

# Ingeniería de Software

---

Silvia Guardati

# Modelo de Requisitos: Objetivos

---

- Analizar y entender el problema (y el negocio).
- Delimitar el sistema.
- Capturar las funcionalidades del sistema.
- El modelo de requisitos describe qué hace el sistema.

# Objetivos (cont.)

---

La base del modelo *requisitos*.

**Importante:** diálogo con el cliente/usuario.

Se usará el *modelo de casos de uso*.

Sobre este modelo se construyen los otros (análisis, diseño, implementación y pruebas) y la documentación.

# Conceptos (1)

---

- **Requisitos:** descripción de los servicios que debe brindar un sistema y sus restricciones.
- **Ingeniería de Requisitos:** disciplina encargada de descubrir, analizar, documentar, verificar y administrar esos requisitos (y restricciones).
- **Sistema:** incluye *hardware, software, personas, información, técnicas, servicios, y otros elementos de soporte.*

# Conceptos (2)

---

- **Requisitos del Sistema:** son los requisitos de todo el sistema.
- **Requisitos del Software:** se refieren sólo al producto de software.

# Requisitos vs. Diseño

---

- Requisitos definen el *QUÉ* (el problema) del sistema.
- Diseño define el *CÓMO* (la solución).

# Reporte de Standish Group (1)

---

- USA: más de \$250 billones x año en aplicaciones IT (175,000 proyectos)
- 31.1% de los proyectos se cancelan antes de terminarse
- 52.7% de los proyectos se terminan pero cuestan más o tardaron más tiempo o no tienen todas las funcionalidades
- 16.2% de los proyectos se terminan en tiempo, con el presupuesto estimado y con todas las funcionalidades

Fuente de varios artículos: <https://www.projectsmart.co.uk/white-papers/>

# Reporte de Standish Group (2)

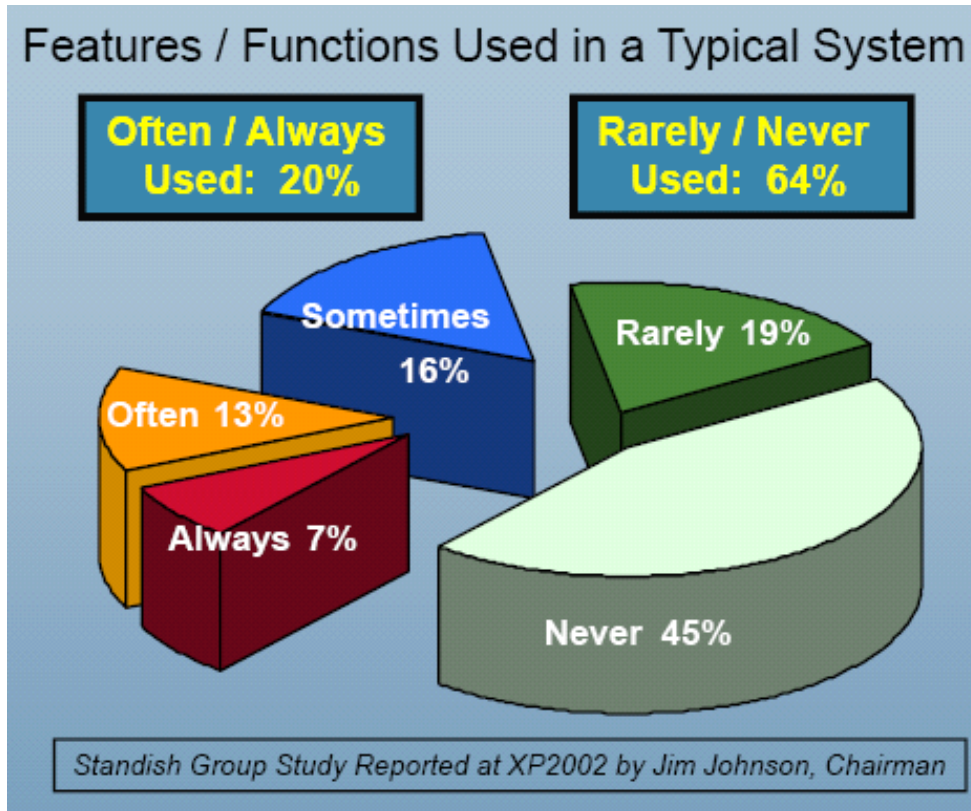
---

Factores de éxito:

1. Involucramiento del usuario: 15,9%
2. Soporte de los ejecutivos/administradores: 13,9%
3. **Requisitos claramente establecidos: 13%**
4. Planeación adecuada: 9,6%
5. Expectativas realistas: 8,2%
6. Otros



# Reporte de Standish Group (3)



¿Ejemplos?

Fuente: <http://theagileexecutive.com/2010/01/11/standish-group-chaos-reports-revisited/>

# Requisitos Funcionales y No Funcionales

---

**Funcionales**: expresan la interacción entre el sistema y su entorno. Son los servicios o funciones que proveerá el sistema.

Ejemplo:

- Se debe registrar cada venta: monto, vendedor, artículos, etc.
- Se debe generar un listado de todas las ventas al final del día.

# Requisitos Funcionales y No Funcionales

---

**No-funcionales**: expresan las restricciones/características que limitan las alternativas de solución. Son restricciones a los servicios o funciones ofrecidos por el sistema.

Ejemplo:

- Las ventas deben registrarse en menos de 4 seg.
- Se debe poder trabajar en red (o no).

# Requisitos No Funcionales (1)

---

**Del Producto**: especifican restricciones sobre el comportamiento del producto. Por ejemplo: desempeño, confiabilidad, portabilidad, usabilidad, etc.

**De la Organización**: especifican restricciones impuestas por las políticas y procedimientos de la organización, del cliente y del desarrollador. Por ejemplo: políticas de confidencialidad, estándares, lenguajes de programación, método de diseño, hardware, etc.

# Requisitos No Funcionales (2)

---

**Externos**: especifican restricciones derivadas del entorno. Por ejemplo: interoperabilidad con sistemas externos, legislativos (privacidad de la información, informes a Hacienda, etc.), éticos, etc.

# Elementos a tener en cuenta (1)

---

**Funcionalidad y restricciones asociadas**: qué se espera del sistema, cuándo debe estar listo, cómo debe operar, alternativas para el mantenimiento del mismo, restricciones de desempeño (velocidad, tiempo de respuesta, capacidad de proceso), etc.

**Datos/Interfaces**: en qué forma deben estar los datos para E/S, con qué frecuencia, quién los provee, precisión en los cálculos, flujo en el sistema, interfaces hacia/de sistemas/personas, etc.

# Elementos a tener en cuenta (2)

---

**Seguridad**: control de acceso a las funciones/datos, aislamiento de los programas, respaldos -frecuencia, disponibilidad-, seguridad física de la información y del equipo, etc.

**Aseguramiento de la Calidad**:

1. Confiabilidad: tiempo medio entre fallas, tolerancia a fallas, etc.
2. Disponibilidad: tiempo “muerto” luego de una falla, mantenimiento estando activo, tiempo máximo permitido de no disponibilidad, etc.
3. Mantenibilidad
4. Portabilidad
5. ...

