2.4 Análisis Dinámico

Es necesario hacer el análisis sobre el modelo de objetos y construir el diccionario de datos antes de poder hacer análisis sobre el modelo dinámico.

- Durante el análisis del modelo dinámico se busca las secuencias de eventos permitidos para cada objeto.

- El conjunto de diagramas de estado resultante del análisis constituye el modelo dinámico del sistema.

- La mayoría de las interacciones se pueden separar en:
  - lógica de aplicación
  - interfaz de usuario

  El modelo dinámico captura la lógica de la aplicación. El análisis debe concentrarse primero en el flujo de información y control, más que en el formato de presentación, como el interfaz de usuario. La misma lógica de programación puede aceptar entradas de líneas de comando, archivos, o ratón, si los detalles superficiales son aislados. (
  El formato de las pantallas se especifica durante la etapa de diseño.)

Basados en la descripción del problema los diagramas de objetos y el diccionario de datos, los siguientes son los pasos a seguir en el análisis del modelo dinámico:
1) Preparar los escenarios
2) Identificar los eventos para cada escenario
3) Preparar los trazos de eventos
4) Preparar el diagrama de flujo de eventos
5) Construir los diagramas de estados
6) Verificar los diagramas de estado
7) Comparar con el modelo de objetos

2.4.1 Preparar los Escenarios

Se debe desarrollar el modelo dinámico por medio de *escenarios*, y no tratando de describir el modelo directamente, para asegurar que los pasos importantes no sean omitidos, y que el flujo general del sistema sea correcto.

- La descripción del problema puede especificar información necesaria pero deja abierta la manera de como se consigue.

Ejemplo ATM: La descripción del problema indica la necesidad de obtener datos de transacción del usuario, pero es impreciso sobre los parámetros exactos necesarios, y cual es el orden para pedirlos.

Los siguientes pasos deben seguirse en la preparación de escenarios:

• Se prepara escenarios para casos "normales", diálogos típicos con el usuario, interacciones sin entradas inusuales o condiciones de error.
• Luego se considera casos especiales, como secuencias de entrada omitidas, valores máximos y mínimos, y excepciones a transacciones normales.
• Luego se considera casos de error, incluyendo errores de usuario como valores inválidos o error de respuesta. Para muchas aplicaciones interactivas, el manejo de errores es la parte más difícil de implementar. Se debe dejar que el usuario cancele una operación o que pueda regresar a un punto bien definido para cada paso.
• Finalmente se debe considerar otro tipo de interacciones, basadas en las básicas, como pedidos de ayuda o consultas sobre el estado del sistema.

Escenario Normal para ATM:

- El ATM pide al usuario insertar la tarjeta; el usuario inserta la tarjeta.
- El ATM acepta la tarjeta y lee su código.
- El ATM pide la contraseña; el usuario entra "1234".
- El ATM verifica el código de banco con el consorcio; el consorcio verifica la cuenta con el banco "39" y notifica al ATM de su aceptación.
- El ATM pide al usuario seleccionar el tipo de transacción (retiro, depósito, transferencia, consulta); el usuario selecciona retiro.
- El ATM pide la cantidad de efectivo; el usuario entra $100.
- El ATM verifica que la cantidad está dentro de los límites predefinidos del sistema y pide al consorcio procesar la transacción; el consorcio pasa el pedido al banco, que eventualmente confirma el éxito de la transacción y regresa el nuevo balance de la cuenta.
- El ATM otorga el dinero y pide al usuario retirarlo; el usuario toma el dinero.
- El ATM pregunta si el usuario desea continuar; el usuario indica que no.
- El ATM imprime el recibo, expulsa la tarjeta, y pide al usario retirarlos; el usuario toma el recibo y la tarjeta.
- El ATM despliega la pantalla principal.

Escenario con Excepciones para ATM:

- El ATM pide al usuario insertar la tarjeta; el usuario inserta la tarjeta.
- El ATM acepta la tarjeta y lee su código.
- El ATM pide la contraseña; el usuario entra "9999".
- El ATM verifica el código de banco con el consorcio; el consorcio verifica la cuenta con el banco correspondiente y notifica al ATM de su rechazo.
- El ATM indica que la contraseña es incorrecta y pide al usuario entrarlo nuevamente; el usuario entra "1234" que es verificado con éxito por el ATM con el consorcio.
- El ATM pide al usuario seleccionar el tipo de transacción (retiro, depósito, transferencia, consulta); el usuario selecciona retiro.
- El ATM pide la cantidad de efectivo; el usuario cambia de idea y presiona "cancelar".
- El ATM expulsa la tarjeta, y pide al usuario retirarla; el usuario toma la tarjeta.
- El ATM despliega la pantalla principal.

Otras variantes:
El usuario no inserta la tarjeta a tiempo, el ATM no puede leer la tarjeta, la tarjeta expiró, la transacción es inapropiada para la cuenta, el valor no es válido, la máquina no tiene dinero o papel, las líneas de comunicación remotas están bajas, o la transacción es negada por patrones sospechosos en el uso de la tarjeta.

Escenarios adicionales deben ser escritos para partes administrativas del sistema del ATM, como *autorización del sistema del ATM*, *autorización de nuevas tarjetas*, *apertura de cuentas*, *añadir bancos al consorcio*, *obtener registros de transacción*.

2.4.2 Identificar los Eventos para cada Escenario

Se examinan los escenarios para identificar el intercambio de eventos entre los objetos del sistema.

