Tecnicatura Universitaria

en Desarrollo de Software

Ingeniería del Software

2020

Captura de Requisitos

Captura de Requisitos

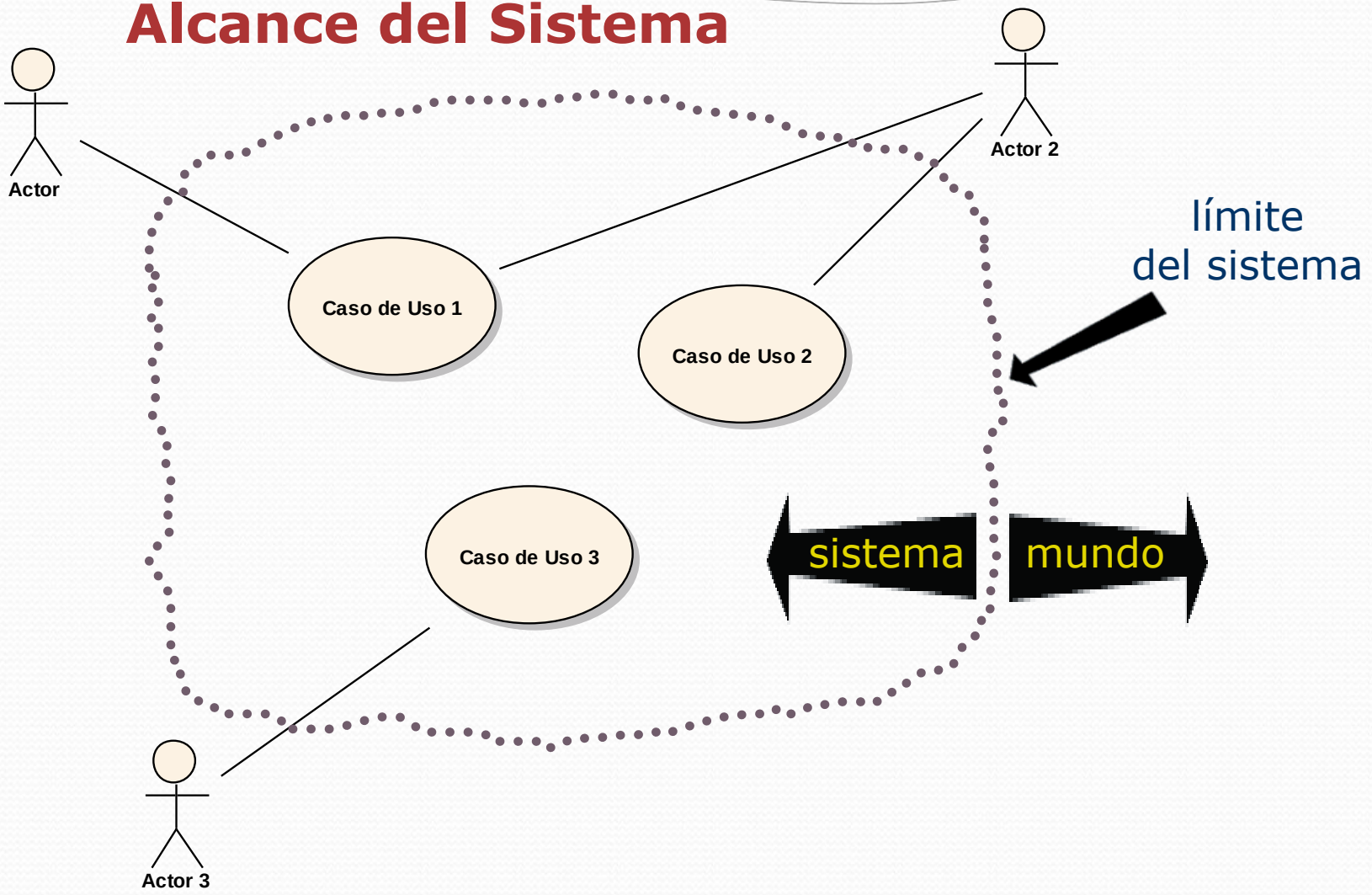
- ***"Es el proceso de averiguar normalmente en circunstancias, difíciles lo que se debe construir"***

- La captura de requisitos **es complicada** porque la mayoría de los usuarios no sabe que partes de su trabajo pueden transformarse en software.
- Con frecuencia los usuarios *no saben cuáles son los requisitos ni tampoco cómo especificarlos de forma precisa*.

