EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Especialidad Análisis, diseño y desarrollo de software

Análisis Avanzados de Software

D1.4 Diagramas de modelado unificado UML

Asesor: M.T.I.C. Leonardo Enriquez

Ingeniero electrónico, sistemas digitales

Diagramas para el modelado

El UML (Lenguaje Unificado de Modelado) es una herramienta que permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas.

Un analista es quien documenta el problema del cliente y lo comunica a los desarrolladores que son los programadores que generan el programa que resolverá el problema y lo distribuirán en equipos de computación.

La finalidad de los diagramas es presentar diversas perspectivas de un sistema a las cuales se les conoce como modelo. El UML describe lo que hará un sistema pero no como implementarlo.

Diagramas estructurales:

- Diagramas de clase.
- Diagramas de objetos.
- Diagramas de casos de uso.

Diagramas de comportamiento:

- Diagramas de estado.
- Diagramas de secuencia.
- Diagrama de colaboraciones o comunicación.
- Diagrama de actividades
- Diagramas de componentes.
- Diagramas de distribución o despliegue

Otras características:

- Paquetes
- Notas
- Estereotipos

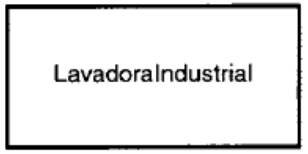
Diagrama de clases (características y representación)

Diagrama de clase

- Se pueden utilizar cuando se desarrolla un modelo de sistema orientado a objetos para mostrar las clases en un sistema y las asociaciones entre dichas clases.
- Una clase es una categoría o grupo de cosas que tienen atributos y acciones similares.
- Una asociación es un vinculo entre clases, que indica que hay una relaciones entre dichas clases.
- Es un diagrama estático.

Representación de un diagrama de clase

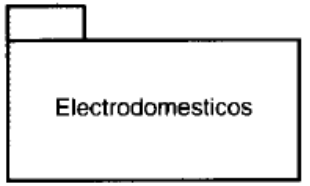
- Un **rectángulo** es el símbolo que representa una clase.
- El nombre de la clase es por convención, una palabra con la primera letra en **mayúscula** y normalmente se coloca en la parte superior del rectángulo. Si el nombre de su clase consta de dos palabras, únalas e inicia cada una con mayúscula.



Representación UML de una clase

Representación de un paquete de clases

- Un paquete del UML juega un papel en el nombre de la clase. Un **paquete organiza diagrama de elementos** y se representa como una carpeta tubular cuyo nombre es una cadena de texto.
- Si una clase es parte de un paquete se podrá dar el nombre Paquete:clase



Paquete del UML

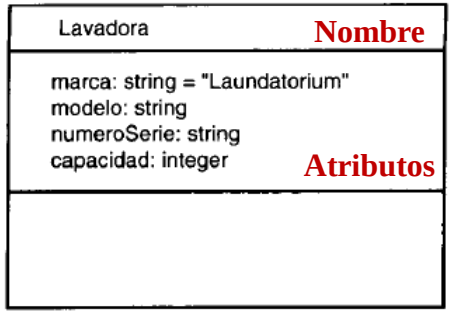
Una clase con un nombre de ruta

Diagrama de clases (Atributos, métodos y funciones)

Atributos de un diagrama de clase

- **Atributos (propiedades)**. Es una **característica de una clase y describe un rango de valores** que la propiedad podrá contener en los objetos, pudiendo contener varios atributos o ninguno.
- Los atributos tienen tipos como cadena (string), flotantes (float), entero (integer) y booleano (boolean).

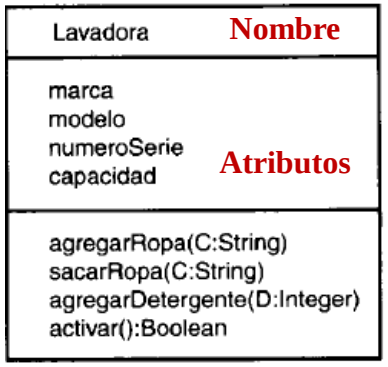
Para indicar un tipo, se utiliza dos puntos **(:)**



Un atributo puede mostrar su tipo así como su valor predeterminado

Operaciones y funciones de un diagrama de clase

- **Operaciones (métodos)**. Una operación es algo que la clase puede realizar, y su nombre se escribe en **minúscula** si consta de una sola palabra, y si el nombre consta de mas palabras se unen e inicia todas con mayúsculas a excepción de la primera.
- La **función** es un tipo de operación que devuelve un valor al finalizar su trabajo, la cual podrá mostrar el tipo de valor que regresara.



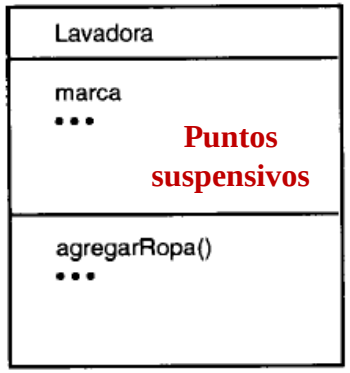
La lista de operaciones de una clase aparece debajo de una línea que las separa de los atributos de la clase

Operaciones (métodos y funciones)

Diagrama de clases (estereotipo)

Puntos suspensivos

- En la practica no siempre se mostrara todos los atributos y operaciones de una clase, sin embargo se pueden mostrar algunos y colocar **tres puntos suspensivos** (…) para omitir ciertos o todos los **atributos** y **operaciones**.



Los puntos suspensivo indican atributos u operación que no se encuentran en todo el conjunto

Estereotipo

- Si se tiene una lista de atributos u operaciones larga se podrá utilizar **estereotipos** para organizarla de forma que sea mas comprensible.
- Un estereotipo se muestra como un nombre bordeado por **dos pares de paréntesis angulares.**
- Para una lista de atributos podrá utilizar un estereotipo como encabezado de un subconjunto de atributos

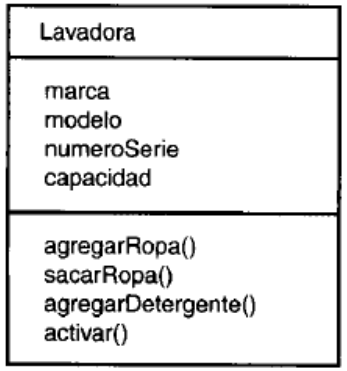
- Un **estereotipo** permite tomar elementos propios de UML y convertirlo en otros, es como un traje de mostrador y modificarlo para que se ajuste a sus medidas.

Un estereotipo se podrá usar para organizar una lista de atributos u operaciones

Diagrama de clases (responsabilidades y restricciones)

Responsabilidades y restricciones

- La responsabilidad es una descripción de lo que hará la clase, siendo la idea incluir información suficiente para describir una clase de forma inequívoca
- Una manera mas formal es agregar una restricción, un texto libre bordeado por llaves, donde este texto especifica una o varias reglas que sigue la clase.



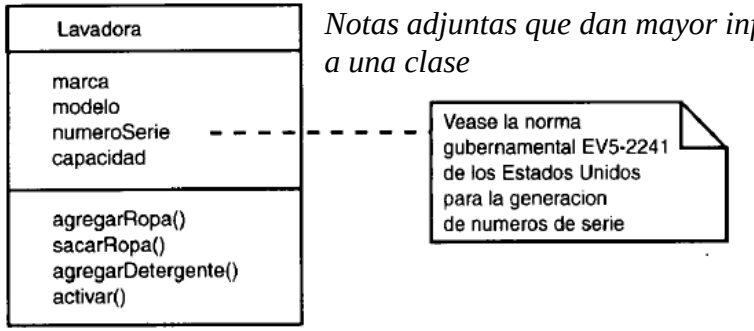
La regla entre llaves restringe al atributo capacidad para contener uno de tres posibles valores

{capacidad = 7, 8 o 9 Kg}

Responsabilidades y restricciones

Notas

- Por encima y debajo de los atributos, operaciones, responsabilidades y restricciones, puede agregar mayor información a una clase en la figura de **notas adjuntas**. Una nota puede contener tanto una imagen como texto.



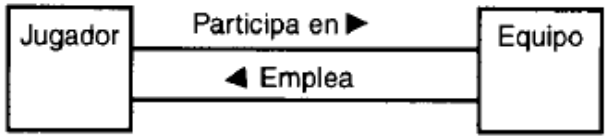
Notas adjuntas que dan mayor información a una clase

Notas

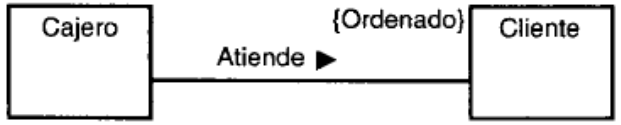
Diagrama de clases (asociaciones y restricciones)

Asociaciones entre clases y restricciones

- Se conoce como asociación cuando las clases se conectan entre si de forma conceptual.
- Cuando una clase se asocia con otra, cada una de ellas juega un papel dentro de tal asociación.
- La asociación puede funcionar en **dirección inversa**, y se podrá mostrar ambas asociaciones en el mismo diagrama con un triangulo relleno que indique la dirección de cada asociación.
- En ocasiones una asociación entre dos clases puede ser cierta regla, indicándose junto a la línea de asociación. Para indicar **la restricción** se hace entre llaves junto a la clase.



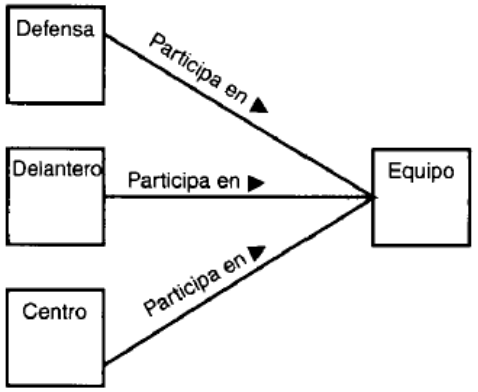
Pueden aparecer dos asociaciones entre clases en el mismo diagrama



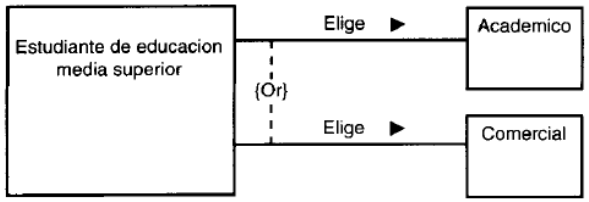
*Puede establecer una restricción en una asociación, que para este caso la asociación **atiende** esta restringida para que el cajero atienda al cliente en turno*

Asociaciones entre múltiples clases y relación O

- Las asociaciones podrían ser completas en las que una clase se puede conectar a varias.
- Otro tipo de restricciones es la **relación O,** distinguida como {Or} en una línea discontinua que conecte a dos líneas de asociación.



Pueden asociarse diversas clases con una en particular

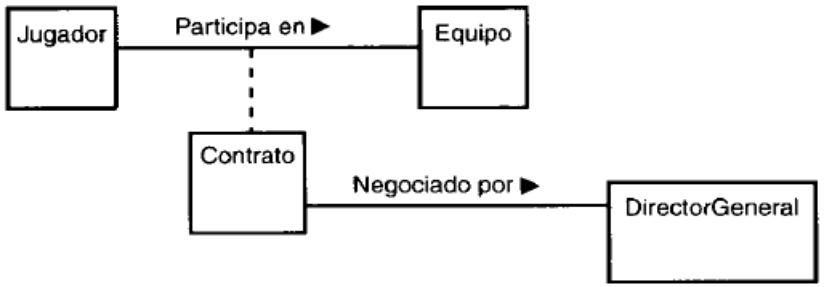


La relación O entre dos asociaciones en una restricción

Diagrama de clases (clases de asociaciones y vínculos)

Clases de asociaciones

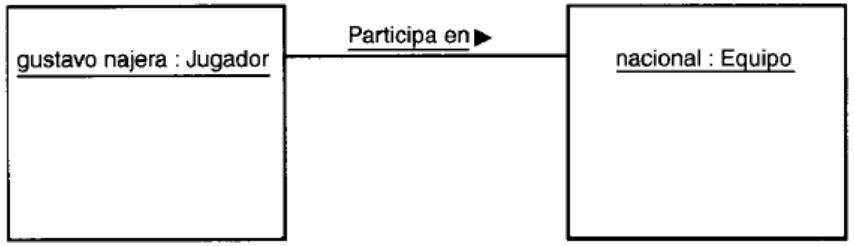
- Una asociación al igual que una clase puede contener atributos y operaciones llamándose **clase de asociación,** la cual tal como una clase estándar, y se utilizara una línea discontinua para conectar a la línea de asociación.



Una clase de asociación modela los atributos y operaciones de una asociación

Vínculos

- Así como un objeto es una instancia de una clase, una asociación también cuenta con instancias, llamándose a esta relación **vinculo** y se podrá representar como una línea que conecta a dos objetos subrayando el nombre del objeto y el nombre del vinculo.



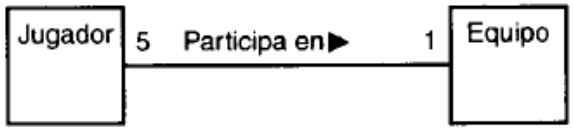
Un vinculo es la instancia de una asociación. Conecta a los objetos en lugar de las clases.

Diagrama de clases (multiplicidad)

Multiplicidad de una clase

- La multiplicidad señala la **cantidad de objetos** de una clase que se **relacionan** con un objeto de la clase asociada.
- Para representar los numero en el diagrama, se coloca sobre la línea de asociación junto a la clase correspondiente.

multiplicidad



La Multiplicidad señala la cantidad de objetos de una clase que pueden relacionarse con un objeto de una clase asociada

Multiplicidades entre clases

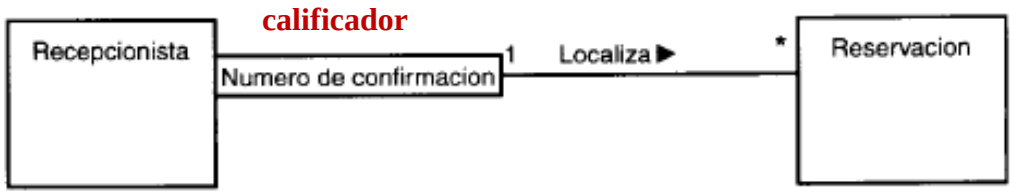
- Existen varios **tipos de multiplicidades** pudiéndose una clase relacionarse con otra en un esquema de uno a uno, uno a muchos, uno a uno o mas, uno o ninguno o uno, uno a un intervalo definido.
- El UML utilizar un asterisco (*) para representar mas y para representar muchos

Posibles multiplicidades y su representación en UML.

Diagrama de clases (asociaciones calificadas y reflexivas)

Asociaciones calificadas

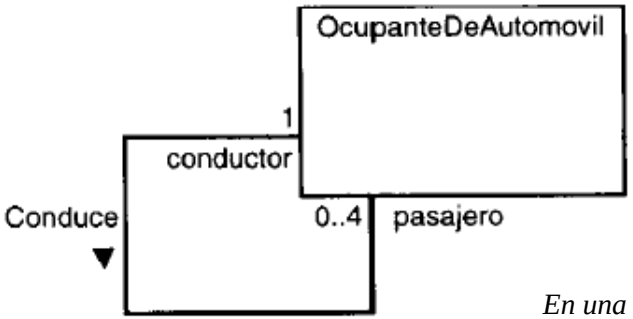
- Cuando la multiplicidad de una asociación es de uno a muchos, con frecuencia se presenta un reto en la **búsqueda**.
- Cuando un objeto de una clase tiene que seleccionar un objeto particular de otro tipo para cumplir con un papel en la asociación, la primera clase deberá atenerse a un atributo en particular para localizar el objeto adecuado. Dicho atributo es un i**dentificador** que puede ser un numero de identidad. En el UML la información de identidad **se conoce** como **calificador** y su símbolo es un pequeño rectángulo adjunto a la clase que hará la búsqueda.



