

# Algunas Herramientas de Apoyo al Análisis y Diseño de Software

Agustín J. González

ELO329: Diseño y programación orientados a  
objetos

# Resumen

- Para desarrollar software hay varias herramientas de gran utilidad independientemente de la metodología usada en el desarrollo.
- Veremos:
  - Descripción de casos de uso del sistema,
  - Tarjetas CRC (**C**lase, **R**esponsabilidad y **C**olaboradores,
  - Diagramas UML (Unified Modeling Language)

# Casos de Uso del sistema

- Recordemos las principales actividades del desarrollo: Definición de requerimientos, análisis, diseño, implementación, pruebas, distribución.
- Se requiere de técnica para capturar información sobre cómo un sistema o negocio trabaja (o se comporta), o sobre cómo se desea que trabaje o comporte
- El estudio de **casos de uso** es una técnica que sirve tanto para definir **requerimientos** como de **análisis**. También es de utilidad durante las **pruebas**.
- Cada caso de uso se concentra en un escenario específico y describe la interacción entre un **actor** principal y el **sistema**.
- Caso de uso = Contexto + Secuencia de acciones
  - Acción = Interacción entre actor(es) y el sistema bajo desarrollo.

# Casos de Uso (cont.)

- Actores pueden ser usuarios, otros sistemas, hardware, todos fuera del sistema de software que estamos definiendo.
- Cada resultado (salida) tiene un valor para uno de los actores
- Se usa variaciones para situaciones excepcionales.
- Se construye en base a levantamiento inicial de requerimientos y en reuniones de análisis con usuarios.
- Un caso de uso debe ser simple, claro y conciso. No debe dar lugar a especificaciones ambiguas.

# Casos de Uso (cont.)

- Se pueden presentar gráficamente usando Diagramas UML de Casos de Uso.
- Son buenos para capturar requerimientos reflejados en comportamientos del sistema ante determinadas entradas (son los requerimientos funcionales).
- No son buenos para requerimientos no funcionales. Ej.: Plataforma, desempeño, seguridad, tiempo de respuesta.
- Cada caso de uso debe ser entendible para el cliente.

# Casos de Uso (cont.)

- Se pueden presentar gráficamente usando Diagramas UML de Casos de Uso.
- Son buenos para capturar requerimientos reflejados en comportamientos del sistema ante determinadas entradas (son los requerimientos funcionales).
- No son buenos para requerimientos no funcionales. Ej.: Plataforma, desempeño, seguridad, tiempo de respuesta.
- Cada caso de uso debe ser entendible para el cliente.

# Plantilla para Casos de Uso

- Ésta es una de varias posibles.
- **Nombre:** Nombre del caso de uso, usar verbo y sustantivo, debe sintetizar el objetivo deseado. Ej. Cambiar nota
- **Propósito:** Resume brevemente qué se desea lograr con este caso de uso.
- **Actores:** Entes externos que participan en el caso de uso.
- **Pre-condiciones:** Pre-requisitos existentes (que se prevén) para el correcto funcionamiento de la funcionalidad especificada en el caso de uso.

# Plantilla para Casos de Uso

- **Evento:** Situación que gatilla el inicio del caso de uso.
- **Pos-condiciones:** Situación que ocurre después de usar la funcionalidad especificada en el caso de uso
- **Tipo:** Manual o automático, Ej.: Un timer expira, es de tipo automático.

# Plantilla para Casos de Uso

- **Curso Normal de Eventos** (o detalle): aquí se describe una secuencia numerada de pasos relatando el flujo básico o feliz del caso de uso. Se sugiere separar en dos columnas:

Actor	Sistema
1)	
	2)

- **Curso Alternativo de eventos:** funcionalidad que se requiere en caso de error.

# Plantilla para Casos de Uso

- **Requerimientos no funcionales:** Especificación narrativa de solicitudes no funcionales del usuario que especifican situaciones de rendimiento, volúmenes de información, seguridad, tiempos de respuesta, etc.
- **Autor:** Persona(s) responsable del análisis y redacción del caso de uso.

# Ejemplo de caso de uso: Sistema de mensajes de voz en teléfono

- Nombre: Dejar un mensaje
- Propósito: El llamador desea **dejar** un **mensaje** en una **casilla** de voz.
- Actor: llamador
- Pre-condición: Existe la casilla buscada.
- Evento: El llamador llama a una casilla.
- Post-condición: El mensaje queda grabado en casilla.
- Tipo: manual

# Ejemplo de caso de uso: Sistema de mensajes de voz en teléfono.

## ■ Curso Normal de Eventos:

Actor	Sistema
1. El <b>llamador</b> marca el número principal del sistema de mensaje de voz.	2. El <b>sistema</b> responde con un mensaje hablado pidiendo: Ingrese el número de la casilla seguido por un signo #.
3. El usuario marca el número de la extensión.	4. El sistema le habla: Usted se ha contactado con la <b>casilla</b> xxxx, Por favor deje su <b>mensaje</b> ahora.
5. El llamador deja el mensaje.	
6. El llamador cuelga.	7. El sistema pone el mensaje en la casilla.

