

Simulación de Eventos Discretos

Trabajo Práctico I

Ricardo Kirkner
rkirkner@dc.uba.ar
LU 479/98

8 de octubre de 2004

Índice

1. Identificación del sistema a modelar	2
1.1. Descripción conceptual	2
1.2. Objetivo del experimento	2
2. Estructura del modelo	2
2.1. Descripción de la estructura	2
2.2. Variables descriptivas	3
2.2.1. Generador de estímulos	3
2.2.2. Sensor de Movimiento/Sonido/Apertura	3
2.2.3. Unidad de Control	4
2.2.4. Log	4
2.2.5. Analizador	4
3. Formalización del modelo	5
3.1. Modelos Atómicos	5
3.1.1. Generador de Estímulos	5
3.1.2. Log	8
3.1.3. Analizador	10
3.1.4. Sensor de Apertura	12
3.1.5. Sensor de Movimiento	14
3.1.6. Sensor de Sonido	17
3.1.7. Unidad de Control	20
3.2. Modelos Acoplados	25
3.2.1. Calibrador	25
3.2.2. Alarma	28
3.2.3. Sistema	30
4. Discusión	33

1. Identificación del sistema a modelar

1.1. Descripción conceptual

El sistema que se desea modelar es un sistema de alarma. La alarma está compuesta por una serie de sensores, que tienen un grado de sensibilidad definido, y una unidad de control, que debe determinar si se debe activar la alarma, en función de los estímulos recibidos por los distintos sensores.

1.2. Objetivo del experimento

El objetivo de modelar este sistema, y luego simularlo, consiste en poder determinar el grado de sensibilidad requerida por cada sensor, de manera que el sistema pueda detectar de forma correcta las intrusiones, reduciendo al mínimo las falsas alarmas.

2. Estructura del modelo

2.1. Descripción de la estructura

El modelo realizado cuenta con el sistema de alarma propiamente dicho, y con un calibrador compuesto por un generador de estímulos, un registro (LOG), y un analizador. La función del calibrador es modificar el sistema de alarma, de manera de lograr determinar el grado de sensibilidad requerido por cada sensor para que la alarma detecte las intrusiones, minimizando las falsas alarmas.

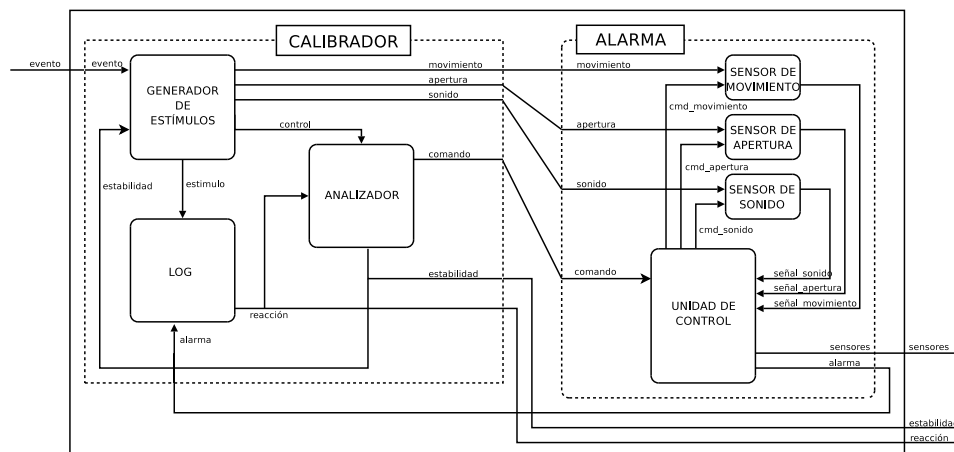


Figura 1: Estructura del modelo

El generador de estímulos recibe la descripción de los eventos que se desean simular, y se encarga de generar los estímulos que estos eventos producirían sobre los sensores. Otra función del generador de estímulos es informar al analizador, si el estímulo generado debería generar o no una alarma. El registro se encarga de registrar los estímulos generados, junto con la respuesta producida por el sistema de alarma. Finalmente, el analizador, verifica si el sistema de alarma se comportó de la manera esperada. Para ello, recibe los estímulos que son enviados al sistema de alarma, junto a las respuestas producidas por el mismo. A partir de analizar estas respuestas, puede determinar si el sistema se encuentra estabilizado, o si hace falta modificar la sensibilidad de los sensores (en cuyo caso puede afectar al sistema de alarma).

2.2. Variables descriptivas

Aquí simplemente enunciaremos las variables que describen los estados internos de cada componente. Para una descripción de cada variable, referirse a la sección correspondiente de la segunda parte del trabajo.

2.2.1. Generador de estímulos

- estado
- eventos
- apertura
- movimiento
- sonido
- control
- estímulos
- estabilidad

2.2.2. Sensor de Movimiento/Sonido/Apertura

En general, los sensores son iguales en lo que respecta a su modelado. Sus variables descriptivas serán:

- estado
- apertura/movimiento/sonido (dependiendo de que sensor se trate)
- senial
- rango

- niveles
- comando

2.2.3. Unidad de Control

- estado
- sonido
- apertura
- movimiento
- alarma
- comando
- sensores

2.2.4. Log

El log deberá registrar los estímulos generados junto a las respuestas producidas por el sistema de alarma. Para ello deberá coordinar la llegada de los mensajes provenientes del *Generador de Estímulos* y de la *Alarma* para crear el mensaje de reacción correspondiente.

- estado
- estímulos
- alarma
- reacción

2.2.5. Analizador

El analizador, recibirá las reacciones generadas por el *Log*, y las almacenará en un registro.

- estado
- registro
- alarma
- estabilidad
- comando
- control

3. Formalización del modelo

Como se puede ver en la Figura 1, el diseño del experimento cuenta con los modelos

- Calibrador
- Alarma
- Generador de Estímulos
- Log
- Analizador
- Sensor de Apertura
- Sensor de Movimiento
- Sensor de Sonido
- Unidad de Control

Estos modelos se pueden agrupar de la siguiente manera:

- Modelos Atómicos
 - Generador de Estímulos
 - Log
 - Analizador
 - Sensor de Apertura
 - Sensor de Movimiento
 - Sensor de Sonido
 - Unidad de Control
- Modelos Acoplados
 - Calibrador
 - Alarma

A continuación daremos una descripción formal de cada uno de ellos.

