Clase 4

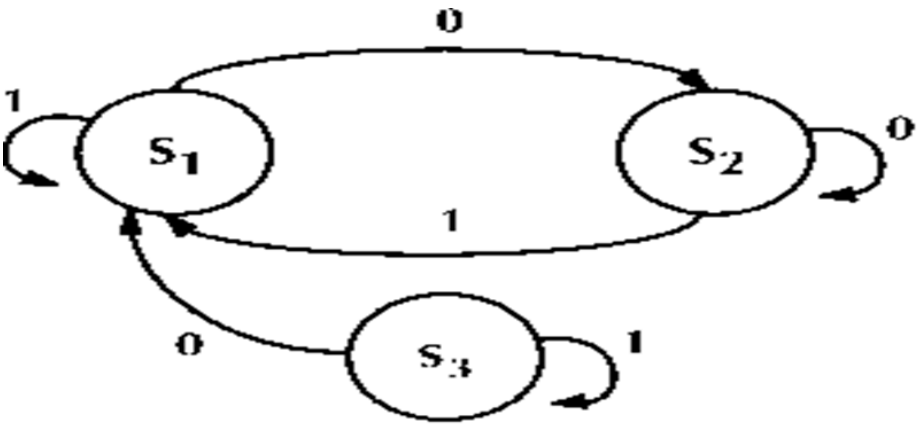
Diagrama de Transición de Estados (DTE)

Volviendo a las ***técnicas de especificación de requerimientos***, las estáticas y dinámicas, veremos más en detalle la técnica dinámica llamada ***Diagramas de Transición de Estados (DTE)***.

Máquinas de Estado Finito:

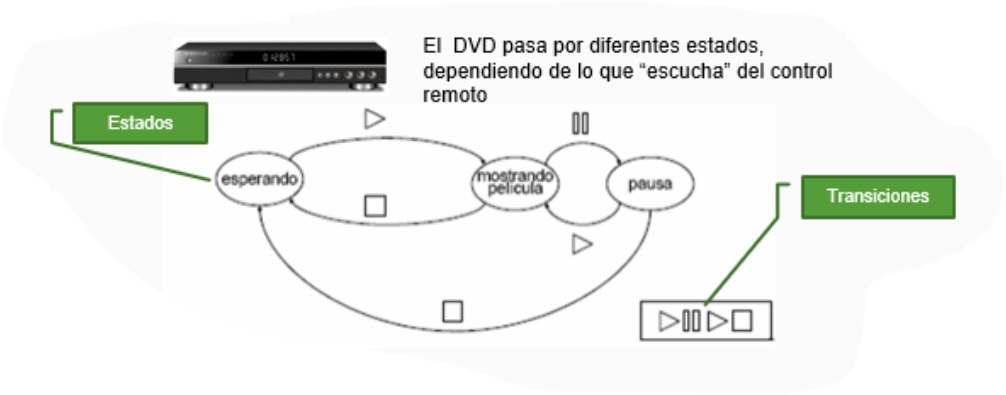
Es un modelo donde se describe al sistema como un ***conjunto de estados*** donde el sistema ***reacciona a ciertos eventos posibles*** (***externos o internos***).

`f(Si, Cj) = Sk` → La siguiente función relaciona un estado y su posible cambio. Al estar en el estado `Si`, la ocurrencia de la *condición* `Cj` hace que el sistema *cambie* al estado `Sk`. Por ejemplo:




```
f(S1, 0) = S2
f(S1, 1) = S1
f(S2, 0) = S2
f(S2, 1) = S1
f(S3, 0) = S1
f(S3, 1) = S3
```