- Se examina cada escenario para encontrar todos los eventos externos, incluyendo
  - interacciones con el usuario o disposistivos externos
  - interacciones entre objetos
  - puntos internos de decisión

- Eventos que afectan el flujo de control deben ser identificados.

Ejemplo ATM: *cuenta buena*, *cuenta mala*, *código malo* son eventos diferentes, no debe ser agrupados como estado de tarjeta.

- Se debe decidir cuales eventos son importantes a distinguir. (Algunos eventos no tienen efecto en el comportamiento del objeto y pueden ser ignorados.)

Ejemplo: los diferentes dígitos en un teclado podrían considerarse como el mismo evento, pero el "return" sería considerado un evento aparte.

- La mayoría de las interacciones y operaciones entre objetos corresponden a eventos.

Ejemplo ATM: *insertar tarjeta* es un evento enviado de *usuario* a *ATM*.
Otorgar efectivo es un evento (implícito) de *ATM* a *usuario*.

- Se asignan eventos a los objetos que los mandan y los reciben (un evento puede ser a la vez una entrada y una salida de un objeto)

Ejemplo ATM: *Entrar código* es un evento enviado por un agente externo *usuario* al objeto *ATM*.

- Se agrupan los eventos en clases de eventos basados en el efecto del flujo de control y los parámetros del evento, aunque los propios valores difieran.

Ejemplo ATM: *entrar código* debería ser una clase de eventos ya que el valor del código no afecta el flujo de control. De forma similar, *otorgar efectivo* es una clase de eventos, ya que la cantidad de efectivo otorgado no afecta el flujo de control.

Ejemplo Cajero: Los eventos del escenario normal son los siguientes:
pedir insertar tarjeta
insertar tarjeta
aceptar tarjeta
leer código de trajeta
leer código de banco
pedir entrar código
entrar contraseña "1234"
verificar contraseña
verificar cuenta
cuenta buena
pedir seleccionar tipo de transacción
seleccionar tipo de transacción "retiro"
pedir cantidad
entrar cantidad "100"
verificar cantidad
procesar transacción
transacción exitosa
otorgar dinero
pedir retirar dinero
tomar dinero
pedir si desear continuar
terminar
impirimir recibo
expulsar tarjeta
pedir retirar recibo y tarjeta
tomar recibo y tarjeta
desplegar pantalla principal

2.4.3 Preparar los trazos de eventos.

Se debe incluir los diferentes eventos en un diagrama mostrando las clases de objetos que los mandan y reciben.

- Se debe mostrar cada *escenario* como un *trazo de eventos*, una lista ordenada de eventos entre objetos diferentes asignados a una columna en una tabla. Si más de un objeto de la misma clase participa en un escenario, se debe asignar una columna separada para cada objeto. El evento aparece como salida del objeto que lo envía y como entrada del recibidor.

Ejemplo Cajero: El diagrama en la Figura 2.194 muestra un trazo de eventos para el escenario normal del ejemplo del ATM.