3.1. Modelos Atómicos

3.1.1. Generador de Estímulos

Como se puede ver en la Figura 2, este modelo cuenta con dos puertos de entrada y cinco puertos de salida, a saber

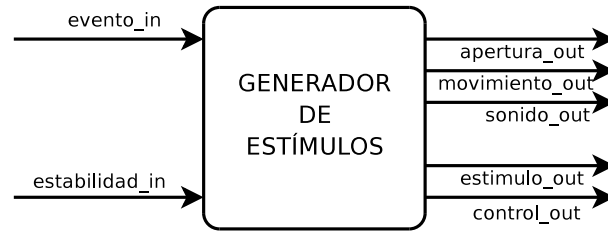


Figura 2: Modelo Atómico Generador de Estímulos

Puertos de Entrada

evento_in: Por este puerto entran las definiciones de los eventos que generarán los estímulos para la alarma. Por este puerto entran todos los datos necesarios para la configuración inicial del sistema.

estabilidad_in: Por este puerto entra la señal de estabilidad producida por el analizador. Esta señal se utiliza para determinar si se debe terminar la simulación, porque el sistema se estabilizó, o si se debe continuar hasta estabilizarlo.

Puertos de Salida

apertura_out: Por este puerto salen los estímulos generados, que deben activar al Sensor de Apertura.

movimiento_out: Por este puerto salen los estímulos generados, que deben activar al Sensor de Movimiento.

sonido_out: Por este puerto salen los estímulos generados, que deben activar al Sensor de Sonido.

control_out: Por este puerto salen los comandos hacia el analizador. Estos comandos incluyen: el momento en que comienza un ciclo de entrenamiento, y el momento en que termina el mismo (esto se utiliza para que el analizador pueda determinar si el ciclo fue efectivo o no, y así determinar si el sistema está estable o no), y los comandos de configuración del analizador.

estimulo_out: Por este puerto salen todos los estímulos generados, de manera de que el Log pueda hacer uso de ellos.

Variables de Estado

Por lo mencionado en la Sección 4, no se pudo terminar esta parte.

Pseudocódigo

Por lo mencionado en la Sección 4, es posible que el siguiente pseudocódigo contenga errores. $\delta_{ext}(x, s, e)$

```
segun x.puerto
evento_in:
eventos = parsear_descripcion(x.valor)
estado = CONFIGURANDO;
holdIn(tiempoReaccion)
// como este es el unico puerto, no deberia llegar aca
// asi que si llego, es un error. por lo tanto, no hago nada
estado = estado
passivate()
```

$\delta_{int}(s, e)$

```
segun estado
GENERANDO_EVENTOS:
evento = dameEvento()
estado = EVENTO_GENERADO
holdIn(tiempoReaccionInterna)
EVENTO_GENERADO:
estimulos = dameEstimulos(evento)
estado = ESTIMULOS_GENERADOS
holdIn(tiempoReaccionInterna)
ESTIMULOS_GENERADOS:
si cantidadEventos > 1
cantidadEventos--
estado = GENERANDO_EVENTOS
holdIn(tiempoReaccionInterna)
sino
estado = TERMINANDO
holdIn(tiempoReaccionInterna)
CONFIGURANDO:
estado = EMPEZANDO
holdIn(tiempoReaccionInterna)
EMPEZANDO:
cantidadEventos = dameCantidadEventos()
estado = GENERANDO_EVENTOS
holdIn(tiempoReaccionInterna)
TERMINANDO:
estado = EMPEZANDO
holdIn(tiempoReaccionInterna)
otro:
// esto no deberia ocurrir nunca, asi que si llego a este
```

```

////////////////////estado, me quedo aca para siempre
////////////////////estado=estado
////////////////////passivate()

```

$\lambda(s)$

```

segun estado
ESTIMULOS_GENERADOS:
para cada estimulo en estimulos
puerto=damePuerto(estimulo)
valor=dameValor(estimulo)
sendOutput(puerto, valor)
sendOutput(estimulo_out, estimulos)
EMPEZANDO:
sendOutput(control_out, 1)
TERMINANDO:
sendOutput(control_out, 0)

```

3.1.2. Log

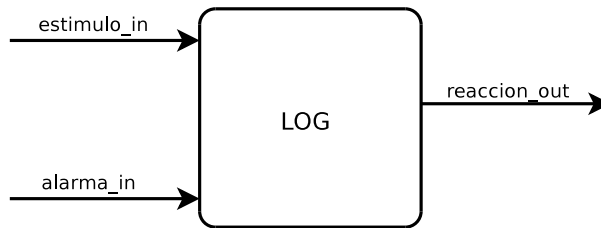


Figura 3: Modelo Atómico Log

Como se puede ver en la Figura 3, este modelo cuenta con dos puertos de entrada y un puerto de salida, a saber

Puertos de Entrada

estimulo_in: Por este puerto entran los estímulos producidos por el Generador de Estímulos. Estos datos se utilizan para confeccionar el reporte de reacción.

alarma_in: Por este puerto entran las señales de alarma producidas por el componente Alarma (propia mente por la Unidad de Control). Estos datos se utilizan para confeccionar el reporte de reacción.

Puertos de Salida

reaccion_out: Por este puerto salen los reportes de reacción. Estos reportes quedan como salida del sistema, y van al Analizador, para que este pueda verificar si el sistema se comportó de manera esperada.

Variables de Estado

Por lo mencionado en la Sección 4, no se pudo terminar esta parte.

Pseudocódigo

Por lo mencionado en la Sección 4, es posible que el siguiente pseudocódigo contenga errores. $\delta_{ext}(x, s, e)$

```
segun x.puerto
    estimulos_in:
        estimulos = x.valor
        si estado == ESPERANDO_ESTIMULOS
            estado = ARMAR_REACCION
            holdIn(tiempoReaccion)
        si estado == LISTO
            estado = ESPERANDO_ALARMA
            holdIn(tiempoReaccion)
        sino
            estado = estado
            passivate()
    alarma_in:
        alarma = x.valor
        si estado == ESPERANDO_ALARMA
            estado = ARMAR_REACCION
            holdIn(tiempoReaccion)
        si estado == LISTO
            estado = ESPERANDO_ESTIMULOS
            holdIn(tiempoReaccion)
        sino
            estado = estado
            passivate()
```

$\delta_{int}(s, e)$

```
segun estado
    ENVIAR_REACCION:
        estado = LISTO
        passivate()
    ARMAR_REACCION:
```

```

reaccion=_armarReaccion(estimulos,alarma)
estado=_ENVIAR_REACCION
holdIn(tiempoReaccionInterna)
otro:
//esto no deberia ocurrir nunca, asi que si llego a este
//estado, me quedo aca para siempre
estado=_estado
passivate()

```

$\lambda(s)$

```

si_estado==_ENVIAR_REACCION
sendOutput(reaccion_out,reaccion)

```

3.1.3. Analizador

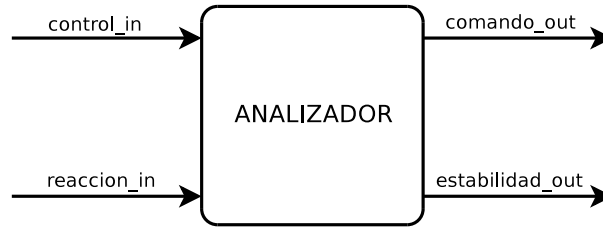


Figura 4: Modelo Atómico Analizador

Como se puede ver en la Figura 4, este modelo cuenta con dos puertos de entrada y dos puertos de salida, a saber

Puertos de Entrada

control_in: Por este puerto entran las señales de control provenientes del Generador de Estímulos.

reaccion_in: Por este puerto entran los reportes de reacción provenientes del Log.

