Problema Resuelto – Statecharts

Maximiliano Cristiá

Ingeniería de Software 1
LCC – FCEIA – UNR
Rosario – Argentina
2024

1. Requerimientos

Vamos a especificar la dinámica de una interfaz gráfica de usuario más o menos simple pero representativa de las interfaces reales. Es decir, nos vamos a focalizar en los aspectos *reactivos* de la interfaz. En otras palabras, vamos a especificar cómo se debe comportar la interfaz cuando el usuario interactúa con ella y cuando llegan datos provenientes de fuentes de datos externas—por ejemplo datos provenientes de una conexión a una computadora remota.

La interfaz de usuario se compone de dos solapas como se muestra en las Figuras 1 y 2. La primera de ellas es la que el usuario utiliza para ingresar ciertos datos y la segunda es para visualizar gráficamente datos que se obtienen a partir de la entrada del usuario y de datos que la aplicación busca en servidores externos. La solapa de ingreso de datos consiste de un cierto número de cuadros de texto donde el usuario ingresa un dato. Cuando esto ocurre la aplicación se conecta con un servidor externo para buscar datos relacionados con la entrada del usuario. Como la búsqueda puede demorar la interfaz muestra un icono en forma de reloj de arena que debe ser rotado cada 4 segundos. Cuando los datos llegan a nuestra aplicación se los muestra en el cuadro de texto correspondiente. Como el servidor puede no responder, la aplicación le avisa esto a la interfaz en cuyo caso esta debe mostrar una *ventana modal*[1] con el mensaje de error correspondiente. A su vez las barras anaranjadas son barras de progreso[2] que indican la cantidad de kilobytes descargados de otros datos que se buscan en otros servidores externos. Para actualizar estas barras la interfaz debe consultar a la aplicación cada 1 segundo. Cuando el usuario posa el puntero sobre la barra se muestra la cantidad de kilobytes descargados (rectángulo negro en la línea intermedia) por a lo sumo 2 segundos si el usuario no mueve el puntero antes. El usuario puede detener la descarga de datos pulsando el botón de cancelación correspondiente.

Todas las búsquedas y descargas de datos se realizan en simultáneo desde el momento en que el usuario ingresa el dato correspondiente y hasta que la búsqueda o descarga termina.

