PROGRAMACIÓN ORIENTADA A OBJETOS

Master de Computación

II MODELOS y HERRAMIENTAS UML

II.4 UML: Modelado dinámico

Modelo dinámico

- El *modelo dinámico* está constituido por los aspectos de un sistema relacionados con el tiempo y con los cambios en los objetos y sus relaciones a lo largo del tiempo.
  - Con el modelo dinámico se describe el *control* en el sistema, es decir, las secuencias de operaciones que ocurren como respuesta a estímulos externos, sin tener en cuenta lo que hacen las operaciones, sobre qué operan o cómo se implementan.
- Los objetos se comunican entre sí mediante el envío de mensajes.
  - La comunicación entre un conjunto de objetos para realizar alguna función se denomina *interacción*.
- Los diagramas de *estado*, *interacción* (*secuencia*, *comunicación*, *tiempo* y *visión de conjunto*) y *actividad* se usan para describir cómo los objetos interactúan dinámicamente en diferentes momentos durante la ejecución del sistema.

Diagrama de estado

- Los *diagramas de estado* capturan los ciclos de vida de los objetos, subsistemas y sistemas.
  - Es una representación gráfica de una máquina de estado finita.
  - Especifica la secuencia de estados de un objeto a lo largo de su ciclo de vida como consecuencia de los eventos que recibe, junto con las respuestas del objeto a esos eventos.
- Los principales conceptos de un diagrama de estado son los *eventos* y los *estados*.
  - Los valores de los atributos de un objeto y los enlaces que mantiene constituyen su *estado*.
  - A lo largo del tiempo, los objetos se estimulan entre sí dando lugar a una serie de cambios en sus estados. Un estímulo individual de un objeto a otro es un *evento*.
  - La respuesta a un evento depende del estado del objeto que lo recibe y puede incluir un cambio de estado o el envío de otro evento al emisor original o a un tercer objeto.

Diagrama de estado

- El patrón de eventos, estados y transiciones de estado para una clase dada puede abstraerse y representarse como un *diagrama de estado*.
- Los diagramas de estados muestran cómo reaccionan los objetos a los eventos y cómo cambian su estado interno.
  - Un diagrama de estado es una cadena de estados y eventos, del mismo modo que un diagrama de clases es una cadena de clases y relaciones.
- El *modelo dinámico* consiste en múltiples diagramas de estado y muestra el patrón de actividad para el sistema completo.
- Los diagramas de estado se ejecutan en concurrencia y pueden cambiar de estado independientemente.

Eventos

- Un *evento* es algo que ocurre en un momento del tiempo, no tiene duración.
  - Por ejemplo, *un usuario aprieta una tecla*.
  - Desde luego, nada es realmente instantáneo; simplemente, un evento es un suceso muy rápido en comparación con la escala de tiempo de una abstracción dada.
- Un evento puede preceder o seguir a otro, o ambos eventos pueden ser inconexos.
  - Dos eventos que no tienen efecto uno sobre otro (son causalmente inconexos) se denominan *concurrentes*.
  - Los eventos concurrentes pueden suceder en cualquier orden, por lo que al modelarlo no se intenta establecer una ordenación entre ellos. Cualquier modelo realista de un sistema distribuido incluirá eventos y actividades concurrentes.
- Un evento es una transmisión de información en un solo sentido de un objeto a otro.
  - Un objeto que envía un evento a otro puede esperar una respuesta, pero ésta es un evento distinto bajo el control del segundo objeto, que puede elegir enviarlo o no.
  - Un evento puede consistir en la recepción de una señal explícita desde otro objeto, una llamada a una operación, la verificación de una condición o el paso de un período determinado de tiempo.
  - Los eventos incluyen tanto condiciones de error como sucesos normales.

Clases de eventos

- Cada evento es un suceso único, pero se agrupan en *clases de eventos* y se da un nombre a cada una para indicar que existe una estructura y un comportamiento comunes.
  - Por ejemplo, *El vuelo 123 parte de Madrid* y *El vuelo 456 parte de Barcelona* son instancias de la clase evento *El vuelo de un avión parte de*.
- La mayoría de clases de eventos tienen *atributos* que representan la información (valores de datos) que transmiten de un objeto a otro.
  - Los atributos se muestran (opcionalmente) entre paréntesis después del nombre de la clase de evento.
  - El tiempo en que ocurre un evento es un atributo implícito de todos los eventos.

Sale el vuelo de un avión (línea aérea, número de vuelo, ciudad)
Botón del ratón pulsado (botón, localización)
String de entrada introducido (texto)
Receptor telefónico levantado
Dígito marcado (dígito)
La velocidad de un motor entra en zona peligrosa

Estados

- Un *estado* es una abstracción de los valores de los atributos y los enlaces de un objeto.
  - El estado de un banco es solvente o insolvente, en función de si sus activos exceden a sus pasivos o no.
- Un estado especifica la respuesta del objeto a eventos de entrada. Esta respuesta puede incluir una acción o un cambio de estado por parte del objeto.
  - Si se marca un dígito en el estado *Señal de línea libre*, desaparece esta señal y la línea de teléfono entra en el estado *Llamando*; si se cuelga el receptor en el estado *Señal de línea libre*, la línea telefónica se corta y pasa al estado *Inactivo*.
- Un estado corresponde al intervalo entre dos eventos recibidos por un objeto. Los eventos representan puntos en el tiempo y los estados, intervalos de tiempo.
  - Después de descolgar el auricular y antes de marcar el primer número, la línea de teléfono se encuentra en el estado *Línea libre.*

Estados

- El estado de un objeto depende de la secuencia de eventos anterior que ha recibido, pero en muchos casos los eventos previos son eclipsados por los eventos siguientes.
  - Los eventos que ocurrieron antes de colgar el teléfono no tienen efecto en el comportamiento futuro; el estado *Inactivo* olvida los eventos previos al evento *colgar*.
- Un estado se suele asociar con una actividad continua o con una actividad que tarda un tiempo en completarse.
  - Existe una dualidad entre eventos y estados; un evento separa dos estados y un estado separa dos eventos.
- Un estado también suele asociarse con el valor de un objeto que satisface alguna condición.
  - *El agua es líquida* equivale a decir que la temperatura del agua es mayor que 0°C y menor que 100°C.
  - Cada valor enumerado de un atributo define un estado distinto. Las marchas en un coche pueden estar en los estados *marcha atrás*, *punto muerto*, *primera*, *segunda*, *tercera*, *cuarta* o *quinta*.
- Al definir estados, se ignoran aquellos atributos que no afectan al comportamiento del objeto y se agrupan en un estado simple todas las combinaciones de valores de atributos y de enlaces que tienen la misma respuesta a los eventos.

Estados

- Tanto los eventos como los estados dependen del nivel de abstracción considerado.
  - Por ejemplo, excepto para mandar 0s y 1s, los números exactos marcados no afectan al control de la línea telefónica, así que se pueden agrupar dentro del estado *Llamando* y dejar el número del teléfono como un parámetro.
- Un estado puede caracterizarse de varias formas y las diferentes descripciones de un estado pueden solaparse.

Estado: *Alarma sonando*
Descripción: la alarma del reloj suena para indicar que ha llegado la hora marcada

Secuencia de eventos que conducen al estado:
Poner alarma (hora alarma)
Cualquier secuencia que no incluya *quitar alarma*
hora actual = hora alarma

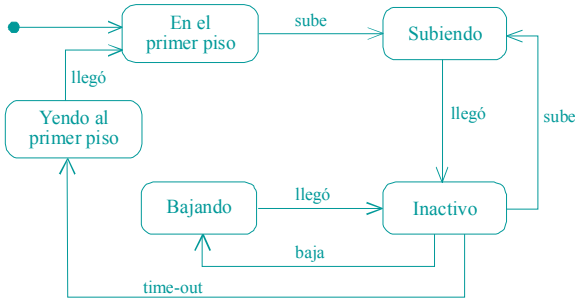
Condición que caracteriza al estado:
(alarma = on) and (hora alarma <= hora actual <= hora alarma + 20 segundos) and
(no se ha pulsado ningún botón desde hora alarma)

Eventos aceptados en el estado:

evento	**acción**	**próximo estado**
hora actual = hora alarma + 20	restablecer alarma	*normal*
botón pulsado (cualquier botón)	restablecer alarma	*normal*

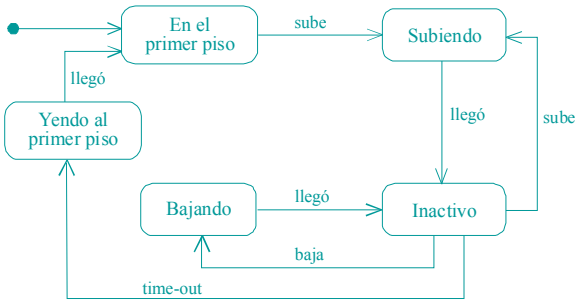
Transiciones

- Un *diagrama de estados* relaciona eventos y estados. Un cambio de estado causado por un evento se denomina una *transición*.
- Un estado se dibuja como un rectángulo de esquinas redondeadas con un nombre opcional.
- Una transición se dibuja como una flecha desde el estado origen al estado destino, etiquetada con el nombre del evento que causa la transición.
- Las transiciones que abandonan un estado deben corresponder a eventos diferentes.