Un calificador en una asociación resuelve el problema de la búsqueda.

Asociaciones reflexivas

- En ocasiones, una clase es una asociación consigo misma. Esto puede ocurrir cuando una clase tiene objetos que pueden jugar diversos papeles.
- Esto lo representara mediante el trazado de una línea de asociación a partir de un rectángulo de la clase hacia el mismo rectángulo de la clase, y en la línea de asociación indicar los papeles, nombre de la asociación, dirección de la asociación y multiplicidad.



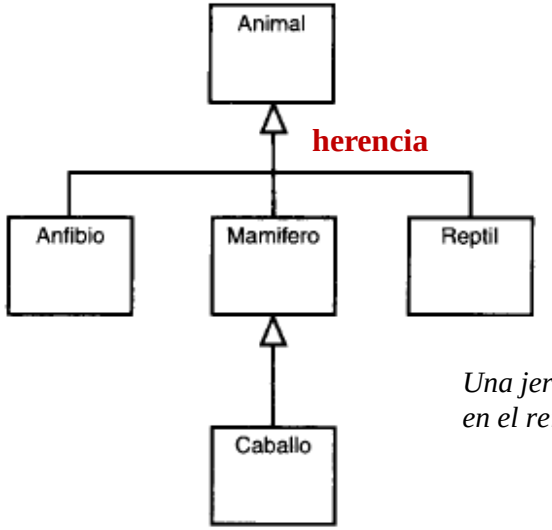
Asociacion reflexiva

En una asociación reflexiva, trazar la línea de la clase hacia si misma y podrá incluir los papeles, nombre de la asociación y su dirección, así como su multiplicidad

Diagrama de clases (herencia y generalización)

Herencia y generalización

- Una clase **puede heredar atributos y operaciones** de otra llamando a esto herencia desde el orientación a objeto y generalización desde UML. En el UML la herencia se representa con una línea que conecta a la clase principal con la secundario.
- En la **generalización**, una clase secundaria (hija) es sustituible por una clase principal (madre), es decir donde quiera que se haga referencia la clase madre, también se hace referencia a la clase hija.
- Con frecuencia las clases secundarias agregan otras operaciones y atributos a los que han heredado.
- Una clase puede no provenir de una clase principal, en cuyo caso será una clase base o clase raíz.
- Si una clase tiene exactamente una clase principal, tendrá una **herencia simple**, y si proviene de varias clases principales, tendrá un **herencia múltiple**.

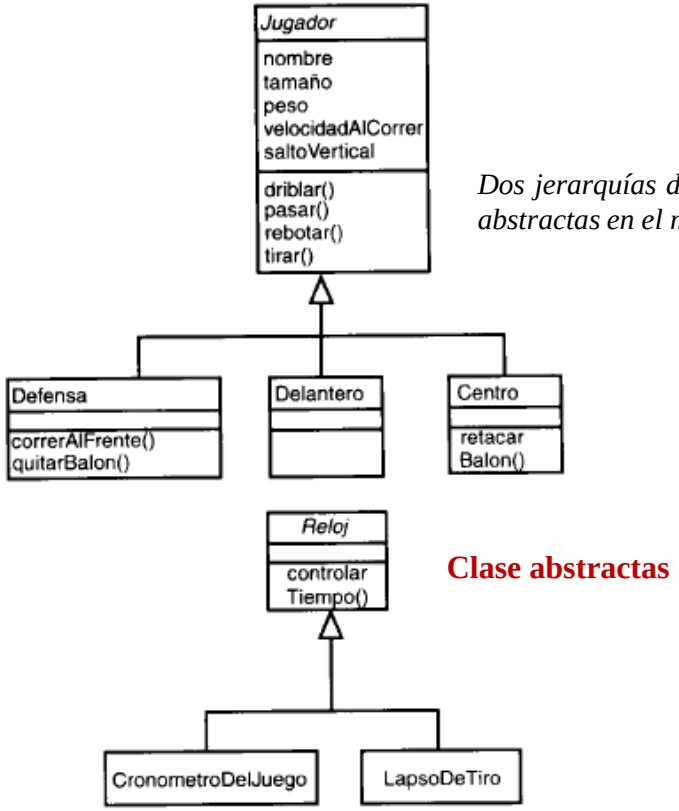


Una jerarquía de herencia en el reino animal

Diagrama de clases (clases abstractas y dependencias)

Clase abstractas

- Las clases abstractas solo se proyectan como bases de herencia y no proveen objetos por lo que no se pueden instanciar.
- Una clase abstracta se distingue por tener su *nombre en cursivas*.
- Se usan para definir subclases y son clases reutilizables.

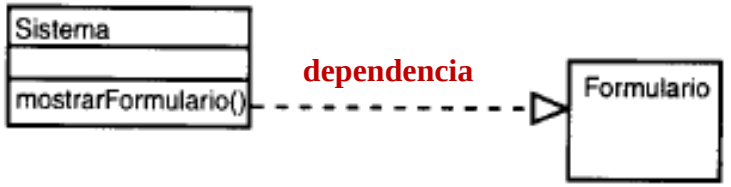


Dos jerarquías de herencia con clases abstractas en el modelo de baloncesto

Clase abstractas

Dependencias

- Una clase que utiliza a otra se le llama **dependencia** a este tipo de relación.
- La notación UML para ello es una línea discontinua con una punta de flecha en forma de triangulo sin relleno que apunta a la clase de la que depende.

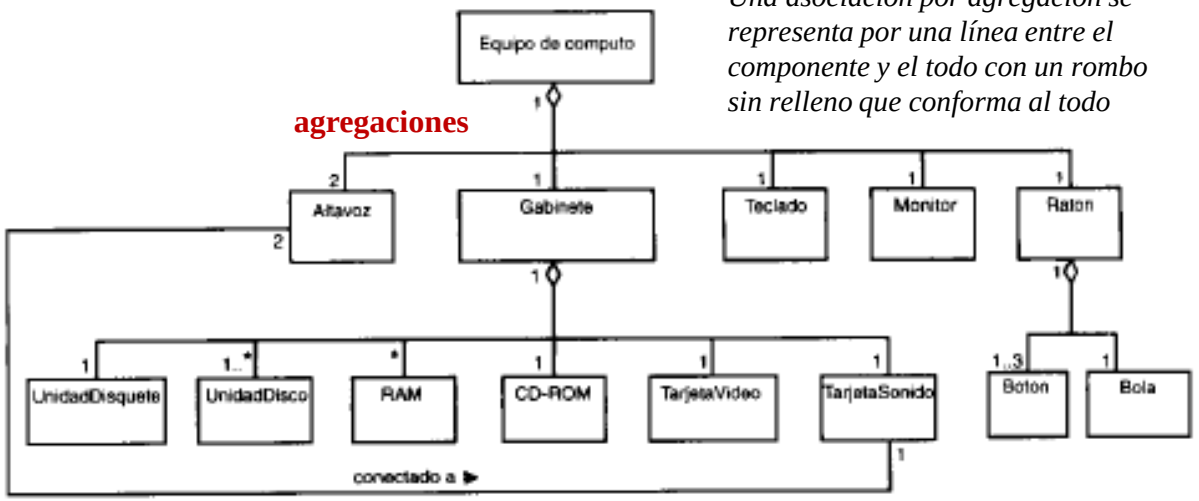


Una flecha representada por una línea discontinua con una punta de fecha en forma de triangulo sin relleno simboliza una dependencia

Diagrama de clases (agregaciones y composiciones)

Agregaciones

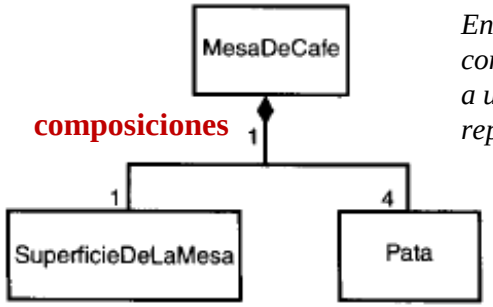
- En ocasiones una clase consta de otras clases, llamando a este tipo especial de relación agregación o acumulación.
- Se puede representar una **agregación** como una **jerarquía** dentro de la clase completa en la parte superior y los componentes por debajo de ella.
- Una línea conectara el todo con un componente mediante un **rombo sin relleno** que se colocara en la línea mas cerca al todo.



Una asociación por agregación se representa por una línea entre el componente y el todo con un rombo sin relleno que conforma al todo

Composiciones

- Una composición es un tipo representativo de una agregación. Cada componente dentro de una composición puede pertenecer tan solo a un todo.
- El símbolo de composición es el mismo que el de una agregación, excepto que el rombo esta relleno.

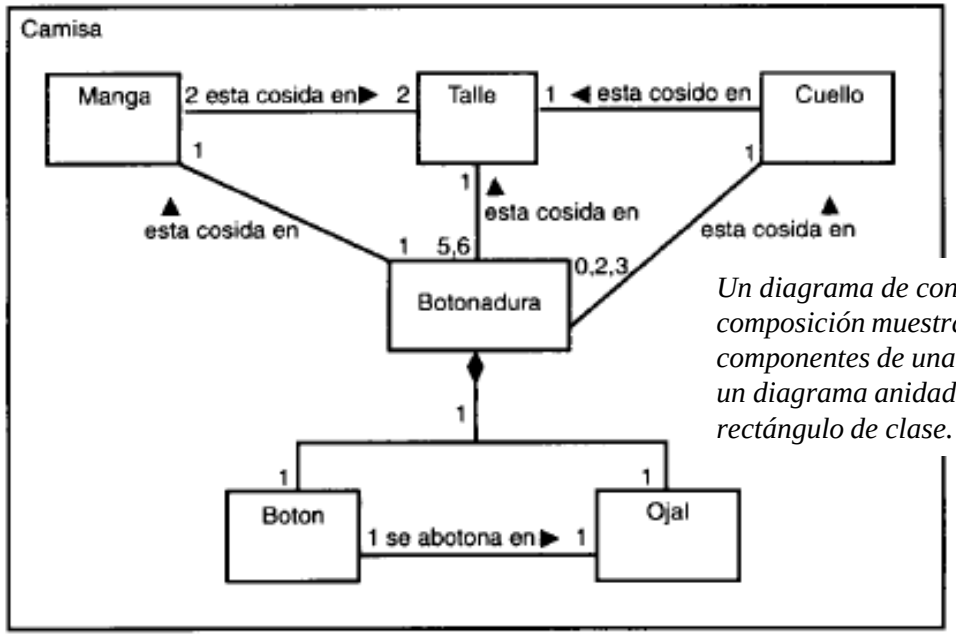


En una composición cada componente pertenece solamente a un todo. Un rombo relleno representa esta relación.

Diagrama de clases (contexto e interface)

Contextos

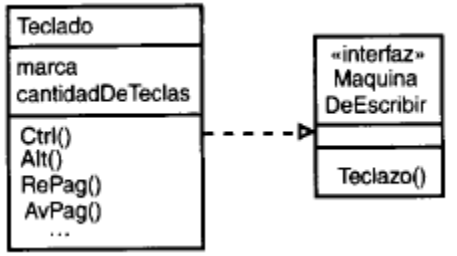
- Cuando se modela un sistema podría producirse con frecuencia **agrupamiento de clases** como **agregaciones o composiciones**. En tal caso se debe enfocar a agruparse por medio de un diagrama de contexto.
- Un diagrama de contexto es un mapa detallado de alguna sección de un mapa de mayores dimensiones.



Un diagrama de contexto de composición muestra los componentes de una clase como un diagrama anidado dentro de un rectángulo de clase.

Interface

- La interfaz es un conjunto de operaciones que especifica **cierto aspecto** de la funcionalidad **de una clase** y es un conjunto de operaciones que una clase presenta en otras.
- Se puede modelar la interfaz del mismo modo que una clase, con un **símbolo rectangular**. La diferencia será que como un conjunto de operaciones una interfaz no tiene atributos.
- Una forma de distinguir una interfaz y una clase que no muestra atributos es la estructura estereotipo y especificar la palabra <<**interfaz**>>
- Al nombre de una **interfaz** se le antecede el nombre de la interfaz con una **"I" mayúscula.**



Una interfaz es un conjunto de operaciones que realiza una clase. Esta ultima se relaciona con una interfaz mediante la realización, misma que se indica por una línea discontinua con una flecha en forma de triangulo sin rellenar que apunte a la interfaz

- Otra forma de representar una clase y su interfaz es con un pequeño circulo que se conecte mediante una línea a la clase

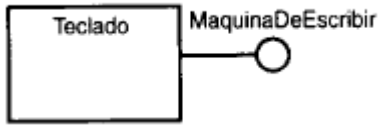


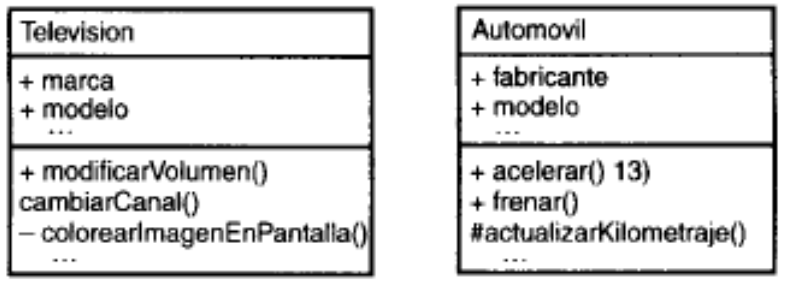
Diagrama de clases (visibilidad de atributos y métodos)

Visibilidad de los atributos y métodos de una clase

- El concepto de visibilidad esta relacionado con las interfaces y la realización. La visibilidad se aplica a atributos u operaciones, y establece la proporción en que otras clases podrán utilizar los atributos y operación de una clase dada (o en operaciones de una interfaz).
- Existen tres niveles de visibilidad:
  - Nivel publico, en el cual la funcionalidad se extiende a otras clases,
  - En el nivel protegido la funcionalidad se otorga solo a las clases que se heredan de la clase original.
  - En el nivel privado solo la clase original puede utilizar el atributo u operación.
- Para especificar la visibilidad de un miembro de la case (atributo o método), se coloca uno de los siguientes signos delante de ese miembro:

+	Público
-	Privado
#	Protegido
/	Derivado (se puede combinar con otro)
~	Paquete

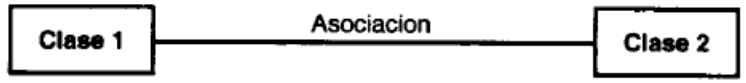
Niveles de visibilidad de los atributos y métodos de una clase

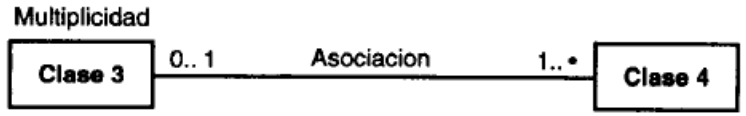


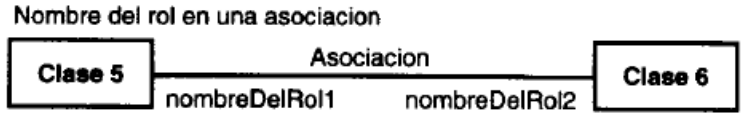
Los atributos y operaciones públicos y privados, tanto de una televisión como de un automóvil.