# Elementos a tener en cuenta (3)

---

**Documentación**: qué se debe documentar, formato, dirigido a quién, cómo debe estar disponible, etc.

**Ubicación y entorno físico**: dónde va a operar, en uno o en varios puntos, restricciones ambientales, etc.

**Usuarios y factores humanos**: tipos de usuarios, entrenamiento requerido, restricciones para su uso, etc.

**Recursos**: personal y equipo para desarrollar, usar y mantener el sistema, de espacio, calendario, materiales, etc.

**Criterios de éxito/aceptación**: que se ejecuten correctamente ciertas tareas, máximo número de defectos, etc.



# Elementos a tener en cuenta (4)

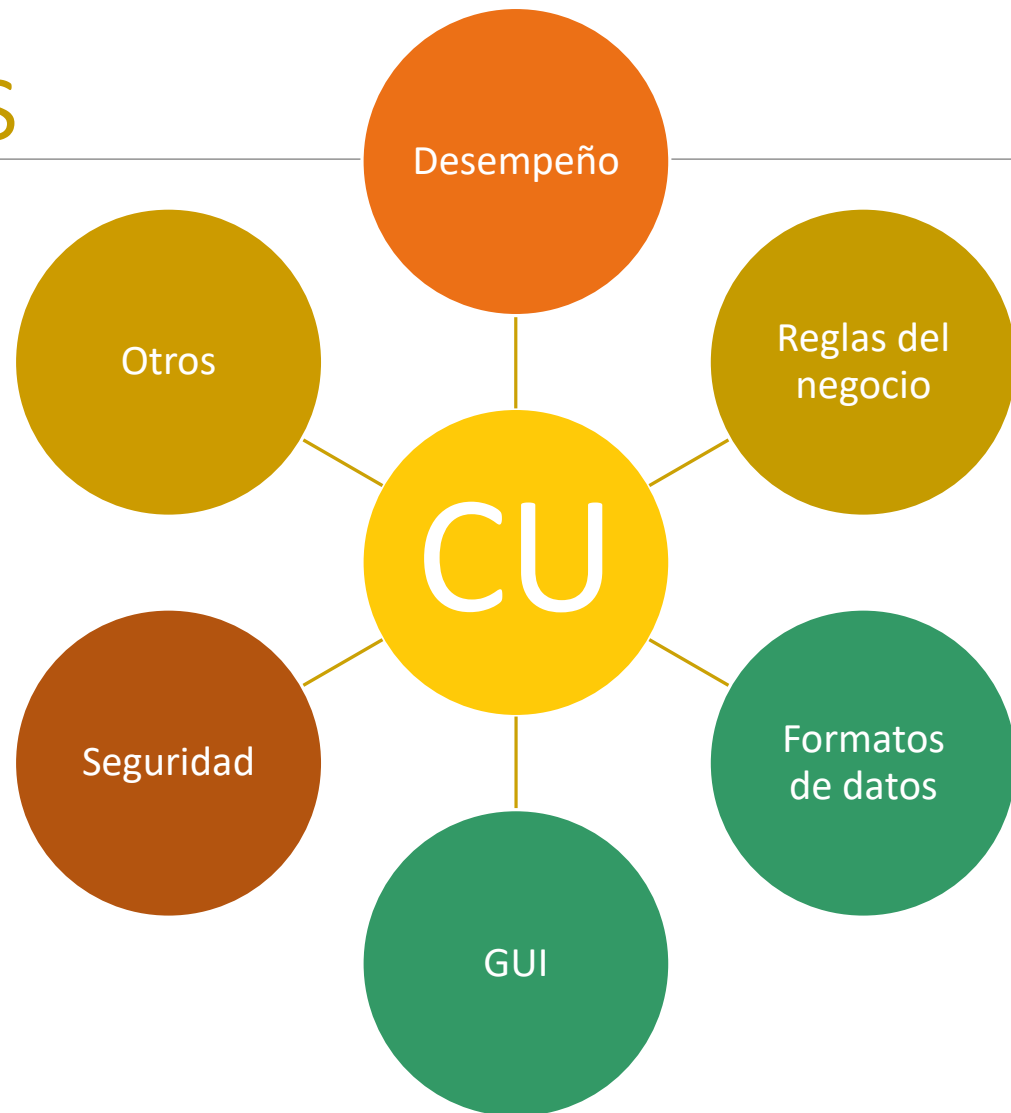
---

**Requisitos del negocio**: tener en cuenta los objetivos del negocio.

- Áreas potencialmente beneficiadas: finanzas, operación del negocio, posicionamiento estratégico / competitivo, adoptar una nueva ley, nivel tecnológico, etc.
- Visión del producto
- Beneficios para los stakeholders
- Descripción de las funcionalidades (alto nivel)
- Prioridades del proyecto
- Limitaciones del producto

# Requisitos

---



# Requisitos - Fuentes

---

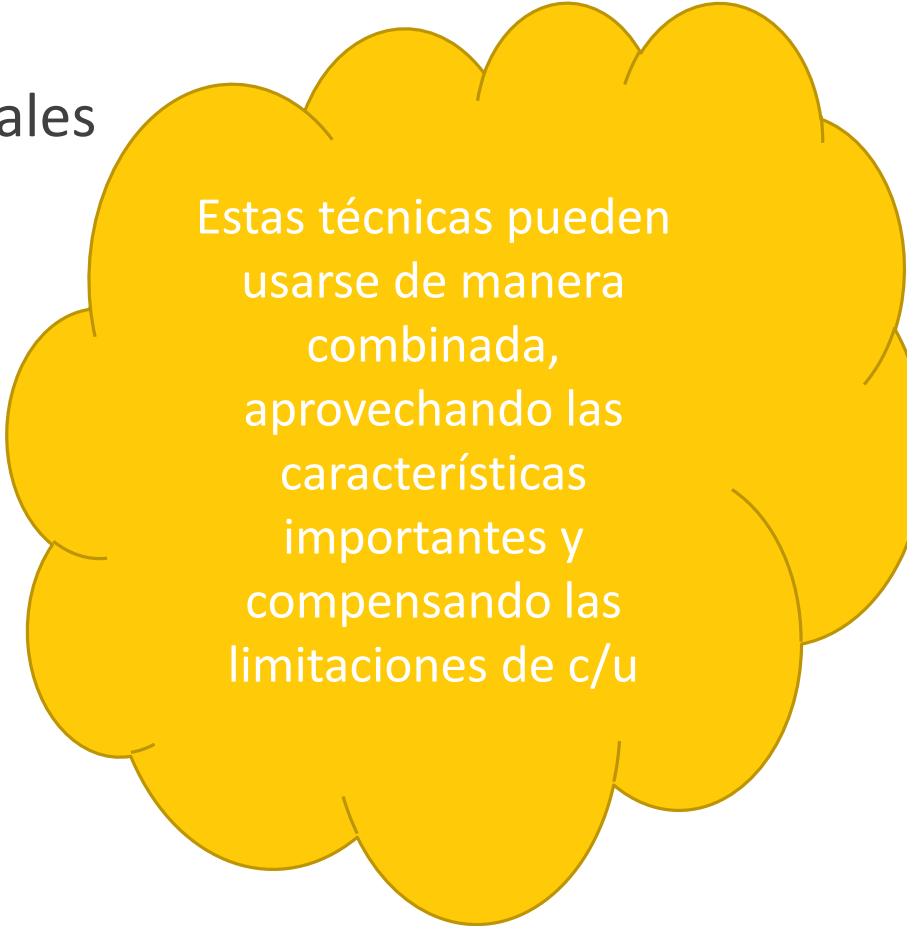
- Necesidades del cliente, usuario, otros interesados
- Organización actual y sistemas
- Versión actual del sistema y su documentación(\*)
- Desarrolladores de versión anterior (\*)
- Artículos y otras fuentes
- Sistemas análogos ya existentes (\*)

(\*) Si aplica

# Técnicas para obtener/representar requisitos

---

- Investigación
- Entrevistas individuales/grupales
- Encuestas/Cuestionarios
- Tormenta de ideas
- Workshop
- Documentación
- Observación/Participación
- Prototipado
- ...