Captura de Requisitos

- El propósito principal es guiar el desarrollo hacia el sistema correcto.
- Mediante una descripción de los requisitos del sistema (capacidades que el sistema debe cumplir) para que pueda **llegarse a un acuerdo entre el cliente y los desarrolladores sobre qué debe y qué no debe hacer el sistema**.
- Para conseguirlo, el cliente (no informático) debe ser capaz de leer y comprender el resultado de la captura, es por esto que hay que utilizar el lenguaje del cliente.

Alcance del Sistema



¿Hasta Donde Llega el Sistema a Construir?

¿Porque Casos de Uso?

- Los CU nos ayudan a capturar los requisitos adecuados.
- Para el usuario, un CU es un modo de usar el sistema.
- Así, si los analistas pueden describir todos los CU que necesita el usuario, entonces saben lo que debe hacer el sistema.
- Cada usuario necesita varios CU distintos, cada uno de los cuales representa los modos diferentes de utilizar el sistema.

Modelo de Casos de Uso

- «El **modelo de CU** sirve como un acuerdo entre clientes y desarrolladores, y proporciona la entrada fundamental para el análisis, diseño, y las pruebas. »
- Un modelo de CU es un modelo del sistema que contiene actores, casos de uso y sus relaciones.
- El modelo de CU describe lo que hace el sistema para cada tipo de usuario (uno o mas actores)
- Otros sistemas externos que interactúan nuestro sistema, también son actores.

Actores

- "Los **actores** representan a terceros fuera del sistema que colaboran con el sistema"
- Si hemos identificado **todos los actores**, tenemos **identificado el entorno externo del sistema**.
- Cada vez que un usuario concreto (humano u otro sistema) interactúa con el sistema, la instancia correspondiente del actor está desarrollando ese papel.

- Así, **una instancia de un actor es por tanto un usuario concreto que interactúa con el sistema.**

Caso de Uso

- Cada forma en que los actores usan el sistema se representa con un Caso de Uso.
- Los casos de uso son "fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.
- *"Especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores" (incluyendo alternativas dentro de la secuencia)*

Caso de Uso

- *Un* CU *entrega un resultado que se puede observar y que añade valor a un actor en concreto. Cada ejecución satisfactoria de un* CU *debe proporcionar algún valor al actor para alcanzar su objetivo. Esto se debe aplicar al actor iniciador.*

Instancia de Caso de Uso

- Una **instancia de CU** es la realización o ejecución de un CU. Ésta interactúa con instancias de actores y ejecuta una secuencia específica de acciones principales.
- Puede haber alternativas en la secuencia de acciones, es decir otros caminos.
- Consideramos **atómicas** las instancias de CU, cada una se ejecuta por completo o no se ejecuta nada, por tanto el comportamiento de cada CU puede interpretarse independiente del de otros CU.

Encontrar actores y Casos de Uso

El objetivo de esto es:

- Delimitar el sistema de su entorno
- Esbozar quién y qué (actores) interactúan con el sistema, y qué funcionalidad (casos de uso) se espera del sistema

Asociación o Inclusión

- Una secuencia e acciones usuario-sistema se puede especificar en un CU o varios, los cuales el actor invoca uno tras otro.
- Tenemos que considerar si un CU es completo por si mismo (*asociación*) o si siempre se ejecuta a continuación de otro CU *(inclusión)*.