Puertos de Salida

comando_out: Por este puerto salen los comandos hacia el componente Alarma (propiamente hacia la Unidad de Control). Estos comandos indican la cantidad de estímulos que llegarán como parte del próximo evento, contienen ordenes de configuración de los sensores, e informan de un requerimiento de información al sistema de alarma.

estabilidad_out: Por este puerto salen las señales de estabilidad. Una señal de valor 1 indica que el sistema está estable, mientras que un 0 indica que el sistema es inestable. Un sistema se dice estable, cuando se comporta de la manera esperada, es decir, cuando produce alarmas ante los eventos que deberían generar alarmas, y no las produce ante eventos que no las deberían generar.

Variables de Estado

Por lo mencionado en la Sección 4, no se pudo terminar esta parte.

Pseudocódigo

Por lo mencionado en la Sección 4, es posible que el siguiente pseudocódigo contenga errores. $\delta_{ext}(x, s, e)$

```

segun x.puerto
control_in:
control = x.valor
estado = ACTUALIZAR_ESTADO
holdIn(tiempoReaccion)
reaccion_in:
si estado == ACTIVO
registro = x.valor
estado = REGISTRANDO
holdIn(tiempoReaccion)
sino
estado = estado
passivate()

```

$\delta_{int}(s, e)$

```

segun estado
REGISTRANDO:
registrar(registro)
estado = ACTIVO
passivate()
ACTUALIZAR_ESTADO:
si control == 1
estado = ACTIVO
sino
si control == 0 y estado == ACTIVO
estado = ANALIZANDO
sino
estado = estado
passivate()
ANALIZANDO:

```

```

estabilidad=x.valor
si estabilidad==1
estado=INFORMAR_ESTABILIDAD
comando=INFORMAR_SENSORES
sino
estado=INFORMAR_ESTABILIDAD
comando=dameComandoAjustar()
holdIn(tiempoReaccionInterna)
INFORMAR_ESTABILIDAD:
estado=INACTIVO
passivate()
otro:
//esto no deberia ocurrir nunca, asi que si llego a este
//estado, me quedo aca para siempre
estado=estado
passivate()
λ(s)

si estado==INFORMAR_ESTABILIDAD
sendOutput(estabilidad_out, estabilidad)
sendOutput(comando_out, comando)

```

3.1.4. Sensor de Apertura



Figura 5: Modelo Atómico Sensor de Apertura

Como se puede ver en la Figura 5, este modelo cuenta con dos puertos de entrada y un puerto de salida, a saber

Puertos de Entrada

apertura_in: Por este puerto llegan los estímulos de apertura.

cmd_apertura_in: Por este puerto llegan los comandos al sensor. Comandos pueden ser utilizados para activar o desactivar el sensor.

Puertos de Salida

senal_apertura_out: Por este puerto sale una señal, indicando el nivel de activación del sensor.

Variables de Estado

Las variables que definen el estado de este modelo son:

estado: Esta variable refleja el estado interno del sensor. Los valores que puede tomar son: ESPERANDO, ESTIMULO_RECIBIDO, INACTIVO, ENVIAR_SENIAL, CONFIGURANDO y ACTIVO.

apertura: Esta variable refleja el valor del estímulo recibido.

senial: Esta variable refleja el valor de la señal producida por el sensor.

activado: Esta variable refleja si el sensor se encuentra activado o desactivado. Cuando el sensor está desactivado, ignora los estímulos provenientes del entorno, y tampoco produce señal alguna. Para el observador, el sensor no existe.

comando: Esta variable refleja el comando recibido por el sensor.

tiempoReaccion: Esta variable define el tiempo que tardará el sensor en responder a los estímulos.

Pseudocódigo

$\delta_{ext}(x, s, e)$

```
segun x.puerto
    apertura_in:
        si activado
            apertura = x.valor
            estado = ESTIMULO_RECIBIDO
            holdIn(tiempoReaccion)
        sino
            // como estoy desactivado, ignoro el mensaje
            estado = estado
            passivate()
    cmd_apertura_in:
        comando = x.valor
        estado = CONFIGURANDO
        holdIn(tiempoReaccion)
    otro:
        estado = estado
        passivate()
```

$\delta_{int}(s, e)$

```
segun estado
    ESTIMULO_RECIBIDO:
        senial = calcularSenial()
```

```

#####estado=_ENVIAR_SENIAL
#####holdIn(0)
#####ENVIAR_SENIAL:
#####estado=_ESPERANDO
#####passivate()
#####INACTIVO:
#####si_activado
#####estado=_ENVIAR_SENIAL
#####holdIn(tiempoReaccion)
#####sino
#####//_como_estoy_desactivado,_ignoro_el_mensaje
#####estado=_estado
#####passivate()
#####CONFIGURANDO:
#####si_esComandoActivacion()
#####activado=_getValorActivacion()
#####si_no_activado
#####estado=_INACTIVO
#####passivate()
#####sino
#####estado=_ACTIVO
#####holdIn(0)
#####sino
#####//_no_es_un_comando_valido,_ignoramos
#####estado=_estado
#####passivate()
#####otro:
#####//_esto_no_deberia_ocurrir_nunca,_asi_que_si_llego_a_este
#####//_estado,_me_quedo_aca_para_siempre
#####estado=_estado
#####passivate()

```

$\lambda(s)$

```

####si_estado==_ENVIAR_SENIAL
#####si_activado
#####sendOutput(senial_apertura_out,_senial)

```

3.1.5. Sensor de Movimiento

Como se puede ver en la Figura 6, este modelo cuenta con dos puertos de entrada y un puerto de salida, a saber

Puertos de Entrada

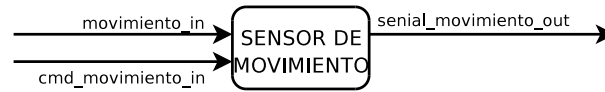


Figura 6: Modelo Atómico Sensor de Movimiento

movimiento_in: Por este puerto llegan los estímulos de movimiento.

cmd_movimiento_in: Por este puerto llegan los comandos al sensor. Comandos pueden ser utilizados para modificar la configuración del sensor, en particular en dos aspectos, alterando las variables *rango* y *niveles*, lo que permite modificar el grado de sensibilidad del sensor; también es posible enviar comandos para requerir información sobre la configuración del sensor, es decir, sobre el valor de estas variables, y para activar o desactivar el sensor.

