
# Ingeniería software

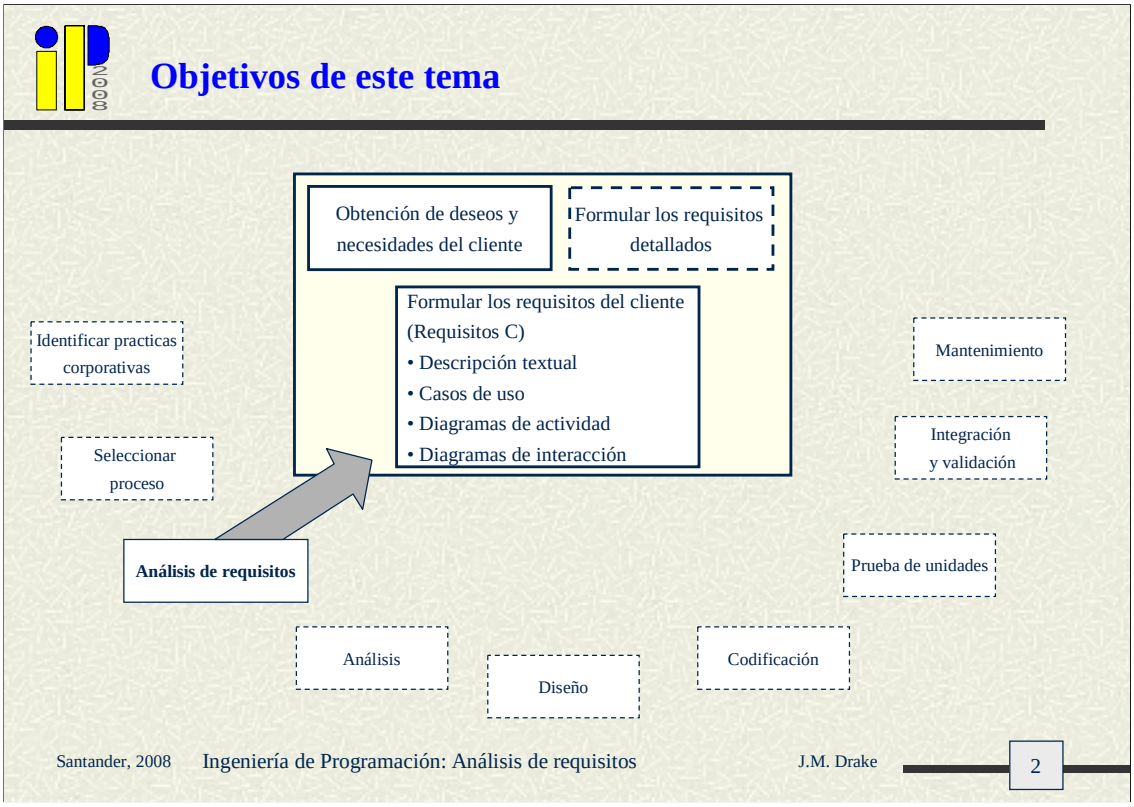
**4º de Físicas**

# Análisis de requisitos y especificación de una aplicación

José M. Drake
Computadores y Tiempo Real

Santander, 2008

1




El análisis global de los requisitos de una aplicación es un proceso de conceptualización y formulación de los conceptos que involucra de forma concreta. Es una parte fundamental del proceso de desarrollo de una aplicación, la mayor parte de los defectos encontrados en el software entregado se originan en la fase de análisis de requisitos, y además son los mas caros de reparar.

Siempre se ha discutido quién es el dueño de los requisitos: el cliente o el desarrollador. Para gestionar esto, es habitual presentar el análisis de requisitos en dos secciones:

•Requisitos de cliente: documentan los deseos y necesidades de los clientes y se expresan en lenguaje claro para él.

•Requisitos detallados: Determina los requisitos de manera específica y estructurada y están destinadas específicamente hacia los desarrolladores.

Los objetivos de este tema son:

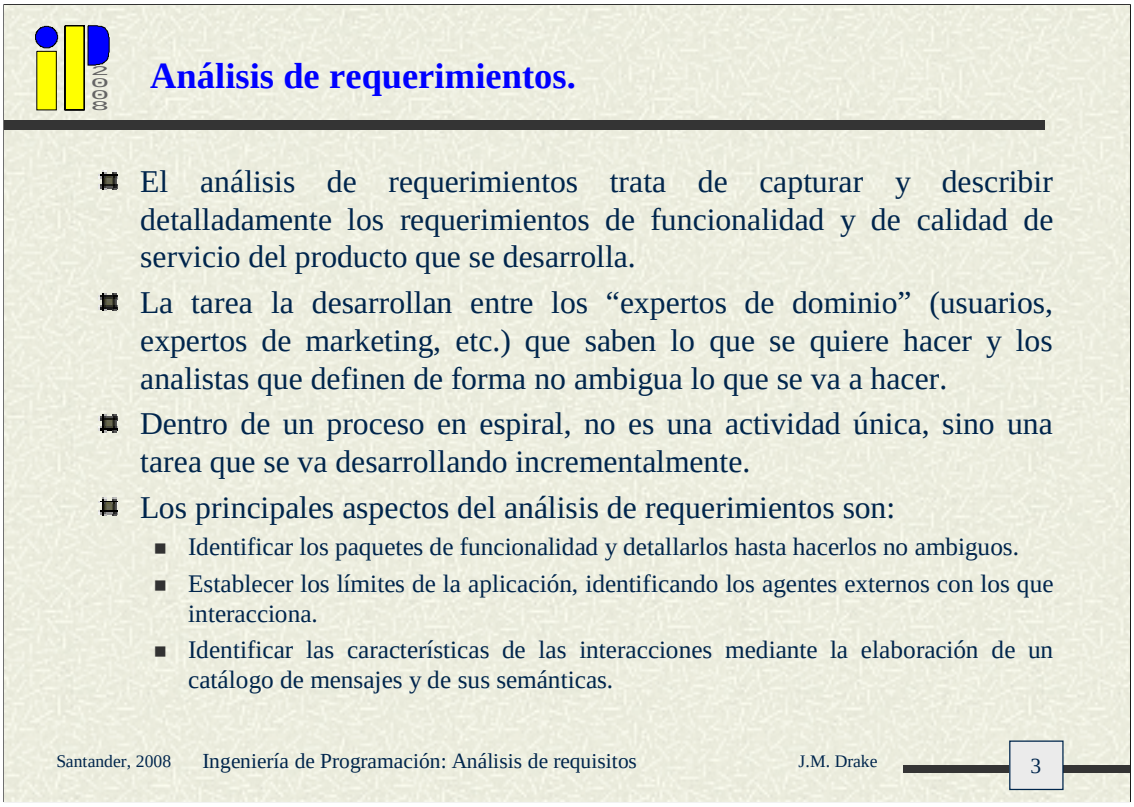
•Distinguir entre requisitos de clientes y requisitos detallados.

•Disponer de recursos para formular de forma clara y sistemática los requisitos del cliente.

- Casos de uso
- Diagramas de actividad
- Diagramas de interacciones, colaboraciones y flujo de datos.
- Descripción de las interfaces de usuario y sus protocolos de uso.

•Ser capaz de describir los documentos de la especificación de requisitos de software.

El resultado del proceso es el documento “Especificación de Requisitos Software”




Para construir *algo* primero debe entenderse lo que debe ser ese *algo*. El proceso de entender y documentar una aplicación software se llama “Análisis de requisitos”. En general los requisitos expresan *qué* se supone debe hacer una aplicación y no intentan expresar *como* logra estas funciones.

El análisis inicial de un sistema debe tratar de descubrir los requerimientos del producto final que se desarrolla en detalle.

Unos de los principales objetivos de UML es hacer que este análisis sea lo suficientemente intuitivo para que los clientes y expertos en el dominio que solicitan el producto puedan comprenderlo, y lo suficientemente formal y riguroso para que se establezca una formulación no ambigua que pueda ser utilizada por los técnicos que la desarrollan.

Los aspectos básicos que deben tratarse en esta fase son:

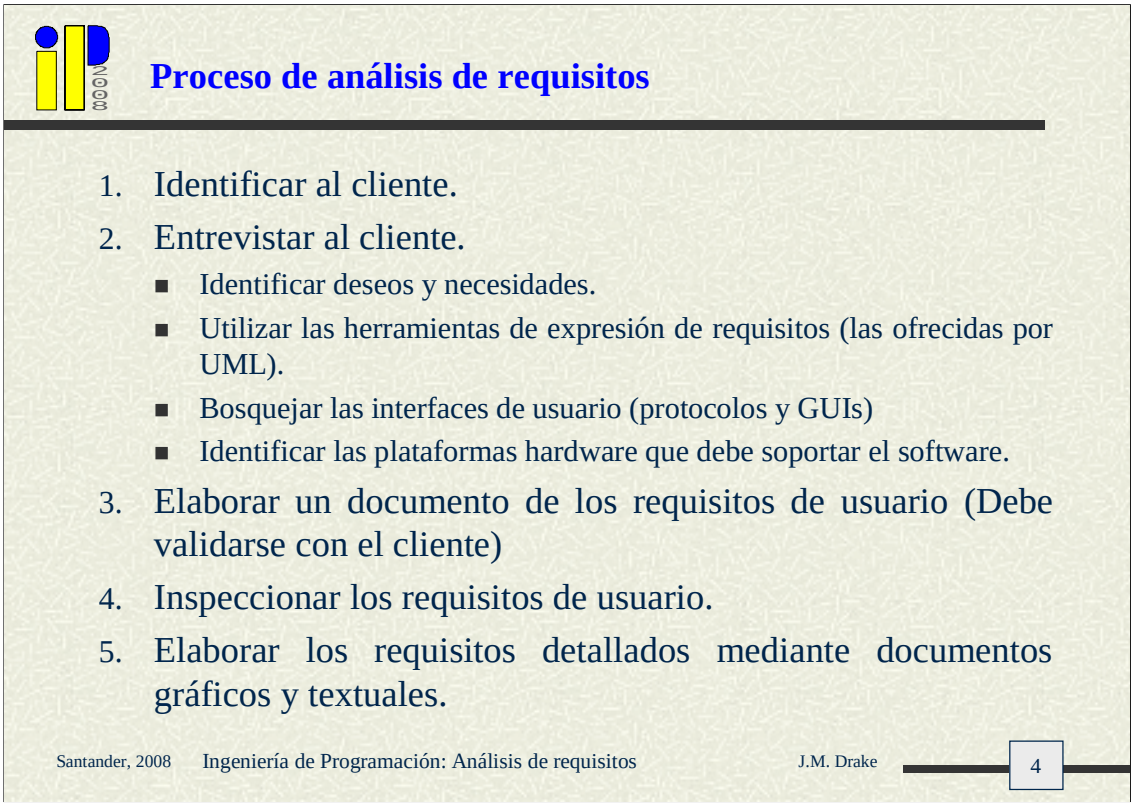
•Determinar los paquetes de funcionalidad y de la calidad de servicio del producto, formulados de una forma independiente de su implementación, y refinar y detallar estas especificaciones hasta que den lugar a una especificación no ambigua del producto que se desarrolla.

•Identificar los actores externos al sistema que interactúan con la aplicación de forma relevante.

•Identificar la semántica y las características de los mensajes que intercambian los actores con el sistema que se desarrolla.

•Refinar los protocolos de interacción que usan los actores para llevar a cabo las diferentes transacciones que se pueden realizar con el sistema.

El análisis de requisitos es una necesidad, no un un lujo. Para apoyarlo considérese su efecto sobre las pruebas del producto concluido. Si alguien le proporciona una caja negra con un cable rojo, rosa y morado que sale de ella, sería imposible probarlo. No se sabe que hace, para que sirve.




El proceso de análisis de requisitos requiere diferentes actividades de alto nivel y que son desarrollados por múltiples agentes (usuarios, expertos de dominio, expertos de marketing, programadores, etc.)

