unified modeling language

Resumen de

Términos de UML

Introducción a UML:

- UML es un lenguaje de modelado de software:
  - Proporciona un vocabulario y reglas para crear modelos software.
  - Suficientemente expresivo para cubrir distintas vistas de la arquitectura del software a lo largo del ciclo de vida.
  - Mayor nivel de abstracción que un lenguaje de programación.

- UML es un lenguaje para visualizar los elementos de un gran sistema software, facilitando:
  - la comunicación entre los participantes (incluidas herramientas) en el desarrollo,
  - la comprensión de las soluciones (notación gráfica),
  - el mantenimiento de las soluciones conceptuales a lo largo del tiempo (documentación).

- UML es un lenguaje para especificar software:
  - Se pueden construir modelos precisos, no ambiguos y completos.
  - Cubre las decisiones de análisis, diseño e implementación.

- UML es un lenguaje para construir software:
  - No es un lenguaje de programación visual, pero sus modelos se pueden conectar de forma directa a una gran variedad de ellos.
  - Correspondencias entre UML y lenguajes: Java, C++, etc.
  - Ingeniería directa: generación de código.
  - Ingeniería inversa: reconstrucción de modelos.

- UML es un lenguaje para documentar:
  - requisitos, arquitectura, diseño, código fuente, pruebas, ...

- El modelo conceptual está compuesto por 3 bloques de construcción básicos:
  - Elementos
    - Abstracciones básicas a partir de las que se construyen los modelos
  - Relaciones
    - Entre los elementos
  - Diagramas
    - Grupo consistente de elementos y sus relaciones

- La documentación con UML se basa en el uso de los diagramas:

  - Diagrama de clases
  - Diagrama de casos de uso
  - Diagrama de secuencia
  - Diagrama de colaboración
  - Diagrama de estados
  - Diagrama de actividades
  - Diagrama de componentes
  - Diagrama de despliegue

Definiciones:

- ***Caso de uso***; descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable, de valor para un actor. Un caso de uso es realizado por una colaboración.

  En relación con los escenarios, un caso de uso es un conjunto de escenarios, siendo un escenario una secuencia de acciones que ilustra un comportamiento, con lo cual un caso de uso describe un conjunto de comportamientos.

  Un caso de uso captura el comportamiento esperado de un sistema, subsistema, clase o interface que se está desarrollando, sin tener que especificar cómo se implementa ese comportamiento.

  Esto es importante porque el análisis del sistema no debería estar influenciado mientras sea posible por cuestiones de implementación, el **qué** frente al **cómo**. Lo que implica el diseño funcional, **el qué**, frente al diseño detallado, **el cómo**.

  Un caso de uso, a la hora de implementarse, se realizará a través de una **colaboración** entre clases y otros elementos que colaboran entre si para llevar a cabo ese comportamiento. Esta sociedad de elementos, tanto su estructura estática como dinámica, se modela en UML como una colaboración.

  Un caso de uso sigue normalmente cuatro fases:
    - El actor envía al sistema una petición y los datos necesarios para llevarla a cabo
    - El sistema valida la petición y los datos
    - El sistema altera su estado interno
    - El sistema devuelve el resultado al actor

- ***Actor***; conjunto coherente de roles que juegan los usuarios de los casos de usos cuando interactúan con estos. Normalmente representan a una persona, un dispositivo hardware u otro sistema al interactuar con el nuestro
  Se pueden definir categorías generales de actores y especializarlos a través de la relaciones de generalización
  Los actores se conectan a los casos de uso mediante asociaciones.

- ***Diagrama de casos de uso***; muestra un conjunto de casos de uso y actores junto con sus relaciones.

  El objetivo es lograr claridad sobre lo que desea el usuario y la forma en la que se va a presentar la solución que se está buscando.
  Muestra las operaciones que se esperan de la aplicación y sus relaciones con el entorno (usuarios u otras aplicaciones).
  Los elementos que intervienen son:

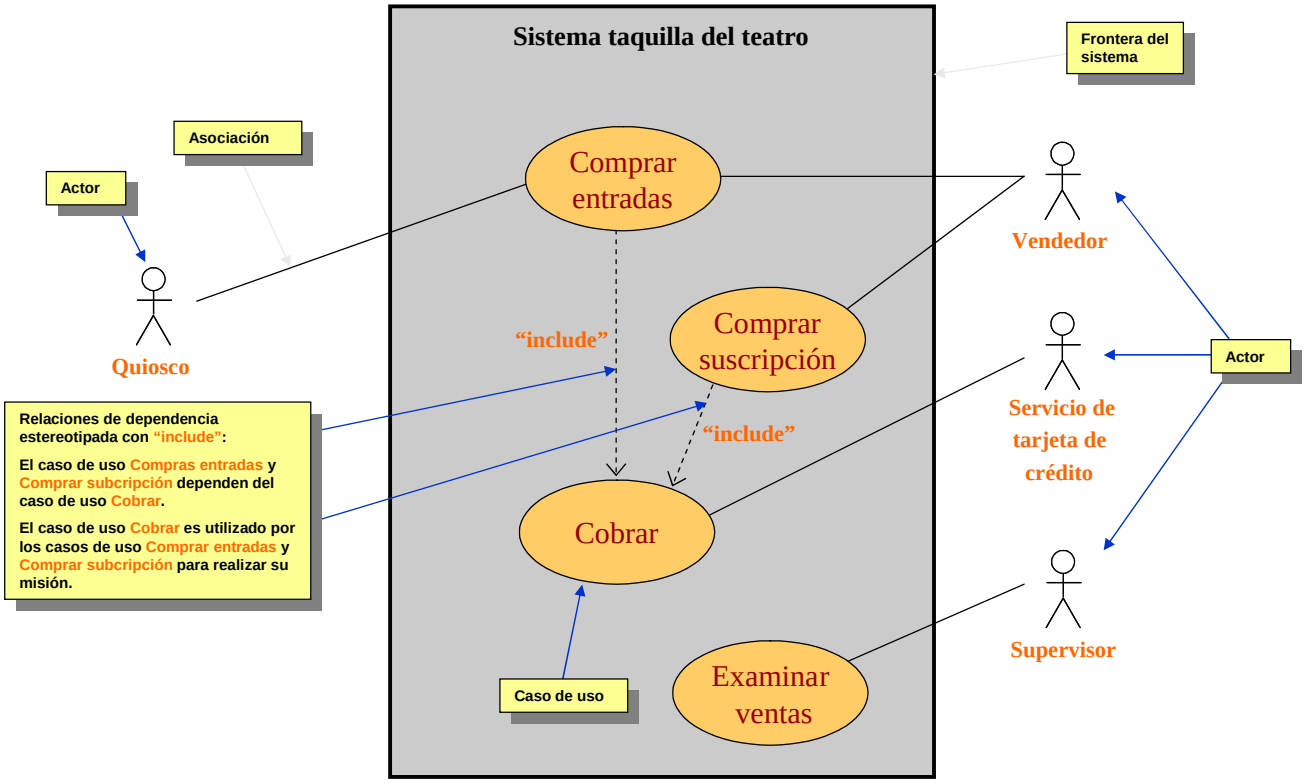
    - Los actores
    - Los Casos de Uso
    - Relaciones de dependencia, generalización y asociación

