Guías para comunicar un diseño

Diagrama de estados

por
Fernando Dodino

Versión 1.2
Marzo 2014

Índice

1 Objetivo del diagrama de estados

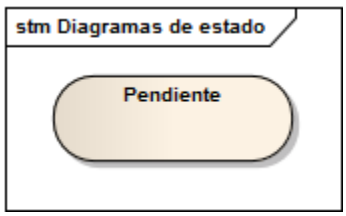
Todos los objetos tienen un ciclo de vida, en el cual alguien los crea, se conoce con otros objetos para pedirles cosas, responde a su vez a los pedidos que le hacen y cuando nadie más tiene interés en él alguien se encarga de que el objeto desaparezca del ambiente.

No todas las historias de los objetos son interesantes, pero algunos manejan estados internos en los cuales hacen o dejan de hacer ciertas cosas. Los objetos a los que nos interesa analizar su ciclo de vida los modelamos con un diagrama de estados.

1.1 Elementos

Estado inicial: es el estado de un objeto cuando se crea.

Estado final: es el estado de un objeto cuando se destruye.



Estado: representa la situación en la que se encuentra un elemento. El estado en el que se encuentra un objeto en un determinado momento es el estado *activo*.

Transición: representa el paso de un elemento de un estado origen a uno destino. Este paso puede darse:

- En forma automática
- Un evento dispara la ejecución de una acción que modifica el estado del objeto.

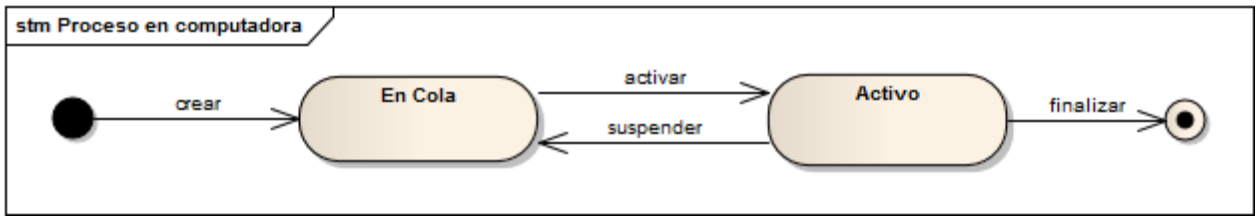
Arriba de la flecha que marca la transición se documenta el evento según el siguiente formato:

`Nombre del evento` *`(parámetros)`* *`[condición a cumplir para que se dispare el evento]`*

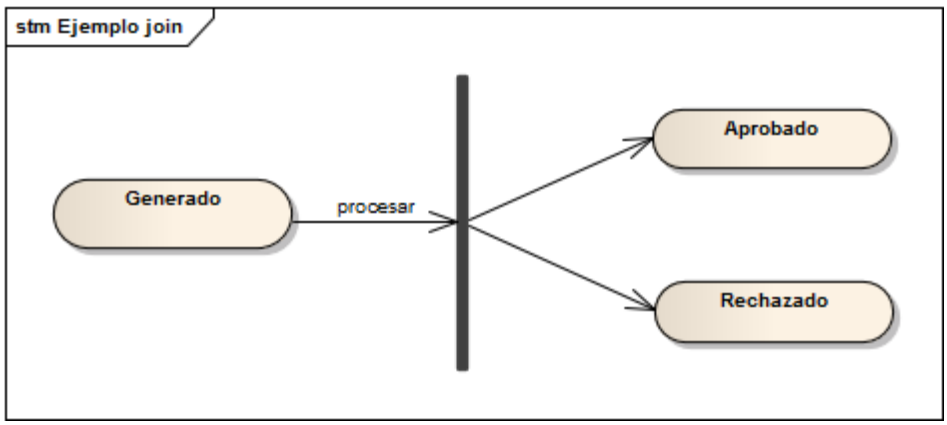
Tanto la lista de parámetros como la condición a cumplir son opcionales.

