DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1 - Traductor SD/DEVS

22 de Octubre de 2017 — Simulación de Eventos Discretos

Integrante	LU	Correo electrónico
Pedro Rodríguez	197/12	pedro3110.jim@gmail.com
Roberto Carlos	111/10	roberto@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - Pabellón I
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Argentina
Tel/Fax: (54 11) 4576-3359
http://exactas.uba.ar

Índice

1. Desarrollo

Para comenzar el proceso del desarrollo del traductor del que se habló en las secciones anteriores, decidimos basarnos en 3 (tres) modelos clásicos implementados en System Dynamics. Estos son: el modelo **Teacup** (modela una taza de café en una habitación, que se enfría progresivamente hasta alcanzar la temperatura ambiente), el modelo **SIR** (modela una población que sucumbe ante una infección que se propaga en la misma, y posteriormente se va recuperando, hasta que no queda ningún infectado), y el modelo **Lotka-Volterra** (modela el proceso de nacimientos y decesos de un población de linces y liebres, que interactúan entre sí con una dinámica de presa-depredador).

Mediante el estudio de estos tres problemas, analizamos cómo los diferentes elementos de cada archivo xmile pueden ser traducidos a elementos de un modelo acoplado DEVS (expresado en formato DEVSML), de forma tal que esta traducción, al ser ejecutada (en este caso en el simulador CD++), replique los resulados que se obtienen simulando el modelo original en simuladores se SD.

A continuación, exponemos cada uno de los modelos mencionados, mencionando cómo nos ayudaron en el desarrollo del traductor. Aprovecharemos que el modelo Teacup traducido contiene pocos archivos para mostrar el código generado, por el traductor, mientras que para la exposición del comportamiento de los otros modelos cuya traducción genera una mayor cantidad de archivos, utilizaremos el lenguage formal típico con que se expresan modelos DEVS.

1.1. Modelo Teacup

1.1.1. Modelo gráfico

Como se puede observar en la figura 1 se cuenta con 1 (un) stock, que llamamos *Teacup Temperature*, 2 (dos) constantes auxiliares (*Room Temperature* y *Characteristic Time*) y un flujo de salida (outflow) denominado *Heat Loss to Room* con origen el stock *Teacup Temperature* y destino vacío. Las flechas negras indican que el flujo de salida proveniente de *Teacup Temperature* utiliza dicha constantes auxiliares en su función interna para determinar el valor de dicho flujo de salida en cada instante en el tiempo. La flecha azul, indica que dicha función también utiliza el valor del stock *Teacup Temperature* para hacer este cálculo.

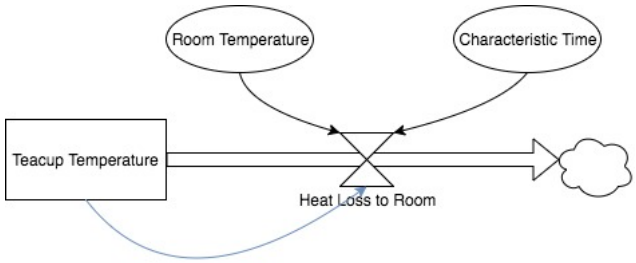


Figura 1: Modelo Teacup expresado en System Dynamics en formato gráfico

Como se puede observar en la figura 2, se utilizan tres tags distintos para los flujos de output e inflow (el tag **flow**), para las constantes auxiliares (el tag **aux**) y para los stocks (el tag **stock**). Asimismo, en cada flujo, se utiliza un tag **eqn** para mostrar la función utilizada por dicho *flow* (que puede ser tanto de input ó output) para agregarle o quitarle respectivamente unidades al stock sobre el cuál operan. Por otro lado, observamos que en los stocks y variables auxiliares se utiliza el tag **eqn** para mostrar el valor inicial de dicho stock ó variable auxiliar. Vale la pena aclarar aquí que en el caso de la variable auxiliar, esta podría ser tanto constante como variable. En este caso en particular, las variables auxiliares son todas constantes, con lo cuál en el tag **eqn** el valor inicial que figura en dicho tag también es el mismo valor que tendrá dicha variable durante toda la simulación.