[1] Wikipedia: Modal window
[2] Wikipedia: Progress bar

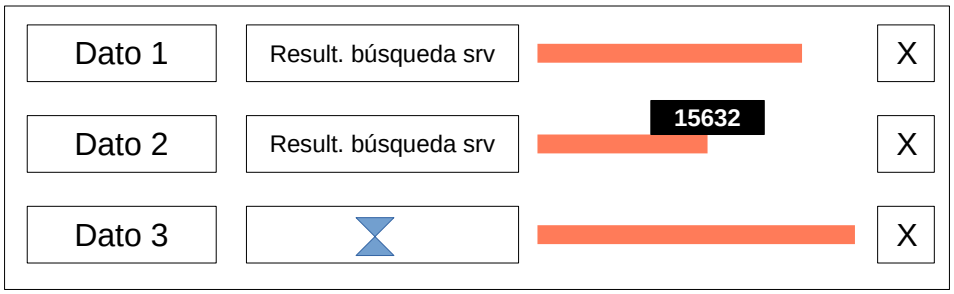


Figura 1: Solapa para ingreso de datos

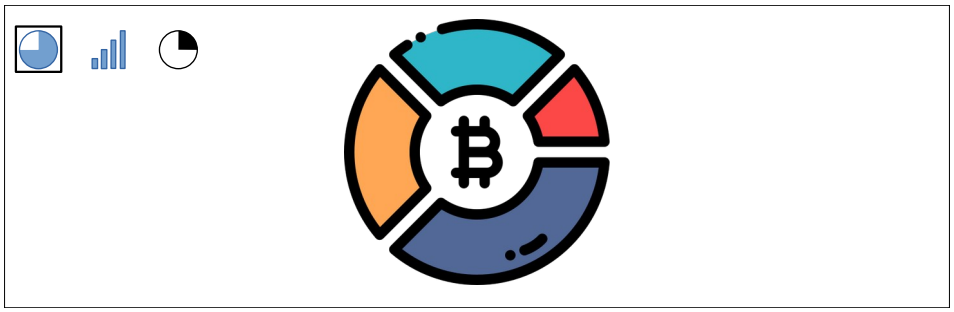


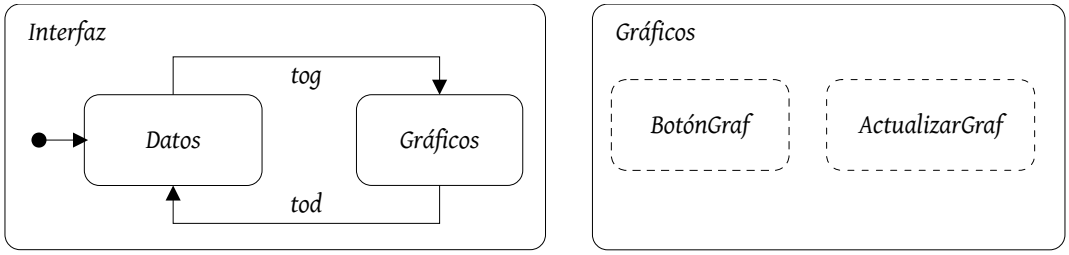
Figura 2: Solapa para visualización de gráficos

Con los datos descargados la aplicación genera ciertos gráficos que se muestran en la segunda solapa. En esta solapa el usuario puede seleccionar uno entre tres formatos de gráficos pulsando el botón correspondiente. El formato por defecto es gráfico de torta. Cuando se activa esta solapa se debe refrescar el gráfico. Si la solapa está activa se debe refrescar el gráfico cada 5 segundos.

La primera solapa que muestra la aplicación es la de ingreso de datos. Si el usuario vuelve de la solapa de gráficos a la de ingreso de datos se deben actualizar las barras de progreso y los cuadros de texto donde se muestran datos provenientes de los servidores externos. Si el usuario *vuelve* a la solapa de gráficos se debe mostrar el mismo formato de gráfico que había seleccionado el usuario anteriormente.

2. Especificación

Vamos a especificar cada solapa en un Statechart dedicado como se muestra en la Figura 3a. Claramente, la solapa para ingreso de datos es la que se activa por defecto al iniciar la interfaz. Los eventos usados en *Interfaz* se designan del siguiente modo:



(a) Especificación de alto nivel de la interfaz

(b) Especificación de la solapa *Gráficos*

Figura 3: Primeras especificaciones

- El usuario activa la solapa de gráficos ≈ *tog*
- El usuario activa la solapa para ingreso de datos ≈ *tod*

En la Figura 3b damos la especificación de alto nivel del estado *Gráficos* como dos máquinas concurrentes: *BotónGraf* se encarga de la selección del tipo de gráfico, mientras que *ActualizarGraf* se encarga de mantener el gráfico actualizado. La especificación de *BotónGraf* se da en la Figura 4. Las designaciones de los eventos que vemos en esta figura son las siguientes:

- El usuario pulsa el botón 'gráfico de torta' ≈ *to*
- El usuario pulsa el botón 'gráfico de barras' ≈ *ba*
- El usuario pulsa el botón 'gráfico de curvas' ≈ *cu*

Como podemos ver, usamos el conector de entrada por historia para mantener la elección del usuario cuando se vuelve del estado *Datos*. Al mismo tiempo decimos que el gráfico de torta es la elección por defecto. De esta forma, en la primera entrada a *Gráficos* el gráfico se presenta en forma de torta, pero si el usuario cambia la elección en la siguiente entrada a la solapa se mostrará el gráfico correspondiente. Usamos un estado-or anónimo para ahorrar una flecha[3].

En la Figura 5 tenemos la especificación de *ActualizarGraf* donde los eventos son los siguientes:

- La interfaz solicita a la aplicación la actualización de datos para el gráfico ≈ *ac*
- La aplicación envía los nuevos datos a la interfaz ≈ *da*
- La interfaz actualiza el gráfico ≈ *ag*

[3]También nos permite disponer de más puntos de conexión para ubicar de mejor manera la flecha de transición.