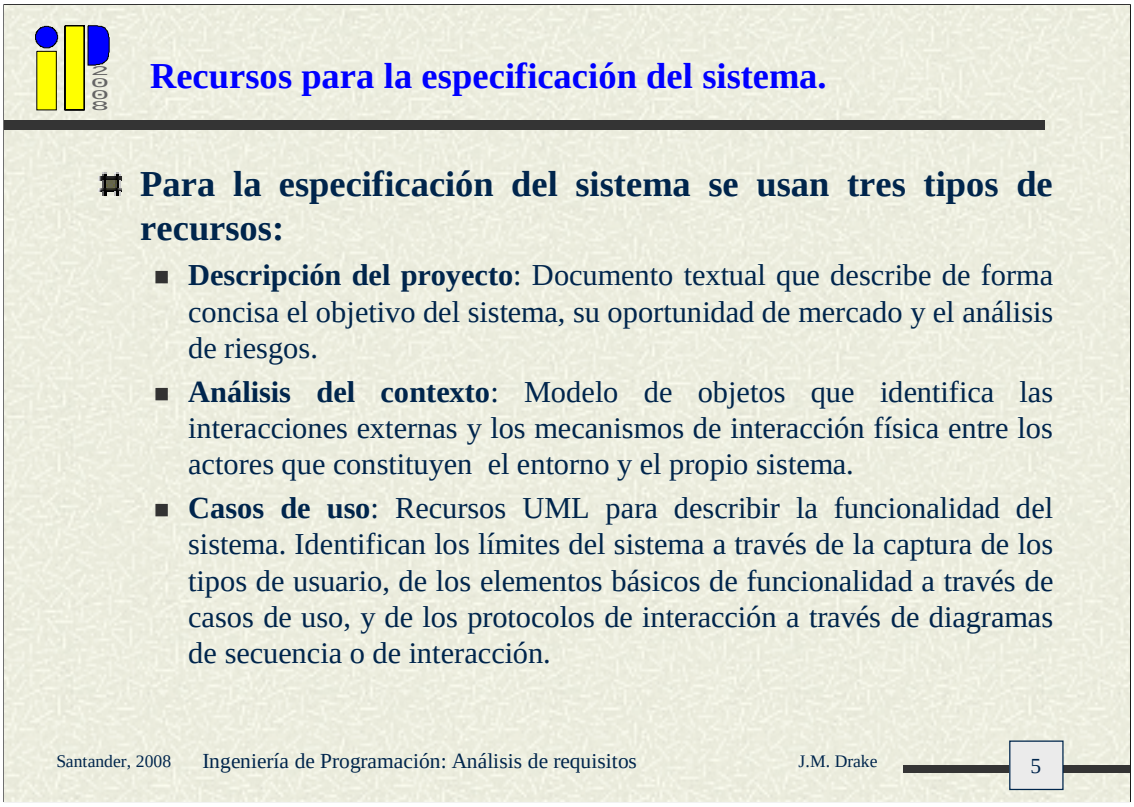
Para la formulación de las especificaciones existen diferentes estándar, el mas conocido es el estándar ANSI, IEEE 830-1993.

Para todas las etapas se pueden utilizar métricas tales como:

- Tiempo dedicado a su análisis.
- Cantidad producida (páginas de requisitos, minutos de interacción con el cliente, etc.)
- Calidad deducida de la autoevaluación (Tasas de defectos en las inspecciones).

**Estándar ANSI IEEE 830-1993**

1. Introducción
   - Propósito.
   - Definiciones acrónimos y abreviaturas
   - Panorama
   - Alcance
   - Referencias
2. Descripción general
   - Perspectiva del producto:(Interfaces del sistema, del usuario, hardware, software, de comunicación, restricciones de memoria, operaciones, requisitos de adaptación)
   - Funciones del producto
   - Restricciones
   - Distribución de requisitos
   - Características del usuario
   - Suposiciones y dependencias
3. Requisitos específicos (Se desarrollará mas adelante)
4. Información de apoyo.



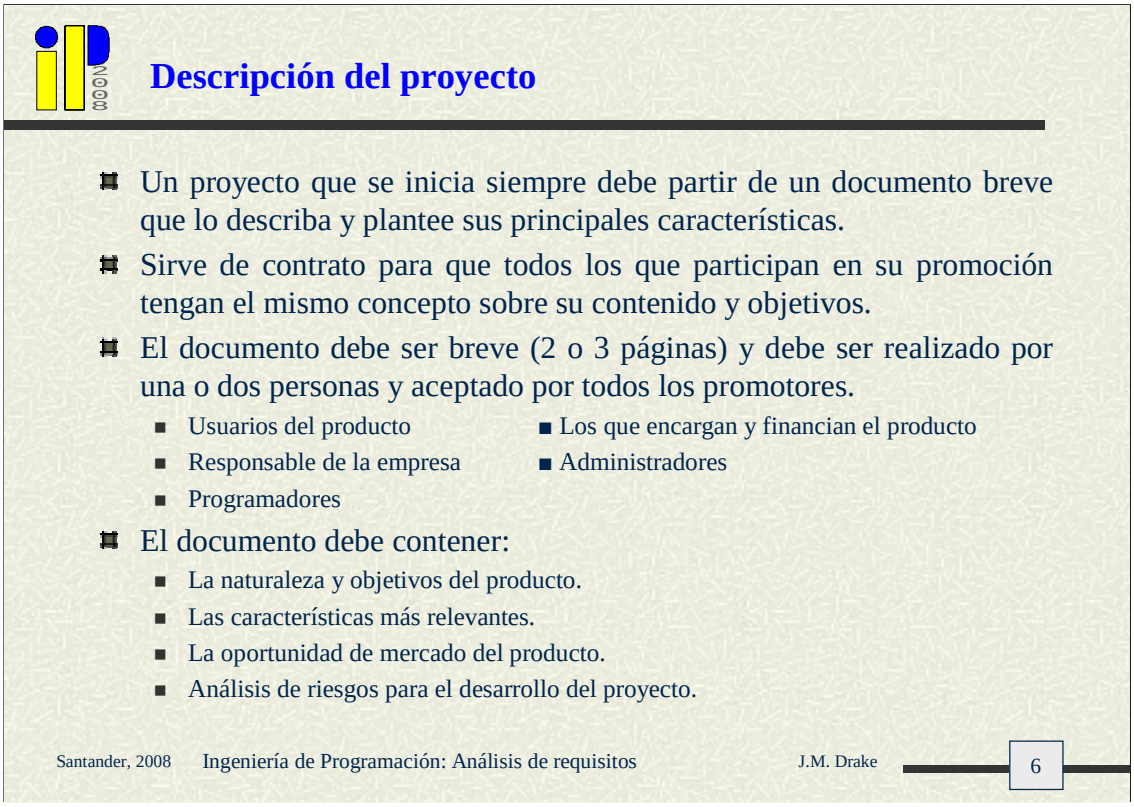

Los sistemas interactúan con su entorno externo (operadores, usuarios, otros sistemas, dispositivos, etc.) y la funcionalidad básica que tienen que ofrecer debe formularse en función de este contexto y con independencia de la forma en que se construyen internamente.

Existen tres vías que pueden utilizarse para realizar la formalización de los requerimientos:

•Descripción del proyecto: Es un paso previo que aunque es obvio tiene una gran importancia, y consiste en generar un documento que de forma concisa resuma la información inicial relativa al proyecto que se inicia. En él debe incluirse la naturaleza y objetivo del proyecto, las características mas relevantes, su oportunidad de mercado, y un análisis de los riesgos que conlleva. Debe ser un documento breve con solo dos o tres páginas, pero que establece un punto de arranque en el que los diferentes responsables de su ejecución (clientes, expertos de dominio y desarrolladores) tienen el mismo concepto sobre lo que se desarrolla.

•Análisis del contexto: Trata de especificar la funcionalidad del sistema a través de la descripción de las interacciones que se pueden producir entre el sistema y el entorno externo. Se formula como diagramas de objetos en los que el sistema aparece como una caja negra sobre la que se identifican los elementos de interacción (sensores y actuadores) y también se identifican los actores externos que interactúan con él, así como los tipos de mensajes que se producen definiendo su semántica y la información que transmiten.

•Casos de uso: Es el recurso específico de UML para describir la funcionalidad y las características de calidad de servicio del sistema. Se basa en identificar los límites del sistema a través de la captura de los actores, de los elementos básicos de funcionalidad a través de casos de uso, y de los protocolos de interacción a través de diagramas de secuencia o de interacción.




A las personas que tienen intereses en el producto que se desarrolla, le denominamos *interesados (stakeholders):*
•Usuarios del producto: son aquellos que van a utilizar directamente el producto como operarios.
•Los que encargan o financian el producto
•Los directivos de la empresa en que se desarrolla el producto.
•Los administradores.
•Los desarrolladores siempre que utilicen la aplicación como forma de actualizar su tecnología.

Cada requisito debería ser:
•Expresado en el lenguaje del usuario.
•Incluido en una lista organizada que facilite su localización y acceso sea sencillo.
• Estar numerado y ser referenciado en el código.
•Ser verificable de forma aislada e ir acompañado de pruebas que posibiliten su verificación.

# Ejemplo de descripción del producto.

**Descripción del producto**

Se quiere desarrollar un software de procesamiento de órdenes de compra para una empresa llamada WEB-Cántabra, que es una revendedora o intermediadora de un conjunto de productos procedentes de diferentes empresas suministradoras.

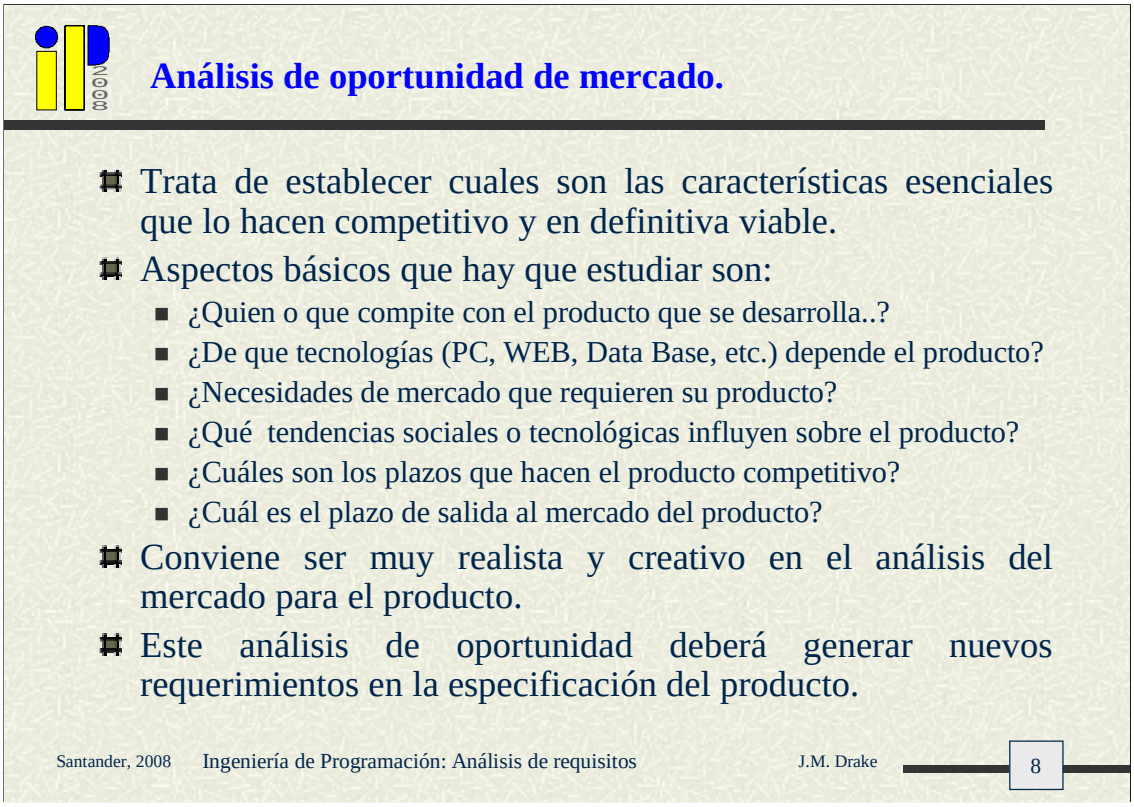
La empresa publica dos veces al año un catálogo de los productos que vende, que es distribuido entre los clientes y otras personas que pueden estar potencialmente interesadas.

**Requerimientos adicionales**

Los clientes encargan los productos, enviando una lista de ellos junto con las órdenes de pago a favor de WEB-Cántabra. WEB_Cántabra cumplimenta las ordenes de pedido y envía los productos a la dirección de los clientes.