Ejemplo: Un proceso en la computadora puede estar activo (está corriendo en CPU)

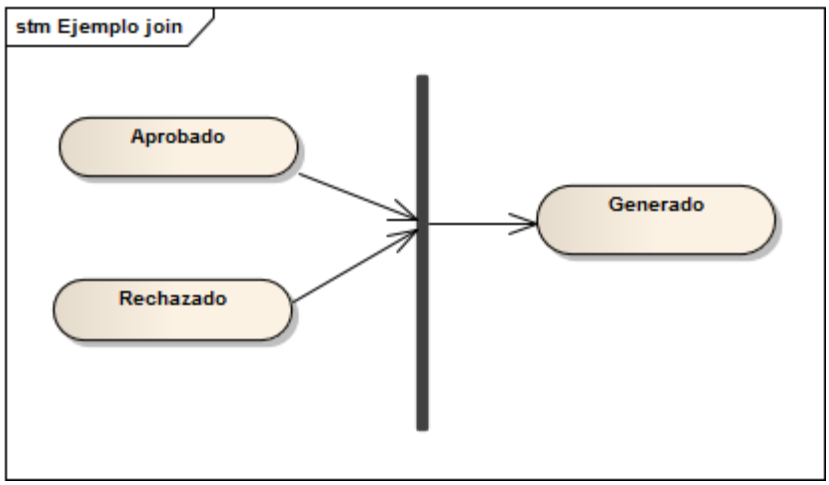
o suspendido (está en cola de procesos). Decidimos no documentar los eventos, sino directamente las acciones (siempre lo importante es documentar lo que nosotros queremos):



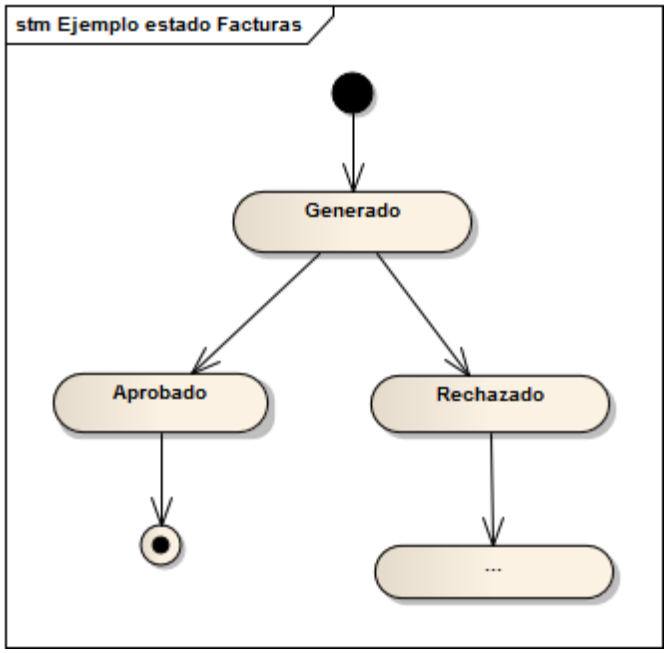
Condicionales: permite mostrar transiciones que dependen de una determinada condición para pasar de un estado a otro. *Ejemplo:* una vez generado, el pedido se procesa…



El join unifica la transición de varios estados a uno de destino, cerrando así la condición. *Ejemplo:*