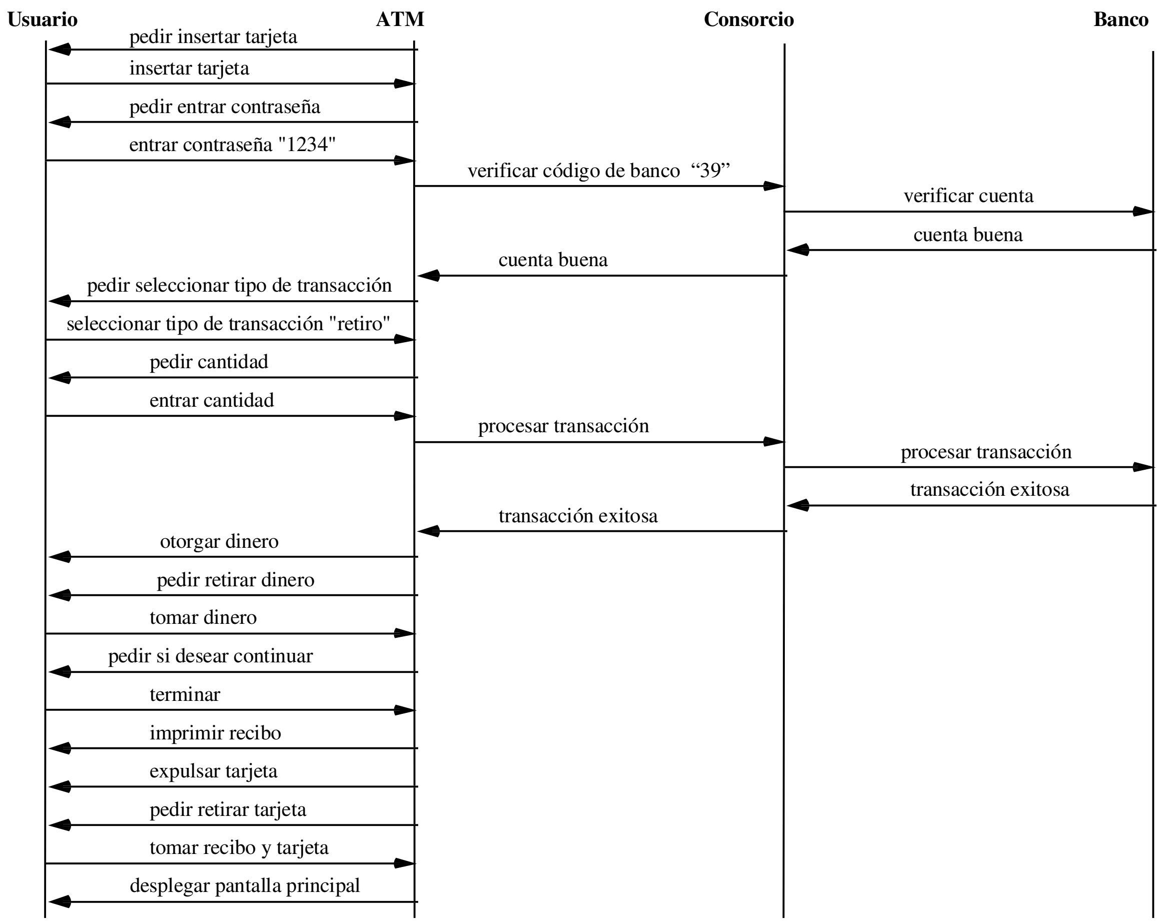

Figura 2.194. Trazo de eventos para escenario normal del ATM.

Ejemplo Cajero: El diagrama en la Figura 2.195 muestra un trazo de eventos para el escenario con excepciones del ejemplo del ATM.

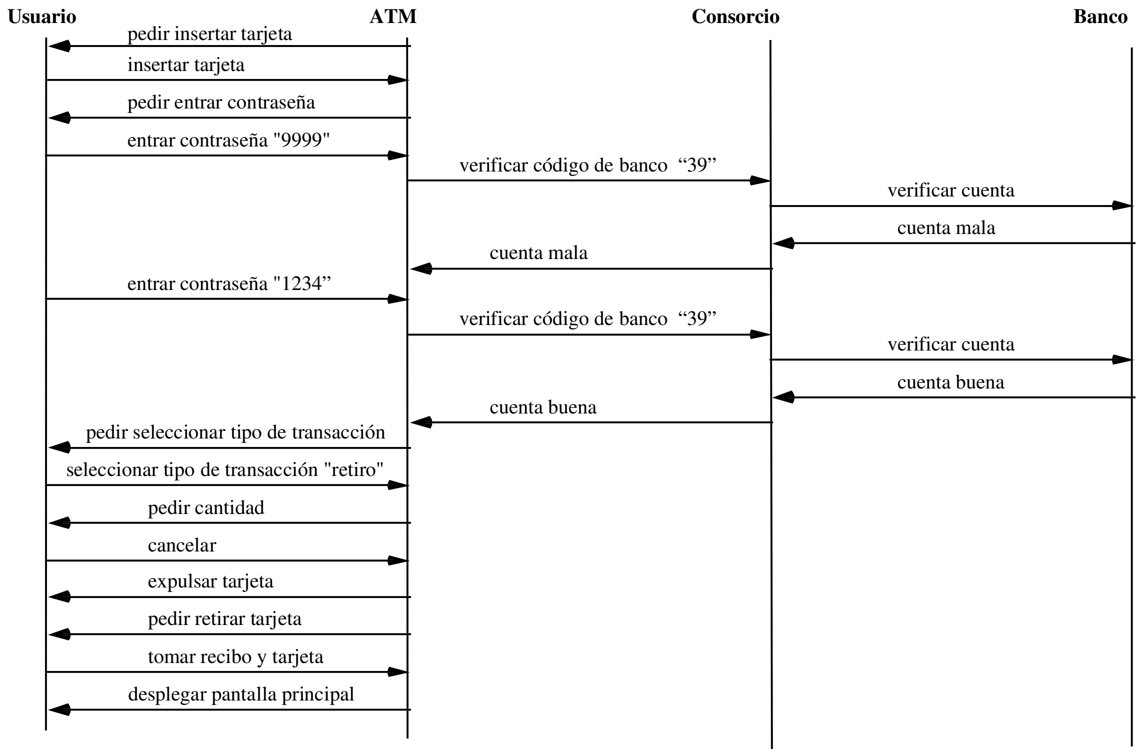

Figura 2.195. Trazo de eventos para escenario con excepciones del ATM.

2.4.4 Preparar el Diagrama de Flujo de Eventos.

Se deben mostrar todos los eventos del sistema con un diagrama de flujo de eventos. Se incluye los eventos de todos los escenarios, y trazos de eventos correspondientes, incluyendo eventos de error.

- El diagrama hace un resumen de eventos entre clases, mostrando los posibles flujos de control, pero sin importar secuencias.

- El diagrama de flujo de eventos es la contraparte dinámica del diagrama de objetos. Los caminos en el diagrama de objetos muestran posibles flujos de información, mientras que los caminos en el diagrama de flujo de eventos muestra posibles flujos de control.

Ejemplo Cajero: La Figura 2.196 muestra el diagrama de flujo de eventos.