A partir de estas observaciones, decidimos representar este mismo modelo en el formalismo DEVS, de la forma en que se muestra en el diagrama de la figura 3

En el diagrama de la figura 3 se debe observar lo siguiente: a cada tag **stock** corresponden: 1 (un) atómico de nombre *nombreStock + Integrator* y 1 (un) atómico Ftot, que acumula todos los inflows y outflows que operan sobre dicho stock para luego efectuar el cálculo final (que es una resta) entre los inflows y los outflows, para determinar la variación de unidades que tendrá dicho stock.

```
<model>
    <variables>
        <flow name="Heat Loss to Room">
            <doc>Heat Loss to Room</doc>
            <eqn>("Teacup Temperature"-"Room Temperature")/"Characteristic Time"</eqn>
        </flow>
        <aux name="Room Temperature">
            <doc>Ambient Room Temperature</doc>
            <eqn>70</eqn>
        </aux>
        <stock name="Teacup Temperature">
            <doc>The average temperature of the tea and the cup</doc>
            <outflow>"Heat Loss to Room"</outflow>
            <eqn>180</eqn>
        </stock>
        <aux name="Characteristic Time">
            <eqn>10</eqn>
        </aux>
    </variables>
</model>
```

Figura 2: Modelo Teacup exprsado en System Dynamics en formato XMILE

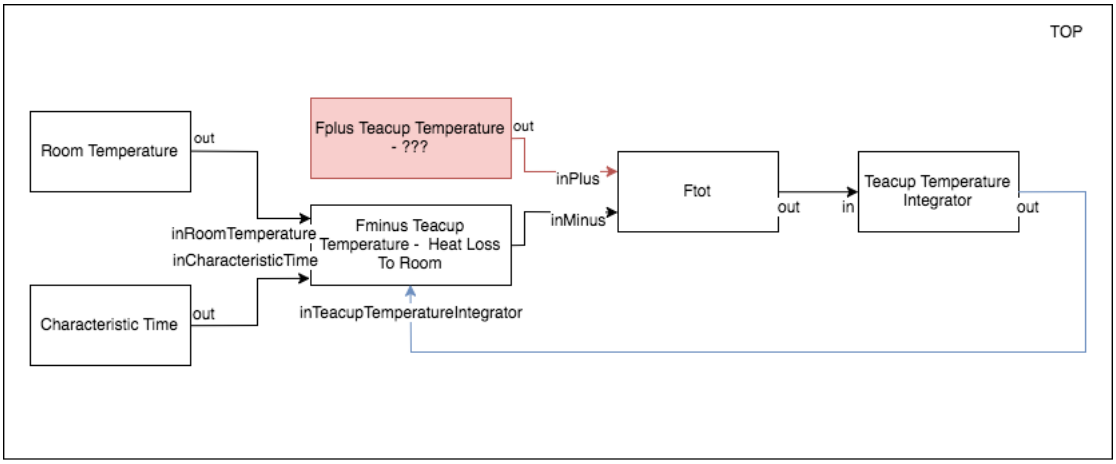


Figura 3: Modelo Teacup expresado en DEVS en formato gráfico

Por otro lado, a cada flujo (en este caso *Heat Loss to Room* corresponden: 1 (un) atómico de nombre *Fminus + nombre del stock sobre el que opera restándole unidades + nombreFlujo*, cuyo output llega a la Ftot correspondiente al integrador sobre el que este opera restándole unidades; y 1 (un) atómico de nombre *Fplus + nombre del stock sobre el que opera sumándole unidades + nombreFlujo*. En el ejemplo de la figura 3, el flujo Fplus no tiene sentido ser, porque no hay ningún inflow que actúe sobre el stock *Teacup Temperature*. Es por esto que lo marcamos en rojo para evidenciar que este stock no debería estar ahí porque no tiene ningún rol en el modelo.

Finalmente, se observa que a cada tag **aux** hacemos corresponder un atómico DEVS cuya salida llega a los atómicos correspondiente al flujo *Heat Loss to Room* que utilizan dichar variables para hacer sus cálculos. En este ejemplo, sólo se conectan con el *FminusTeacupTemperature - Heat Loss to Room*, porque el stock *Teacup Temperature* no tiene ningún inflow.

También podemos observar en el diagrama la línea azul, que se corresponde con la línea azul del diagrama 1. De esta forma, mostramos que el atómico *FminusTeacupTemperature - Heat Loss to Room* utiliza también el output del atómico *Teacup Temperature Integrator* para realizar sus cálculos.

Llegados a esta instancia, nos preguntamos qué pasaría con nuestro mapeo a DEVS si más de un outflow operara sobre *Teacup Temperature*. Así, decidimos pensar cómo sería la traducción gráfica del siguiente modelo (figura 4), que modela un segundo outflow que opera sobre el stock *Teacup Temperature* (en el ejemplo, decimos que la taza no sólo se enfría de acuerdo al rate de pérdida de calor de la taza contra la habitación, sino también contra una máquina enfriadora *Cooling Machine* que agregamos - como podría ser por ejemplo un ventilador -). Llegamos a la conclusión que el modelo DEVS correspondiente a este nuevo modelo debería ser el de la figura 5.