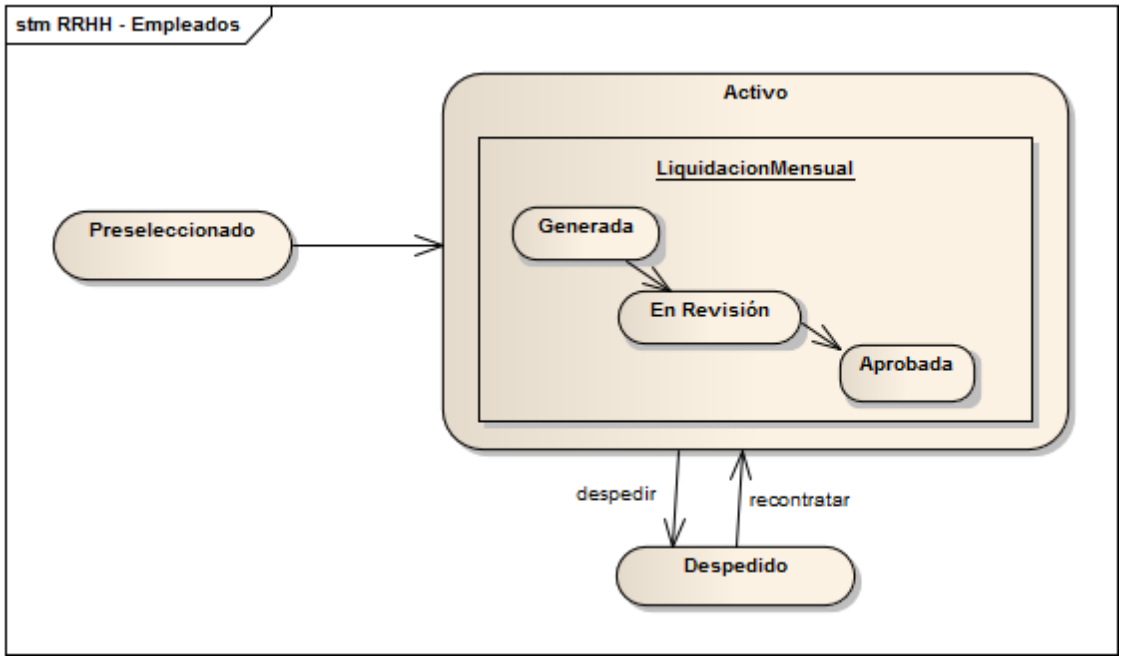
De todas formas la mayoría de los autores prefiere utilizar el diagrama de actividades para mostrar las condiciones que definen el flujo de acciones de un proceso en particular. Como diseñadores podríamos haber modelado el diagrama de estados de arriba simplemente así:



Estados anidados: cuando el estado de un objeto afecta al estado de otro/s, podemos explotar un estado para mostrar el diagrama de estados de otro objeto.

Ejemplo: tenemos el ciclo de vida de un empleado. Primero la empresa pre-selecciona candidatos, hasta que finalmente contrata a un empleado, que pasa a estar en estado activo. Entonces se registra el ciclo de vida de cada una de las liquidaciones mensuales, que confecciona el Departamento de Sueldos. Luego de un proceso de revisión interno, cada liquidación es aprobada por la Gerencia para que se deposite el dinero en la cuenta del empleado.

Se puede generar dos diagramas de estado o bien unificarlos, anidando el ciclo de vida de una liquidación de sueldos dentro del estado *activo* del empleado, como se muestra a continuación:

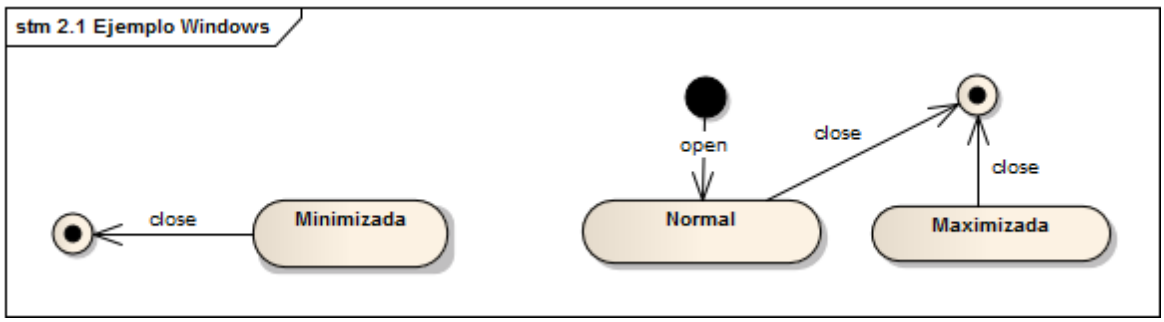

stm RRHH - Empleados
Activo
LiquidacionMensual
Preseleccionado
Generada
En Revisión
Aprobada
despedir
recontratar
Despedido