  Se utilizan para especificar el comportamiento deseado del un sistema o subsistema:

    - Describe el conjunto de **secuencias de acciones** que lleva a cabo el sistema para producir un resultado para un **actor.**
    - Capturan el comportamiento deseado del sistema, sin especificar como se lleva a cabo dicho comportamiento

Principalmente son un medio de comunicación para que los desarrolladores y los usuarios lleguen a un consenso en la especificación
Ayudan a **validar** el sistema durante el desarrollo
Los casos de uso son principalmente descripciones textuales
La notación gráfica de UML (**diagrama de casos de uso**) solo muestra los nombres y sus relaciones
Al ser textuales, son informales y no son buenas para razonar acerca del sistema

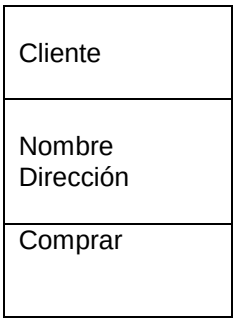
Ejemplo:



Muestra los casos de uso de la taquilla de un teatro.
Por caso de uso se entiende una descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable, de valor para un actor.
Dicho esto, el diagrama de casos de uso describe las acciones que realiza el sistema cuando interactúa con los usuarios del mismo.
Los actores son el **Vendedor**, el **Supervisor**, el **Quiosco** y el **Servicio de tarjetas de crédito**. El quiosco es otro sistema que acepta pedidos de un cliente, por lo que el cliente no es un actor en la aplicación de la taquilla.
Los casos de uso de la aplicación, **Comprar entradas**, **Comprar subscripción**, **Cobrar** y **Examinar ventas**, son las actividades que el sistema realiza al comunicarse con los actores. Respecto al caso de uso **Cobrar**, forma parte de los casos de uso **Comprar entradas** y **Comprar Subscripción**, y estos casos de uso dependen del caso de uso **Cobrar**.
El caso de uso **Cobrar** está relacionado con el actor **Servicio de tarjetas de crédito**, que representa otro sistema que se encarga de las tarjetas de crédito.

- ***Clase***; descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Describe estructura de los objetos de un sistema:
  - identidad,
  - relaciones con otros objetos,
  - atributos y operaciones

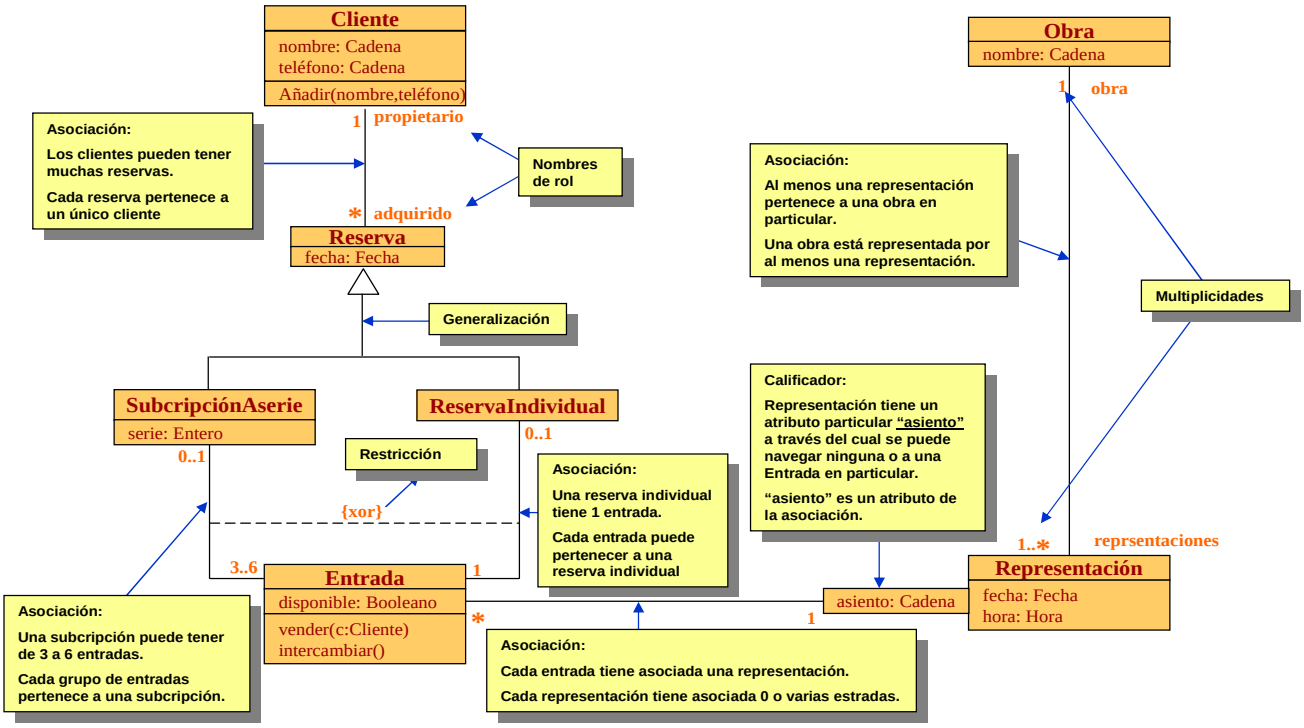
  Es el más característico del Diseño O.O.
  Gráficamente se representan en los diagramas de clases
  Las clases se representan mediante un rectángulo con tres divisiones internas.
  La primera de ellas especifica el nombre de la clase, la segunda los atributos de la misma y la tercera los métodos asociados.



