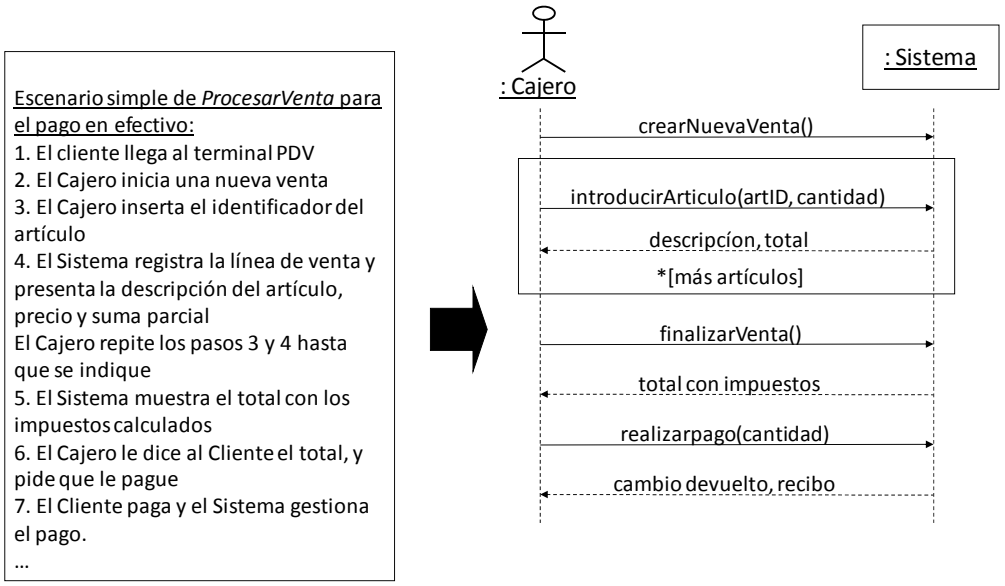

Figura 4. Los DSS se derivan de los casos de uso.

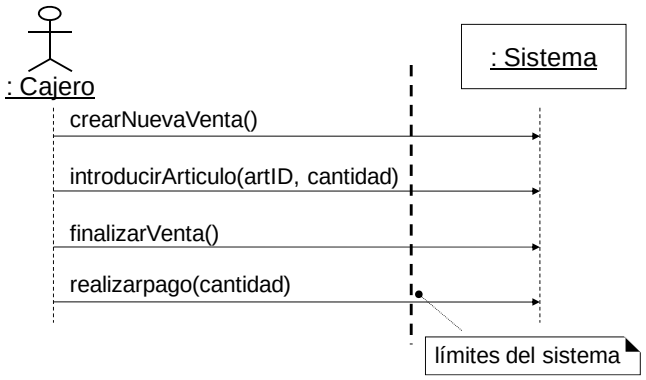

Figura 5. Definición de los límites del sistema.

Consideremos el caso de uso Procesar Venta para identificar los eventos del sistema. Primero, debemos determinar los actores que interactúan directamente con el sistema software. El cliente interactúa con el cajero, pero para este escenario simple de pago en efectivo, no interactúa directamente con el sistema PDV -sólo lo hace el cajero-. Por tanto, el cliente no es un generador de eventos del sistema; sólo lo es el cajero.

1.4. Asignación de Nombres a los Eventos y Operaciones

Los eventos del sistema (y sus operaciones del sistema asociadas) deberían expresarse al nivel de intenciones en lugar de en términos del medio de entrada físico o a nivel de elementos de la interfaz de usuario.

También se mejora la claridad, el comenzar el nombre de un evento del sistema con un verbo (añadir..., insertar..., finalizar..., crear...), como en la Figura 6, puesto que resalta la orientación de orden de estos eventos.

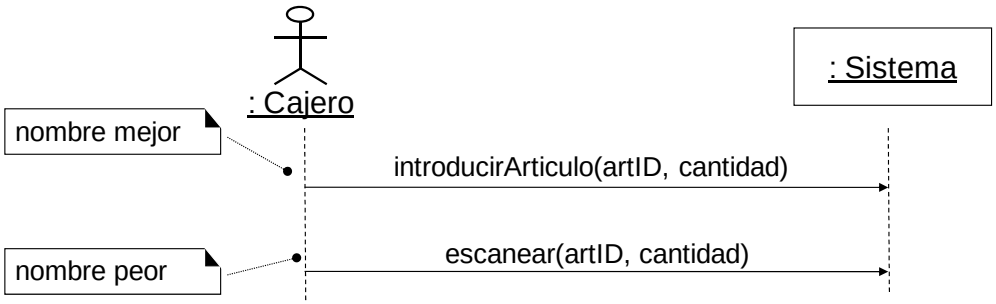

Figura 6. Elección de los nombres de los eventos y las operaciones a un nivel abstracto.