Otro ejemplo, que considera un conjunto de estados por los que pasa un DVD dependiendo de los eventos que genere un control remoto:



f(A, ▷) = B

f(B, □) = A

f(B, ▯▯) = C

f(C, ▷) = B

f(C, □) = A

A: esperando

B: mostrando

C: pausa

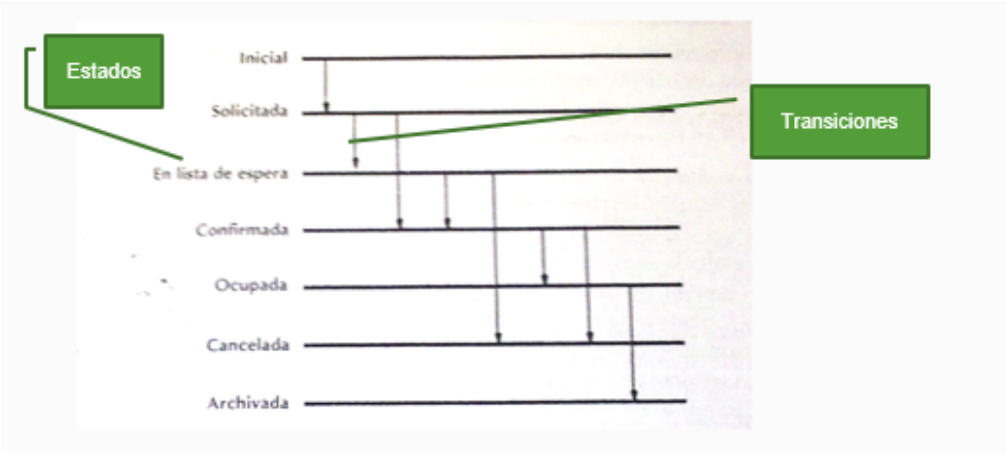
- Cuando se está en el estado A (esperando) y sucede el triángulo, se cambia el estado a B (mostrando).
- Cuando se está en B (mostrando) y sucede el cuadrado o el stop, se cambia a A (esperando) o C (pausa) respectivamente.
- Cuando se está en C (pausa) y sucede el triángulo o el cuadrado, se cambia a B (mostrando) y A (esperando) respectivamente.

Definición formal → Formalmente, un ***autómata finito*** (AF) puede ser descrito como una 5-tupla (S, Σ, T, s, A) donde:

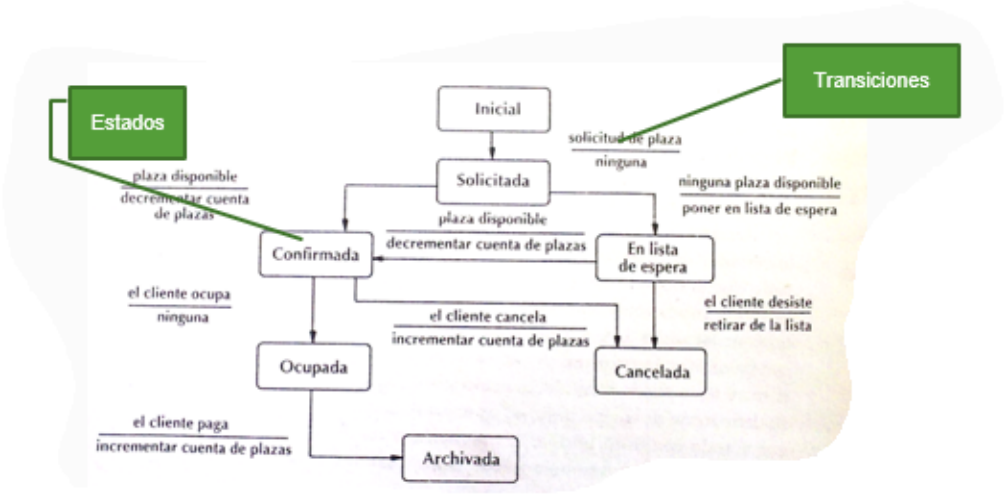
- Σ (sigma) es un alfabeto → Es el conjunto de símbolos de entrada que la máquina puede leer, puede ser el conjunto de eventos que el sistema recibe.
- S un conjunto de estados → Todos los estados posibles en los que puede estar el sistema.
- T es la función de transición → Define cómo pasa de un estado a otro según la entrada recibida.
- s es el estado inicial → El estado en el que arranca el sistema (uno solo).