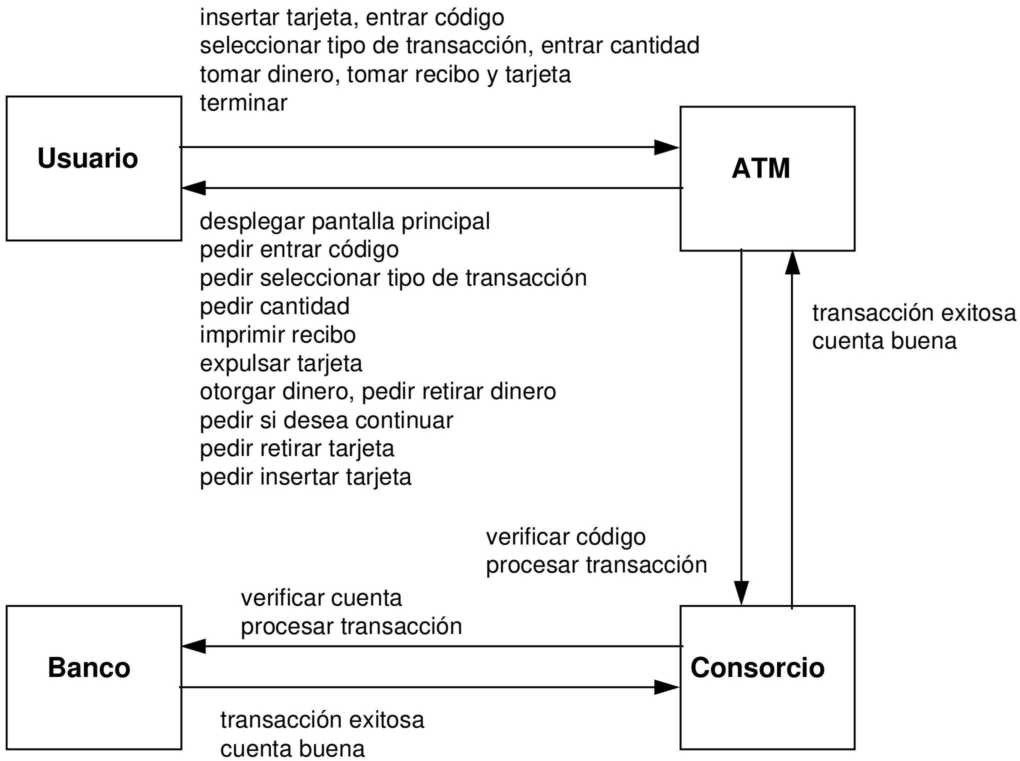

Figura 2.196. Diagarama de flujo de eventos para el sistema "normal" de cajeros.

Ejemplo Cajero: La Figura 2.197 muestra el diagrama de flujo de eventos con eventos adicionales.

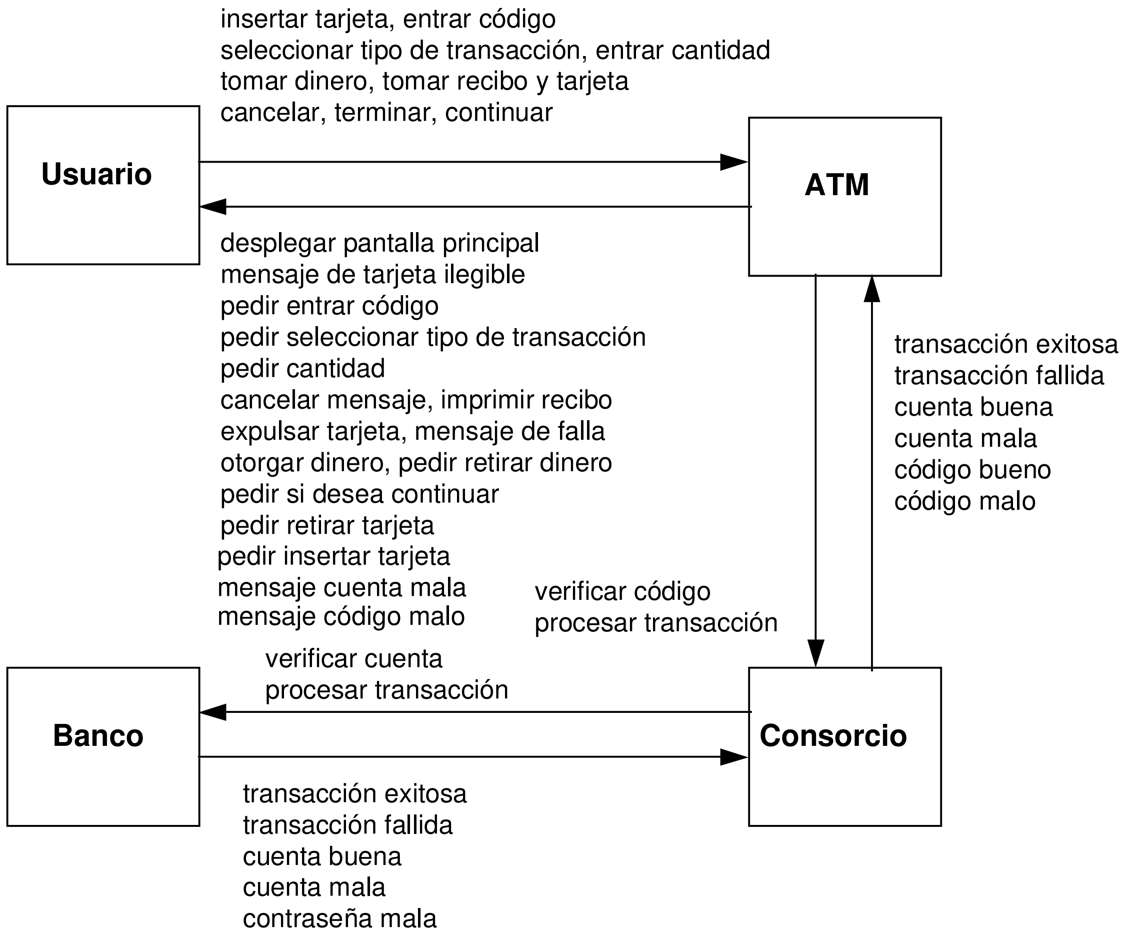

Figura 2.197. Diagarama de flujo de eventos para el sistema "con excepciones" de cajeros.