La aplicación software debe mantener la información completa de las órdenes desde que el pedido llega, hasta que el producto es enviado.

WEB-Cántabra debe proporcionar un servicio rápido. Debe ser capaz de servir el pedido del cliente por el medio más rápido y eficiente posible.




El proceso de análisis de requisitos debe contener un análisis de factibilidad y un análisis de oportunidad de mercado. El resultado debe ser un informe que recomienda si es conveniente llevar a cabo la ingeniería de requerimientos y el proceso de desarrollo del sistema.
Un estudio de factibilidad es un estudio corto, que responde a si el producto contribuye a los objetivos de negocio de la empresa. Debe responder a varias preguntas:
•¿Corresponde el producto a los objetivos generales de la empresa?
•¿Se puede utilizar el producto haciendo uso de la tecnología actual y con las restricciones de tiempo y costo?
•¿Puede integrarse el sistema a otros que existen en la organización?
Un estudio de factibilidad comprende la evaluación y recolección de la información y a la elaboración de los informes.

## Ejemplo de análisis de oportunidad de mercado.

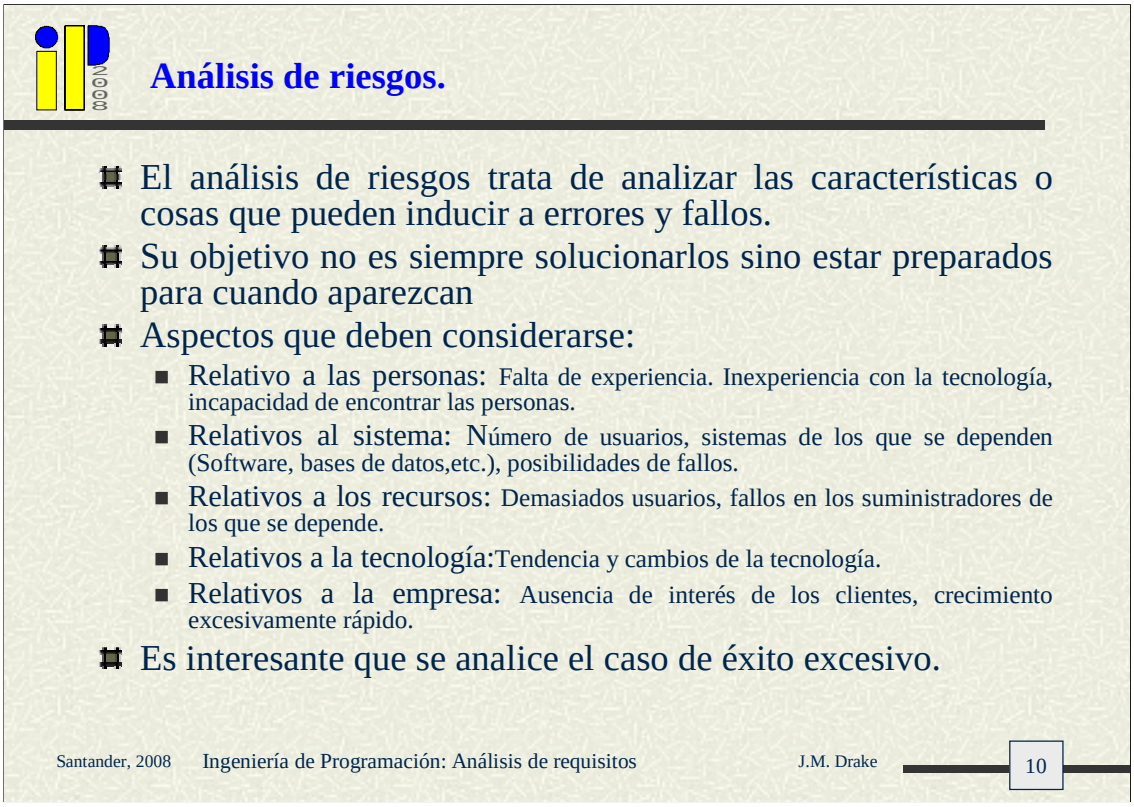
**Análisis de oportunidad del mercado**

La mayoría de los responsables familiares trabajan al menos a tiempo parcial, no tienen tiempo para la compra de productos de consumo familiar y le sería muy favorable la compra y la recepción de los productos desde el domicilio y con horario muy abierto.

La compra a través de WEB y de la Red es cada vez mas popular y está tomando un segmento del 25% en este sector del mercado.

Las empresas que actualmente proporcionan este servicio suministran el producto con plazos que varían entre 2 y 10 días, sin que el cliente tenga capacidad de comprobar el estado de ejecución de su pedido.

Las empresas basan su marketing en ofertas de regalos y de bonos acumulativos de descuento.

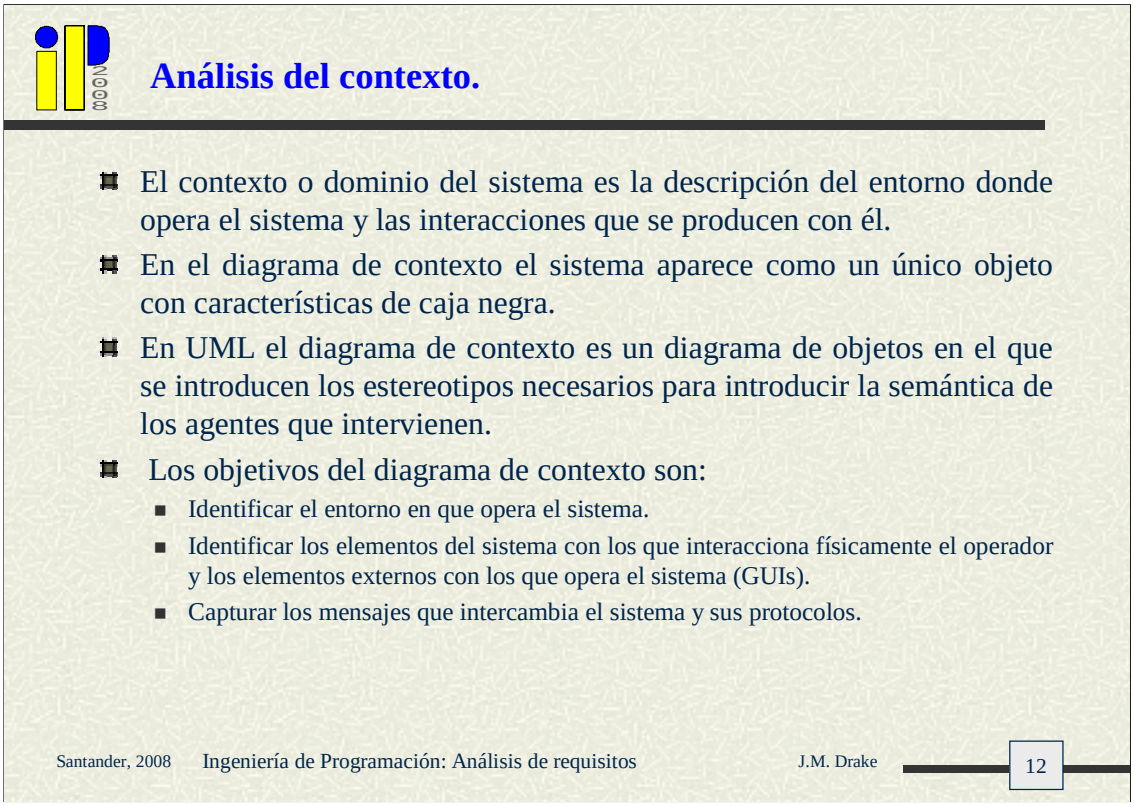



Se necesita considerar los factores de riesgo en el proyecto. Se necesita incluir las cosas que pueden conducir a equivocaciones. Describir solo las características que nos gustarían que ocurrieran nos llevan directamente al desastre. Es muy importante tener en cuenta que cosas pueden ir mal y planificar el desarrollo del proyecto para que no ocurran.
También hay que prever la posibilidad de que el proyecto tenga un éxito salvaje que nos puede sobrepasar y conducir también al desastre.

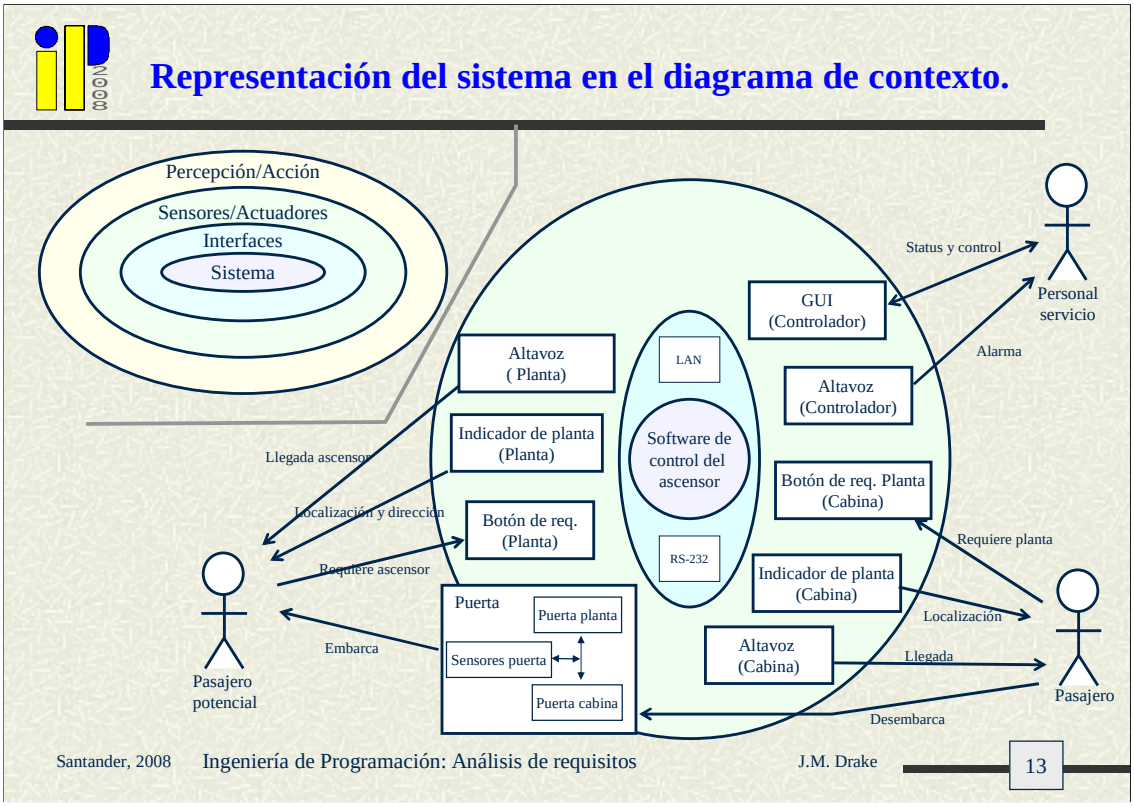
## Ejemplo de análisis de riegos.

**Análisis de riesgos**

- Una de las tres personas que van a desarrollar el software es su primer trabajo profesional.
- ¿Cómo podemos prever la perdida de pedidos de clientes por fallo del sistema?
- Un gran sector de los clientes son personas no expertas en medios informáticos, por lo que el sistema tiene que ser muy fácil de utilizar.
- Si el sistema es inmediatamente inundado de ordenes de pedido ¿Que ocurrirá?.
- ¿Cómo resolver el caso de un fallo de la base de datos?




El contexto del sistema es un mapa del mundo que es de interés para el sistema. Esta herramienta es directamente heredada de las estrategias estructuradas.
El término sistema representa a todo aquello que es objeto del diseño y se muestra con características de caja negra. Incluye todo el sistema bajo desarrollo: el software, el hardware eléctrico y mecánico. En el diagrama de contexto se representa como un simple elemento (el sistema), rodeado de múltiples elementos que constituyen el entorno donde opera.
UML no tiene un recurso específico para representar los diagramas de contexto, por ello, se utiliza un diagrama de objetos o diagrama de colaboración, en el que la semántica de los objetos se explicita mediante estereotipos.
El beneficio de utilizar los diagramas de contexto es que se captura el contexto en que opera el sistema, incluyendo los actores con los que interacciona el sistema. Así mismo también ayuda a identificar y caracterizar los mensajes y eventos que fluyen entre el sistema y su entorno.




