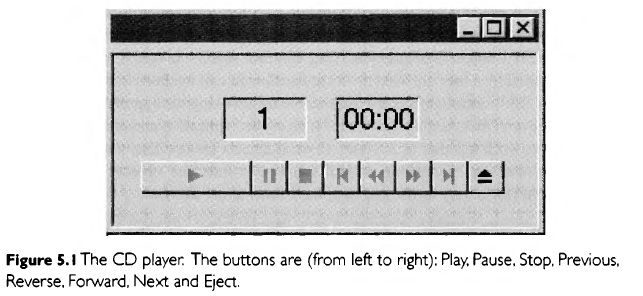
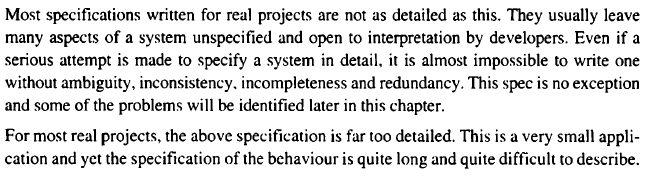
Statecharts

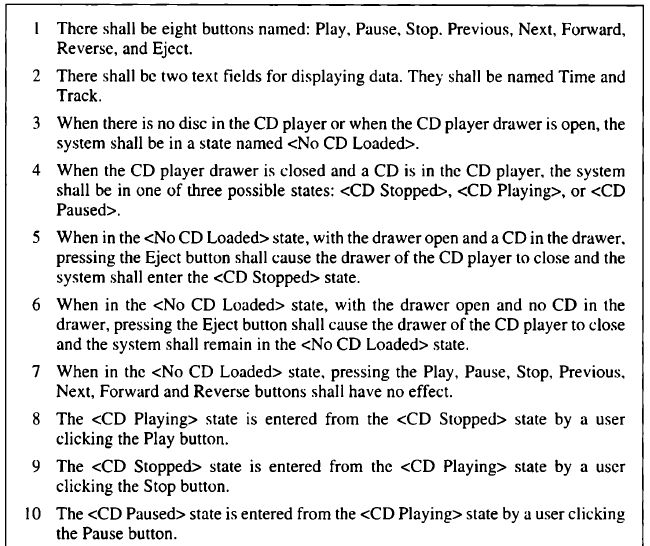
La notación de cartas de estado resuelve los serios problemas asociados con los diagramas de transición de estado. En esta sección se buscará mostrar qué tan limitados son los diagramas de transición de estados. Para ello, se proporcionarán un diagrama de transición de estados y una carta de estados de una aplicación que controlaría un reproductor de CD.

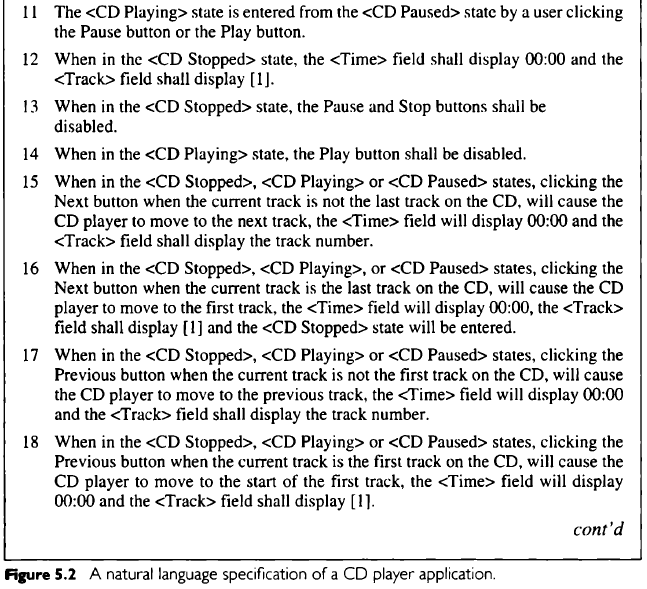
Una especificación en lenguaje natural de una interface de usuario de un reproductor de CD

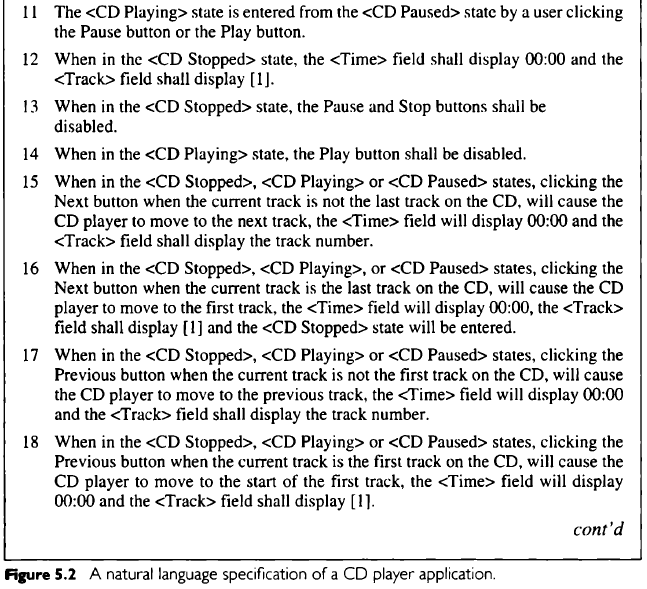
Aunque el comportamiento de una aplicación reproductor de CD pueda ser intuitivo, describir el comportamiento de tal aplicación en lenguaje natural resulta ser sorprendentemente difícil. Hay muchas reglas relacionadas a cuándo los botones están habilitados o deshabilidados, lo que se despliega en los campos de texto de la ventana y lo que sucede cuando el usuario proporciona eventos a la aplicación. Una ilustración simple de un reproductor de CD se da en la Figura 5.1 y una especificación del reproductor en lenguaje natural se da en la Figura 5.2.

