INVESTIGACION DE DIAGRAMAS DE ESTADO

PRESENTADO POR:

LAURA EIRENE RAMOS J.

LINDA CAROLINA PEREZ P.

FREDY HERRERA HERRERA

AREA DE METODOLOGIA

PROFESOR: CARLOS A. PINEDA

TECNOLOGÍA EN SISTEMAS V

TECNOLÓGICA FITEC

BUCARAMANGA

2011

**DIAGRAMAS DE ESTADO**

**Introducción:**

A veces hay objetos cuyo comportamiento puede variar a lo largo del tiempo; cuando esto sucede, se dice que el objeto tiene estados. Existen algunos tipos de aplicaciones, como las de tiempo real, para las cuales el modelado de estado es especialmente importante.

Conforme un sistema interactúa con los usuarios y (posiblemente) con otros sistemas, los objetos que lo conforman pasan por cambios necesarios para ajustar las interacciones. Por esa razón se necesita contar con un mecanismo para cambios en el modelo. Un cambio en un sistema se da debido a que los objetos que componen dicho sistema modificaron su estado como respuesta a los sucesos y al tiempo. Un diagrama de estados también se conoce como un "motor de estado."

Los diagramas de casos de uso deberían capturar solamente cómo un actor puede usar un sistema, pero no cómo debe ser construido dicho sistema.

Las clases y las interacciones implementan los casos de uso en el sistema. Las interacciones son expresadas en diagramas de secuencia y/o colaboración. Entonces hay un enlace entre la visión funcional y la visión dinámica del sistema. Las clases utilizadas en la implementación de los casos de uso son modeladas y descritas en los diagramas de clase, en los diagramas de estado y/o actividad.

Los diagramas de estado describen gráficamente los eventos y los estados de los objetos. Los diagramas de estado son útiles, entre otras cosas, para indicar los eventos del sistema en los casos de uso.

En el diagrama de estados o dinámico tenemos que distinguir los siguientes ***elementos:***

* Las diferentes situaciones en que se puede encontrar un objeto(los estados).
* Qué cambios de estado son posibles (transiciones).
* Cuál es el hecho que los produce (acontecimiento).

El diagrama de estado se utiliza normalmente para describir objetos del dominio del usuario y se documenta por lo general en la etapa de análisis.

**COMPONENTES:**

* **Eventos:**

Un *evento* es un acontecimiento importante a tomar en cuenta para el sistema.

Es una ocurrencia que puede causar la transición de un estado a otro de un objeto. Esta ocurrencia puede ser una:

• Condición que toma el valor de verdadero (normalmente descrita como una expresión booleana). Es un Evento Cambio.

• Recepción de una señal explícita de un objeto a otro. Es un Evento Señal.

• Recepción de una llamada a una operación. Es un Evento Llamada.

• paso de cierto período de tiempo, después de entrar al estado actual, o de cierta hora y fecha concretas. Es un Evento Tiempo.

El nombre de un evento tiene alcance dentro del paquete en el cual está definido y puede ser usado en los diagramas de estado por las clases que tienen visibilidad dentro del paquete. Un evento no es local a la clase donde está declarado.

* **Acciones:**

Una acción es una operación atómica, que no se puede interrumpir por un evento y que se ejecuta hasta su finalización. Una acción puede ser:

• Una llamada a una operación (al objeto al cual pertenece el diagrama de estado o también a otro objeto visible),

• La creación o la destrucción de otro objeto,

• El envío de una señal a un objeto.

* **Actividades :**

Cuando un objeto está en un estado, generalmente está esperando a que suceda algún evento. Sin embargo, a veces, queremos modelar una actividad que se está ejecutando. Es decir, mientras un objeto está en un estado, dicho objeto realiza un trabajo que continuará hasta que sea interrumpido por un evento.

Por lo tanto, una acción contrasta con una actividad, ya que ésta última puede ser interrumpida por otros eventos.

* **Transición:**

Una transición es una relación entre dos estados, e indica que, cuando ocurre un evento, el objeto pasa del estado anterior al siguiente.

Una transición simple es una relación entre dos estados que indica que un objeto en el primer estado puede entrar al segundo estado y ejecutar ciertas operaciones, cuando un evento ocurre y si ciertas condiciones son satisfechas.