2 Ejemplos

2.1 Windows

(basado en un ejercicio del libro *Learning UML*, de Sinan Si Alhir, Ediciones O'Reilly)

Dado el siguiente diagrama de estados:

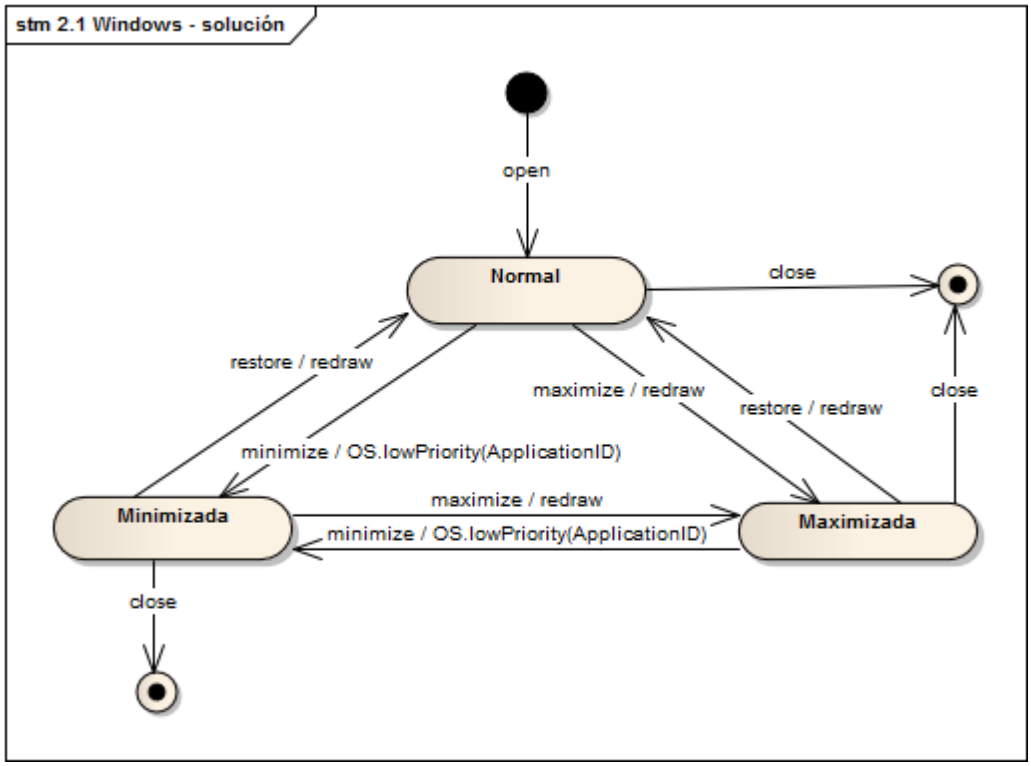


a. Agregar los eventos *restore* (que devuelve la ventana a estado Normal), *minimize* y *maximize*.

