Introducción a máquinas de estados finitos (FSM) y Statecharts

Técnicas Digitales II
UTN-FRH

Introducción

- Las máquinas de estados finitos (FSM) son un modelo de comportamiento de un sistema donde las salidas del mismo dependen tanto de las entradas actuales como anteriores. Para manejar la dependencia de las entradas anteriores se definen los estados.

- Las máquinas de estados se definen como un conjunto de estados que sirven de intermediarios en esta relación de entradas y salidas, haciendo que el historial de señales de entrada determine, para cada instante, un estado para la máquina de forma tal que la salida depende únicamente del estado y las entradas actuales

Introducción

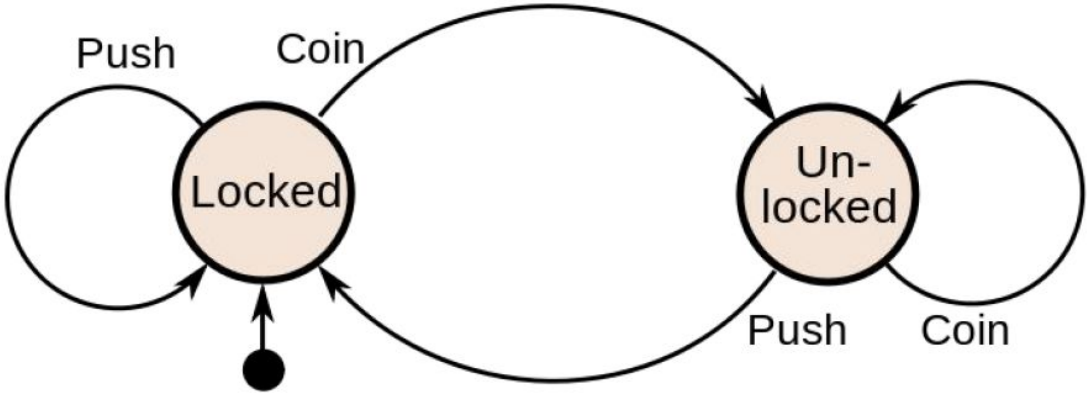
- Las máquinas de estados finitos o diagramas de estados nos van a permitir:
  - Desarrollar un modelo de mayor abstracción que un programa en ensamblador o C.
  - Simplificar la comprensión, simulación y desarrollo del sistema a desarrollar.
  - Mejorar el mantenimiento de la aplicación.
  - Estructurar la solución de software e incluso utilizar generadores de código automáticos para implementarla.
  - Estructurar y optimizar el tiempo de procesador entre las diferentes tareas que tiene que resolver el sistema a desarrollar.

Partes de una FSM

- **Estado**. Situación o condición del sistema donde se realiza alguna actividad o se espera algún evento externo.
- **FSM**. Sistema que puede ser particionado en un conjunto acotado de estados que no se solapan.
- **Evento**. Ocurrencia en tiempo y espacio de algo significativo para el sistema.
- **Acción**. Respuesta del sistema a un cambio de estado, evento o cualquier cambio de interés para el sistema.
- **Transición**. Es el cambio entre un estado y otro.

Reconociendo las partes

- Vamos a reconocer las partes de una máquina de estados finitos (FSM) que resuelve un molinete.
  - Eventos
  - Estados (estado inicial)
  - Acciones
  - Transiciones



Clasificación de las FSM

- Hay varias maneras de clasificar a las FSM e incluso pueden solaparse las diferentes clasificaciones.
  - **Reconocedoras**. Son aquellas que tienen una salida binaria que generan una salida positiva o activa luego de una secuencia de entrada determinada (TD1).
  - **Transductoras**. Convierten una secuencia de señales de entrada en una secuencia de acciones, pudiendo ser una salida binaria o una salida más compleja.
  - **FSM tipo Moore**. Las salidas actuales dependen únicamente del estado actual del sistema.
  - **FSM tipo Mealy**. Las salidas actuales dependen del estado actual y de las entradas actuales del sistema.

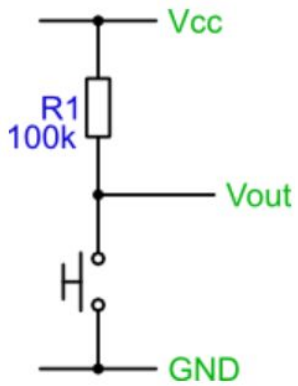
Metodología de diseño

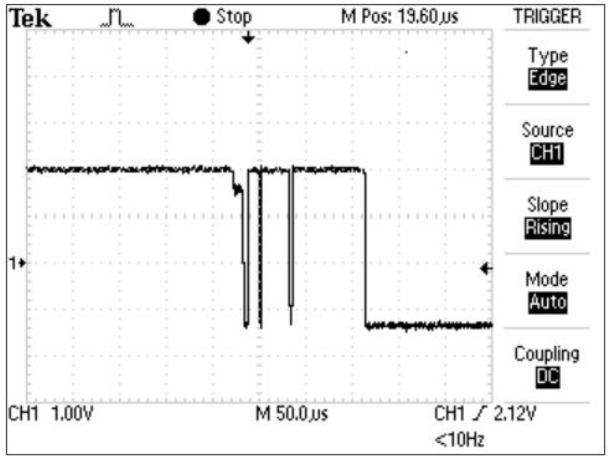
- Como en cualquier diseño se debe partir de un conjunto de especificaciones del sistema.

- Pasos recomendados para el diseño:
  - Identificar eventos y acciones.
  - Identificar estados.
  - Añadir las transiciones.