Así, "introducirArticulo" es mejor que "escanear" (esto es, escanear con láser) porque captura la intención de la operación, al mismo tiempo que permanece abstracta y sin compromiso respecto a las elecciones de diseño sobre qué interfaz utilizar para capturar el evento del sistema.

1.5. Mostrar el Texto del Caso de Uso

A veces, es deseable mostrar al menos fragmentos del texto del caso de uso del escenario, con el fin de aclarar o enriquecer las dos vistas (ver Figura 7). El texto proporciona los detalles y el conte.xto; el diagrama resume visualmente la interacción.

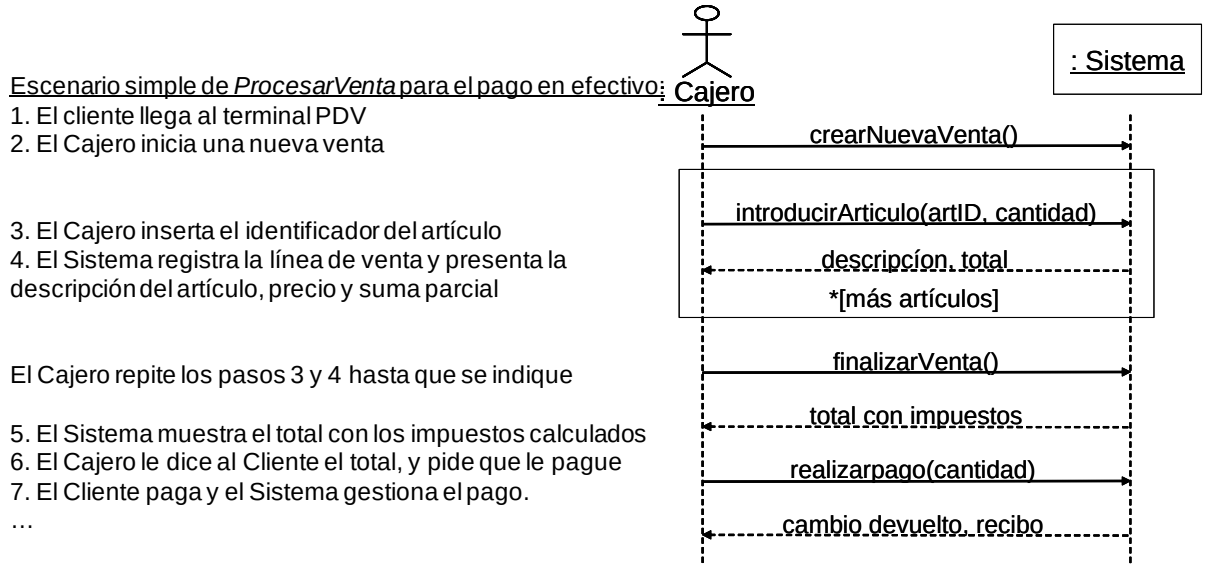

Figura 7. DSS con texto del caso de uso.

1.6. Los DSS y el Glosario

Los términos representados en los DSS (operaciones, parámetros, valores de retorno) son concisos. Éstos podrían necesitar una explicación más adecuada de manera que, durante el trabajo de diseño, esté claro qué es lo que entra y lo que sale. Si no se explicó en los casos de uso, podría utilizarse el Glosario.

Sin embargo, como siempre que discutimos la creación de artefactos distintos al código (lo esencial del proyecto), hay que ser desconfiado. Con los datos del Glosario debería hacerse algún uso o decisión realmente significativa, de lo contrario es un trabajo innecesario de poco valor.

2. Contratos de las Operaciones del Sistema

Los contratos de las operaciones pueden ayudar a definir el comportamiento del sistema; describen el resultado de la ejecución de las operaciones del sistema en función de los cambios de estado de los objetos del dominio. Esta sección explora su uso.

Los casos de uso son el principal mecanismo para describir el comportamiento del sistema y, normalmente es suficiente. Sin embargo, algunas veces se necesita una descripción más detallada del comportamiento del sistema. **Los contratos describen el comportamiento detallado del sistema en función de los cambios de estado de los objetos del Modelo del Dominio, después de la ejecución de una operación del sistema**.