Transiciones

- Si un objeto está en un estado y ocurre un evento de los que etiquetan una de sus transiciones, el objeto entra en el estado destino de la transición y se dice que la transición se ha *disparado*.
  - Si más de una transición abandona un estado, el primer evento que ocurra causa que se dispare la transición correspondiente. Si ocurre un evento que no corresponde a una transición que abandona el estado actual, se ignora el evento. Una secuencia de eventos corresponde a un camino a través del grafo.
- Un estado está activo cuando se alcanza mediante una transición y se convierte en inactivo cuando se abandona a través de una transición.



Descripción de transición

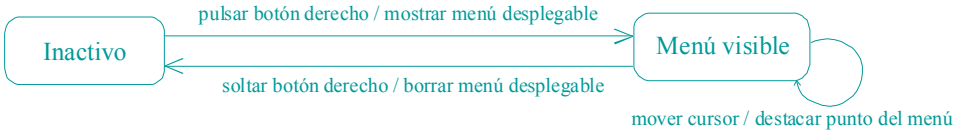
- La etiqueta asociada a la transición (*transition description*) describe las circunstancias que provocan el cambio de estado. La sintaxis UML completa es *trigger [guard] / behavior*.
  - *Trigger* es un evento que causa una transición.
  - *Guard* es una condición booleana que se evalúa cuando se produce un evento, para determinar si la transición asociada a él puede ocurrir.
  - *Behavior* es una actividad asociada al evento, que no se puede interrumpir y que se ejecuta mientras sucede la transición.

- La actividad (behavior) asociada al evento se describe usando operaciones, atributos y enlaces del clasificador al que pertenece, así como cualquier parámetro del evento de disparo. Esta actividad puede generar explícitamente eventos, como el envío de señales o la invocación de operaciones.
  - *boton_izdo_raton_pulsado(localizacion) / color := capta_color (localización); pluma.poner (color)*

Descripción de transición

- La combinación de condiciones de guarda y eventos de disparo permite modelar distintos tipos de cambios de estado.

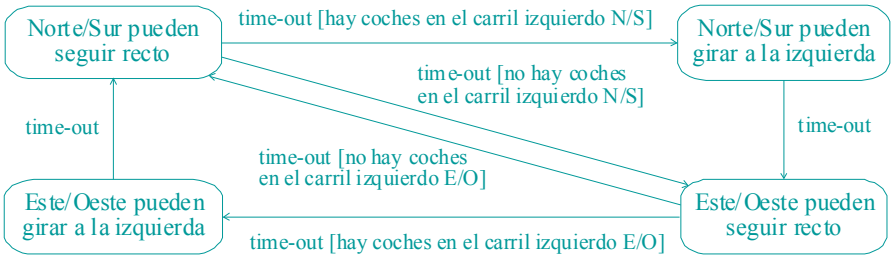
- Si se especifica un evento sin condición de guarda, la transición ocurre cuando se produce el evento asociado a la transición.



Descripción de transición

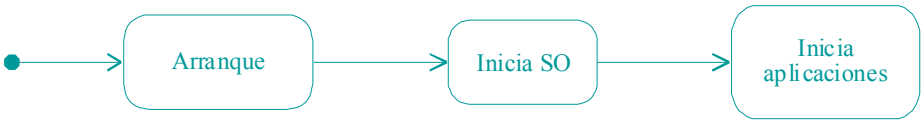
- Una transición con condición de guarda se dispara cuando ocurre su evento correspondiente, pero sólo si la condición es cierta.

- También se pueden usar condiciones de guarda para modelar una elección entre transiciones.



Descripción de transición

- Si no se especifican evento ni condición, la transición ocurre inmediatamente después de que finalice el comportamiento interno (si existe) del estado origen de la transición.
  - Si no hay actividad asociada al estado origen, la transición sin etiqueta se dispara tan pronto como se alcanza el estado.
- Si un estado tiene una o más transiciones sin evento asociado, pero ninguna de sus condiciones de guarda se satisfacen, el estado permanece activo hasta que se verifica alguna de las condiciones o hasta que un evento causa el disparo de otra transición.
  - El cambio en el valor de una condición es un evento implícito.



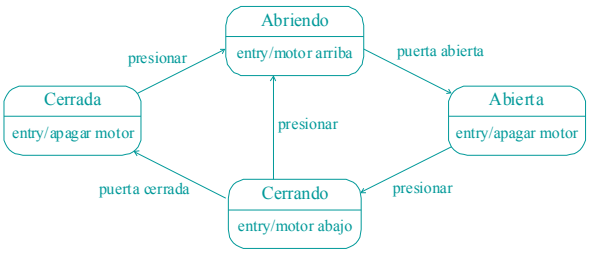
Descripción de estado

- Los diagramas de estado serían de poca utilidad si sólo describieran patrones de eventos. La descripción del comportamiento de un objeto debe especificar lo que hace el objeto como respuesta a los eventos.

- Con los estados se asocian *actividades* que tardan un tiempo en completarse.
  - Las actividades incluyen operaciones continuas, así como operaciones secuenciales que terminan por sí mismas después de un intervalo de tiempo.
  - La notación *do/ A* dentro del rectángulo de estado indica que la actividad *A* comienza al entrar en el estado y se interrumpe al salir.
  - Si un evento causa una transición desde el estado antes de que la actividad se complete, entonces la actividad acaba prematuramente.
    - Por ejemplo, un robot podría encontrar resistencia en su avance, lo que causaría su parada.

Descripción de estado

- Junto con la actividad que tiene lugar mientras el estado está activo, un estado puede describir comportamiento que se ejecuta al entrar (*entry*) o salir (*exit*) del estado o cuando ocurre un evento mientras el objeto se encuentra en ese estado, pero sin provocar cambio de estado en el objeto.
  - Un comportamiento de entrada (entry) se representa dentro del rectángulo del estado con la notación *entry/ comportamiento*.
  - Siempre que se alcance el estado mediante una transición, se realiza el comportamiento de entrada. Un comportamiento de entrada equivale a asociarlo con cada transición que conduce al estado. Si una transición de entrada ya tiene una actividad asociada, ésta se realiza primero.
  - Un comportamiento de salida (exit) se representa dentro del rectángulo del estado con la notación *exit/ comportamiento*.
  - Siempre que se abandona un estado por medio de cualquier transición de salida, se ejecuta primero el comportamiento de salida (exit).



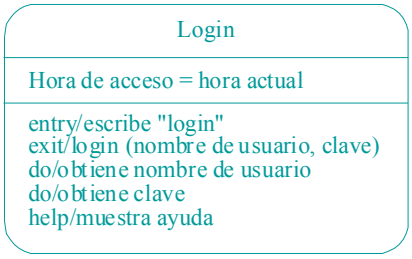
Descripción de estado

- Una transición *interna* es una transición que origina la ejecución de un comportamiento sin causar un cambio de estado.
  - La transición interna se describe dentro del rectángulo del estado con la notación *trigger [guard]/behavior*.
  - Cuando ocurre una transición interna, se ejecuta su comportamiento pero no los de entrada o de salida del estado.
  - Hay que diferenciar entre un comportamiento causado por una transición interna y una auto-transición; esta última causa la ejecución de los comportamientos de salida y entrada del estado.

- Las actividades *do* pueden ser interrumpidas por eventos que causan transiciones fuera del estado, pero los comportamientos de entrada y salida se completan a pesar de todo, ya que se consideran instantáneos.
  - Si se interrumpe una actividad *do*, el comportamiento de salida se ejecuta de todos modos.

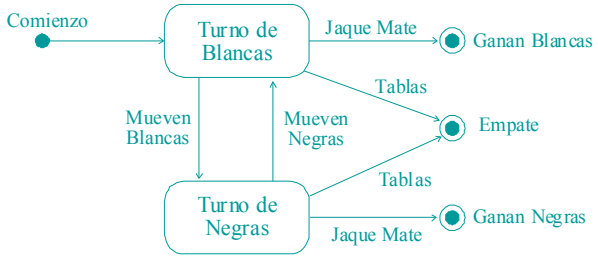
Descripción de estado

- Si se especifican múltiples comportamientos en un estado, se realizan en el siguiente orden:
  - Actividades de la transición entrante.
  - Comportamientos de entrada (*entry*).
  - Actividades *do*.
  - Comportamientos de salida (*exit*).
  - Actividades sobre las transiciones salientes.
- Un estado también puede contener variables de estado que corresponden a los atributos de la clase asociada al diagrama de estados.



Pseudoestados inicial y final

- Con los diagramas de estado se pueden representar secuencias finitas de estados, con un pseudoestado inicial y un pseudoestado final, o también bucles sin fin, donde no se sabe o no importa cómo se inicia el bucle.
- Los diagramas finitos representan objetos con vidas finitas.
  - Un pseudoestado inicial se alcanza al crear el objeto, y alcanzar el pseudoestado final supone la destrucción del objeto.
  - Un pseudoestado inicial se representa con un círculo sólido, que puede etiquetarse para indicar diferentes condiciones iniciales.
  - Un pseudoestado final se representa con una diana, que puede etiquetarse para distinguir condiciones finales.