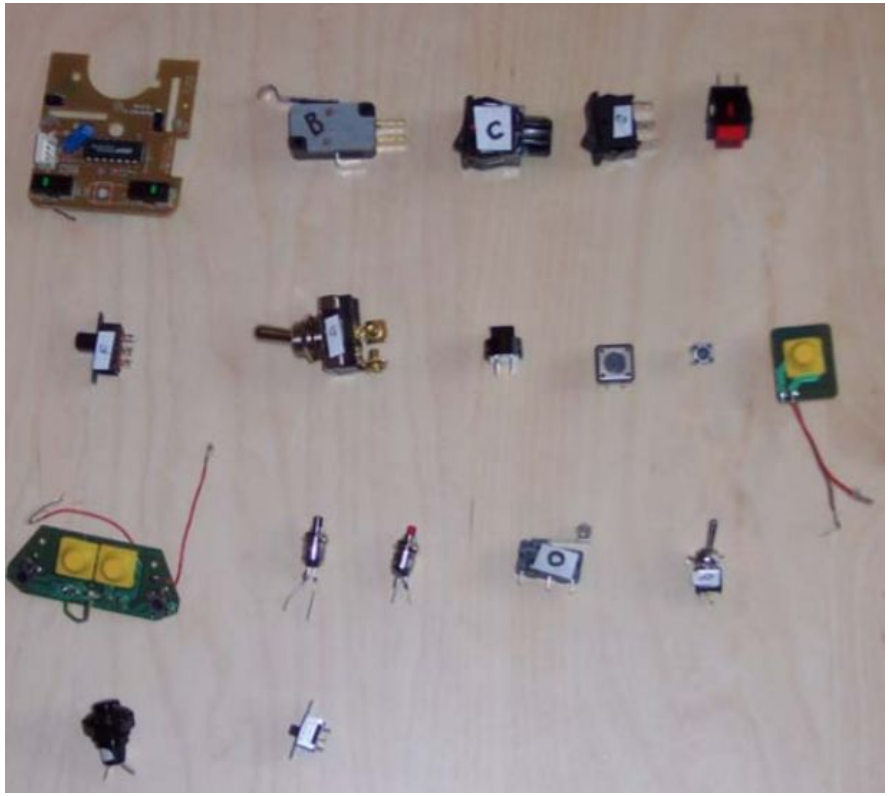
Diseño de una FSM

- Diseño de un antirrebote para un pulsador.
  - En la captura del osciloscopio se ve la tensión Vout. El problema que nos aparece es que al presionar el pulsador el contacto mecánico ***"rebota"*** y no tenemos una transición clara de uno a cero, sino más bien una serie de "rebotes" hasta que se estabiliza. Cuando se libera el pulsador sucede exactamente lo mismo. Para circuitos contadores, este comportamiento genera cuentas espurias, comportamiento errático, etc





Rebotes en pulsadores



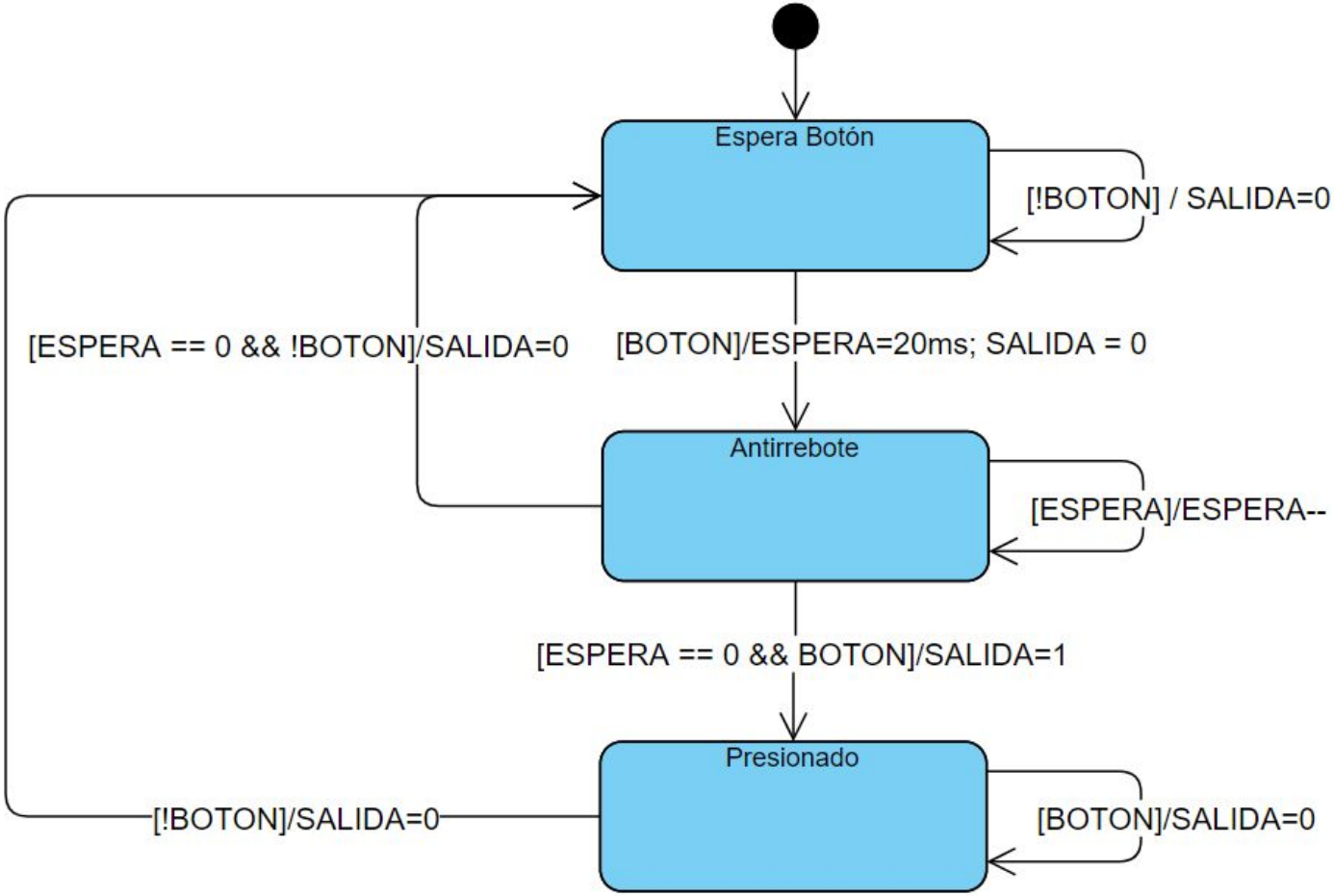
- Los rebotes dependen de la mecánica del pulsador por lo que vamos a tener una gran variabilidad en el tiempo y cantidad de los rebotes.
- Como regla general el tiempo del rebote se suele considerar del orden de 1ms a 20ms.