- A es un conjunto de estados de aceptación o finales → Es un subconjunto de S, son los estados donde “termina” una ejecución válida. Pueden representar estados de éxito o condiciones donde el sistema cumple una meta.

Representación en Gráfico de Persiana:



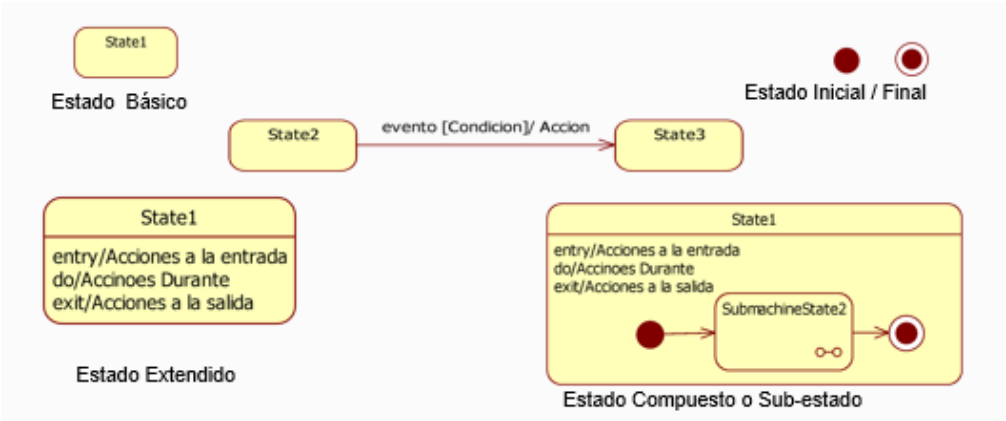
Representación en Máquina de Estado Finito:



Representación en Diagrama de Transición y estados:

Máquinas de Estado Finito:

- Notación **UML *Diagrama de Transición y Estado (DTE)*** → La MEF es la definición formal (matemática). El DTE es la representación gráfica (diagramada). Ambos describen el mismo sistema, pero con distintos fines (MEF = precisión, análisis formal; DTE → comprensión, comunicación, documentación). La MEF es la representación formal y conceptual, a veces se puede dibujar como un esquema abstracto pero está pensado para definir matemáticamente cómo funciona el sistema, sin tanto detalle visual. El DTE es la representación gráfica de esa misma máquina, se dibuja con recuadros de estado y flechas de transición, incluye marcas para el estado inicial y finales, y está pensado para ser leído fácilmente.



- ***Evento***: Es un ***suceso significativo*** que debe tenerse en cuenta, que ***influye en el comportamiento y evolución del sistema***. Tiene lugar en un *punto del tiempo* y *carece de duración* respecto a la granularidad temporal del sistema.

No tiene sentido preguntarse por lo que sucede mientras está teniendo lugar el evento.

- ***Transición***: Las transiciones se producen como ***consecuencia de eventos***. Pueden o no tener un *procesamiento asociado*.

- Evento → ***Obligatorio***. Ocurre y dispara un cambio.
- Condición → ***Opcional***, depende del problema, *puede haber transiciones sin condiciones*. Son verificaciones que habilitan o bloquean transiciones.
- Acción → ***Opcional***, puede haber *transiciones sin acciones*. Es una operación atómica, que no se puede interrumpir por un evento y que se ejecuta hasta su finalización. Es lo que se ejecuta en el cambio de un estado a otro.