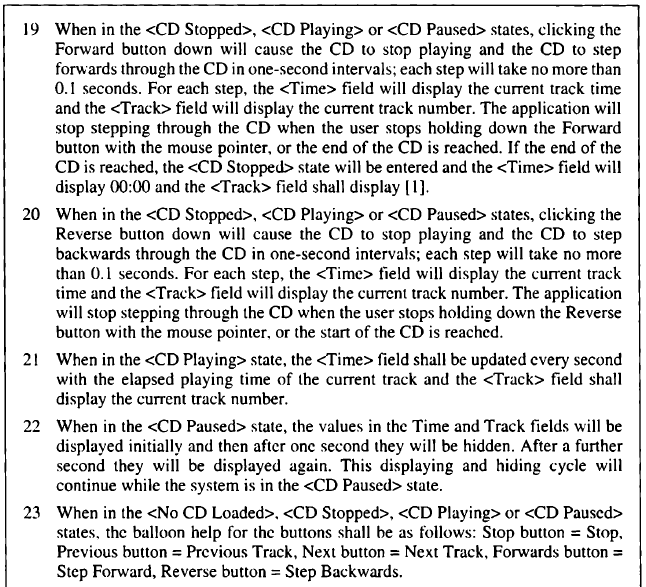


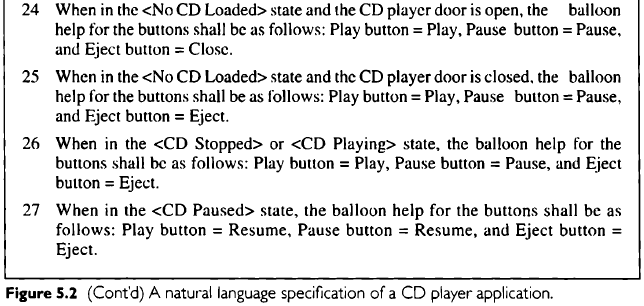












Típicamente, las descripciones detalladas de interfaces de usuario no se hacen porque mantenerlas se vuelve más difícil que mantener el software mismo. Es muy difícil modificar repetidamente una especificación en lenguaje natural sin introducir inconsistencias.

Desafortunadamente, cuando se escribe una especificación menos detallada, muchos aspectos del comportamiento de la interface de usuario se dejan a criterio del desarrollador de la aplicación. Para una aplicación pequeña, esto podría no ser importante si el desarrollador es sensible a las necesidades de los usuarios y entiende las tareas que ellos realizarán con la aplicación. Sin embargo, en un sistema grande con varios desarrolladores trabajando sobre diferentes partes del sistema, mantener la consistencia entre los módulos de la interface de usuario es un problema muy significativo. Es fácil producir una interface de usuario que funciona en formas diferentes porque diferentes desarrolladores toman diferentes decisiones. Si se hacen buenas pruebas, se resolverán muchas de las inconsistencias, pero esta es una forma cara de enfrentar el problema. Un problema adicional con las interfaces de usuario grandes es que muy pocos de lso desarrolladores habrán conocido a los usuarios y entendido las tareas que ellos estarán realizando con el sistema. Sin tal conocimiento será difícil para los desarrolladores tomar buenas decisiones acerca del diseño de la interface de usuario.

En resumen, escribir una especificación breve en lenguaje natural obliga a los desarrolladores a tomar decisiones acerca de muchos aspectos del comportamiento de la interface de usuario. Y escribir una especificación detallada en lenguaje natural, con la cual se intente especificar cada aspecto de la interface de usuario, casi seguro resultará en que la especificación será abandonada completamente porque consume demasiado tiempo y es difícil mantenerla actualizada.

Sin una especificación, hacer pruebas se reduce a una inspección subjetiva de la interface de usuario, porque no hay algo que defina lo que se supone que la aplicación debe hacer. Además, almacenar todo el conocimiento detallado de una aplicación en las cabezas del equipo de desarrollo actual es un enfoque de muy corto plazo y que está condenado a fallar en el largo plazo. El mantenimiento de largo plazo de tal sistema será extremadamente difícil si los miembros del equipo original ya no trabajan en el proyecto.

¡La solución a este problema no es escribir una especificación nada más! Permítame poner mi punto de vista en claro. No estoy en contra de las especificaciones de las interfaces de usuario. Solo estoy en contra de las especificaciones escritas en lenguajes naturales. Uno de los problemas principales con el uso de lenguajes naturales para especificaciones es su ambigüedad inherente. La semántica de cada palabra no está bien definida y cuando las palabras son combinadas en sentencias y párrafos, la ambigüedad se puede hacer mayor. Aun cuando se produzcan descripciones precisas y exactas, con frecuencia son muy largas y difíciles de entender.