Diseño del antirrebote

- Especificación.
  - El antirrebote o debounce trata de:
    - Detectar una transición en el pulsador.
    - Esperar un tiempo de “guarda”.
    - Volver a leer el pulsador.
    - Si se confirma la transición generar la salida y esperar a que se vuelva al estado inicial, sino directamente volver al estado inicial.

- Identificar eventos y acciones.
- Identificar estados.
- Añadir las transiciones.
- Obtener el diagrama de estados.

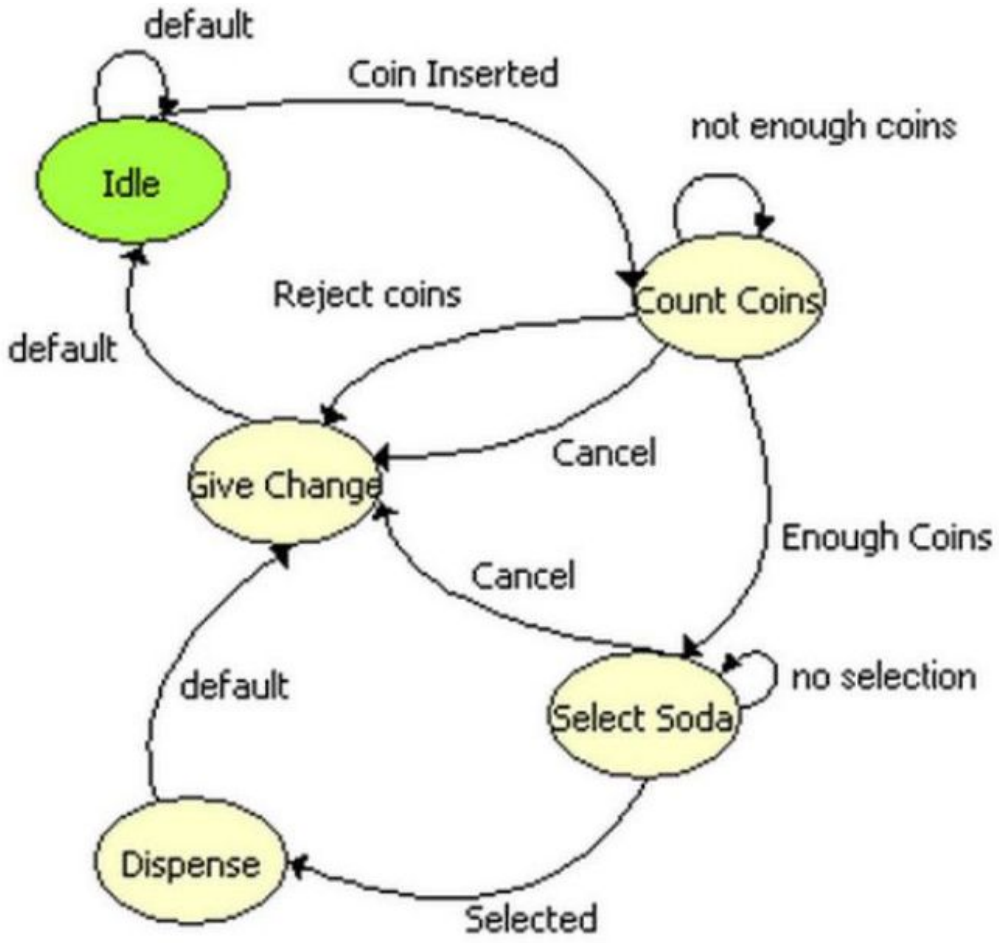
Antirrebote (una posible solución)



Diseño de una FSM (2)

- Se desea construir una máquina de vender bebidas que:
  - Espere una cantidad determinada de monedas para entregar la bebida y el vuelto.
  - Permite elegir el tipo de bebida (todas con el mismo costo)
  - Tenga un botón (sin rebote) para cancelar la compra.

Diseño de una FSM (2)



Resultados del diseño con FSM

- Complejidad y máquinas de estados. Con un bajo aumento de la complejidad, la cantidad de estados y transiciones aumenta casi exponencialmente: “explosión de estados”