Puertos de Salida

senal_movimiento_out: Por este puerto sale una señal, indicando el nivel de activación del sensor.

Variables de Estado

Las variables que definen el estado de este modelo son:

estado: Esta variable refleja el estado interno del sensor. Los valores que puede tomar son: ESPERANDO, ESTIMULO_RECIBIDO, INACTIVO, ENVIAR_SENIAL, CONFIGURANDO y ACTIVO.

movimiento: Esta variable refleja el valor del estímulo recibido.

senal: Esta variable refleja el valor de la señal producida por el sensor.

activado: Esta variable refleja si el sensor se encuentra activado o desactivado. Cuando el sensor está desactivado, ignora los estímulos provenientes del entorno, y tampoco produce señal alguna. Para el observador, el sensor no existe.

comando: Esta variable refleja el comando recibido por el sensor.

rango: Esta variable refleja el rango de sensado actualmente definido en el sensor.

niveles: Esta variable refleja la cantidad de niveles discretos en los que será particionado el rango de sensado.

tiempoReaccion: Esta variable define el tiempo que tardará el sensor en responder a los estímulos.

Pseudocódigo

$\delta_{ext}(x, s, e)$

```
segun x.puerto
movimiento_in:
si activado
movimiento = x.valor
estado = ESTIMULO_RECIBIDO
holdIn(tiempoReaccion)
sino
// como estoy desactivado, ignoro el mensaje
estado = estado
passivate()
cmd_movimiento_in:
comando = x.valor
estado = CONFIGURANDO
holdIn(tiempoReaccion)
otro:
estado = estado
passivate()
```

$\delta_{int}(s, e)$

```
segun estado
ESTIMULO_RECIBIDO:
senal = calcularSenial()
estado = ENVIAR_SENIAL
holdIn(0)
ENVIAR_SENIAL:
estado = ESPERANDO
passivate()
INACTIVO:
si activado
estado = ENVIAR_SENIAL
holdIn(tiempoReaccion)
sino
// como estoy desactivado, ignoro el mensaje
estado = estado
passivate()
CONFIGURANDO:
si esComandoActivacion()
activado = getValorActivacion()
sino no activado
estado = INACTIVO
passivate()
```



```

#####sino
#####estado=ACTIVO
#####holdIn(0)
#####si_esComandoRango()
#####rango=getValorRango()
#####senal=calcularSenial()
#####estado=ESPERANDO
#####passivate()
#####si_esComandoNiveles()
#####niveles=getValorNiveles()
#####senal=calcularSenial()
#####estado=ESPERANDO
#####passivate()
#####//si_llegamos_aca,no_es_un_comando_valido,ignoramos
#####estado=estado
#####passivate()
#####otro:
#####//esto_no_deberia_ocurrir_nunca,asi_que_si_llego_a_este
#####//estado,me_quedo_aca_para_siempre
#####estado=estado
#####passivate()

λ(s)

#####si_estado==ENVIAR_SENIAL
#####si_activado
#####sendOutput(senial_movimiento_out,senial)

```

3.1.6. Sensor de Sonido



Figura 7: Modelo Atómico Sensor de Sonido

Como se puede ver en la Figura 7, este modelo cuenta con dos puertos de entrada y un puerto de salida, a saber

Puertos de Entrada

sonido_in: Por este puerto llegan los estímulos de sonido.

cmd_sonido_in: Por este puerto llegan los comandos al sensor. Comandos pueden ser utilizados para modificar la configuración del sensor,

en particular en dos aspectos, alterando las variables *rango* y *niveles*, lo que permite modificar el grado de sensibilidad del sensor; también es posible enviar comandos para requerir información sobre la configuración del sensor, es decir, sobre el valor de estas variables, y para activar o desactivar el sensor.

Puertos de Salida

senial_sonido_out: Por este puerto sale una señal, indicando el nivel de activación del sensor.

Variables de Estado

Las variables que definen el estado de este modelo son:

estado: Esta variable refleja el estado interno del sensor. Los valores que puede tomar son: ESPERANDO, ESTIMULO_RECIBIDO, INACTIVO, ENVIAR_SENIAL, CONFIGURANDO y ACTIVO.

sonido: Esta variable refleja el valor del estímulo recibido.

senial: Esta variable refleja el valor de la señal producida por el sensor.

activado: Esta variable refleja si el sensor se encuentra activado o desactivado. Cuando el sensor está desactivado, ignora los estímulos provenientes del entorno, y tampoco produce señal alguna. Para el observador, el sensor no existe.

comando: Esta variable refleja el comando recibido por el sensor.

rango: Esta variable refleja el rango de sensado actualmente definido en el sensor.

niveles: Esta variable refleja la cantidad de niveles discretos en los que será particionado el rango de sensado.

tiempoReaccion: Esta variable define el tiempo que tardará el sensor en responder a los estímulos.

Pseudocódigo

$\delta_{ext}(x, s, e)$

```

segun x.puerto
    sonido_in:
        si activado
            sonido = x.valor
            estado = ESTIMULO_RECIBIDO

```

```

    holdIn(tiempoReaccion)
    sino
    // como estoy desactivado, ignoro el mensaje
    estado = estado
    passivate()
    cmd_sonido_in:
    comando = x.valor
    estado = CONFIGURANDO
    holdIn(tiempoReaccion)
    otro:
    estado = estado
    passivate()

 $\delta_{int}(s, e)$ 

    segun estado
    ESTIMULO_RECIBIDO:
    senial = calcularSenial()
    estado = ENVIAR_SENIAL
    holdIn(0)
    ENVIAR_SENIAL:
    estado = ESPERANDO
    passivate()
    INACTIVO:
    si activado
    estado = ENVIAR_SENIAL
    holdIn(tiempoReaccion)
    sino
    // como estoy desactivado, ignoro el mensaje
    estado = estado
    passivate()
    CONFIGURANDO:
    si esComandoActivacion()
    activado = getValorActivacion()
    si no activado
    estado = INACTIVO
    passivate()
    sino
    estado = ACTIVO
    holdIn(0)
    si esComandoRango()
    rango = getValorRango()
    senial = calcularSenial()
    estado = ESPERANDO
    passivate()