El diagrama de contexto muestra al sistema interactuando con uno o mas agentes externos. En el objeto sistema es útil identificar los sensores y actuadores a través de los que los agentes externos operan con el sistema.
En la figura se muestra como guía la representación del sistema en cuatro capas:
•La capa mas externa representa las percepciones que el usuario o el entorno tiene del sistema y describe lo que se ve desde fuera o lo que hace el sistema sobre el entorno.
•La siguiente capa describe la interfaz física, esto es, los elementos de visualización, señalización los elementos de control del sistema (sensores) y los elementos de actuación del sistema sobre el entorno (actuadores).
•La tercera capa representa las interfaces eléctricas con las que se conectan los sensores y actuadores con el sistema.
•La última capa es el sistema que se desarrolla representado como una caja negra.

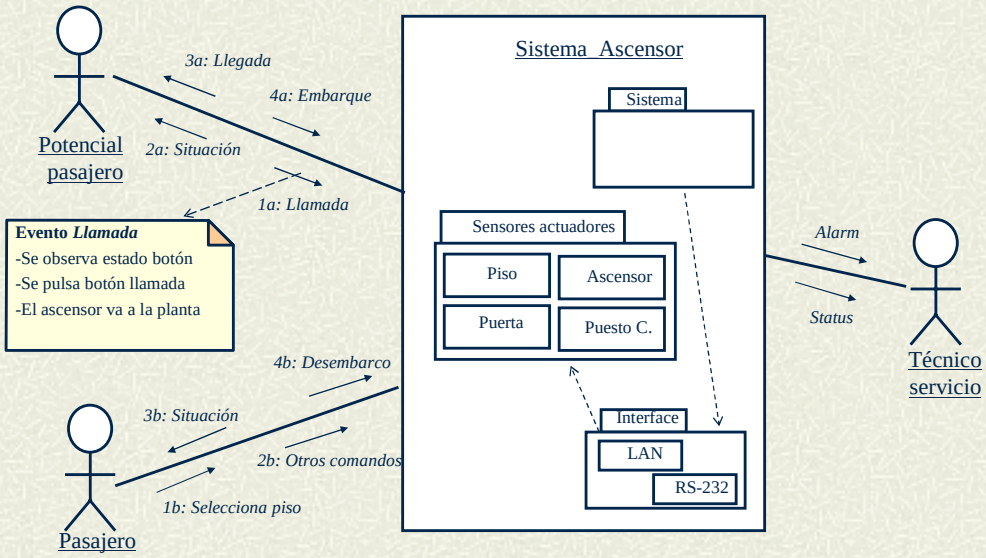
## Objetos del contexto en el ejemplo ascensor

| Percepción/ /Acción | Sensor/Actuador | Interfaz | Sistema |
|---|---|---|---|
| Pasajero sale del ascensor, obstruyendo la puerta | Puerta<br>Sensor puerta | Ethernet LAN | Ascensor |
| Un pasajero requiere el ascensor | Botón de llamada del ascensor | Ethernet LAN | Ascensor |
| El pasajero selecciona el piso | Botón de piso | Ethernet LAN | Ascensor |
| El pasajero mantiene la puerta abierta sujetandola | Puerta | Ethernet LAN | Ascensor |
| Pasajero espera que el ascensor llegue al piso. | Display de piso | Ethernet LAN | Ascensor |

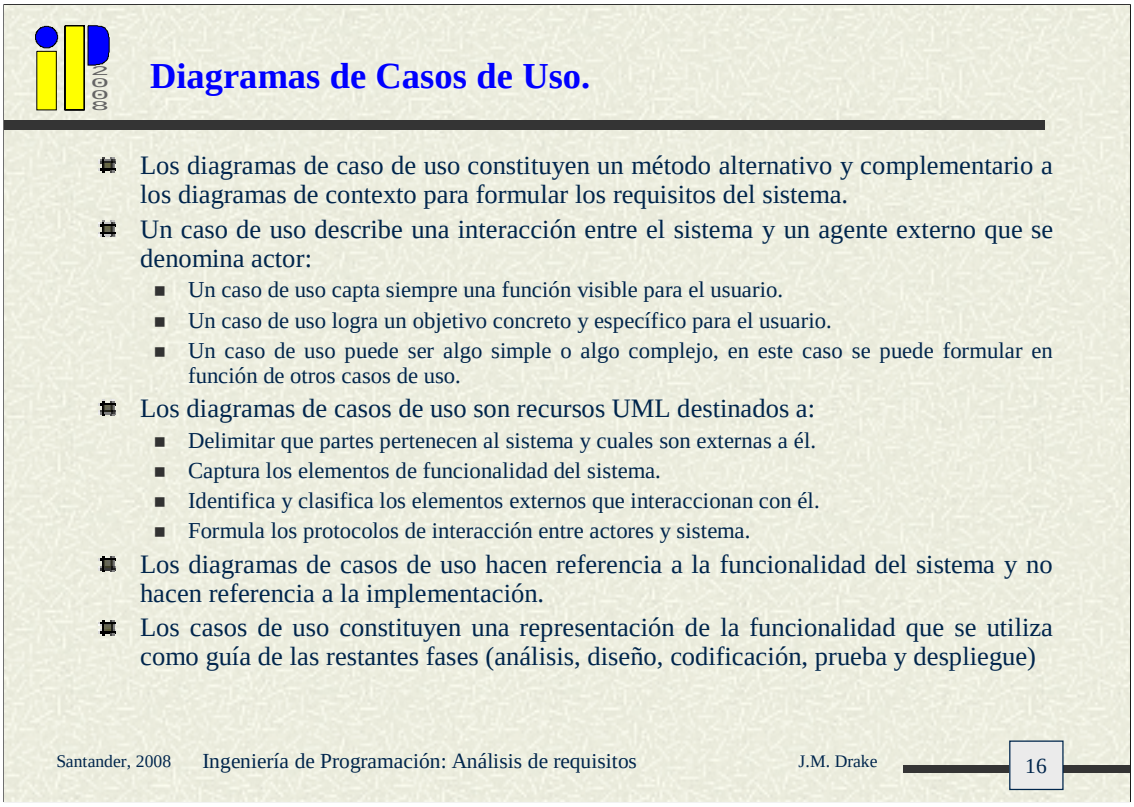
Santander, 2008 Ingeniería de Programación: Análisis de requisitos J.M. Drake 14

Los objetos de las dos columnas intermedias pueden ser considerados como perteneciente al sistema o al entorno en función del ámbito de la aplicación.
Se considerarán externos al sistema cuando se produzcan alguna de las siguientes condiciones:
•En la especificación del problema explícitamente se formulan como externos.
•Los objetos ya está diseñados con anterioridad del sistema.
•Los objetos son suministrados con independencia del sistema.
•Los objetos son una persona o equipo externo que introduce o obtiene información del sistema.

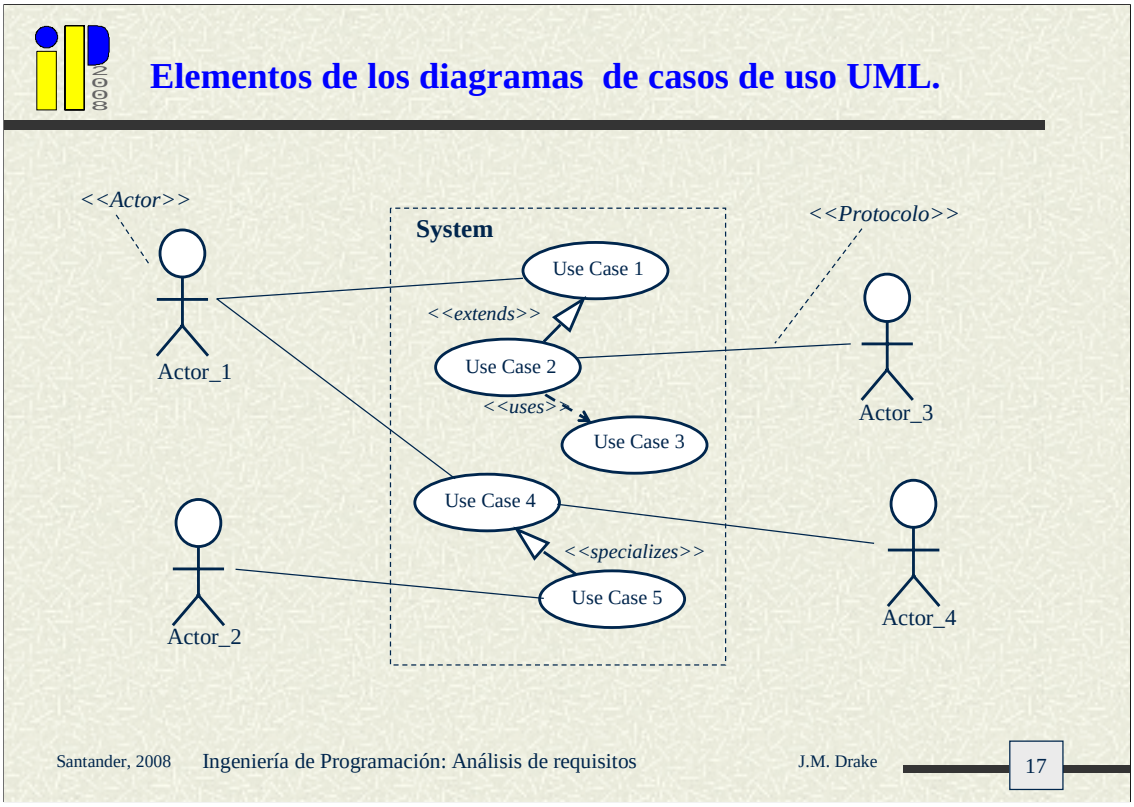
## Ejemplo: Captura de contexto y protocolo




Santander, 2008 Ingeniería de Programación: Análisis de requisitos J.M. Drake 15




Los diagramas de casos de uso son un método alternativo y complementario a los diagramas de contexto como medio de especificar los requisitos de una aplicación software.
Jacobson creó la idea de los casos de uso al observar que, a pesar del gran número de ejecuciones potenciales, muchas aplicaciones están concebidas en términos de un número relativamente pequeño de interacciones típicas.
Muestran los tipos básicos de interacción entre el sistema y los elementos del entorno que operan con él. Los diagramas de casos de uso proporcionan un recurso alternativo bien para verificar el diagrama de contexto o de ayuda para construirlo.
Los casos de uso capturan una vista general de la funcionalidad del sistema con un método muy adecuado para ser interpretado por personas no técnicas como son los usuarios y los expertos de dominio. Suelen interpretarse como una guía de los escenarios de uso del sistema sobre los que se especifican los requerimientos del sistema.
Los casos de uso son también un método de descomposición de la funcionalidad del sistema en elementos de funcionalidad más básica, ya que UML proporcionan una semántica para expresar los casos de usos mas generales en función de casos de usos mas simples.
Los casos de uso son contenedores de la información que describen los requerimientos del sistema. Estos se pueden formular con diferentes niveles de detalles, mas cualitativos para el uso de los expertos de dominio y mas detallado y formales para el uso de los analistas de la aplicación.