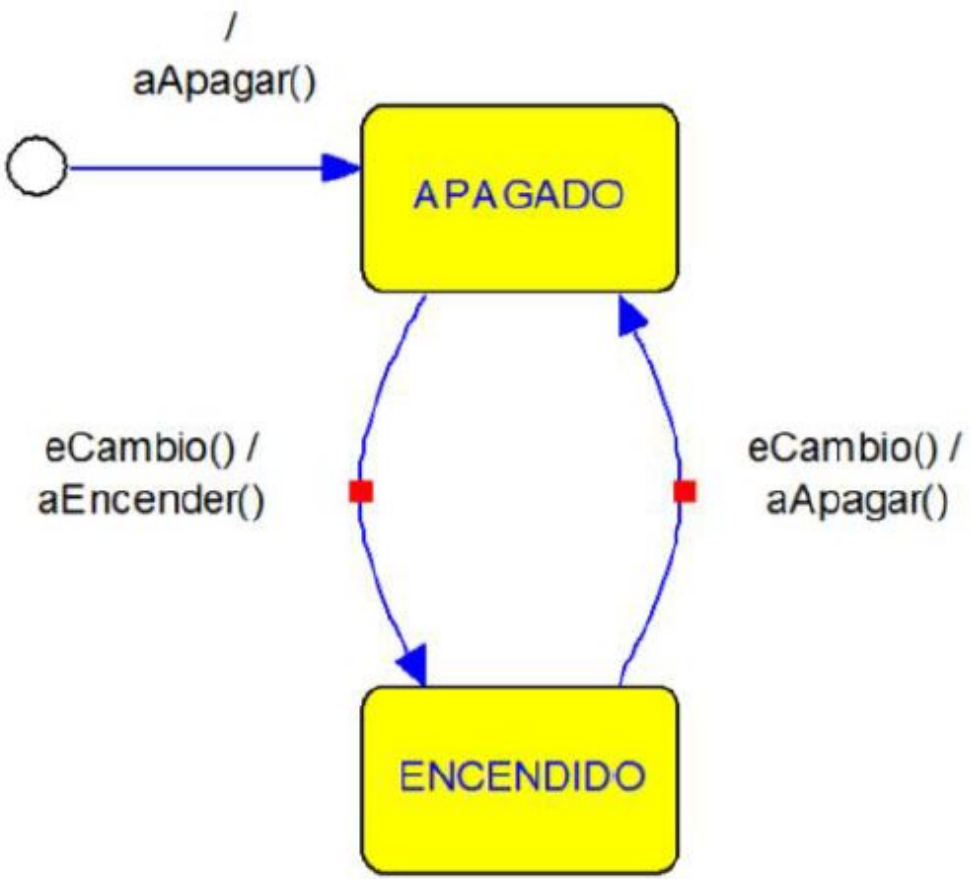
- ¿Cómo atacar a la complejidad?
  - Jerarquía
  - Modelos más completos.

- Uso de statecharts.

Statecharts ¿Qué son?

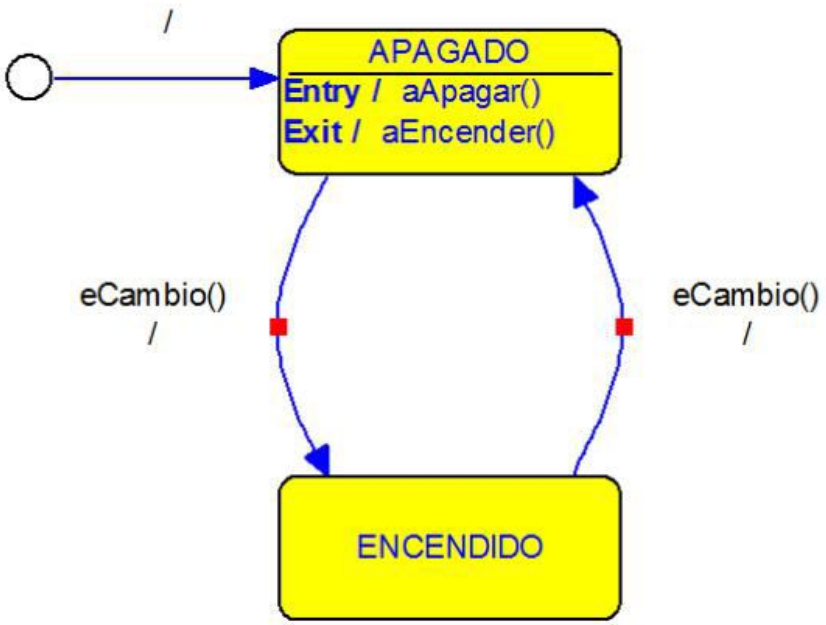
- Los statecharts son una extensión de las máquinas de estados finitos. Estos definen:
  - Acciones de Entrada y Salida
  - Guardas
  - Estados jerárquicos
  - Concurrencia
  - Transiciones internas
  - Pseudoestados

Elementos principales de un statechart



- Eventos
  - eCambio()
- Acciones
  - aEncender()
  - aApagar()
- Estados
  - APAGADO
  - ENCENDIDO
- Transiciones
  - Tres transiciones. Inicio, desde apagado a encendido y desde encendido a apagado.

Acciones de entrada y salida

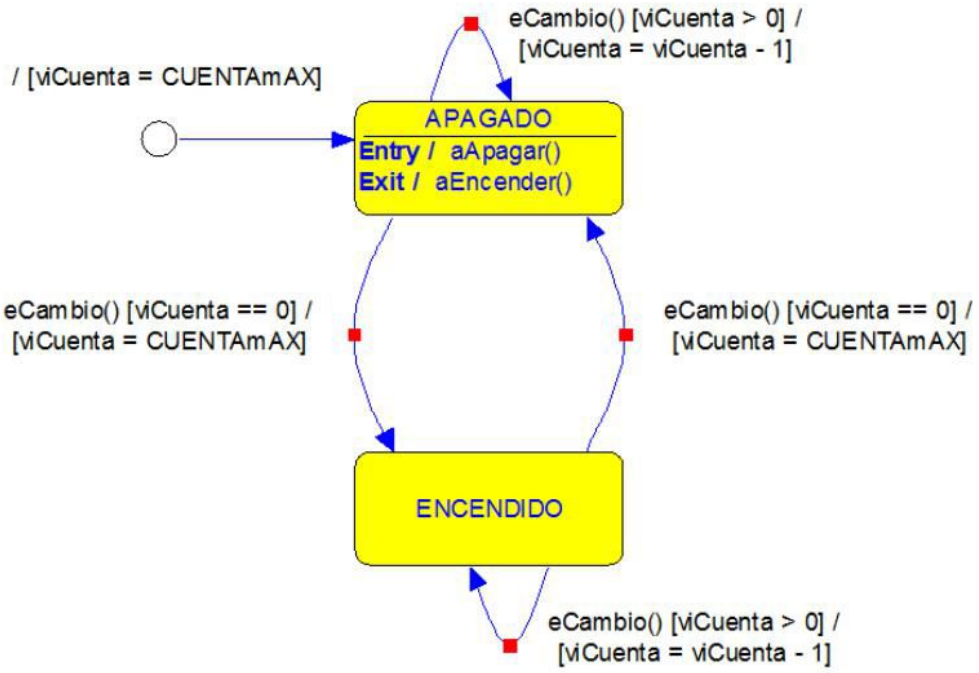


- Las acciones Entry y Exit se ejecutan cuando se entra a un estado y cuando se sale del mismo.
- Son acciones opcionales.
- Se ejecutan siempre que se entra o se sale del estado, no importa por qué transición se realice.