Construcción de un DTE:

1. Identificar los ***estados***.
2. Si hay un ***estado complejo*** se puede explotar.
3. Identificar el ***estado inicial*** → Desde el estado inicial, se identifican los ***cambios de estado con flechas***.
4. Se analizan las ***condiciones y las acciones*** para pasar de un estado a otro.
5. Se verifica la ***consistencia***:
    - Se han *definido todos los estados*.
    - Se pueden *alcanzar todos los estados*.
    - Se pueden *salir de todos los estados*.
    - En cada estado, el sistema *responde a todas las condiciones posibles* (*normales y anormales*).

Ejemplo:

Se requiere modelar el comportamiento de un sistema de autenticación simple para una aplicación móvil. El sistema debe seguir las siguientes reglas:

Cuando se abre la aplicación por primera vez, el sistema muestra una "*Pantalla de Inicio*". Desde allí el usuario puede presionar el botón "*Ingresar*". Esto lo lleva al "*Formulario de Ingreso*".

Proceso de autenticación: en el "*Formulario de Ingreso*", si el usuario ingresa las credenciales correctas y presiona "*Autenticar*", el sistema activa la sesión del usuario. Si las credenciales son incorrectas, el sistema muestra un mensaje de error y permanece en el "*Formulario de Ingreso*".

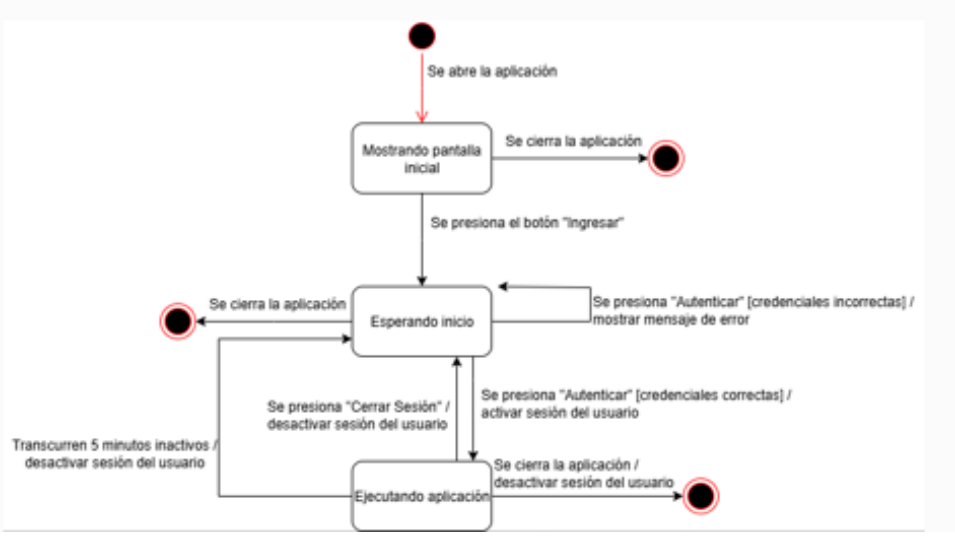
Una vez iniciada la sesión, el usuario puede presionar el botón "*Cerrar Sesión*", lo que lo devuelve a la "*Pantalla de Inicio*". Inactividad: después de 5 minutos de inactividad, la sesión del usuario expira automáticamente, y el sistema regresa a la "*Pantalla de Inicio*". Si la aplicación se cierra, el sistema finaliza, independientemente del estado en el que se encuentre.

Funciones:

- Inicialmente (al abrir la app) *visualiza la pantalla inicial*.
- Visualizar *formulario de ingreso*.
- Ingresar *datos de autenticación* (correctos o incorrectos).
- Presionar botón *cerrar sesión*.
- Dejar la *aplicación inactiva* X tiempo.
- Finaliza si la *aplicación se cierra*.