# Ejemplo de caso de uso: Sistema de mensajes de voz en teléfono.

- Curso Alternativo de eventos
- Es común especificar variantes de un caso de uso:
- Variante 1:
  - 3A1. El usuario ingresa un número de extensión inválido.
  - 4A1. El sistema de mensaje de voz responde:
    - Usted ha marcado un número de casilla inválido.
  - 5A1: Continúa con paso 2.
- Variante 2
  - 5A2. El usuario cuelga en lugar de dejar un mensaje.
  - 7A2. El sistema de mensaje de voz descarta el mensaje vacío.

# Tarjeta CRC: Class, Responsibilities, Collaborators

# Tarjeta CRC: Class, Responsibilities, Collaborators.

- Es una herramienta principalmente de **diseño**.
- Creamos una tarjeta por cada clase (fijarse en sustantivos en casos se uso)
- El nombre de la clase va en la parte superior.
- Responsabilidades a la izquierda y
  - 1-3 responsabilidades (fijarse en verbos en casos de uso)
- Colaboradores a la derecha.
  - Colaboradores de la clase, no de cada responsabilidad.

<Nombre de clase>	
<Responsabilidades>	<Colaboradores>
.....	.....

# Ejemplo tarjeta CRC:

- Típicamente los **sustantivos** de los casos de uso son una buena pista para encontrar **candidatos a clases**.
- Los **verbos** de los casos de uso son **candidatos a responsabilidades (métodos)**.

Mailbox	
<i>manage passcode</i>	MessageQueue
<i>manage greeting</i>	
<i>manage new and saved messages</i>	

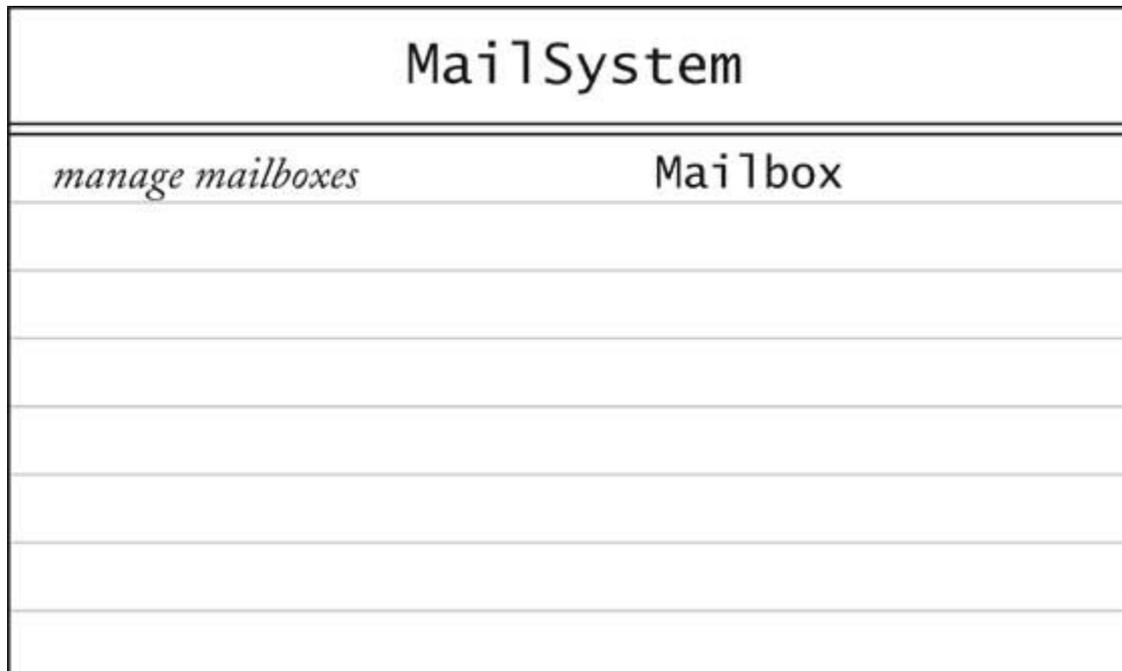
(Casilla)

# Recorrido de Caso de Uso

- El recorrido de los casos de uso permite identificar otras clases.
- Caso de uso: Dejar un Mensaje
- Llamador se conecta al sistema de mensajería.
- Llamador marca extensión.
- “Alguien” debe ubicar la casilla (Mailbox).
- Ni la casilla ni el mensaje pueden hacer esto.
- Surge una **nueva clase**: SistemaMensajeria (MailSystem).
- Responsabilidad: Administrar las casillas.

# CRC inicial para: SistemaMensajeria

- Usar los casos de uso para llenar las tarjetas CRC.
- Cambiar las tarjetas a gusto. Es común hacer cambios al considerar nuevos casos de uso.
- Lo común: el primer diseño no es el perfecto.