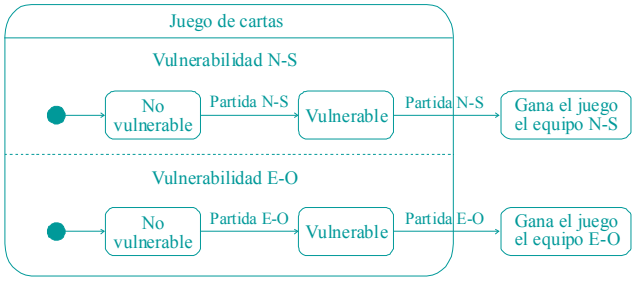


Estados compuestos

- La concurrencia dentro del estado de un objeto surge cuando el objeto puede dividirse en subconjuntos de atributos o enlaces, cada uno de ellos con su propio subdiagrama.
  - Los subdiagramas se ejecutan en paralelo y el estado del objeto comprende un estado de cada subdiagrama.
  - No es necesario que los diagramas sean independientes; el mismo evento puede causar transiciones en más de un subdiagrama.
- La concurrencia dentro de un *estado compuesto* de un objeto se representa dividiendo el estado compuesto en *regiones* separadas por una línea punteada.
  - Cada subdiagrama pertenece a una región.
  - Un estado de una región es un subestado del estado compuesto.
  - El nombre del estado compuesto completo se puede escribir en una zona del recuadro separada por una línea de los subdiagramas concurrentes.

Estados compuestos

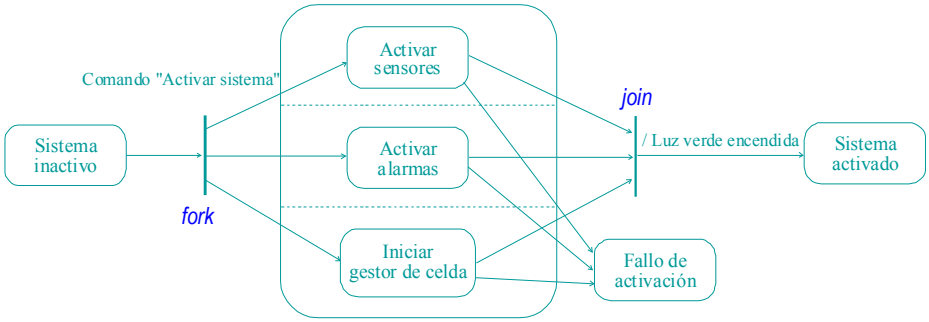
- Cuando se activa un estado compuesto, se activa el pseudoestado inicial de cada región y comienza la ejecución del subdiagrama correspondiente.
  - Los subdiagramas se interrumpen si se produce un evento que provoque el abandono del estado compuesto.
  - Si los subdiagramas pueden ejecutarse sin interrupción, la actividad del estado compuesto finaliza cuando se ha completado cada subdiagrama.
  - Las transiciones entre regiones del mismo estado compuesto no están permitidas.



Juego de cartas con estados concurrentes

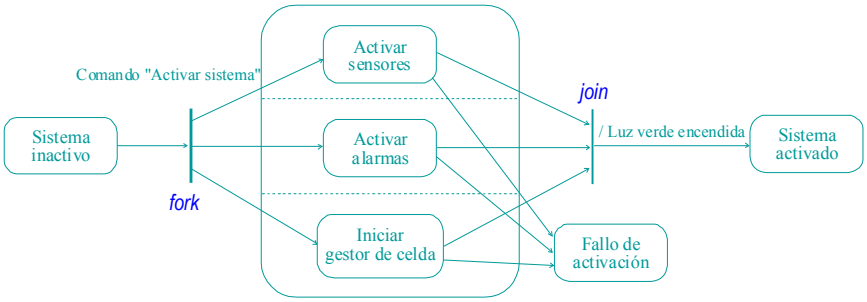
Sincronización

- Para mostrar explícitamente la sincronización de dos (o más) actividades en concurrencia se usan los pseudoestados *fork* (*división de control*) y *join* (*fusión de control*).
  - El pseudoestado de *división de control* divide la transición de entrada al estado compuesto en dos o más transiciones de salida y el pseudoestado de *fusión de control* reúne sus transiciones de entrada en una transición de salida.



Sincronización

- Con esta notación se indica que, aunque los pasos internos de las actividades concurrentes no están sincronizados, estas actividades han de completarse antes de que el objeto pueda avanzar a su próximo estado.
  - Todos los subdiagramas deben alcanzar el pseudoestado de fusión antes de que el estado compuesto termine.
  - Si hay algún subdiagrama en el estado compuesto que no está conectado al pseudoestado de fusión, termina automáticamente cuando se alcanza el pseudoestado de fusión y se abandona el estado compuesto.
  - Los comportamientos de salida (exit), si los hay, de todos los subdiagramas se realizan cuando se alcanza el pseudoestado de fusión.

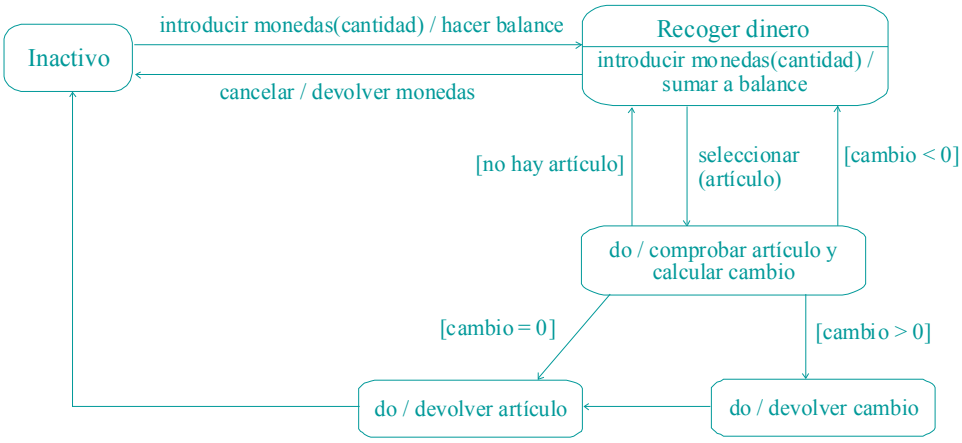


Diagramas de estado anidados

- Los diagramas de estado pueden estructurarse para facilitar una descripción concisa de sistemas complejos.
  - Un estado puede tener anidados subestados y transiciones que componen otro diagrama de estados, de modo que es posible reutilizarlo en otro punto del diagrama de alto nivel.
  - Además, los estados y las transiciones pueden ordenarse en jerarquías de generalización con herencia de estructura y comportamiento comunes, de manera similar a la herencia de atributos y operaciones en clases.

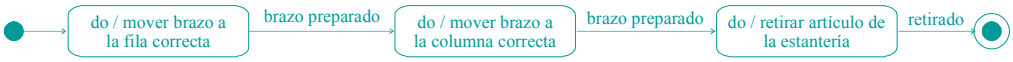
- Una actividad en un estado puede expandirse como un diagrama de estado de más bajo nivel con una transición de entrada y una de salida. En este diagrama, cada estado representa una etapa de la actividad.

Diagramas de estado anidados

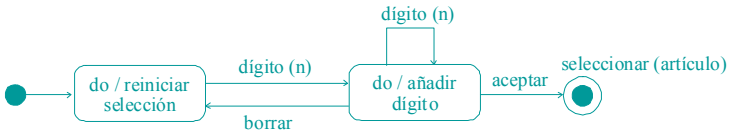


Modelo para una máquina expendedora

Diagramas de estado anidados



Actividad *devolver artículo* de la máquina expendedora



Transición *seleccionar artículo* de la máquina expendedora

Generalización de estados

- Un diagrama de estados anidado es una forma de generalización (relación *or*) sobre los estados.
  - Un objeto en un estado del diagrama de alto nivel debe estar exactamente en un estado del diagrama anidado; debe estar en el primer estado *o* en el segundo *o* en alguno de los otros estados.
  - Todos los estados del diagrama anidado son refinamientos del estado del diagrama de alto nivel. En general, los estados del diagrama anidado pueden interaccionar con otros estados del diagrama de alto nivel.

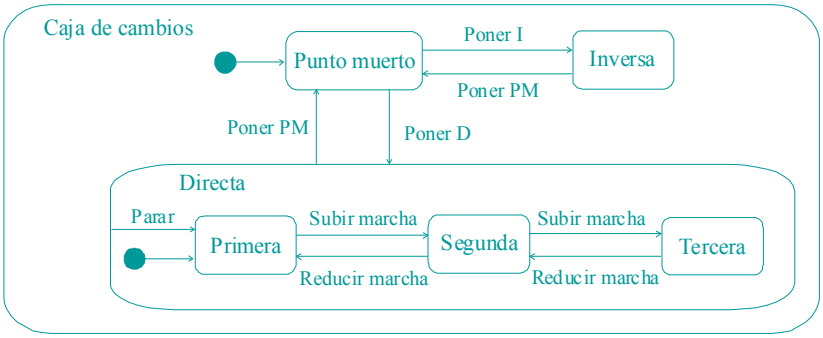


Diagrama de estados con generalización de la caja de cambios de un coche

Generalización de estados

- Los estados pueden tener subestados que heredan las transiciones de sus superestados.
  - Cualquier transición que alcanza a un estado se aplica a todos sus subestados, a menos que sea redefinida por una transición equivalente en el subestado.
  - Las transiciones desde un superestado son heredadas por cada uno de sus subestados.

