Introducción al Modelado de Sistemas
Capítulo 5

Camilo Rueda

27 de agosto de 2013

1. Refinamiento

Cada máquina que se ha descrito hasta ahora incluye todas las observaciones que se plantean para el sistema. Esta, sin embargo, no es una buena estrategia de diseño para modelos complejos. Entender un sistema complejo es un proceso en el que paulatinamente se consideran observaciones más detalladas. Estas solamente se tienen en cuenta una vez se ha logrado comprender bien el comportamiento del sistema con respecto a otras observaciones más globales. Lo que se construye, entonces, no es un modelo del sistema sino una *jerarquía* de modelos organizada en niveles. Los modelos en los niveles superiores representan visiones globales, muy generales, del comportamiento del sistema, mientras que los de niveles inferiores dan cuenta de los detalles precisos. Un modelo de un nivel inferior se dice que *refina* (o es un *refinamiento* de) un modelo en el nivel superior, como se muestra en la figura 1. En los niveles superiores los modelos se llaman *abstracciones* y en los inferiores *modelos concretos*.

Intuitivamente, un refinamiento es un modelo que observa *el mismo sistema* que su abstracción, pero con más detalle. Es decir, la abstracción y el modelo concreto son dos puntos de vista sobre el mismo sistema, luego sus comportamientos deberían ser coherentes. Pero, qué es el *comportamiento* de un modelo? En la siguiente sección consideramos la respuesta a esta pregunta.

1.1. Comportamiento de un sistema

Los modelos de sistemas que se han considerado hasta ahora constan de una máquina y de un contexto. El comportamiento del sistema se describe por las observaciones que resultan de la ejecución de la máquina. Supongamos un modelo (máquina) con tres variables numéricas x, y, z y cuatro eventos llamados $Ev1$, $Ev2$, $Ev3$ y $Ev4$. Las observaciones de este sistema se pueden representar en un gráfico como el de la figura 2. En este gráfico (que en matemáticas se llama *grafo*), cada observación (o *estado*) se representa en los óvalos, y cada *transición* entre observaciones, ocasionada por la ocurrencia de un evento,

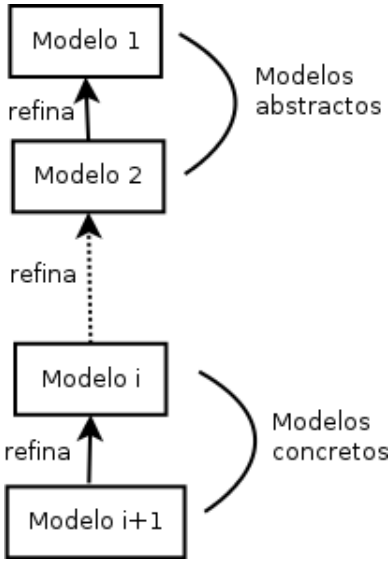

Figura 1: Jerarquía de modelos

se representa como una flecha. Por ejemplo, la transición de la observación del estado 1 a la observación del estado 2 se produce cuando ocurre un evento llamado $Ev1$. Esta transición modifica los valores de las tres variables x, y, z, asignándoles 5,11 y 1, respectivamente. El comportamiento se lee de la siguiente manera: "cuando el sistema está en el estado 1 y ocurre el evento $Ev1$, pasa al estado 2 que representa la observación $x = 5, y = 11, z = 1$. El comportamiento del sistema incluye entonces las observaciones y de qué modo estas cambian cuando ocurre un evento.

Por ejemplo, el comportamiento de la máquina $huellaMaq$ definida al comienzo del capítulo 3, está dado por el grafo (parcial) de transiciones de la figura 3. El sistema comienza en el estado señalado por la flecha, con la observación $luz = apagado, huella = libre, fase = 1$, que corresponde a la inicialización (no se muestra la variable $cont$). En ese estado el único evento que puede ocurrir es $ponerHuella$. La ejecución de este evento corresponde a la transición de ese mismo nombre, que llega al estado con la observación $luz = apagada, huella = ocupado, fase = 2$. En este estado hay dos transiciones posibles, que corresponden a la ocurrencia de alguno de los eventos $luzVerde$ y $luzRoja$. El sistema sigue alguna de esas transiciones (pero no las dos!) para continuar el proceso.

Cuando hay una jerarquía de modelos, debe haber coherencia entre lo que sucede cuando se sigue el grafo de la abstracción y lo que sucede cuando se sigue el del refinamiento, como analizaremos en la siguiente sección.