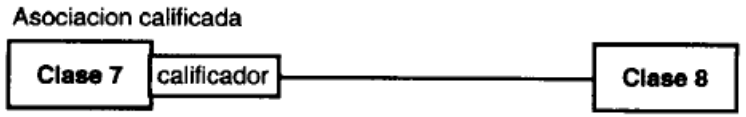
Diagrama de clase (ejemplos)

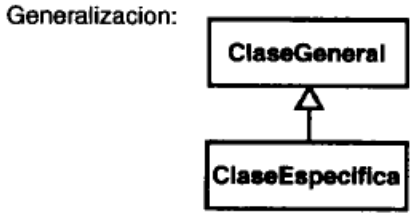
Diagrama de clases

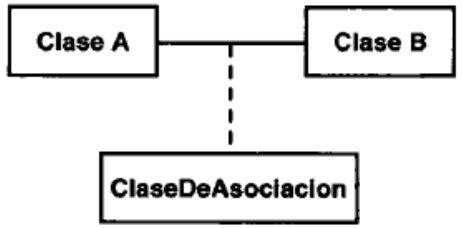


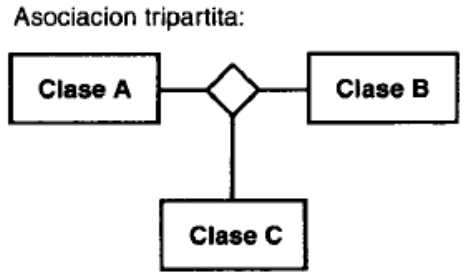


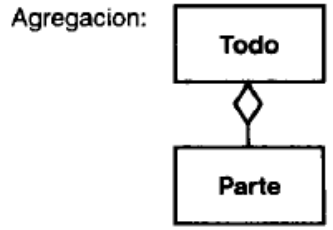


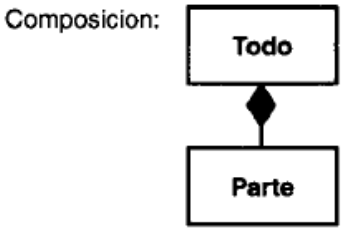












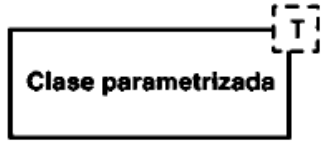
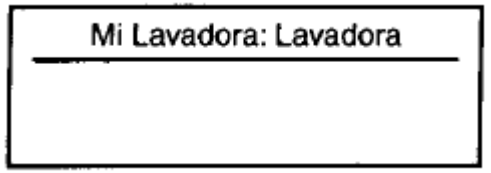


Diagrama de objeto

Diagrama de objeto y su representación

- Un objeto es una representación de algo en el mundo real, como un paciente, una receta, un medico, etc.
- Conforme se desarrolla una implementación, por lo general se necesitan definir los objetos de implementación adicional que se usan para dar la funcionalidad requerida del sistema.
- Un objeto es una instancia de clase (una entidad que tiene valores específicos de los atributos y acciones).
- Se representa en un rectángulo tal como una clase, pero el nombre esta subrayado. El nombre de la instancia especifica se encuentra a la izquierda de los dos puntos (:) y el nombre de la clase a la derecha.



El símbolo UML del objeto

Propiedades

- **Abstracción.** Se refiere a quitar las propiedades y acciones de un objeto para dejar solo aquellas que sean necesarios.
- **Herencia.** Un objeto es una instancia de una clase, por lo que la instancia tiene todas las características de la clase de la que proviene.
- **Polimorfismo**. Una operación puede tener el mismo nombre en diferentes clases.
- **Encapsulamiento**. Cuando un objeto trae consigo su funcionada oculta, por lo que los objetos encapsular lo que hacen, ocultando su funcionalidad interna de sus operaciones de otros objetos y del mundo exterior.
- **Envío de mensajes**. Un objeto envía a otro un mensaje para realizar una operación y el objeto receptor ejecutar la operación.
- **Asociaciones**. Los objetos se relacionan entre si de alguna forma, y en ocasiones los objetos pueden asociarse en mas de una forma.
- **Agregación**. Agregación o adición, es otro tipo de asociación entre objetos. Un objeto agregado consta de un conjunto de objetos que lo componente y una composición es un tipo especial de agregación. En un objeto compuesto, los componentes solo existen como parte del objeto compuesto.

Diagrama de casos de uso (características y representación)

El caso de uso

- Un caso de uso puedo tomarse como un **simple escenario** que describa lo que se espera el usuario de un sistema. Es una estructura que ayuda a los analistas a trabajar con los usuarios para determinar la forma en que se usara el sistema.
- Cada escenario describe una secuencia de eventos. Cada secuencia se inicia por una persona, otro sistema, una parte del hardware o por el paso del tiempo.
- A las entidades que inician secuencias se les conoce como **actores**. El termino actor se refiere a un rol especifico de un usuario del sistema.
- El resultado de la secuencia debe ser algo utilizable ya sea por el actor que la inicio o por otro actor.

Representación de un modelo de caso de uso

- Una **elipse** representa a un caso de uso, una figura agregada represente a un actor. El actor que inicia se encuentra a la izquierda del caso de uso y el que recibe a la derecha.
- Una **línea asociativa** conecta a un actor con el caso de uso y representa la comunicación entre el actor y el caso de uso.
- Un **rectángulo** con el nombre del sistema en algún lugar de el representa el confín del sistema. El rectángulo envuelve a los casos de uso del sistema.
- Los **actores**, casos de uso y líneas de interconexión componen un modelo de caso de uso.

En un modelo de caso de uso, una figura agregada representa a un actor, una elipse a un caso de uso y una línea asociativa representa la comunicación entre el actor y el caso de uso.

Cada caso de uso se lista en modo de texto a:

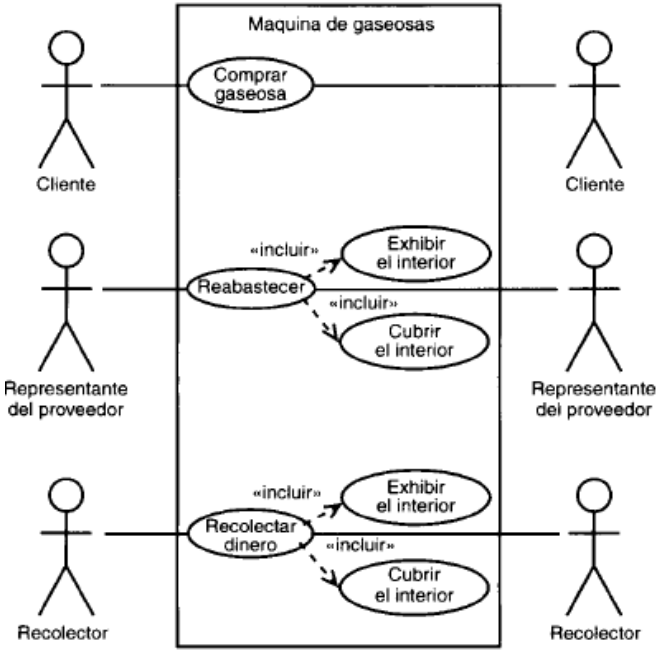
- ✓ El actor que inicia al caso de uso
- ✓ Condiciones previas para el caso de uso
- ✓ Pasos en el escenario
- ✓ Condiciones posteriores cuando se finaliza el escenario
- ✓ El actor que se beneficia del caso de uso

Diagrama de casos de uso (inclusión y extensión)

Relaciones entre casos de uso, los casos de uso se pueden relacionar entre si por dos formas, la inclusión y la extensión.

Inclusión

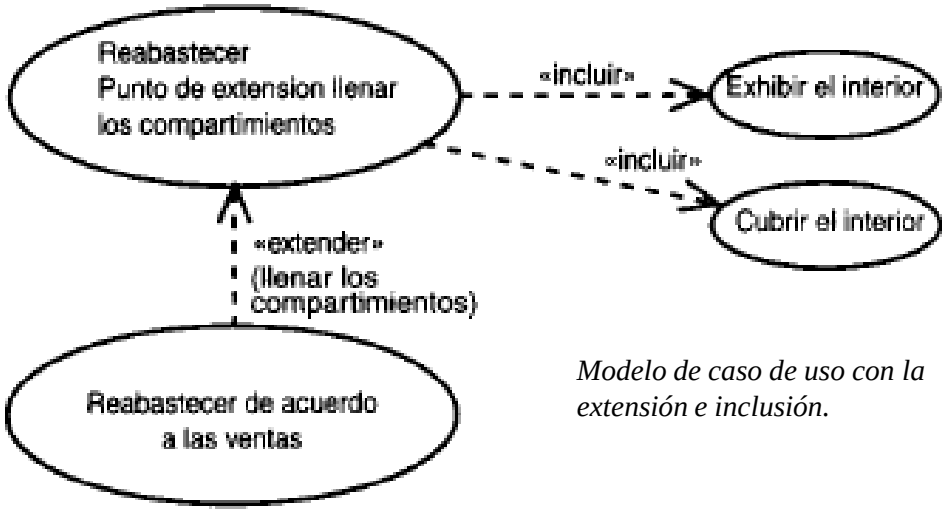
- La **inclusión** permite volver a utilizar los pasos de un caso dentro de otro, el cual se representa con una línea discontinua con una punta de flecha que conecta las clases apuntando hacia la clase dependiente. Justo sobre la línea se agrega un estereotipo con la palabra <<**incluir**>>



Modelo de caso de uso con la inclusión

Extensión

- La **extensión**, permite crear un caso de uso mediante la adición de pasos a uno existente, este se representa con una línea discontinua con punta de flecha, y junto con un estereotipo que muestra <<**extender**>>

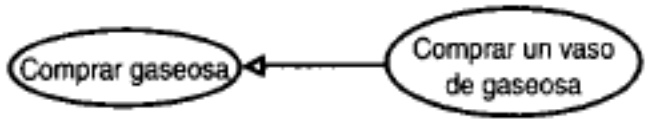


Modelo de caso de uso con la extensión e inclusión.

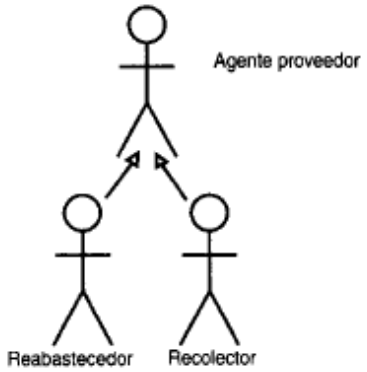
Diagrama de casos de uso (generalización y agrupamiento)

Generalización

- Los casos de uso al igual que las clases se pueden heredar entre si, heredando acciones y significado del primario, esto se representa con una punta de fecha en forma de triangulo sin rellenar que apunta hacia el caso de uso primario.
- La relación de **generalización** puede establecerse entre actores, así como entre casos de uso.



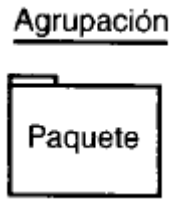
Un caso de uso puede heredar el sentido y comportamiento de otro



En los casos de uso, los actores pueden estar en una relación de generalización

Agrupamiento

- **Agrupamiento,** en ciertos diagramas de caso de uso, podría tener varios casos de uso que querrá organizar, y eso puede ocurrir cuando un sistema consta de varios subsistemas. La forma de organizar los casos de uso en por medio de paquete que se relacionen (carpeta tabular).

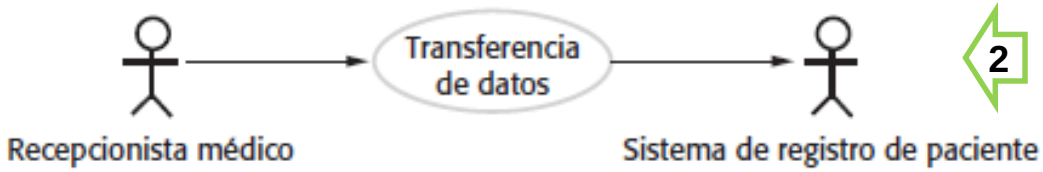


Forma de organizar los casos de uso por medio de paquetes relacionados

Diagrama de casos de uso (ejemplo)

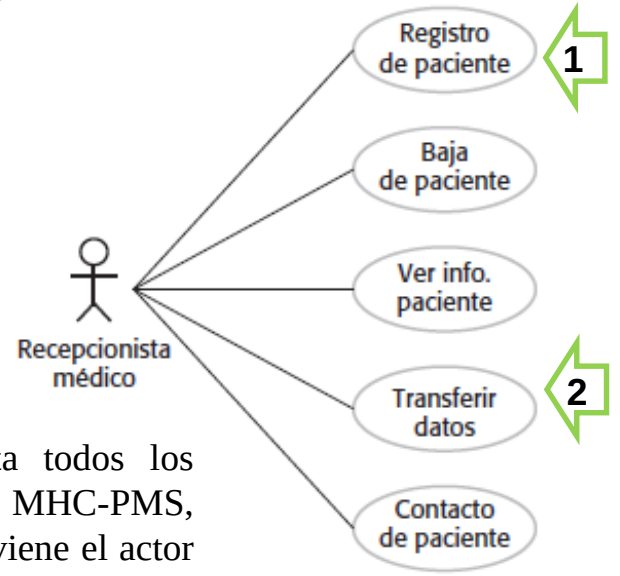
MHC-PMS: Transferencia de datos	
Actores	Recepcionista médico, sistema de registros de paciente (PRS).
Descripción	Un recepcionista puede transferir datos del MHC-PMS a una base de datos general de registro de pacientes, mantenida por una autoridad sanitaria. La información transferida puede ser información personal actualizada (dirección, número telefónico, etc.) o un resumen del diagnóstico y tratamiento del paciente.
Datos	Información personal del paciente, resumen de tratamiento.
Estímulo	Comando de usuario emitido por recepcionista médico.
Respuesta	Confirmación de que el PRS se actualizó.
Comentarios	El recepcionista debe tener permisos de seguridad adecuados para acceder a la información del paciente y al PRS.

Observe que en este caso de uso hay dos actores: el operador que transfiere los datos y el sistema de registro de pacientes. La notación con figura humana se desarrolló originalmente para cubrir la interacción entre individuos, pero también se usa ahora para representar otros sistemas externos y el hardware.



Número	RF-XXX		
Nombre			
Descripción			
Precondiciones			
Secuencia normal	Paso	Acción	
	1		
	2		
	3		
Postcondición			
Excepciones	Paso	Acción	
	1		
	2		
Prioridad	☐ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tabla de registro para el caso de uso



La figura presenta todos los casos de uso en el MHC-PMS, en los cuales interviene el actor "recepcionista médico".

Diagrama de caso de uso (resumen)

Resumen

1. Los **diagramas de caso** de uso se empiezan con **entrevista con los clientes** por lo que dan como resultado casos de uso de alto nivel que muestra los requerimientos funcionales del sistema.
2. El caso de prueba (un conjunto de condiciones o variables bajo las cuales un analista determinar si una aplicación, o un sistema software es parcial o completamente satisfactoria) permite obtener los **requerimientos funcionales**.
3. Los diagramas de caso de uso conciben los casos de uso facilitando la **comunicación entre** analistas y usuarios, analistas y clientes.
4. En un diagrama el símbolo del caso de uso es una **elipse**, y el símbolo de un actor es una figura adjunta.
5. Una línea **asociativa** conecta a un actor con el caso de uso.
6. Los casos de uso van por lo general **dentro de un rectángulo** que representa el confín del sistema.
7. La **inclusión** se representa por una línea de dependencia con un estereotipo <<incluir>> y la **extension** por una línea de dependencia con un estereotipo <<extender>>.
8. Las otras dos relaciones entre casos de uso son **generalización**, en la que un caso de uso hereda el sentido y acciones de otro, y el **agrupamiento**, mismo que organiza un conjunto de casos de uso. La generalización se representa por la misma línea que muestra la herencia entre clases y el agrupamiento se representa por el icono del paquete.