Acciones

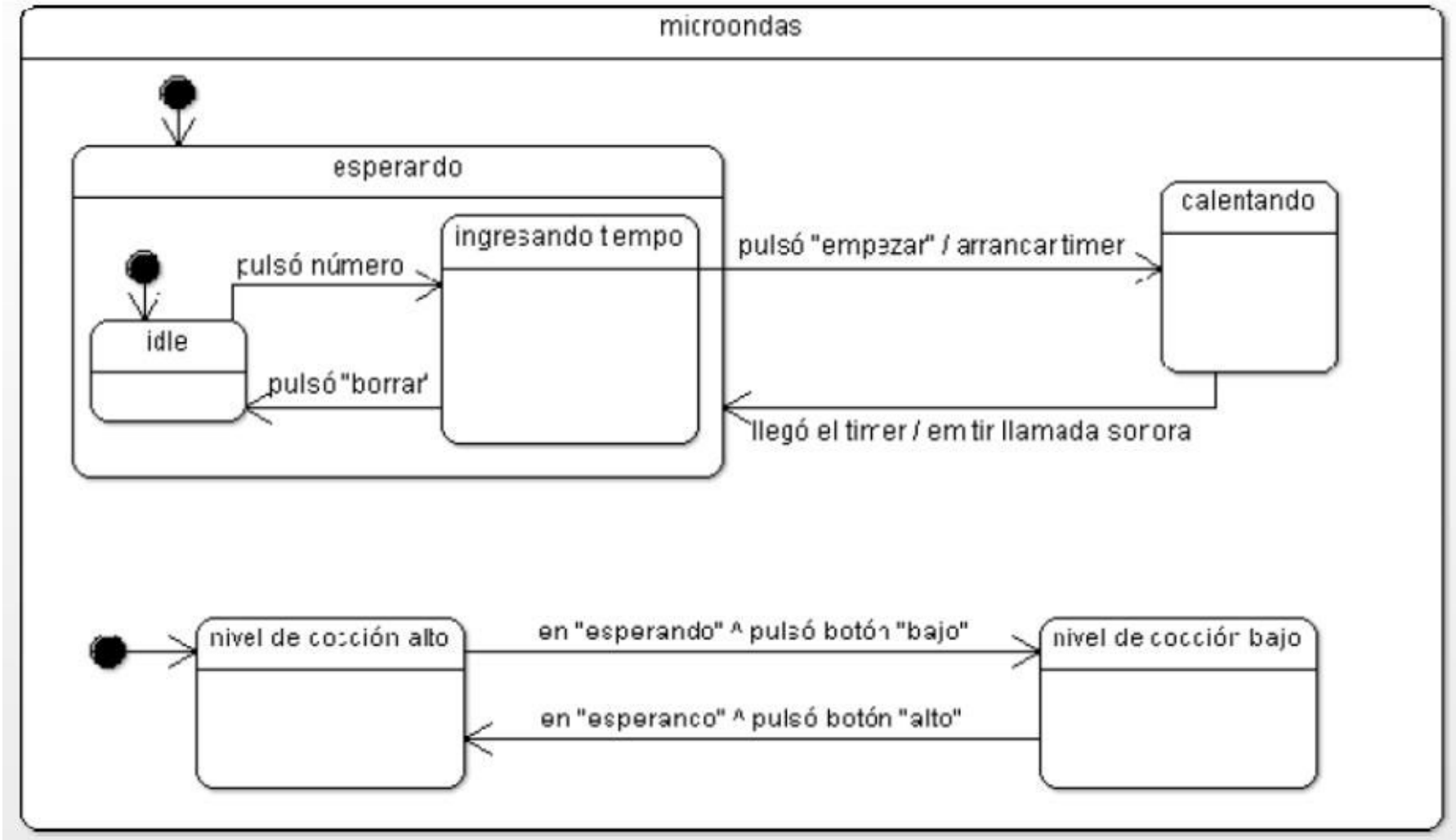
- **Tareas (do)**: Acciones que se realizan mientras se permanezca en el estado.
- **Acción de Entrada**: Se ejecuta únicamente cuando se entra en el estado, independientemente de la transición que lo genera.
- **Acción de Salida**: Se ejecuta únicamente cuando se sale del estado, independientemente de la transición que lo genera.
- **Acción en Transición**: Se ejecuta únicamente en una transición entre estados.