Se pueden definir contratos para las operaciones del sistema -operaciones que el sistema, como una caja negra, ofrece en su interfaz pública para manejar los eventos del sistema entrantes-. Las operaciones del sistema se pueden identificar descubriendo estos eventos del sistema, como se mostraba en la Figura1.

El conjunto completo de operaciones del sistema, de todos los casos de uso, define la interfaz pública del sistema, viendo al sistema como un componente o clase individual En UML, el sistema como un todo se puede representar mediante una clase.

Antes de examinar las razones para la escritura de un contrato, merece la pena presentar un ejemplo. A continuación, se presenta un contrato para la operación del sistema *introducirArticulo*.

Contrato CO2: introducirArticulo

Operación:	introducirArticulo(articuloID:ArticuloID, cantidad: integer)
Referencias cruzadas:	Caso de Uso: Procesar Venta
Responsabilidades:	El sistema registra el artículo comprado y muestra su descripción y el total acumulado hasta el momento
Precondiciones:	Hay una venta en curso
Postcondiciones	• Se creó una instancia de LineaDeVenta ldv • ldv se asoció con la Venta actual • ldv se asoció con una EspecificaciónDelProducto, en base a la coincidencia del articuloID • ldv.cantidad pasó a ser cantidad

2.1. Secciones del Contrato

La descripción de cada una de las secciones del contrato se muestra en el siguiente esquema.

Contrato <Identificador>:<Nombre>

Operación:	Nombre de la operación y parámetros
Referencias cruzadas:	(opcional, pero aconsejable) Casos de uso en los que pueden tener

	lugar esta operación
Responsabilidades:	Descripción informal de las responsabilidades o tareas que debe realizar la operación
Salidas:	(opcional) Por ejemplo, mensajes o informes que se envían fuera del sistema. No se refieren a la interfaz de usuario, sino a flujos de información
Notas:	(opcional) Sugerencias de diseño, algoritmos, etc.
Excepciones:	(opcional) Responsabilidades de la operación en casos excepcionales
Precondiciones:	Suposiciones relevantes sobre el estado del sistema o de los objetos del Modelo del Dominio, antes de la ejecución de la operación. No se comprobará en la lógica de esta operación, se asume que son verdad, y son suposiciones no triviales que el lector debe saber que se hicieron
Postcondiciones	El estado de los objetos del Modelo del Dominio después de que se complete la operación. Se discute con detalle a continuación

Veamos a continuación en detalle la sección de postcondiciones. Nótese que cada una de las postcondiciones .del ejemplo *introducirArticulo* incluía una categorización como *creación de instancias* o *formación de asociaciones*. He aquí un punto clave: La postcondición describe cambios en el estado de los objetos del Modelo del Dominio. Los cambios de estado del Modelo del Dominio comprenden la creación de instancias, formación o rotura de asociaciones y cambio en los atributos.

Las postcondiciones **no son acciones** que se ejecutarán durante la operación; más bien, **son declaraciones** sobre los objetos del Modelo del Dominio que son verdad cuando la operación ha terminado -después de que el humo se haya despejado-.

En resumen, las postcondiciones se dividen en estas categorías:

- Creación y eliminación de instancias.
- Modificación de atributos.
- Formación y rotura de asociaciones (siendo precisos, enlaces UML).

Como ejemplo de postcondición que rompe una asociación, considérese una operación que permite la eliminación de líneas de venta. La postcondición podría decir "Se rompió la asociación seleccionada de la *LineaDeVenta* con la *Venta*". En otros dominios, cuando se cancela un préstamo o cuando alguien deja de ser socio de alguna organización, se rompen las asociaciones.

La postcondición de la eliminación de instancias es más rara, porque en el mundo real uno, normalmente, no se preocupa de forzar explícitamente la destrucción de una cosa. Sin embargo, como ejemplo: en muchos países, después de que una persona se haya declarado en bancarrota y hayan pasado siete o diez años, se deben destruir todos los registros de su declaración de bancarrota, por ley. Nótese que esto es una perspectiva conceptual, no de implementación. No son declaraciones sobre liberar la memoria del ordenador ocupada por objetos software.

La cualidad importante es ser declarativo y enunciar con un estilo orientado al carmbio en lugar de orientado a la acción, puesto que las postcondiciones son declaraciones sobre los estados o resultados, en lugar de una descripción de las acciones a ejecutar, o un diseño de una solución.