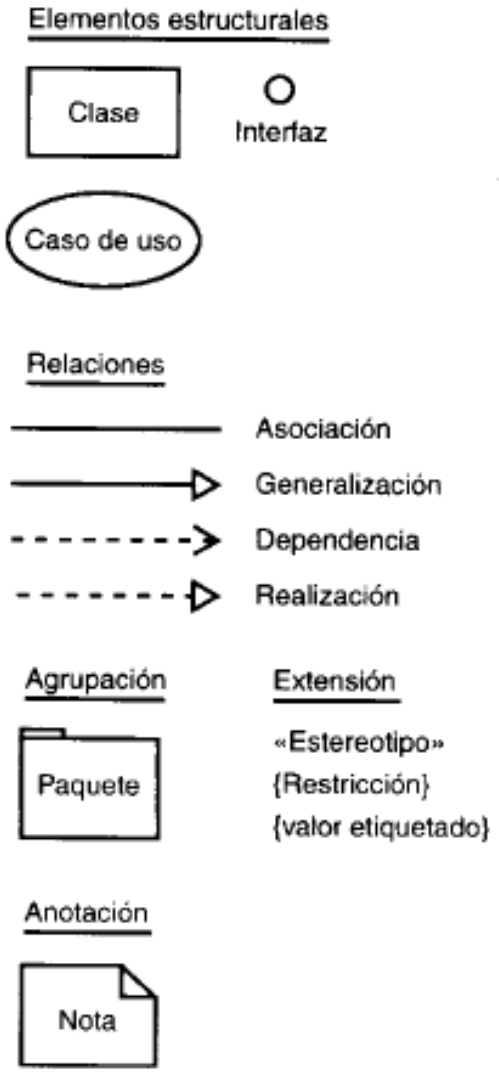


Diagrama de estados (características y representación)

Diagrama de estado

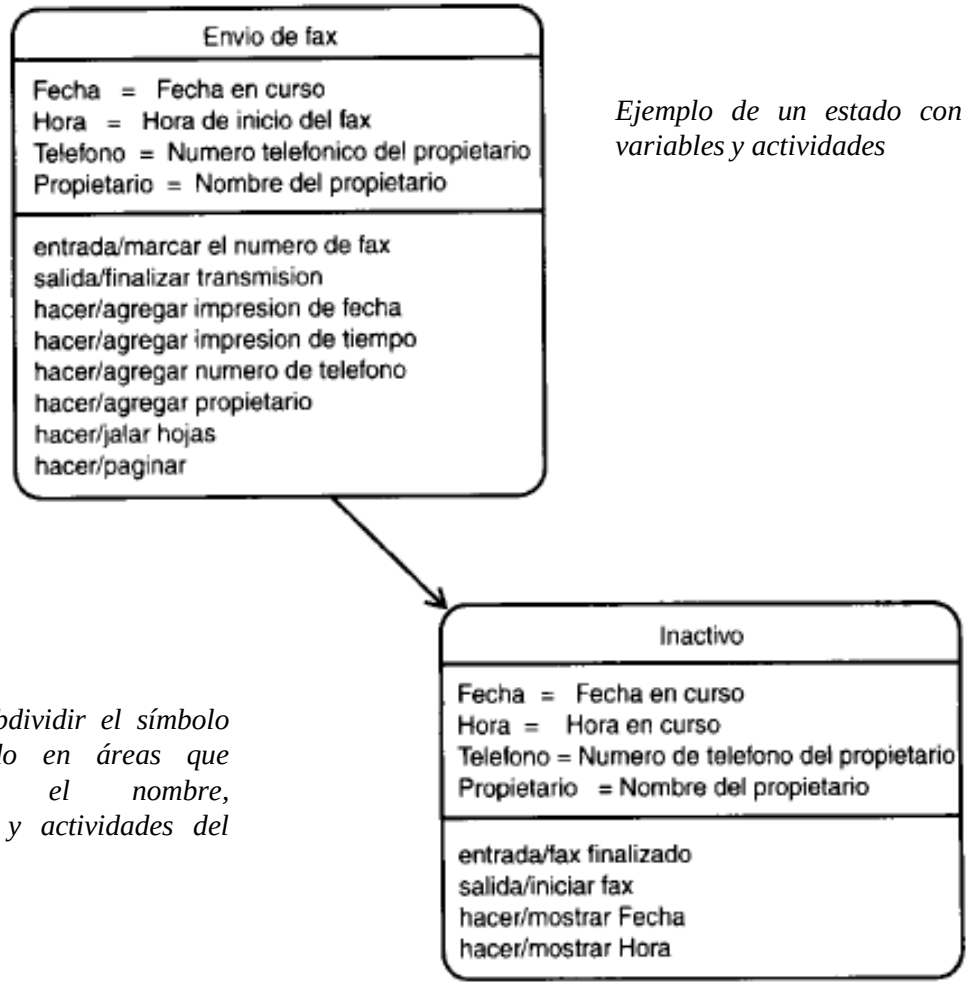
- Un diagrama de estado o motor de estado, caracteriza un cambio en un sistema, es decir cuando un objeto modifica su estado en respuesta a los sucesos y al tiempo.

Representación de un diagrama de estados

- Un **rectángulo de vértices redondeados** representa un estado, junto con una línea continua y una punta de flecha, mismas que representan a una transición.
- La punta de la flecha apunta hacia el estado donde se hará la transición. La figura también muestra un circulo relleno que simboliza un punto inicial y un punto final.
- Es posible dar detalles dividiendo un **símbolo de clase en tres áreas (nombre, atributos y operaciones).** El área superior contendrá el nombre del estado, el área central contendrá variables de estado, y el área inferior actividades.

El icono para el estado es un rectángulo de vértices redondeados, y el símbolo de una transición es una línea continua y una punta de fecha

Puede subdividir el símbolo del estado en áreas que muestren el nombre, variables y actividades del estado



Ejemplo de un estado con variables y actividades

Diagrama de estados (sucesos, acciones y condiciones de seguridad)

Sucesos y acciones

- Las **variables** de estado son como **cronómetros o contadores**, y las actividades sucesos y acciones (entre las mas utilizadas son las entradas, salida, y hacer).
- A las líneas de transición se le pueden agregar detalles que **provoquen un suceso**, la actividad de computo que se ejecute y haga que suceda la modificación del estado. Una GUI puede establecerse en uno de tres estados, *inicialización, operación, apagar* por ejemplo.

Los estados y transiciones de una interfaz grafica del usuario incluyen el desencadenamiento de eventos, acciones y transiciones no desencadenadas.

Condiciones de seguridad

Si se ha pasado cierto tiempo sin que haya interacción con el usuario, una **GUI** hará una transición del estado operación a un estado como un protector de pantalla y una condición de seguridad añadida. La condición de seguridad se establece como expresión booleana.

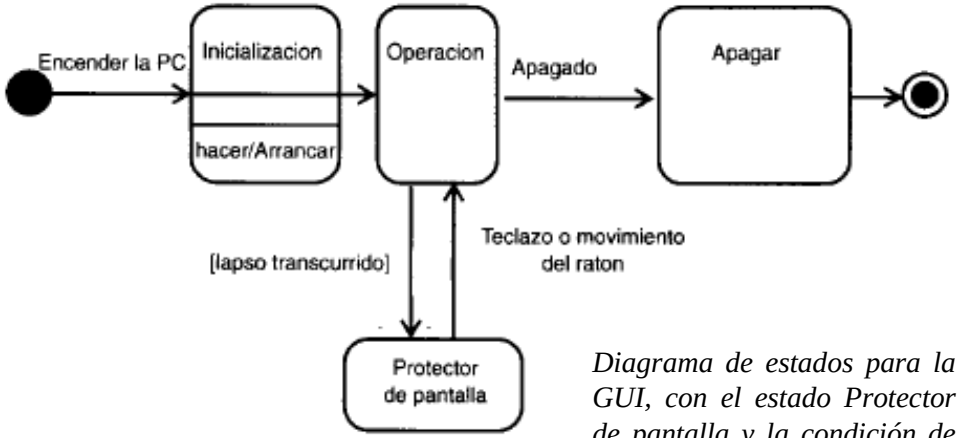
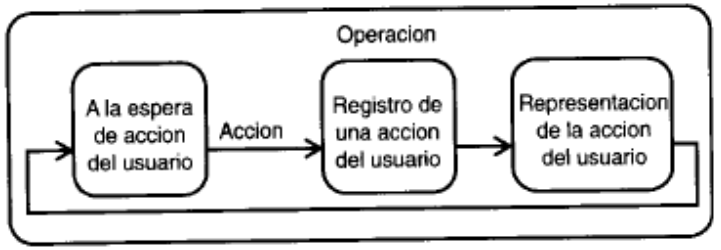


Diagrama de estados para la GUI, con el estado Protector de pantalla y la condición de seguridad.

Diagrama de estados (subestados, mensajes y señales)

Subestados

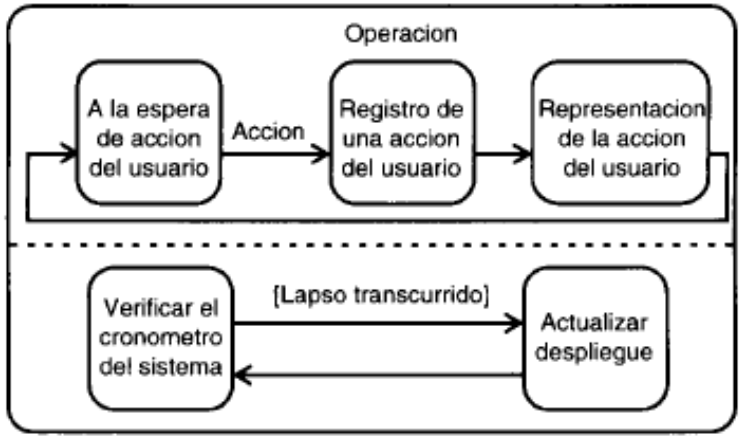
Cuando una GUI esta en el estado operación, muchas cosas ocurren aunque no evidentes en una pantalla, tales acciones serán cambios de estado, por lo que una GUI antes de encontrase al estado Operación a travesara por cambios llamados subestados.



Subestados secuenciales dentro del estado operación de la GUI.

Tipos de Subestados

1. **Subestados secuenciales**, suceden uno detrás de otro. Por ejemplo: En una interface GUI, las acciones del usuario desencadena la secuencia de transición a partir de "A la espera de acción del usuario", transcendiendo al segundo estado "Registro de una acción del usuario", y después el tercer estado "representación de la acción del usuario".
2. **Subestados concurrentes**, dentro del estado Operación, la GUI no solo aguarda a que usted haga algo, también verifica el cronometro del sistema y posiblemente actualiza el despliegue de una aplicación luego de un intervalo especifico. Un subestado concurrente se puede representar con una línea discontinua entre los estados concurrentes.



Los subestados concurrentes suceden al mismo tiempo. Una línea discontinua los separa.

Mensajes y señales

- A través de los mensajes los objetos se comunican mediante el envió de mensajes entre si.
- Un mensaje que desencadena una transición en el diagrama de estado del objeto receptor se conoce como señal.

Diagrama de secuencias (características y representación)

Diagramas de secuencia

- **Los diagramas de secuencia se usan para modelar las interacciones entre los actores y los objetos en un sistema,** así como las interacciones entre los objetos en si**.**

Representación de un diagrama de secuencia

- **Un diagrama de secuencias** consta de ***objetos*** que representan la interactividad con otros objetos.
- Se representan con un rectángulo con nombre (subrayado), mensajes representados por líneas continuas con una punta de flecha y el tiempo representado como una progresión vertical.
- **Los objetos** se colocan en la parte superior del diagrama de izquierda a derecha y se acomodan de manera que simplifiquen el diagrama.
- La extensión que esta debajo de cada objeto será **una *línea discontinua*** conocido como la línea de vida de un objeto. Junto con la línea de vida de un objeto se encuentra un pequeño rectángulo conocido como activación, el cual representa la ejecución de una operación que realiza el objeto. La longitud del rectángulo se interpreta como la duración de la actividad.

Representación de un objeto en un diagrama de secuencia

Diagrama de secuencias (mensaje y tiempo)

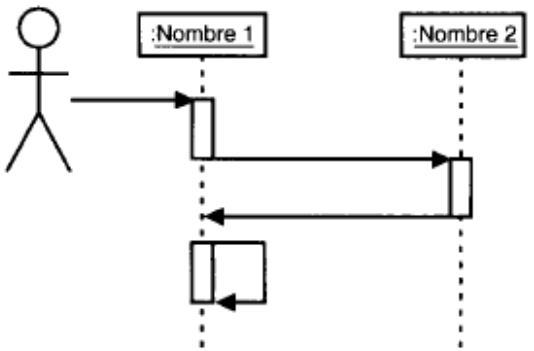
Mensaje

- **Un mensaje** que va de un objeto a otro pasa de la línea de vida de un objeto a la de otro. Un objeto puede enviarse un mensaje a si mismo. Un mensaje puede ser **simple, sincrónico o asincrónico**.
  - Un mensaje simple es la transferencia del control de un objeto a otro.
  - Si un objeto envía un mensaje sincrónico, esperar la respuesta a tal mensaje antes de continuar con su trabajo.
- En el diagrama de secuencia, los símbolos varían, por ejemplo la punta de la fecha de un mensaje simple esta formada por dos líneas, la punta de la fecha de un mensaje sincrónica esta rellena y la de un asincrónica tiene una solo línea.

Símbolos para los mensajes en un diagrama de secuencia.

Tiempo

- El diagrama representa al tiempo en dirección vertical. El tiempo se inicia en la parte superior y avanza hacia la parte inferior.
- Un mensaje que este mas cerca de la parte superior ocurrirá antes que uno que este cerca de la parte inferior.
- Los diagramas de secuencia tiene dos dimensiones. La dimensión horizontal es la disposición de los objetos y la dimensión vertical muestra el paso del tiempo.



- *Un diagrama de secuencia los **objetos** se colocan de izquierda a derecha en la parte superior.*
- *Cada **línea de vida** de un objeto es una línea discontinua que se desplaza hacia abajo del objeto.*
- *Una **línea continua** con una punta de flecha conecta a una línea de vida con otra, y representa un mensaje de un objeto a otro.*
- *El **tiempo** se inicia en la parte superior y continua hacia abajo.*
- *Aunque un **actor** es el que normalmente inicia la secuencia, su símbolo no es parte del conjunto de símbolos del diagrama de secuencia.*

Diagrama de secuencias (GUI y estados)

La GUI

- Recordándose que un diagrama de estado muestra los cambios por los que pasa una GUI, un diagrama de secuencias representa las interactividades de la GUI con otros objetos.
- Por ejemplo, supongamos que el caso de usuario de una GUI, la secuencia seria:

1. La GUI notifica al sistema operativo que se oprimió una tecla.
2. El sistema operativo le notifica a la CPU.
3. El sistema operativa actualiza la GUI
4. La CPU notifica a la tarjeta de video.
5. La tarjeta de video envía un mensaje al monitor.
6. El monitor presenta el carácter alfanumérico en la pantalla, con lo que se hará evidente al usuario.