La mejor forma de capturar el comportamiento preciso de una interface de usuario es producir un modelo del comportamiento de la interface de usuario en un lenguaje gráfico que tenga una semántica bien definida. La ventaja de este enfoque es que muchos de los problemas con las especificaciones en lenguaje natural son evitados y la especificación misma puede ser convertida directamente en objetos de la capa de control, sin los problemas usuales asociados con requerimientos de traducción en un diseño y entonces un diseño a código.

Un lenguaje de especificación para la capa de control

Máquinas de estado finito

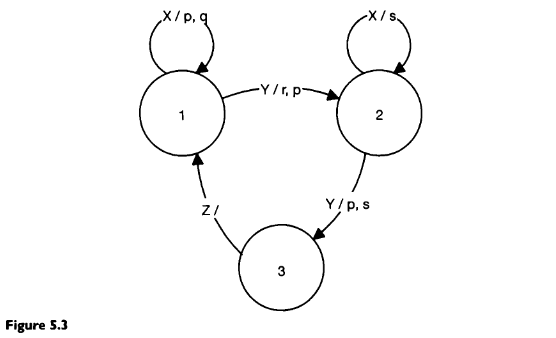
Una máquina de estado finito es un modelo de un sistema el cual solamente puede estar en uno de un número dado de posibles estados en cualquier momento en el tiempo. Una máquina de estado finito acepta eventos de entrada (o estímulos) que causan una salida (o acciones) y posiblemente un cambio en el estado. Ambos, las acciones de salida y el siguiente estado de la máquina son funciones puras del evento de entrada y el estado actual. Entonces



Hay muchas notaciones usadas comúnmente para representar máquinas de estado finito. Dos de tales representaciones son: los diagramas de transición de estado (STDs) y las matrices de transición de estado (STMs).

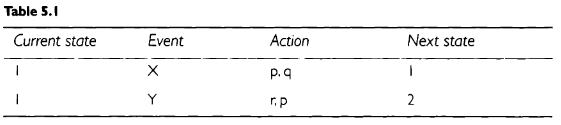
Diagramas de transición de estado

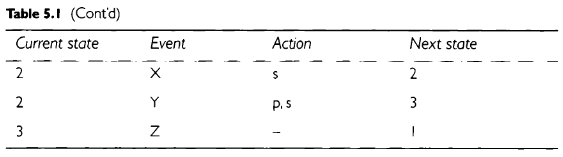
En los STDs, se usan círculos para denotar estados y se utilizan flechas que concetan dos estados para denotar una transición potencial de un estado a otro (véase Figura 5.3)

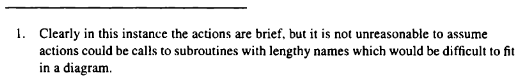


A cada flecha evento se le da una etiqueta formada con dos partes. La primera parte es el evento que causará que la transición ocurra entre los dos estados. La segunda parte de la etiqueta está separada del evento por una diagonal (/) y lista las acciones que ocurren en respuesta al evento.

Frecuentemente no es práctico incluir las acciones asociadas con una transición de estado sobre la flecha evento porque podrían ser varias acciones con descripciones largas. En tales casos, el diagrama puede ser dibujado sin las etiquetas de acción y se puede usar una tabla para suplementar el diagrama y proporcionar los detalles completos$^{1}$ (véase Tabla 5.1).

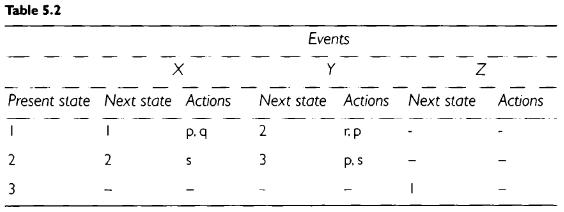






Matrices de transición de estado

Una STM es una tabla en la cual cada fila es etiquetada con un estado y cada columna es etiquetada con un evento de entrada (véase Tabla 5.2). A cada estado posible del sistema se le asigna una fila y a cada evento de entrada posible se le asigna una columna. Una combinación de un estado y un evento define el siguiente estado y las acciones del sistema requeridas y estos por lo tanto aparecen en la intersección de una columna y una fila.

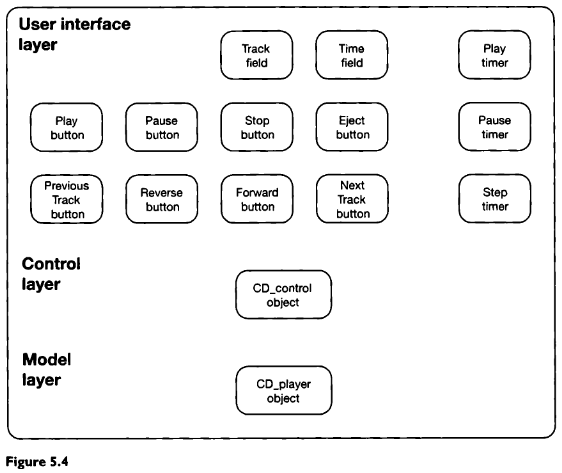


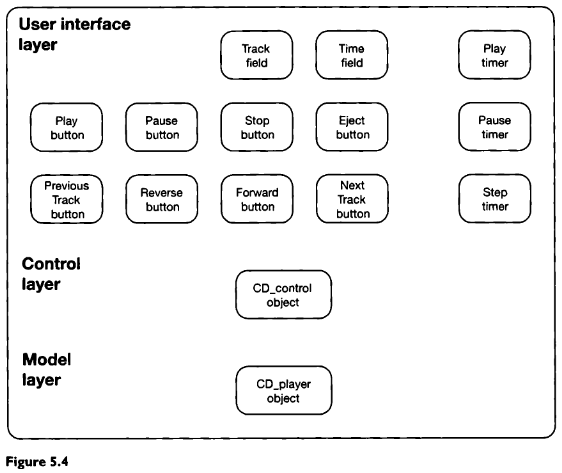
Modelos de Mearly y Moore de máquinas de estado finito