***Una transición simple*** se representa gráficamente como una línea continua dirigida desde el estado origen (so urce) hasta el estado destino (target). Puede venir acompañada por un texto con el siguiente formato:

nombre-evento ‘(’lista-argumentos‘)’ ‘[’guard-condition‘]’ ‘/’ expresión-acción

‘^’ cláusula-envío

donde nombre-evento y lista-argumentos describen el evento que da lugar a la transición y forman lo que se denomina event-signature.

Donde guard-condition es una condición (expresión booleana) adicional al evento y necesaria para que la transición ocurra.

Si la guard-condition se combina con una event-signature, entonces para que la transición se dispare tienen que suceder dos cosas: debe ocurrir el evento y la condición booleana debe ser verdadera.

Donde expresión-acción es una expresión procedimental que se ejecuta cuando se dispara la transición.

Es posible tener una o varias expresión-acción en una transición de estado, las cuales se delimitan con el carácter “/”.

Donde cláusula-envío es una acción adicional que se ejecuta con el cambio de estado, por ejemplo, el envío de eventos a otros paquetes o clases.

***Transición interna***

Es una transición que permanece en el mismo estado, en vez de involucrar dos estados distintos. Representa un evento que no causa cambio de estado. Se denota como una cadena adicional en el compartimiento de acciones del estado.

* **Estados:**

Un estado es la condición de un objeto en un momento determinado: el tiempo que transcurre entre eventos.

Un estado identifica una condición o una situación en la vida de un objeto durante la cual satisface alguna condición, ejecuta alguna actividad o espera que suceda algún evento. Un objeto permanece en un estado durante un tiempo finito (no instantáneo).

***Estados avanzados***

Las características de los estados y de las transiciones, vistas en los apartados anteriores, resuelven un gran número de problemas a la hora de modelar un diagrama de estado. Sin embargo, hay otra característica de las máquinas de estado de UML, los subestados, que nos ayudan a simplificar el modelado de aquellos comportamientos complejos.

Un estado simple es aquel que no tiene estructura. Un estado que tiene subestados, es decir, estados anidados, se denomina estado compuesto. Un estado compuesto puede contener bien subestados secuenciales (disjuntos) o bien subestados concurrentes (ortogonales).

Se representa gráficamente por medio de un rectángulo con los bordes redondeados y con tres divisiones internas. Los tres compartimentos alojan el nombre del estado, el valor característico de los atributos del objeto en ese estado y las acciones que se realizan en ese estado, respectivamente. En muchos diagramas se omiten los dos compartimentos inferiores.

Fecha-login = fecha-actual

Entry /type “login”

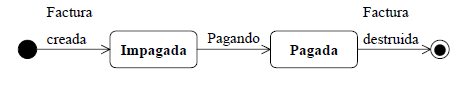
Exit/login (user name, password)

Do/get user name

Do/get password

Help/display help

**Login**



En esta figura está representado el estado Login junto con sus tres divisiones. Asimismo, los diagramas de estado tienen un punto de comienzo, el estado inicial, que se dibuja mediante un círculo sólido relleno, y un (o varios) punto de finalización, el estado final, que se dibuja por medio de un círculo conteniendo otro más pequeño y relleno (es como un ojo de toro). Dichos estados, inicial y final, aparecen marcados en la segunda figura.

***Compartimento del nombre:***

Cada estado debe tener un nombre.

En el primer ejemplo de la tenemos el nombre de estado: Login, mientras que en el segundo ejemplo hay dos nombres de estado: Impagada y Pagada.

***Compartimento de la lista de variables:***

El segundo compartimento es el compartimento de las variables de estado, donde los atributos (variables) pueden ser listados y asignados. Los atributos son aquellos de la clase visualizados por el diagrama de estado.

En el primer ejemplo tenemos la variable de estado: fecha-login, a la cual se le ha asignado el valor de la fecha del día.

***Compartimento de la lista de acciones:***

El tercer compartimento es el compartimento de las transiciones internas, donde se listan las actividades o las acciones internas ejecutadas en respuesta a los eventos recibidos mientras el objeto está en un estado, sin cambiar de estado. La sintaxis formal dentro de este compartimento es:

nombre-evento ‘(’lista-argumentos‘)’ ‘[’guard-condition‘]’ ‘/’ expresión-acción

Las siguientes acciones especiales tienen el mismo formato, pero representan palabras reservadas que no se pueden utilizar para nombres de eventos:

Entry ‘/’ expresión-acción

Exit ‘/’ expresión-acción

Las acciones Entry y exit no tienen argumentos, pues están implícitos en ellas. Cuando se entra al estado o se sale del estado, se ejecuta la acción atómica especificada en expresión-acción.