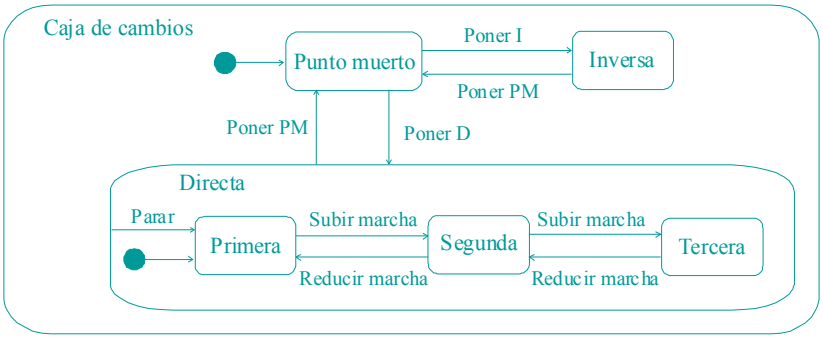


Diagrama de estados con generalización de la caja de cambios de un coche

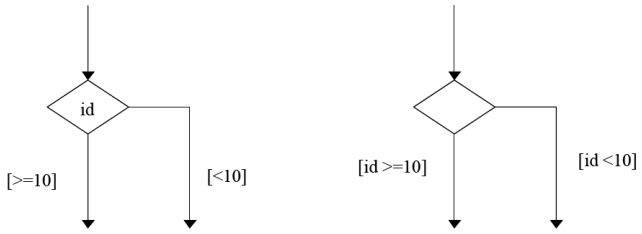
Generalización de estados

- El tránsito hacia dentro o hacia fuera de un subestado puede causar la ejecución de varios comportamientos de entrada o de salida, si la transición atraviesa varios niveles de generalización.
  - Los comportamientos de entrada (entry) se ejecutan de fuera hacia dentro y los de salida (exit) de dentro hacia fuera.

- Es posible representar situaciones más complicadas, tales como una transición explícita desde un subestado a un estado externo o una transición explícita hacia el interior del superestado.
  - En casos más simples donde no hay interacción salvo para inicio y terminación, los estados anidados pueden dibujarse simplemente como un diagrama separado y referenciarse por un nombre en una declaración *do*, como en el ejemplo de la máquina expendedora.

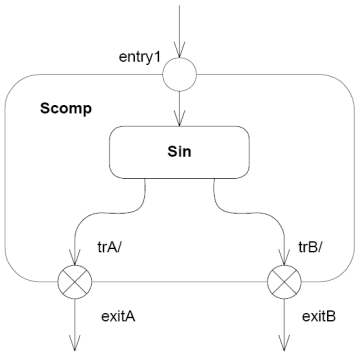
Pseudoestados adicionales

- Un pseudoestado es un tipo especial de estado que permite representar cambios de estado complejos al combinarse con transiciones básicas.
- Además de los pseudoestados *inicial*, *final*, *fusión de control (join)* y *división de control (fork)*, otros pseudoestados adicionales son:
  - *Elección (choice)*: Se usa para modelar una elección entre transiciones de salida en función del valor de una condición de guarda asociada a ellas. Al menos una de las condiciones debe ser cierta para que el modelo sea correcto. Este pseudoestado se representa con el símbolo de un rombo.



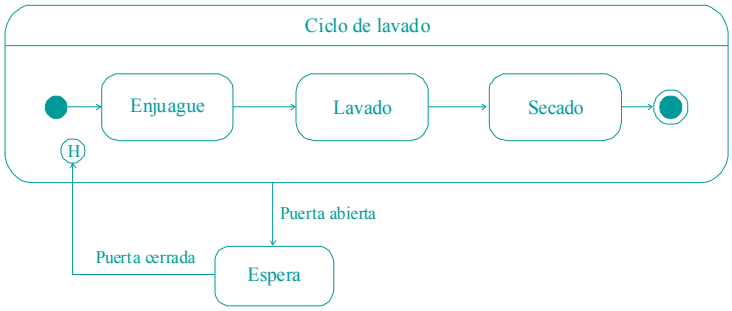
Pseudoestados adicionales

- *Punto de entrada (entry point)*: Representa un punto de entrada a un estado compuesto. Desde el punto de entrada se puede alcanzar un subestado diferente del inicial por defecto. Se muestra con un círculo dibujado en el borde del estado compuesto y se le puede asociar un nombre que describa su significado.
- *Punto de salida (exit point)*: Representa un punto de salida de un estado compuesto. Cuando se alcanza este pseudoestado, se abandona el estado compuesto y se dispara la transición de salida que tenga conectada en el diagrama de alto nivel. Se muestra mediante un círculo con una cruz dibujado en el borde del estado compuesto, también se le puede asociar un nombre que describa su significado.



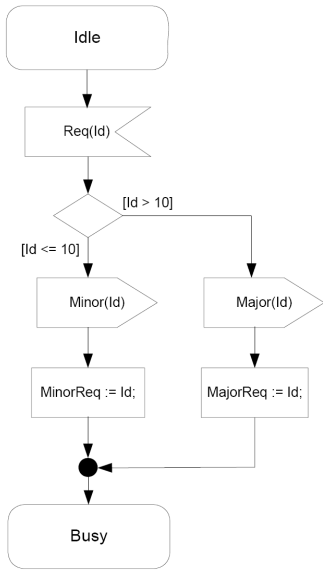
Pseudoestados adicionales

- *Indicador de historia superficial H (shallow history)*: Se usa dentro de una región en un estado compuesto para memorizar el último subestado visitado y recuperarlo en una transición entrante en el superestado que lo engloba. El subestado debe estar en el mismo nivel que el pseudoestado H. Se representa con un círculo que contiene una H.
- *Indicador de historia profunda H* (deep history)*: Se usa igual que el pseudoestado H, salvo que no importa la profundidad del subestado dentro de la región. Se representa con un círculo que contiene una H*.



Señales (signals)

- Las transiciones pueden representarse con más detalle usando iconos explícitos para el envío y la recepción de señales y para la actividad asociada a la transición.
  - Una transición que recibe una señal se representa dividiéndola en dos transiciones con un pentágono cóncavo entre ellas. La descripción de la señal se muestra dentro del símbolo de recibir señal.
  - Una transición que envía una señal se representa dividiéndola en dos transiciones con un pentágono convexo entre ellas. La descripción de la señal se muestra dentro del símbolo de enviar señal.
  - El resultado de la transición se muestra con un rectángulo dentro del que se describe la actividad asociada. El rectángulo se conecta con su origen y su destino mediante el símbolo de transición.

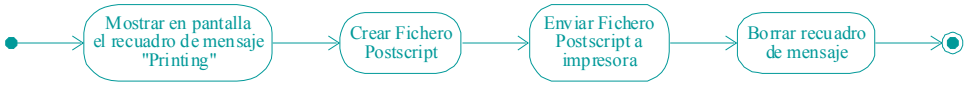


Relación entre los diagramas de estados y de clases

- El diagrama de estados de una clase es heredado por sus subclases.
  - Las subclases heredan tanto los estados del ancestro como las transiciones.
- Las subclases pueden tener sus propios diagramas de estado.
- Normalmente, el diagrama de estado de una subclase será una adición independiente, ortogonal y concurrente al diagrama de estado heredado de la superclase, definido sobre un conjunto diferente de atributos (normalmente, los añadidos en la subclase).
  - Si los diagramas de estado de la superclase y de la subclase tratan con conjuntos disjuntos de atributos, la subclase tiene un estado compuesto formado por diagramas de estado concurrentes.

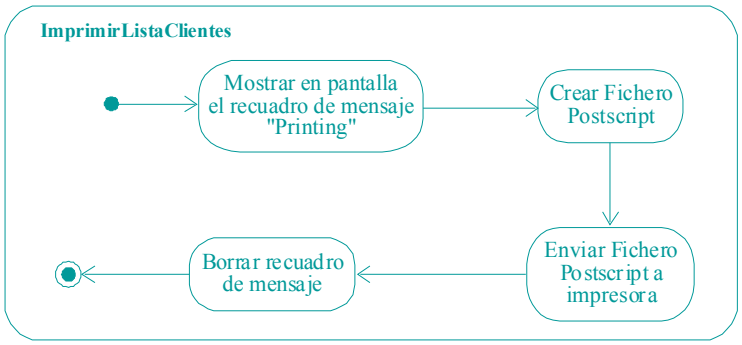
Diagrama de actividad

- El diagrama de actividad muestra la coordinación de tareas de alto nivel para realizar un proceso determinado dentro del sistema.
  - Se usa para describir el funcionamiento de un clasificador, como un caso de uso o un objeto.
- El diagrama de actividad captura *actividades* compuestas de *acciones*.
  - Una *acción* representa un paso simple dentro de la actividad, en el que se realiza un procesado de datos, y se dibuja como un rectángulo de esquinas redondeadas. Dentro del símbolo de acción, se coloca su nombre o un texto que la describe.
  - El flujo de control a través de la actividad se muestra conectando las acciones con flechas que apuntan a la próxima acción.
  - Se puede asociar un nombre y una condición de guarda a la flecha de transición entre acciones, aunque normalmente no se especifica nada, indicando que cuando se completa la ejecución de una acción, el control pasa a la siguiente.