```

```

#####si_esComandoNiveles()
#####niveles=_getValorNiveles()
#####senal=_calcularSenial()
#####estado=_ESPERANDO
#####passivate()
#####//si_llegamos_aca,_no_es_un_comando_valido,_ignoramos
#####estado=_estado
#####passivate()
#####otro:
#####//esto_no_deberia_ocurrir_nunca,_asi_que_si_llego_a_este
#####//estado,_me_quedo_aca_para_siempre
#####estado=_estado
#####passivate()

λ(s)

#####si_estado==_ENVIAR_SENIAL
#####si_activado
#####sendOutput(senial_sonido_out,_senal)

```

3.1.7. Unidad de Control

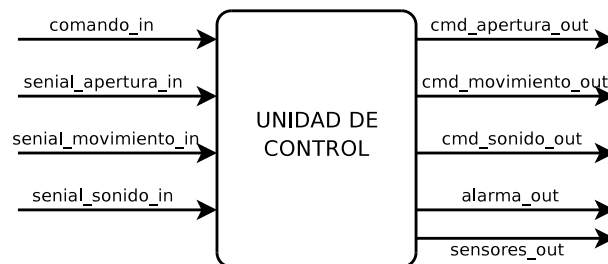


Figura 8: Modelo Atómico Unidad de Control

Como se puede ver en la Figura 8, este modelo cuenta con cuatro puertos de entrada y cinco puertos de salida, a saber

Puertos de Entrada

comando_in: Por este puerto llegan comandos a la Unidad de Control.

Los comandos que puede recibir este componente son para indicarle la cantidad de estímulos que forman parte del mismo evento (de manera que pueda analizar la situación una vez recibidos todos los estímulos que deberían activarse simultáneamente), para requerirle información sobre el estado de los sensores (valores de rango y niveles de cada sensor) y para configurar los distintos sensores (la unidad de control recibe

la nueva configuración de un sensor, y le indica al mismo que debe modificarse). Además puede recibir la orden de activar o desactivar un sensor dado.

senial_apertura_in: Por este puerto llega la señal generada por el Sensor de Apertura.

senial_movimiento_in: Por este puerto llega la señal generada por el Sensor de Movimiento.

senial_sonido_in: Por este puerto llega la señal generada por el Sensor de Sonido.

Puertos de Salida

cmd_apertura_out: Por este puerto salen los comandos de configuración y/o requerimiento de información para el Sensor de Apertura.

cmd_movimiento_out: Por este puerto salen los comandos de configuración y/o requerimiento de información para el Sensor de Movimiento.

cmd_sonido_out: Por este puerto salen los comandos de configuración y/o requerimiento de información para el Sensor de Sonido.

alarma_out: Por este puerto salen las alarmas generadas en función de la activación de los distintos sensores.

sensores_out: Por este puerto sale la información sobre el estado de todos los sensores en un momento dado.

Variables de Estado

Las variables que definen el estado de este modelo son:

estado: Esta variable refleja el estado interno del sensor. Los valores que puede tomar son: ESPERANDO, RECIBIENDO_SENIALES, RECIBIENDO_COMANDOS, CONFIGURANDO, ANALIZANDO_EVENTO, ENVIANDO_ALARMA, ENVIANDO_CMD_APERTURA, ENVIANDO_CMD_MOVIMIENTO, ENVIANDO_CMD_SONIDO, ENVIANDO_SENSORES.

sonido: Esta variable refleja los valores de las señales recibidas del Sensor de Sonido y aún no procesadas.

apertura: Esta variable refleja los valores de las señales recibidas del Sensor de Apertura y aún no procesadas.

movimiento: Esta variable refleja los valores de las señales recibidas del Sensor de Movimiento y aún no procesadas.

sensores: Esta variable refleja los valores de configuración de cada sensor.

comando: Esta variable refleja los valores de los comandos recibidos por la Unidad de Control aún no procesados.

comandoActivo: Esta variable refleja el valor del comando que se está ejecutando en la Unidad de Control.

alarma: Esta variable refleja el valor de la alarma que se produjo en función de los estímulos recibidos.

cantidadApertura: Esta variable refleja la cantidad de señales provenientes del Sensor de Apertura que deben ser considerados como parte del evento actualmente analizado por la Unidad de Control. Este valor puede ser 0 o 1.

cantidadMovimiento: Esta variable refleja la cantidad de señales provenientes del Sensor de Movimiento que deben ser considerados como parte del evento actualmente analizado por la Unidad de Control. Este valor puede ser 0 o 1.

cantidadSonido: Esta variable refleja la cantidad de señales provenientes del Sensor de Sonido que deben ser considerados como parte del evento actualmente analizado por la Unidad de Control. Este valor puede ser 0 o 1.

tiempoReaccion: Esta variable define el tiempo que tardará el sensor en responder a las señales y/o comandos.

tiempoReaccionInterna: Esta variable define el tiempo que tardará el sensor en responder a cambios de estado internos.

rangoApertura: Esta variable refleja el rango de sensado actual del Sensor de Apertura.

rangoMovimiento: Esta variable refleja el rango de sensado actual del Sensor de Movimiento.

rangoSonido: Esta variable refleja el rango de sensado actual del Sensor de Sonido.

nivelesApertura: Esta variable refleja el valor actual de la cantidad de niveles de discretizado del Sensor de Apertura.

nivelesMovimiento: Esta variable refleja el valor actual de la cantidad de niveles de discretizado del Sensor de Movimiento.

nivelesSonido: Esta variable refleja el valor actual de la cantidad de niveles de discretizado del Sensor de Sonido.

umbralAlarma: Esta variable define el umbral de activación de la alarma. Si la suma de las señales provenientes de los sensores involucrados en el evento analizado supera este umbral, se activa una alarma. En caso contrario, no.