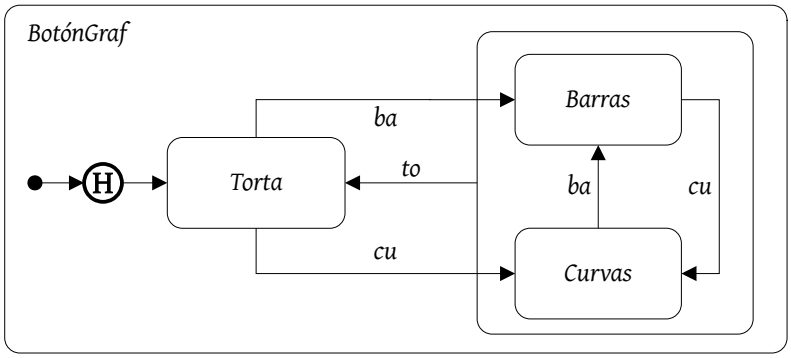


Figura 4: Especificación de *BotónGraf*

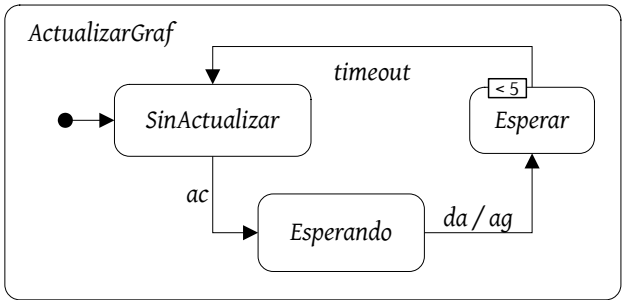


Figura 5: Especificación de *ActualizarGraf*

Dadas las dificultades de usar el grafismo propuesto por Harel para los estados temporizados, nosotros optamos por poner un pequeño cuadro de texto que muestra el límite de tiempo, cerca del vértice superior izquierdo del estado que queremos temporizar (*Eperar*). De esta forma, cuando se activa la solapa *Gráficos* la interfaz solicita a la aplicación nuevos datos y queda en espera de ellos; luego solicita la actualización cada 5 segundos como lo indican los requerimientos.

Como surge de nuestra especificación, cuando se activa la solapa *Gráficos* la interfaz hace dos cosas al mismo tiempo: permite que el usuario seleccione el tipo de gráfico (*BotónGraf*) y actualiza el gráfico (*ActualizarGraf*).

Ahora pasamos a la especificación de la solapa *Datos*. Si bien el usuario no puede ingresar datos en más de un cuadro de texto al mismo tiempo, una vez que ingresó datos en dos o más cuadros de texto la interfaz debe mantener actualizado los cuadros de texto que muestran los resultados de las búsquedas y las barras de progreso. En consecuencia, la especificación de *Datos* consiste de *N* máquinas *Fila* que ejecutan en paralelo como muestra la Figura 6. Los eventos que usamos allí son los siguientes:

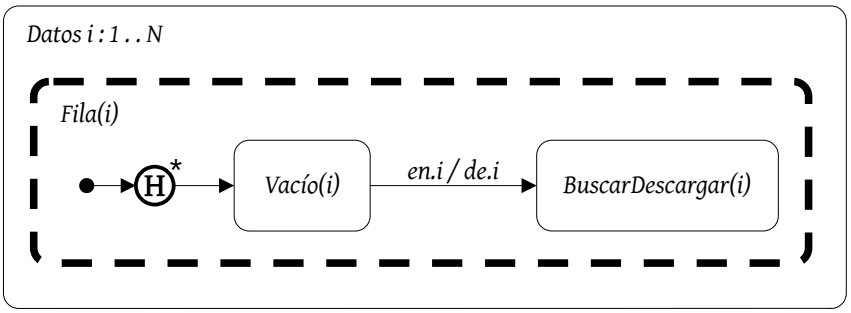


Figura 6: Especificación de alto nivel de *Datos*

- El usuario ingresa un dato en el cuadro de texto $i \approx en.i$
- La interfaz le pasa a la aplicación el dato ingresado por el usuario en el cuadro de texto $i \approx de.i$

