INSTITUTO KHIPU

Semestre III

Sesión 26

PROGRAMA DE ESTUDIOS

DESARROLLO DE SISTEMAS DE INFORMACIÓN

ANÁLISIS Y DISEÑO DE SISTEMAS

Tema:

DIAGRAMAS DE COLABORACIÓN Y ESTADOS - ESTADO Y COMPORTAMIENTO DE UN SISTEMA.

DIAGRAMAS DE COLABORACIÓN Y ESTADOS - ESTADO Y COMPORTAMIENTO DE UN SISTEMA.

Como ya hemos mencionado, el objetivo de los diagramas de estado es describir el comportamiento de un sistema con la máxima precisión. Entre otras cosas, esta **representación gráfica de los procesos** debería dar respuesta a las siguientes preguntas:

- ¿Qué sucede cuando el objeto está en un estado concreto?
- ¿En qué estado debe estar el objeto para cambiar de comportamiento?
- ¿Cuáles son los desencadenantes?
- ¿Qué propiedades debe tener el objeto para poder cambiar de estado?

Por lo tanto, los diagramas de estado UML se utilizan para optimizar cualquier proceso de desarrollo donde sea útil visualizar los estados del objeto y las condiciones para que se produzca la transición de un estado a otro. Suelen emplearse, por ejemplo, en el **diseño de sistemas embebidos** (en inglés, *embedded systems*), donde las señales automatizadas y los procesos en segundo plano deben estar perfectamente coordinados. En este caso, el diagrama de estado ayuda a los desarrolladores a **visualizar** todas las **funciones de control y regulación** más importantes en un solo esquema.

La **función de interrupción de la toma de agua** que tienen casi todas las **lavadoras** puede servirnos de ejemplo para imaginar un diagrama de estado UML. En este contexto, esta función se representaría como un sistema independiente. En este caso, el esquema mostraría en qué estado y bajo qué condiciones se activa la función.

Nota

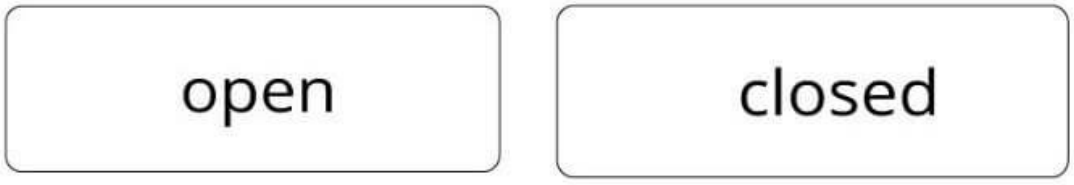
En varios sectores de la industria, como el transporte o la tecnología sanitaria, los diagramas de estado se utilizan para **explicar procesos complejos**. También se emplean en la ingeniería de requisitos y en la gestión de productos y proyectos.

Diagrama de estado: estructura y componentes

Aunque los diagramas de estado UML se basan solo en unos pocos elementos, combinarlos de manera inteligente nos permite representar fácilmente secuencias de estado complejas. ¿Cuáles son los **componentes principales** y cuál es la **estructura básica** de un diagrama de estado?

Estados

Los estados son el componente principal de un diagrama de estado. Cada **estado real** se muestra siempre en un rectángulo de esquinas redondeadas. Por ejemplo, una puerta puede tener dos valores de estado:



Los dos posibles estados de una puerta: puede estar abierta o cerrada, pero no ambas cosas al mismo tiempo.

Asimismo, en el diagrama de estado de la puerta se indicaría que siempre debe cumplirse la siguiente condición:

- El objeto siempre se encuentra en uno de los dos estados: la puerta está abierta o cerrada, pero nunca abierta y cerrada al mismo tiempo.

En los diagramas de estado más complejos, el rectángulo puede dividirse en hasta tres zonas donde se muestran especificaciones de comportamiento (ver transición).

Transición: ¿cómo se cambia de estado?

Para pasar de un estado al siguiente, se debe **desencadenar un evento** que provoque una transición. Esta **transición de estado** comunica los estados entre sí y se representa mediante una flecha. Puede haber condiciones para que se desencadene dicha transición. En términos generales, los diagramas de estado

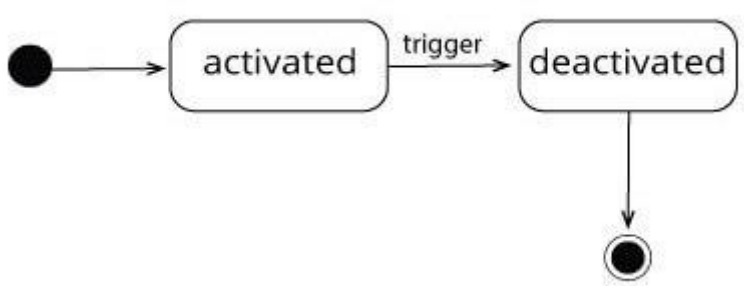
UML representan **transiciones internas y externas**. Un diagrama de estado siempre debe presentar alguna transición externa, pero no es obligatorio que incluya transiciones internas.