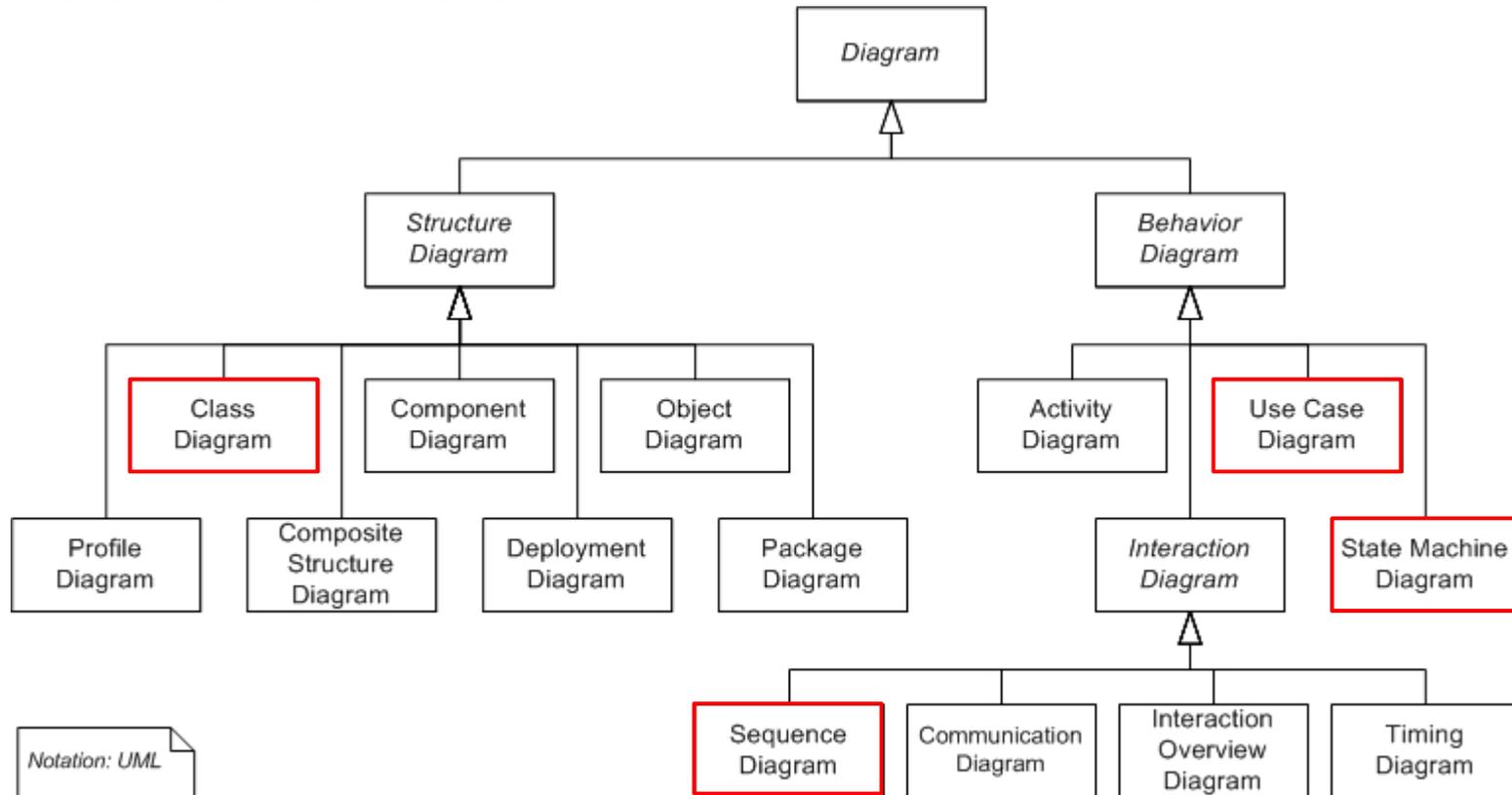
(SistemaMensajeria)

# Diagramas UML

- UML= Unified Modeling Language
- Hay varios tipos de diagramas.
- Nosotros veremos sólo cuatro tipos:
  - Diagrama de Casos de Uso
  - Diagrama de Clases (ya conocen algo)
  - Diagrama de Secuencia (ídem)
  - Diagrama de Estados