Estas postcondiciones se expresan en el contexto de los objetos del Modelo del Dominio ¿Qué instancias se pueden crear? -aquellas del Modelo del Dominio; ¿qué asociaciones se pueden formar?- las que se encuentran en el Modelo del Dominio; y así sucesivamente.

Expresados en un estilo declarativo de cambio de estado, los contratos son una herramienta excelente para el análisis de requisitos que describen los cambios de estado que requiere una operación del sistema (en función de los objetos del Modelo del Dominio) sin tener que describir cómo se van a llevar a cabo. En otras palabras, el diseño del software y la solución se puede diferir, y uno puede centrarse, analíticamente en qué debe suceder, en lugar de en cómo se va a realizar. Además, las postcondiciones soportan detalles de grano fino y una declaración más específica de cuál debe ser el resultado de la operación.

También es posible expresar este nivel de detalle en los casos de uso, pero normalmente no es deseable, puesto que entonces pasan a ser excesivamente elocuentes y detallados. Considérese la postcondición del ejemplo anterior. No se hace ningún comentario sobre el modo de crear una instancia de *LineaDeVenta*, o cómo se asocia con una *Venta*. Esto podría ser una declaración sobre escribir en folios y que se grapen, la utilización de la tecnología lava para crear objetos software.

Se debería pensar en las postcondiciones utilizando el siguiente símil: El sistema y sus objetos se presentan en el escenario de un teatro.

1. Antes de la operación, se toma una fotografía del escenario.

2. Se baja el telón y se aplica la operación del sistema (ruido metálico de fondo de martillos o campanas, gritos, chirridos...).

3. Se sube el telón y se toma una segunda fotografía.

4. Se comparan las fotografías anterior y posterior, y se expresan como postcondiciones el cambio en el estado del escenario (Se creó una *LineaDeVenta*...).

A continuación se analiza la motivación de las postcondiciones de la operación del sistema *introducirArticulo*.

1. **Creación y eliminación de instancias.** Después de introducir el *articuloID* y la cantidad de un artículo, ¿qué nuevo objeto debe haberse creado? Una *LineaDeVenta*. Por tanto:

   – Se creó una instancia de *LineaDeVenta ldv* (creación de instancias);

   Obsérvese el nombre de la instancia. Este nombre simplificará las referencias a la nueva instancia en otras sentencias de la post -condición.

2. **Modificación de atributos.** Después de que el cajero haya introducido el *articuloID* y la cantidad, ¿qué atributos de los objetos nuevos o de los ya existentes deberían haberse modificado? La cantidad de la *LineaDeVenta* debería haber pasado a ser igual que el parámetro cantidad. De ahí:

- *ldv.cantidad* pasó a ser *cantidad* (modificación de atributos).

3. **Formación y rotura de asociaciones.** Después de que el cajero haya introducido el *articuloID* y la cantidad, ¿qué asociaciones entre los objetos nuevos o los ya existentes deberían haberse formado o roto? La nueva *LineaDeVenta* debería haberse relacionado con su *Venta*, y su *EspecificacionDelProdueto*. Así:

   - *ldv* se asoció con la *Venta* actual (formación de asociaciones).

   - *ldv* se asoció con una *EspecificacionDelProducto*, en base a la coincidencia del articuloID (formación de asociaciones).

Nótese la indicación informal de que se forma una relación con una *EspecificacionDelProducto* particular -aquella cuyo *articuloID* se corresponda con el parámetro. Aunque es posible utilizar otros lenguajes más sofisticados y formales, como utilizar el Lenguaje de Restricciones de Objetos (OCL, Object Constraint Language), nuestra recomendación es simplificar en la medida de lo posible.

Es normal, durante la creación de los contratos, descubrir la necesidad de registrar nuevas clases conceptuales, atributos o asociaciones en el Modelo del Dominio. Es conveniente enriquecer el Modelo del Dominio cuando se hagan nuevos descubrimientos mientras se realizan los contratos de las operaciones.

2.2. Guías sobre los Contratos

2.2.1. ¿Cuándo son Útiles los Contratos? Contratos vs. Caso de Uso

Los casos de uso son el principal repositorio de requisitos del proyecto. Podrían proporcionar la mayoría o todos los detalles necesarios para saber qué hacer en el diseño, en cuyo caso, los contratos no son útiles. Sin embargo, hay situaciones en las que los detalles y la complejidad de los cambios de estado requeridos, son difíciles de capturar en los casos de uso.