En el diagrama de estado de un ascensor, por ejemplo, se podría especificar la siguiente condición para la acción “cerrar la puerta del ascensor”: que el ascensor haya estado abierto al menos cinco segundos antes de que el estado cambie de “abierto” a “cerrado”.

Transición externa: cambio de estado

La transición que figura en el siguiente ejemplo se considera externa y tiene como resultado que el **objeto cambie de estado** (*entry*/*exit*).

Ejemplo: después de que se active la alarma de una radio-despertador, **el estado cambia** de “alarma activada” a “alarma desactivada”.



Cuando se activa la alarma, el objeto cambia de estado: si hace un momento la alarma estaba activada, ahora está desactivada.

Transición interna: estado inalterado

Una transición interna no desencadena un cambio de estado, sino una **actividad**.

Ejemplo: algunos **sistemas de contabilidad** vuelven a enviar las facturas sin pagar automáticamente al cliente (transición externa). Si lo que **envían** es un **recordatorio** de que la factura está pendiente de abonar, este representa una transición interna: es decir, aunque hay una actividad (“enviar el recordatorio”), la factura permanece en el mismo estado (“no pagada”) hasta nuevo aviso.

Eventos: ¿por qué se cambia de estado?

Mediante los eventos es posible describir con más detalle **las condiciones bajo las cuales se abandona un estado** para pasar al siguiente. En el caso de la transición del estado inicial al primer estado real, no es necesario, pero se puede añadir más información de manera opcional. Si no se indica ningún evento, significa que el evento ocurre automáticamente tan pronto como se hayan finalizado todas las actividades en los estados anteriores.

Si no se indica el desencadenante, significa que este evento **siempre está teniendo lugar**. Los eventos pueden representarse como una especificación de comportamiento dentro del estado o dentro de la transición hacia otro estado (ver transición).

Un evento desencadenante debe cumplir las siguientes tres **condiciones**:

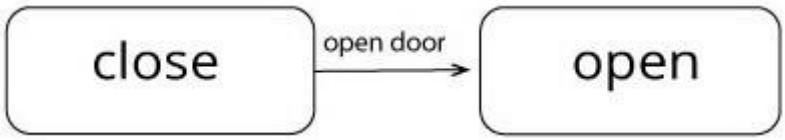
- ***entry***: el evento se activa automáticamente cuando se desencadena un estado, es decir, en todas las transiciones entrantes.
- ***exit***: el evento se desencadena cuando se abandona un estado, es decir, en todas las transiciones salientes.
- ***do***: el evento se desencadena una y otra vez si no se cambia de estado.

Estas indicaciones pueden anotarse dentro del propio estado para simplificar la representación del comportamiento bajo el cual se cambia de estado. Hay dos opciones para mostrar estos desencadenantes de manera gráfica. Una de ellas es indicarlos dentro del recuadro de estado correspondiente, como ilustra el siguiente ejemplo de diagrama de estado:

El estado de una puerta es “cerrada”. Para entrar en este estado, primero debe tener lugar el evento

"cerrar la puerta" (entry). Cuando se abandona el estado, se produce el evento "abrir la puerta" (exit). Durante el estado, "la puerta está (permanentemente) cerrada" (do).

Los eventos también se pueden indicar mediante una flecha de transición:



En los diagramas de estado sencillos, los eventos se anotan sobre la flecha de transición.

Pseudoestados

En los diagramas de estado UML, si algún elemento de control influye en el funcionamiento de una máquina de estados, pero no tiene asignado ningún valor, se denomina pseudoestado. En UML 2, la versión actual del lenguaje de modelado unificado, se definen los siguientes diez pseudoestados:

- **Estado inicial** (en inglés, *initial*): sin transición entrante y con una transición saliente que revela cuál es el estado al principio de la secuencia.
- **Estado final** (en inglés, *final*): sin transición saliente; fin de la secuencia de comportamiento.
- **Bifurcación** (en inglés, *fork*): división en varios estados paralelos.
- **Sincronización** (en inglés, *join*): sincronización de varios estados paralelos.
- **Unión** (en inglés, *junction*): nodo de unión de varias transiciones en serie.
- **Elección** (en inglés, *choice*): nodo desde el cual pueden iniciarse diversas transiciones sobre la base de una decisión previa.
- **Punto de entrada** (en inglés, *entry point*): síntesis de transiciones similares que entran en un estado compuesto.
- **Punto de salida** (en inglés, *exit point*): síntesis de transiciones similares que se originan en un estado compuesto.
- **Historial superficial** (en inglés, *shallow history*): almacenamiento del último subestado activo de un estado compuesto.

- **Historial profundo** (en inglés, *deep history*): almacenamiento del último subestado activo de todos los niveles jerárquicos de un estado compuesto.

Diagramas complejos

Dependiendo de la **complejidad** del proceso, es posible incluir subestados en el esquema que muestran una **imagen detallada de cada estado del objeto** y de su posible comportamiento. Esta versión más compleja de los diagramas de estado UML suele ser la más habitual, especialmente en el ámbito técnico.