*Nota: En ocasiones es muy estricto mostrar los **estados** de uno o varios de los objetos en el diagrama de secuencia.*

*Obsérvese que la secuencia se **origina y finaliza** en el estado operativo de la GUI*

Diagrama de estado. *Los estados y transiciones de una interfaz grafica del usuario incluyen el desencadenamiento de eventos, acciones y transiciones no desencadenadas.*

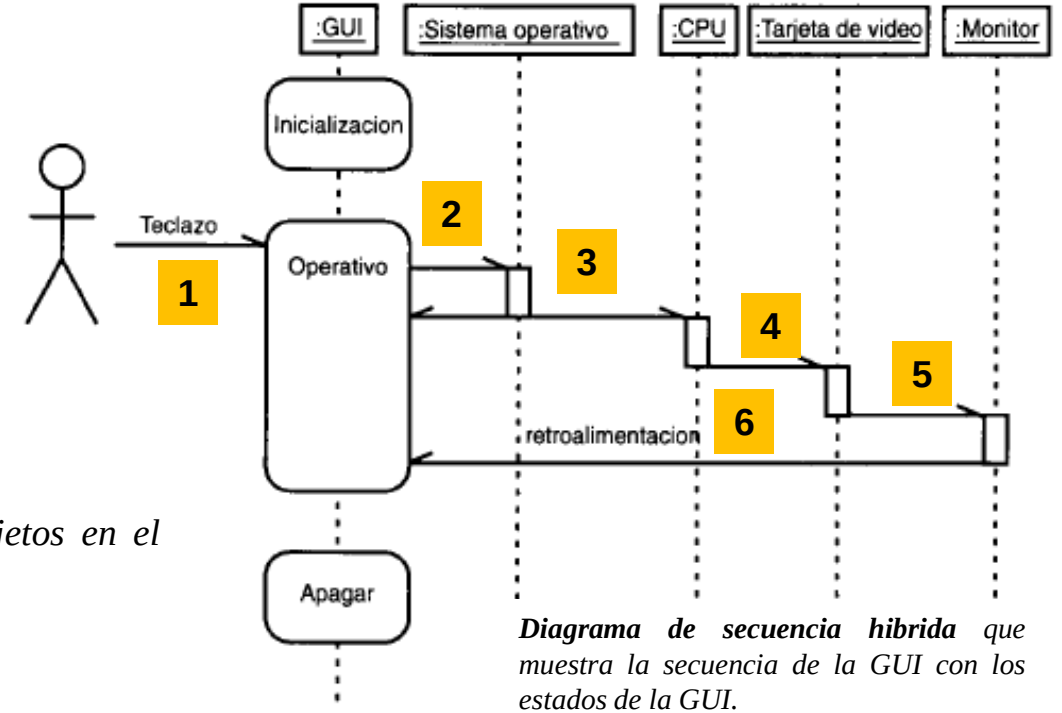
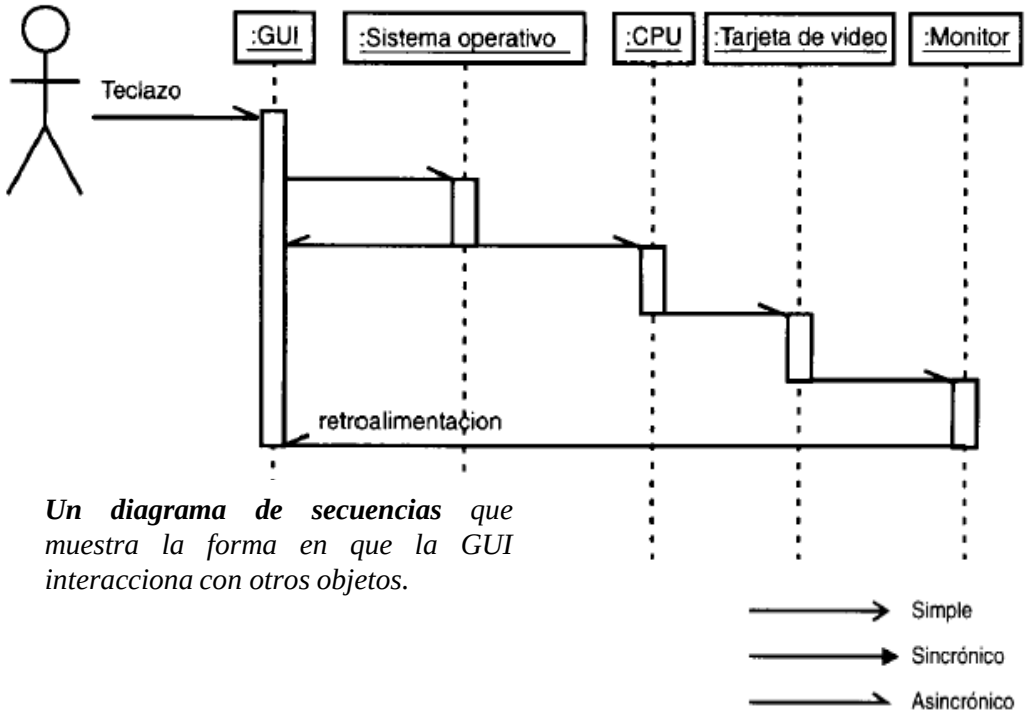


Diagrama de secuencia hibrida *que muestra la secuencia de la GUI con los estados de la GUI.*

Diagrama de secuencia (GUI y el caso de uso)

El diagrama de secuencia en una GUI

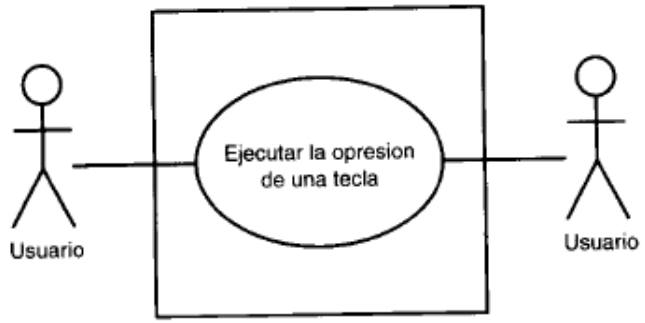
- En la figura que se muestra aquí, el diagrama de secuencia de la GUI, como se ve, los mensajes son asincrónicos (ninguno de los componentes guarda algo antes de continuar.
- Cuando se teclea en un procesador de texto, en ocasiones no ve aparecer en la pantalla el carácter correspondiente a la tecla que haya oprimido sino hasta después de haber oprimido algunas mas
- El ejemplo que se indica aquí muestra las interacciones de objetos que se realizan durante un escenario sencillo: “Ejecutar la presión de una tecla”.



Un diagrama de secuencias *que muestra la forma en que la GUI interacciona con otros objetos.*

El caso de uso

- El diagrama de secuencia muestra las interacciones de objetos que se realizan durante un escenario sencillo: la opresión de una tecla, el cual representa gráficamente las interacciones del sistema en el caso de uso, delineando el caso de uso dentro del sistema



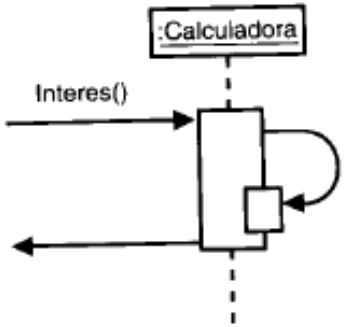
El caso de uso *representado gráficamente por un diagrama de secuencia*

Diagrama de secuencia (recursividad)

Recursividad

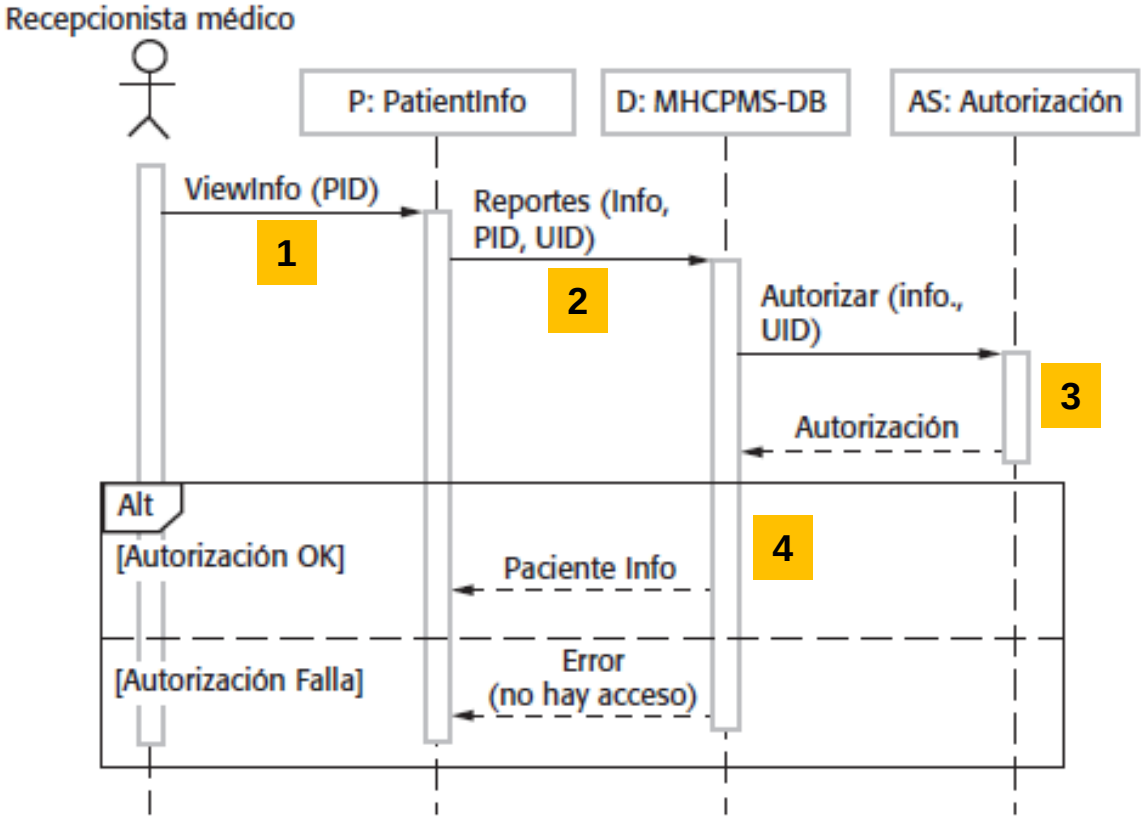
- En ocasiones un objeto cuenta con una operación que se invoca a si misma (recursividad), y es una características fundamental de varios lenguajes de programación.
- Para representar esto se dibuja una flecha de mensaje fuera de la activación que signifique la operación, y un pequeño rectángulo sobrepuesto en la actividad.

Suponiendo que en un objeto calculadora uno de sus operaciones sea el calculo de intereses, para calcular el interés compuesto para un periodo que incluya varios periodos, la operación del calculo de intereses del objeto tendrá que invocarse a si misma varias veces.



*Como representar la **recursividad** en un diagrama de secuencia.*

Diagrama de secuencia (ejemplo de recepcionista al ver información)



1. El recepcionista médico activa el método ViewInfo (ver información) en una instancia P de la clase de objeto PatientInfo, y suministra el identificador del paciente, PID. P es un objeto de interfaz de usuario, que se despliega como un formato que muestra la información del paciente.
2. La instancia P llama a la base de datos para regresar la información requerida, y suministra el identificador del recepcionista para permitir la verificación de seguridad (en esta etapa no se preocupe de dónde proviene este UID).
3. La base de datos comprueba, mediante un sistema de autorización, que el usuario esté autorizado para tal acción.
4. Si está autorizado, se regresa la información del paciente y se llena un formato en la pantalla del usuario. Si la autorización falla, entonces se regresa un mensaje de error.

Diagrama de colaboraciones (características y representación)

Diagrama de colaboraciones

- Los diagramas de colaboraciones muestran la forma en que los objetos colaborar entre si, tal como sucede con un diagrama de secuencias.
- Los diagramas de secuencias destacan la sucesión de las interacciones, y los diagramas de colaboraciones destacan el contexto y organización general de los objetos que interactúan. El diagrama de secuencia se organiza de acuerdo al tiempo, y el de colaboración de acuerdo al espacio.
- Estos diagramas también son conocidos como diagramas de comunicación.

Representación de un diagrama de colaboraciones

- Para representar un mensaje, se dibuja una **flecha cerca de la línea de asociación** entre dos objetos, esta fecha apunta al objeto receptor. El tipo de mensaje se mostrara en una etiqueta cerca de la flecha; por lo general, el mensaje le indicara al objeto receptor porque ejecute una de sus operaciones. El mensaje **finalizara con un par de paréntesis** dentro de los cuales colocara los parámetros con los que funcionara la operación.
- Se podrá convertir cualquier diagrama de secuencia en diagrama de colaboraciones y viceversa. Por medio de esto podrá representar la información en secuencia en un diagrama de colaboraciones. Para ello, agregara una cifra a la etiqueta de un mensaje, misma que corresponderá a la secuencia propia del mensaje. **La cifra y el mensaje se separaran mediante dos puntos (:).**

Simbología del diagrama de colaboraciones

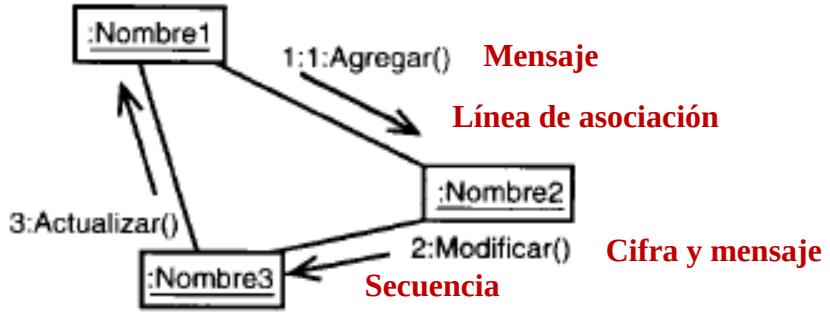


Diagrama de colaboraciones (la GUI, cambios de estado)

La GUI

En el ejemplo para una GUI en un diagrama de colaboraciones, un actor inicia la secuencia de interacción al oprimir la tecla, con lo que los mensajes ocurrirán de manera secuencial. Tal secuencia seria:

1. La GUI notifica al sistema operativo que se oprimo una tecla.
2. El sistema operativo le notifica a la CPU.
3. El sistema operativo actualiza la GUI.
4. La CPU notifica a la tarjeta de video.
5. La tarjeta de video envía un mensaje al monitor.
6. El monito presenta el carácter alfanumérico en la pantalla, con lo que se hará evidente al usuario.