Es preciso seguir algunos pasos:

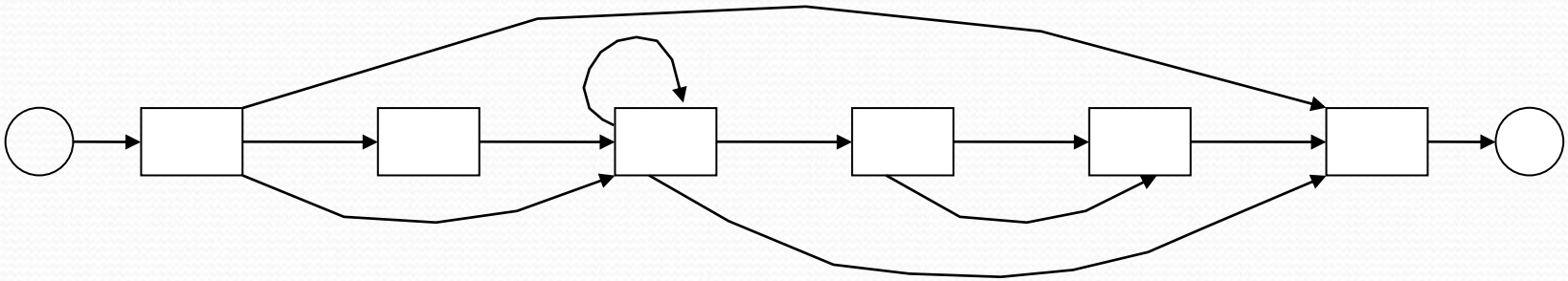
1. *Encontrar los **actores***
2. *Encontrar los **casos de uso***
3. *Describir brevemente cada caso de uso*
4. *Describir los caminos básicos de todos los casos de uso*
5. *Por ultimo, podemos refinar el modelo al:*
   - *Factorizar comportamientos comunes y compartidos (como casos incluidos)*
   - *Agregar comportamientos adicionales u opciones (como extensiones a un CU)*

2- Encontrar Casos de Uso

- *Para encontrar los* CU, se propone por cada actor encontrado una función del sistema. Entonces se va repasando los actores uno por uno y proponiendo los CU para ellos.
- Elegimos un nombre para cada caso de uso de forma que nos haga pensar en la secuencia de acciones concreta que añade valor a un actor.
- El nombre de un CU a menudo comienza con un verbo, y debe reflejar cual es el objetivo de la interacción entre el actor y el sistema.

3-Descripción Breve

- Descripción Breve: Consiste explicar cada CU en unas pocas palabras, o frases que **resumen** las acciones que se realizan.
- En el Camino Básico se coloca una **detallada descripción paso a paso** de lo que el sistema necesita hacer en la interacción con el actor (punto 4).



4-Pre-Post-Condición

- *Debo definir el estado inicial como **precondición***
- *En el **camino básico** debe asegurarme de que vaya:*
  - *Cómo y cuándo comienza el CU (la primera acción)*
  - *El orden en que las acciones se deben ejecutar (numeración)*
  - *La interacción de los actores con el sistema*
  - *Describir explícitamente lo que hace el sistema (las acciones que ejecuta)*
  - *Separar las responsabilidades del sistema y las de los actores*
  - *Los cambios que se producen en el sistema*
  - *La utilización de objetos, valores de atributos y recursos del sistema*
  - *Cómo y cuándo termina el CU*
  - *Los Casos de uso incluidos por ese CU*
- *Definir los posibles estados finales como **postcondiciones***
- *En cada **camino alternativo** los CU extendidos del caso base, excepciones, errores, etc.*

4-Camino Básico

- El **Camino básico** de cada CU es una descripción textual de la secuencia de acciones del CU.

- Especifica lo que el sistema hace y como interactúa con los actores cuando se lleva a cabo ese CU

4-Camino Básico y Alternativos

- Elegir el camino básico completo (flechas rectas) desde el estado del inicio hasta el final, y describir ese camino en una sección la descripción.
- Entonces podemos describir en secciones separadas el resto de los caminos alternativos o desviaciones del camino básico (flechas curvas).