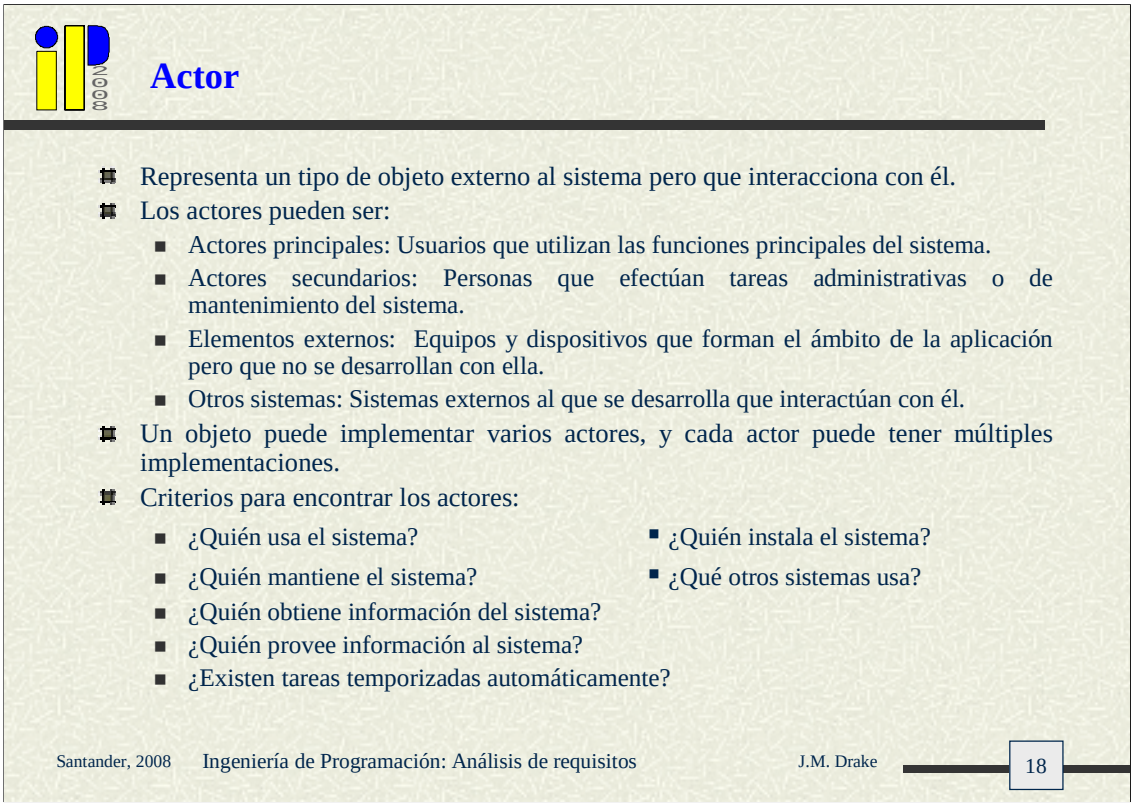

El modelo de casos de uso de una aplicación se compone de actores, el sistema (como una caja negra) y los propios casos de uso. La funcionalidad de un sistema se formula examinando las necesidades funcionales que requiere del sistema de cada actor, expresadas en forma de familias de interacciones que se incluyen en un caso de uso.

Un actor representa un papel interpretado por una persona u otro sistema que interacciona con el sistema que se describe. La misma persona física puede representar diferentes papeles y así mismo, muchas personas pueden ser representadas por un mismo actor. Los actores deben ser descritos de una forma clara mediante un texto de sólo algunas líneas.

Los casos de uso se determinan observando y precisando, actor por actor, las secuencias de interacción (que llamaremos escenarios) que llevan a cabo con el sistema. Los casos de uso agrupan a familias de escenarios que desempeñan un mismo papel funcional visto desde el usuario. Los casos de uso son abstracciones del diálogo entre los actores y el sistema: describen interacciones potenciales, sin entrar en los detalles de cada usuario.

Los escenarios son instancias de los casos de uso. Los escenarios representan secuencias de mensajes entre objetos que colaboran para producir algún tipo de comportamiento del sistema. Los escenarios constituyen un método muy intuitivo de representar los detalles de los requerimiento, en particular los que implican restricciones temporales.

Los enlaces entre actores y casos de uso representan los protocolos, que constituyen el hilo de conductor de la secuencia de eventos que intercambian ambos.




Los actores son cualquier cosa que interacciona el sistema que se desarrolla, por ejemplo, personas, otros software, hardware, dispositivos, redes, almacenes de datos, etc.. Cada actor define un particular “role”. Cada entidad externa al sistema puede ser representado por uno o mas actores. Así, una persona física puede ser representada por varios actores , debido a que la persona juega diferentes papeles con relación al sistema. O varios objetos físicos pueden estar representados por un mismo actor, porque ellos mantienen una misma interacción en relación con el sistema.

Existen cuatro grandes tipos de actores:

•Los actores principales: Agrupan a alas personas que utilizan las funciones principales del sistema. En el caso del software de un cajero son los clientes que hacen uso de él.

•Los actores secundarios: Son las personas que hacen tareas administrativas o de mantenimiento. En el ejemplo de un cajero es p.e. la persona que recarga el cajero.

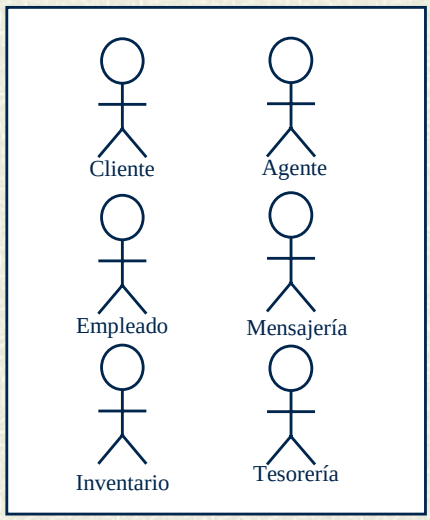
•Los Elementos externos: Agrupa a los dispositivos que forman parte del ámbito de la aplicación y que deben ser utilizado. En el ejemplo del cajero pueden ser p.e. la impresora.

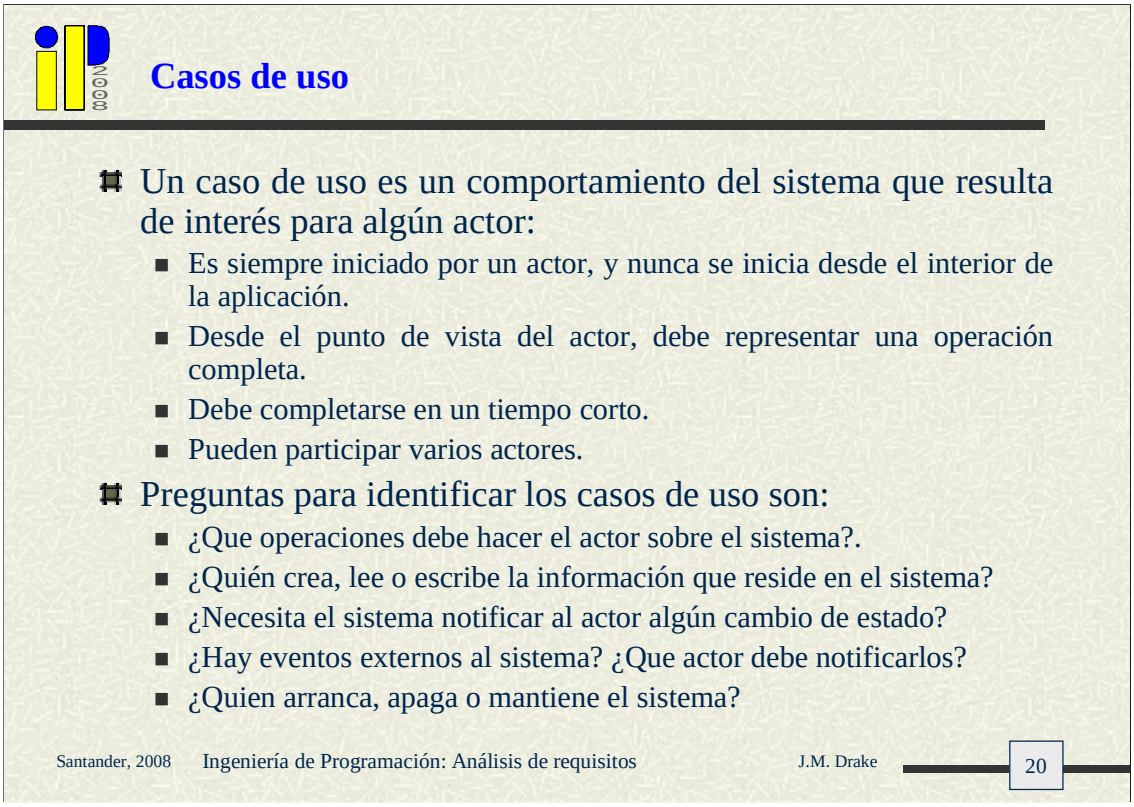
•Otros sistemas: Agrupa a los sistemas externos con lo que el sistema debe interactuar. En el ejemplo del cajero, un actor de este tipo puede ser el sistema del sistema bancario.

Los actores son siempre externos al sistema. Para tratar de localizar los actores, debemos buscar categorías de cosas que aparezcan como consecuencia de las siguientes preguntas: ¿Quien usa el sistema? ¿Quién instala, arranca y apaga el sistema? ¿Quien mantiene al sistema? ¿Que otros sistemas usa el sistema? ¿Quién obtiene información del sistema? ¿Quién provee información al sistema? ¿Ocurre algo de forma automática en tiempos prefijados?.

## Actores del ejemplo WEB-Cántabra.

- **Cliente**: Persona que realiza pedidos a WEB-Cántabra.
- **Agente**: Empleado de WEB_Cántabra que procesa los pedidos.
- **Empleado**: Empleado de WEB_Cántabra que empaqueta, etiqueta y envía los pedidos.
- **Mensajería**: Empresa que realiza los envíos.
- **Gestor del inventario**: Gestor de la base de datos de la empresa.
- **Tesorería**: Software que lleva las cuentas de los clientes y de la empresa.





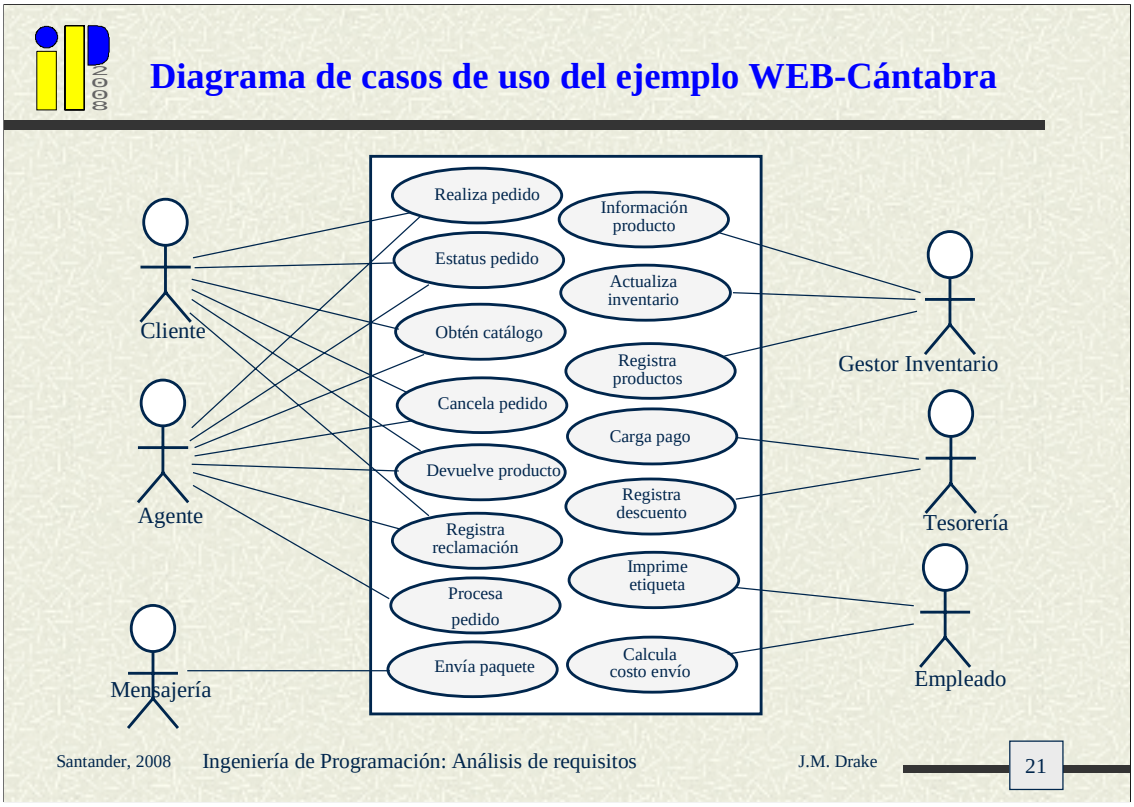

Un caso de uso es un comportamiento del sistema que produce un resultado de interés para algún actor. Los casos de uso describen cosas que los actores quieren que el sistema haga. Un caso de uso debe ser una tarea completa desde el punto de vista del actor, y debe corresponder a una tarea que se realiza en un tiempo relativamente breve,especialmente si debe ser realizado por múltiples actores. Cuando estas condiciones no se cumplen, es mejor definir varios casos de uso independientes.