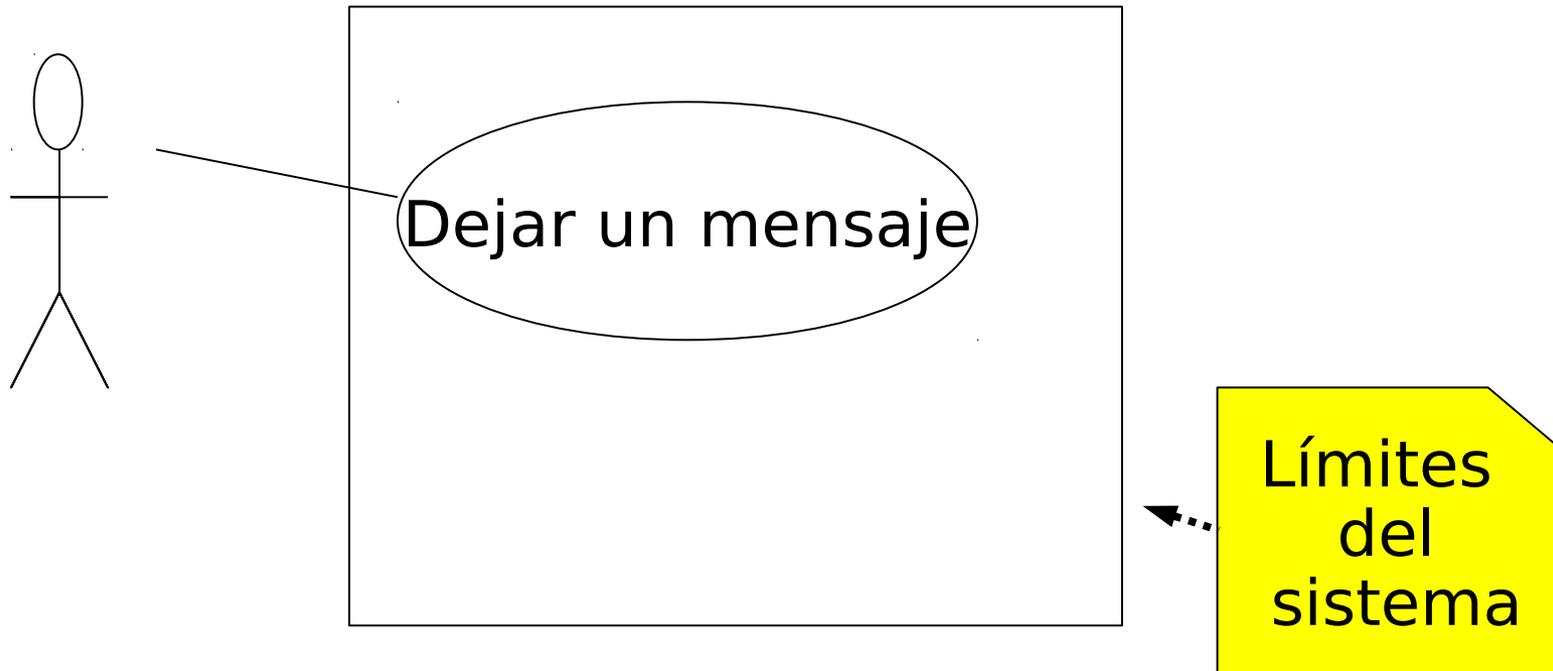
# Tipos de Diagramas UML

- Usaremos sólo 4



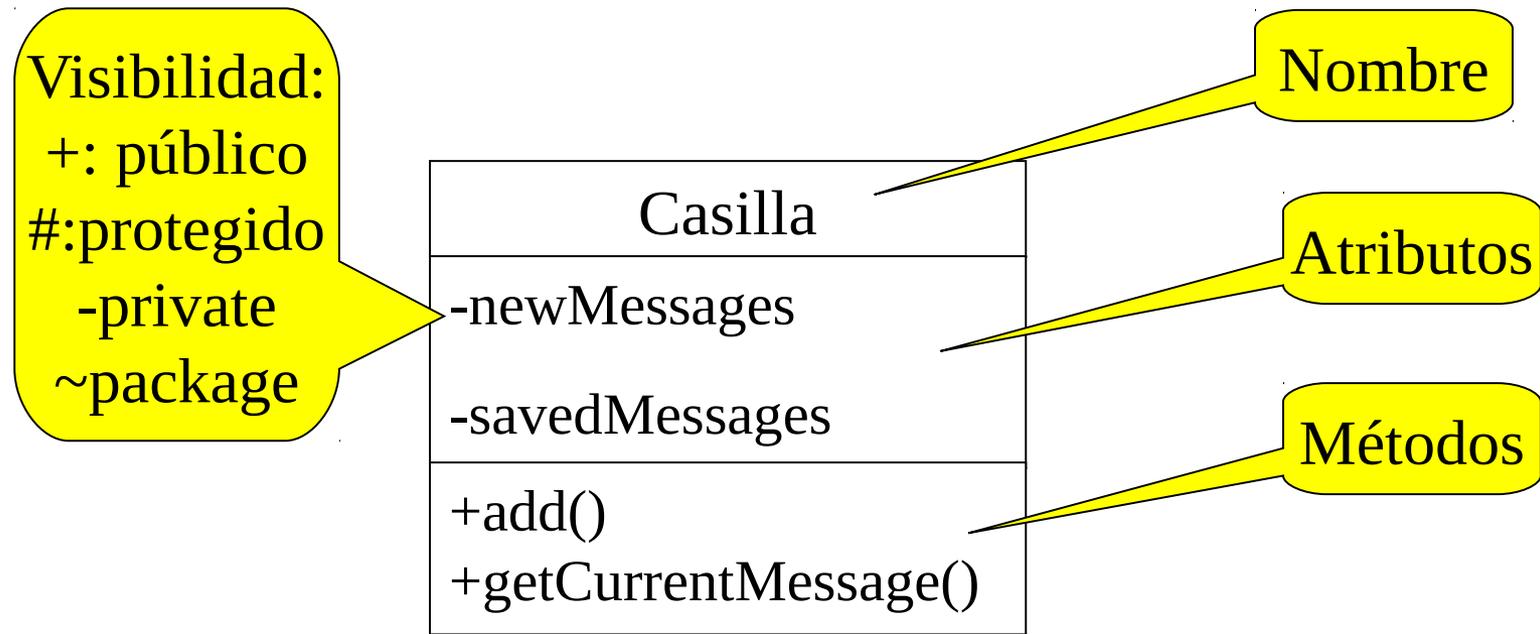
# Diagrama de Casos de Uso

- Su objetivo es presentar gráficamente una funcionalidad provista por el sistema.