Condiciones de guarda

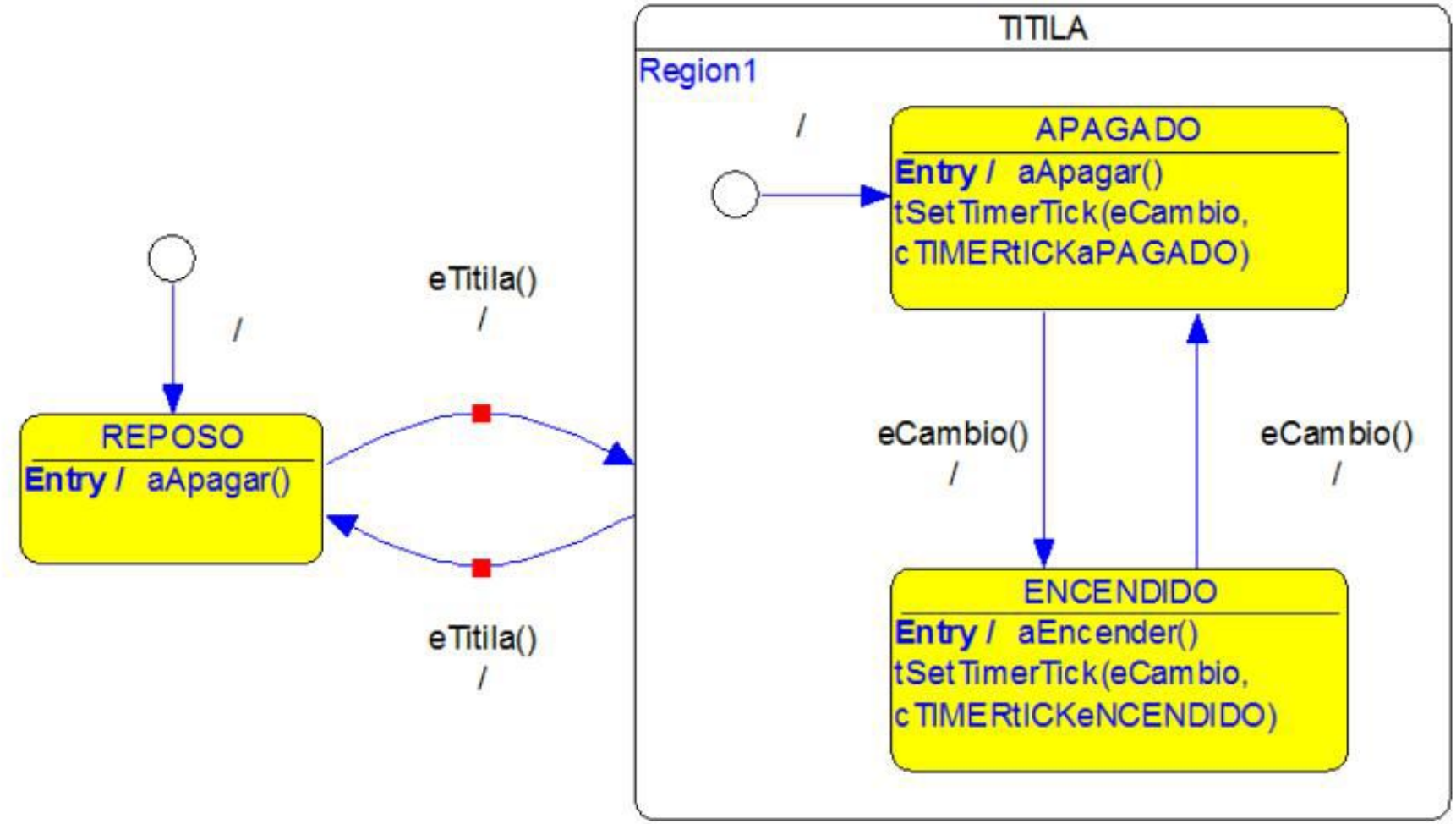


- Las condiciones de guarda son expresiones booleanas que permiten habilitar o deshabilitar acciones o transiciones en función de su evaluación.
- Son evaluadas dinámicamente.
- Tienen un fuerte impacto en la generación de código.