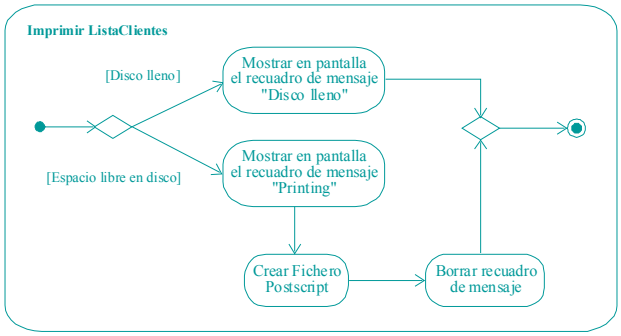
Nodos de control

- Opcionalmente, el diagrama de actividad puede encerrarse en un marco de actividad (activity frame, también un rectángulo de esquinas redondeadas), donde el nombre de la actividad se muestra en la esquina superior izquierda.
- Un diagrama de actividad normalmente tiene un nodo inicial y un nodo final.
  - El nodo inicial se muestra con un círculo sólido y el nodo final se muestra con un círculo que rodea un círculo sólido más pequeño (una diana). La actividad termina cuando se alcanza el nodo final.



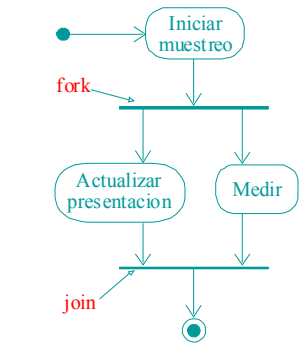
Nodos de control

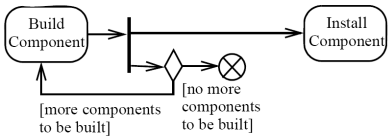
- La ejecución de secuencias de acciones diferentes en función del valor de una condición se indica con nodos de *decisión* (decision).
  - El símbolo de decisión (un rombo) tiene una transición de entrada y dos o más de salida. Las transiciones de salida se etiquetan con condiciones de guarda.
  - En un modelo bien diseñado, sólo una de las condiciones de salida será cierta, para evitar situaciones de carrera indeseadas.
- Los flujos de control alternativos se unen en un nodo de *reunión* (merge), para marcar el final del comportamiento condicional.
  - El nodo de reunión (rombo) tiene dos o más transiciones de entrada y una de salida.



Nodos de control

- Las acciones pueden realizarse en paralelo, lo que se muestra dividiendo el flujo de ejecución en varias líneas, cada una de las cuales va a una de las acciones.
  - Para representarlo se usan los nodos de *división* (fork) y *fusión* (join) de control.
  - El nodo de división divide el flujo de control de entrada en múltiples flujos concurrentes de salida.
  - El nodo de fusión sincroniza múltiples flujos de entrada en un flujo de salida, de modo que todas las acciones paralelas han de completarse antes de continuar la ejecución posterior al nodo de fusión.
  - Para mostrar el final de un flujo de control sin finalizar la actividad completa, se usa el nodo *final de flujo* (flow final), que se representa con un círculo que rodea una X. Indica que uno de los flujos paralelos termina, pero el resto puede continuar su ejecución.



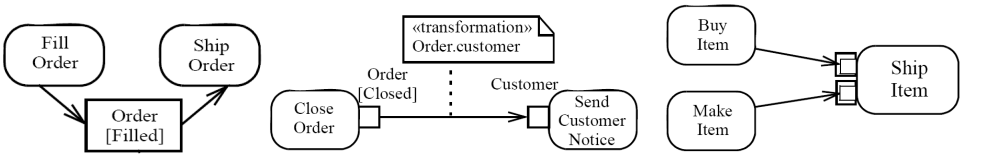


Objetos

- En los diagramas de actividad se pueden usar nodos objeto para representar flujo de datos a través de una actividad. Un nodo objeto muestra que las acciones lo pueden usar, crear o modificar.
  - El nodo objeto se dibuja con un rectángulo con el nombre del objeto dentro.
  - Cuando un objeto es entrada de una acción, se muestra con una flecha desde el objeto a la acción; cuando el objeto es salida de una acción, se muestra con una flecha desde la acción hasta el objeto.
- Una representación alternativa de que un objeto es entrada o salida de una acción utiliza pines (pins).
  - Un *pin* se muestra con un pequeño rectángulo sobre el borde de la acción de la que es entrada o salida. Si se trata de un pin de entrada, las flechas dirigidas hacia la acción apuntan al pin; si es un pin de salida las flechas que abandonan la acción parten del pin.

Objetos

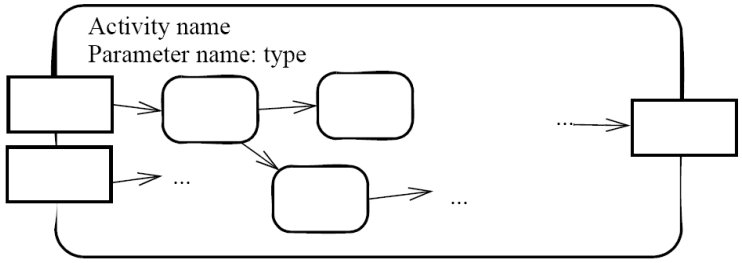
- Se puede representar el cambio de estado del objeto cuando fluye a través de la actividad, mostrando su estado entre corchetes dentro del nodo objeto.
- Si la acción sólo necesita parte del objeto, se usa una transformación (transformation) para mostrar cómo la salida de una acción proporciona la entrada de otra.
- Para mostrar conjuntos de entradas o salidas alternativas en una acción, se usa la notación *conjuntos de parámetros* (parameter sets).
  - Se dibuja un rectángulo rodeando los pines incluidos dentro del conjunto.
  - Con esta notación, en una ejecución dada, la acción puede usar los pines de entrada de uno de los conjuntos y generar salidas en uno de los conjuntos de pines de salida.



Using parameter sets to express "or" invocation

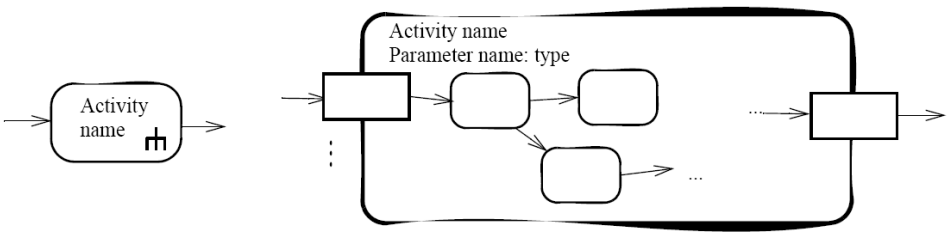
Objetos

- Los nodos objeto también pueden ser entradas y salidas que requiere la actividad completa.
  - Para representarlo se dibuja el nodo objeto sobre el borde del marco de actividad.
  - Si es un objeto de entrada, una flecha se dirige desde el objeto a la primera acción; si es un objeto de salida, una flecha se dirige desde la última acción hacia el objeto.



Invocar otras actividades

- Para reducir la complejidad del diagrama de actividad, se pueden mostrar los detalles de una acción en un diagrama separado.
  - Un *nodo de invocación* de una actividad (call activity node) se asocia con la actividad dándoles el mismo nombre. Cuando se alcanza este nodo, se ejecuta el diagrama de actividad asociado.



Señales (signals)

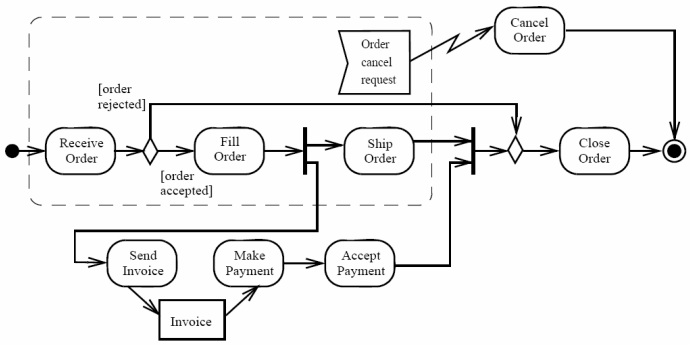
- En los diagramas de actividad puede mostrarse el envío y la recepción de señales (signals) o mensajes, representando interacciones con participantes externos.
  - La recepción de una señal desde un proceso externo activa una acción en el diagrama de actividad.
  - El envío de una señal a un participante externo producirá algún efecto en el receptor, pero esto no aparece en el diagrama de actividad. Las señales se envían de forma asíncrona, es decir, la actividad no espera respuesta y avanza a la próxima acción después del envío de la señal.
  - El envío de una señal se representa con un pentágono convexo y la recepción, con un pentágono cóncavo. La combinación de los símbolos de envío y recepción de una señal equivale a una llamada síncrona, en la que se espera una respuesta.