- ***Diagrama de clases***; diagrama que muestra un conjunto de clases, interfaces y colaboraciones y sus relaciones. Muestra una colección de elementos declarativos estáticos del modelo.

  El objetivo es mostrar el conjunto de clases que componen el sistema, junto con las relaciones que existen entre estas.
  Se utilizan para modelar el diseño estático de un sistema
  Los elementos que intervienen son:
  - Clases
  - Relaciones de dependencia, generalización y asociación.
  - Paquetes
  - Interfaces

Diagrama de clases



Contiene parte del modelo del dominio "venta de entradas".

El diagrama describe la estructura, u organización, de las clases necesarias para que una persona asista a una obra de teatro, a través de la adquisición de entradas.

Las clases que se contemplan son **Cliente**, **Reserva**, **SubcripciónAserie**, **SubcripciónIndividual**, **Entrada**, **Representación** y **Obra**.

- Un cliente, expresado como **1Cliente**, puede adquirir ninguna, 1 o muchas reservas, expresado como ***Reserva**.

- Un tipo de **Reserva** puede ser una **ReservaIndividual,** que deriva de la clase **Reserva,** y otro tipo una **SubcripciónAserie**, que deriva de la clase **Reserva**.

- Una entrada, expresado como **1Entrada**, puede pertenecer a una o ninguna reserva individual, expresado como **0..1ReservaIndividual**, o bien de 3 a 6 entradas, expresado como **3..6Entrada**, pueden pertenecer a una o ninguna subscripción a serie, expresado como **0..1SubcripciónAserie**.

- Una representación, expresado como **1Representación**, puede tener muchas o ninguna entrada, expresado como ***Entrada**, además desde una representación podemos encontrar una entrada en particular a través del calificador *asiento*. Se dice que la clase **Representación** es una clase calificada por el atributo *asiento* de la asociación.

- Una o varias representaciones, expresado como **1..*Representación**, es de una obra, expresado como **1Obra**.

- ***Interacción.*** Una interacción es un comportamiento que implica un intercambio de mensajes entre varios objetos en un contexto determinado con un objetivo determinado
  Secuencia de mensajes que implementan un comportamiento dentro de una colaboración. Un **mensaje** es la especificación de una comunicación entre objetos que transmite información con el fin de desencadenar una actividad; la recepción de una instancia de un mensaje se considera normalmente una instancia de un evento. Comunicación unidireccional entre dos objetos, un flujo de objeto con la información de un remitente a un receptor.

  Su objetivo es el establecimiento de prototipos de colaboración. Estos prototipos son los escenarios. Un **Escenario** es secuencia específica de acciones que ilustra un comportamiento. Se puede utilizar para ilustrar la interacción o la ejecución de una instancia de un caso de uso.

  Las interacciones se llevan a cabo entre objetos no entre clases
  Un enlace es una conexión semántica entre objetos

  Elementos que intervienen en las interacciones:

  Objetos: instancias concretas de clases
  Enlaces: enlazan instancias y soporte al envío de mensajes
  Mensajes: desencadenan operaciones
  Funciones: implementadas por los extremos de los enlaces

- ***Diagrama de interacción***; muestra una interacción, que consta de un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. El término genérico de interacción abarca los diagramas de colaboración, de secuencia y de actividades.
  Complementan los modelos obtenidos de los casos de uso iniciales y los diagramas de clase
  Son útiles para modelar aspectos dinámicos de cualquier interacción entre cualquier instancia en cualquier vista del sistema (clases, interfaces, componentes,...)
  Las interacciones se pueden "adornar" con restricciones temporales (marcas temporales)

  Hay dos tipos de diagramas: de secuencia y de colaboración.

  - Un **diagrama de secuencia** es un diagrama en el que se destaca la ordenación temporal de los eventos
  - Un **diagrama de colaboración** destaca la organización estructural de los objetos que envían y reciben los mensajes

  Son semánticamente equivalentes
  - Se puede generar uno a partir del otro, sin perdida de información
  - Visualmente, sin embargo, esta información puede ser más difícil de percibir

- ***Diagramas de colaboración***; diagrama de interacción que resalta la organización estructural de los objetos que envían y reciben mensajes.
  Destacan la organización de los objetos que participan en una interacción
  Es un grafo en el que los nodos son los objetos y los arcos los enlaces
  Los arcos se etiquetan con los mensajes que envían y reciben los objetos
  Dan una visión del flujo de control en el contexto de la organización de los objetos que colaboran

  Permiten indicar que objetos actúan como atributos de otros.
  Permiten indicar que objetos son temporales y cuales no.
  Permite indicar que relaciones estructurales actúan en una colaboración

Hay dos características que los distinguen de los diagramas de secuencia:

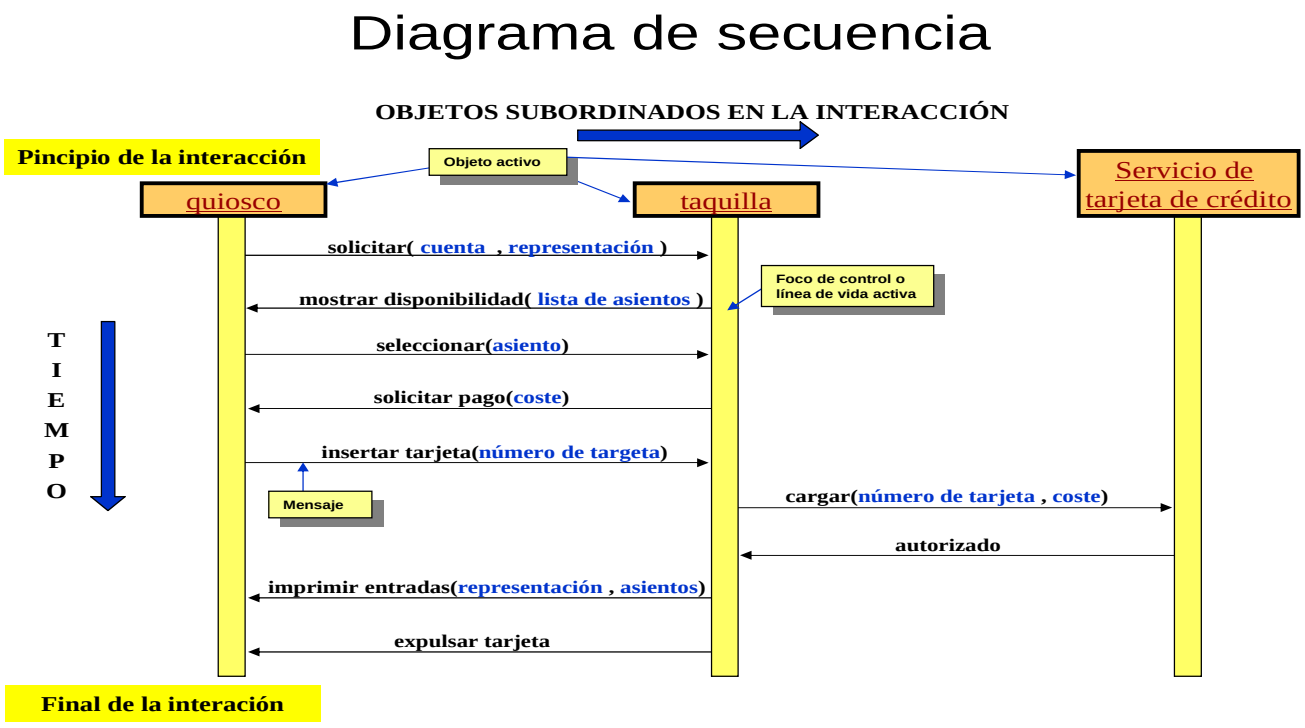
- El camino :Para indicar como se enlazan los objetos se utilizan estereotipos en los extremos de los enlaces (local, global y self)
- El número de secuencia : Para indicar la ordenación de los mensajes se utiliza la numeración decimal de Deway (1, 1.1,.....)

- ***Diagrama de secuencia***; diagrama de interacción que destaca la ordenación temporal de mensajes.
Un diagrama de secuencia muestra la interacción de un conjunto de objetos de una aplicación a través del tiempo. Posibilita la representación de la secuencia temporal de envío de mensajes entre los objetos pertenecientes a las clases diseñadas en el contexto de una operación. El concepto de mensaje en el contexto de los diagramas de secuencia, materializa y unifica todas las formas de comunicación entre objetos.

  Los **Objetos** se representan mediante un rectángulo que encabeza una línea discontinua, que representa la línea de vida del objeto.

  Los **Mensajes** se denotan mediante una flecha dirigida desde el objeto que emite el mensaje hasta el objeto que lo ejecuta.
  Los tipos de mensajes posibles son los siguientes:
  - Simple
  - Síncrono
  - Abandono
  - Time-Out
  - Asíncrono

  Los diagramas de secuencia se suelen asociar a los casos de uso, mostrando como estos se realizan a través de interacciones entre sociedades de objetos

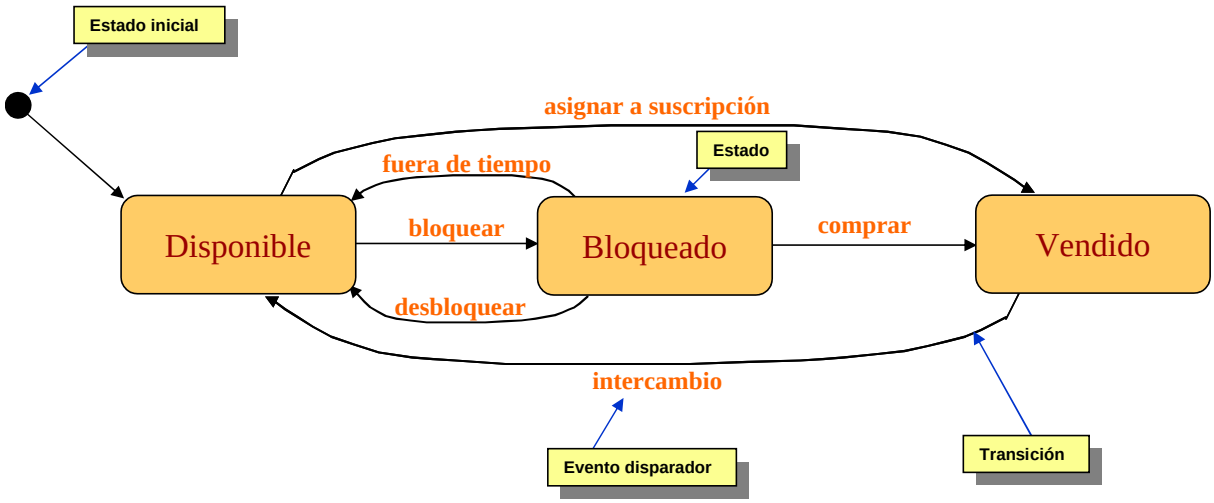


En este caso, se muestra el diagrama de secuencia para el caso de uso **Comprar entrada** este caso de uso lo inicia el cliente en el **Quiosco** comunicándose con la **Taquilla**. Se ha de resaltar que este diagrama está realizado a alto nivel de detalle, representando a los actores **Quiosco**, **Servicio de tarjeta de crédito** y al sistema **Sistema taquilla de teatro** del diagrama de casos de uso, como objetos.