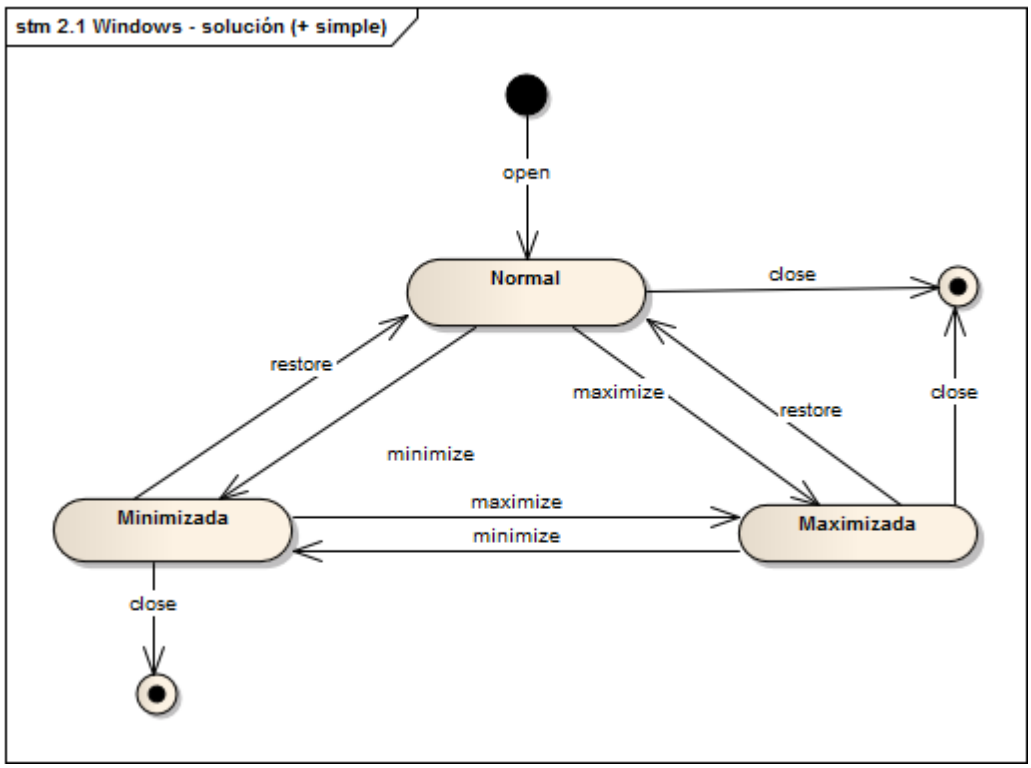
b. Cada vez que hay un restore o se maximiza la ventana se realiza la acción *redraw* para volverse a dibujar.

c. Cada vez que una ventana se minimiza le pide al sistema operativo que reduzca la prioridad de la aplicación enviándole el mensaje *lowPriority(ApplicationID)* para permitir que las otras aplicaciones accedan más tiempo al procesador.

Una posible solución:



Otra opción es no documentar las acciones, de la siguiente manera:



¿Cuál de los dos diagramas es más claro? ¿Cuál comunica más? ¿Cuál es necesario actualizar más seguido? Estas son preguntas que debemos tener en cuenta a la hora de documentar.

2.2 Facturas

A continuación se describe un diagrama de estados para las facturas que emite una importante empresa de Telecomunicaciones:

“Mensualmente el proceso de facturación genera las facturas para los clientes residenciales del Interior y Capital. Posteriormente se lanza la facturación de los Grandes Clientes Nacionales e Internacionales.

“La factura se envía a los respectivos domicilios y comienza a correr el plazo para pagar dicha factura. Una vez recibida la factura, el cliente puede cancelar la deuda contraída en cualquiera de las oficinas de pago de la empresa.