# Diagrama de Clases

- Cada clase es representada por:



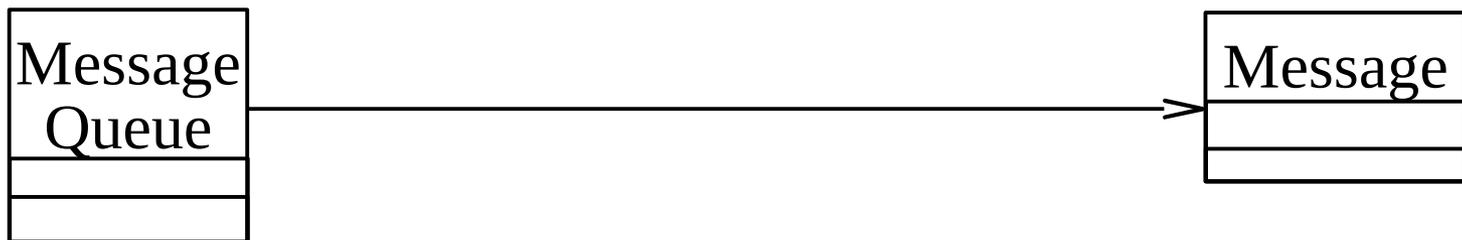
- La representación varía según la herramienta usada.

# Tipos de Relaciones entre clases

- **Asociación:** es la relación más general. Resenta una familia de relaciones, la asociación puede ser unidireccional o bidireccional. Puede tener roles, algunos la usan a cambio de agregación.



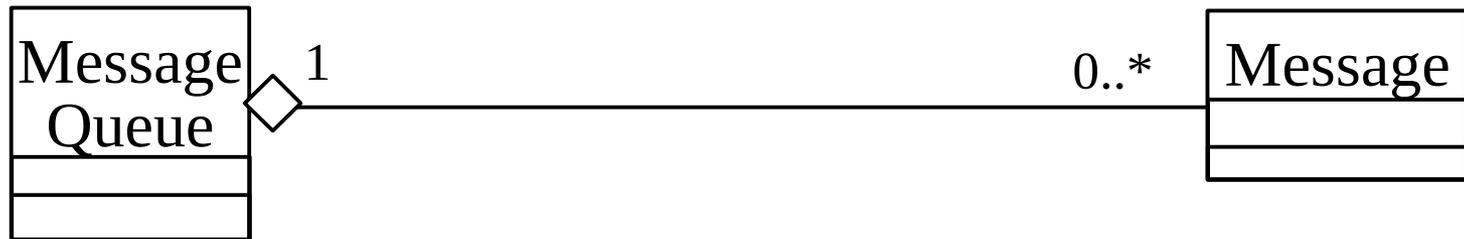
- Para indicar direccionalidad se usan flechas



- Por ejemplo un mensaje no sabe en qué cola de mensajes está.

# Tipos de Relaciones entre Clases (cont)

- **Agregación:** relación “tiene” o “contiene”, la parte puede existir fuera del todo.



- **Composición:** Caso especial de agregación. Contenido no existe fuera de la clase. La parte sólo existe en la medida que el todo exista.

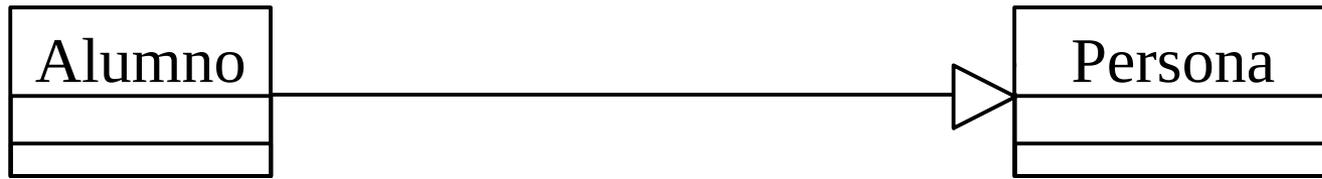


# Diferencia entre Agregación y Composición

- La composición es más específica.
- Por ejemplo:
  - Un comité tiene miembros. Es agregación porque los miembros existen por sí solos incluso fuera de ese comité o en otro comité.
  - Una Familia tiene hijos. Es composición porque los hijos por sí solos no tienen sentido, provienen de una pareja. Si carecemos de familias, no tendremos hijos.
  - ¿Qué relación habría entre equipo de fútbol y jugador?

# Tipos de Relaciones entre Clases (cont)

- **Herencia:** Cuando se cumple la relación es-un y además hay una relación de sub-tipo válida. Una clase extiende la otra.



- **Interfaces:** Describe un conjunto de métodos.
- No hay estado ni implementación.