- El objeto **Quiosco** envía una solicitud, solicitar(cuenta, representación), al objeto **Taquilla**, esto implica, visto en el diagrama de casos de uso, que el actor **Quiosco** solicita al **Sistema taquilla de teatro** comprar una entrada.
- El objeto **Taquilla** envía la disponibilidad de asientos, mostrar disponibilidad(lista de asientos), al objeto **Quiosco**, esto implica, visto en el diagrama de casos de uso, que el **Sistema taquilla de teatro** muestra al actor **Quiosco** los asientos que quedan disponibles para la obra.
- El objeto **Quiosco** envía el asiento seleccionado, seleccionar(asiento), al objeto **Taquilla**, esto implica, visto en el diagrama de casos de uso, que el actor **Quiosco** indica al **Sistema taquilla de teatro** que asiento selecciona.
- El objeto **Taquilla** solicita el pago del asiento seleccionado, solicitar pago(coste), al objeto **Quiosco**, esto implica, visto en el diagrama de casos de uso, que el **Sistema taquilla de teatro** solicita al actor **Quiosco** el pago del asiento seleccionado.
- El objeto **Quiosco** envía la tarjeta del cliente, insertar tarjeta(número de tarjeta), al objeto **Taquilla**, esto implica, visto en el diagrama de casos de uso, que el actor **Quiosco** inserta la tarjeta del cliente con el número del mismo en el **Sistema taquilla de teatro**.
- El objeto **Taquilla** envía el número de tarjeta de crédito y el coste, cargar(número de tarjeta, coste), al objeto **Servicio de tarjeta de crédito**, esto implica, visto en el diagrama de casos de uso, que el **Sistema taquilla de teatro** envía al actor **Servicio de tarjeta de crédito**, que es un sistema, el número de tarjeta y el coste para que cobre el asiento reservado.
- El objeto **Servicio de tarjeta de crédito** envía la autorización del cobro, autorizado, al objeto **Taquilla**, esto implica, visto en el diagrama de casos de uso, que el actor **Servicio de tarjeta de crédito**, que es un sistema, envía la autorización del cobro al **Sistema taquilla de teatro**.
- El objeto **Taquilla** envía la impresión de las entradas de la representación junto con los asientos seleccionados, imprimir entradas(representación, asientos), al objeto **Quiosco**, esto implica, visto en el diagrama de casos de uso, que el **Sistema taquilla de teatro** envía al actor **Quiosco** la impresión de las entradas, especificando la representación de la obra y los asientos seleccionados.
- El objeto **Taquilla** envía la tarjeta, del cliente, expulsar tarjeta, al objeto **Quiosco**, esto implica, visto en el diagrama de casos de uso, que el **Sistema taquilla de teatro**, envía al actor **Quiosco** la tarjeta del cliente.

- ***Diagrama de estados***; permiten la representación del ciclo de vida de los objetos.
  El comportamiento de los objetos puede describirse de manera formal en términos de estados y eventos.
  Los objetos que no representan un comportamiento reactivo muy marcado pueden considerarse como objetos que se encuentran siempre en el mismo estado.
  Cuando un objeto muestra un comportamiento se dice que tiene asociado un autómata
  Elementos:
  - Estado: condición o situación de un objeto durante la cual:
    - se satisface alguna condición
    - se realiza alguna actividad
    - se espera algún evento
  - Evento: especificación de un acontecimiento significativo que ocupa un lugar en el tiempo y en el espacio
  - Transición: relación entre dos estados que indica como los objetos cambian de estado (eventos + condiciones)
  - Actividad: ejecución no atómica en curso dentro de una máquina de estados
  - Acción: computación atómica ejecutable que produce un cambio de estado en el modelo o devuelve un valor

Diagrama de estados



Este diagrama de estados muestra los estados de la historia de una entrada para una representación, nos referimos a la clase **Entrada** del diagrama de clases anterior en relación con la clase **Representación**.

Un objeto de tipo **Entrada** puede tener tres estados posibles:

- **Disponible**; en este estado la entrada no esta adquirida, ie puede ser comprada por un cliente.
- **Bloqueado**; en este estado la entrada no puede ser adquirida, ie en este momento un cliente está estudiando la posibilidad de comprar la entrada, pero aún no esta comprada, con lo cual esta entrada en particular no está disponible para que otro cliente distinto.
- **Vendido**; en este estado la entrada ya está vendida, ie el cliente la ha comprado.

Estado inicial del objeto:

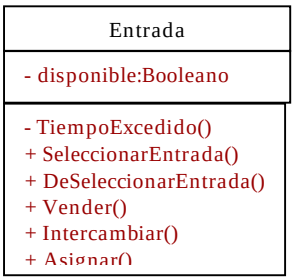
- El estado inicial se indica con un punto negro junto a una flecha, cuyo origen es el punto negro y destino es el estado inicial, **Disponible**.

Descripción de las transiciones:

- **bloquear**, transición desde el estado **Disponible** al estado **Bloqueado**, implica que la clase **Entrada** debe de contar con un método de bloqueo, por ejemplo SeleccionarEntrada(), (se selecciona una entrada en particular con el fin de evaluar su compra).
- **fuera de tiempo**, transición desde el estado **Bloqueado** al estado **Disponible**, implica que la clase **Entrada** debe de contar con un método automático de desbloqueo, que estará en función del tiempo que el objeto esté en el estado **Bloqueado**, por ejemplo TiempoExcedido(), (al seleccionar la entrada, el objeto queda bloqueado, no disponible, paralelamente se inicia un computo de tiempo, que tras excederse llamará al método TiempoExcedido()).