Como se puede ver, agregamos el atómico *FminusTeacupTemperature - Heat Loss to Cooling Machine*, con los mismos inputs que *FminsuTeacupTemperature - Heat Loss to Room*, pero con la novedad de que los outputs de estos atómicos llegan a distintos puertos de entrada del atómico Ftot que teníamos antes. Estos

puertos de entrada están nombrados de acuerdo al nombre del flujo que opera sobre el stock correspondiente al Ftot en cuestión (en este caso, *Heat Loss to Room* y *Heat Loss to Cooling Machine* ambos operan sobre *Teacup Temperature* en carácter de quitadores de unidades), para que el modelo gráfico que más claro.

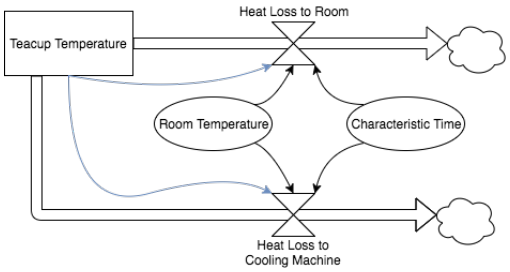


Figura 4: Modelo Teacup (versión 2) expresado en SD en formato gráfico

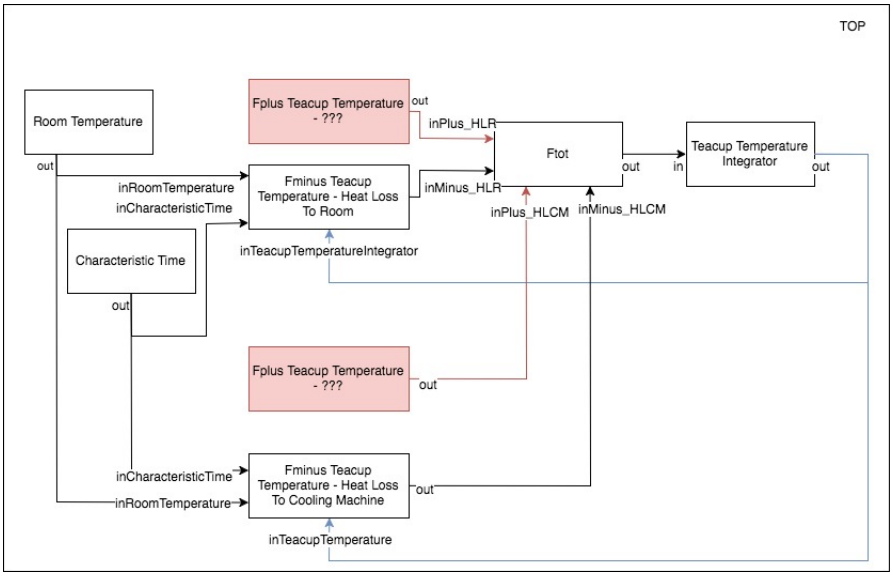


Figura 5: Modelo Teacup (versión 2) expresado en DEVS en formato gráfico

Finalmente, pensamos en una forma de modularizar el modelo acoplado tan complejo que quedó tras las conversiones, tanto en la versión original como en la versión 2 del modelo Teacup. Lo que obtuvimos, fue lo que se muestra en las figuras 6 y 7.

Aquí se puede observar que definimos un modelo acoplado por cada **stock**, que engloba al integrador correspondiente, a la Ftot correspondiente, a los Fminus's correspondientes (outflows que tienen como origen al **stock** en cuestión), y a los Fplus's correspondientes (inflows que contienen como origen al **stock** en cuestión). Vale la pena aclarar aquí que dichos, inflows y outflows que operan sobre cada **stock** corresponden cada uno de ellos a un **flow** en el modelo expresado en System Dynamics, y es interesante notar que visualmente es muy simple e intuitivo ver cómo se relacionan cada uno de los modelos expresados en SD con su contraparte en DEVS.

Todos estos serán factores muy importantes a tener en cuenta a la hora de desarrollar la primera versión del traductor, ya que querremos que esta sea lo más general posible, y que sea fácilmente modificable para ser capaz de representar nuevos modelos cada vez más complejos a medida que el proyecto avance.