Estas técnicas pueden usarse de manera combinada, aprovechando las características importantes y compensando las limitaciones de c/u

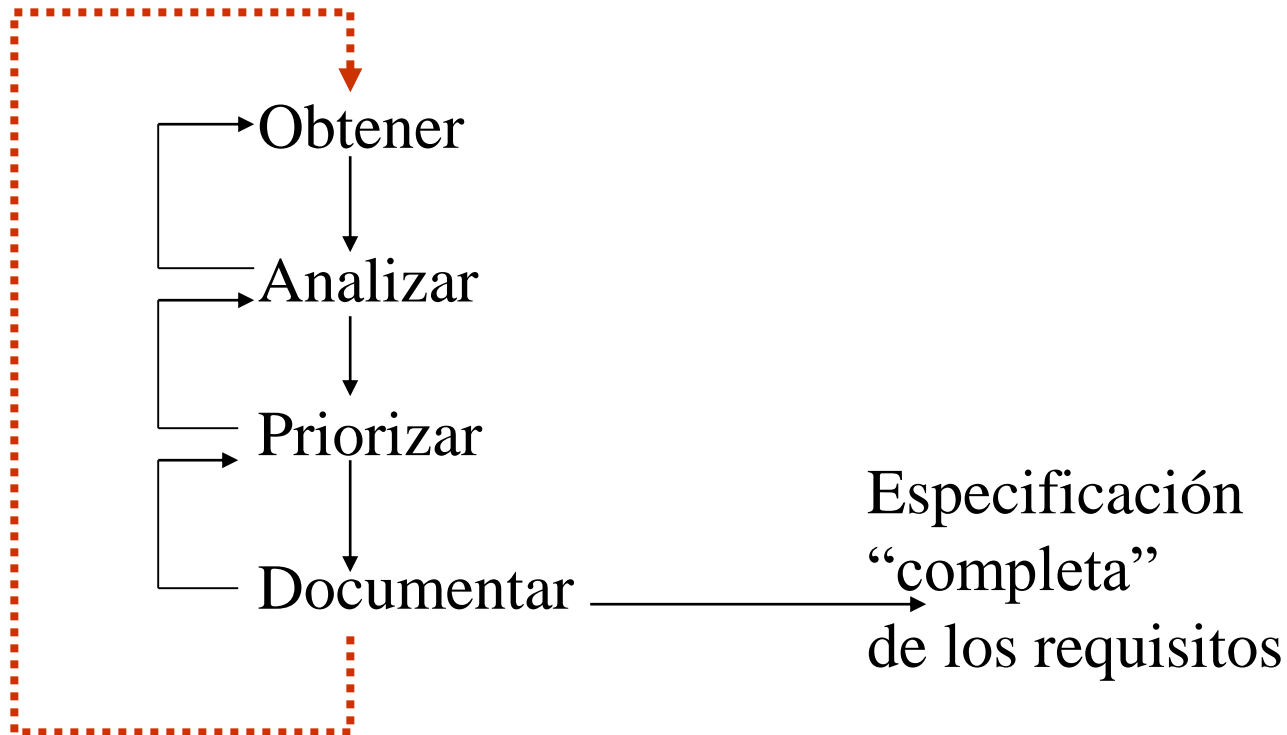
Casos de Uso

# Problemas que pueden presentarse:

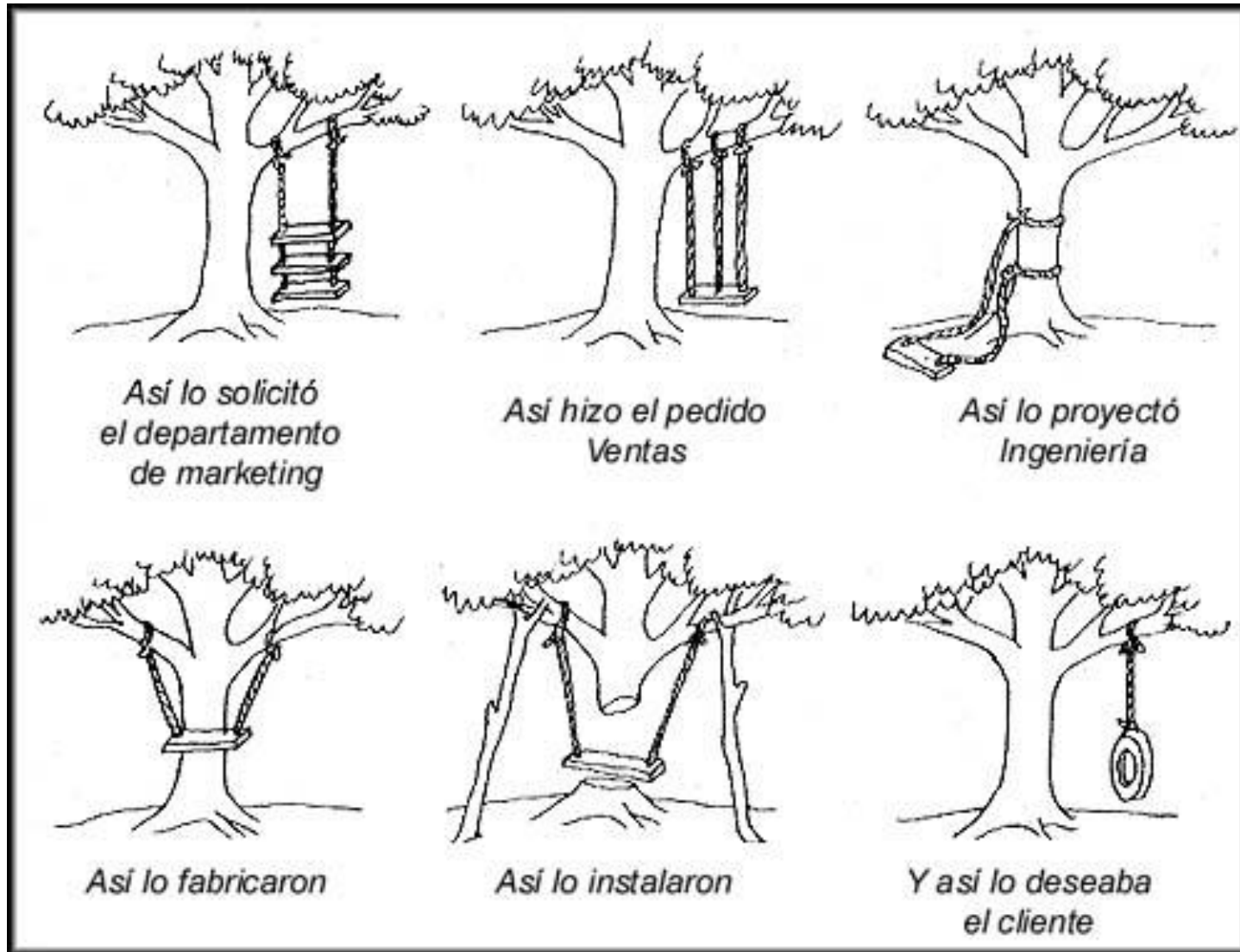
---

- Usuarios/clientes no saben lo que quieren del sistema, sólo en términos generales.
- Usuarios/clientes **no conocen el costo** de sus peticiones.
- Usuarios/clientes expresan los requisitos en sus términos y con **conocimiento implícito** de su propio trabajo.
- Distintos usuarios manifiestan distintos requisitos.
- *Influyen factores políticos.*
- La **prioridad** que se da a los requisitos **varía** con el tiempo.
- Se agregan/quitan **requisitos** a medida que se avanza en el proyecto.

# Manejo de los requisitos



# Que **no** nos pase...



# Modelo de Requisitos

---

Compuesto básicamente por:

- **Modelo de comportamiento (CU)**
- Modelo de presentación (o interfaces)
- Modelo de información (o del dominio del problema)

- *Modelo de Comportamiento*: describe la funcionalidad del sistema desde la perspectiva del usuario.
- *Modelo de Presentación*: describe cómo interactúa el sistema con actores externos.
- *Modelo de Información*: describe la estructura del sistema en términos de clases y objetos (o según el paradigma usado).



# Modelo de Casos de Uso (CU)

---

- Describe procesos de negocio.
- **Describe al sistema en términos de sus funcionalidades.**
- Cada CU representa una funcionalidad (lo inicia un actor y está formado por una secuencia de eventos).
- Deben reflejar los requisitos.
- Constituyen un formato simple y estructurado que permite a usuarios y desarrolladores trabajar juntos.
- No son de gran ayuda para identificar aspectos no funcionales.
- Se usan en las otras etapas del desarrollo.