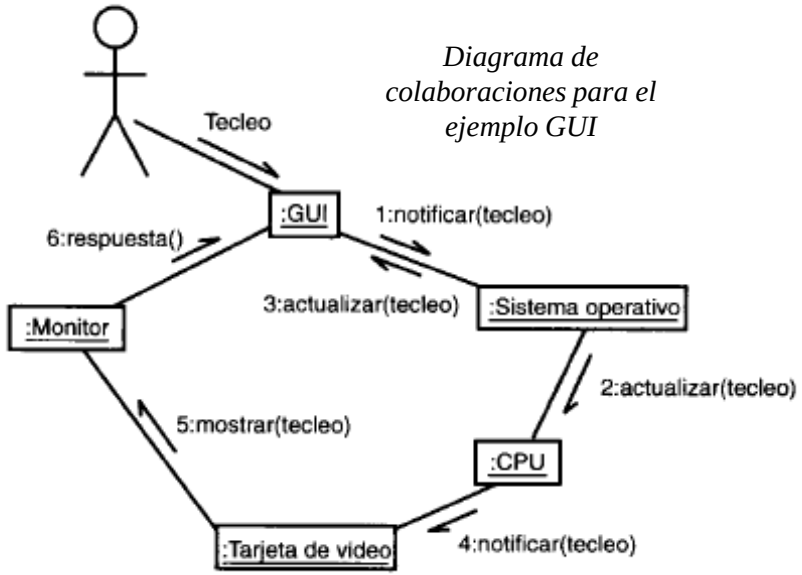
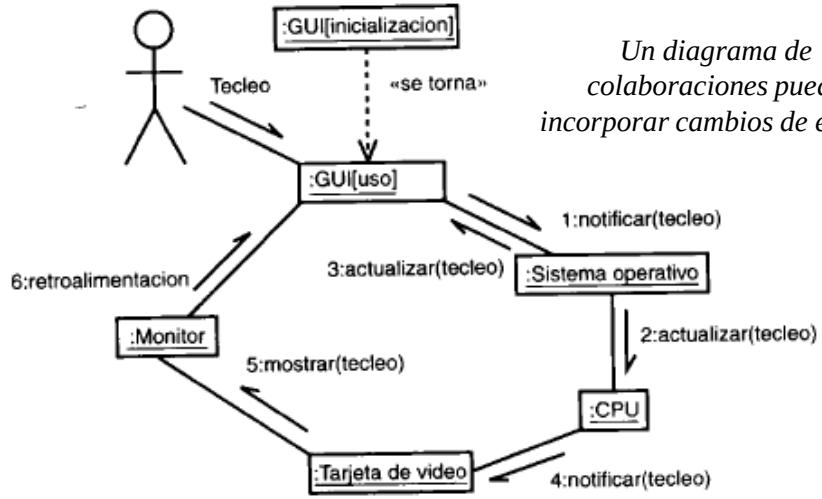


Diagrama de colaboraciones para el ejemplo GUI

Diagrama de colaboraciones y cambios de estado

- Los cambios de estado en un diagrama de colaboraciones se indican dentro del rectángulo.
- Se agrega otro rectángulo al diagrama que haga las veces del objeto y se indica el estado modificado.
- Se conecta a los dos con una línea discontinua y se etiqueta la línea con un estereotipo <<se torna>>

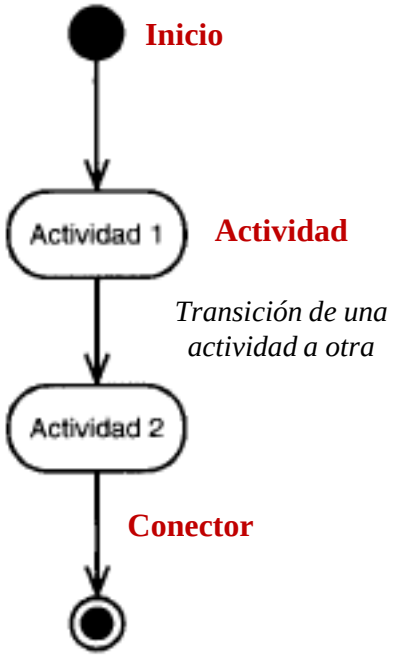


Un diagrama de colaboraciones puede incorporar cambios de estado

Diagrama de actividades (características y representación)

Diagrama de actividades

- Un diagrama de actividades ha sido diseñado para mostrar una visión simplificada de lo que ocurre durante una operación o proceso.
- Un diagrama de actividad se puede utilizar para modelar un caso de uso, una clase o un método mas complejo.



Representación de un Diagrama de actividades

- A cada actividad se le representa por un **rectángulo** con las esquinas redondeadas. El procesamiento dentro de una actividad se lleva a cabo y al realizarse, se continua con la siguiente actividad.
- Una **flecha** representa la transición de una u otra actividad.
- El diagrama de actividad cuenta con un **punto inicial** (representado por un circulo relleno) y **uno final** (representado por una diana).

Diagrama de actividades (decisiones, rutas concurrentes)

Decisiones

- Una secuencia de actividades llegara a un punto donde se realizara alguna **decisión**. Ciertas condiciones le llevaran por un camino y otras por otro (pero ambas son mutuamente exclusivas). Se podrá representar de **dos formas**:
  1. La primera es mostrar las rutas posibles que parten directamente de una actividad.
  2. La segunda es llevar la transición hacia un rombo.
- De cualquier forma que se represente la decisión se debe indicar con una instrucción entre **corchetes** junto a la ruta correspondiente.

Rutas concurrentes

- Conforme se modelan actividades se tendrá la oportunidad de separar una transición en dos rutas que se ejecuten al mismo tiempo (concurrente) y luego se reúnan. Para representar esta división, se utilizara una línea gruesa perpendicular a la transición y las rutas partirán de ella.

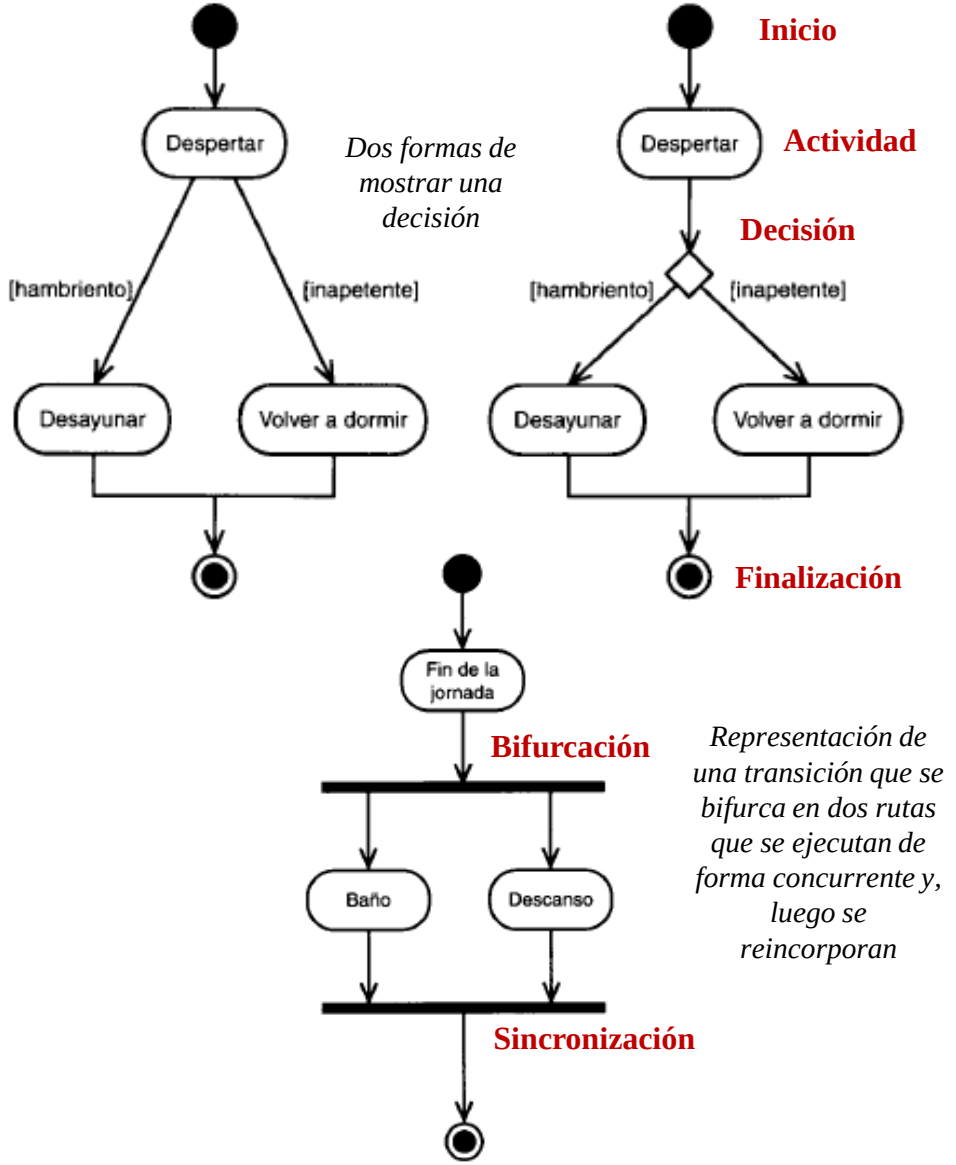


Diagrama de actividades (indicaciones)

Indicaciones

- Durante una secuencia de actividades, es posible enviar una indicación. Cuando se reciba, la indicación provocara que se ejecute una actividad. El símbolo para enviar una indicación es un pentágono convexo, y el que la recibe es un pentágono cóncavo.
- En termino de UML, el **pentágono convexo** simboliza al envío de un **evento** y el **cóncavo** simboliza la recepción del **evento**.

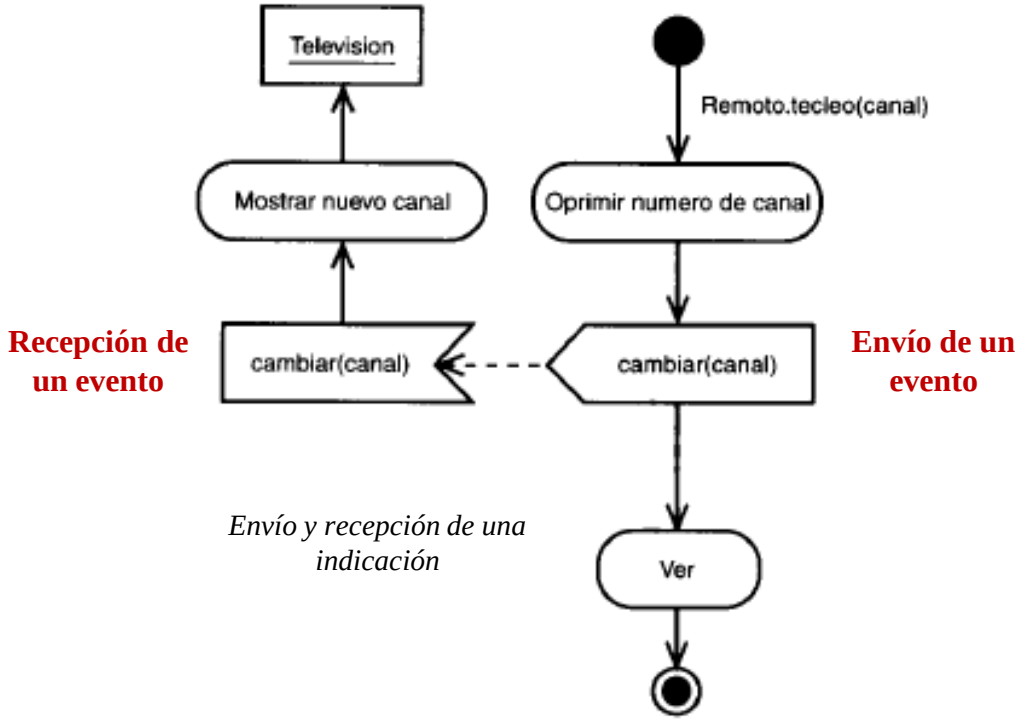


Diagrama de actividades (proceso de creación de un documento)

Proceso de creación de un documento

Posible secuencia de actividades para utilizar una aplicación de oficina para crear un documento:

1. Abrir la aplicación para procesamiento de textos.
2. Crear un archivo
3. Guardar el archivo con un nombre único en una carpeta
4. Teclear el documento
5. Si se necesitan ilustraciones, se abre la aplicación relacionada, se generan los gráficos y se colocan en el documento.
6. Si se necesita una hoja de calculo, se abre la aplicación correspondiente y se coloca en el documento.
7. Se guarda el archivo
8. Se imprime el documento
9. Se sale de la aplicación de oficina