1.1.2. Modelo en código

En esta sección, primeramente mostraremos el modelo .ma que deseamos ser capaces de generar a partir del archivo .devsml generado por nuestro traductor a partir del archivo .xmile del modelo Teacup que ya mencionamos previamente. Por razones de tiempo y para simplificar las cosas en esta primera aproximación al problema, sólo trabajaremos con la versión aplanada del modelo. Es decir, no intentaremos generar varias capas de acoplados, sino un sólo acoplado Top, que contendrá a todos los atómicos que hagan falta.

Como se puede observar en la figura 8, el archivo .ma consta de la definición del componente acoplado Top mediante [top], la definición de todos los componentes (modelos atómicos o acoplados, aunque en este

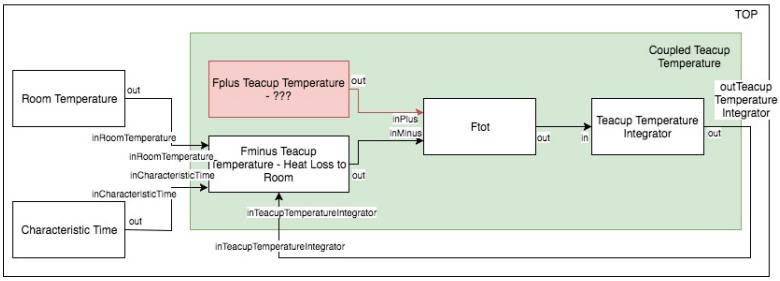


Figura 6: Modelo Teacup expresado en DEVS en formato gráfico utilizando varios niveles de acoplamiento

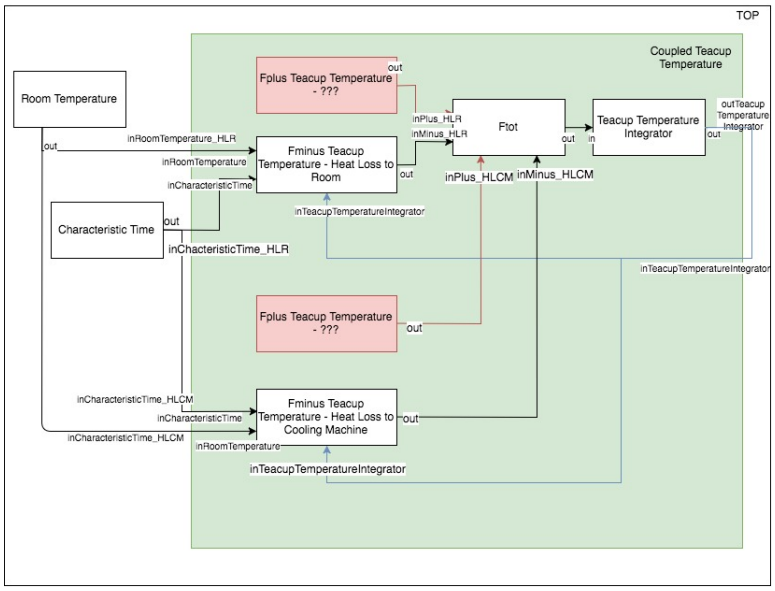


Figura 7: Modelo Teacup (versión 2) expresado en DEVS en formato gráfico utilizando varios niveles de acoplamiento

```
[top]
components : teacupTemperatureIntegrator@QSS1 ftTeacupTemperature@Ftot
fmTeacupTemperatureHeatLossToRoom@FminusTeacupTemperature roomTemperature@Cte characteristicTime@Cte

% Puertos: Input de parametros. Output de variables de interes
in : inRoomTemperature inCharacteristicTime

out : outTeacupTemperature

% Links inputs a constantes (conexiones con el top model)
link : inRoomTemperature inValue@roomTemperature
link : inCharacteristicTime inValue@characteristicTime

% Links constantes a f's que las usan
link : out@characteristicTime inCharacteristicTime@fmTeacupTemperatureHeatLossToRoom
link : out@roomTemperature inRoomTemperature@fmTeacupTemperatureHeatLossToRoom

% Links acoplado minimal integrador Teacup Temperature
link : out@fmTeacupTemperatureHeatLossToRoom inMinus@ftTeacupTemperature
link : out@ftTeacupTemperature in@teacupTemperatureIntegrator

% Links modelo
link : out@teacupTemperatureIntegrator inTeacupTemperatureIntegrator@fmTeacupTemperatureHeatLossToRoom

% Liks output variables de interes
link : out@teacupTemperatureIntegrator outTeacupTemperature

% Integradores
[teacupTemperatureIntegrator]
x0 : 180
dQRel : 1e-2
dQMin : 1e-4
```