Inicialmente cada cuadro de texto donde el usuario puede ingresar un dato está vacío. Cuando el usuario ingresa un dato en uno de ellos se activa el estado *BuscarDescargar* correspondiente. La entrada por historia sirve para el momento en que el usuario vuelve a activar la solapa *Datos*. Es decir, cuando la aplicación se inicia todas las filas van a estar en *Vacío*. Luego el usuario ingresa un dato y se activa *BuscarDescargar*, luego activa la solapa *Gráficos* pero la búsqueda y descarga continúan. Al volver a la solapa *Datos* la interfaz debe reiniciar en un estado donde las búsquedas y descargas sigan activas (y seguramente más avanzadas), lo que se logra volviendo al estado más reciente que será *BuscarDescargar* para aquellas filas en las que el usuario haya ingresado un dato. El asterisco junto a la entrada por historia implica que si se entra a *BuscarDescargar* se lo hará a los estados más internos de esta máquina.

La especificación de alto nivel de *BuscarDescargar* se muestra en la Figura 7a. Como podemos ver tenemos tres máquinas ejecutando en paralelo: *Buscar*, se encarga del comportamiento del cuadro de texto donde se verá la información proveniente del servidor externo o el icono del reloj si aun la información no ha llegado; *BarraProg*, modela la actualización de la barra de progreso; y *AccionesUsr*, describe cómo reacciona la interfaz cuando el usuario posa el puntero sobre la barra de progreso o cancela la descarga. Observar que por cuestiones físicas el usuario no puede hacer ambas cosas al mismo tiempo.

Comenzamos por la especificación de *Buscar* en la Figura 7b en la cual se usan los siguiente eventos:

- La interfaz dibuja el reloj del cuadro de texto $i \approx dr.i$
- La interfaz gira el reloj del cuadro de texto $i \approx gr.i$
- La aplicación pasa a la interfaz el dato recibido del servidor $i \approx ds.i$
- La aplicación comunica que el servidor i no responde $\approx sc.i$

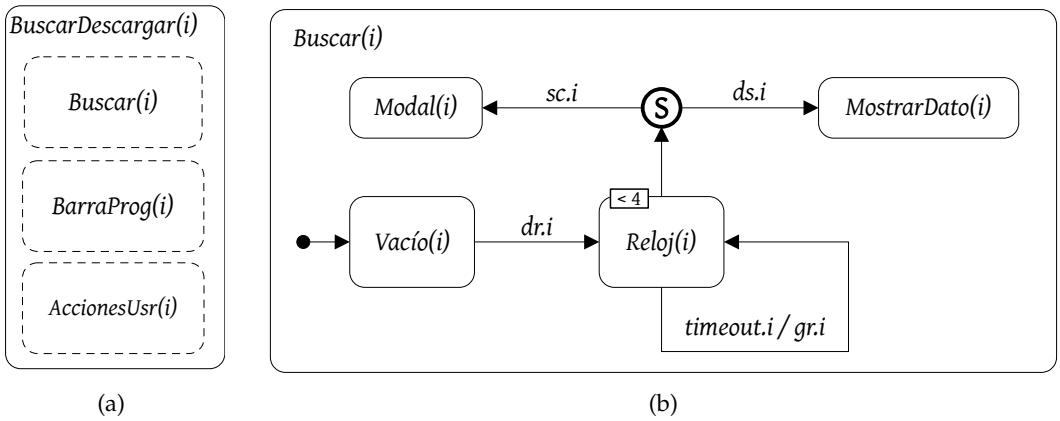


Figura 7: Especificación de alto nivel de *BuscarDescargar* y especificación de *Buscar*

Inicialmente el cuadro de texto *i* está vacío. Notar que hay dos estados *Vacío*: en *Fila* y en *Buscar*. Para evitar la ambigüedad podemos tomar el nombre completo de cada estado: *Fila*.*Vacío* y *Buscar*.*Vacío*. Entonces, la interfaz dibuja el reloj (*dr*) e inicia una espera de 4 segundos para girar el reloj (*gr*) en caso de que el dato no llegue a tiempo. Si lo hace (*ds*) se muestra el dato correspondiente. También puede ocurrir que la aplicación avise que el servidor está caído (*sc*) en cuyo caso se muestra la ventana modal que piden los requerimientos. Usamos un conector S para especificar las dos salidas posibles del estado *Reloj* más que nada por las limitaciones gráficas del editor.