# Unified Modeling Language (UML)

---

Permite crear diferentes tipos de diagramas:

1. **Diagramas de Casos de Uso (cap. 18)**
2. Diagramas de Clases
3. Diagramas de Objeto
4. Diagramas de Estado
5. Diagramas de Secuencia
6. Diagramas de Colaboración
7. Diagramas de Actividad
8. Diagramas de Componentes
9. Diagramas de Liberación

...

# Casos de Uso

---

- Técnica para entender y describir requisitos (funcionales).
- Énfasis en el uso del producto.
- Describen cómo el sistema debe comportarse desde el punto de vista del usuario.
- CU → como caja negra: especifican qué debe hacer el sistema, sin especificar cómo debe hacerlo.
- Se describen mediante documentos de texto.

# CU – Conceptos (1)

---

Un **escenario** es la descripción de qué sucede cuando alguien interactúa con el sistema. Debe dar un resultado de valor observable para un actor particular.

- Un escenario es *una instancia de* un CU. Puede verse como una forma particular de usar el sistema.
- Un **resultado de valor observable** ayuda a generar sistemas que hagan lo que las personas realmente necesitan.

# CU – Conceptos (2)

---

- Un CU es un conjunto de escenarios para una tarea (representa las distintas maneras que tiene un actor de relacionarse con el sistema).
- Los CU integrados forman el modelo de requisitos.
- Permite un desarrollo flexible y concurrente.

# CU – Elementos (1)

---

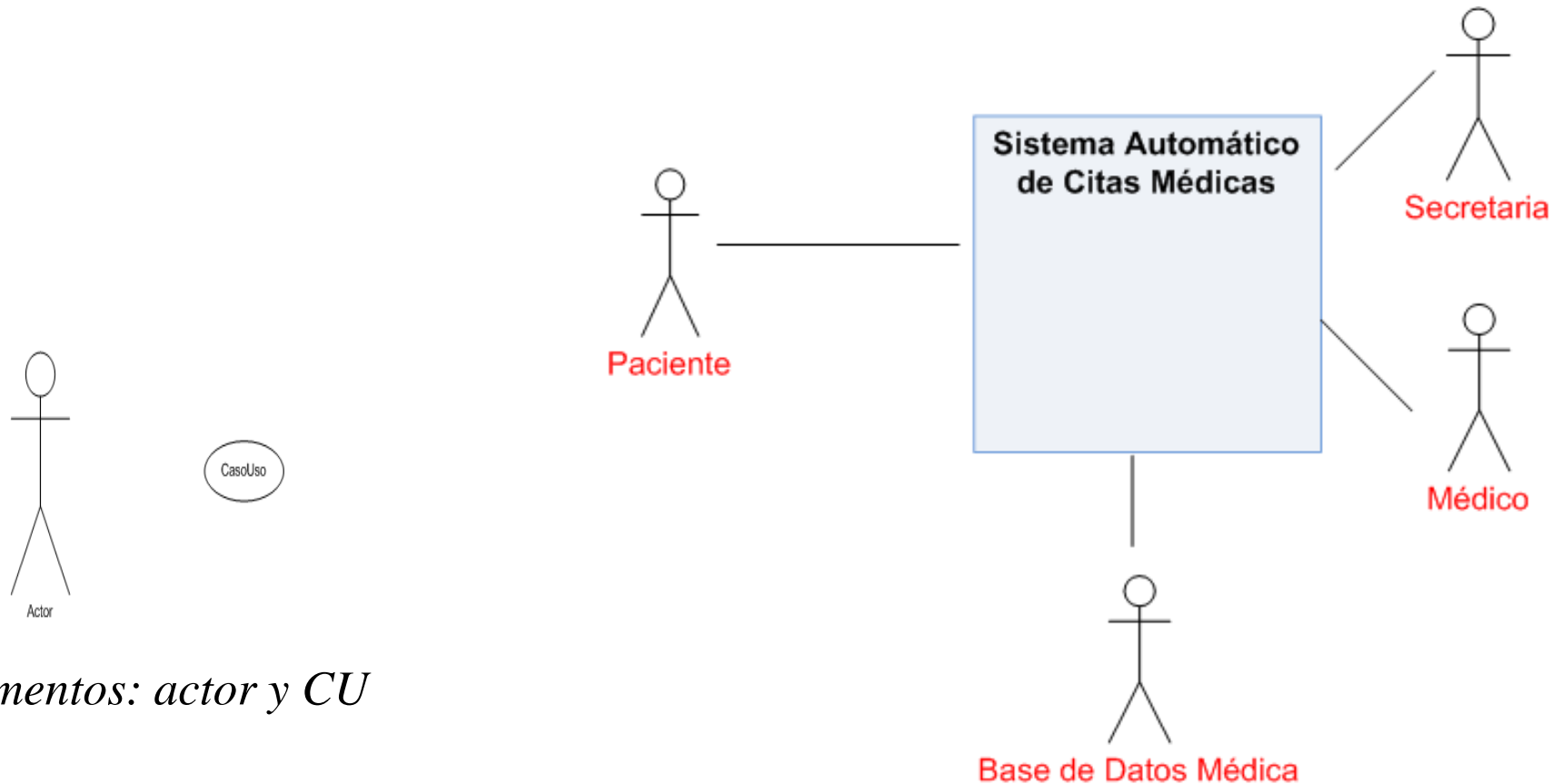
- El *actor* es quien o qué inicia la tarea.
- Es externo al sistema.
- Un actor es un rol que usuarios humanos, hardware externo u otros sistemas desempeña.
- Un *CU* puede estar comunicado con varios actores.
- Un actor puede estar involucrado en varios CU.
- Un actor es una clase de usuario.
- Los actores deben identificarse antes que los CU.