La siguiente palabra clave representa la llamada de una actividad:

do ‘/’ expresión-acción

La transición especial do nos sirve para especificar una actividad que se ejecuta mientras se está en un estado, por ejemplo, enviando un mensaje, esperando o calculando. Dicha actividad es la que aparece en expresión-acción.

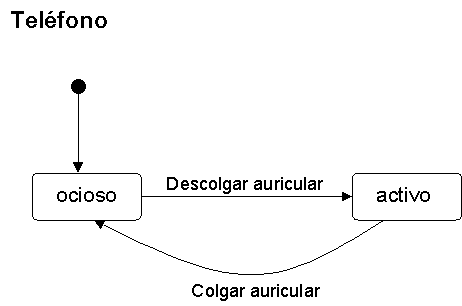
Por último, las transiciones internas, que son esencialmente interrupciones, se especifican teniendo en cuenta la sintaxis formal. Por ejemplo:

Help / display help

La transición interna heló representa un evento que no causa ningún cambio de estado, pues sólo muestra, en cualquier momento, una ayuda al usuario que viene expresada en display help.

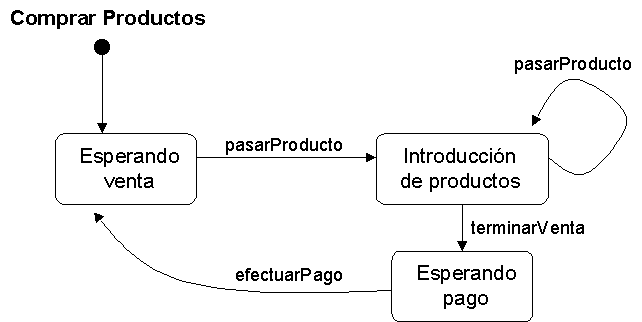
En el primer ejemplo de la Figura aparecen todos los tipos de eventos especificados anteriormente, junto con las acciones y actividades que se realizan al entrar, salir o estar en el estado Login.

En UML, los estados se representan mediante óvalos. Las transiciones se representan mediante flechas con el nombre del evento respectivo. Se acostumbra poner un estado inicial (círculo negro). Por ejemplo:

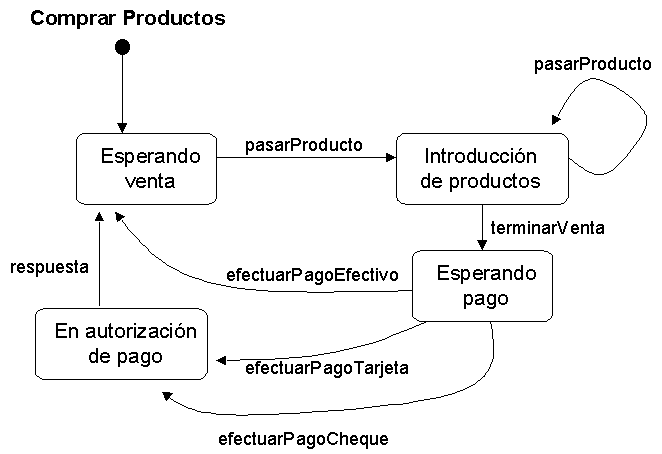


Un diagrama de estado representa el ciclo de vida de un objeto: los eventos que le ocurren, sus transiciones, y los estados que median entre estos eventos.  
  
En particular, es útil hacer diagramas de estado para describir la secuencia permitida de eventos en los casos de uso. Por ejemplo, en el caso de uso *comprar Productos* no está permitido efectuar *pago Tarjeta* mientras no haya ocurrido el evento *terminar Venta*.