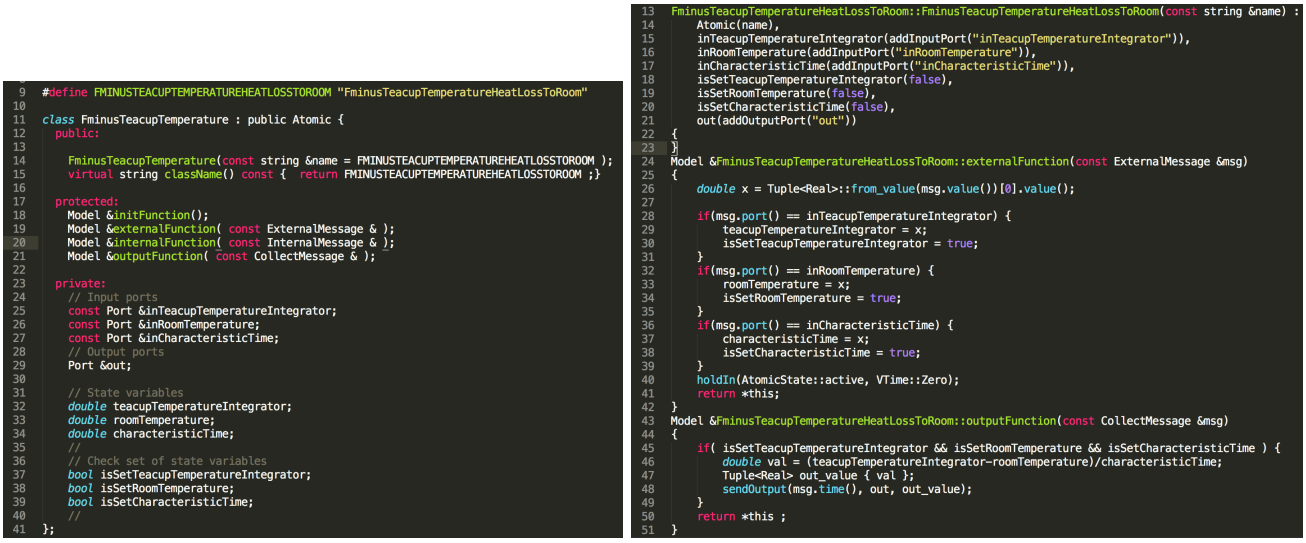
Figura 8: Archivo .ma correspondiente al modelo Teacup para simular el modelo en el simulador CD++

ejemplo, como dijimos van a ser todos atómicos) que forman parte del acoplado Top, los puertos de entrada

y salida del acoplado (en este caso, la entrada son las variables constantes auxiliares, y la salida es la úncia variable de interés del modelo - la temperatura de la taza -) y los links entre cada uno de los componentes del acoplado.

Compárese la figura 3 con la figura 8, y observése el mapeo 1-1 que hay entre cada línea del archivo .ma con el diagrama correspondiente.

Ahora bien, los atómicos utilizados en este modelo, deberán tener un comportamiento. Obviamente, queremos también automatizar la generación de archivos que le den comportamiento a estos atómicos, para que el modelo .ma mostrado más arriba pueda ser ejecutado. Exponemos a continuación las partes más relevantes del código C++ que nuestro traductor genera junto con el .ma para darle el comportamiento deseado a cada atómico.



(a) Archvo .h

(b) Archivo .cpp

Figura 9: Código relevante de los archivos .h y .cpp generados para el atómico FminusTeacupTemperature

(a) Archvo Ftot.cpp

(b) Archivo Cte.cpp

Figura 10: Código relevante de los archivos .cpp generados para los atómicos Cte y Ftot

1.1.3. Generación de modelo ejecutable en CD++

Para poder generar los archivos .ma, .h y .cpp que mencionamos en la sección anterior, previamente debemos generar un archivo .devsml que contenga toda la información necesaria para este fin.

```
<components>

  <!-- Stock : Teacup Temperature Integrator -->
  <atomicRef name="ft Teacup Temperature" model="Ftot">
    <!-- Puertos del atomico -->
    <port name="inMinus" type="in" />
    <port name="inPlus" type="in" />
    <port name="out" type="out" />
    <!-- State variables del atomico -->
    <parameter name="plus" value="0" />
    <parameter name="minus" value="0" />
  </atomicRef>
  <atomicRef name="Teacup Temperature Integrator" model="QSS1">
    <parameter name="x0" value="180" />
    <parameter name="dQRel" value="1e-2" />
    <parameter name="dQMin" value="1e-4" />
  </atomicRef>
  <!-- Flow : Heat Loss to Room -->
  <atomicRef name="fm Teacup Temperature - Heat Loss to Room" model="Fminus">
    <!-- Puertos del atomico -->
    <port name="Teacup Temperature Integrator" type="in" />
    <port name="Room Temperature" type="in" />
    <port name="Characteristic Time" type="in" />
    <port name="out" type="out" />
    <!-- Funcion usada en output function del atomico -->
    <parameter name="function" value="(Teacup Temperature Integrator-Room Temperature)
    /Characteristic Time" />
  </atomicRef>
  <!-- Parameters / Constantes -->
  <atomicRef name="Characteristic Time" model="Cte">
    <port name="value" type="in" />
    <port name="out" type="out" />
    <parameter name="value" value="10" />
  </atomicRef>
  <atomicRef name="Room Temperature" model="Cte">
    <port name="value" type="in" />
    <port name="out" type="out" />
    <parameter name="value" value="70" />
  </atomicRef>
  </components>
```