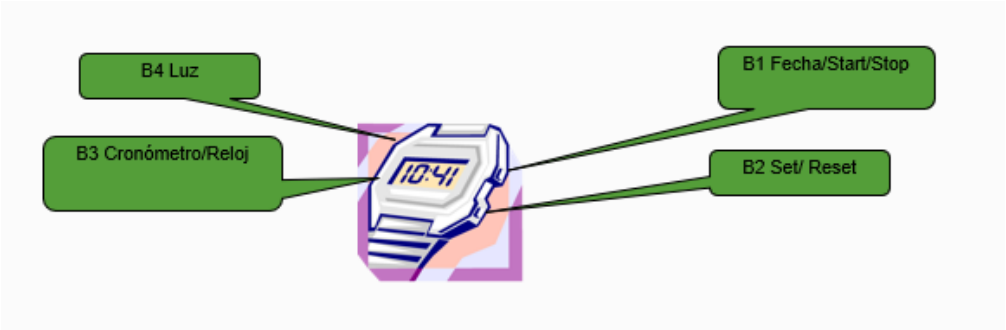
Debemos:

1. Identificar los estados.
2. Si hay un estado complejo, explotarlo.
3. Identificar el estado inicial.
4. Analizar las condiciones y las acciones para pasar de un estado a otro.
5. Verificar la consistencia.



Otro ejemplo:

Reloj Cronómetro → El reloj posee una pantalla y 4 botones:



Funciones:

- Inicialmente (al colocar la pila) visualiza la ***hora prefijada***.
- Visualizar la ***hora***.
- Visualizar la ***fecha***.
- Modificar ***Hora y Fecha***.
- ***Encender la Luz*** por 5 seg.
- ***Iniciar / Detener / Resetear*** Cronómetro.
- ***Deja de funcionar*** al finalizarse la pila.

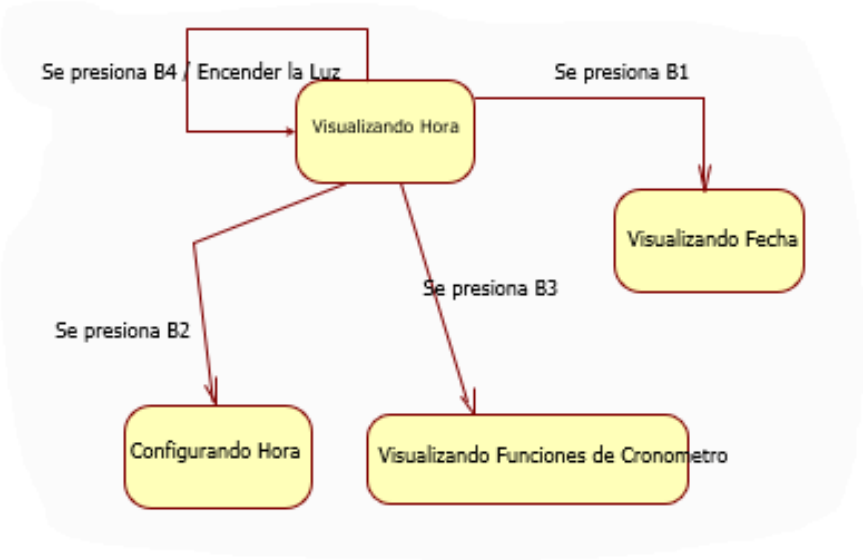
1. *Identificar los estados*:
    - Visualizando hora.
    - Visualizando fecha.
    - Visualizando funciones cronometro.
    - Cronometrando.
    - Configurando hora y fecha.
2. *Identificar estados complejos*:
    - No es necesario.
3. *Estado inicial*:
    - En este caso, el sistema inicia al colocarse la pila y pasaría al estado visualizando hora.
4. *Visualizando hora*

    → Se presiona B1 Visualiza la fecha.

    → Se presiona B2 Modificar la hora y fecha.

    → Se presiona B3 Visualiza el cronometro.