En las secciones precedentes, se han discutido los modelos de Mearly de máquinas de estado finito. En este tipo de modelo, las acciones están asociadas con la transición de un estado a otro. Un modelo alternativo de una máquina de estado finito es un modelo de Moore, en el cual las acciones están asociadas con el estado en lugar de la transición entre estados. En un modelo de Moore, una transición es etiquetada solamente con un evento y el estado es etiquetado con el nombre del estado y las acciones asociadas con ese estado. Los modelos de Mearly y de Moore son idénticos con respecto a su poder expresivo y existen algoritmos simples para convertir uno en el otro.

Máquina de estado finito para el reproductor de CD

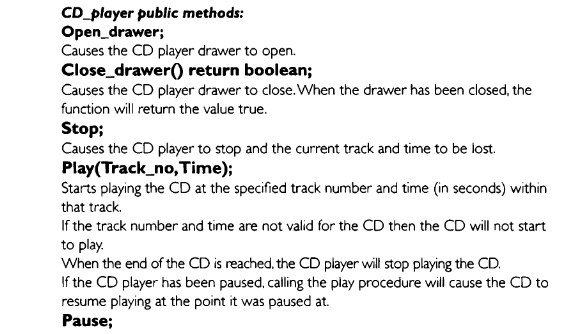
La aplicación reproductor de CD introducida previamente tiene la siguiente arquitectura UCM (véase Figura 5.4)

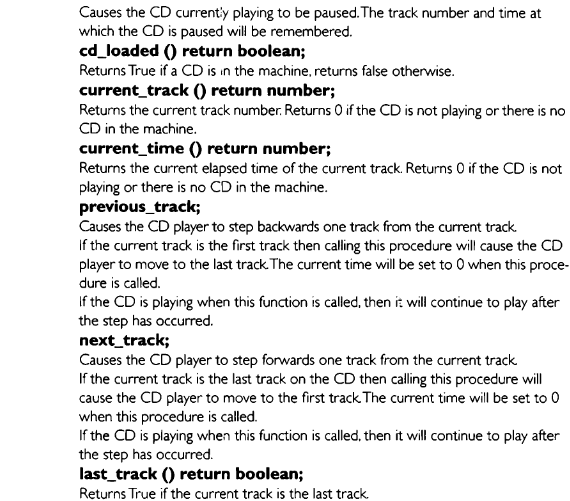


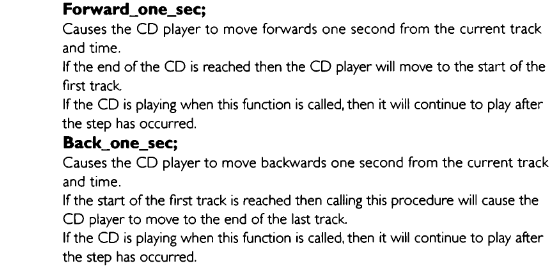
REF: 48/265

La capa de control contiene un objeto y este coordinará el comportamiento de los objetos de la interface de usuario de tal manera que se comporten como el reproductor de CD especificado previamente.

La capa de modelo contiene un objeto que controla el reproductor de CD real. El objeto proporciona un conjunto de métodos los cuales pueden ser llamados por el objeto de control. Esos métodos permiten que el reproductor de CD sea controlado y permiten que se obtenga información acerca del reproductor de CD. A continuación, se da una descripción de los métodos públicos del objeto CD\_player:





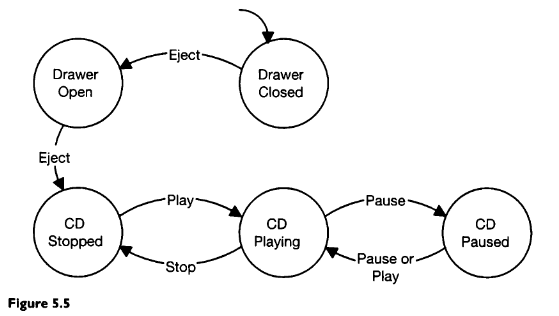


El propósito de esta sección es indagar si es posible usar una máquina de estado finito para controlar el comportamiento de los objetos de la interface de usuario que constituye un reproductor de CD. Dado que la aplicación está controlando un dispositivo eléctrico, y que el uso de máquinas de estado está esparcido en la industria eléctrica, enfatizo que la máquina de estado está siendo usada para controlar los objetos de la interface de usuario y no el reproductor de CD mismo. El control del reproductor de CD estará escondido en el objeto cd\_player y no está contemplado en el libro [Horrocks]. Aquí se identificarán las ventajas y desventajas de usar una máquina de estado finito para controlar objetos de la interface de usuario.