Un caso de uso describe no solo una funcionalidad o una motivación, sino también una interacción entre un actor y el sistema bajo la forma de un flujo de eventos. La descripción que proporciona el caso de uso se refiere a lo que se espera que la interacción realice y no a como lo realiza.

Un caso de uso debe ser simple, y en caso contrario, se le debe fragmentar.

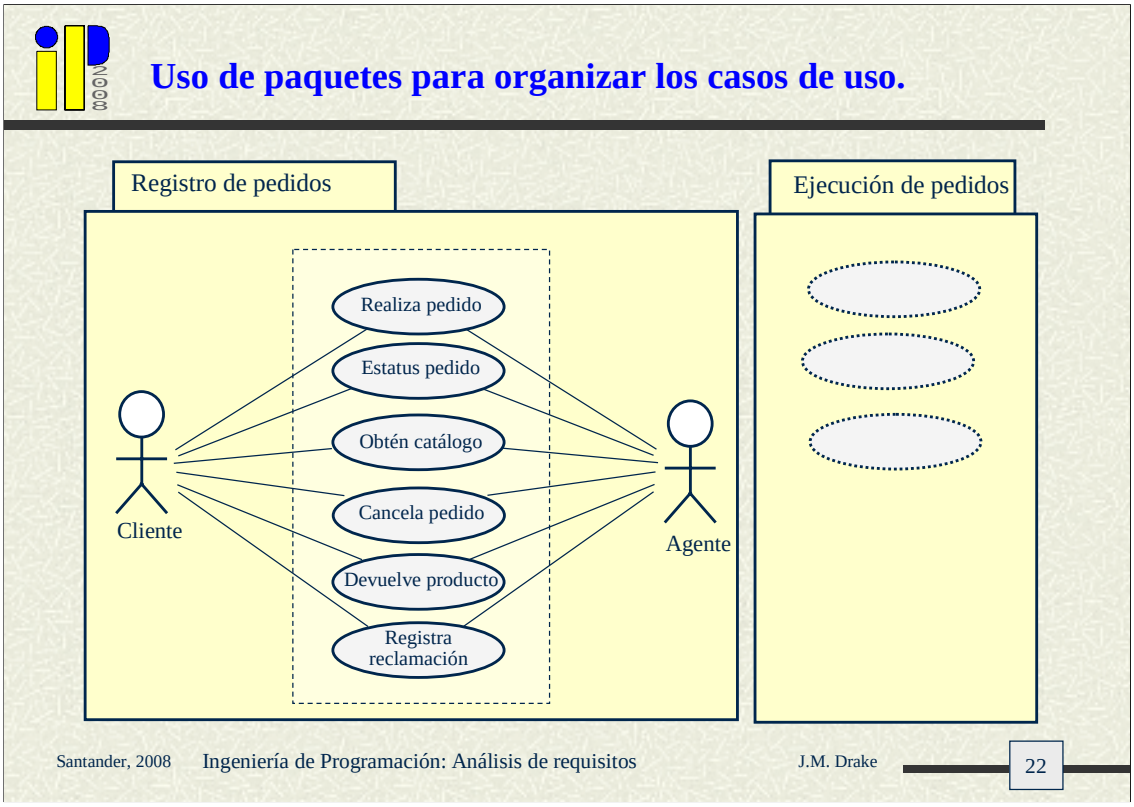
En UML los casos de uso son siempre iniciados por un actor, y no deben iniciarse espontáneamente desde el interior del sistema.

Es importante no solo considerar los casos de uso que describen la funcionalidad básica sino también considerar aquellos que describen las tareas de mantenimiento como, quien y como se arranca el sistema, se apaga, verifica el correcto funcionamiento de la instalación, etc. Todas estas funciones requieren prever una funcionalidad adicional que deben modelarse como casos de uso.




**Casos de uso**

**Realiza pedido**: Un cliente crea un pedido, selecciona los productos y ordena el pago
**Estatus pedido**: Un cliente requiere información del estado de su pedido.
**Obtén catálogo**: Un cliente requiere el catálogo de productos.
**Cancela pedido**:Un cliente da de baja un pedido ya registrado.
**Devuelve producto**: Un cliente devuelve un producto por fallo.
**Registra reclamación**: Un cliente envía un mensaje de reclamación a la empresa.
**Procesa pedido**: El agente procesa el pedido realizado por un cliente
**Envía paquete**: El sistema de mensajería informa de que envía un paquete
**Información del producto**: Informa el estado de un producto en el inventario.
**Actualiza inventario**: Se analiza el inventario y se realiza los pedidos a los suministradores.
**Registra producto**: Da de alta en el inventario un producto recibido de los suministradores.
**Carga pago**: Anota el pago relativo a un pedido.
**Registra descuento**: Anota en la cuenta de un cliente un descuento recibido.
**Imprime etiqueta**: Imprime la etiqueta de envío de un pedido.
**Calcula costo envío**: Calcula el gasto de envío de un pedido.

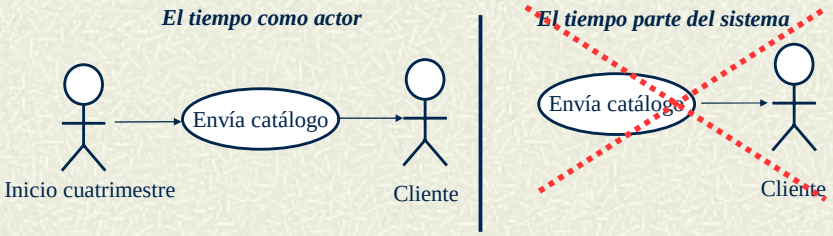


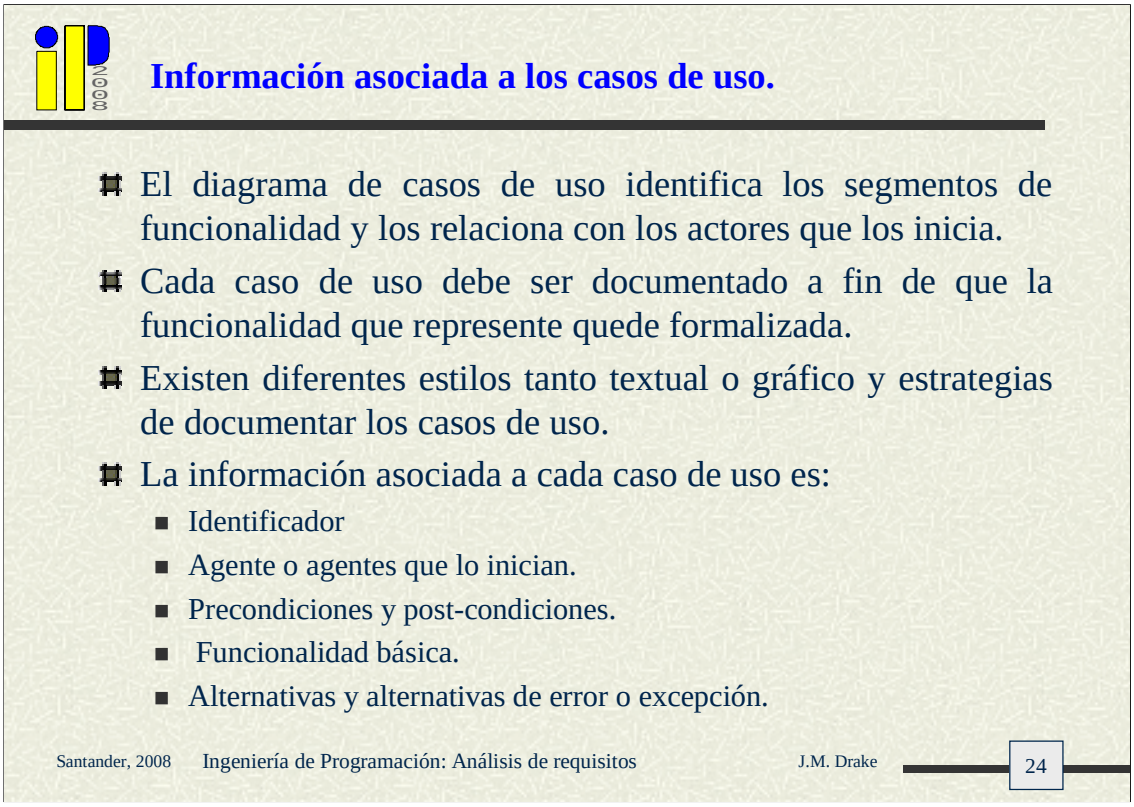

Si el diagrama de casos de uso que resulta es demasiado grande y enmarañado, se pueden crear diferentes diagramas de casos de uso y cada uno de ellos representaría una vista del diagrama general. Cada diagrama puede representar un área de principal de funcionalidad del sistema.
O la funcionalidad relativa a un actor, etc. Incluso puede ser conveniente que un mismo caso de uso a parezca en diferentes diagramas. En estos casos la utilización de una herramienta CASE es imprescindible para mantener la coherencia del modelo.
En sistema grandes puede ser útil organizar los diagramas en paquetes, cada uno de ellos representa la funcionalidad de un área o de un subsistema.

## Gestión de interacciones temporizadas.

- Ciertas actividades tienen lugar en instantes determinados de tiempo y son iniciadas desde el reloj interno.
- Hay dos formas de abordar esta situación:
  - El tiempo puede ser tratado como un actor que inicia el caso de uso.
  - El tiempo se considera parte del sistema y el actor que se relaciona con el caso de uso realiza la interacción motivado por ella.

Es preferible la primera situación ya que mantiene el principio de que los casos de uso se inician siempre por un actor.






Durante la fase de inicio del proyecto hay que decidir lo que hace el sistema que se desarrolla, y debe identificarse la funcionalidad desde un punto de vista conceptual y abstracto. Durante la posterior fase de elaboración hay que detallar los requerimientos del sistema hasta que este quede no ambiguo y pueda ser desarrollado por los diseñadores y programadores.

Cada caso de uso incluye todos los detalles sobre lo que tiene que hacerse para conseguir la funcionalidad. Se necesita considerar la funcionalidad básica, las alternativas a ella, el funcionalidad que se produce cuando se presentan errores o estados de excepción, las precondiciones o estado de partida previsto para el caso de uso y las post-condiciones o situación que tiene que producirse cuando el caso de error concluya. Los casos de uso puede incluir pueden incluir condiciones, bifurcaciones, alternativas y bucles de flujo de control.

Cuando el caso de uso es sencillo, es fácil describir linealmente la información asociada, sin embargo cuando el caso de uso se hace complejo y presenta muchas ramas o bucles, existen diferentes estilos de describir los casos de uso.

La descripción de un caso de uso comprende los siguientes elementos:

•Inicio del caso de uso:Identifica el evento o interacción que desencadena el caso de uso.

•Final del caso de uso: El evento o situación con la que el caso de uso concluye.

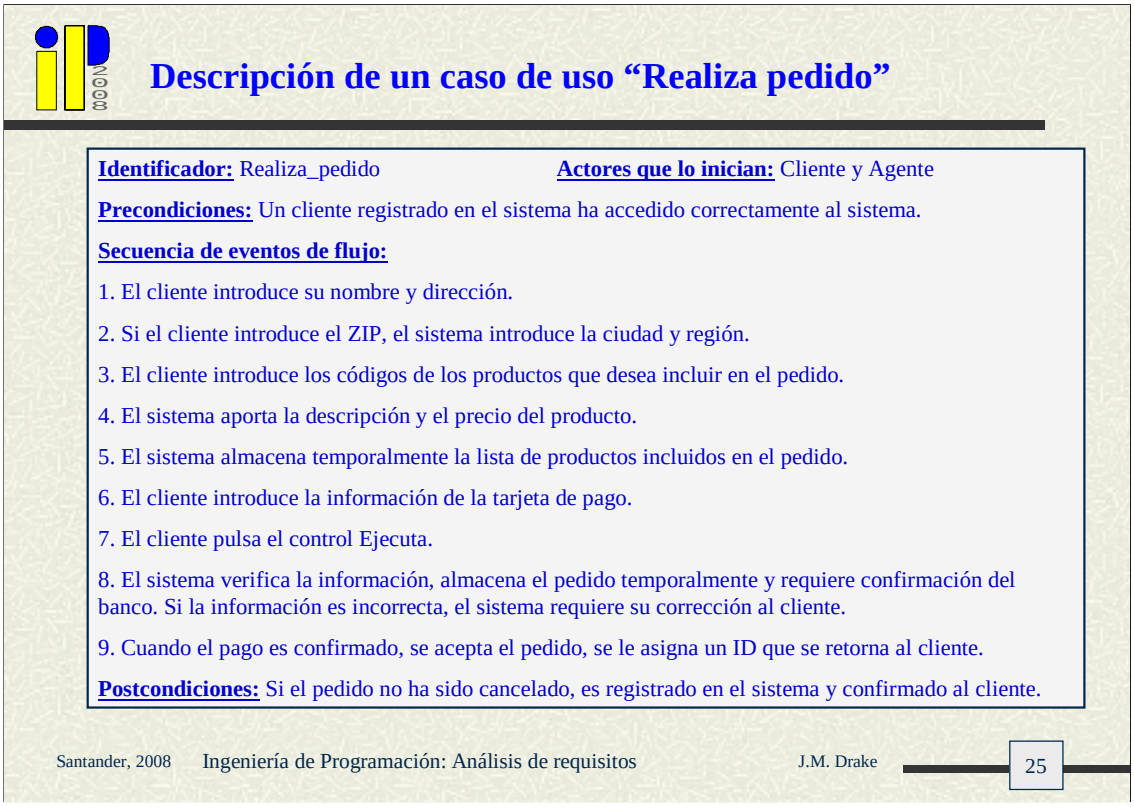
•La interacción entre sistema y los actores: describe la secuencia abierta de interacciones que constituyen el caso de uso.