Eventos temporales

- Para modelar un evento temporal, como un periodo de espera, se usa el símbolo de un reloj de arena, con un texto asociado que indica el tiempo de espera.
  - Si no se dibuja transición de entrada al símbolo, se modela un evento recurrente que se activa con la frecuencia indicada en el texto asociado.

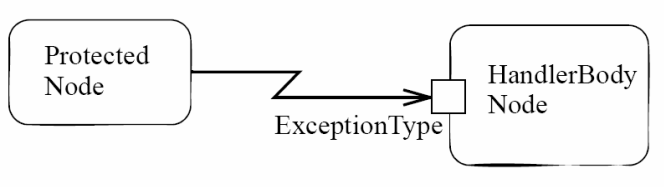
Interrupciones

- Para modelar que un proceso puede ser interrumpido por un evento se usan *regiones de interrupción* (interruption regions).
  - Una región de interrupción se representa con un rectángulo de esquinas redondeadas y línea punteada, que encierra las acciones que se pueden interrumpir junto con el evento que causa la interrupción.
  - El evento de interrupción (símbolo de envío de señal) se conecta a una línea quebrada de salida, que abandona la región de interrupción y apunta a la acción que continúa la ejecución.



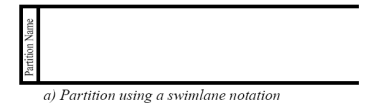
Manejo de excepciones

- Cuando se produce una excepción (condición de error) durante la ejecución de una acción, termina la acción y se ejecuta el manejador de excepción (exception handler) si existe.
- La salida del manejador de excepción pasa a la próxima acción después del nodo protegido, como si éste hubiera terminado su ejecución normalmente.
  - El manejador de excepción se representa usando el símbolo de acción con un pequeño recuadro en el borde.
  - Desde el nodo protegido hasta el recuadro se dibuja una línea quebrada, que se etiqueta con el tipo de la excepción captada.

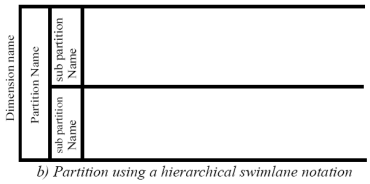


Particiones

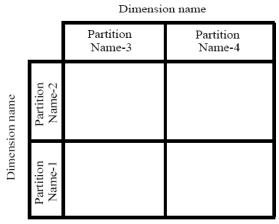
- En las actividades pueden intervenir múltiples participantes que son responsables de distintas acciones. Se usan *particiones* (partitions) para agrupar las acciones respecto a su grupo de responsabilidad.
  - Las particiones se muestran con dos líneas paralelas que dividen el diagrama en columnas o filas (dependiendo de la orientación del diagrama de actividad) que se denominan pasillos (swimlanes).
  - El nombre de la partición se coloca en un extremo y los nodos que se ejecutan en la partición se dibujan entre las dos líneas.
  - Pueden mostrarse particiones multidimensionales, mediante la intersección de particiones verticales y horizontales.



a) Partition using a swimlane notation



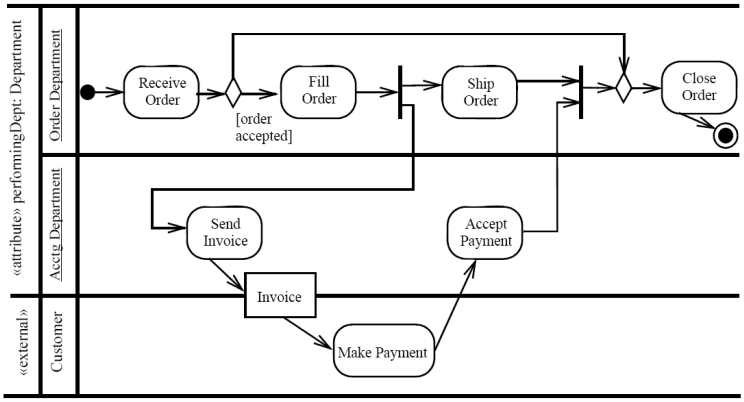
b) Partition using a hierarchical swimlane notation



c) Partition using a multidimensional hierarchical swimlane notation

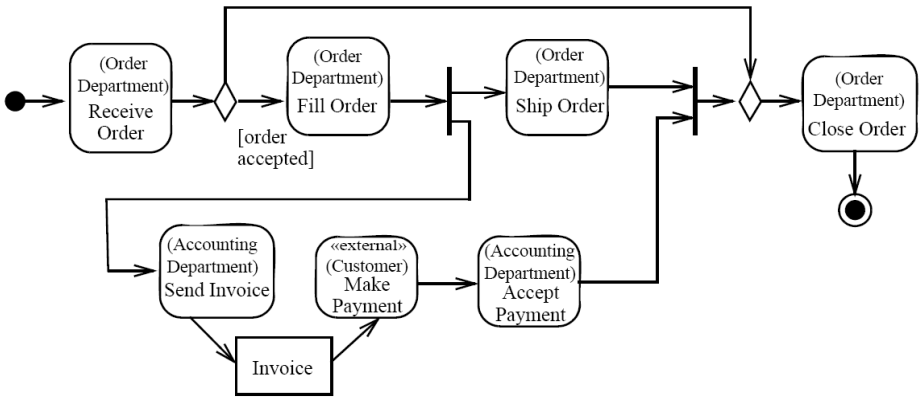
Particiones

- Si las acciones de una partición ocurren fuera del alcance del modelo, se etiqueta la partición como <<external>>.
- Para indicar que una partición representa una clase, se indica <<class>> antes de su nombre, y se suele subrayar el nombre si la partición representa una instancia.
- Si la partición representa un atributo con un valor específico, se indica <<attribute>> seguido de su nombre encima de los nombres de las particiones a las que se aplica su valor.



Particiones

- También se puede mostrar la responsabilidad usando anotaciones, es decir, incluyendo en el nodo el nombre del participante responsable entre paréntesis.
  - Aunque esta notación hace más compacto el diagrama, se muestran los participantes con menor claridad.



Regiones de expansión

- Una región de expansión (expansion region) muestra que las acciones que encierra se realizan sobre una colección de elementos de entrada.
  - Se representa con un rectángulo de esquinas redondeadas y línea punteada, que rodea las acciones, y cuatro cuadrados alineados sobre el borde de dos lados del rectángulo, representando las colecciones de entrada y salida.
  - Se dibuja una flecha apuntando desde los cuadrados de entrada a la primera acción dentro de la región y otra flecha apuntando desde la última acción a los cuadrados de salida.
  - Se pueden usar las etiquetas <<parallel>>, <<iterative>> o <<streaming>> en la esquina superior izquierda para indicar una ejecución concurrente, secuencial o continua, respectivamente.

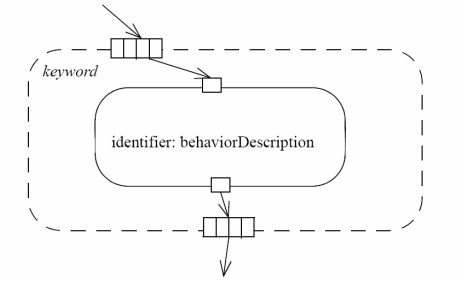


Diagrama de secuencia

- El *diagrama de secuencia* es un tipo de diagrama de interacción que muestra el orden de las interacciones entre las partes del sistema.
- Un diagrama de secuencia se compone de una colección de *participantes*, es decir, las partes del sistema (normalmente, objetos) que interactúan entre sí durante la secuencia, mediante el envío de mensajes.
  - Un participante se representa con un rectángulo y una línea punteada vertical llamada línea de vida del participante, que indica su ejecución durante la secuencia.
  - El nombre del participante se escribe dentro del rectángulo con el formato name[selector]: class_name ref decomposition, con el significado siguiente:
    - *Name*: nombre de la instancia.
    - *Selector*: es opcional e indica una instancia en particular dentro de una estructura con varios elementos (como un array, por ejemplo).
    - *Class_name*: nombre de la clase del participante.
    - *Decomposition*: es opcional y se usa para hacer referencia a otro diagrama de interacción que muestra los detalles del procesado del mensaje que recibe este participante.

Diagrama de secuencia

- En un diagrama de secuencia, ocurre una interacción cuando un participante decide enviar un mensaje a otro participante.
  - Los mensajes se representan con flechas horizontales, orientadas desde el emisor del mensaje hacia el destinatario.
  - Un mensaje es una comunicación entre participantes que transporta información con la intención de que se realice una acción.
  - El envío y la recepción de un mensaje se consideran normalmente eventos. Los mensajes pueden ser señales, llamadas a operación o algo similar, por ejemplo, RPC (Remote Procedure Calls) en C++.