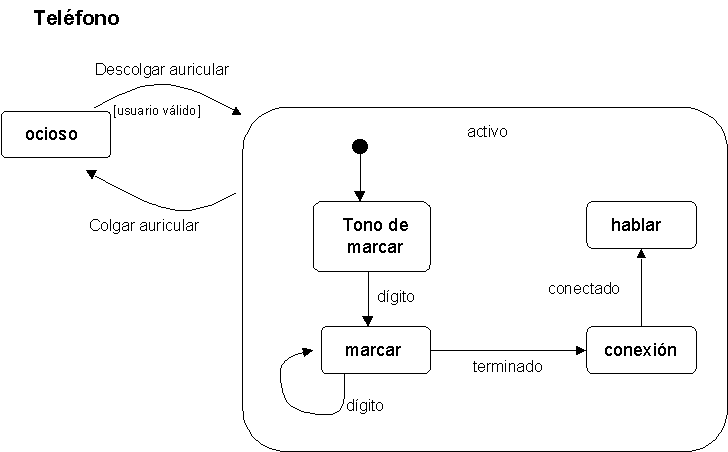
Un diagrama de estado que describe los eventos globales del sistema y su secuencia en un caso de uso es un *diagrama de estado para casos de uso*. Por ejemplo, una versión simplificada del diagrama de estados para el caso de uso *comprar Productos* es el siguiente:



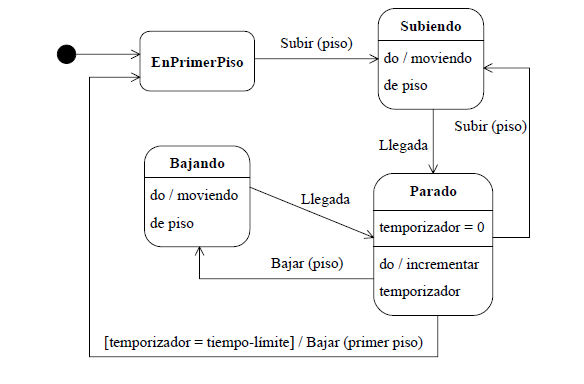
Una versión más completa del diagrama anterior se muestra en la siguiente figura:



El diagrama anterior aun no está completo, pues falta considerar algunos casos excepcionales, como por ejemplo, si al rechazar una tarjeta de crédito o un cheque, el cliente decide pagar usando otro método, por ejemplo pagando en efectivo.  
  
Una transición puede tener una *protección condicional*, o prueba booleana, que permite pasar al siguiente estado solamente si esta protección es válida. Estas protecciones se colocan entre paréntesis debajo de los eventos (ver validación del usuario al descolgar el auricular, en la siguiente figura). También se pueden tener sub-estados anidados.



En la siguiente figura tenemos un diagrama de estado para un ascensor, donde se combinan los estados con las transiciones simples.



El ascensor empieza estando en el primer piso. Puede subir o bajar. Si el ascensor está parado en un piso, ocurre un evento de tiempo rebasado después de un período de tiempo y el ascensor baja al primer piso. Este diagrama de estado no tiene un punto de finalización (estado final).

El evento de la transición entre los estados EnPrimerPiso y Subiendo tiene un argumento, piso (el tipo de este parámetro ha sido suprimido). Lo mismo sucede con los eventos de las transiciones entre Parado y Subiendo y entre Parado y Bajando.

El estado Parado (Idle state) asigna el valor cero al atributo temporizador, luego lo incrementa continuamente hasta que ocurra el evento Bajar (piso) o el evento Subir (piso) o hasta que la guard-condition [temporizador = tiempo-límite] se convierta en verdadera.

La transición de estado entre Parado y EnPrimerPiso tiene una guard-condition y una expresión-acción. Cuando el atributo temporizador es equivalente a la constante tiempo-límite, se ejecuta la acción Bajar (primer piso) y el estado del ascensor cambia de Parado a En Primer Piso.