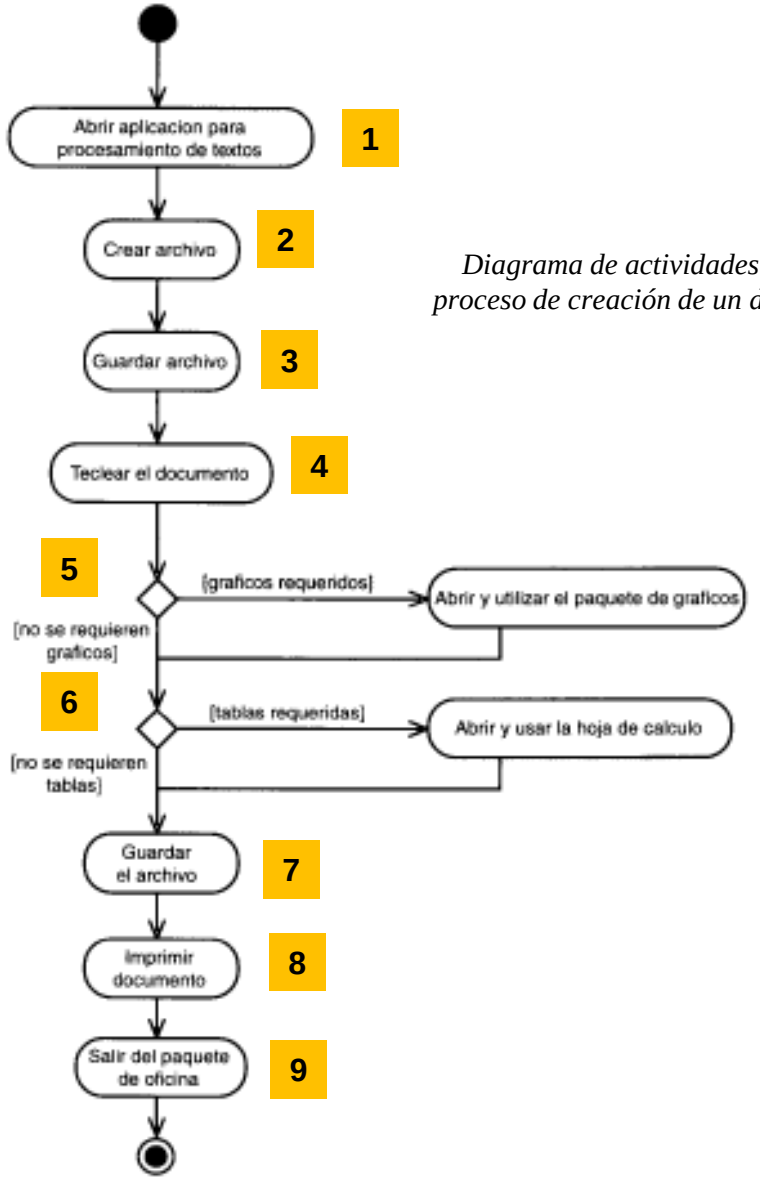


Diagrama de actividades para el proceso de creación de un documento

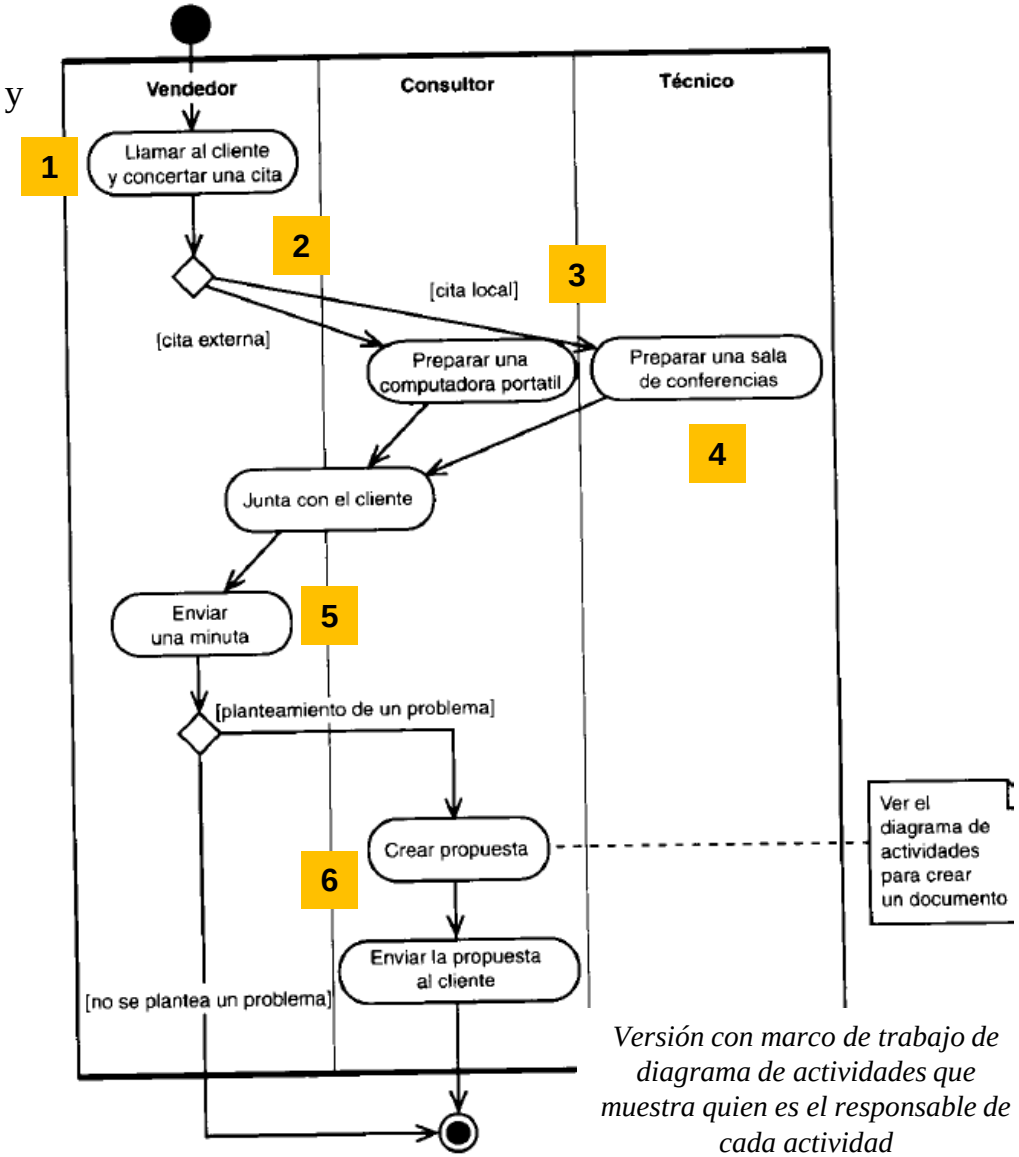
Diagrama de actividades (ejemplo y marcos de responsabilidad)

Marcos de responsabilidad

Uno de los aspectos mas útiles del diagrama de actividades es su facultad para expandirse y mostrar quien tiene las responsabilidad en un proceso.

Veamos las actividades que podrían ser para el caso de una firma de consultoría y el proceso de negociación involucrado en una junto con un cliente:

1. Un vendedor hace una llamada al cliente y concierta una cita
2. Si la cita es en la oficina del consultor, los técnicos corporativos prepararan una sala de conferencias para hacer una presentación.
3. Si la cita en la oficina del cliente, un consultor preparara una presentación en una laptop.
4. El consultor y el vendedor se reunirán con el cliente en el sitio y la hora convenida.
5. El vendedor crea una minuta.
6. Si la reunión ha planteado la solución de un problema, el consultor creara una propuesta y la enviara al cliente.



Versión con marco de trabajo de diagrama de actividades que muestra quien es el responsable de cada actividad

Componentes (características y tipos)

Componentes

- **Un componente de software** es una parte física de un sistema, y se encuentra en la computadora, no en la mente del analista, por ejemplo una tabla, archivo de datos, ejecutable**,** biblioteca de vínculos dinámicos, documentos…
- Un diagrama de componente se utiliza después de haber creado el diagrama de clase y puede ser la implementación de mas de una clase.
- Este diagrama proporciona una vista de alto nivel de los componentes dentro de un sistema.
- Los componentes pueden ser un componente de software, como una base de datos o una interfaz de usuario ; un componente de hardware como un circuito, un microchip o un dispositivo; o una unidad de negocio como un proveedor, nomina o un envio.
- Se utilizan en el Desarrollo basado en components para describer Sistema con arquitectura orientada a servicios.

Tipos de componentes

Como modelador se encontrara con tres tipo de componentes:

- **Componentes de distribución**, que conforman el fundamente de los sistemas ejecutables, por ejemplo DLL, ejecutables, controles Active X y Java Bean, archivo .HLP (ayuda)
- **Componentes para trabajar en el producto,** a partir de los cuales se han creado los componentes de distribución como archivos de base de datos y de código.
- **Componentes de ejecución,** creados como resultado de un sistema en ejecución.

Componentes (interfaces)

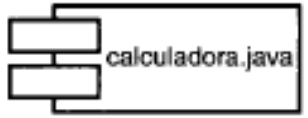
Interfaces

- Cuando se trate de componentes tendrá que tratar con interfaces, dado que en la **orientación a objetos un objeto** oculta al mundo exterior lo que haces (encapsulación u ocultamiento de información). El objeto tiene que presentar un rostro al mundo exterior a esto se conoce como interfaz del objeto.
- La **interfaz** es un conjunto de operaciones que especifica algo respecto al comportamiento de una clase.
- Una interfaz puede se imaginado como una clase que **solo contiene operaciones** (no atributos), por lo que la interfaz es un conjunto de operaciones que representa una clase u otras.
- Una interfaz puede ser física o conceptual. La interfaz que utiliza una clase es la misma que la que utiliza su implementación de software (componente). Como modelador, esto significa que la misma forma en que represente una interfaz para una clase representara una interfaz para un componente.
- Un componente puede hacer disponible su interfaz para que otras componentes puedan utilizar las operaciones que contiene. El componente que proporciona los servicios se dice que provee una **interfaz de exportación**, y la que accede a los servicios se dice que utiliza una **interfaz de importación.**

Diagrama de componentes (características y representación)

Diagrama de componentes

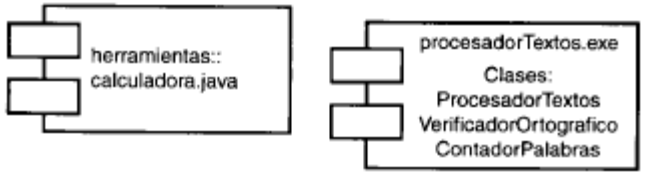
- Un diagrama de componentes contiene componentes, interfaces y relaciones.



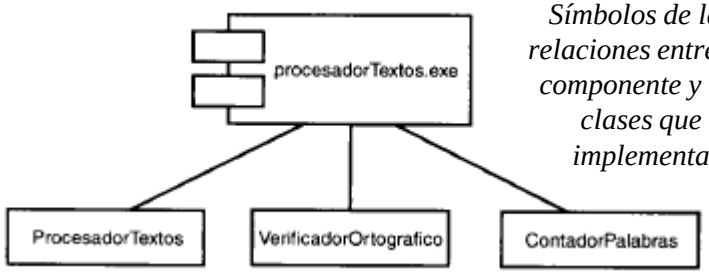
Símbolo que representa a un componente

Representación de un diagrama de componentes

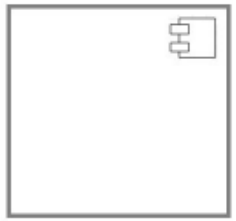
- El símbolo principal de un diagrama de componentes es un **rectángulo** que tiene otros dos sobrepuestos en su lado izquierdo.
- Se debe colocar el nombre del componentes dentro del símbolo, siendo el nombre una cadena.
- Si el componente es miembro de un paquete, puede utilizar el nombre del paquete como prefijo para el nombre del componente. También puede agregar información que muestre algún detalle del componente.
- La figura de la derecha muestra las clases que implementa un componente en particular. Aunque es una forma de hacer esto, por lo general desordena el diagrama, por lo que se debe ver las relaciones de dependencia entre el componentes y las clases.
- En ultimas versiones de UML un componente consiste en un rectángulo con un rectángulo mas pequeño en la esquina superior derecha con pestanas



Adición de información al símbolo del componente



Símbolos de las relaciones entre un componente y las clases que implementa



Otra notación de componente

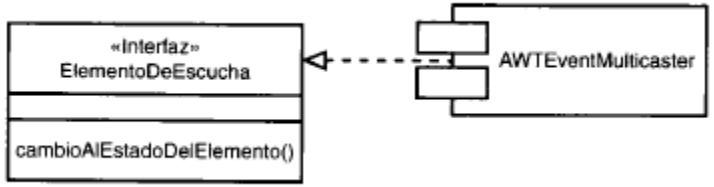
Diagrama de componentes (representación de interfaces y dependencia)

Representar las interfaces

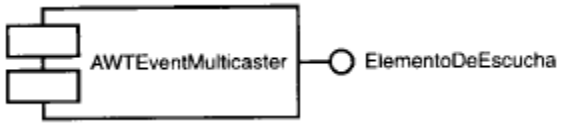
Existen dos forma para representar a un componente y sus interfaces.

- La primera muestra la interfaz, como un rectángulo que contiene la información que se le relaciona, conectando al componente por la línea discontinua y una punta de flecha representada por un triangulo sin rellenar que visualiza la realización.

- La segunda forma de representar un componente y sus interfaces es representar la interfaz como un pequeño circulo que se conecta al componente por una línea continua. En este contexto, la línea representa la relación de realización.



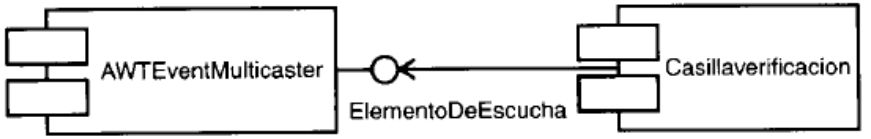
Puede representar a una interfaz como un rectángulo con información, conectado al componente por una flecha de realización.



Puede representar a una interfaz como un pequeño circulo, conectado al componente por una línea continua que, en este contexto se interpreta como realización

Dependencia

- Además de la realización, puede representar a la **dependencia**, que es la relación entre un *componente y una interfaz de importación*.
- La dependencia se vislumbra con una línea discontinua con una punta de flecha. Puede mostrar la realización y la dependencia en el mismo diagrama.



Una interfaz que realiza un componente y otra de la que depende

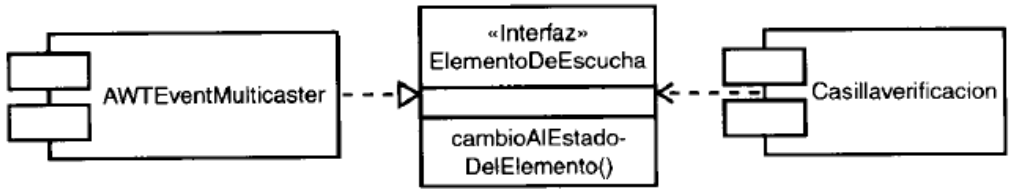
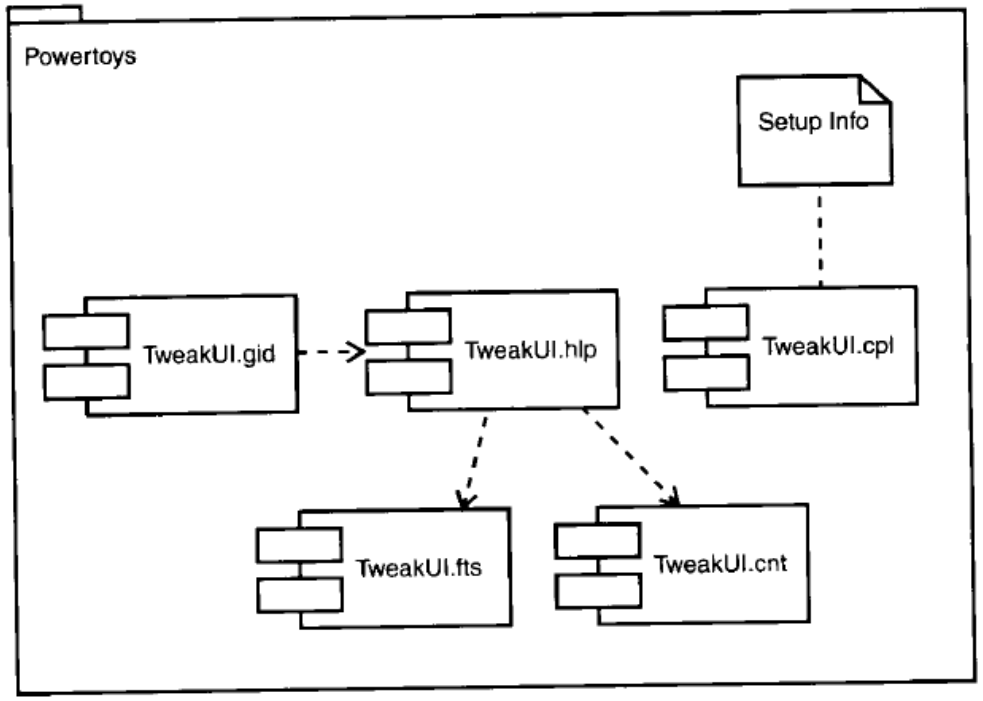


Diagrama de componentes (ejemplo de aplicación de los diagramas de componentes)

Ejemplo utilizando un paquete llamado PowerToys del sitio web de Microsoft.

- Si se utiliza cualquier versión de Win32, estará familiarizado con las flechas pequeñas que se encuentran en la esquina inferior izquierda de cada icono de acceso directo.
- Microsoft cuenta con un paquete llamado PowerToys que permite eliminar estas flechas y hacer varias otras cosas con la GUI, mediante una aplicación llamada TwerUI que es parte del paquete.
- Dentro del sitio Web de Microsoft se puede obtener PowerToys gratis y al descomprimirse se verán varios archivos con extensiones .dll, así como un archivo de ayuda y otro .CNT.
- Haciendo click en el archivo de ayuda se genera un archivo .GID, y utilizando la característica Buscar se creara un archivo .FTS

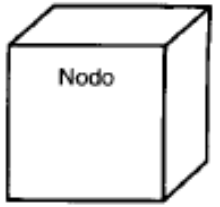


Modelado de TweakUI en el paquete PowerToys

Diagrama de distribución (características y representación)

Diagrama de distribución

- Se encarga de representar una estructura física del sistema, donde cada componente se puede representar como nodos.
- Un nodo es cualquier elemento que sea un recurso de hardware, es decir, es nuestra denominación genérica para los equipos.
- Es posible usar dos tipos de nodos: un procesador, el cual puede ejecutar un componente, y un dispositivo que no lo ejecuta
- Estos diagramas también pueden ser conocidos como diagrama de despliegue.



Representación de un nodo en el UML

Representación de Diagrama de distribución

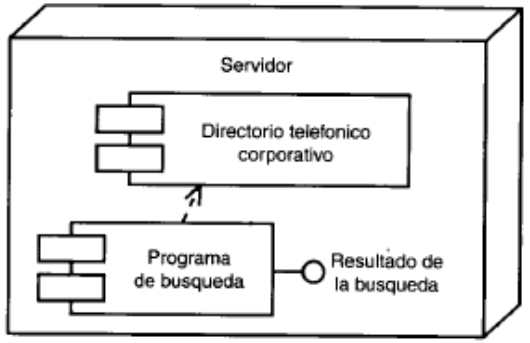
- En UML, un cubo representa un nodo el cual deberá asignarse con un nombre, y podrá utilizar un estereotipo para indicar el tipo de recurso que sea. El nombre es una cadena de texto.
- Si el nodo es parte de un paquete, **su nombre puede contener también el del paquete**. Puede dividir el cubo en compartimientos que agreguen información (como componentes colocados en el nodo).