Pseudocódigo

$\delta_{ext}(x, s, e)$

```

segun x.puerto
    senial_apertura_in:
        agregar(apertura, x.valor)
        estado = RECIBIENDO_SENIALES
        holdIn(tiempoReaccion)
    senial_movimiento_in:
        agregar(movimiento, x.valor)
        estado = RECIBIENDO_SENIALES
        holdIn(tiempoReaccion)
    senial_sonido_in:
        agregar(sonido, x.valor)
        estado = RECIBIENDO_SENIALES
        holdIn(tiempoReaccion)
    comando_in:
        agregar(comando, x.valor)
        estado = RECIBIENDO_COMANDOS
        holdIn(tiempoReaccion)
    otro:
        estado = estado
        passivate()

```

$\delta_{int}(s, e)$

```

segun estado
    RECIBIENDO_SENIALES:
        si llegaronTodasLasSeniales()
            armarEvento()
            estado = ANALIZANDO_EVENTO
            holdIn(tiempoReaccionInterna)
        sino
            estado = estado
            passivate()
    RECIBIENDO_COMANDOS:
        si esComandoConfiguracion()

```

```

#####estado=CONFIGURANDO
#####holdIn(tiempoReaccionInterna)
#####si_esComandoSeniales()
#####getCantidadSeniales()
#####estado=RECIBIENDO_SENIALES
#####holdIn(tiempoReaccionInterna)
#####si_esComandoInfo()
#####sensores=getSensores()
#####estado=ENVIANDO_SENSORES
#####holdIn(tiempoReaccionInterna)
#####//si_llegue_hasta_aca_es_porque_el_comando_era_invalido
#####//ignoramos
#####estado=estado
#####passivate()
#####ANALIZANDO_EVENTO:
#####alarma=generarAlarma()
#####estado=ENVIANDO_ALARMA
#####holdIn(tiempoReaccionInterna)
#####ENVIANDO_ALARMA:
#####borrarSenialesUsadas()
#####borrarComandoActual()
#####si_haySeniales()
#####estado=RECIBIENDO_SENIALES
#####holdIn(tiempoReaccionInterna)
#####sino
#####estado=estado
#####passivate()
#####ENVIANDO_CMD_APERTURA:
#####ENVIANDO_CMD_MOVIMIENTO:
#####ENVIANDO_CMD_SONIDO:
#####borrarComandoActual()
#####si_hayComandos()
#####estado=RECIBIENDO_COMANDOS
#####holdIn(tiempoReaccionInterna)
#####sino
#####estado=ESPERANDO
#####passivate()
#####ENVIANDO_SENSORES:
#####borrarSensores()
#####estado=ESPERANDO
#####passivate()
#####CONFIGURANDO:
#####si_esCmApertura()
#####borrarComandoActual()

```



```

comandoActivo=_getValorCmdApertura()
estado=_ENVIANDO_CMD_APERTURA
holdIn(tiempoReaccionInterna)
si_esCmdMovimiento()
borrarComandoActual()
comandoActivo=_getValorCmdMovimiento()
estado=_ENVIANDO_CMD_MOVIMIENTO
holdIn(tiempoReaccionInterna)
si_esCmdSonido()
borrarComandoActual()
comandoActivo=_getValorCmdSonido()
estado=_ENVIANDO_CMD_SONIDO
holdIn(tiempoReaccionInterna)
//si_llegue_aca_es_porque_el_comando_era_invalido
//ignoramos
estado=_estado
passivate()
otro:
//esto_no_deberia_ocurrir_nunca,_asi_que_si_llego_a_este
//estado,_me_quedo_aca_para_siempre
estado=_estado
passivate()

```

$\lambda(s)$

```

segun_estado
ENVIANDO_ALARMA:
sendOutput(alarma_out,_alarma)
ENVIANDO_CMD_APERTURA:
sendOutput(cmd_apertura_out,_comandoActivo)
ENVIANDO_CMD_MOVIMIENTO:
sendOutput(cmd_movimiento_out,_comandoActivo)
ENVIANDO_CMD_SONIDO:
sendOutput(cmd_sonido_out,_comandoActivo)
ENVIANDO_SENSORES:
sendOutput(sensores_out,_sensores)

```

3.2. Modelos Acoplados

3.2.1. Calibrador

Como se puede ver en la Figura 9, este modelo cuenta con dos puerto de entrada y seis puertos de salida, a saber

Puertos de Entrada

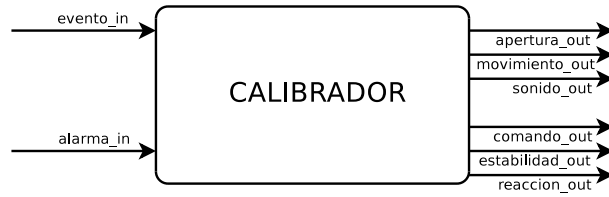


Figura 9: Modelo Acoplado Calibrador

evento_in: Este puerto se conecta al puerto homónimo del *Componente Atómico Generador de Estímulos*(ver 3.1.1) y al puerto homónimo del *Componente Acoplado Sistema*(ver 3.2.3).

alarma_in: Este puerto se conecta al puerto homónimo del *Componente Atómico Log*(ver 3.1.2) y al puerto *alarma_out* del *Componente Acoplado Alarma*(ver 3.2.2).

Puertos de Salida

apertura_out: Este puerto se conecta al puerto homónimo del *Componente Atómico Generador de Estímulos*(ver 3.1.1) y al puerto *apertura_in* del *Componente Acoplado Alarma*(ver 3.2.2).

movimiento_out: Este puerto se conecta al puerto homónimo del *Componente Atómico Generador de Estímulos*(ver 3.1.1) y al puerto *movimiento_in* del *Componente Acoplado Alarma*(ver 3.2.2).

sonido_out: Este puerto se conecta al puerto homónimo del *Componente Atómico Generador de Estímulos*(ver 3.1.1) y al puerto *sonido_in* del *Componente Acoplado Alarma*(ver 3.2.2).

comando_out: Este puerto se conecta al puerto homónimo del *Componente Atómico Analizador*(ver 3.1.3) y al puerto *comando_in* del *Componente Acoplado Alarma*(ver 3.2.2).

estabilidad_out: Este puerto se conecta al puerto homónimo del *Componente Atómico Analizador*(ver 3.1.3) y al puerto homónimo del *Componente Acoplado Sistema*(ver 3.2.3).

reaccion_out: Este puerto se conecta al puerto homónimo del *Componente Atómico Log*(ver 3.1.2) y al puerto homónimo del *Componente Acoplado Sistema*(ver 3.2.3).