•El intercambio de información: Describe las informaciones o datos que se intercambian en las interacciones.

•La cronología y el origen de las informaciones: Informaciones que se requieren (internas o externas)

•Las repeticiones de las iteraciones: Bucles de iteraciones que se realizan dentro del caso de uso.

•Las opciones de las iteraciones:Bifurcaciones o ramificaciones de interacciones.


Descripción de un caso de uso “Realiza pedido”
Identificador: Realiza_pedido
Actores que lo inician: Cliente y Agente
Precondiciones: Un cliente registrado en el sistema ha accedido correctamente al sistema.
Secuencia de eventos de flujo:
1. El cliente introduce su nombre y dirección.
2. Si el cliente introduce el ZIP, el sistema introduce la ciudad y región.
3. El cliente introduce los códigos de los productos que desea incluir en el pedido.
4. El sistema aporta la descripción y el precio del producto.
5. El sistema almacena temporalmente la lista de productos incluidos en el pedido.
6. El cliente introduce la información de la tarjeta de pago.
7. El cliente pulsa el control Ejecuta.
8. El sistema verifica la información, almacena el pedido temporalmente y requiere confirmación del banco. Si la información es incorrecta, el sistema requiere su corrección al cliente.
9. Cuando el pago es confirmado, se acepta el pedido, se le asigna un ID que se retorna al cliente.
Postcondiciones: Si el pedido no ha sido cancelado, es registrado en el sistema y confirmado al cliente.
Santander, 2008
Ingeniería de Programación: Análisis de requisitos
J.M. Drake
25

## Pre- y post- condiciones

- Una pre-condición es la situación en que debe estar el sistema para que el caso de uso se ejecute en la forma prevista. Si la precondición no se satisface, el caso de uso puede seguir secuencias no previstas.
- Una post-condición es una característica que se tiene que satisfacer en el sistema si se ha partido de unas precondiciones ciertas y se ha ejecutado el caso de uso.

  La post-condición tiene que ser cierta con independencia de que rama o alternativa del caso de uso se ha seguido.



Las precondiciones y post-condiciones que ocurría o ocurre antes y después de que se ejecute el caso de uso. Ellos establecen el estado en que debe estar el sistema al iniciarse el caso de uso (precondición) o en que estado va a quedar el sistema al terminar el caso de uso (post-condición). La post-condición debe cumplirse con independencia de que rama o alternativa de flujo del caso de uso se ha seguido.

# Caso de uso con alternativas condicionales

**Identificador:** Encuentra_pedido **Actores que lo inician:** Cliente y Agente.

**Precondiciones:** Un cliente registrado en el sistema ha accedido correctamente al sistema.

**Secuencia de eventos de flujo:**

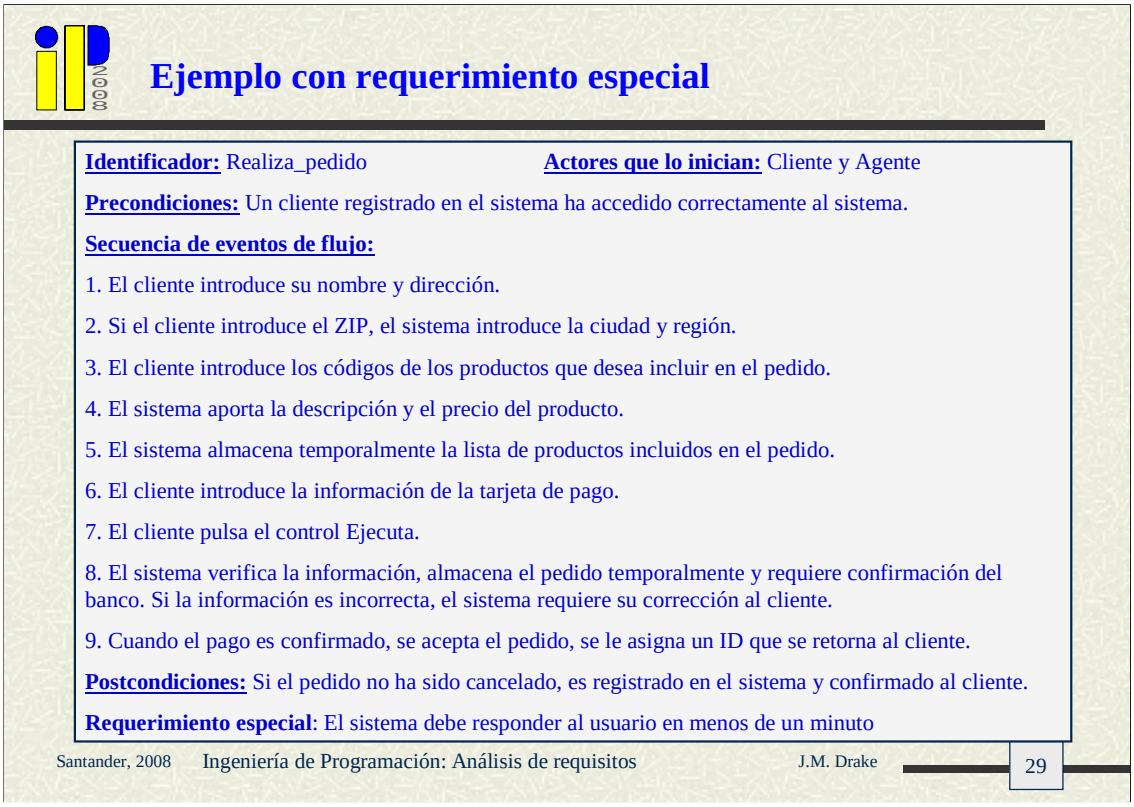
1. El cliente entra el ID del pedido, o el nombre del cliente.
2. El cliente pulsa el control Busca.
3. Si el usuario ha introducido el ID de un pedido:
   3.a) El sistema visualiza la orden y el caso de uso finaliza.
4. Si el usuario ha entrado el nombre del cliente:
   4.a) El sistema retorna una lista de todos los pedidos pendientes para el cliente.
   4.b) El usuario selecciona el pedido que es de su interés.
   4.c) El sistema visualiza el pedido y el caso de uso finaliza.

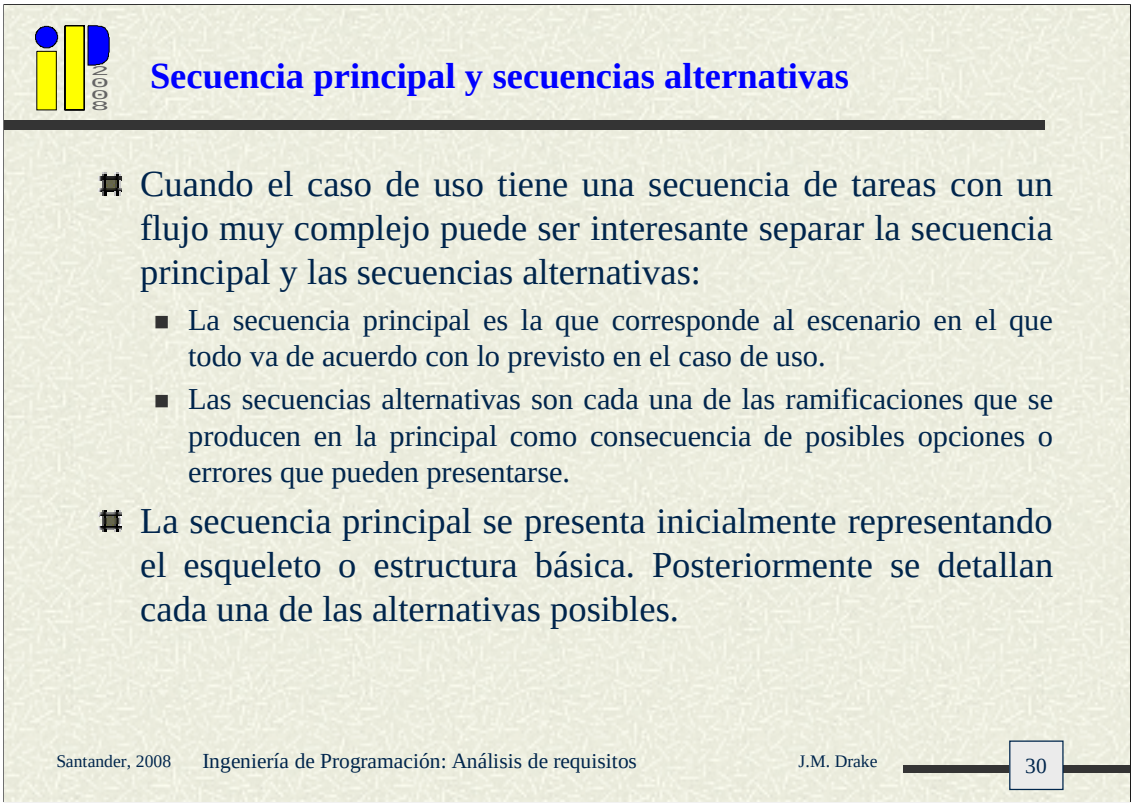
**Postcondiciones:** Ninguna.

# Ejemplo caso de uso con bucle

**Secuencia de eventos de flujo:**

1. El cliente introduce su nombre y dirección.
2. Si el cliente introduce el ZIP, el sistema introduce la ciudad y región.
3. El cliente introduce los códigos de los productos que desea incluir en el pedido.
4. Por cada producto del pedido:
   - 4.a El sistema aporta la descripción y el precio del producto.
   - 4.b El sistema suma el precio del producto en el total del pedido.
5. El sistema almacena temporalmente la lista de productos incluidos en el pedido.
6. El cliente introduce la información de la tarjeta de pago.
7. El cliente pulsa el control Ejecuta.
8. El sistema verifica la información, almacena el pedido temporalmente y requiere confirmación del banco. Si la información es incorrecta, el sistema requiere su corrección al cliente.
9. Cuando el pago es confirmado, se acepta el pedido, se le asigna un ID que se retorna al cliente.