Como agregar información a un nodo

Diagrama de distribución (relaciones de dependencia y estereotipo)

Relaciones de dependencia de un nodo

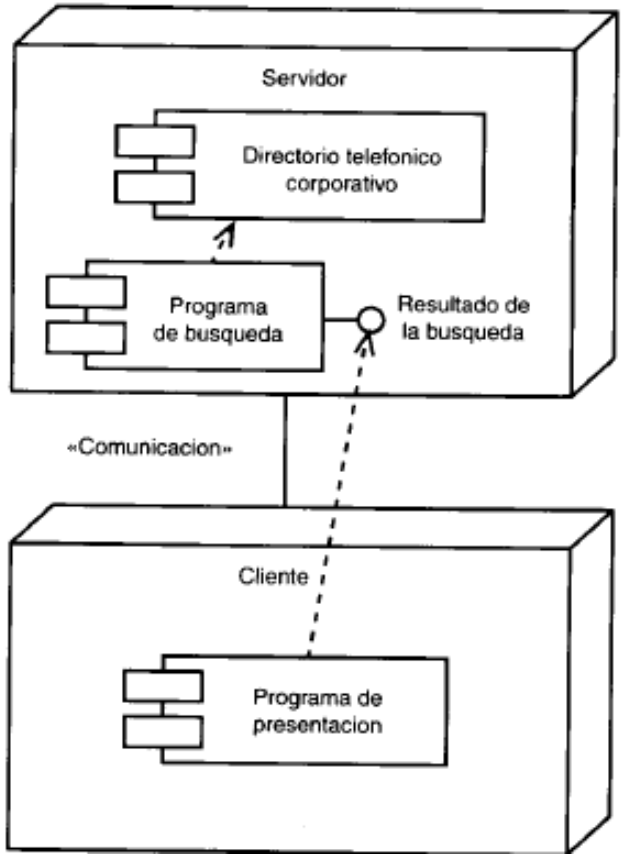
- Otra forma de indicar los componentes distribuidos es la de mostrarlos en relaciones de dependencia con un nodo.



Puede mostrar los componentes en relaciones de dependencia con un nodo

Estereotipo

- Una **línea** que asocia a dos cubos representara una conexión entre ellos. Podrá utilizar un estereotipo para dar información respecto a la conexión.
- Tenga en cuenta que una conexión no es necesariamente un cable o alambre. También puede visualizar conexiones inalámbricas como las infrarrojas o satelitales.
- Aunque la **conexión** es el tipo común de asociación entre dos nodos, es posible utilizar otros (como la agregación o la dependencia).



Representación de conexiones entre nodos

Diagrama de distribución (ejemplo de un equipo domestico)

Diagrama de distribución de un equipo de computo domestico

- Para modelar un equipo de computo, se ha incluido el procesador y los dispositivos, a la vez de que se ha modelado la conexión telefónica del proveedor de servicios de internet y su conexión.
- La nube que representa Internet no es parte de una simbología UML, sin embargo es útil para clarificar el modelo.

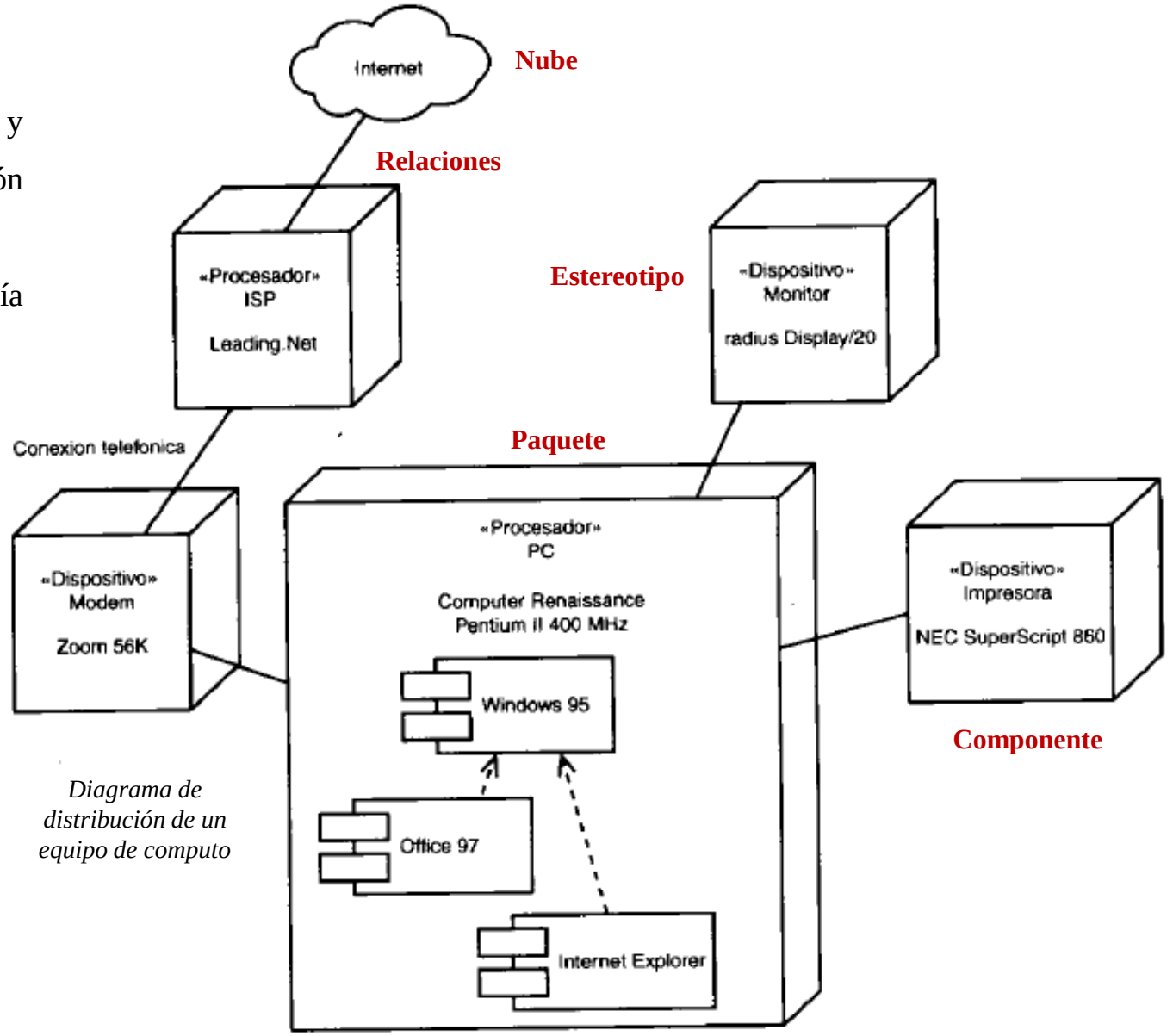
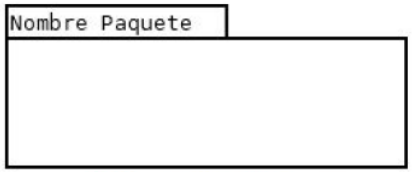


Diagrama de distribución de un equipo de computo

Diagrama de paquete (características y representación)

Diagrama de paquete

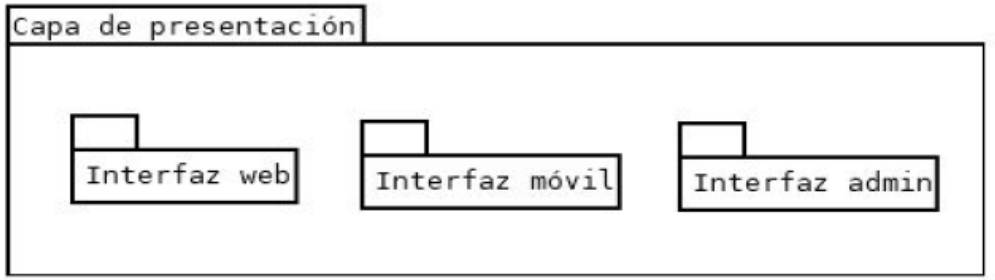
- Los diagramas de paquete sirven para organizar los elementos del software en un grupo, al cual se le denomina paquete.
- Un paquete es un conjunto de clases, casos de uso, componentes u otros paquetes.



Representación de un diagrama paquete

Representación de Diagrama de paquete

- Es importante al identificar el nombre del paquete que este sea representativo de las funciones que lo integran.
- Se representa con un símbolo simulando una carpeta, con el nombre en la parte superior.
- Un paquete puede contener otros paquetes.

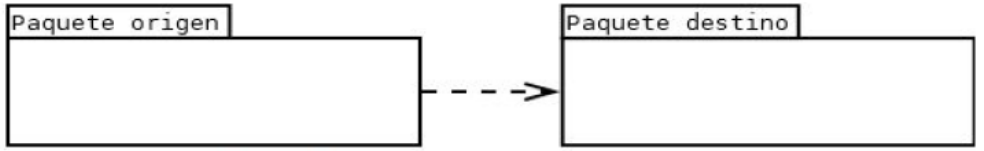


Notación de un paquete que contiene otros paquetes

Diagrama de paquete (dependencia y ejemplo)

Dependencia entre paquetes

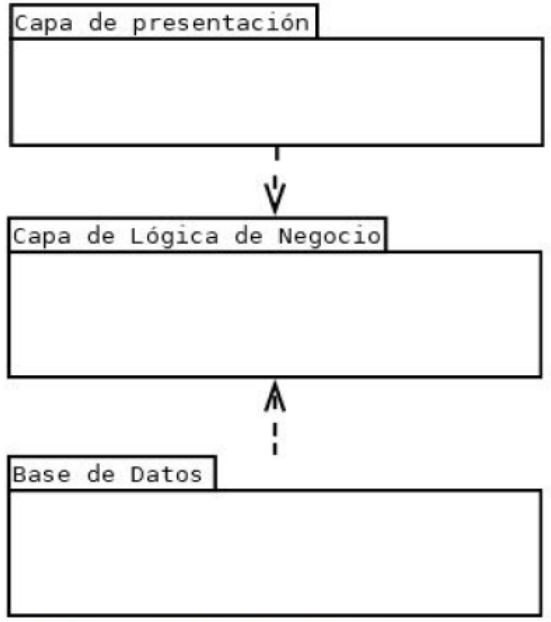
- Una dependencia entre paquetes representa que un paquete necesita de los elementos incluidos en otro paquete para poder funcionar.
- Una dependencia se representa con una flecha discontinua que va desde el paquete que requiere la función hasta el paquete que ofrece dicha función.



Paquete que incluye otros paquetes

Ejemplo de relación de dependencia

- Un ejemplo de relación de dependencia que puede encontrarse es un patrón de diseño de arquitectura de 3 capas: Capa de lógica de negocio, capa de presentación y base de datos.



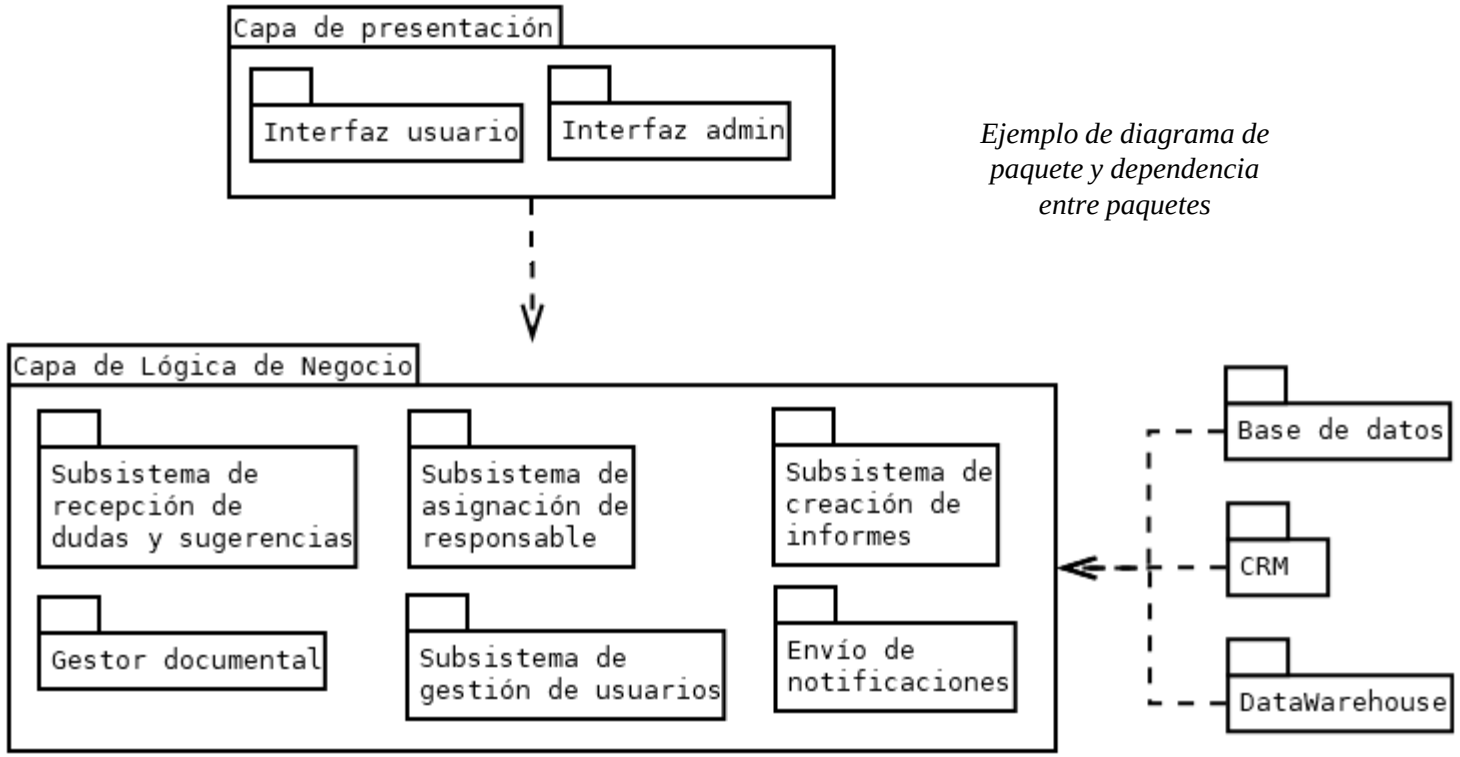
Ejemplo de dependencia entre paquetes

Diagrama de paquete (Ejemplo)

Ejemplo de un diagrama de paquetes

- La aplicación tiene como objetivo la recepción y gestión de quejas y sugerencias, la cual esta compuesta por los paquetes:

1. Capa de presentación, que incluye a su vez los paquetes:
   - Interfaz de usuario
   - Interfaz Admin.
2. Capa de lógica de negocio, con los paquetes
   - Subsistema de recepción de dudas y sugerencias
   - Subsistema de asignación de responsable
   - Subsistema de creación de informes
   - Gestor documental
   - Subsistema de gestión de usuarios
   - Envio de notificaciones.
3. Base de datos
4. CRM
5. DataWarehouse



Ejemplo de diagrama de paquete y dependencia entre paquetes

Bibliografía

Pressman, R. S. (2010). Ingeniería de Software, Un enfoque practico Séptima Edición. Ciudad de México: Mc Graw Hill.

Sommerville. (2011). Ingeniería de Software 9 Edición. Estado de México: Pearson.

UNID Universidad Interamericana para el desarrollo. (2018). Ingeniería de software. Ciudad de México: UNID.

J. Schmuller. (2010). Aprendiendo UML en 24 horas. Ciudad de México: Prentice Hall.