Por ejemplo, considere un sistema de reservas de vuelos y la operación del sistema *añadirNuevaReserva*. La complejidad es muy alta considerando todos los objetos del dominio que se deben cambiar, crear y asociar. Estos detalles de grano fino se pueden escribir en detalle en el caso de uso asociado a esta operación, pero dará lugar a un caso de uso extremadamente detallado (por ejemplo, anotando cada atributo que se debe cambiar en todos los objetos).

Obsérvese que el formato de la postcondición del contrato ofrece y promueve un lenguaje muy preciso, analítico y exigente que soporta una detallada minuciosidad.

Si, únicamente basándose en los casos de uso y mediante continuas colaboraciones (verbales) con un experto en la materia de estudio, los desarrolladores pueden entender cómodamente qué hacer, entonces conviene evitar la escritura de los contratos. Sin embargo, en aquellas situaciones donde la complejidad es alta y añade valor la precisión detallada, los contratos son otra herramienta de requisitos.

Muy a menudo, los contratos no estarán muy justificados de manera que si un equipo está creando contratos para todas las operaciones del sistema de cada caso de uso, es una advertencia de que, o bien los casos de uso son algo deficientes, o no hay suficiente y continua colaboración o acceso a los expertos en la materia de estudio, o el equipo está haciendo demasiada

documentación innecesaria.

El caso de estudio del PDV NuevaEra muestra más contratos de los que probablemente sean necesarios, por cuestiones pedagógicas. En la práctica, la mayoría de los detalles que recogen se pueden inferir de manera obvia a partir del texto de los casos de uso. Por otro lado, "obvio" es un concepto muy escurridizo. Esto quiere decir, que a efectos de la asignatura, se considerará que se realizan todos los contratos de las operaciones del sistema.

2.2.2. Consejos acerca de la Escritura de Contratos

Se deben establecer las postcondiciones de forma declarativa, con una sentencia impersonal expresada en pasado para destacar que se trata de una declaración de un cambio de estado en lugar del diseño de la manera en la que se va a realizar. Por eiemplo:

- (mejor) Se creó una LineaDeVenta
- (peor) Cree una LineaDeVenta.

Se debe establecer una relación entre los objetos existentes o aquellos creados recientemente mediante la definición de la formación de asociaciones. Por ejemplo, no es suficiente que se cree una instancia de *LineaDeVenta* cuando tenga lugar la operación *introducirArticulo*. Después de que se complete la operación, también debería cumplirse que la nueva instancia creada se asoció con la *Venta*; de ahí que:

- La *LineaDeVenta* se asoció con la *Venta* (formación de asociaciones).

2.2.3. El error más habitual en la creación de contratos

El problema más común es olvidarse de incluir la formación de asociaciones. En particular, cuando se crean nuevas instancias, es muy probable que se necesiten establecer asociaciones con varios objetos. ¡No lo olvidéis!

2.3. Ejemplo del PDV NuevaEra: Contratos de las Operaciones del Sistema de Procesar Venta

Contrato CO1: crearNuevaVenta

Operación:	crearNuevaVenta()
Referencias cruzadas:	Caso de Uso: Procesar Venta
Precondiciones:	Ninguna
Postcondiciones	• Se creó una instancia de Venta v • v se asoció con el Registro • Se inicializaron los atributos de v

Contrato CO2: introducirArticulo

Operación:	introducirArticulo(articuloID:ArticuloID, cantidad: integer)
Referencias cruzadas:	Caso de Uso: Procesar Venta
Precondiciones:	Hay una venta en curso
Postcondiciones	• Se creó una instancia de LineaDeVenta ldv • ldv se asoció con la Venta actual • ldv se asoció con una EspecificaciónDelProducto, en base a la coincidencia del articuloID • ldv.cantidad pasó a ser cantidad

Contrato CO3: finalizarVenta

Operación:	finalizarVenta()
Referencias cruzadas:	Caso de Uso: Procesar Venta
Precondiciones:	Hay una venta en curso
Postcondiciones	• Venta.esCompleta pasó a ser verdad

Contrato CO4: realizarPago

Operación:	realizarPago(cantidad:Dinero)
Referencias cruzadas:	Caso de Uso: Procesar Venta
Precondiciones:	Hay una venta en curso
Postcondiciones	• Se creó una instancia de Pago p • p se asoció con la Venta actual • La Venta actual se asoció con la Tienda (para añadirlo al registro histórico de las ventas completadas) • p.cantidadEntregada pasó a ser cantidad

3. Glosario

En su forma más simple, el Glosario es una **lista de los términos relevantes y sus definiciones**. Es sorprendentemente habitual que un término, frecuentemente técnico o propio del dominio, se utilice de forma ligeramente distinta por diferentes personas involucradas; esto tiene que resolverse para reducir los problemas de comunicación y los requisitos ambiguos. A continuación aparece un ejemplo de contenidos de un Glosario.