2.4.5 Construir los Diagramas de Estados

Se debe preparar un diagrama de estado para cada clase de objeto que tenga un comportamiento dinámico importante, mostrando los eventos que el objeto recibe y envia.

- Se prepara una diagrama de estados para cada objeto con un comportamiento dinámico importante.

- Cada escenario debe corresponder a un camino a través de uno de los diagramas de estado.

Los siguientes son los pasos a seguir:

1. Para hacer el diagrama de estados de una clase particular se toma del diagrama de trazo de eventos los eventos afectando a la clase que es modelada.

- Se organizan los eventos en un camino cuyos arcos están etiquetados por eventos de entrada encontrados en una columna en el trazo.

- Los eventos de salida etiquetan las actividades de los estados de la clase.

- El diagrama inicial va a ser una secuencia de eventos y estados.
- Se provee nombres de estado a grupos significativos de atributos.
- Se caracteriza los estados con una descripción.

Ejemplo Cajero: Las clases *ATM*, *estación de cajero*, *consorcio* y *banco* son objetos activos que intercambian eventos. Las clases *tarjeta de dinero*, *transacción* y *cuenta* son objetos pasivos que no intercambian eventos. *Cliente* y *cajero* son agentes externos al sistema y no tienen que ser implementados por dentro.

Ejemplo Cajero: El diagrama inicial para el escenario y trazo de eventos "normal" se muestra en las Figuras 2.198, 2.199, y 2.200, para el *ATM*, *consorcio* y *banco*, respectivamente. Algunos eventos de salida han sido combinados como actividades secuenciales dentro de estados sencillos. (El diagrama para la *estación de cajero humano* es bastante similar a la del *ATM*.)

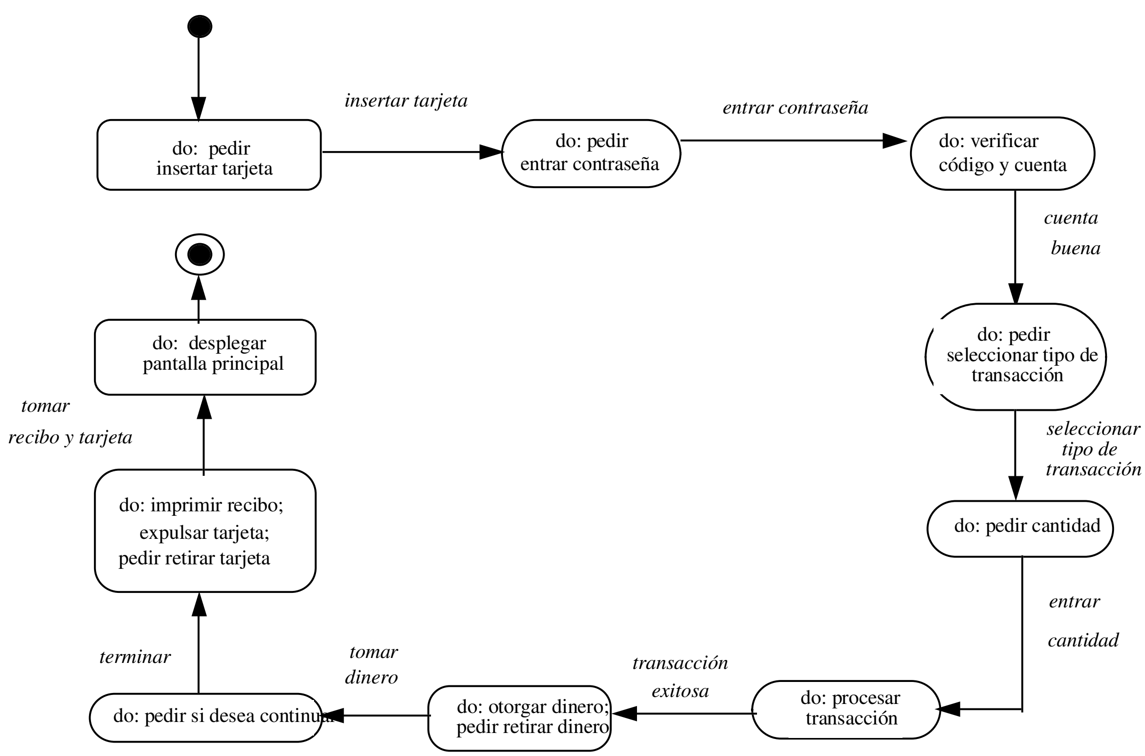

Figura 2.198. Diagrama de estado para el escenario normal para la clase *ATM*.

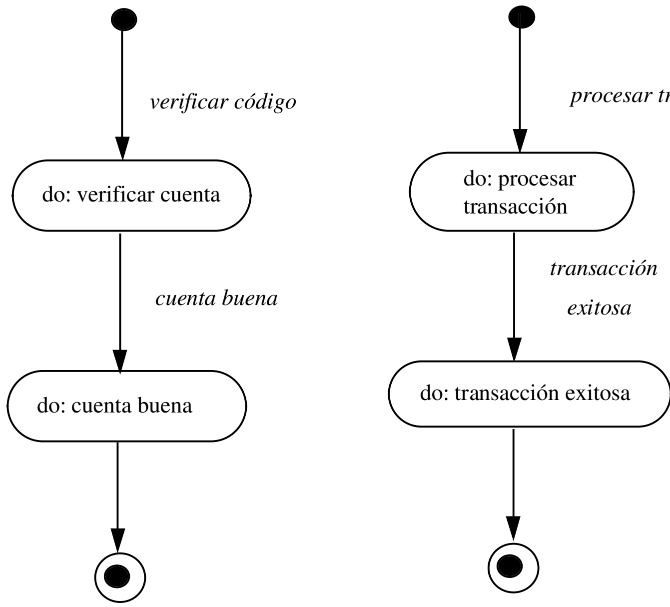

Figura 2.199. Diagrama de estado para el escenario normal para la clase *consorcio*.