Como se mencionó antes, una máquina de estado finito puede ser representada en varias formas diferentes. La elección de la representación es algo completamente personal porque todas las representaciones son equivalentes y no ofrecen diferencia alguna en poder expresivo. En el texto [Horrocks], el autor prefiere usar diagramas de transición de estado complementados con tablas evento-acción. Los requerimientos especificados en la figura 5.2 serán usados como base para construir el diagrama de transición de estado.

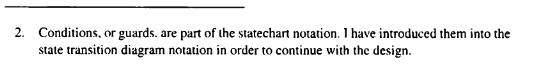
Los requerimientos 1 y 2 identifican los objetos de la interface de usuario que son visibles para los usuarios. Ellos no especifican comportamiento alguno que necesite ser modelado en un diagrama de transición de estado.

El STD básico mostrado en la Figura 5.5 se puede dibujar basándose en los requerimientos 3 a 14. Cuando la aplicación empieza, la flecha inicial indica que la aplicación empieza con la bandeja (drawer) del reproductor de CD cerrada. (Nota: LA bandeja del CD podría estar abierta cuando se inicie la plaicación, por lo tanto, una acción asociada con el evento de inicio debe causar que la bandeja se cierre.)

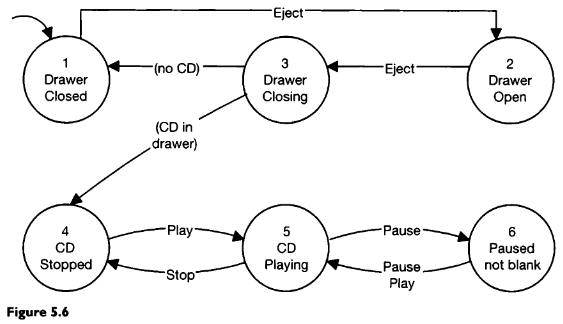


Un usuario puede abrir la bandeja del CD dando clic en el botón Eject y entonces, después de colocar un Cd en la bandeja de CD, cerrar la bandeja dando clic en el botón Eject. Esto causará que la aplicación entre al estado CD Stopped. Un usuario puede empezar a reproducir el CD en la máquina dando clic en el botón Play y esto causará que la aplicación entre al estado CD Playing. El CD puede ser pausado por un usuario dando clic en el botón Pause (causando una transición al estado CD Paused) y se puede hacer que continúe la reproducción del CD dando clic en el botón Pause o en el botón Play (causando una transición del estado CD Paused al estado CD Playing). Finalmente, el CD puede ser detenido dando clic en el botón Stop causando una transición al estado CD Stopped.

Este modelo de la aplicación reproductor de CD asume que un usuario pone un CD en la máquina antes de dar clic en el botón Eject para cerrar la bandeja de CD. Esta limitación del modelo inmediatamente demuestra una limitación significativa de las máquinas de estado finito. Para determinar el siguiente estado después de que un usuario da clic en el botón Eject, la aplicación necesita revisar si hay un CD en la máquina. Entonces, para que el reproductor de CD sea modelado exactamente, el siguiente estado no es simplemente una función del estado actual y el evento proporcionado por el usuario. En lugar de eso, el siguiente estado debe ser una función del estado actual, el evento proporcionado por el usuario y alguna condición que debe ser evaluada a true para que la transición ocurra. Esto es, se requiere una extensión simple a la notación del diagrama de transición de estado básica la cual permita que se asocien condiciones con los eventos$^{2}$.

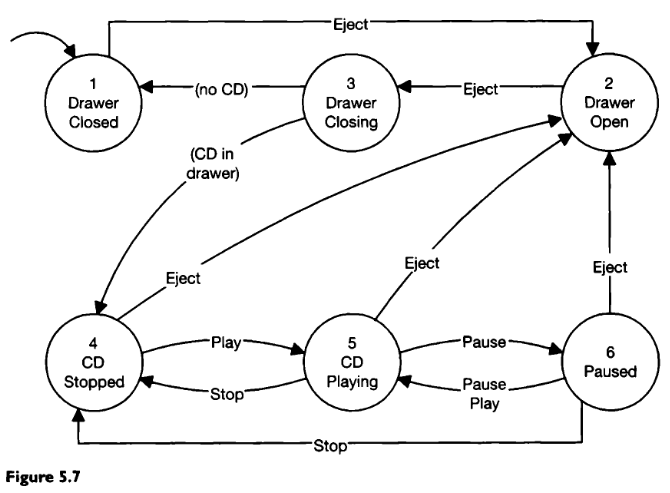


Cuando ocurre un evento, si la condición asociada se evalúa a true, entonces la transición y las acciones correspondientes serán ejecutadas. Usando condiciones en los diagramas de transición de estado, se obtiene un modelo más exacto de la aplicación reproductor de CD (véase Figura 5.6). Note que las condiciones son puestas entre paréntesis para distinguirlas de los eventos.



El diagrama ahora contiene un estado adicional llamado Drawer Closing. Cuando la bandeja de CD está abierta, la aplicación está en el estado 2 (Drawer Open). La bandeja de CD puede entonces ser cerrada dando clic en el botón Eject y la aplicación entra en el estado 3 mientras esta acción está ocurriendo. Después de que la bandeja de CD de ha cerrado, la aplicación se moverá del estado 3 al estado 1 si no hay un CD en la bandeja o del estado 3 al estado 4 si hay un CD en la bandeja. Se necesita el estado Drawer Closing porque el siguiente estado no se puede determinar hasta que la bandeja de CD está cerrada y la aplicación ha detectado la presencia o ausencia de un CD. En esta instancia, las transicione de 3 a 1 y de 3 a 4 son determinadas solamente por condiciones. Sin embargo, es más típico que una transición dependa de ambas cosas, un evento y una condición.