- **desbloquear**, transición desde el estado **Bloqueado** al estado **Disponible**, implica que la clase **Entrada** debe de contar con un método de desbloqueo, a través del cual se pueda deseleccionar la entrada voluntariamente, por ejemplo DeSelecionarEntrada().
- **comprar**, transición desde el estado **Bloqueado** al estado **Vendido**, implica que la clase **Entrada** debe de contar con un método de venta, por ejemplo Vender(), (en este caso la entrada queda no disponible, pero, ya no existe ningún computo de tiempo, ie no se pueden disparar las transiciones **fuera de tiempo** ni **desbloquear**, ya que la entrada ha sido vendida).
- **intercambio**, transición desde el estado **Vendido** al estado **Disponible**, implica que la clase **Entrada** debe de contar con un método para intercambiar una entrada que ya ha sido vendida, por ejemplo Intercambiar().
- **asignar a suscripción**, transición desde el estado **Disponible** al estado **Vendido**, implica que la clase **Entrada** debe de contar con un método de asignación directa para una entrada que ya ha sido vendida, por ejemplo Asignar().

Una vez analizado el diagrama de estados, obtenemos una primera aproximación para construir la clase **Entrada**:



- ***Actividad***; ejecución no atómica en curso, dentro de una máquina de estados. Las actividades producen alguna acción, compuesta de acciones atómicas ejecutables que producen un cambio en el estado del sistema o devuelven un valor. Una actividad puede ser interrumpida por otros eventos.

- ***Diagramas de actividades***; muestra el flujo de control entre actividades.
  El Diagrama de Actividades esta orientado a la representación de las operaciones denominadas acciones y en definitiva a mostrar el comportamiento interno de un método.
  Los elementos principales son:
    - Actividades
    - Transiciones

  Se pueden considerar un caso especial del Diagrama de Estados en el que:
    - La mayoría de los estados son actividades
    - La mayoría de las transiciones se disparan implícitamente por la terminación de acciones

  Diferencias con un Diagrama de Estados.
    - Un diagrama de actividad se usa para modelar una secuencia de acciones en un proceso. Simplifica las representaciones que pueden realizarse en un diagrama de estados
    - Un Diagrama de Estados para modelar los estados discretos de un objeto a lo largo de su ciclo de vida.

Ejemplo:

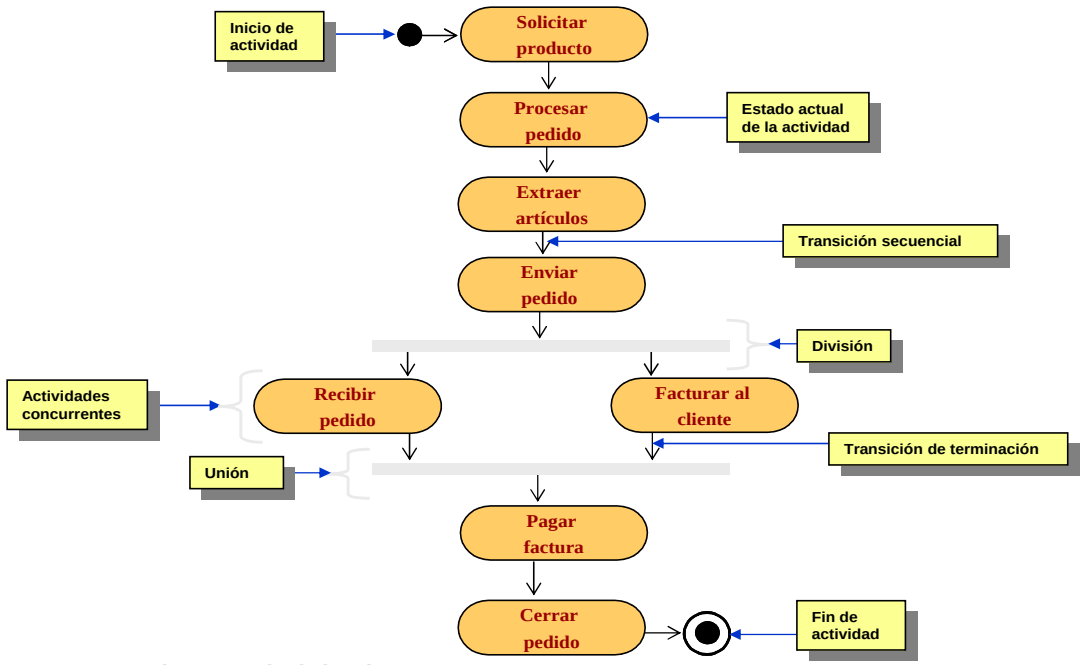


Diagrama de Actividades 1

Diagrama de actividades 1

Muestra el flujo de control entre actividades a realizar por un sistema. Una actividad es una ejecución no atómica dentro de una máquina de estados. Las actividades se pueden descomponer en subactividades, siendo estas computaciones atómicas ejecutables.

Muestra las actividades a realizar por un sistema de pedidos, realizado por un cliente. Los pasos a pegar por el sistema son:

Proceso básico de gestión de clientes, ventas y almacén

Un cliente solicita un producto, cuando lo recibe lo verifica y paga la factura

El departamento de ventas procesa el pedido del cliente solicitándolo al almacén, se lo factura al cliente y cuando cobra cierra el pedido

El almacén proporciona el producto solicitado y lo envía al cliente

La actividad inicial es a la que apunta un círculo negro, en este caso la actividad inicial es **Solicitar producto**. Y por otro lado la última actividad es aquella cuya transición de salida tiene como destino un círculo hueco que contiene a un círculo negro, en este caso la última actividad es **Cerrar pedido**.

En este diagrama aparece una división de transiciones, la actividad origen **Enviar pedido**, dispara una transición, tras haber terminado, que se bifurca en dos transiciones, cuyos destinos son los estados de actividades **Recibir pedido** y **Facturar al cliente**.