# Tipos de Relaciones entre clases (cont)

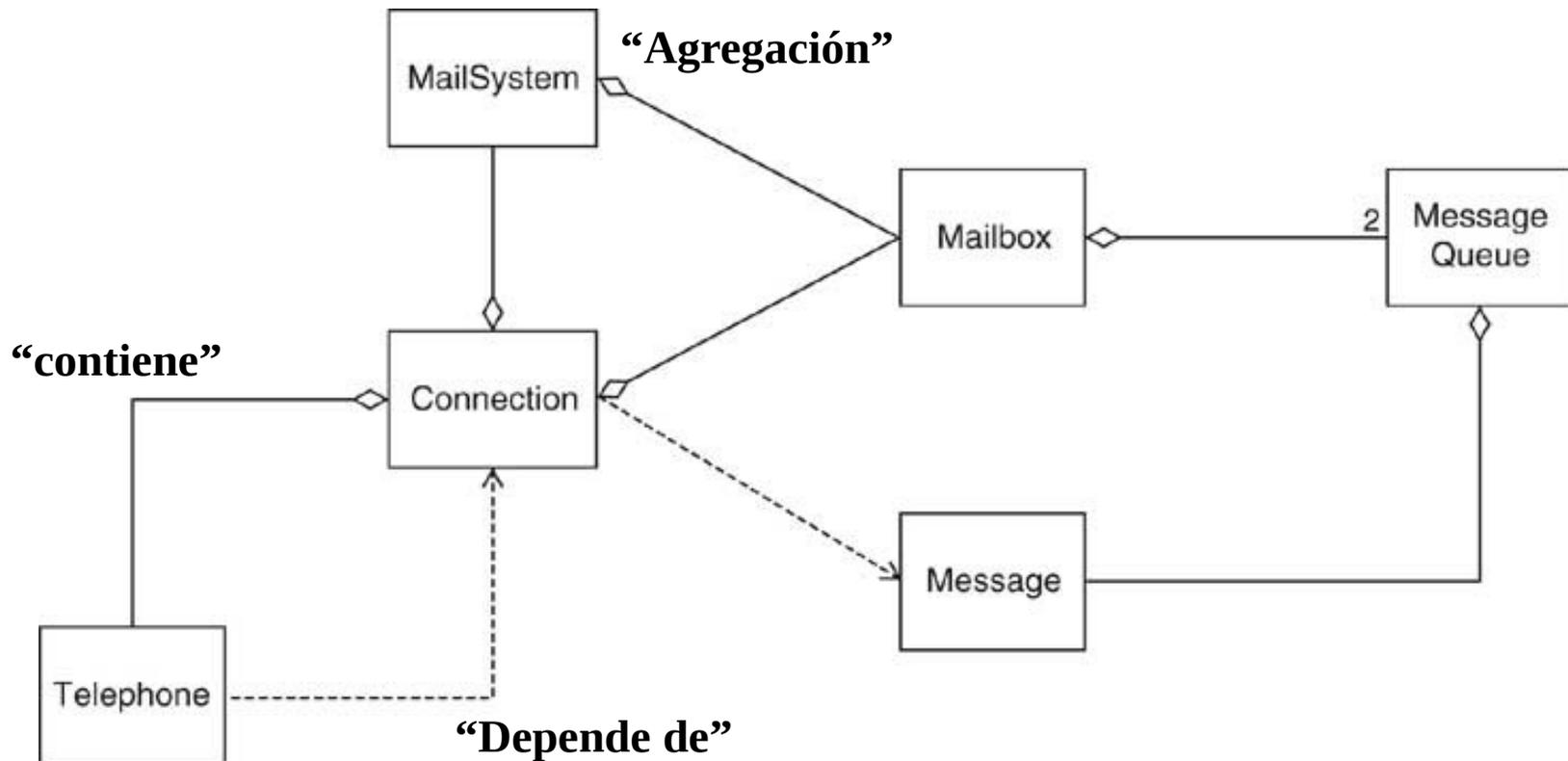
- **Dependencia:** Es la más débil de las asociaciones. Indica que una clase usa otra en algún momento. Existe dependencia si una clase aparece en un parámetro o variable local de un método de la otra.



- **Recomendaciones**
- Usar UML para informar, no para impresionar.
- No dibujar un único diagrama sobrecargado.
- Cada diagrama debe tener un propósito específico.
- Omitir detalles no esenciales.

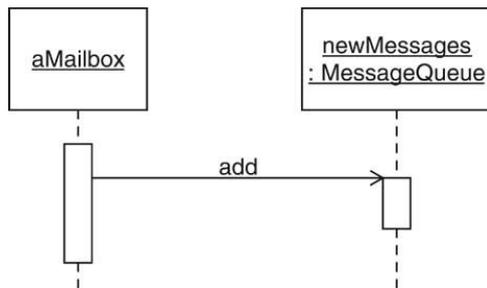
# Diagrama de clases para sistema de mensajería

- A esto se llega luego de analizar varios casos de uso y construir las tarjetas CRC para cada clase.



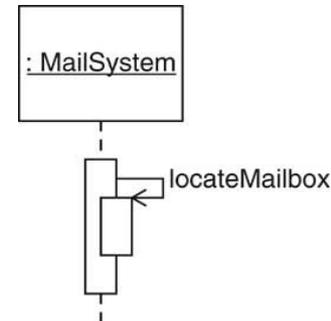
# Diagrama de Secuencia

- Cada diagrama *muestra la dinámica* de un escenario.

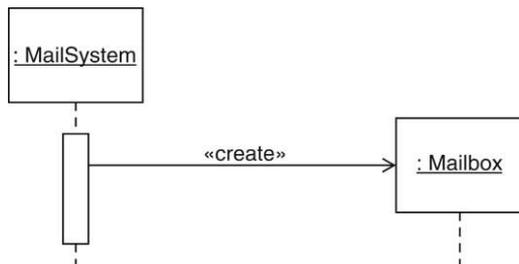


- Incorporación de un nuevo mensaje.