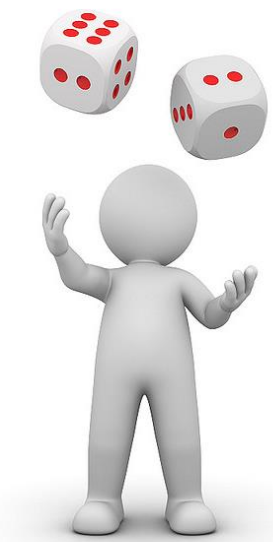
El camino básico debe ser el más normal, que se percibe habitualmente, aquel que proporciona el valor mas obvio»

5-Reestructurar el modelo

El modelo de Casos de uso una vez armado se debe reestructurar, con el objetivo de:

- *Extraer descripciones de funcionalidad (CU) generales y compartidas que pueden ser utilizadas por descripciones de CU más específicas. (**Include**)*

- *Extraer descripciones de funcionalidad (CU) adicionales u opcionales que pueden extender descripciones de CU mas específicas. (**Extend**)*



Inclusion Vs. Extension

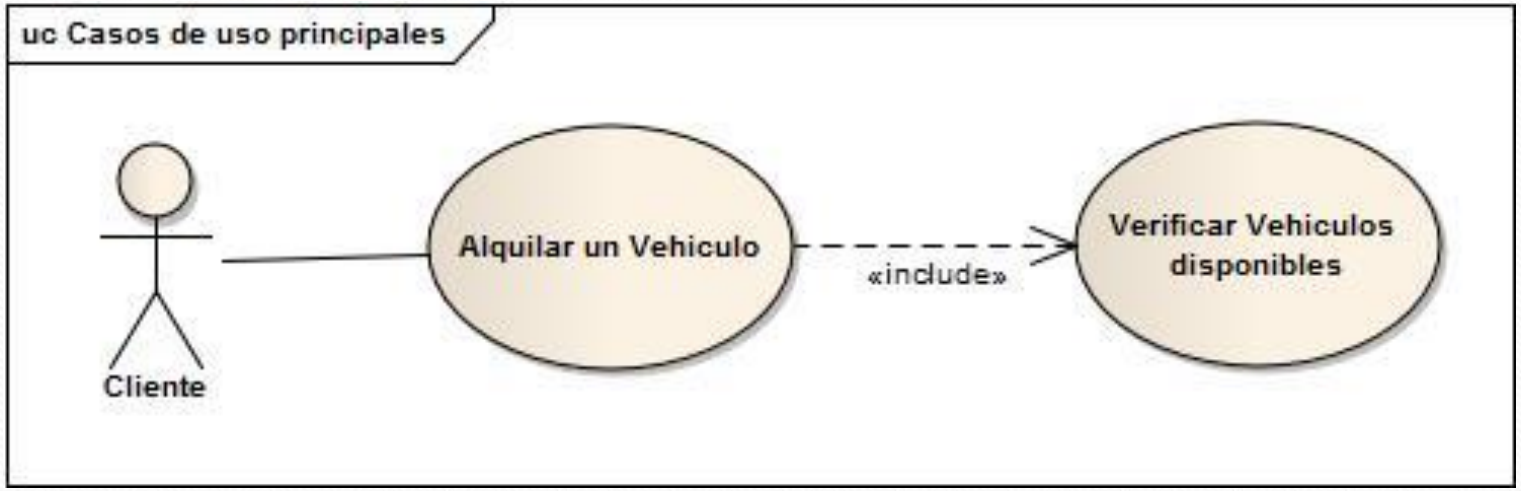
- *El **include** tiene el fin de reducir la redundancia, éste código puede extraerse y describirse en un caso de uso separado que puede ser después reutilizado por el caso de uso original.*
  - Si un CU A incluye a un CU B indica que una instancia del caso A incluirá también el comportamiento especificado en B.
- La relación ***extend*** modela la adición de una secuencia de acciones a un CU. La extensión se comporta como si fuera algo que se añade a la descripción, en ciertos casos particulares del caso de uso.

Inclusion

- El conjunto de pasos que ocurren al realizar una compra con tarjeta de crédito en un negocio.
- Se va a *solicitar la autorización de la tarjeta* de crédito con la que vas a pagar, siempre que se *registra una venta en cuotas.*