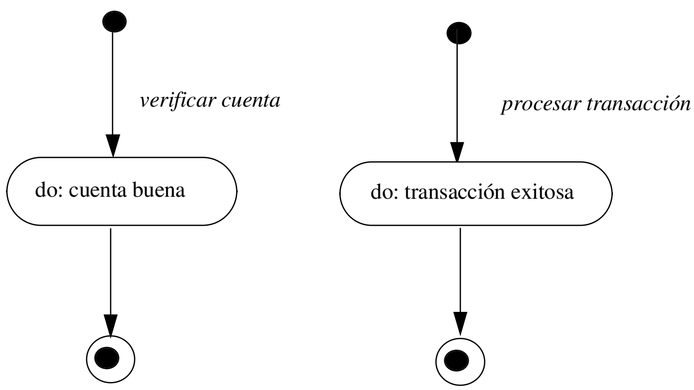

Figura 2.200. Diagrama de estado para el escenario normal para la clase *banco*.

2. Si una secuencia de eventos puede repetirse indefinidamente, ella forma un ciclo, y se debe reemplazar tales secuencias cuando sea posible.

Ejemplo Cajero: El diagrama inicial para el escenario y trazo de eventos "normal" conteniendo un ciclo en lugar de un diagrama finito se muestra en la Figura 2.201.

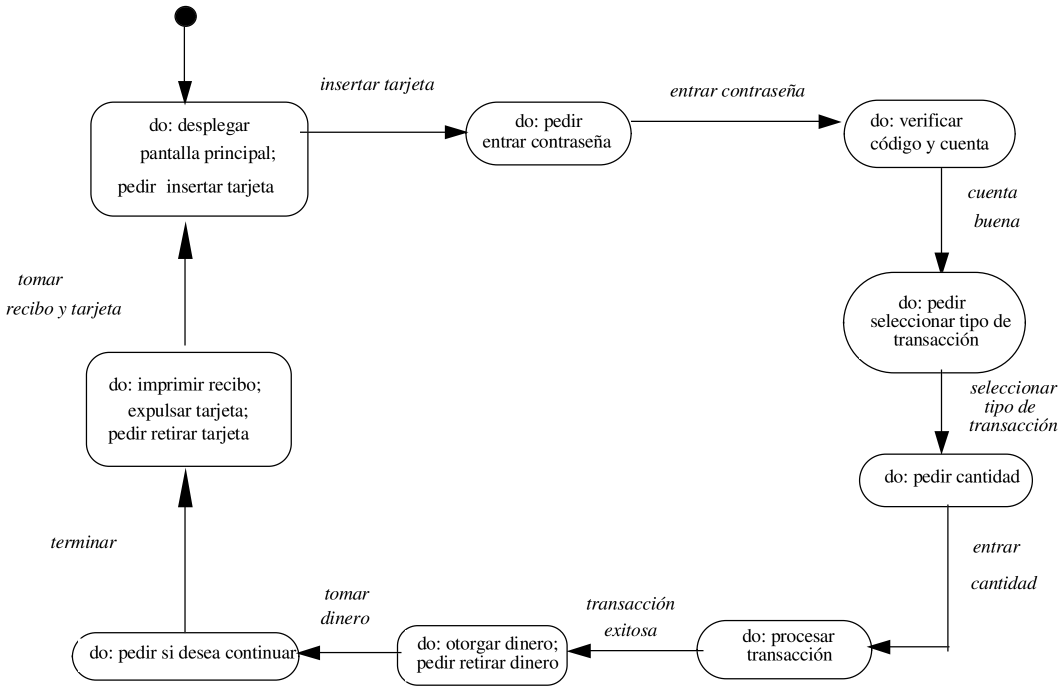

Figura 2.201. Diagrama de estado para el escenario normal para la clase *ATM* conteniendo un ciclo.

3. Luego de completar el diagrama de estado basado en un escenario, se debe unificar con otros escenarios en el diagrama de estado, encontrando el punto donde los escenarios divergen de escenarios previos.

- Mientras se examina los estados y escenarios se puede añadir otros posibles eventos que ocurren en cada estado.

- Si las actividades dentro de un estado solo mandan eventos a otros objetos, entonces se deben elimar tales actividades y utilizar la nomenclatura de un segundo evento.

- Lo más difícil es decidir en cual estado un camino alterno se junta al diagrama existente.

4. Luego que los eventos normales han sido considerados, se añade casos de borde y casos especiales, como pedidos para cancelar una transacción luego de haber sido mandada a procesar.

- Manejando errores de usuario requiere más código y pensamiento que lo normal, complicando el programa.

- En casos en que el usuario no responde rápidamente se puede generar una interrupción en la interacción (*time out*).

- Se termina el diagrama de estados de una clase cuando el diagrama cubre todos los escenarios, y maneja todos los eventos que pueden afectar al objeto en cada uno de sus estados.

- Se puede sugerir nuevas preguntas ("que pasa si") para considerar cómo eventos todavía no manejados afectan el estado del objeto.

- Si existen interacciones complejas con entradas independientes se puede organizar el modelo como diagramas anidados.