Término	*Definición e Información*	*Alias*
artículo	Un artículo o servicio en venta	
autorización de pago	Validación llevada a cabo por un servicio externo de autorización de pago, que hará o garantizará el pago al vendedor	
Solicitud de autorización de pago	Un compuesto de elementos enviados electrónicamente a un servicio de autorización, normalmente como un array de caracteres. Los elementos comprenden; ID de la tienda número de cuenta del cliente, cantidad y fecha.	
UPC	Código de 12 dígitos que identifica un artículo. Normalmente se representa mdiante un código de barras en los artículos. Diríjase a http://wwww.uc-council.org para ver más detalles.	Código de Producto Universal
…	…	…

El objetivo no es recoger todos los posibles términos, sino aquellos que no están claros, son ambiguos o que requieren algún tipo de elaboración relevante como el formato de la información o las reglas de validación.

Se recomienda comenzar el Glosario cuanto antes.

El Glosario juega también el rol de **diccionario de datos**, un documento que recoge los datos sobre los datos, es decir, metadatos. Si bien al comienzo del proyecto el glosario será un documento sencillo de términos y descripciones, que gradualmente irá evolucionando hacia algo más complejo. El Glosario podría contener elementos tales como:

- Casos de uso.
- Actores.
- Conceptos.
- Atributos y relaciones.
- Etc.

Los atributos de los términos podrían contener:

- Categoría.

- Descripción.
- Formato (tipo, longitud, unidad).
- Relaciones con otros elementos.
- Rango de valores.
- Reglas de validación.

El Glosario no está destinado sólo a términos atómicos como el “precio del artículo”. Puede y debe incluir términos compuestos, como “venta” (que incluye otros elementos tales como la fecha y la ubicación), y alias utilizados para describir una colección de datos que se transmiten entre los actores en los casos de uso.

4. Paquetes

Un modelo del dominio puede crecer fácilmente y llegar a ser lo suficientemente amplio para que sea conveniente dividido en paquetes que incluyen conceptos fuertemente relacionados. Esto sirve de ayuda para mejorar la comprensión y para abordar trabajo de análisis en paralelo, en el que diferentes personas realizan el análisis del dominio en diferentes subdominios. A continuación se ilustra la estructura de paquetes para el Modelo del Dominio del UP.

Un paquete en UML se representa mediante una carpeta (ver Figura 8). Podrían mostrarse dentro de un paquete otros paquetes subordinados. Si el paquete describe sus elementos, el nombre del paquete se coloca en la etiqueta; en otro caso, se centra en la propia carpeta.

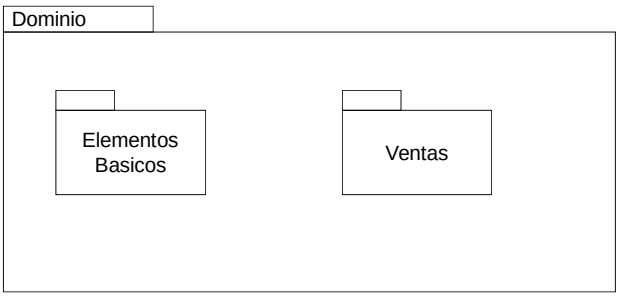

Figura 8. Un paquete UML.

Un elemento **pertenece** al paquete donde está definido, pero podría ser **referenciado** en otros paquetes. En ese caso, el nombre del elemento se califica con el nombre del paquete utilizando el formato del nombre de camino *NombrePaquete::NombreElemento* (ver Figura 9). Una clase que se muestra en un paquete que no es al que pertenece se podría modificar con nuevas asociaciones, pero por lo demás permanece sin alterar.

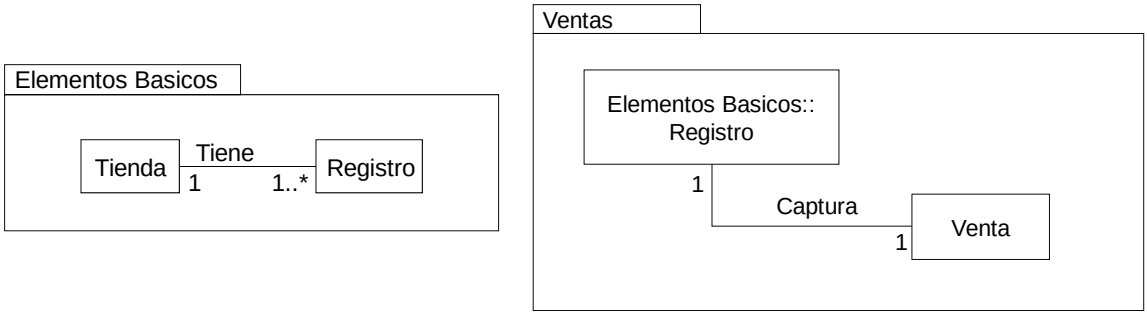

Figura 9. Una clase referenciada en un paquete.