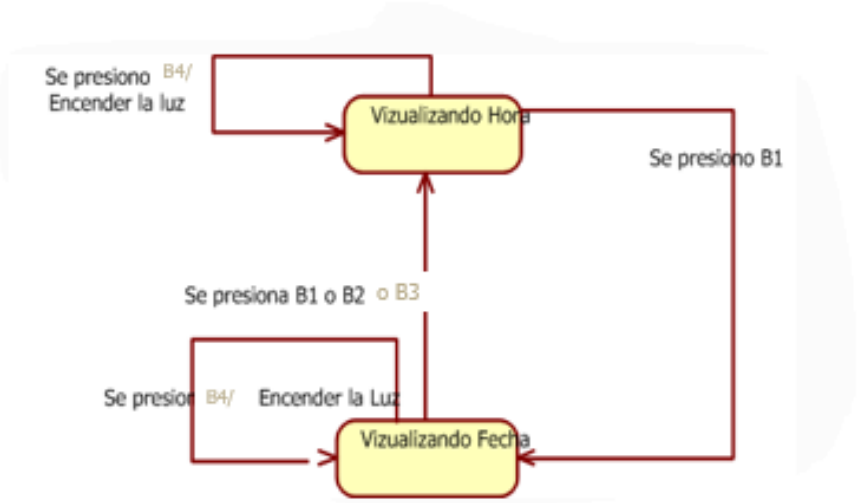
    → Se presiona B4 Enciende la luz.



4. *Visualizando fecha*

   → Estando en el estado Visualizando fecha, presionando B1 o B2 o B3 vuelve a visualizar la hora.

   → En cualquier momento se puede encender la luz con el botón B4.



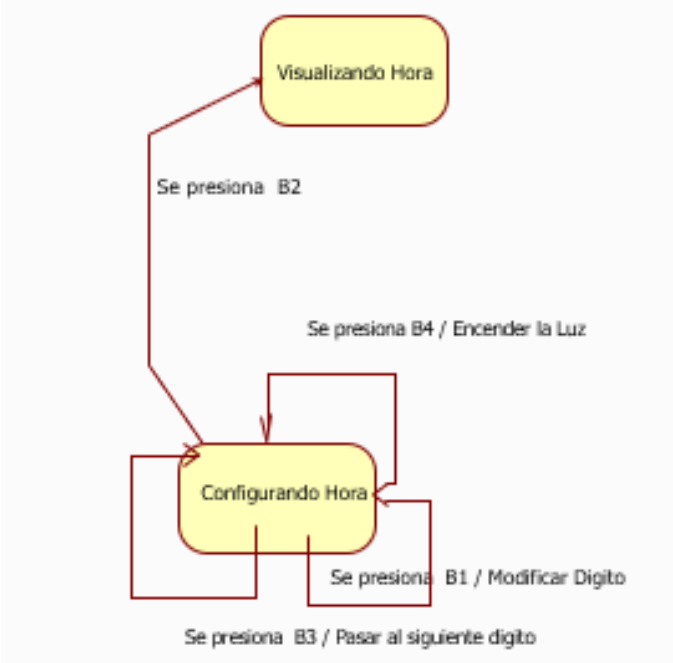
4. *Configurando Hora y Fecha*

   → Se presiona B1 modifico el digito.

   → Se presiona B2 vuelve a visualizar la hora.

   → Se presiona B3 modifico el digito a modificar (Hora, minuto, segundo, día, mes).

   → Se presiona B4 enciende la luz.