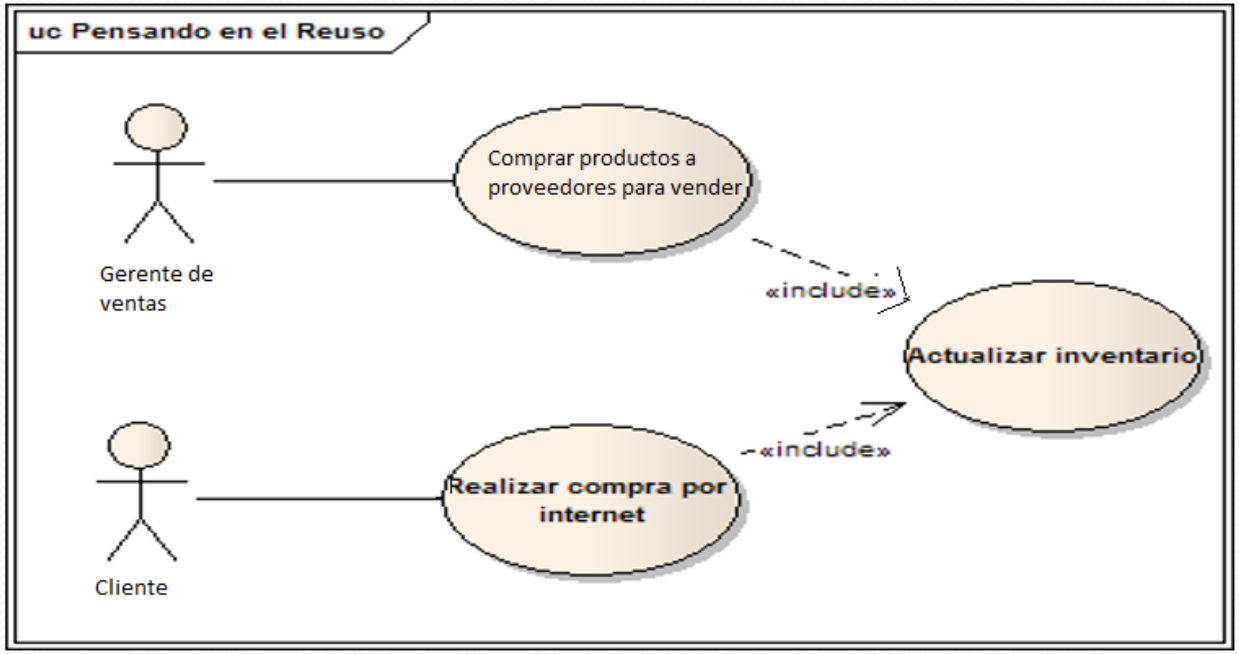
Inclusion

- El caso de uso “Verificar Vehículos disponibles” está incluido en el caso de uso “Alquilar un Vehículo”, el segundo es parte esencial del primero.
- Sin el segundo, el primero no podría funcionar bien.



Reuso: Evitando el Retrabajo

- Una de las razones por las cuales deberías de considerar el uso de este tipo de relaciones, es porque identificas que hay pasos que son iguales en dos o más casos de uso.
- Así, No hay que escribir y mantener a cada uno de las copias.



Reuso: Evitando el Retrabajo

- No querrás correr el riesgo de que esos pasos se diseñen, programen y prueben de maneras diferentes y con esfuerzos aislados por ti o tu equipo de desarrollo.

- Finalmente son la misma cosa! ¿para qué querríamos trabajar doble? Queremos economizar, ser más eficientes en el desarrollo, y ahí es cuando viene el beneficio de identificar estos tipos de relaciones; porque es una oportunidad de identificar **reutilización**

Mejor mantener el modelo simple

- **Son relaciones usadas para unir 2 casos de uso, cuyos flujos de eventos ocurren, normalmente en una sola sesión del usuario**
- **Podríamos colocarlos como un sólo caso de uso en lugar de dos, ya que ocurren juntos.**