Esta transición de estado

[Temporizador = tiempo-límite] / Bajar (primer piso)

se puede convertir en una clausula-envío tal como:

[Temporizador = tiempo-límite] ^ Self.Bajar (primer piso)

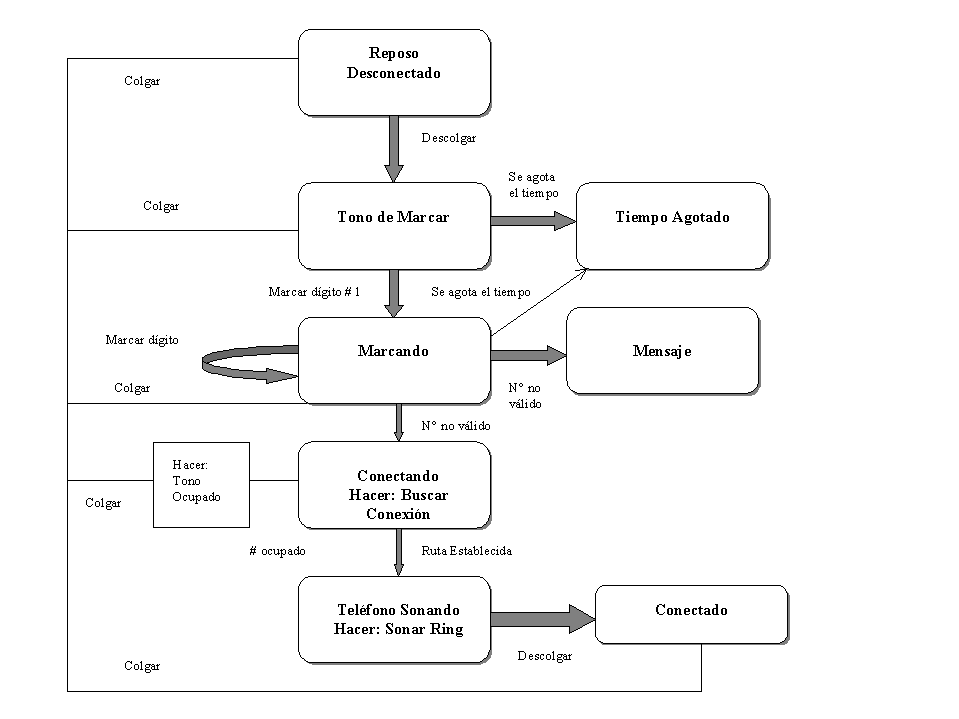
donde la expresión destino es, en este caso, el propio objeto que se evalúa a sí mismo, y el nombre del evento es Bajar (primer piso), evento significativo al objeto contenido en la expresión destino.

## Cuándo utilizar los diagramas de estados:

Los diagramas de estados son buenos para describir el comportamiento de un objeto a través de varios casos de uso. No son tan buenos para describir un comportamiento que involucra cierto número de objetos que colaboran entre ellos. Así pues, es útil combinar los diagramas de estados con otras técnicas. Por ejemplo, los diagramas de interacción son buenos para la descripción del comportamiento de varios objetos en un mismo caso de uso. Por su parte, los diagramas de actividades son buenos para mostrar la secuencia general de las acciones de varios objetos y casos de uso.

Hay quienes consideran que los diagramas de estado son naturales, pero muchos no los consideran así. Preste atención al modo en que los emplean quienes trabajan con ellos; podría ocurrir que su equipo no considere útiles los diagramas de estados, debido a su modo de trabajar. Esto no sería un gran problema; como siempre, deben combinarse las técnicas que sean de utilidad.

Este es otro ejemplo de diagrama de estado de una llamada:



BIBLIOGRAFIA:

Nos basamos de las siguientes fuentes para recopilar información:

* <http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r33019.PDF>
* <http://jms32.eresmas.net/tacticos/UML/UML08/UML0801.html>
* El libro Ingeniería del software; Escrito por Benet Campderrich Falgueras pagina 71.

**CONCLUSIÓN**

Un Diagrama de Estados sirve para mostrar una secuencia de estados por los que pasa ya sea un caso de uso, un objeto a lo largo de su vida, o todo el sistema. El diagrama indica los eventos que causan que un estado cambie a otro y cuáles son las respuestas y acciones que genera este.

Para utilizar diagramas de estados, no es necesario dibujar uno por cada clase del sistema. Aunque éste es el enfoque que emplean los detallistas, casi siempre resulta ser un esfuerzo inútil. Por eso se deben utilizar los diagramas de estados sólo para aquellas clases que presenten un comportamiento interesante y variado, es decir cuando su construcción le ayude a comprender lo que sucede.

Algunos objetos de interfaz de usuario (IU) y de control tienen el tipo de comportamiento que es útil describir mediante diagramas de estados en todo caso resultan beneficiosos para la comprensión de cada sistema y su simplificación.