# CU – Elementos (2)

---

- Los actores se clasifican en: primarios y secundarios.
  - **Actores primarios**: determinan la secuencia lógica de ejecución del sistema. Pueden ser personas u otros sistemas. Inicia una interacción con el sistema → alcanzar un objetivo.
  - **Actores secundarios**: encargados del mantenimiento y supervisión del sistema o responden al sistema (pueden ser máquinas o sistemas.)
- **Delimitación del sistema**: representa al sistema como caja negra. Facilita identificar los actores.

# CU – Elementos/Ejemplo



*Elementos: actor y CU*

*Delimitación del sistema según los actores*



# CU: a tener en cuenta

---

- El actor primario inicia el CU.
- La comunicación entre actor y CU *generalmente* no lleva flecha (depende de la herramienta usada).
- El CU especifica la secuencia completa de eventos desde que el actor lo inicia hasta que se lleva a cabo la funcionalidad deseada.
- Nombre de CU: acción (verbo).
- Se identifican a partir de la descripción del problema, entrevistas con actores, observación, encuestas, etc.

# CU- Ejemplo

---

- Actor principal: Paciente.
- Caso de uso: Hacer cita.
- Descripción: *Un paciente entra al portal de un hospital para hacer una cita para su revisión anual.*



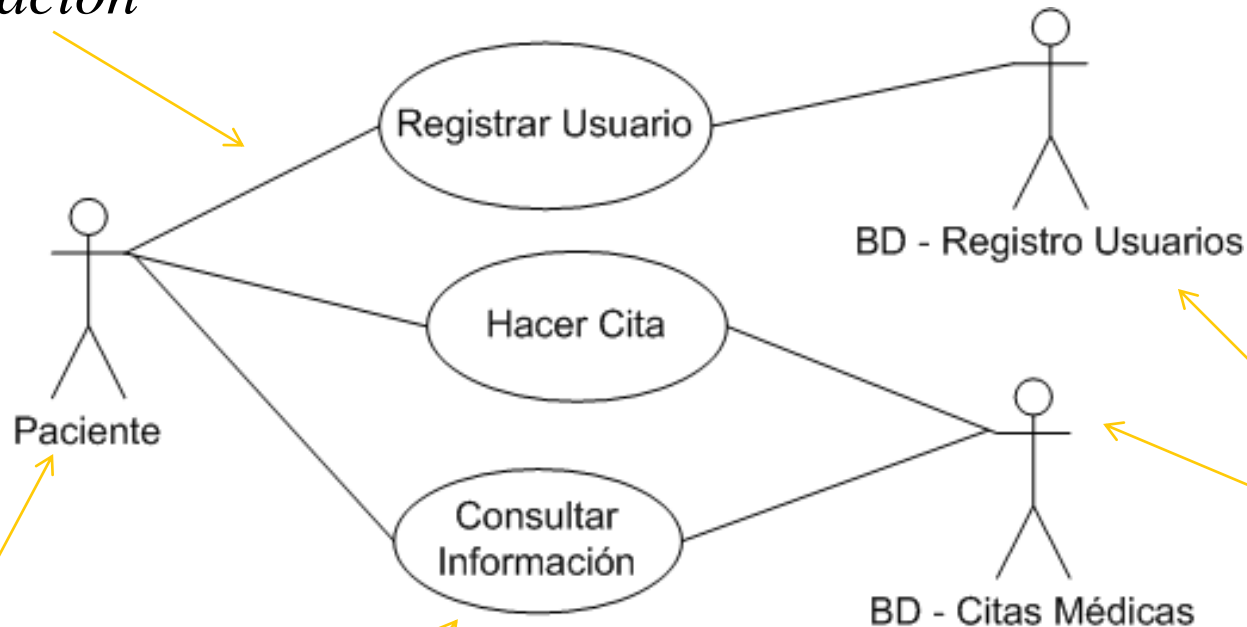
# Diagrama de casos de uso

---

- Un **diagrama de CU** es una colección de actores, CU y la comunicación entre ellos.
- Los CU modelan elementos *dinámicos* del sistema.
- El Diagrama de CU es una descripción estática.
- Señala los límites del sistema.
- Los CU son independientes del método de diseño que se utilice y, en consecuencia, del paradigma de programación. (Se asocian naturalmente con la OO).

# Diagrama de CU - Ejemplo

*Comunicación*



*Actor principal*

*Caso de uso*

*Actor secundario*

# Desarrollo del Caso de Uso: Hacer cita

---

## Flujo principal:

1. El PH pide nombre de usuario y contraseña.
2. El usuario ingresa nombre de usuario y contraseña.
3. El PH verifica que los datos son correctos.
4. El PH despliega opciones: consultas, hacer cita.
5. El usuario elige hacer cita.
6. El PH pide nombre del médico.
7. El usuario lo ingresa.
8. El PH verifica que nombre médico es correcto.
9. El PH despliega turnos disponibles.
10. El usuario elige día y hora.
11. El PH confirma turno.
12. El PH actualiza la BD-CitasMédicas.

# CU: Hacer Cita

---

## Flujos alternativos (o secundarios):

3A. Nombre de usuario o contraseña incorrectos

1. El PH despliega el mensaje “Datos incorrectos”.
2. Fin CU

8A. Nombre del médico incorrecto/incompleto

1. El PH despliega el mensaje “Nombre incorrecto”.
2. regresa a paso 6

8B. ...

...

Si la condición puede suceder en cualquier momento, se usa \*  
en lugar de enumerar. Por ejemplo:

\* *Se interrumpe la conexión*

# Caso de Uso: Hacer cita

---

## Flujo principal (otra representación)

Usuario	PH
	1. El PH pide nombre de usuario y contraseña.
2. El usuario ingresa nombre de usuario y contraseña.	
	3. El PH verifica que los datos son correctos.
	4. El PH despliega opciones: consultas, hacer cita.