- Ubicación de casilla

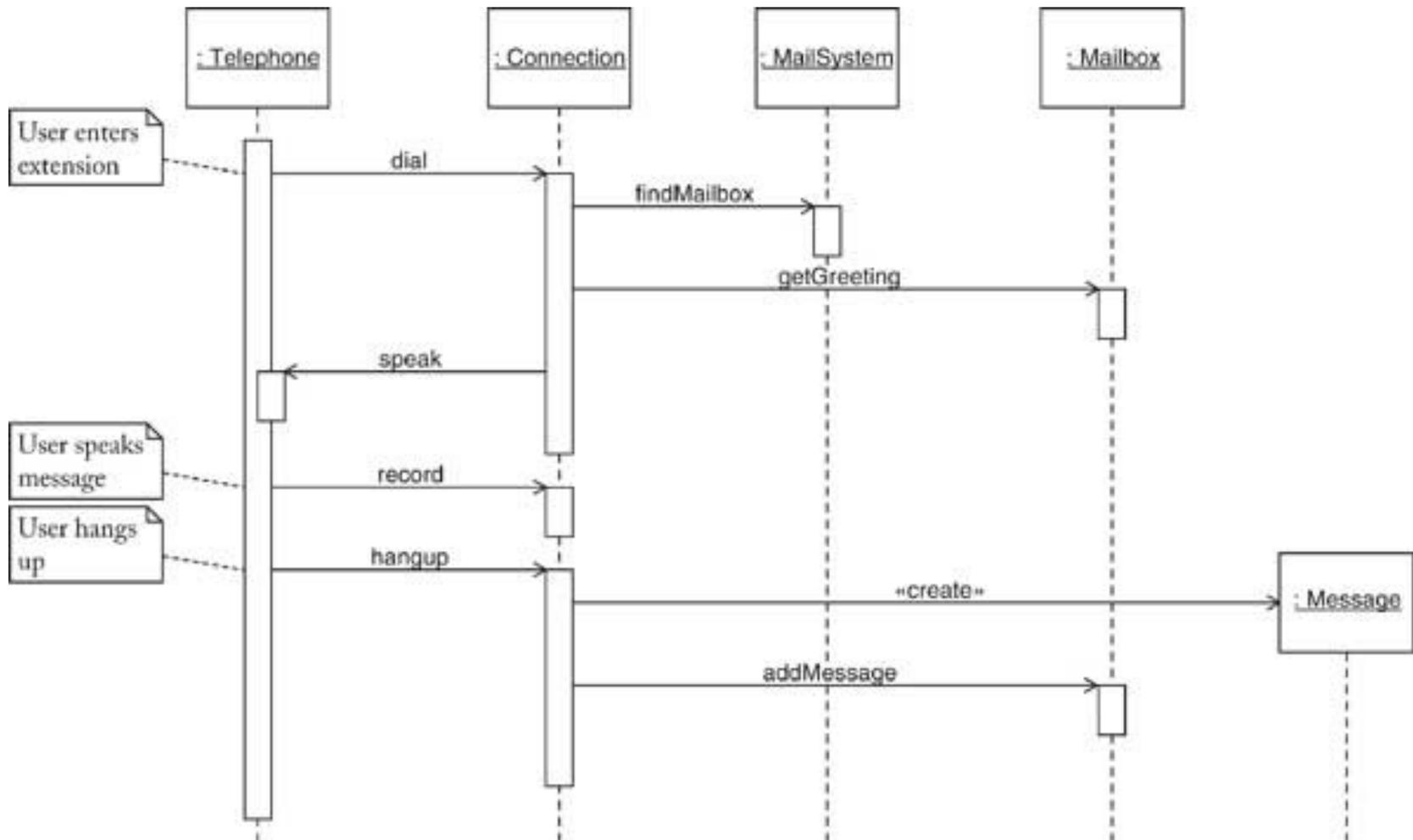


Objeto  
:Clase



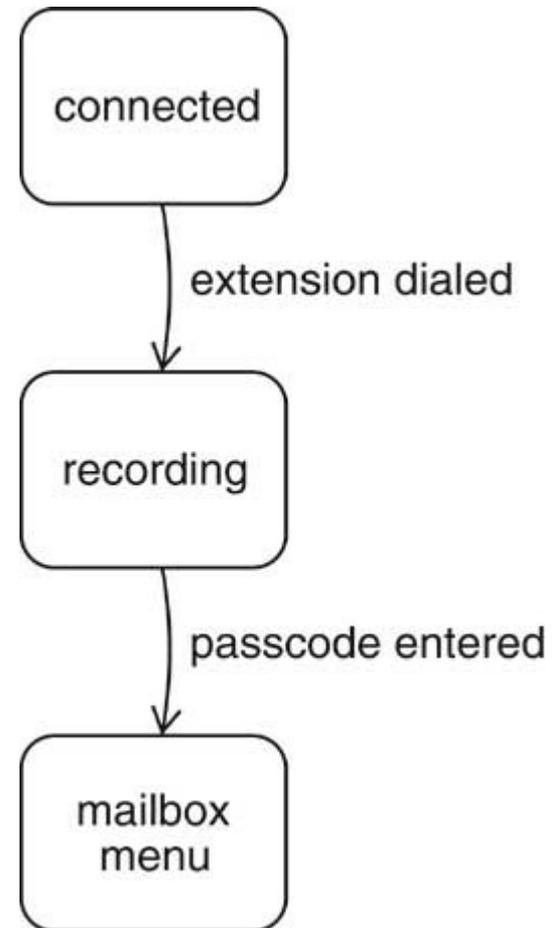
- Creación de una casilla

# Diagrama de secuencia para: “Dejar un Mensaje”

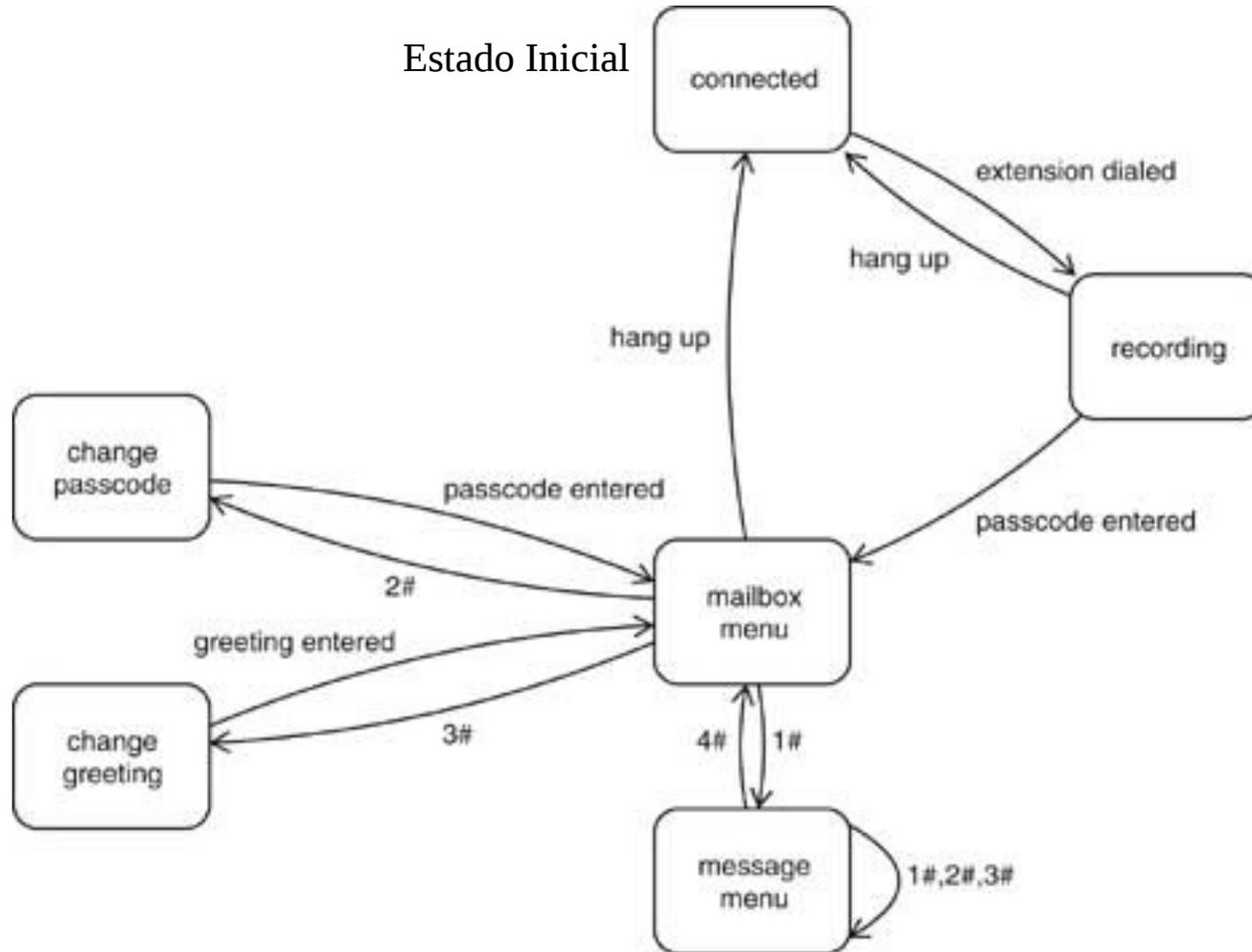


# Diagrama de Estados

- Son utilizados en las clases cuyos objetos tiene estados de interés.
- Similares a los diagrama de estados que verán o vieron en “Sistemas Digitales”.



# Diagrama de Estado para la conexión del dueño de la casilla



# Herramientas de software

- Varios IDE (Integrated Development Environment) incluyen facilidades para crear estos diagramas; por ejemplo:
- **Jgrasp** lo provee, en su versión estándar.
- **Eclipse** no lo provee directamente, pero se puede incluir agregando un plug-in para ese propósito
- **NetBeans** incluye un módulo que lo permite, para ello se debe bajar la versión completa e instalar lo necesario. Cuando las clases ya se tienen, se usa
  - Botón derecho en proyecto->reverse engineer
  - Luego en Model: seleccionar las clases a diagramar
  - Botón derecho en las clases->Create diagram from selected elements.
- También pueden revisar Umbrello.