(a) Atómicos 1-1 para cada Flow

```
<components>

  <!-- Stock : Teacup Temperature Integrator -->
  <atomicRef name="ft Teacup Temperature" model="Ftot">
    <!-- Puertos del atomico -->
    <port name="inMinus" type="in" />
    <port name="inPlus" type="in" />
    <port name="out" type="out" />
    <!-- State variables del atomico -->
    <parameter name="plus" value="0" />
    <parameter name="minus" value="0" />
  </atomicRef>
  <atomicRef name="Teacup Temperature Integrator" model="QSS1">
    <parameter name="x0" value="180" />
    <parameter name="dQRel" value="1e-2" />
    <parameter name="dQMin" value="1e-4" />
  </atomicRef>
```

(b) Atómicos 1-1 para cada Stock

Figura 11: Parte relevante del código .devsml generado por cada Constante, Stock y Flow

```
<internal_connections>

  <!-- Link : Cte's a f's que las usan -->
  <!-- Cte :  Room Temperature -->
  <connection component_from="Room Temperature" port_from="out" component_to="fm Teacup Temperature -
   Heat Loss to Room" port_to="Room Temperature"/>
  <!-- Cte :  Characteristic Time -->
  <connection component_from="Characteristic Time" port_from="out" component_to="fm Teacup
  Temperature - Heat Loss to Room" port_to="Characteristic Time"/>
  <!-- Links modelo acoplado minimal integrador Teacup Temperature -->
  <connection component_from="fm Teacup Temperature" port_from="out" component_to="ft Teacup
  Temperature" port_to="inMinus" />
  <connection component_from="ft Teacup Temperature" port_from="out" component_to="Teacup
  Temperature Integrator" port_to="in" />
  <!-- Links modelo (las flechas azules en el modelo grafico de SD) -->
  <connection component_from="Teacup Temperature Integrator" port_from="out" component_to="fm Teacup
  Temperature - Heat Loss to Room" port_to="Teacup Temperature Integrator" />
  </internal_connections>
```

(a) Conexiones internas

```
<!-- Links inputs a constantes -->
<!-- external input -->
<external_input_connections>
  <connection component_from="top" port_from="Characteristic Time" component_to="Characteristic Time"
   port_to="value" />
  <connection component_from="top" port_from="Room Temperature" component_to="Room Temperature"
  port_to="value" />

</external_input_connections>
```

(b) Conexiones externas

```
<coupled name="top" model="Coupled" >
  <!-- Input / Output -->
  <!-- Inputs -->
  <inputs>
    <port name="Characteristic Time" />
    <port name="Room Temperature" />
    </inputs>
  <!-- Outputs -->
  <outputs>
    <port name="Teacup Temperature" />
  </outputs>
  <components>

    <!-- Stock : Teacup Temperature Integrator -->
```

(c) Puertos de entrada/salida del modelo Top

Figura 12: Parte relevante del código .devsml generado por cada conexión y para los puertos del modelo Top

1.2. Modelo Formal

A continuación, exponemos formalmente los modelos atómicos utilizados en el acoplado Top, con la descripción de su comportamiento interno.