- Se deben juntar los caminos alternos. (Se junta en el estado en el que el objeto se "olvida" cual camino fue seguido.)

Ejemplo: Insertar dos monedas de $500 es equivalente a insertar una moneda de $1000 a una máquina.

- Se debe evitar mezclar demasiado los datos e información del estado para tratar casos especiales, como dos caminos idénticos excepto por algunas circunstancias
  - se puede agregar parámetros para distinguir entre los atributos de los eventos
  - se puede usar transiciones condicionales
  - se puede usar subdiagramas concurrentes

- Se debe tener cuidado con dos caminos que parezcan idénticos pero que se puedan distinguir en algunas circunstancias.

Ejemplo: Algunos sistemas las secuencias de entrada se repiten si el usuario hace un error al entrar información, pero se abortan después de un número de errores. Se puede incluir un parámetro, *número de errores*, para guardar información, y por lo menos una transición debe depender del valor del parámetro.

- Una alternativa para partir el diagrama de estado en subdiagramas concurrentes, es usar un subdiagrama para la línea principal y otro para distinguir la información.

Ejemplo: Un subdiagrama que permite a un usuario hacer errores, distinguiendo los estados *sin error*, *error*.

Ejemplo Cajero: Los diagramas de estado para la clase *ATM, consorcio, y banco* se muestran en lan Figura 2.202, 2.203, y 2.204, respectivamente.

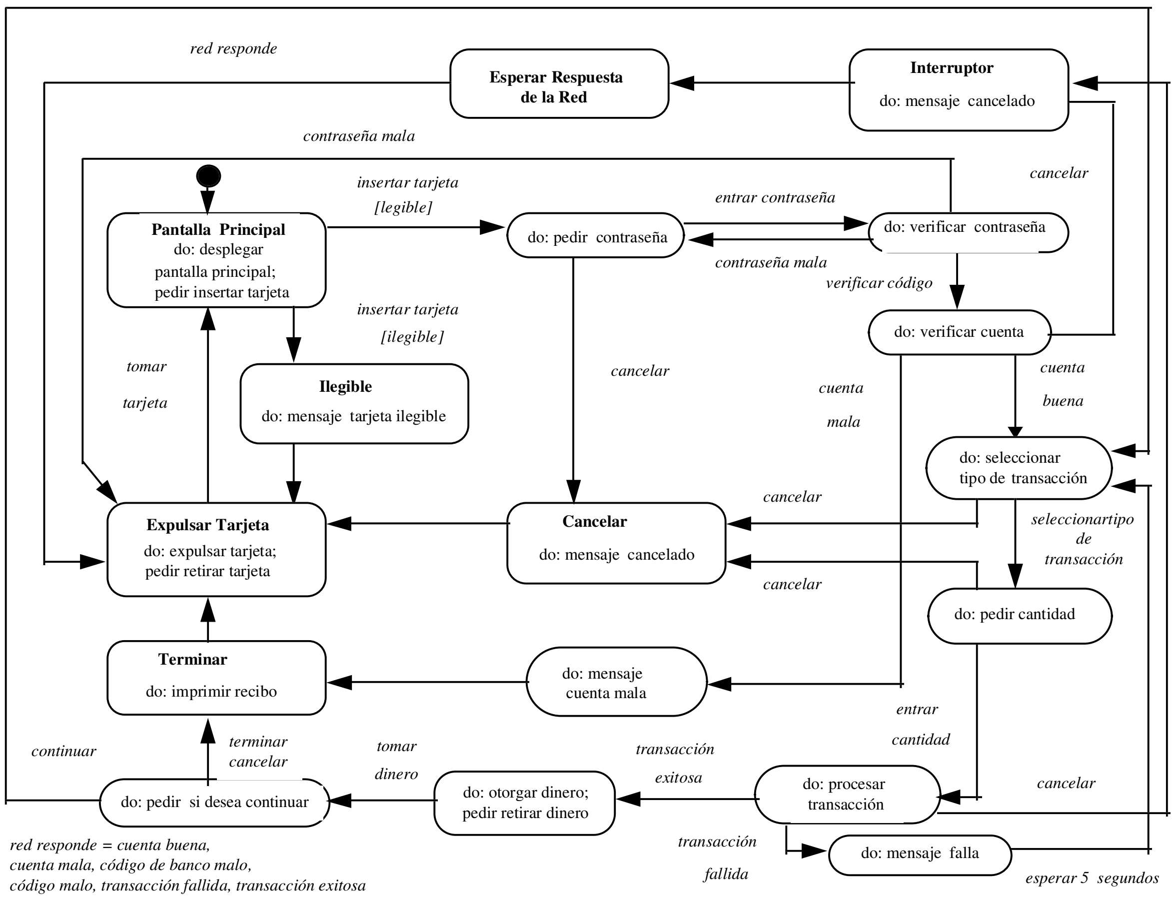

Figura 2.202. Diagrama de estado para el *ATM*.