BarraProg y *AccionesUsr* deben estar sincronizados porque el usuario puede interrumpir la descarga de datos de uno o más servidores pulsando el botón de cancelación (Figura 1). En este sentido designamos los siguientes eventos:

- La interfaz pregunta a la aplicación si debe extender la barra de progreso $i \approx ab.i$
- La aplicación comunica que se descargó un cantidad positiva de kb de datos del servidor *i* desde la última consulta $i \approx kb.i$
- La aplicación comunica que no aumentó la cantidad de kb descargados desde el servidor *i* desde la última consulta $\approx nkb.i$
- La interfaz extiende la barra de progreso $i \approx eb.i$
- El usuario cancela la descarga de datos del servidor $i \approx ca.i$
- La interfaz comunica a la aplicación que hay que cancelar la descarga de datos del servidor $i \approx ica.i$
- El usuario posa el puntero sobre la barra de progreso $i \approx mkb.i$
- La interfaz imprime la cantidad de kb descargados del servidor $i \approx ikb.i$

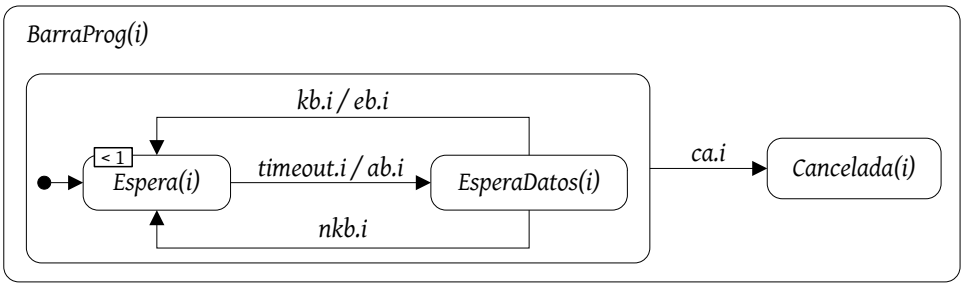


Figura 8: Especificación de *BarraProg*

- La interfaz borra la cantidad de kb descargados del servidor $i \approx dkb$

Con estos eventos podemos entender las especificaciones de *BarraProg* y *AccionesUsr* mostradas en la Figuras 8 y 9. Respecto a *BarraProg*, la interfaz comienza esperando 1 segundo para consultar a la aplicación si debe extender la barra de progreso (*ab*). La aplicación puede responder que hay más datos que la vez anterior (*kb*) o que la cantidad no aumentó (*nkb*). En el primer caso la interfaz extiende la barra (*eb*), mientras que en el segundo no hace nada; en ambos casos vuelve a esperar 1 segundo. El proceso de extender o no la barra puede ser cancelado en cualquier momento si el usuario cancela la descarga de datos (*ca*).

En cuanto a *AccionesUsr*, inicialmente el usuario no ejecuta ninguna de las dos acciones disponibles. Desde el estado *Ninguna* se puede salir porque el usuario cancela la descarga (*ca*), en cuyo caso la interfaz se lo comunica a la aplicación (dado que es esta quien puede efectivamente detener la descarga) y vuelve al estado inicial; o porque el usuario posa el puntero en la barra de progreso (*mkb*), en cuyo caso comienza el proceso para mostrar la cantidad de kilobytes descargados. Este proceso comienza con la interfaz consultando a la aplicación por el estado de la descarga (*ab*), luego la interfaz espera la respuesta de la aplicación (*kb*) e imprime la cantidad recibida (*ikb*). En ese momento la interfaz inicia una espera de 2 segundos de la cual sale por *timeout* o porque el usuario mueve el puntero fuera de la barra (*nmkb*); en cualquier caso borra el cartel y vuelve al estado inicial. Para especificar esta doble salida de *DatosKb* usamos la disyunción de eventos (or).