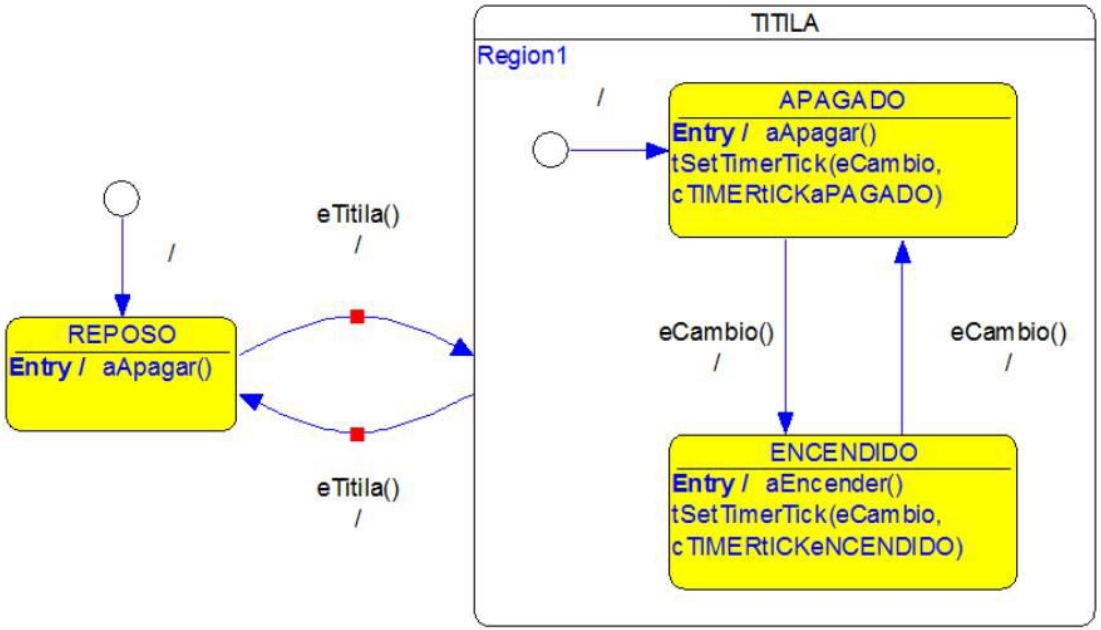
Concurrencia



Estados jerárquicos



Transiciones Internas

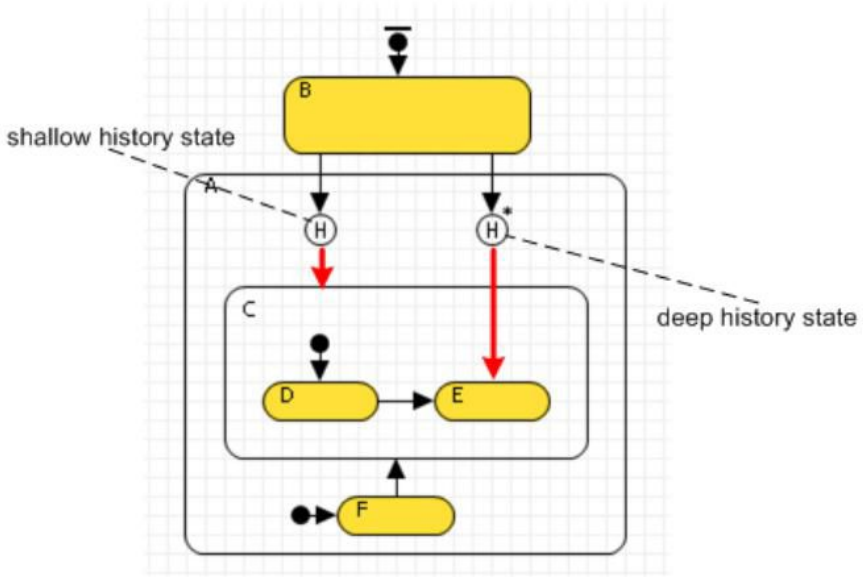


- Las transiciones generadas por los eventos eCambio() no van a generar acciones Entry o Exit en el estado TITILA.
- Estas transiciones se las conoce como transiciones internas.

Pseudoestados

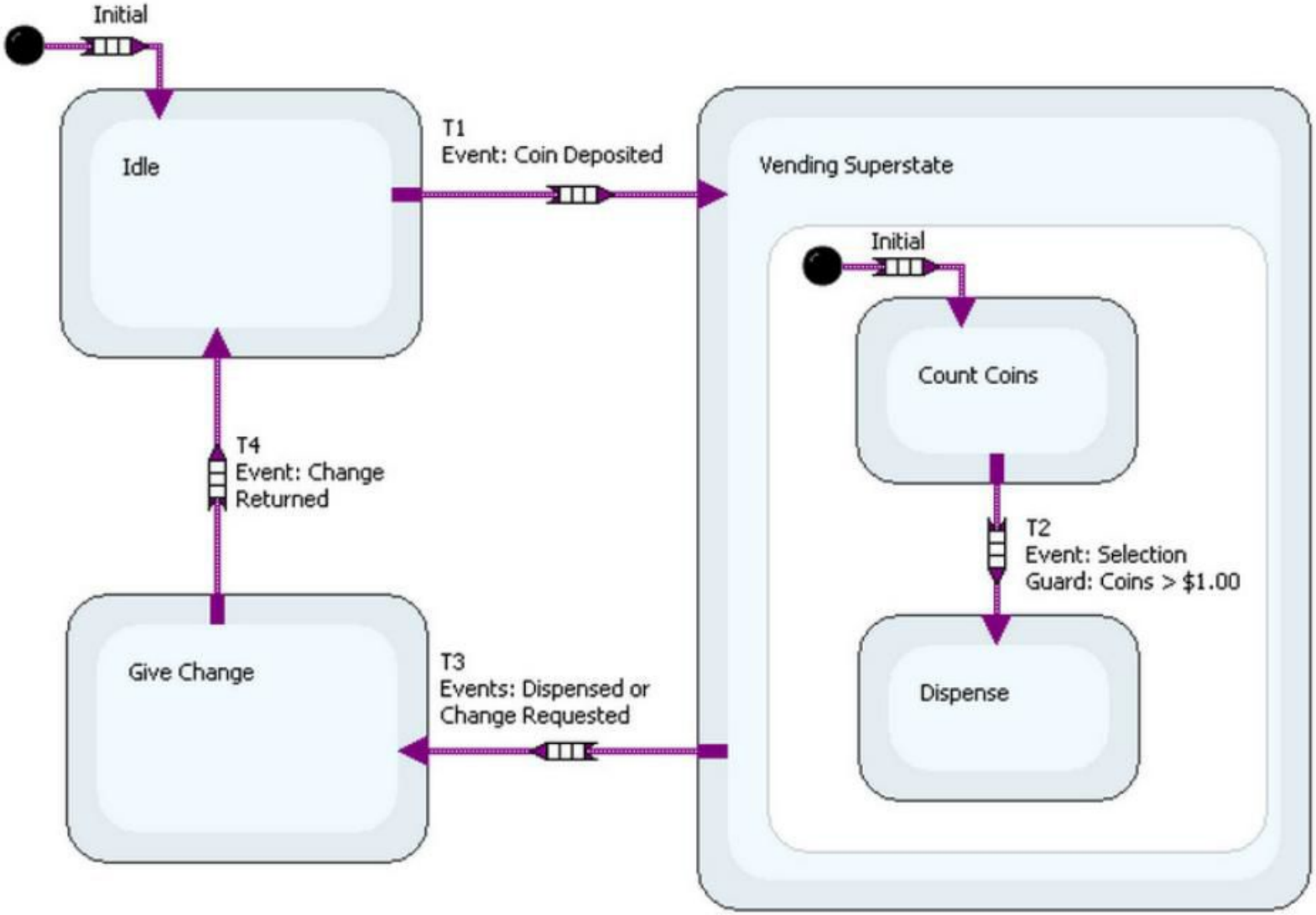
- Son similares a los estados, con la diferencia que el sistema no puede permanecer en ellos. Tipos:
  - Inicial (Reset). Se indica con un círculo negro relleno.
  - Junturas. Se indican con un punto negro. Sirven para juntar o bifurcar transiciones que tienen partes en común.
  - Historia. Se indican con un círculo que contiene una “H”. Si se entra en una transición que termina en este pseudoestado, el próximo estado pasa a ser el último estado que tenía la (sub) máquina en cuestión.
  - Fork/Join. Con un fork se “abre” una transición para que pueda terminar en estados de máquinas concurrentes. Join hace lo contrario.
  - Final. Se indica con dos círculos concéntricos, el interior negro y el exterior vacío.