- **Cte** : $roomTemperature \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
  $X = \{inValue\}$
  $S = \{value\}$
  $Y = \{out\}$
  $\delta_{int}(\langle value \rangle, e, x) = \{\}$
  $\delta_{ext} = \{value := xxx\}$
  $t_a(s) = \{\}$

- **Cte** : $characteristicTime \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
  $X = \{inValue\}$
  $S = \{value\}$
  $Y = \{out\}$
  $\delta_{int} = \{\}$
  $\delta_{ext} = \{value := xxx\}$
  $t_a(s) = \{\}$

- **Fminus** : $fmTeacupTemperatureHeatLossToRoom \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
  $X = \{\}$
  $S = \{\}$
  $Y = \{\}$
  $\delta_{int} = \{\}$
  $\delta_{ext} = \{\}$
  $t_a = \{\}$

- **Ftot** : $ftTeacupTemperature \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
  $X = \{\}$
  $S = \{\}$
  $Y = \{\}$
  $\delta_{int} = \{\}$
  $\delta_{ext} = \{\}$
  $t_a = \{\}$

- **QSS1** : $teacupTemperatureIntegrator \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
  $X = \{\}$
  $S = \{\}$
  $Y = \{\}$
  $\delta_{int} = \{\}$
  $\delta_{ext} = \{\}$
  $t_a = \{\}$

1.3. Propuesta de algoritmo traductor

A continuación, y basándonos en lo aprendido durante la construcción del traductor para el modelo Teacup, detallaremos un algoritmo inicial y muy básico que pensamos debería servir para traducir archivos .xmile a archivos .devsml, a partir de la captura de toda la información relevante contenida en el archivo .xmile.

En esta versión inicial, nos restringimos a traducir a modelos DEVS aplanados solamente. De esta manera, se simplifican algunos detalles de la implementación del traductor. Además de estos, nos restringimos a archivos .xmile que también estén aplanados. Es decir, no lidiamos con la presencia de diferentes módulos con modelos internos, y la interacción de los modelos internos a través de los módulos en lugar de directamente. De este problema nos encargaremos más adelante, cuando estudiemos el modelo Lotka-Volterra.

Para este fin, dividiremos la explicación del algoritmo en partes, para explicar las figuras 11(a), 11(b), 12(a), 12(b) y 12(c), explicando como cada una de estas se corresponde con las líneas del archivo .ma expuesto en la figura 8.

1.3.1. Traducción de Stocks

1.3.2. Traducción de Flows

1.3.3. Generación de puertos es E/S del modelo DEVS

1.3.4. Generación de conexiones internas del modelo DEVS

1.3.5. Generación de conexiones externas del modelo DEVS

1.4. Modelo SIR

1.4.1. Modelo gráfico

En la figura **??** se observa en color azul los atómicos Fplus y Fminus asociados al flujo *Succumbing*, mientras que en naranja se observan los atómicos Fplus y Fminus asociados al flujo *Recovering*. En violeta se pueden ver las conexiones que representan de qué manera los diferentes atómicos correspondientes a cada uno de los flujos utilizan para realizar sus cálculos internos el output de *InfectedIntegrator*, correspondiéndose cada una de estas flechas con una de las flechas del modelo gráfico mostrado en la figura **??**. Como ya explicamos anteriormente, en este ejemplo sólo trabajaremos con la versión aplanada del modelo DEVS, para simplificar levemente el problema de traducción.

Nuevamente, dejamos en rojo los atómicos que no tienen cabida en el modelo, pero que igualmente exponemos, para dejar explícito que esos atómicos podrían estar si hubiera un inflow/outflow que se les corresponda, pero que en este ejemplo no lo están.