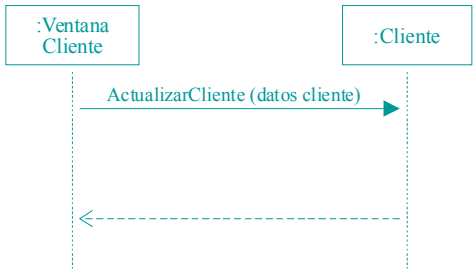


Diagrama de secuencia

- El orden de envío de los mensajes viene dado por la posición sobre el eje vertical.
  - El diagrama de secuencia se lee de arriba hacia abajo mostrando el orden del intercambio de mensajes que tiene lugar a lo largo del tiempo.
  - La descripción o signatura del mensaje se escribe sobre la flecha y tiene el formato attribute = signal_or_message_name (arguments) : return_type, con el siguiente significado:
    - *Attribute*: es opcional, indica que el valor devuelto desde el mensaje se almacena en el atributo especificado.
    - *Signal_or_message_name*: es el nombre del mensaje enviado.
    - *Arguments*: lista de los argumentos del mensaje separados por comas.
    - *Return_type*: tipo del valor devuelto desde el mensaje.

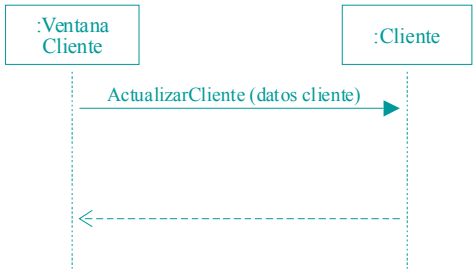


Diagrama de secuencia

- Los distintos tipos de mensaje se representan con distintos símbolos para las flechas horizontales:
  - *Mensaje síncrono*: Sólo desencadena una operación cuando el destinatario acepta el mensaje, y el emisor del mensaje se bloquea hasta ese momento.
  - *Mensaje asíncrono*: Representa un envío de mensaje donde no hay retorno explícito y no se interrumpe la ejecución del emisor.
  - *Mensaje de retorno*: Es una representación opcional para mostrar explícitamente que el flujo de control vuelve al emisor original del mensaje.
- La flecha que simboliza un mensaje puede representarse oblicua para materializar las demoras de transmisión no despreciables respecto a la dinámica general de la aplicación.

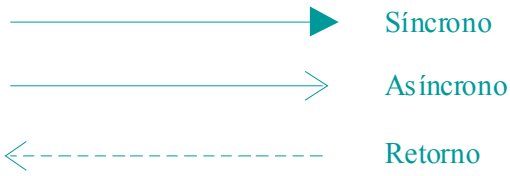
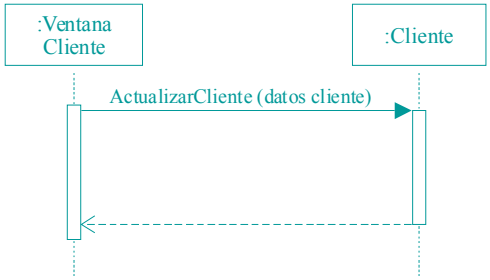


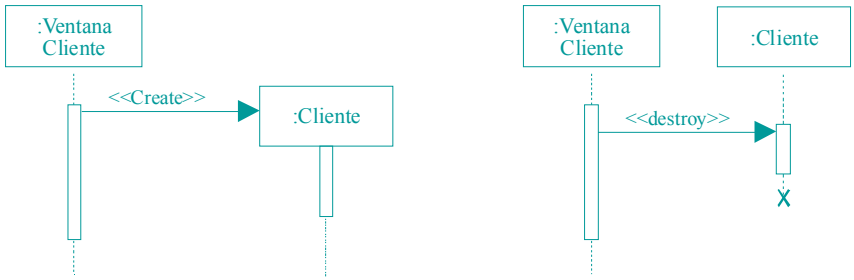
Diagrama de secuencia

- Los diagramas de secuencia también permiten representar los períodos de actividad de los participantes.
  - Cuando se recibe un mensaje, comienza una actividad en el receptor; esto se llama *activación*. La activación muestra qué participantes están en ejecución en algún punto del tiempo.
  - Un participante activado está, bien ejecutando su propio código, o bien esperando el retorno desde otro participante al cual ha enviado un mensaje.
  - La activación se muestra como una banda rectangular sobre la línea de vida del participante. El inicio y el fin de una banda corresponden al inicio y al fin de un período de actividad, respectivamente.



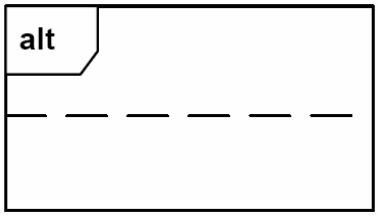
Mensajes de creación y destrucción

- Los diagramas de secuencia pueden mostrar cómo se crean y se destruyen los participantes, como parte de la interacción que se describe.
  - Para mostrar la creación de un participante se envía un mensaje síncrono <<create>> a la línea de vida del participante o se dibuja el rectángulo correspondiente en el punto donde se ha creado sobre el eje vertical de tiempo.
  - Cuando un participante se destruye, se marca con una X mayúscula y se termina su línea de vida en el punto en que se destruyó. La marca de destrucción puede ir precedida de un mensaje síncrono <<destroy>> enviado al participante.



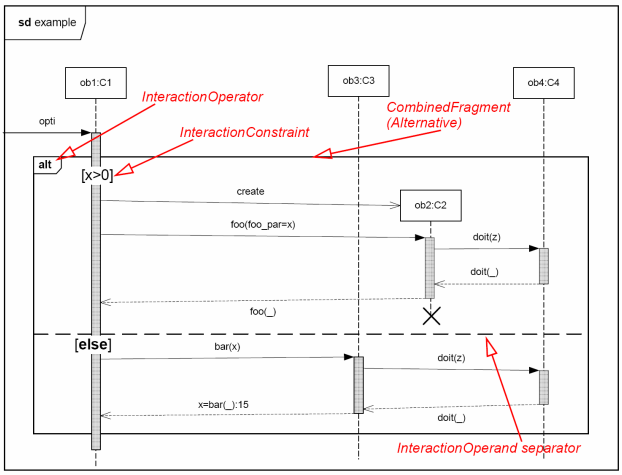
Fragmento combinado

- Un fragmento combinado es un contenedor donde se colocan fragmentos o porciones de una interacción. Permite describir de forma compacta y concisa interacciones complejas como lazos o alternativas.
  - Se representa con un rectángulo que encierra los fragmentos de interacción y se define mediante un operador de interacción y los correspondientes operandos.
  - El operador se indica dentro de un pentágono en la esquina superior izquierda del rectángulo del fragmento combinado.
  - Los operandos se separan con una línea punteada horizontal dentro del rectángulo.
  - No se permiten mensajes entre fragmentos de interacción y los operandos se leen de arriba hacia abajo.



Fragmento combinado

- Un fragmento de interacción puede tener una condición de guarda que indica si el fragmento puede ejecutarse o no, con el formato [boolean_expression].
  - Esta condición se muestra sobre el primer mensaje del fragmento de interacción, en la parte superior de la línea de vida asociada.



Operadores de interacción

- Algunos de los operadores de interacción en UML 2 son los siguientes:
  - *Ref*: Representa una interacción definida en otro punto del modelo. Permite reutilizar una colección de interacciones.
  - *Assert*: Especifica que la interacción que contiene el fragmento combinado debe ocurrir como está indicado; si no, el fragmento se declara inválido y se produce una excepción.
  - *Loop*: Ejecuta un conjunto de interacciones un número de veces especificado.
  - *Break*: Cuando se ejecutan las interacciones definidas dentro de este fragmento, se abandona el fragmento de interacción que lo contiene.
  - *Alt*: Especifica que un conjunto de interacciones se ejecuta bajo ciertas condiciones.
  - *Opt*: Las interacciones definidas dentro de este fragmento se ejecutan si la condición de guarda es cierta.
  - *Neg*: Sirve para indicar que las interacciones incluidas en este fragmento no están permitidas.
  - *Par*: Las interacciones incluidas en cada operando pueden ejecutarse en paralelo, aunque manteniendo el orden del operando original.
  - *Critical*: Las interacciones son parte de una región crítica, donde se modifica un participante compartido, y se tratan como un bloque atómico.

Conceptos de tiempo y duración

- Los diagramas de secuencia pueden tener etiquetas y comentarios (texto o pseudocódigo) en el margen izquierdo o en el derecho.
- Es posible expresar restricciones de tiempo con etiquetas.
  - Podría ser útil restringir el tiempo entre dos mensajes o el tiempo que toma a un mensaje llegar a su destino (el tiempo de transición), como se muestra en la figura.