Si un elemento del modelo **depende** de algún modo de otro, se podría representar la dependencia con una relación de dependencia, descrita por una línea con punta de flecha. Una dependencia entre paquetes indica que los elementos del paquete dependiente conocen o están acoplados de algún modo con los elementos del paquete destino.

Por ejemplo, si un paquete referencia a un elemento que pertenece a otro, existe una dependencia. Por tanto, el paquete de *Ventas* tiene una dependencia con el paquete *Elementos* Basicos (ver Figura 10).

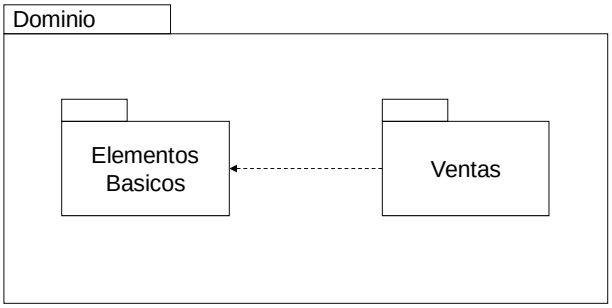

Figura 10. Dependencia entre paquetes.

A veces, no es conveniente dibujar un diagrama de paquetes, pero no obstante es deseable indicar el paquete al que pertenecen los elementos. En esta situación, se incluye una nota (un rectángulo con la esquina doblada), como! ilustra en la Figura 11.

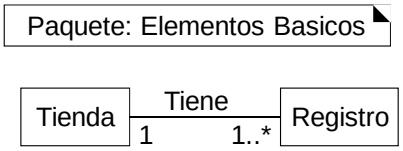

Figura 11. Representación de la pertenencia a un paquete con una nota.

A la hora de organizarse en paquetes las clases del modelo del dominio se aplican las siguientes guías generales. Se deben poner juntos los elementos que:

- Se encuentran en el mismo área de interés -estrechamente relacionados por conceptos u objetivos.
- Están juntos en una jerarquía de clases.
- Participan en los mismos casos de uso.
- Están fuertemente asociados.

Resulta útil que todos los elementos relacionados con el modelo del dominio tengan como raíz un paquete denominado *Dominio*, y todos los conceptos básicos, comunes, compartidos, se definan en un paquete que se puede llamar algo así como *Elementos Básicos* o *Conceptos Comunes*, en ausencia de cualquier otro paquete significativo en el que colocarlos.

En base al criterio anterior, la organización de paquetes para el Modelo del Dominio del PDV se muestra en la Figura 12, Figura 13, Figura 14 y Figura 15.

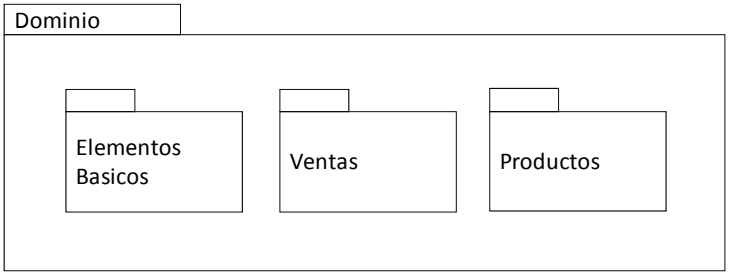

Figura 12. Paquetes de conceptos del dominio.

En este ejemplo, es conveniente que exista un paquete Basicol/Misc (ver Figura 13). Este paquete contiene conceptos ampliamente compartidos o aquellos sin una ubicación obvia.