Ejemplo con requerimiento especial
Identificador: Realiza_pedido
Actores que lo inician: Cliente y Agente
Precondiciones: Un cliente registrado en el sistema ha accedido correctamente al sistema.
Secuencia de eventos de flujo:
1. El cliente introduce su nombre y dirección.
2. Si el cliente introduce el ZIP, el sistema introduce la ciudad y región.
3. El cliente introduce los códigos de los productos que desea incluir en el pedido.
4. El sistema aporta la descripción y el precio del producto.
5. El sistema almacena temporalmente la lista de productos incluidos en el pedido.
6. El cliente introduce la información de la tarjeta de pago.
7. El cliente pulsa el control Ejecuta.
8. El sistema verifica la información, almacena el pedido temporalmente y requiere confirmación del banco. Si la información es incorrecta, el sistema requiere su corrección al cliente.
9. Cuando el pago es confirmado, se acepta el pedido, se le asigna un ID que se retorna al cliente.
Postcondiciones: Si el pedido no ha sido cancelado, es registrado en el sistema y confirmado al cliente.
Requerimiento especial: El sistema debe responder al usuario en menos de un minuto
Santander, 2008
Ingeniería de Programación: Análisis de requisitos
J.M. Drake
29




Un caso de uso detallado puede ser bastante complicado. Esto no suele ser lo que ocurre en la fase de inicio. Sino cuando evoluciona y va acumulando detalles. A fin de hacer que los casos sigan siendo básicamente legibles,es conveniente describirlo en dos bloques: Primero se describe la funcionalidad básica utilizando la secuencia principal, y luego se describen las variaciones que pueden realizarse sobre él a través de secuencias alternativas que hacen referencia al principal.
La secuencia principal corresponde a la ejecución del caso de uso mas habitual, aquel que no tiene errores ni se opta por atajos. A menudo esta secuencia se denomina secuencia "feliz", la que se produce cuando todo va bien. Al no tener secuencias alternativas, la secuencia principal es lineal y en consecuencia muy legible.
Una secuencia alternativa es una que se produce cuando se elige una situación o evento posible diferente del considerado en la principal.
Las secuencias alternativas son muy útiles para declarar eventos de error, de excepción o de aborto en el que el instante de ocurrencia no está bien definido (puede ocurrir en cualquier instante).

# Secuencias alternativas al caso de uso Realiza_perdido

**Identificador:** Realiza_pedido **Actores que lo inician:** Cliente y Agente.

**Precondiciones:** Un cliente registrado en el sistema ha accedido correctamente al sistema.

**Secuencia de eventos de flujo:**

1. El cliente introduce su nombre y dirección.
2. Si el cliente introduce el ZIP, el sistema introduce la ciudad y región.
3. El cliente introduce los códigos de los productos que desea incluir en el pedido.

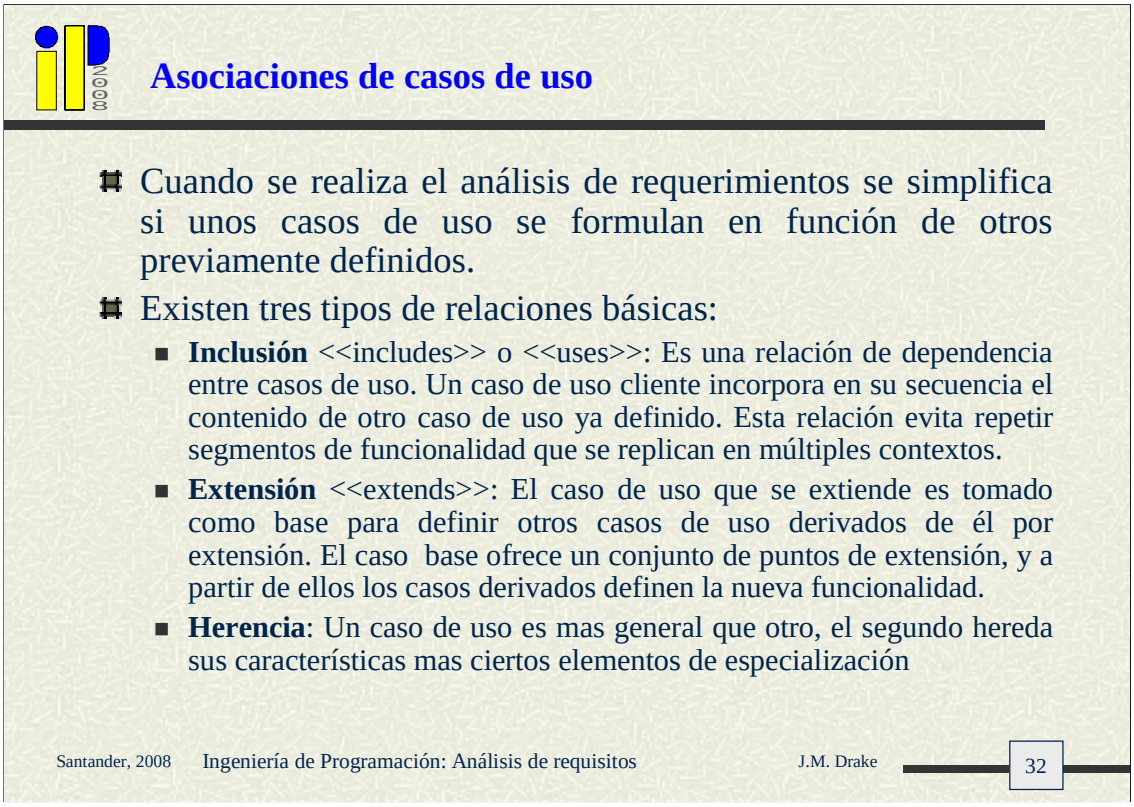
**...** (ver trasparencia 28)

**Secuencias alternativas:**

•En cualquier instante, antes de pulsar “Ejecutar” el cliente puede pulsar “Cancelar”. El pedido no es registrado y el caso de uso finaliza.

•En el paso 8, si cualquier información es incorrecto, el sistema avisa al cliente para que corrija la información.

•En el paso 9, si el pago no es confirmado, el sistema avisa al cliente para que corrija las información de la tarjeta o cancele. Si el cliente opta por corregir, se pasa al paso 6. En caso contrario, termina.



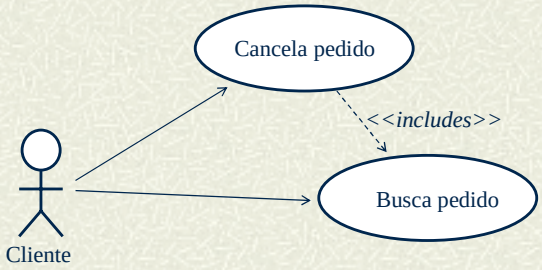

UML define tres tipos de relaciones entre casos de uso:

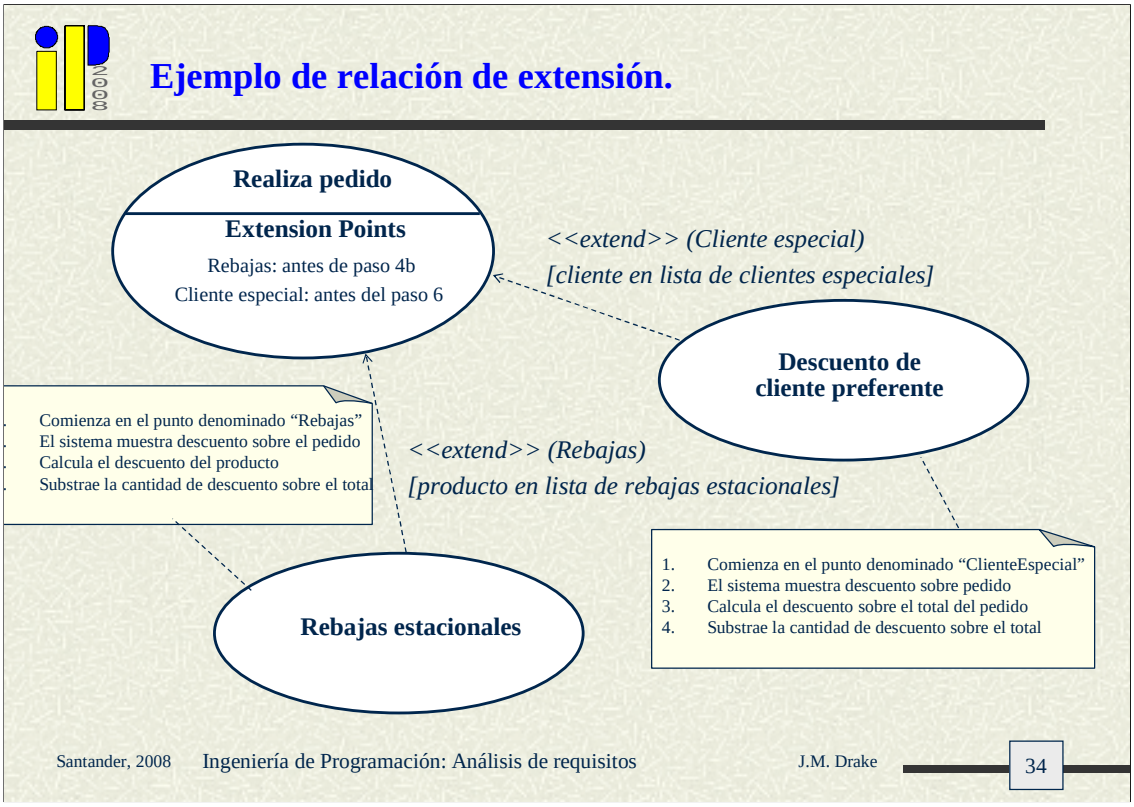
•**Inclusión**: La relación inclusión utiliza un estereotipo de dependencia.Establece que un segmento de comportamiento que esta representado por el caso de uso base es utilizado por otro caso de uso que llamamos caso de uso cliente. Esta relación es especialmente útil si existe un segmento de funcionalidad que se repite en diferentes contextos y no se desea repetir su especificación.

•**Extensión**: La relación de extensión es representada por una relación de dependencia entre los dos casos de uso, con la flecha apuntando hacia la clase base. El caso de uso a ser extendido es denominado caso de uso base, y el caso de uso extendido es llamado caso de uso cliente. La idea es que el caso de uso base identifica un número de puntos de extensión específicos en descripción de comportamiento. Estos puntos son localizaciones a partir de los que los clientes pueden insertar un comportamiento adicional.

•**Generalización**: Esta relación declara que un caso de uso (llamado general) es una forma mas general que otro (llamado derivado). Al igual que en la generalización de las clases, la clase mas especializada hereda de la base todas sus características, de las cuales algunas pueden ser sobrescritas y especializadas por la derivada.

## Ejemplo de relación de inclusión





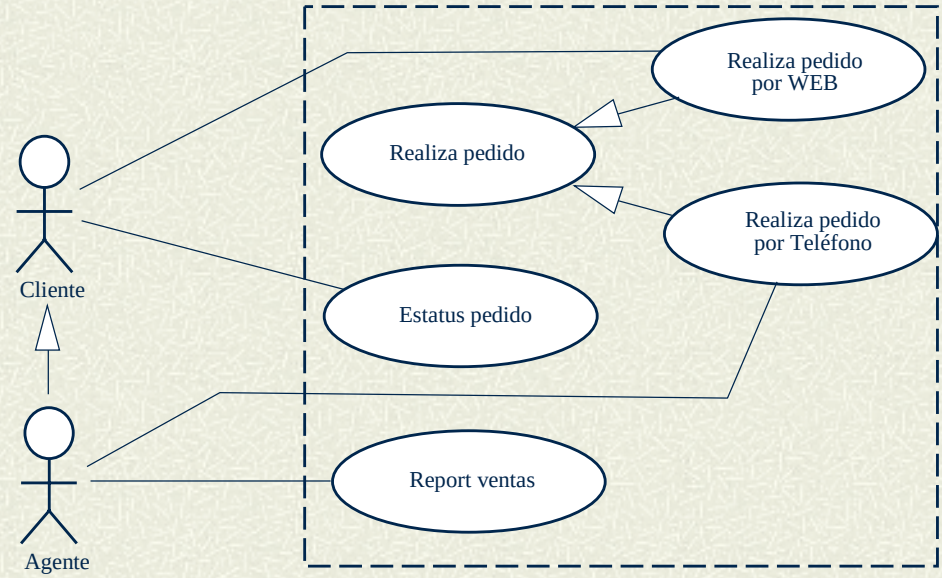

**Extensión de caso de uso: "Rebajas estacionales"**
**Secuencia básica:**

1. El caso de uso comienza en el paso de nominado "Rebajas".
2. El sistema muestra el descuento sobre el pedido.
3. El sistema calcula el descuento sobre el producto.
4. El sistema substrae la cantidad de descuento sobre la suma total del pedido.

**Extensión de caso de uso: "Descuento de cliente preferente"**
**Secuencia básica:**

1. El caso de uso comienza en el paso de nominado "Cliente especial".
2. El sistema muestra el descuento sobre el pedido.
3. El sistema calcula el descuento total sobre el pedido del cliente.
4. El sistema substrae la cantidad de descuento sobre la suma total del pedido.

## Ejemplo de “Herencia” entre Casos de Uso



## Casos de uso e interfaces