Por otro lado, en este diagrama aparece una unión de transiciones, las actividades de origen **Recibir pedido** y **Facturar al cliente**, tras terminar disparan sus respectivas transiciones, uniéndose en una única transición cuyo destino es la actividad **Pagar factura**. Si una transición es disparada antes que otra, la unión de transiciones no se dispara hasta que la otra transición haya sido disparada. En otras palabras, una unión de transiciones no se realiza hasta que todas las transiciones origen hayan sido disparadas.

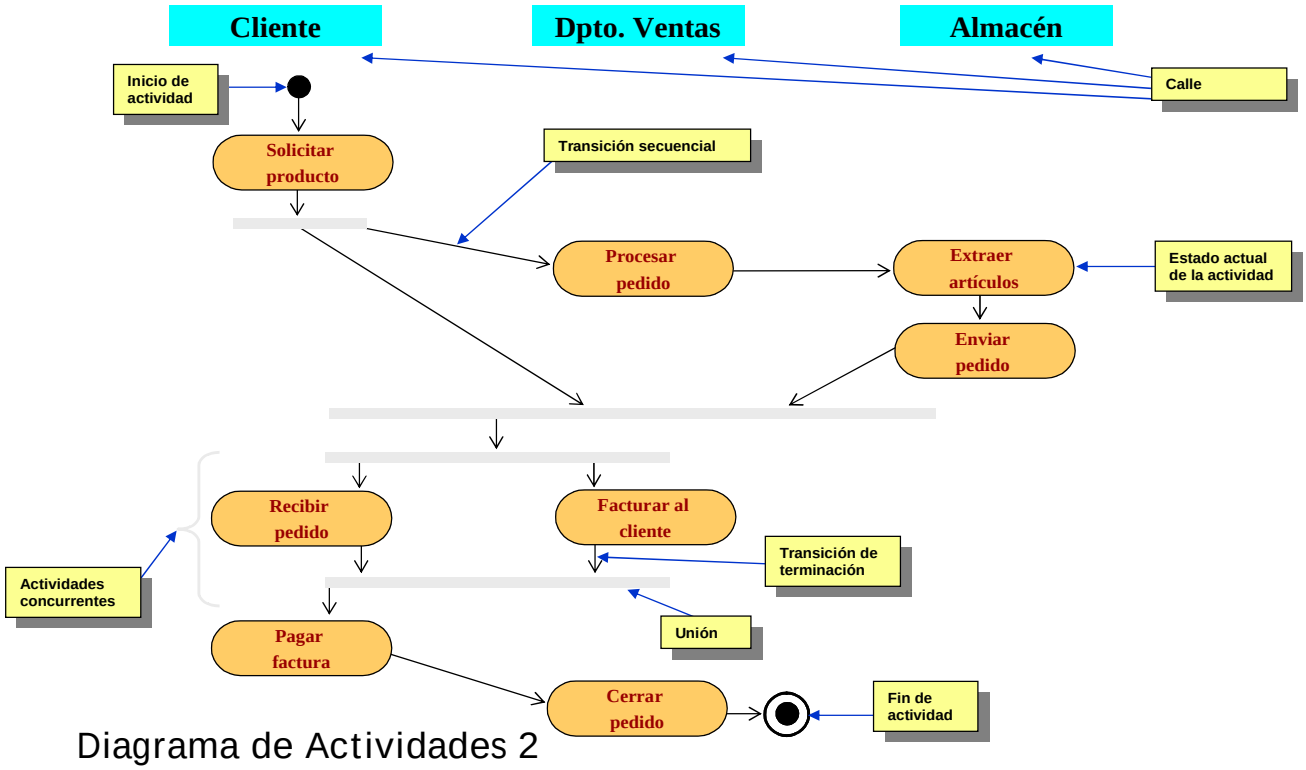


Diagrama de Actividades 2

Diagrama de actividades 2

Este diagrama es el mismo que el anterior, pero en este caso se distribuyen las actividades por calles. Una calle marca las responsabilidades que corresponde a un usuario o rol, particular, en el diagrama, en este caso, las calles son **Cliente**, **Dpto. Ventas** y **Almacén**.

Cada una de las calles tiene sus propias actividades:

Cliente; las actividades para esta calle son **Solicitar producto**, **Recibir pedido** y **Pagar factura**. Es decir, estas actividades le corresponde realizarlas al cliente.

Dpto. Ventas; las actividades para esta calle son **Procesar pedido**, **Facturar al cliente** y **Cerrar pedido**. Es decir, estas actividades le corresponde realizarlas al departamento de ventas.

Almacén; las actividades para esta calle son **Extraer artículos** y **Enviar pedido**. Es decir estas actividades le corresponde realizarlas al almacén.

En este caso aparece el diagrama de distinta forma, con una división de transiciones más y una unión de transiciones más.

La primera división, que tiene como actividad de origen **Solicitar pedido**, tiene una actividad origen **Procesar pedido** y, por otro lado, una unión de transiciones (esta transición es necesaria debido a haber utilizado calles en el diagrama de actividades). Hasta que la actividad **Enviar pedido** no haya lanzado su transición de terminación, no se dispara la unión de transiciones, (esta unión de transiciones es necesaria debido a haber utilizado calles en el diagrama de actividades).

- ***Componente***; parte física y reemplazable de un sistema que conforma un conjunto de interfaces y proporciona la realización de los mismos. Se puede ver como un empaquetado de componentes lógicos como clases, interfaces y colaboraciones.