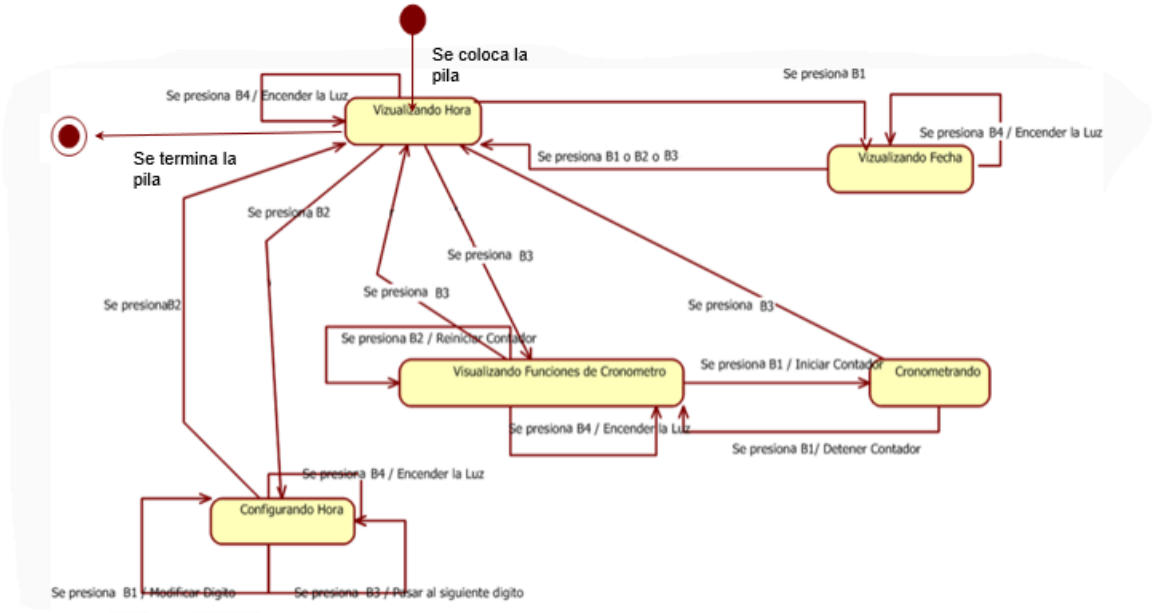
4. *Continuar con todos los estados*.
5. *Se verifica la consistencia*:

   → Se han definido todos los estados.

   → Se pueden alcanzar todos los estados.

   → Se pueden salir de todos los estados.

   → En cada estado, el sistema responde a todas las condiciones posibles (normales y anormales).



Ventajas del DTE:

- ***Mejora la comprensión***: Ofrece un mapa visual que hace fácil entender cómo un sistema reacciona a diferentes eventos.
- ***Previene errores***: Ayuda a encontrar fallos de lógica y caminos que no deberían existir, todo esto en la etapa de diseño, lo que ahorra tiempo y esfuerzo en la codificación.
- ***Valida el diseño***: Confirma que todos los requisitos del sistema han sido considerados y que el comportamiento es coherente.
- ***Facilita las pruebas***: Proporciona un plan para crear pruebas, asegurando que se validan todas las interacciones posibles del sistema.

Inconvenientes:

- ***Complejidad***: Pueden volverse muy complejos y difíciles de manejar en sistemas con una gran cantidad de estados y transiciones.
- ***Enfoque en un solo objeto***: Un DTE típicamente modela el comportamiento de un único objeto o componente. Para entender el sistema completo, se necesita una colección de DTEs.

- ***No modela la concurrencia*:** Los DTEs básicos no son ideales para modelar sistemas concurrentes, donde varios estados pueden cambiar al mismo tiempo.

¿Cuándo usar DTE?

- ***Sistemas de tiempo real***: Cuando el software está impulsado por eventos externos, como un sistema de control para una máquina o un dispositivo.
- ***Modelado del comportamiento de la interfaz de usuario*:** Para aplicaciones con interfaces de usuario complejas, un DTE puede mostrar cómo la pantalla o un componente de la interfaz cambia en respuesta a las acciones del usuario.
- ***Sistemas con un número finito de estados***: Cuando el componente de software puede estar en un número limitado de estados bien definidos (por ejemplo, una válvula que puede estar "abierta" o "cerrada").
- ***Diseño de algoritmos de control*:** En el diseño de software para sistemas de control, donde el estado actual del sistema determina su próximo comportamiento.