El diagrama de estado mostrado en la Figura 5.6 resalta algunos de los problemas de la especificación en lenguaje natural de la aplicación. Primeramente, no hay algo que especifique el comportamiento de la aplicación cuando se da clic en el botón Eject después de que un CD está cargado. Supondremos que se le puede dar clic al botón Eject cuando el CD está detenido, reproduciéndose o en pausa, y en los tres casos, el evento causará que la bandeja de CD se abra. En segundo lugar, no hay algo que especifique qué sucede cuando un CD ha sido pausado y un usuario da clic en el botón Stop. ¿La aplicación permanece en el estado pausado (estado 6)? o ¿entra al estado detenido (estado 4)? Supondremos que se entrará al estado detenido (estado 4). Capturando estos requerimientos, el diagrama queda como en la figura 5.7.

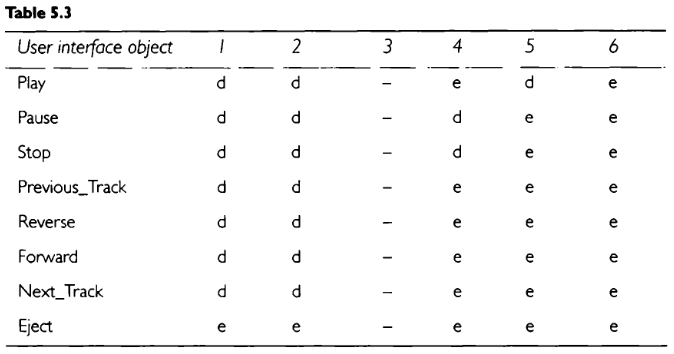


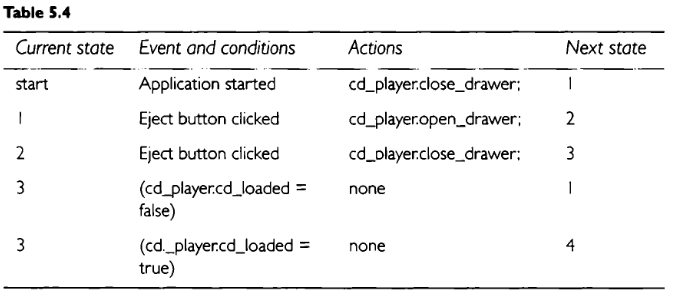
La tabla 5.3 define los atributos de los objetos de la interface de usuario en los seis estados del diagrama en la Figura 5.7. Note que en la tabla 5.3 ‘d’ representa deshabilitado (disabled) lo cual significa que el objeto se debe mostrar en color gris claro y no debe estar disponible para uso por el usuario; ‘e’ representa habilitado lo cual significa que el objeto está disponible para que un usuario interactúe con él.

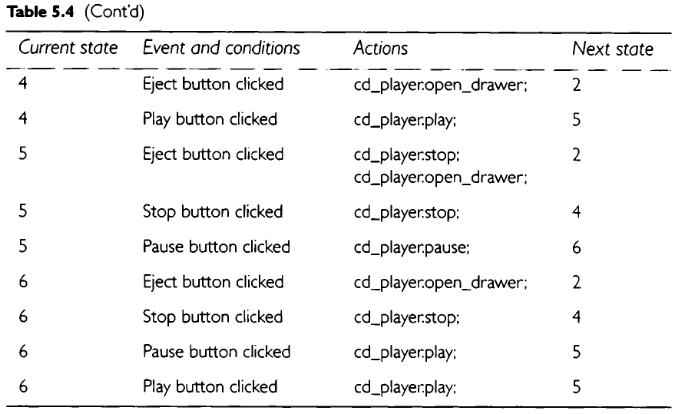
La tabla 5.4 define las acciones y las transiciones de estado que son ejecutadas en respuesta a los varios eventos y condiciones que pueden ocurrir.

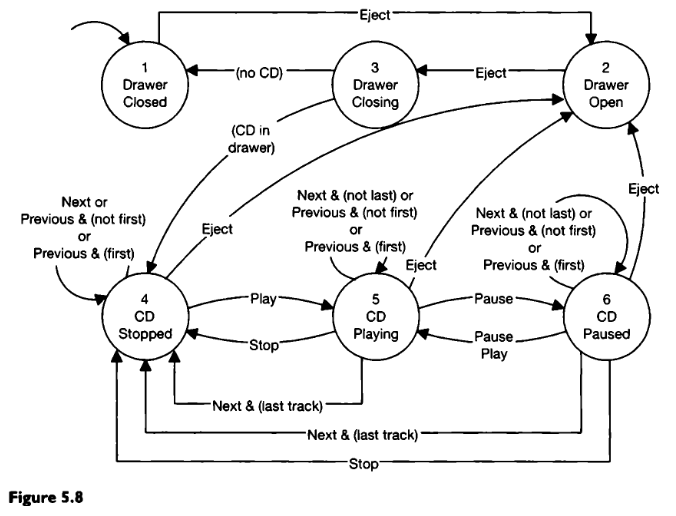
Los requerimientos 15 a 18 especifican el comportamiento de los botones Next Track y Previous Track. Esos botones permiten al usuario “caminar” a través de un CD, ya sea hacia atrás o hacia adelante una pista a la vez. El comportamiento de los botones cambia cuando se llega al principio o al final del CD. Específicamente, si el CD está en la última pista cuando se le da clic al botón Next Track, entonces, el CD dejará de reproducirse. Y si el CD está en la primera pista cuando se le da clic al botón Previous Track, entonces, el CD se moverá al inicio de la primera pista.