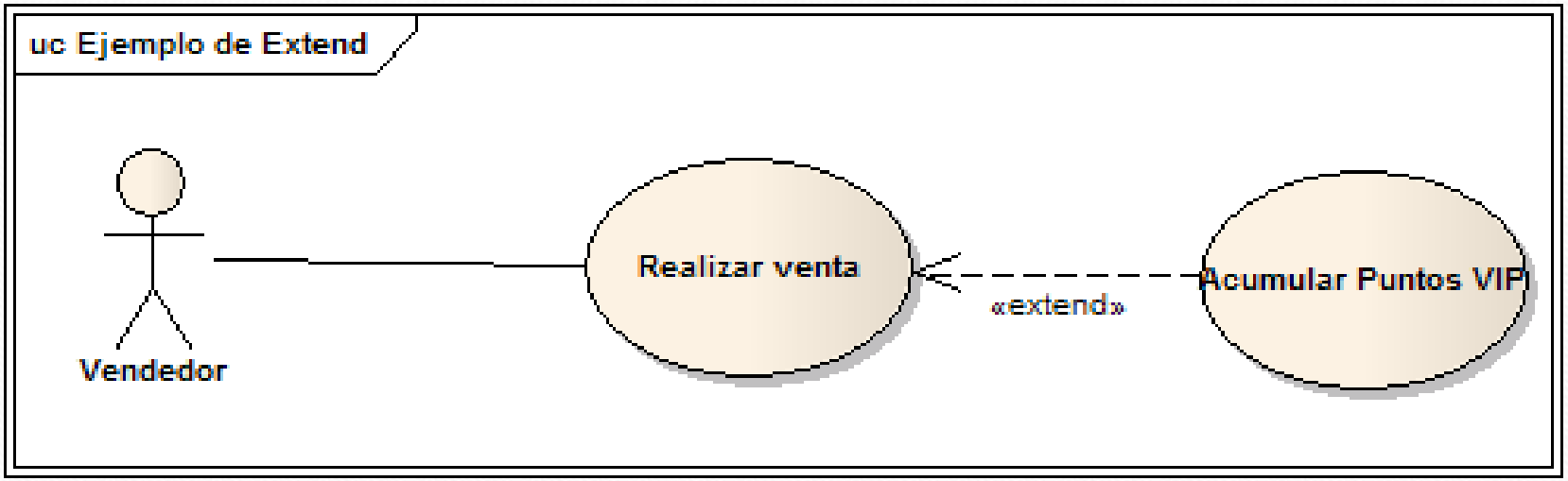
- *Obviamente, hay una razón por la cual decidimos separarlos en dos.* REUTILIZACION DE CODIGO.
- *Y porque además ofrecen un único punto de código a corregir o cambiar evitando riesgos de modificar en una copia y no en los demás.*

Extensión

- Una extensión también la podemos ver, como si dos flujos fueran parte de un único proceso.
- Pero, en el "***extend***" hay situaciones en que el caso de uso de extensión no es indispensable que ocurra, y cuando lo hace ofrece un valor extra (extiende) al objetivo original del caso de uso base.
- En cambio *en el "include" es necesario* que ocurra el caso incluido, tan sólo para satisfacer el objetivo del caso de uso base.

Extensión

- Puedes “Realizar Venta” sin “Acumular Puntos de Cliente VIP”, cuando no eres un cliente VIP. Pero, si eres un cliente VIP sí acumularás puntos.
- Por tanto, “Acumular Puntos” es una extensión de “Realizar Venta” y sólo se ejecuta para cierto tipo de ventas, no todas.



Error: Inclusiones de Inclusiones

- ***No es raro ver modelos de casos de uso que llegan a tener decenas de inclusiones y extensiones, incluso las inclusiones y extensiones se vuelven a extender a varios niveles,* generando una maraña de casos de uso *que no ofrecen valor al ser mostrados explícitamente.***