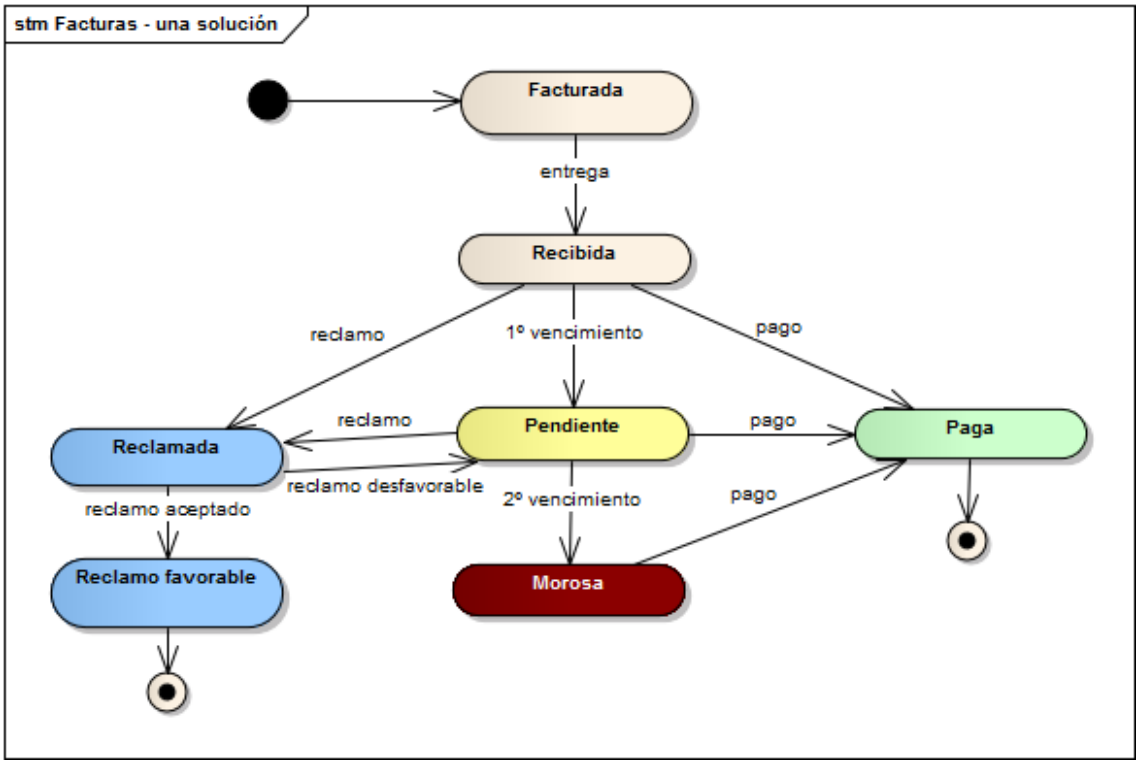
“Pasado el vencimiento, la factura pasa a estar pendiente. A veces el cliente reclama la factura ya que considera que hay conceptos mal calculados. En dicho caso la oficina de Reclamos estudia el caso y finalmente emite una resolución favorable o desfavorable. En caso de fallar a favor del cliente se refacturan los conceptos y vuelve a comenzar el ciclo. En caso de fallar a favor de la empresa la factura vuelve a estar pendiente.

“Pasado el segundo vencimiento la factura pasa a estar morosa y todos los datos de la factura y el cliente se envían al Departamento de Legales, donde el estudio de abogados hace la intimación judicial para que el cliente cancele la deuda. Eventualmente el cliente puede o no pagar.”

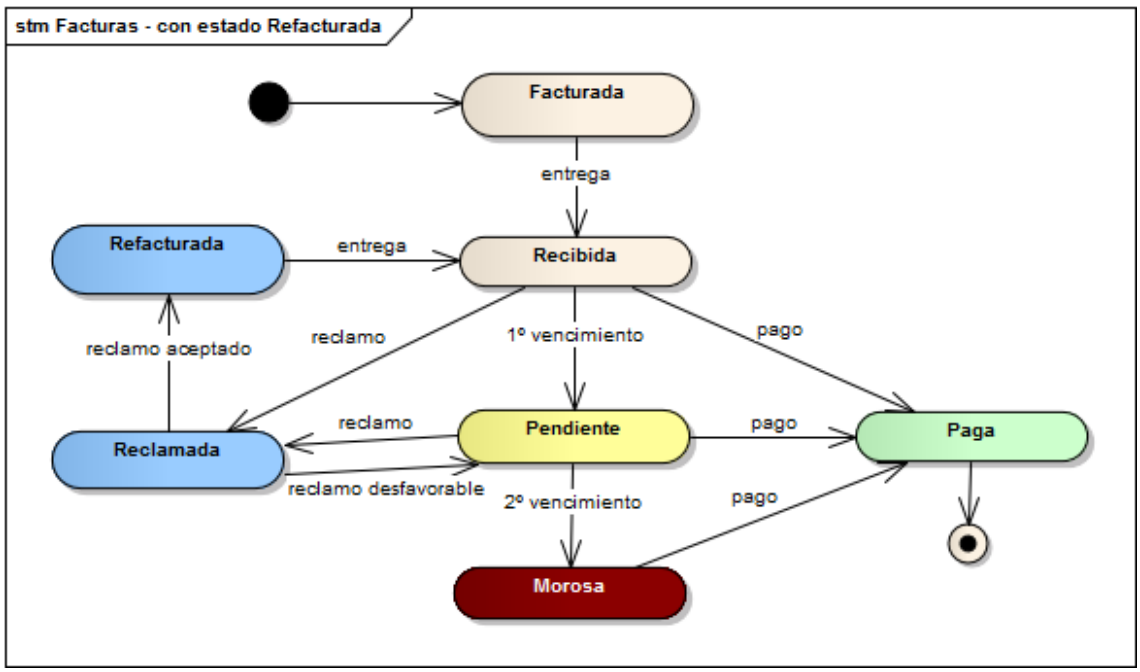
Bien, ¿cómo graficar este ejemplo con un diagrama de transición de estados?