El diagrama de estados puede ser extendido para capturar este comportamiento (véase Figura 5.8).

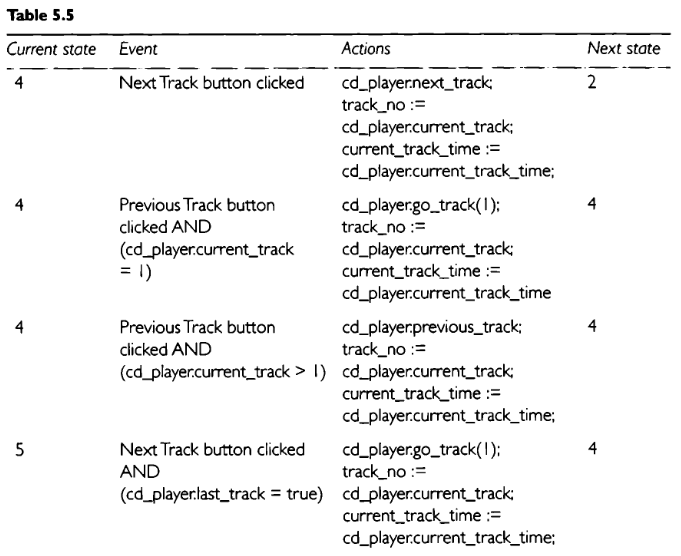


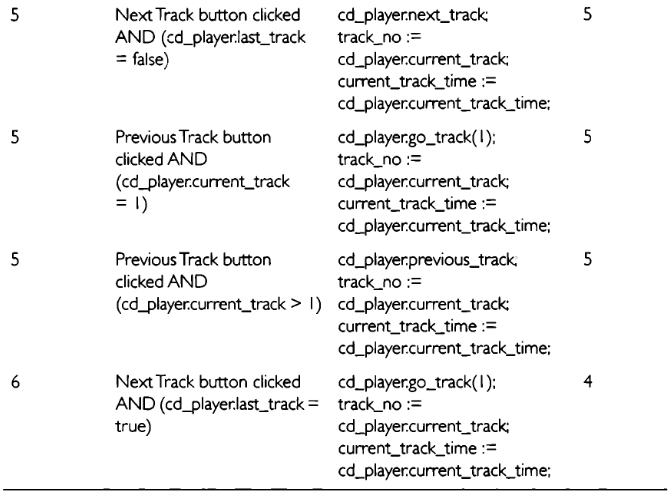


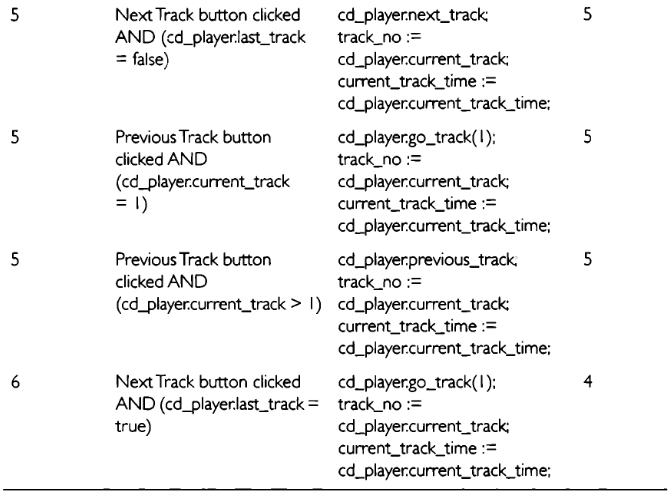


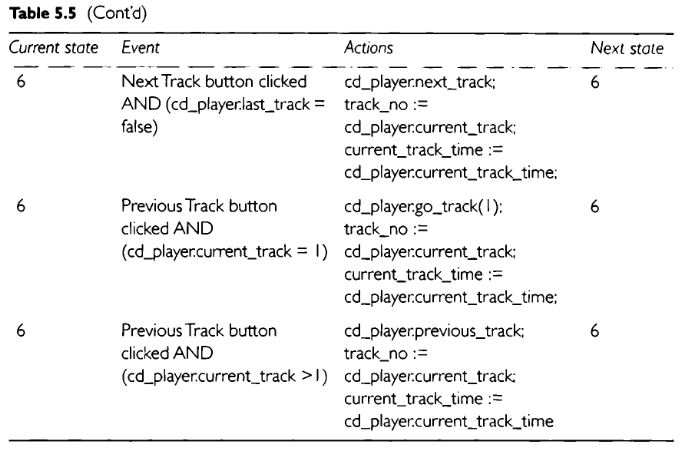


Se han agregado 11 eventos extra a este diagrama y los detalles de las transiciones extras están dados en la Tabla 5.5.