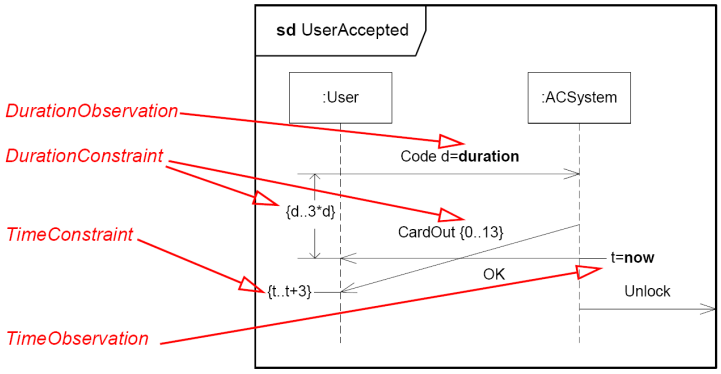
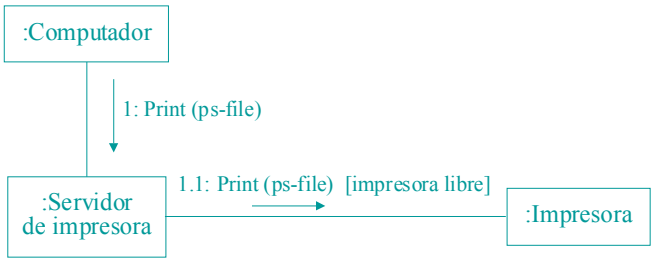


Diagrama de comunicación

- Un diagrama de comunicación muestra un contexto (un conjunto de participantes y sus relaciones) y una interacción (cómo cooperan los participantes para realizar una tarea específica).
  - Los diagramas de secuencia y de comunicación muestran interacciones, pero el diagrama de secuencia se enfoca en el orden de los mensajes, y el diagrama de comunicación se enfoca en los enlaces de comunicación entre participantes.
  - Los enlaces de comunicación sirven de soporte de transmisión para los mensajes.
  - Con cada mensaje se asocia una etiqueta que define, entre otras cosas, un número de secuencia para el mensaje.



Representación de mensajes

- El formato de una etiqueta de mensaje es el siguiente:

  sequence_number: name [recurrence_or_guard]

  - *Sequence_number*: es un número de secuencia que especifica el orden en que se invoca el mensaje.
    - El mensaje 1 siempre comienza una secuencia de mensajes; el mensaje 1.1 es el primer mensaje anidado dentro del tratamiento del mensaje 1; el mensaje 1.2 es el segundo mensaje anidado dentro del tratamiento del mensaje 1. Así, la numeración indica tanto la secuencia como el anidamiento de mensajes.
    - Para representar un mensaje concurrente se usa una letra. Por ejemplo, 1.2a y 1.2b son mensajes concurrentes enviados en paralelo.
  - *Name*: es el nombre del mensaje y su lista de argumentos.
  - *[recurrence_or_guard]*: es opcional y permite especificar una ejecución iterativa o condicional.
    - Si se especifica una condición, ésta debe ser cierta para que se envíe el mensaje.
    - La iteración se indica con un rango de valores precedido con un asterisco (*).
- Ejemplos de etiquetas de mensajes son:

  1: display() 1.1a: continue() 3.1: foo() [x<0] 3.2: bar () [x>=0]

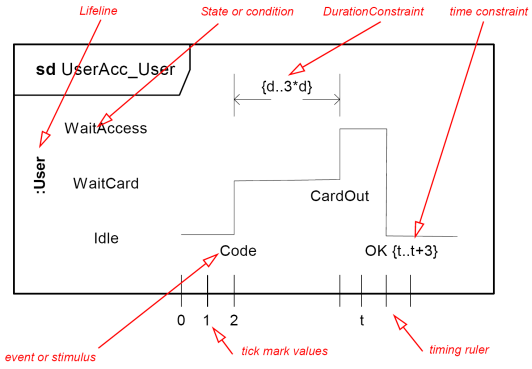
  2: nextPrim(prim) *[n = 1..z]

Diagramas de comunicación vs. Diagramas de secuencia

- Los diagramas de comunicación muestran los participantes en la interacción y sus enlaces (la red de participantes que están colaborando), lo que en muchas situaciones puede facilitar la comprensión de la interacción.
- En el diagrama de secuencia el orden de envío de mensajes o flujo de mensajes es más fácil de ver, puede leerse de arriba hacia abajo.
- Se elige un diagrama de comunicación cuando mostrar los participantes y sus enlaces facilita comprender la interacción.
- Se elige un diagrama de secuencia cuando sólo se necesita mostrar la secuencia ordenada de la interacción.

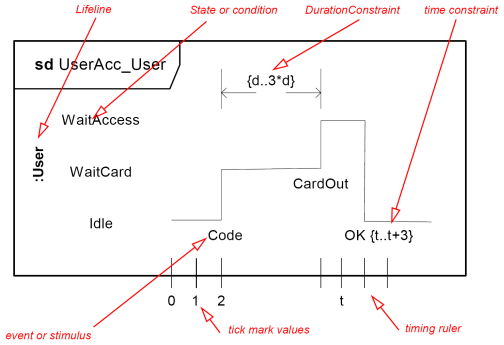
Diagramas de tiempo

- El diagrama de tiempo (Timing diagram) es una representación especial de interacción enfocada en especificar los tiempos de envío de mensajes entre los participantes en la interacción.
  - Su principal propósito es mostrar el cambio de estado de un participante a lo largo del tiempo, en respuesta a los eventos aceptados.
  - El diagrama de tiempo se lee de izquierda a derecha. El participante se muestra a la izquierda del diagrama y, a continuación, se listan sus posibles estados, seguidos por una representación gráfica de las transiciones entre los estados.



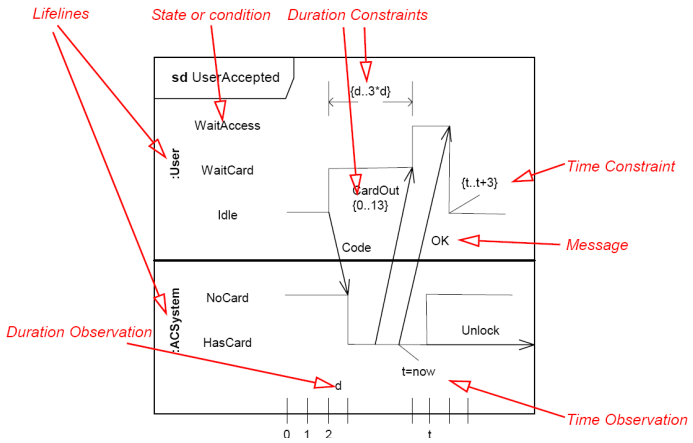
Diagramas de tiempo

- El estado de un participante en un tiempo dado se indica con una línea horizontal llamada *línea de estado* (state-line).
  - Se puede usar la longitud de la línea de estado para indicar cuánto tiempo permanece el participante en ese estado.
  - Para asociar medidas de tiempo, se muestran marcas (tick marks) en la parte inferior del marco del diagrama.
  - Para representar tiempo relativo, se marca un instante concreto con una variable cuyo nombre puede usarse en una condición para indicar, por ejemplo, que un mensaje debe recibirse dentro de un intervalo de tiempo determinado.



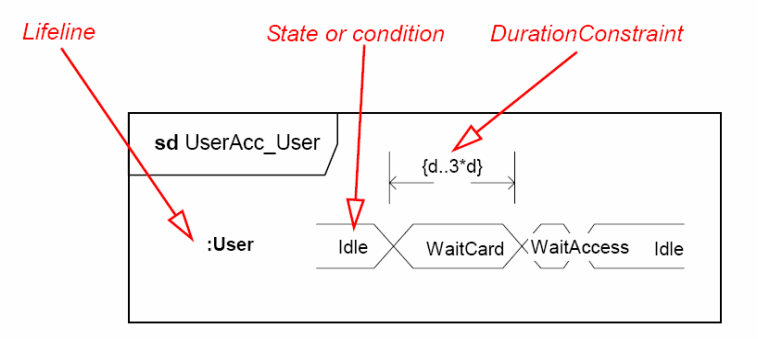
Diagramas de tiempo

- Se pueden representar las líneas de estado de varios participantes en el diagrama, apilándolas en vertical.
  - Los mensajes entre participantes se muestran con una flecha desde la línea de estado del emisor hasta la línea de estado del receptor, con lo que se representa el disparo que origina una transición de estado.



Diagramas de tiempo

- UML permite una notación alternativa que simplifica el diagrama, mostrando los nombres de los estados entre dos líneas horizontales que se cruzan cuando cambia el estado.
  - Con esta notación, no se representan los mensajes que provocan el cambio de estado.



Diagramas de visión de conjunto

- El diagrama de visión de conjunto de la interacción (Interaction overview diagram) da una representación de alto nivel del flujo de control entre las diversas interacciones que implementan una especificación del sistema, tal como un caso de uso.
  - Se trata de una variante de diagrama de actividad, donde los nodos son interacciones completas que se pueden representar con su propio diagrama (de secuencia, de comunicación o de tiempo).
  - Los fragmentos alternativos se representan usando nodos de decisión (decision) y de reunión (merge) como los del diagrama de actividad.
  - Los fragmentos paralelos se representan con nodos de división de control (fork) y de fusión de control (join).
  - Los lazos de ejecución se muestran como los ciclos en el diagrama de actividad.
  - Los nombres de los participantes en las interacciones se muestran a continuación del nombre del diagrama con el formato lifelines lista de participantes.