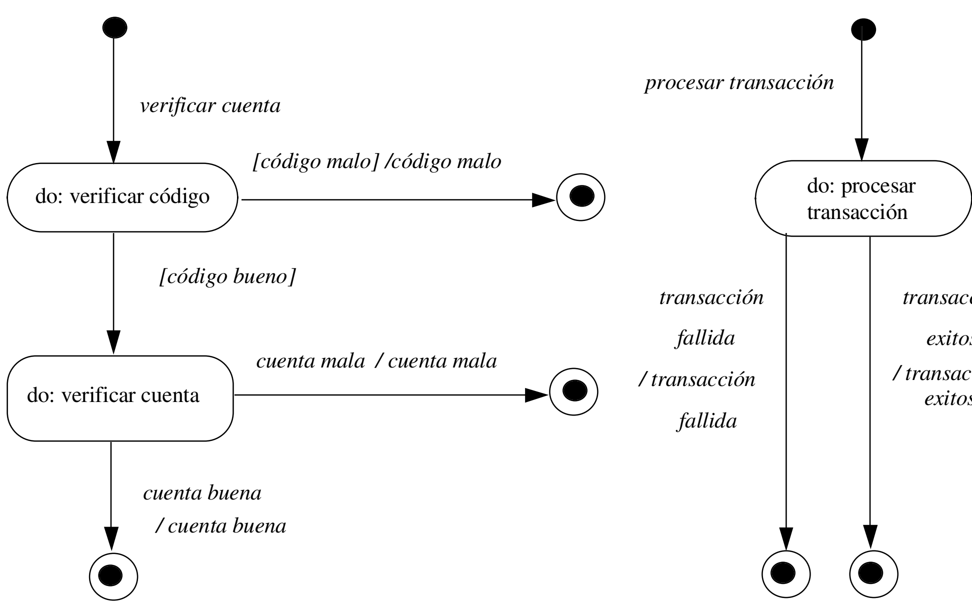

Figura 2.203. Diagrama de estado para el *consorcio* con excepciones.

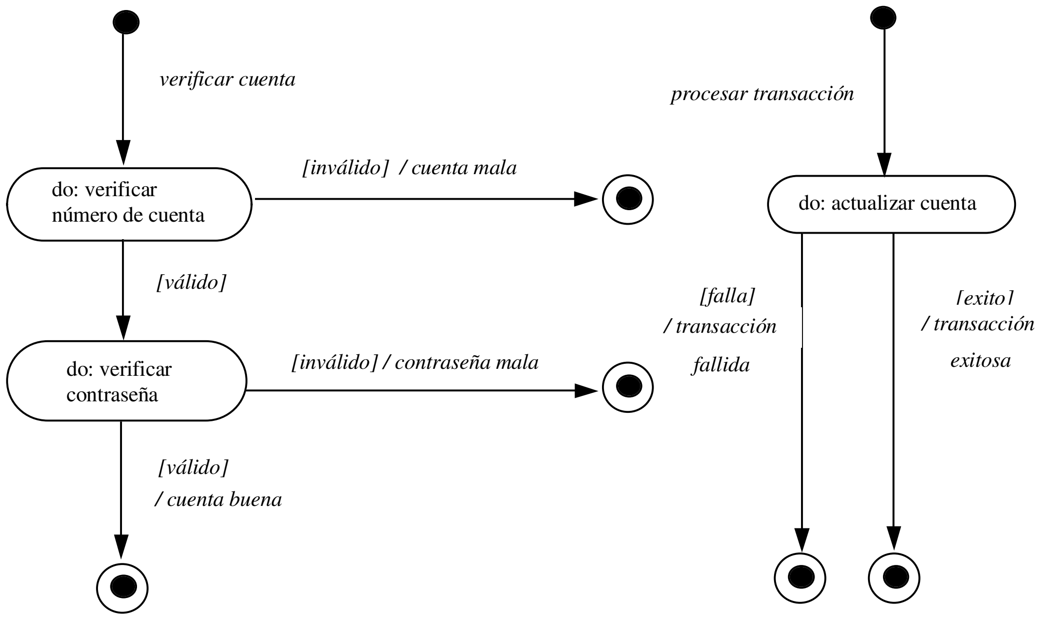

Figura 2.204. Diagrama de estado para el *banco* con excepciones.

2.4.6 Verificar los Diagramas de Estado

Se debe verificar la consistencia entre los diferentes diagramas de estado para cada clase. Se debe seguir los efectos de un evento de entrada de un objeto a otro y ver que los escenarios correspondan.

- identificar eventos ausentes
- usar anidacion para eliminar interacciones complejas con entradas independientes
- identificar el mandador y recibidor de cada eventos
- verificar la consistencia de los eventos compartidos entre estados diferentes del diagrama
- ubicar los estados sin predecesores o sucesores y verificarlos
- seguir los efectos de los eventos de entrada a los diferentes objetos asegurando que los escenarios correspondan
- cuidarse de condiciones de "carrera"

- Estados sin predecesores o sucesores son sospechosos, al menos que especifiquen puntos de comienzo o fin en la secuencia de interacción.

- Como los objetos son esencialmente concurrentes, se debe verificar que no existan errores de entrada en momentos no esperados.

Ejemplo Cajero: Una cuenta puede ser accesada concurrentemente por más de una máquina a la vez. Se debe asegurar que se haga una sola actualización cada vez. El examen de los diagramas de estado muestran que *código malo* es mandado a consorcio pero no es recibida por ATM. Debe ser añadida, seguida por *imprimir código malo* y *expulsar tarjeta*.

2.4.7 Comparar con el Modelo de Objetos

Analizar si los eventos en el modelo dinámico identifican atributos adicionales que deban ser agregados al modelo de objetos.

- Verificar si las restricciones de generalizacion son consistentes entre los dos modelos.
- Verificar que los dos modelos reflejen relaciones de agregación similares.
- Iterar el proceso de análisis en ambos modelos, agregando información adicional según la descripción del problema.
- Validar los modelos con el cliente e incorporar retroalimentación.
- Actualizar el diccionario de datos para reflejar los nuevos cambios al modelo de objetos.