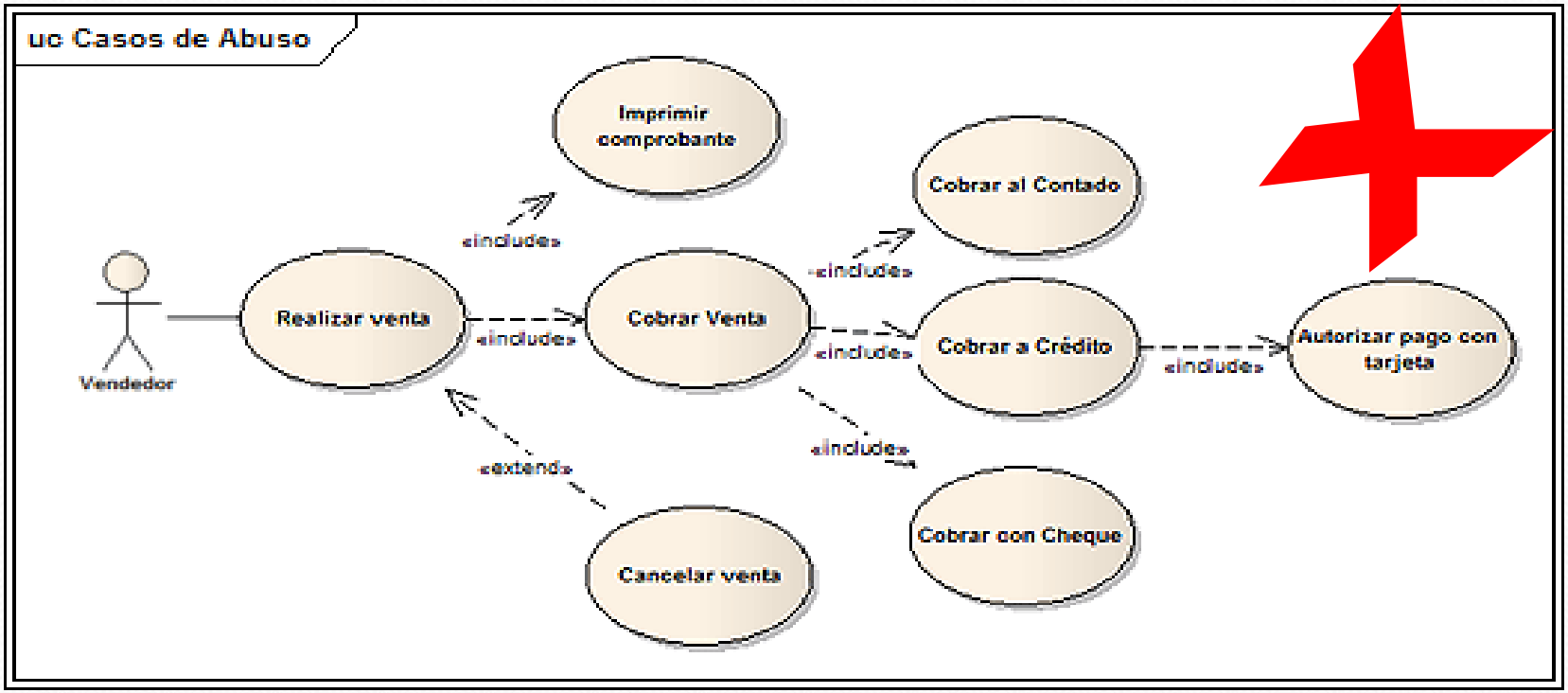
- Debe de existir una razón importante para que decidamos dividir un caso de uso en dos que serán unidos por medio de alguna de estas relaciones

Error: Inclusiones de Inclusiones

- Es importante comprender que ***el objetivo de estos tipos de relaciones NO consiste (remarco la negación) en motivar la división de los casos de uso en la mayor cantidad de pedazos.***
- Debe de existir una razón importante para que decidamos dividir un caso de uso en dos que serán unidos por medio de alguna de estas relaciones.
- Si entendemos esto y somos congruentes, obtendremos un beneficio real para el proyecto; fin último del uso de UML.

Error: Inclusiones de Inclusiones

- No se deben partir los casos de uso en infinidad de “include” y “extend”, esos son pasos dentro del detalle y se verán en diagramas de secuencia mas adelante.



Error: Inclusiones de Inclusiones

- La razón por la que la gente suele partir sus casos de uso en infinidad de “include” y “extend” es porque quieren conocer, entender y comunicar el máximo detalle de los casos de uso en el diagrama. Hay quien llega a utilizar, erróneamente, estas relaciones para mostrar el orden en que se ejecutan estos casos de uso.
- **Debemos de recordar que al modelar el diagrama de casos de uso no buscamos analizar el detalle, y mucho menos los flujos.**
- Todo ese detalle lo podremos plasmar en otro tipo de diagramas, como los diagramas de interacción, de actividad, de estados, o simplemente un texto en una especificación.