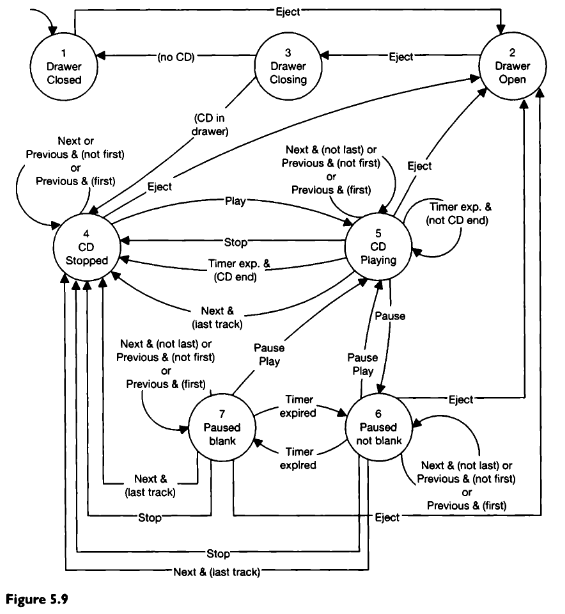




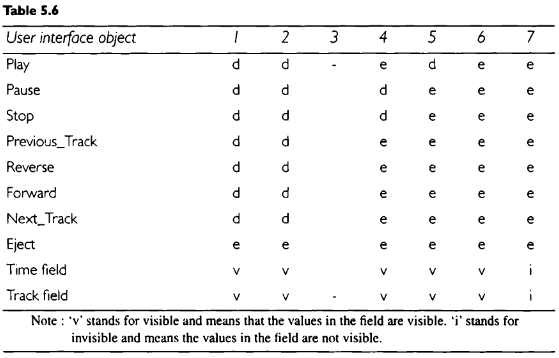
Nótese la duplicación de eventos. EL evento ‘previous & (not first)’ y el evento ‘previous & (first)’ están enganchados a los estados 4,5 y 6. Además, el evento ‘next & (not last)’ y el evento ‘next & (last)’ están enganchados a los eventos 5 y 6. De hecho, entre los eventos agregados hay solamente 5 eventos distintos, pero han sido duplicados porque pueden ocurrir en diferentes contextos. Esto no es bueno. El diagrama de transición de estados está empezando a verse demasiado complicado y desestructurado, y el reproductor de CD no ha sido completamente especificado aún.

Los frequerimientos 21 y 22 tienen que ver con el comportamiento del campo Time y el campo Track cuando un CD se está reproduciendo y cuando es pausado. Cuando unCD se está reproduciendo, el campo Track despliega el número de pista actual y el campo Time despliega la cantidad de tiempo que la pista actual ha sido reproducida. Nótese que la especificación no define qué sucede cuando un CD se está reproduciendo y se llega al final de la última pista. Hemos supuesto que se detendrá y hemos agregado la transición del estado 5 al estado 4. Nótese que podría haberse diseñado para que se volviera a reproducir la pista 1.

Cuando un CD está pausado, el campo Track despliega el número de pista y el campo Time despliega la cantidad de tiempo que la pista actual ha sido reproducida hasta el momento en que el CD fue pausado. Cuando un CD ha estado pausado por un segundo, los valores en los campos Track y Time son escondidos por un segundo antes de reaparecer otra vez un segundo más tarde. Miesntras un CD está pausado este comportamiento se repite continuamente. Para modelar este comportamiento en el diagrama de transición de estados se requiere un segundo estado CD Paused (véase Figura 5.9).



A través de la adición del estado 7, el número de flechas de eventos se ha incrementado en 12 y la estructura del diagrama se ha vuelto caótica y difícil de leer. Los estados 6 y 7 son casi idénticos –la única diferencia es si los valores en los campos Time y Track son visibles o no (véase Tabla 5.6). Sin embargo, debido a esta diferencia, todas las flechas de eventos que tocan el estado 6 deben ser duplicadas para el estado 7.



El diagrama en la Figura 5.9 se ha vuelto difícil de entender y será difícil hacerle cambios. En este punto, debe ser claro que el enfoque de transición de estados no es factible. Sin embargo, vale la pena presionar y completar la parte final del diseño solo para enfatizar lo complicado que puede resultar.

Los requerimientos 19 y 20 requieren

Pag. 41 (52/265) del libro [Horrocks]

REFERENCIAS:

[Horrocks] Ian Horrocks, “Constructing the User Inteface with Statecharts,” Addison Wesley, 1999.

[Samek] Miro Samek, “Practical UML Statecharts in C\_C++ Event-Driven Programming for Embbeded Systems,” Newnes Elsevier, 2009.

El libro Practical UML Statecharts in C\_C++ Event-Driven Programming for Embbeded Systems, (compartido por un(a) discente del grupo 3MM11 durante el semestre 2021-1 –Agosto 2020-Enero 2021) lo encontré en este link:

[http://caxapa.ru/thumbs/429264/Miro\_Samek\_Practical\_UML\_Statecharts\_in\_.pdf](https://meet.google.com/linkredirect?authuser=0&dest=http%3A%2F%2Fcaxapa.ru%2Fthumbs%2F429264%2FMiro_Samek_Practical_UML_Statecharts_in_.pdf)