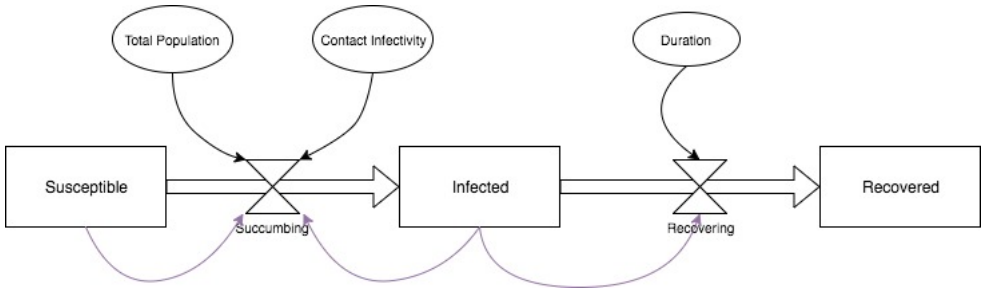


Figura 13: Modelo SIR expresado en System Dynamics en formato gráfico

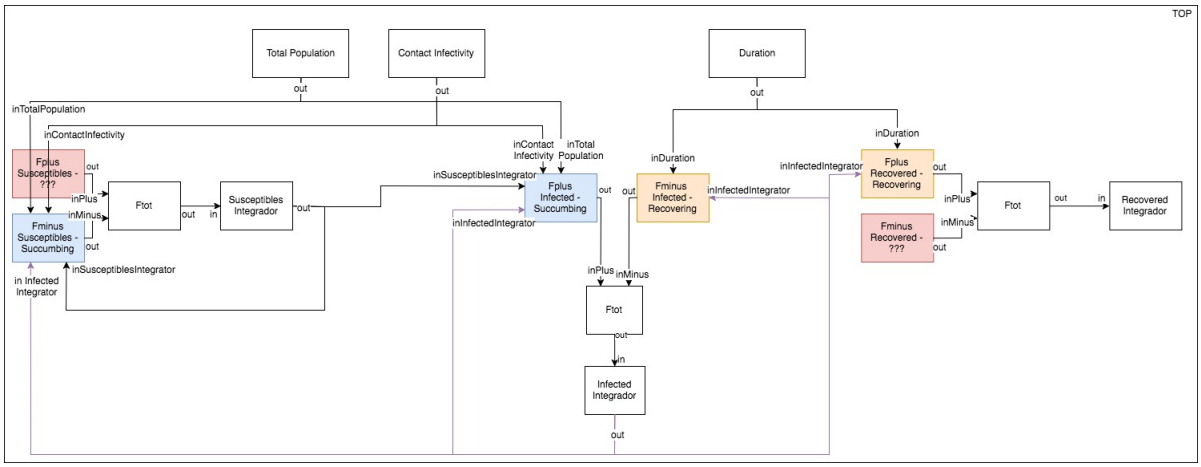


Figura 14: Modelo SIR expresado en DEVS en formato gráfico

1.4.2. Modelo formal

A continuación, exponemos formalmente los modelos atómicos utilizados en el acoplado Top, con la descripción de su comportamiento interno.

- **Cte** : $totalPopulation \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
- **Cte** : $contactInfectivity \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$

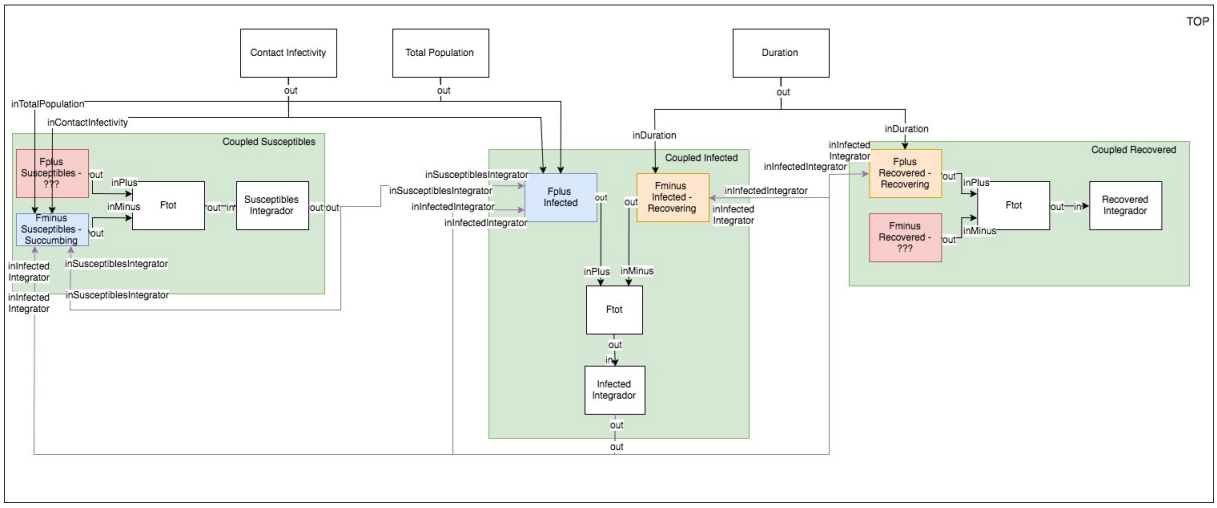


Figura 15: Modelo SIR expresado en DEVS en formato gráfico con varios niveles de acoplamiento

- **Cte** : $duration \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
- **Fminus** : $fmSusceptiblesSuccumbing \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
- **Ftot** : $ftSusceptibles \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
- **QSS1** : $susceptiblesIntegrator \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
- **Fplus** : $fpInfectedSuccumbing \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
- **Fminus** : $fmInfectedRecovering \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
- **Ftot** : $ftInfected \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
- **QSS1** : $infectedIntegrator \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
- **Fplus** : $fpRecoveredRecovering \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
- **Ftot** : $ftRecovered \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
- **QSS1** : $recoveredIntegrator \rightarrow \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$

1.4.3. Generación de modelo ejecutable en CD++

1.5. Modelo Lotka-Volterra