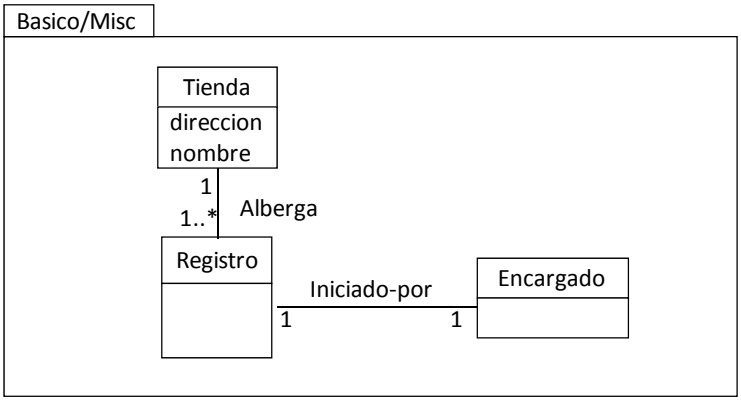

Figura 13. Paquete básico.

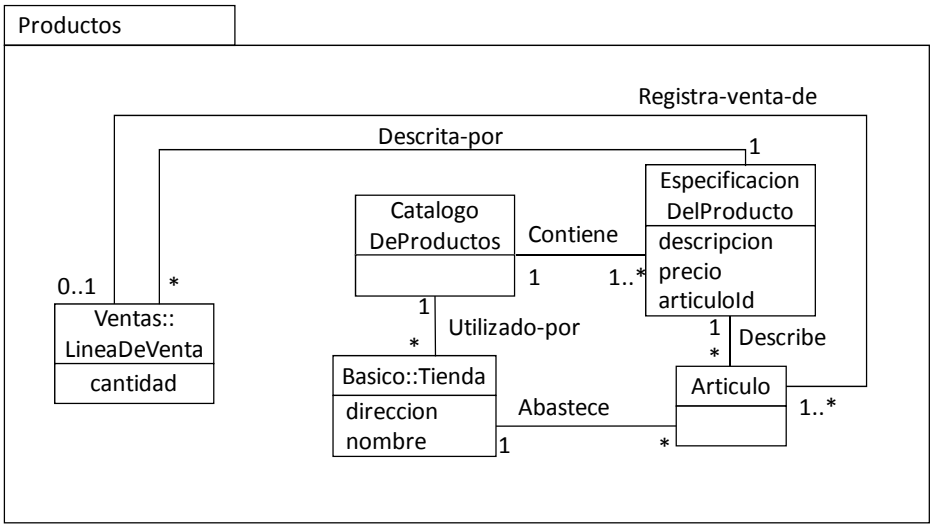

Figura 14. Paquete de productos.

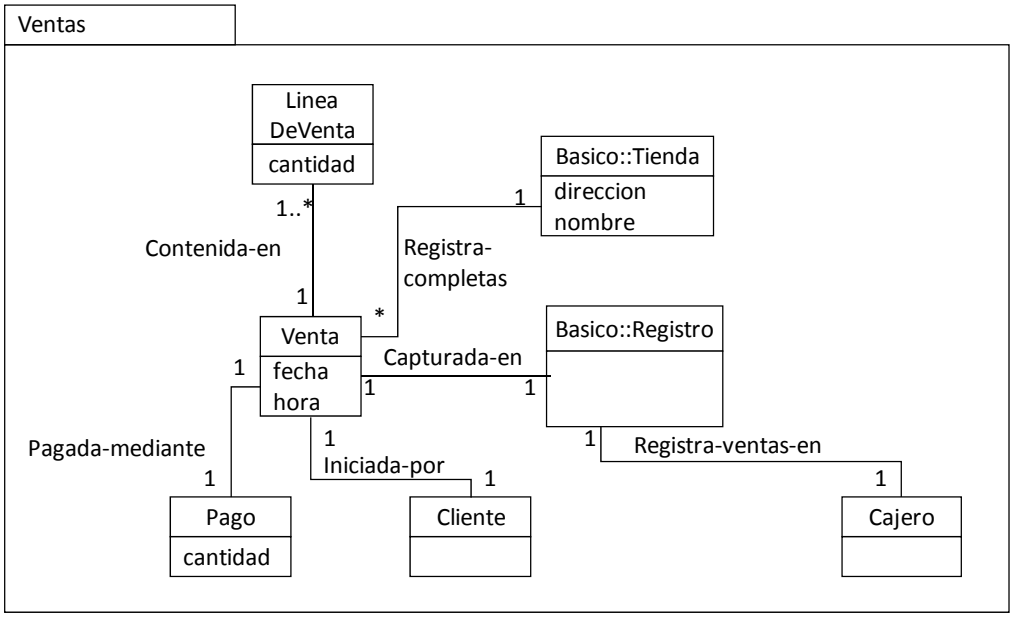

Figura 15. Paquete de ventas.