- **Estado compuesto**: esta estructura permite **definir** un estado **en profundidad**.

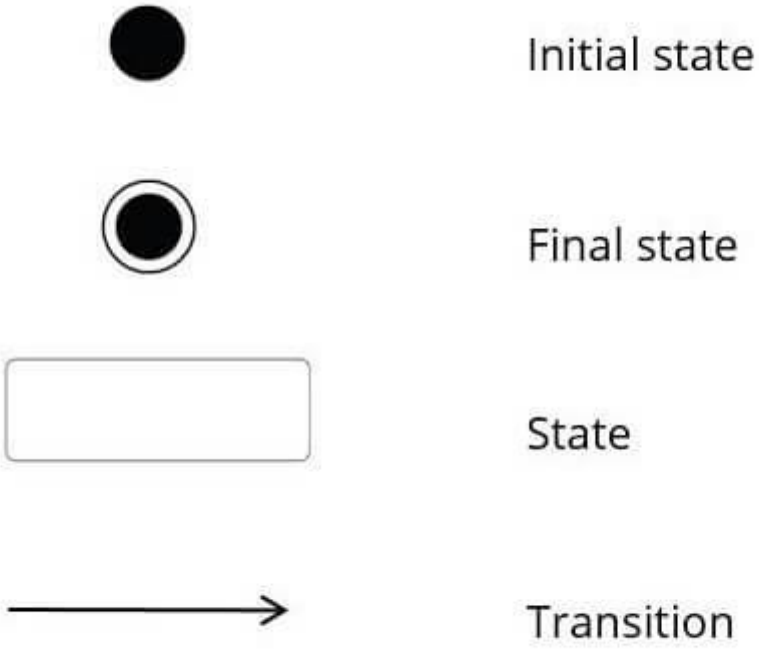
Ejemplo: una puerta puede estar en dos estados: “abierta” o “cerrada”. Los subestados del estado “cerrada” podrían ser “bloqueada” y “desbloqueada”.

- **Estado de submáquina**: el estado incluye un diagrama de estado **subordinado**. Un subestado que consista en varios subestados se denomina un estado complejo. Los diversos subestados pueden tanto ejecutarse **independientemente el uno del otro** como estar relacionados entre sí.

Ejemplo: la función de despertador de un smartphone es una de las muchas funciones que puede estar relacionada con otros estados. Si se programan distintas alarmas para diferentes horas y días de la semana, el proceso completo consistirá en subestados que se ejecutan de forma independiente.

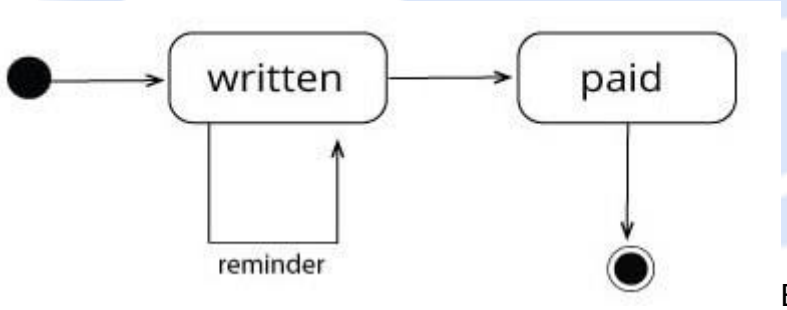
Crear un diagrama de estado: ejemplo de un diagrama simple

Los diagramas de estado pueden aplicarse a objetos de todo tipo. En el siguiente ejemplo, te mostramos cómo incluir cada elemento en el diagrama de una factura. Estos son los elementos más importantes de un diagrama de estado UML:



Los elementos más importantes de un diagrama de estado UML.

Si combinamos los elementos para nuestro ejemplo, un diagrama sencillo podría tener este aspecto:



En este ejemplo, el diagrama de estado UML presenta una transición interna.

En el punto de partida, la factura se encuentra en el pseudoestado "**escrita**" (*written*). En el mejor de los casos, se producirá la **transición hacia el estado "pagada"** (*paid*). Esta transición podría describirse con más detalle indicando el evento "enviar".

Una vez que se ha pagado, la factura se encuentra en el **estado "pagada"**. Antes de llegar a este estado, puede ser necesario **enviar un "recordatorio"** (*reminder*). En este caso, como la factura no cambia de estado, aunque se ocasione una actividad, se trata de una **transición interna**. Si no se paga la factura, se enviarán hasta tres recordatorios.

FUENTE:

- https://www.uml.org/
- https://www.codingdojo.la/2023/06/16/guia-del-ciclo-de-vida-del-desarrollo-de-software/
- https://aws.amazon.com/es/what-is/sdlc/
- BURCH, John; GRUDNISKY, Gary. "Diseño de Sistemas de Información", Grupo Noriega editores.
- SENN, James A. "Análisis y diseño de sistemas de información", 2da. ed., McGraw-Hill.

INSTITUTO

KHIPU