Variables de Estado

Las variables que definen el estado de este modelo son:

estado:

Especificación formal

```
Calibrador = <X,Y,D,EIC,EOC,IC,select>
X = {evento,alarma}
Y = {movimiento,sonido,apertura,comando,estabilidad,reaccion}
D = {GeneradorDeEstimulos, Log, Analizador}
EIC = {(Calibrador.evento_in,GeneradorDeEstimulos.evento_in),
        (Calibrador.alarma_in,Log.alarma_in)}
EOC = {(GeneradorDeEstimulos.movimiento_out,Calibrador.movimiento_out),
        (GeneradorDeEstimulos.sonido_out,Calibrador.sonido_out),
        (GeneradorDeEstimulos.apertura_out,Calibrador.apertura_out),
        (Log.reaccion_out,Calibrador.reaccion_out),
        (Analizador.comando_out,Calibrador.comando_out),
        (Analizador.estabilidad_out,Calibrador.estabilidad_out)}
IC = {(GeneradorDeEstimulos.control_out,Analizador.control_in),
        (GeneradorDeEstimulos.estimulo_out,Log.estimulo_in),
        (Log.reaccion_out,Analizador.reaccion_in),
        (Analizador.estabilidad_out,GeneradorDeEstimulos.estabilidad_in)}
select({GeneradorDeEstimulos, Log, Analizador}) = GeneradorDeEstimulos
select({Log, Analizador}) = Log
```

Esquema de acoplamiento

```
[calibrador]
components : gen@GeneradorDeEstimulo log@Log analizador@Analizador
in : evento_in
in : alarma_in

out : apertura_out
out : movimiento_out
out : sonido_out
out : comando_out
out : estabilidad_out
out : reaccion_out

Link : evento_in evento_in@gen
Link : alarma_in alarma_in@log

Link : apertura_out@gen apertura_out
Link : movimiento_out@gen movimiento_out
Link : sonido_out@gen sonido_out
Link : comando_out@analizador comando_out
```

Link : estabilidad_out@analizador estabilidad_out
 Link : reaccion_out@log reaccion_out

Link : reaccion_out@log reaccion_in@analizador
 Link : estimulo_out@gen estimulo_in@log
 Link : control_out@gen control_in@analizador

3.2.2. Alarma

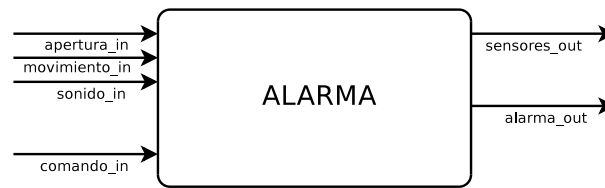


Figura 10: Modelo Acoplado Alarma

Como se puede ver en la Figura 10, este modelo cuenta con cuatro puertos de entrada y dos puertos de salida, a saber

Puertos de Entrada

apertura_in: Este puerto se conecta al puerto *apertura_out* del *Componente Acoplado Calibrador*(ver 3.2.1) y al puerto homónimo del *Componente Atómico Sensor de Apertura*(ver 3.1.4).

movimiento_in: Este puerto se conecta al puerto *movimiento_out* del *Componente Acoplado Calibrador*(ver 3.2.1) y al puerto homónimo del *Componente Atómico Sensor de Movimiento*(ver 3.1.5).

sonido_in: Este puerto se conecta al puerto *sonido_out* del *Componente Acoplado Calibrador*(ver 3.2.1) y al puerto homónimo del *Componente Atómico Sensor de Sonido*(ver 3.1.6).

comando_in: Este puerto se conecta al puerto *comando_out* del *Componente Acoplado Calibrador*(ver 3.2.1) y al puerto homónimo del *Componente Atómico Unidad de Control*(ver 3.1.7).

Puertos de Salida

sensores_out: Este puerto se conecta al homónimo del *Componente Atómico Unidad de Control*(ver 3.1.7) y al puerto homónimo del *Componente Acoplado Sistema*(ver 3.2.3).

alarma_out: Este puerto se conecta al puerto *alarma_in* del *Componente Acoplado Calibrador*(ver 3.2.1) y el puerto homónimo del *Componente Atómico Unidad de Control*(ver 3.1.7).

Variables de Estado

Las variables que definen el estado de este modelo son:

estado:

Especificación formal

```

Alarma = <X,Y,D,EIC,EOC,IC,select>
X = {movimiento,apertura,sonido,comando}
Y = {sensores,alarma}
D = {SensorDeApertura, SensorDeMovimiento, SensorDeSonido, UnidadDeControl}
EIC = {(Alarma.movimiento_in,SensorDeMovimiento.movimiento_in),
        (Alarma.apertura_in,SensorDeApertura.apertura_in),
        (Alarma.sonido_in,SensorDeSonido.sonido_in),
        (Alarma.comando_in,UnidadDeControl.comando_in)}
EOC = {(UnidadDeControl.sensores_out,Alarma.sensores_out),
        (UnidadDeControl.alarma_out,Alarma.alarma_out)}
IC  = {(UnidadDeControl.cmd_apertura_out,SensorDeApertura.cmd_apertura_in),
        (UnidadDeControl.cmd_movimiento_out,SensorDeMovimiento.cmd_movimiento_in),
        (UnidadDeControl.cmd_sonido_out,SensorDeSonido.cmd_sonido_in),
        (SensorDeApertura.senial_apertura_out,UnidadDeControl.senial_apertura_in),
        (SensorDeMovimiento.senial_movimiento_out,
            UnidadDeControl.senial_movimiento_in),
        (SensorDeSonido.senial_sonido_out,UnidadDeControl.senial_sonido_in)}
select({SensorDeApertura, SensorDeMovimiento,
        SensorDeSonido, UnidadDeControl}) = UnidadDeControl
select({SensorDeApertura, SensorDeMovimiento, SensorDeSonido}) = SensorDeApertura
select({SensorDeMovimiento, SensorDeSonido}) = SensorDeMovimiento

```

Esquema de acoplamiento

```

[alarma]
components : uc@UnidadDeControl sA@SensorDeApertura \
            sM@SensorDeMovimiento sS@SensorDeSonido

in : apertura_in
in : movimiento_in
in : sonido_in
in : comando_in

```

```
out : sensores_out
out : alarma_out
```

```
Link : apertura_in apertura_in@sA
Link : movimiento_in movimiento_in@sM
Link : sonido_in sonido_in@sS
Link : comando_in comando_in@uc
```

```
Link : sensores_out@uc sensores_out
Link : alarma_out@uc alarma_out
```

```
Link : cmd_movimiento_out@uc cmd_movimiento_in@sM
Link : cmd_apertura_out@uc cmd_apertura_in@sA
Link : cmd_sonido_out@uc cmd_sonido_in@sS
Link : senial_apertura_out@sA senial_apertura_in@uc
Link : senial_movimiento_out@sM senial_movimiento_in@uc
Link : senial_sonido_out@sS senial_sonido_in@uc
```

3.2.3. Sistema

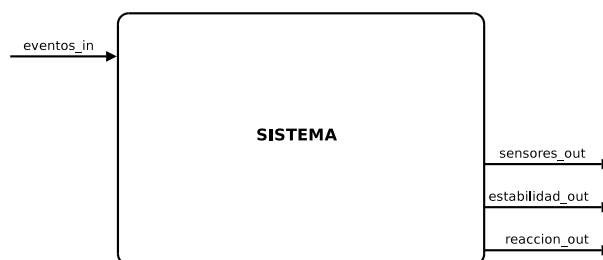


Figura 11: Modelo Acoplado Sistema

Como se puede ver en la Figura 11, este modelo cuenta con un puerto de entrada y tres puertos de salida, a saber

Puertos de Entrada

evento_in: Este puerto se conecta al puerto *in* del *Componente top* y al puerto homónimo del *Componente Acoplado Calibrador*(ver 3.2.1).