- ¿Cuáles son los estados que nos interesa representar? ¿Cuáles son los estados finales? Que el cliente no pague la factura ¿implica un estado final?
- ¿Cómo comunicamos la refacturación de los conceptos por error de la empresa?
- ¿Cómo son las transiciones entre los distintos estados?
- También es interesante pensar: ¿cuáles son las responsabilidades del objeto Factura?
- Y esas responsabilidades, ¿dependen del estado en que esté? Por ejemplo, la factura no puede cancelarse contra otro comprobante de crédito si está en estado reclamada o paga. Tampoco tiene sentido pagar una factura si está reclamada o paga. Estas son decisiones de diseño que pueden implicar utilizar el pattern State: para mostrar la solución estática que debe implementar el programador utilizamos el diagrama de clases (el diagrama de estados ayuda a complementar esa visión).

Una posible solución:



También se podría haber creado un estado “Refacturada”, como sigue:



El trabajo del analista funcional es darse cuenta de que hay una pregunta para hacer al usuario: ¿interesa dejar registrada la refacturación en el historial de estados de la factura, o generamos directamente una nueva factura?

Se podría discutir si el estado Morosa no constituye un estado final. En realidad podríamos definir un estado Incobrable, que es cuando la empresa considera la

deuda como una pérdida (aún cuando siga reclamando por vía judicial el pago). Aún así, el cliente podría llegar a pagar la factura, por lo que el estado final “natural” de la factura es cuando se cancela totalmente.

3 Resumen

Este diagrama tiene reminiscencias del diagrama de autómatas finitos (modelos de Moore/Mealy) y más recientemente en el diagrama activo de flujo de datos (ADFD) de la metodología Shlaer y Mellor.

¿Quién hace el diagrama de estados? El analista funcional o bien el técnico.

¿Para quién es el diagrama de estados? Puede ser tanto para el usuario final como complemento al diagrama de casos de uso y de actividad, como para el técnico que debe implementar la solución.

¿Cuándo queremos hacer un diagrama de estado? En el caso del técnico, cuando queremos analizar si el estado de un objeto es lo suficientemente complejo. ¿Cómo sabemos si es “suficientemente complejo”? Un buen síntoma es que tenemos pocas operaciones, pero cada operación tiene o no sentido dependiendo del estado en que está cada objeto. En ese caso se justifica utilizar un “State Pattern” (de los patrones de diseño del libro del Gang of Four) y una muy buena forma de documentarlo es a través del diagrama de estados.

El segundo ejemplo de la presente práctica es un buen ejemplo de un diagrama de estados que muestra cómo podría utilizarse un patrón State para el objeto Factura.

Otro ejemplo: si al modelar el estado de una Cuenta bancaria tenemos este diagrama de estados

está claro que en este caso el diagrama de estados no aporta ningún valor agregado al conocimiento funcional del negocio para el usuario final. En lo que respecta a la parte técnica, no hay justificación para crear un objeto que represente el estado Activo, porque las operaciones del objeto Cuenta bancaria no dependen del estado en que se encuentre.

Una vez más vemos que en la fase de Diseño el rol que ejercen las personas es fundamental para generar sólo la documentación que se necesita.