# Relación entre CU

---

- Inclusión
- Extensión
- Generalización



# Inclusión

---

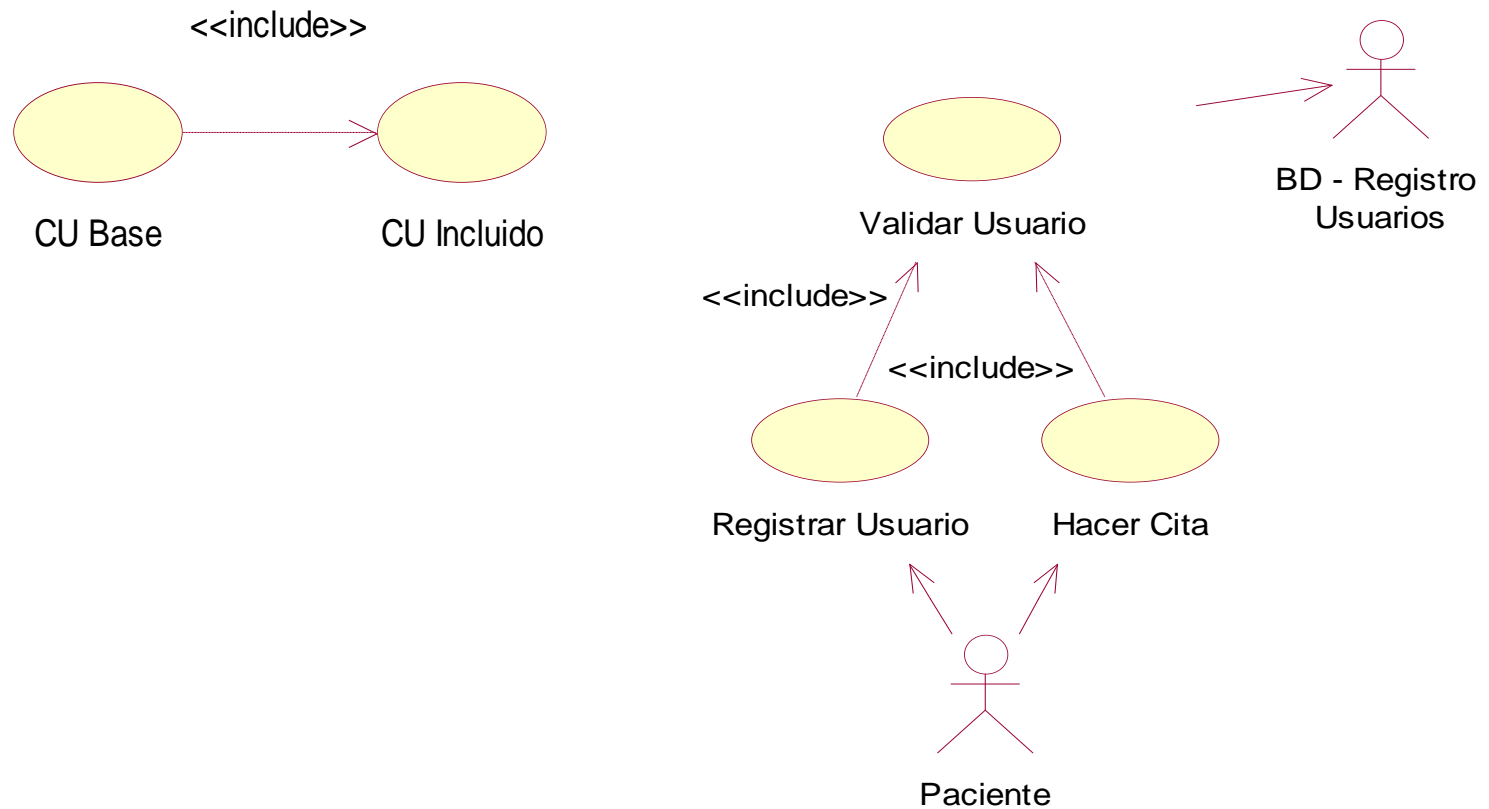
- Escenarios comunes a más de un CU.
- Representa la presencia obligada del CU incluido.
- El flujo del CU que se ejecuta sigue con el comportamiento descrito en el CU incluido y, al terminar éste, retoma el primero.
- El CU incluido representa comportamiento encapsulado que puede ser reusado.

# Inclusión (cont.)

---

- **Punto de inclusión:** punto del CU donde se **debe** insertar comportamiento adicional.
- En el texto se subraya el CU que se inserta.
- La relación se etiqueta con <<incluye>> (<<include>>).
- En OO se implementa por medio de asociaciones de clases.

# Inclusión - Ejemplo



# Extensión

---

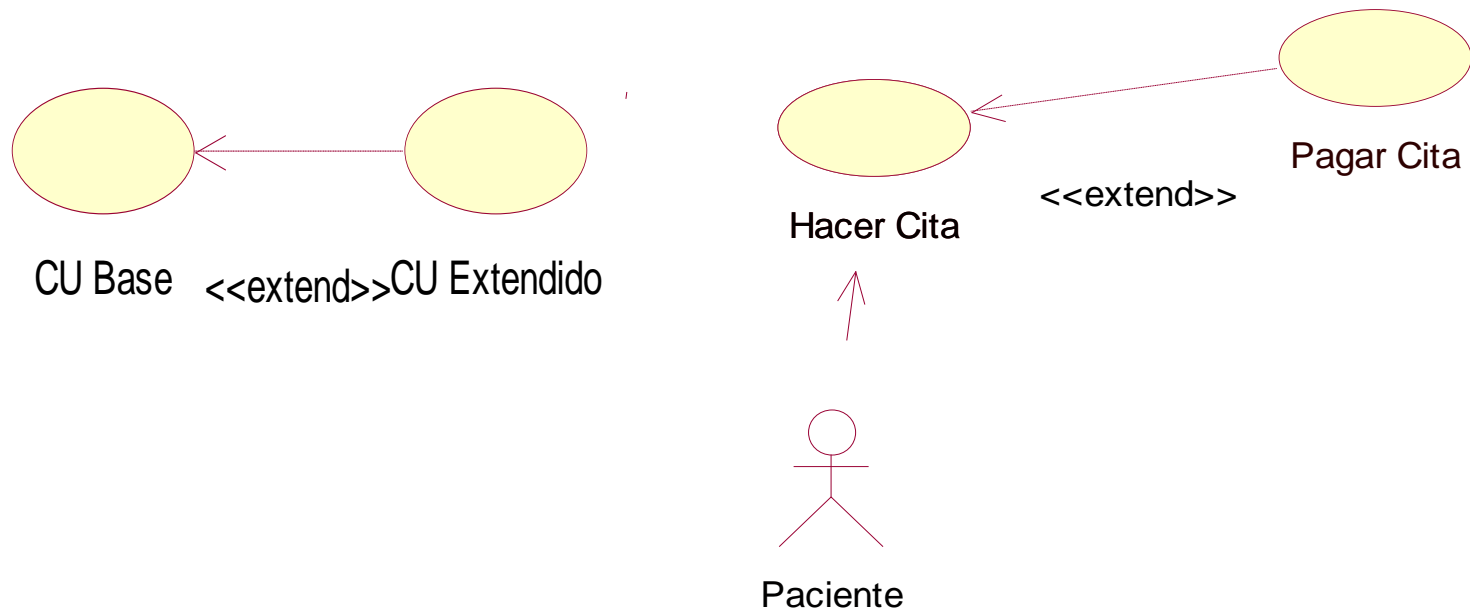
- Es un CU que agrega funcionalidad a otro CU.
- Facilita representar escenarios que serían difíciles de tratar como flujo alternativo.
- Permite destacar ciertos escenarios.
- Representa una funcionalidad condicional.