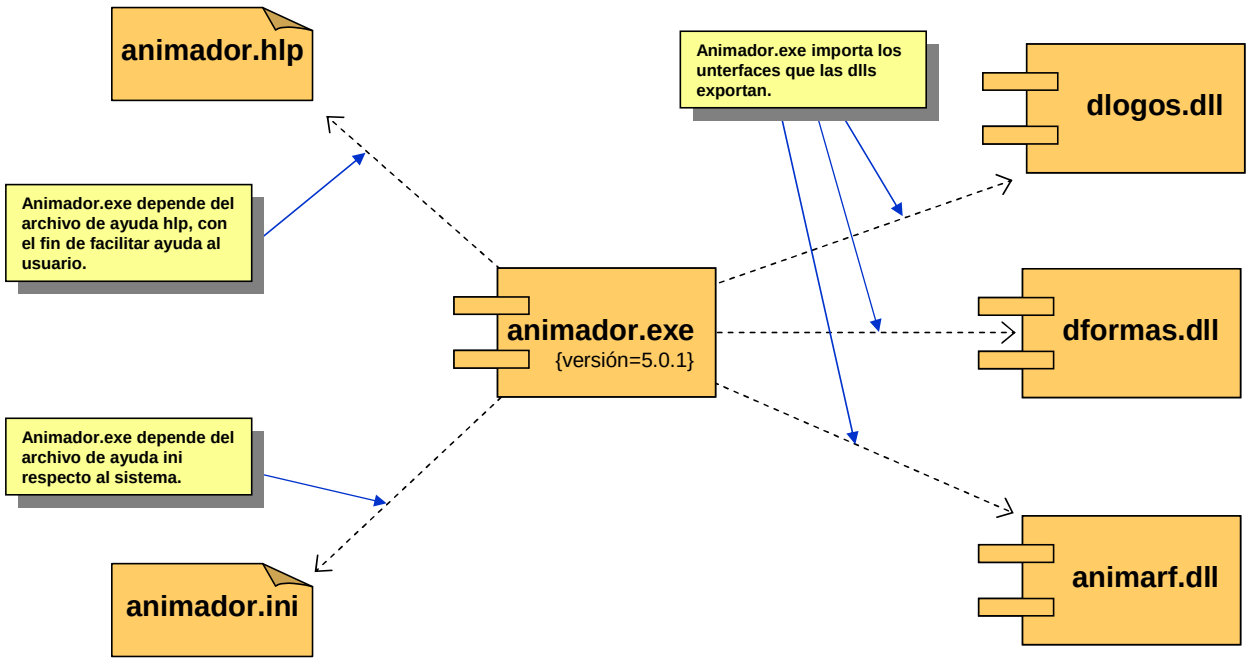
  En muchos sentidos los componentes son como las clases:
  - Ambos tienen nombre
  - Ambos pueden realizar un conjunto de interfaces
  - Ambos pueden participar en relaciones de dependencia, generalización y asociación
  - Ambos pueden anidarse
  - Ambos pueden participar en interacciones

  Sin embargo, hay diferencias significativas. Las diferencias entre componentes y clases son:
  - Las clases representan abstracciones lógicas y los componentes elementos físicos (secuencias de bits)
  - Los componentes residen directamente en un nodo
  - Los componentes representan el empaquetamiento físico de elementos lógicos (de un distinto nivel de abstracción)
  - Las clases tienen atributos y operaciones directamente accesibles, los componentes sólo tienen operaciones alcanzables a través de sus interfaces

- ***Diagramas de componentes***; muestra la organización y las dependencias entre un conjunto de componentes.
  Se utilizan para modelar los elementos físicos que forman parte del sistema
  - Ejecutables
  - Bibliotecas
  - Tablas
  - Archivos
  - Documentos

  Deben definir abstracciones precisas con interfaces bien definidos y que permitan la reemplazabilidad

Diagrama de componentes



Muestra la organización de un programa llamado animador.exe. Este programa se utiliza para dibujar cualquier tipo de figura o forma, muñecos, formas geométricas, etc.…, y darlas animación. Además se pueden dibujar logos para cada animación.

animador.exe importa los interfaces de tres componentes **dlogos.dll**, **dformas.dll** y **animarf.dll**, esta importación se indica mediante relaciones de dependencia, (esto no significa que **animador.exe** dependa de los tres componentes, esto significa que importa una serie de interfaces que las librerías dinámicas exportan, de hecho estos componentes podrían ser sustituidos por otros, siempre y cuando estos otros componentes dieran soporte a estas interfaces).

Descripción de cada uno de estos componentes:

- **dlogos.dll**; este componente soporta los interfaces que posibilita a **animador.exe** dibujar el logo para cada animación realizada.
- **dformas.dll**; este componente soporta los interfaces que posibilita a **animador.exe** dibujar formas geométricas, muñecos, etc.…
- **animarf.dll**; este componente soporta los interfaces que posibilita a **animador.exe** animar lo que se ha dibujado.

En este diagrama existen dos archivos **animador.hlp** y **animador.ini**, que no son componentes, respecto a estos dos archivos **animador.exe** depende, para la ayuda del programa y para la relación de la aplicación con el sistema respectivamente, por esta razón se indica explícitamente la dependencia de **animador.exe** con estos dos archivos.

- ***Diagramas de objetos***; muestra un conjunto de objetos y sus enlaces en un momento dado.

  Los diagramas de objetos son para visualizar, especificar, y documentar elementos estructurales, también para construir los aspectos estáticos de sistemas a través de ingeniería directa e inversa.

  - Visto desde un diagrama de clases; representa las instancias de los elementos de estos diagramas. Los diagramas de objetos se emplean para modelar la vista de diseño estática y la vista de procesos estática de un sistema, a igual que un diagrama de clases pero desde un punto de la perspectiva de instancias reales o prototípicas.

  - Visto desde un diagrama de interacción; representa los objetos que colaboran en una interacción (diagramas de colaboración), pero sin mostrar el paso de mensajes.

- ***Objeto***;
  1. Manifestación concreta de una abstracción; entidad con unos límites bien definidos e identidad que encapsula estado y comportamiento.

  2. Entidad de ejecución con identidad, que puede ser distinguida del resto de las identidades de ejecución.

  3. Cada objeto es una instancia directa de una clase que lo describe totalmente y una instancia indirecta de sus antecesores.