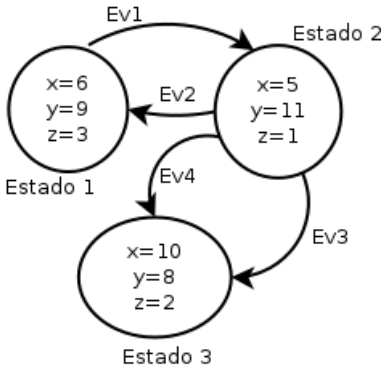

Figura 2: Observaciones sobre las variables x, y, z

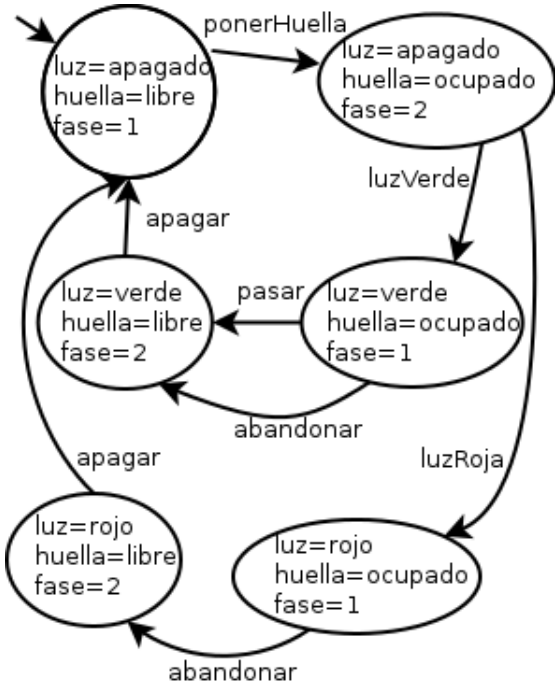

Figura 3: Grafo de la máquina *huellaMaq*

1.2. Simulación de comportamientos

Para que haya coherencia entre ellos, los grafos de la abstracción y del modelo concreto deben estar relacionados de alguna manera. La relación que tiene uno con otro se llama *simulación*, que no es otra cosa que el juego infantil de "seguir al líder". En este juego, el líder realiza acciones que los seguidores deben copiar. Por ejemplo, si el líder salta al borde de un muro abriendo los brazos, el seguidor debe también hacerlo. Note que en este juego nunca el seguidor tendrá que hacer *exactamente* todo lo que se puede observar del líder. Por ejemplo, el líder puede haber puesto en el borde del muro antes el pie izquierdo que el derecho, mientras que el seguidor puede haber puesto los dos pies al tiempo. Lo esencial es que lo más significativo que se observe del líder (saltar al muro y abrir los brazos) se copie.

La coherencia entre la abstracción y el refinamiento consiste en poder realizar este juego con el modelo concreto como líder y con la abstracción como seguidor. Es decir, cada acción de la máquina del refinamiento debe poder copiarla la máquina de la abstracción. Para que la copia sea válida, deben resultar compatibles, como en el juego, lo más significativo que se observe de las acciones del refinamiento. La noción fundamental es la de *compatibilidad*. Lo que se observa en el modelo abstracto corresponde a un cierto punto de vista sobre el sistema. Lo que se observa en el concreto corresponde a otro punto de vista. Como cada observación está determinada por una colección de *variables*, los puntos de vista diferentes entre abstracción y modelo concreto pueden dar lugar a variables diferentes en ambos modelos. Por ejemplo, puede suceder que el modelo abstracto observe el estado del sistema mediante una variable x, mientras que el modelo concreto lo observe con dos variables y, z. Qué quiere decir en estas circunstancias *compatibilidad*? La compatibilidad se define mediante una propiedad que deben satisfacer en cada estado los valores de las variables de la abstracción y del modelo concreto. Esta propiedad se llama *invariante de encadenamiento.* Por ejemplo, un posible invariante de encadenamiento podría establecer que

$$x = y + z$$

En este caso, los estados de la figura 4 serían compatibles porque los valores que contienen cumplen el invariante.

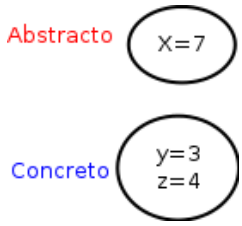

Figura 4: Estados compatibles para el invariante $x = y + z$

Podemos entonces replantear el juego de la simulación de la siguiente manera:

1. Comprobar que el esatdo inicial de la abstracción y el modelo concreto cumplen el invariante de encadenamiento.
2. Ejectutar una transición cualquiera en el modelo concreto. Por ejemplo, ejecutando un evento $ev1$, obteniendo el estado E.
3. En el modelo abstracto:
   - Si el evento $ev1$ existe, ejecutarlo. Sea en este caso E_1 el estado que se obtiene.
   - Si el evento $ev1$ no existe, permanecer en el estado en que está (no hacer transición). Sea E_2 ese estado.
4. Verifique que los valores de las variables en los estados E y en el que se haya obtenido en la abstraccin en el paso anterior, E_1 o E_2, cumplan el invariante de encadenamiento
5. Repita los pasos $2, 3, 4$ hasta que haya considerado todas las transiciones del modelo concreto.

Si para un modelo abstracto M_1 y uno concreto M_2, es posible hacer el anterior juego de simulación, entonces se dice que M_2 es un refinamiento válido de M_1.