## Extensión (cont.)

---

- **Punto de extensión:** punto del CU donde se **puede** insertar comportamiento adicional.
- Al terminar el CU extendido, se vuelve al CU base, en la sentencia siguiente al punto de extensión.
- La relación se etiqueta con <<extiende>> (<<extend>>), la flecha apunta al CU base.
- En OO se implementa por medio de asociaciones de clases.

# Extensión - Ejemplo



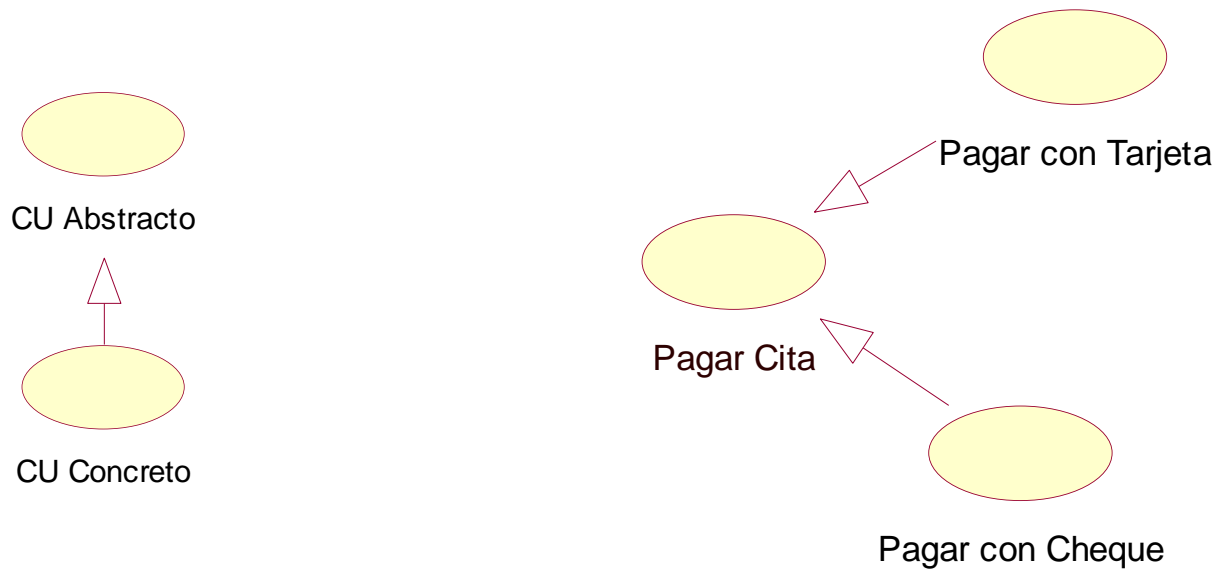
# Generalización

---

- Compartir funcionalidades.
  - Generalizar CU, favoreciendo el reuso.
  - CU abstractos: no se instancian.
  - CU concretos: sí se instancian.
- 
- En OO se implementa por medio de herencia de clases.