Puertos de Salida

sensores_out: Este puerto se conecta al homónimo del *Componente Acoplado Alarma*(ver 3.2.2) y al puerto *out* del *Componente top*.

estabilidad_out: Este puerto se conecta al puerto homónimo del *Componente Acoplado Calibrador*(ver 3.2.1) y al puerto *out* del *Componente top*.

reaccion_out: Este puerto se conecta al puerto homónimo del *Componente Acoplado Calibrador*(ver 3.2.1) y al puerto *out* del *Componente top*.

Variables de Estado

Las variables que definen el estado de este modelo son:

estado:

Especificación formal

```
Sistema = <X,Y,D,EIC,EOC,IC,select>
X = {evento}
Y = {sensores,estabilidad,reaccion}
D = {Calibrador, Alarma}
EIC = {(Sistema.evento_in,Calibrador.evento_in)}
EOC = {(Calibrador.estabilidad_out,Sistema.estabilidad_out),
        (Calibrador.reaccion_out,Sistema.reaccion_out),
        (Alarma.sensores_out,Sistema.sensores_out)}
IC = {(Calibrador.movimiento_out,Alarma.movimiento_in),
        (Calibrador.sonido_out,Alarma.sonido_in),
        (Calibrador.apertura_out,Alarma.apertura_in),
        (Calibrador.comando_out,Alarma.comando_in),
        (Alarma.alarma_out,Calibrador.alarma_in)}
select({Calibrador, Alarma}) = Calibrador
```

Esquema de acoplamiento

```
[top]
components : calibrador alarma

in : evento_in

out : sensores_out
out : estabilidad_out
out : reaccion_out

Link : evento_in evento_in@calibrador

Link : sensores_out@alarma sensores_out
```

Link : estabilidad_out@calibrador estabilidad_out
Link : reaccion_out@calibrador reaccion_out

Link : apertura_out@calibrador apertura_in@alarma
Link : movimiento_out@calibrador movimiento_in@alarma
Link : sonido_out@calibrador sonido_in@alarma
Link : alarma_out@alarma alarma_in@calibrador

4. Discusión

Antes que nada, es necesario hacer una aclaración. En este trabajo se cometió una estimación incorrecta del esfuerzo necesario, debido a que se estimó incorrectamente la complejidad del sistema real.

Por esta razón, no fue posible implementar la totalidad del sistema, de la manera que se había pensado originalmente.

Debido a esto, se decidió implementar uno solo de los componentes acoplados, en particular, el componente *Alarma*, dado que el otro componente servía como marco experimental, y en realidad no agregaba valor al modelado del sistema real, sino que era un metodo utilizado para la optimización del problema que se intentaba resolver. Es decir, la función del componente *Calibrador*, era generar los estímulos para introducir en la alarma, y debía ajustar los valores de los sensores de forma automática, para evitar que se tuvieran que crear archivos de eventos y comandos manualmente, dado que esta última tarea, es claramente complicada (si se desean generar situaciones complejas).

Sin embargo, para poder analizar la técnica de modelado DEVS, no es necesario construir un sistema completamente automatizado, y si bien es más costoso simular una situación más compleja (el costo reside en la construcción del archivo .ev), la simulación del componente *Alarma* es la misma (y nos sirve para analizar la técnica de modelado y simulación DEVS).

Por la misma razón mencionada, no fue posible realizar casos de test extensos, sino que por la falta de tiempo hubo que limitarse a realizar un testing superficial.

Nótese que dado que no se pudo terminar el componente *Calibrador*, es posible que las secciones referentes a los componentes que forman parte del mismo, se encuentren incompletas (no hubo tiempo de revisarlas).

Claramente, de disponer más tiempo, las tareas que faltarían realizar serían:

1. Generar casos de test más completos y realizar testing exhaustivo de los componentes implementados.
2. Implementar el componente *Calibrador*, para poder automatizar la simulación.
3. Generar casos de test y realizar testing sobre los nuevos componentes.
4. Informar sobre los cambios realizados.
5. Informar sobre la diferencia entre las situaciones analizadas manualmente y las situaciones que puede generar el *Calibrador*.

5. Conclusiones

Lamentablemente, por lo visto en la Sección 4, no se pudo analizar la simulación de la forma que se pretendía al comienzo de este trabajo, y por lo tanto, no se pudieron extraer conclusiones referentes al problema de optimización planteado.

Sin embargo, se pudo estudiar la técnica de modelado y simulación DEVS, y se pudo analizar las facilidades y complejidades que tiene aparejadas.

En este caso en particular, se pudo modelar con relativa facilidad, pero al momento de implementar el funcionamiento de los componentes su vieron varias dificultades. Algunas de ellas tuvieron que ver con el lenguaje que debía ser utilizado para implementar (C++), y lamentablemente, en varios momentos, los problemas no estuvieron en el modelado o la implementación de la semántica DEVS dentro del simulador (las funciones, δ_{ext} , δ_{int} y λ , sino con el lenguaje de programación en sí, y con los comportamientos complejos que se deseaban implementar (como por ejemplo, la generación aleatoria, según cierta distribución, de eventos).

El hecho de que el simulador esté ligado a un determinado lenguaje, limita además la forma en que se implementan los modelos, dado que cada lenguaje de programación tiene una estructura subyacente, además de sus propias limitaciones, que el modelador se ve incapacitado de evitar. Probablemente, si los modelos pudieran ser implementados, independientemente del lenguaje de programación (por ejemplo permitiendo programar en cualquier lenguaje, y luego linkeando contra una librería), seguramente algunas de las limitaciones encontradas se resolverían.

Otra cuestión es que si bien es posible implementarlo como un efecto secundario de los modelos, sería interesante que el simulador pudiera indicar la sucesión de estados y el pasaje de mensajes (por medio de las funciones δ_{ext} , δ_{int} y λ) de una manera más intuitiva y gráfica, para poder estudiar el comportamiento de la simulación con mayor facilidad. Claro que esto tiene su complejidad (que no debe ser menor) y que no tiene que ver con la técnica de modelado y simulación, sino con la implementación de la herramienta CD++.