En la última página encontrarán la especificación de *Interfaz* expandida en un único Statechart. Creemos que es ilustrativo de la utilidad de la técnica de especificación jerárquica que alienta Statecharts.

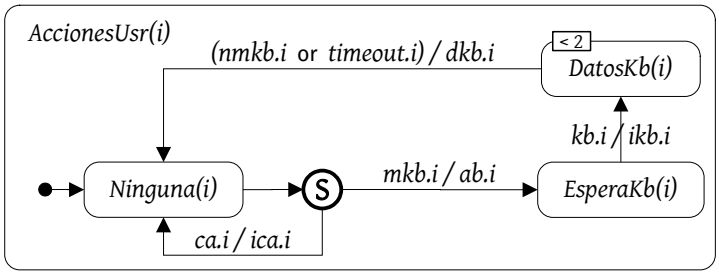
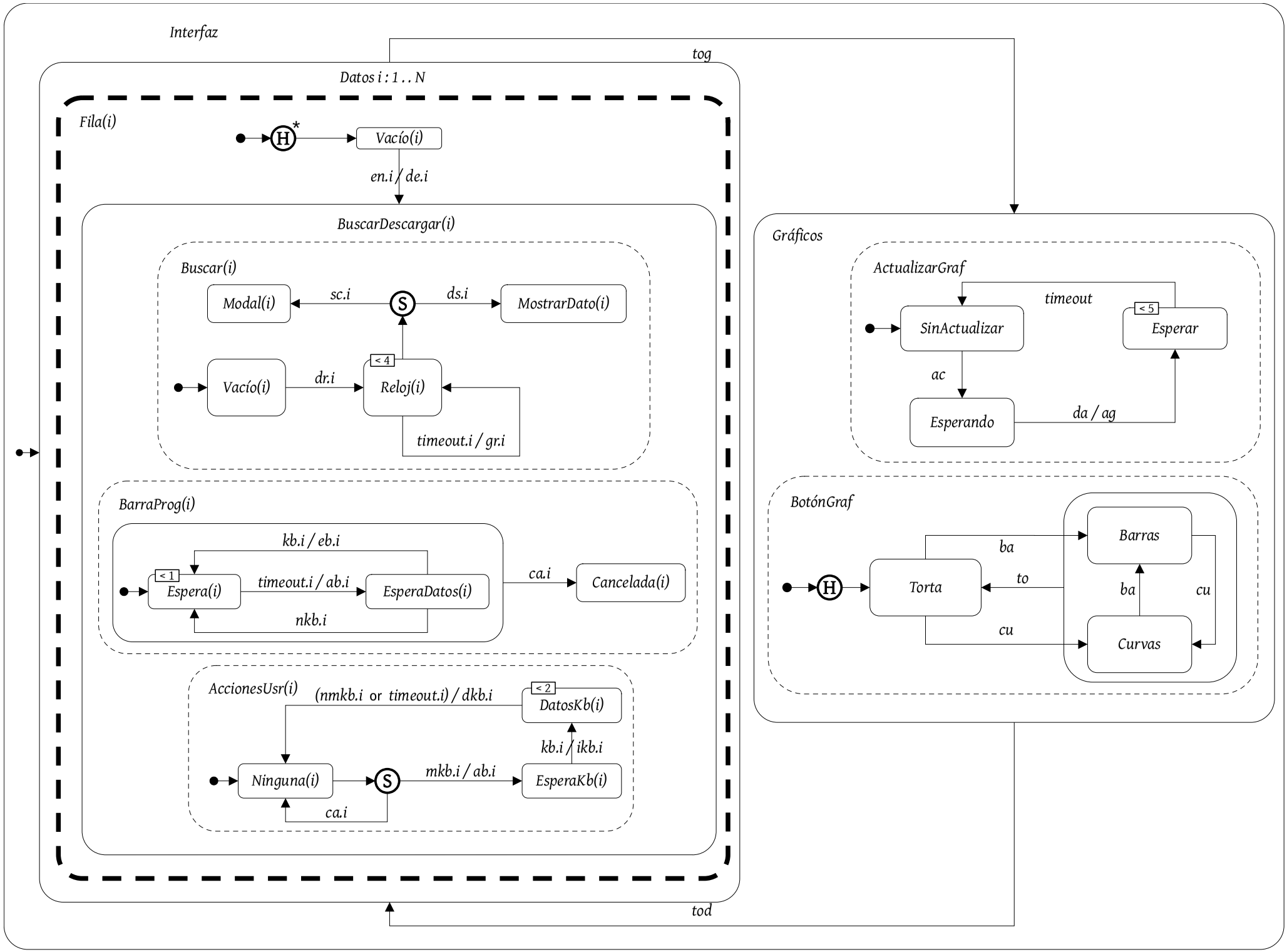