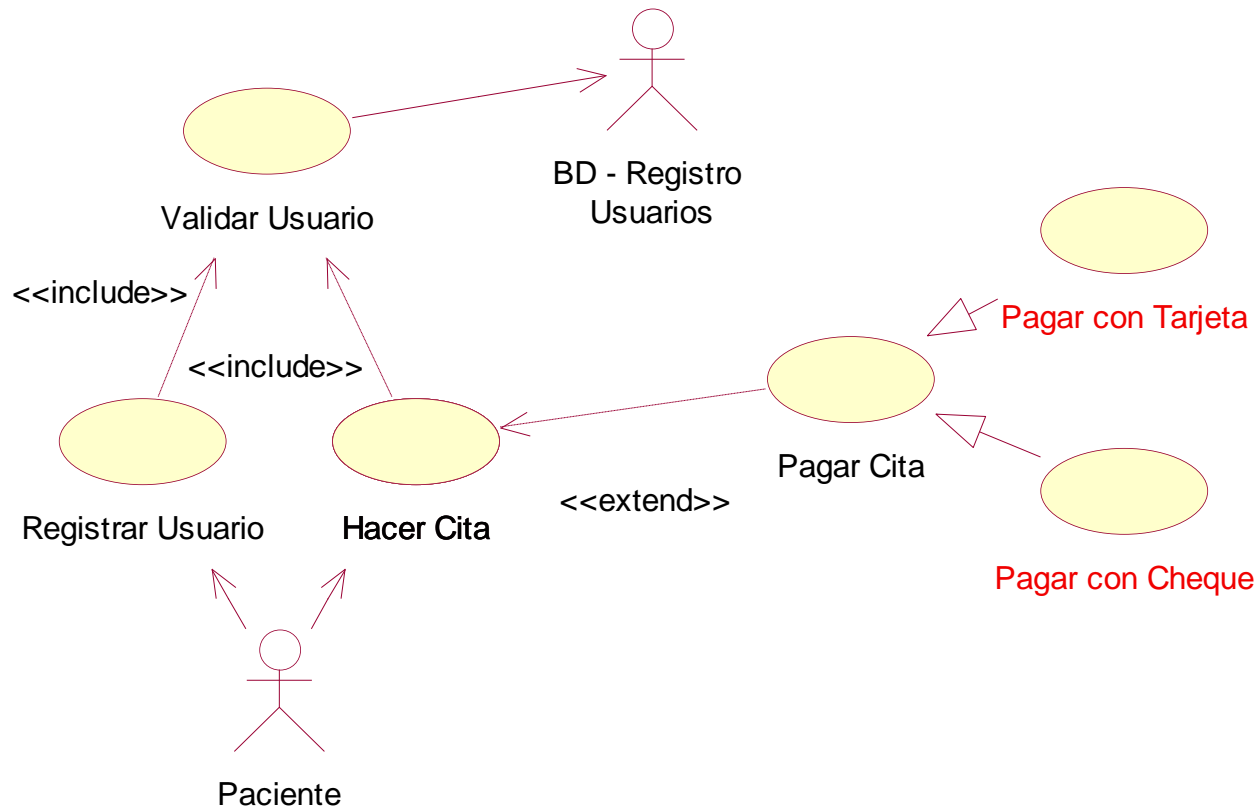
# Generalización - Ejemplo

---

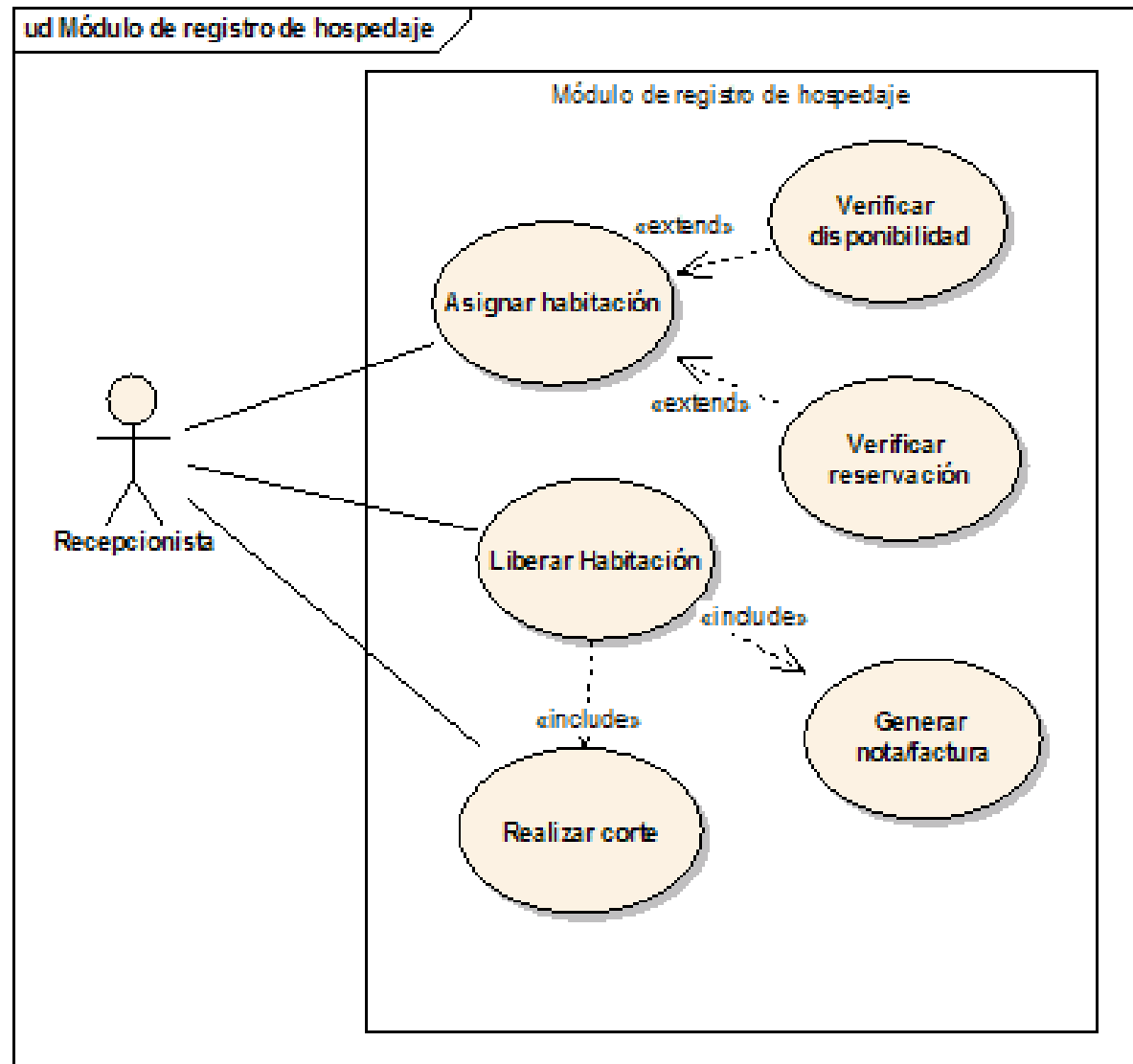




# Relaciones entre CU – Ejemplo 1



# Relaciones entre CU – Ejemplo 2



# Beneficios de desarrollar los CU

---

- ***Determinan requisitos***: al hacer los CU se pueden identificar nuevos requisitos y/o escenarios no contemplados.
- ***Favorece la comunicación con el cliente***: la notación, fácil de entender, constituye una excelente manera de comunicación entre desarrolladores y clientes.
- ***Facilita la definición de casos de prueba***: la colección de escenarios para los CU pueden sugerir un conjunto de casos de prueba para dichos escenarios.

# Construcción del Modelo (1)

---

- Definir frontera
- Identificar actores
- Para cada actor: identificar qué quiere hacer (cada acción será un CU, nombrar cada CU).
- Priorizar los CU

# Construcción del Modelo (2)

---

- Para cada CU:
  - Determinar si participan otros actores.
  - Documentarlo.
- Generar el diagrama de CU
- Definir los CU más importantes:
  - Revisarlos y refinarlos (proceso iterativo).
  - Identificar CU asociados (“incluye” y “extiende”) y generalización.

# Lista de objetivos por actor

---

Se genera una tabla con: actor, objetivo (tarea), prioridad

Sirve como punto inicial para la negociación entre los distintos involucrados.

Actor	Objetivo (tarea)	Prioridad
Cliente	Sacar dinero	1
	Consultar saldo	2
	Hacer pago servicio	3
Ejecutivo banco	Monitorear ATM	1
...		

# Participantes en esta etapa

---

- **Cliente y usuarios:** manifiestan los requisitos que deberán ser cubiertos (elementos obligatorios y elementos opcionales).
- **Analista/Diseñadores:** deben comprenderlos para poder diseñar una solución adecuada.
- **Supervisores del Contrato,** establecen: hitos de control, cronogramas, etc.
- **Gerentes del negocio:** entienden el impacto en la organización.
- **Verificadores:** deben comprenderlos para poder diseñar pruebas para verificar si el sistema los satisface.

# Documentación del CU

---

1. El nivel de documentación depende de la etapa en la que se esté (resumido, pensando en el usuario, etc.) y del destinatario.
2. Se debe mantener un lenguaje simple.
3. Se puede combinar texto, con descripción de actividades, máquinas de estado, etc.
4. Se pueden tener distintas plantillas y elegir la que más se ajusta al proyecto actual.

[Plantilla y ejemplos de CU.](#)



# Ejemplo de plantilla Casos de Uso.doc

---

1. **Nombre del CU:** nombre dado al CU (verbo, relacionado a la acción que define).
2. **Objetivo:** definir brevemente el objetivo del CU.
3. **Precondiciones:** establecen las condiciones que deben cumplirse antes de iniciar un CU. Por ejemplo: el usuario ha ingresado y ha sido validado.
4. **Descripción:** basada en el flujo principal.
5. **Flujo principal:** describe el escenario del CU de mayor interés para el actor.
6. **Flujos alternativos:** son todos los otros escenarios (bifurcaciones del flujo principal).
7. **Postcondiciones:** establecen las condiciones que deben cumplirse al completar el CU.

# Preguntas

---

1. ¿Qué es un caso de uso?
2. ¿Qué elementos componen un CU?
3. ¿Qué es lo primero que se debe identificar en un CU?
4. ¿Para qué sirve un diagrama de CU?
5. ¿Qué relaciones se pueden definir entre CU?
6. Características de cada una, diferencias, ejemplo.
7. ¿Los actores son internos o externos al sistema?

# Ejercicios

---

En clase: resolver la lista de ejercicios

**Proyecto:** en el proyecto elegido, en equipo, definir los CU. A tener en cuenta:

- Descripción del problema
- Diagrama de CU: límites del sistema, CU, actores, relación entre CU.
- Priorizar los CU.
- Desarrollo de los CU (texto).
- Pueden usar la plantilla proporcionada como base para el documento a entregar.
- **Fecha de entrega:** pendiente