Pseudoestados: historia

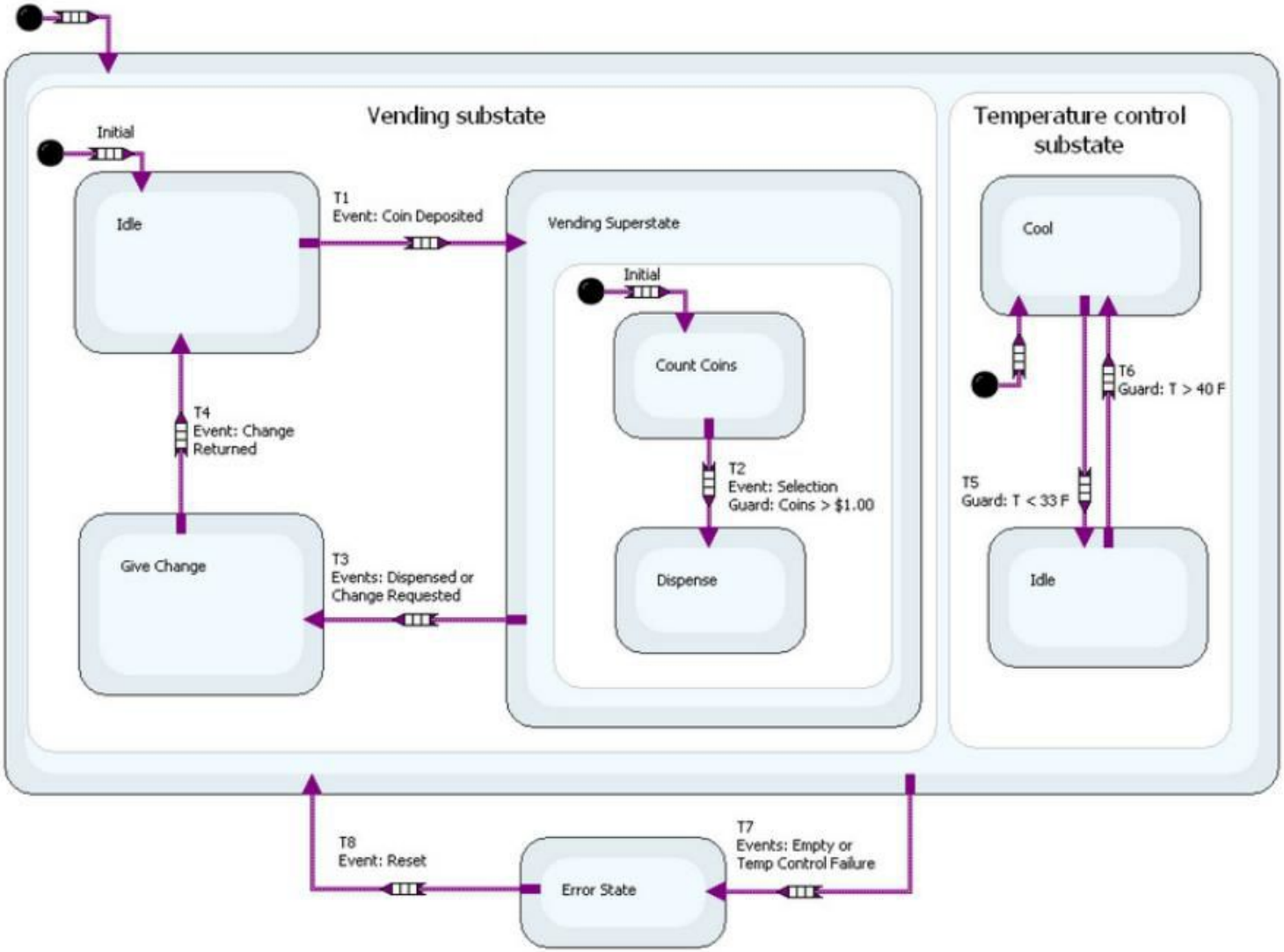


- El estado historia, recuerda al estado de la jerarquía actual.
- El estado historia profunda, recuerda el estado más interior de la jerarquía.

Ejemplo de una expendedora



Ejemplo de una expendedora



Método de diseño propuesto

1. Identificar eventos
2. Identificar acciones
3. Identificar estados
4. Agrupar por jerarquías
5. Agrupar por concurrencia
6. Agregar las transiciones y acciones

Bibliografía

- Object-Oriented Analysis and Design with Applications (3rd Edition). Grady Booch.
- The Unified Modeling Language Reference Manual. Booch, Jacobson, Rumbaugh.
- Object-Oriented Analysis and Design with Applications (3rd Edition). Grady Booch.
- The Unified Modeling Language Reference Manual. Booch, Jacobson, Rumbaugh.
- Practical Statecharts in C/C++: Quantum Programming for Embedded Systems. Miro Samek.
- Statecharts. Andrés Djordjalian. Seminario de Sistemas Embebidos. FIUBA.
    - www.indicart.com.ar/seminario-embebidos/Statecharts.pdf
- Diagrama de Estado. Juan Manuel Cruz.
    - http://laboratorios.fi.uba.ar/lse/seminario/material-1erC2011/Sistemas%20Embebido%202011%201er%20Cuatrimestre%20-%20Diagrama%20de%20Estado%20-%20Cruz.pdf