1.3. Ejemplo de refinamiento

Suponga el sistema de acceso a un lugar mediante la huella, que se definió en el capítulo 3. Se dispone de un *banco de huellas* que determina si la huella se acepta o no para pasar la puerta. Es decir, si la huella que se pone en el lector es aceptable, la luz verde se prende. De lo contrario, se prende la roja. Esto se puede definir como un refinamiento del modelo del capitulo 3. Hay un concepto nuevo, que es el de "huella". Lo definimos en el contexto:

context $huellaCtx2$ **extends** $huellaCtx$
sets
$HUELLAS$
constants
$aceptadas$
$rechazadas$
$ninguna$

axioms
@$axr1$ $aceptadas \subseteq HUELLAS$
@$axr2$ $rechazadas \subseteq HUELLAS$
@$axr3$ $ninguna \in HUELLAS$
@$axr4$ $ninguna \notin aceptadas$
@$axr5$ $ninguna \notin rechazadas$
@$axr6$ $aceptadas \neq \varnothing$
@$axr7$ $rechazadas \neq \varnothing$
@$axr8$ $aceptadas \cap rechazadas = \varnothing$
end

El axioma @$axr3$ define una huella especial, que representa "ausencia de huella". Es decir, es un valor que se puede usar cuando nadie ha puesto su huella en el lector. Los axiomas @$axr6$ y @$axr7$ dicen que debe haber al menos una huella entre las que se aceptan

y una entre las que se rechazan. El axioma @$axr8$ expresa que no puede haber a la vez una huella aceptada y rechazada.

El refinamiento define una nueva observación, que son las huellas de las personas que ya han cruzado la puerta. Además, se observa la huella que se ha puesto en el lector (cuando se haya puesto alguna):

machine $huellaMaqRef$ **refines** $huellaMaq$ **sees** $huellaCtx2$
variables
$pasaron$ // las huellas de los que ya pasaron
$huellaPuesta$ // la huella puesta en el lector

invariants
@$invr1$ $pasaron \subseteq HUELLAS$
@$invr2$ $pasaron \subseteq aceptadas$
@$invr3$ $huellaPuesta \in HUELLAS$
@$invr4$ $huellaPuesta \notin pasaron$
@$invr5$ $fase = 1 \land luz = verde \Rightarrow huellaPuesta \in aceptadas$

El invariante @$invr2$ garantiza que las huellas de los que ya pasaron eran aceptables. El invariante @$invr4$ establece que la huella puesta en un momento dado no puede ser de alguien que ya pasó. El invariante @$invr5$ dice que si la luz esta en verde es porque alguien puso una huella aceptable.

events
event INITIALISATION
then
@$accr1$ $huellaPuesta := ninguna$
@$accr2$ $pasaron := \varnothing$
end

event $ponerHuella$(**abstracto**)
when
@$grd1$ $fase = 1$
@$grd2$ $luz = apagado$
then
@$acc1$ $huella := ocupado$
@$acc2$ $fase := 2$
end

event $ponerHuella$ **extends** $ponerHuella$
any
h
when
@$grdr1$ $h \in HUELLA$
@$grdr2$ $h \notin pasaron$
@$grdr3$ $h \neq ninguna$
@$grdr4$ $huellaPuesta = ninguna$
then
@$accr1$ $huellaPuesta := h$
end

El evento extendido $ponerHuella$ escoge una huella cualquiera, que no sea de las que ya pasaron, y la pone en el lector. Note que no es necesario mencionar las variables de la abstracción porque un evento extendido se asume que agrega cosas. Es decir, lo que hubiera en la abstracción se asume como si se hubiera incluido también en el evento.

```
event luzVerde(abstracto)
when
@grd1  fase = 2
@grd2  huella = ocupado
@grd3  luz = apagado
then
@acc1  luz := verde
@acc2  fase := 1
end
```

```
event luzVerde extends luzVerde
when
@grdr1  huellaPuesta ∈ aceptadas
end
```

El evento extendido $luzVerde$ agrega la condición de que la huella que se haya puesto debe ser una de las que se reconoce como aceptables.

```
event luzRoja(abstracto)
when
@grd1  fase = 2
@grd2  huella = ocupado
@grd3  luz = apagado
then
@acc1  luz := rojo
@acc2  fase := 1
end
```

```
event luzRoja extends luzRoja
when
@grdr1  huellaPuesta ∈ rechazadas
end
```

El evento extendido $luzRoja$ agrega la condición de que la huella que se haya puesto debe ser una de las que se reconoce como inaceptables.

```
event pasar(abstracto)
when
@grd1  fase = 1
@grd3  luz = verde
then
@acc1  huella := libre
@acc2  fase := 2
@acc3  cont := cont + 1
end
```

```
event pasar extends pasar
then
@actr1  huellaPuesta := ninguna
@actr2  pasaron := pasaron ∪ {huellaPuesta}
end
```

El evento extendido $pasar$ no agrega condiciones sino acciones. Garantiza que ya no haya

huella puesta y que la huella que estaba se agregue a las que ya pasaron.

```
event abandonar(abstracto)
when
@grd1  fase = 1
@grd3  luz ≠ apagado
then
@acc1  huella := libre
@acc2  fase := 2
end
```

```
event abandonar extends  abandonar
then
@actr1  huellaPuesta := ninguna
end
```

El evento extendido *abandonar* agrega una acción que garantiza que ya no haya huella puesta.