Figura 9: Especificación de *AccionesUsr*

Para pensar, discutir y analizar

1. Clasifique los eventos designados en EC, MC, S y U considerando que la interfaz es la máquina y el resto el entorno.
2. Modelar la posibilidad de que el usuario elimine el dato ingresado en un cuadro de texto de forma tal que en ese momento se cancela la búsqueda de datos y la barra de progreso se reduce a cero.
3. Los eventos *timeout* en *Buscar* son ambiguos. Harel determina cómo eliminar esa ambigüedad. Solucione el problema.
4. *MostrarDato* debería ser más una acción de la interfaz que un estado. Lo mismo aplica para *Modal*.
5. El usuario debería poder cerrar la ventana modal que se abre cuando el servidor no responde.
6. La especificación de *BarraProg* debería empezar en un estado no temporizado desde el cual la interfaz ejecuta *ab* y de allí comienza el ciclo de consulta cada 1 segundo. Es decir no tiene mucho sentido que la interfaz espere un segundo para preguntar por primera vez si hay que extender la barra o no.
7. Los requerimientos piden que el usuario pueda cancelar la descarga en cualquier momento. Verifique si *AccionesUsr* cumple con ese requerimiento. Justifique. En caso de que no lo haga modifique el Statechart.
8. Más arriba dijimos "observar que por cuestiones físicas el usuario no puede hacer ambas cosas al mismo tiempo". ¿Está dicho en el modelo? Si no lo está, hágalo. ¿En cuál de las descripciones de WRSPM debería ser incluido?
9. Usamos el evento *kb* para dos cosas diferentes en los Statecharts *BarraProg* y *AccionesUsr*.
10. Cuando el usuario cancela la descarga de datos sigue siendo posible que se muestre la cantidad de kilobytes descargados. Este no parece un comportamiento razonable.
11. Suponga que la interfaz cuenta con un botón para reiniciar una descarga cancelada. Extienda el modelo para tener en cuenta este requerimiento.
12. Verifique si el requerimiento que pide actualizar la solapa de datos cuando se vuelve a ella, se está cumpliendo en la especificación. Si no es así corrija el modelo.


Interfaz
tog
Datos i : 1 . . N
Fila(i)
H*
Vacío(i)
en.i / de.i
BuscarDescargar(i)
Buscar(i)
Modal(i)
sc.i
S
ds.i
MostrarDato(i)
Vacío(i)
dr.i
< 4
Reloj(i)
timeout.i / gr.i
BarraProg(i)
kb.i / eb.i
< 1
Espera(i)
timeout.i / ab.i
EsperaDatos(i)
ca.i
Cancelada(i)
nkb.i
AccionesUsr(i)
(nmkb.i or timeout.i) / dkb.i
< 2
DatosKb(i)
kb.i / ikb.i
Ninguna(i)
S
mkb.i / ab.i
EsperaKb(i)
ca.i
Gráficos
ActualizarGraf
timeout
SinActualizar
< 5
Esperar
ac
Esperando
da / ag
BotónGraf
H
Torta
ba
to
cu
Barras
ba
cu
Curvas
tod