Introducción al Modelado de Sistemas

Camilo Rueda

20 de enero de 2016

1. Introducción

El presente ensayo trata sobre una manera de representar sistemas. En su definición más general, un sistema es una totalidad compuesta de una colección de partes. La palabra "totalidad" significa que hay un cierto comportamiento global que se puede identificar de la interacción de las partes. Es decir, consideramos un sistema como un agregado con tres caractersticas fundamentales:

- Sus partes están *relacionadas*
- Tiene un comportamiento *observable*
- Es *dinámico*: al observarlo en varios momentos se identifican *cambios*

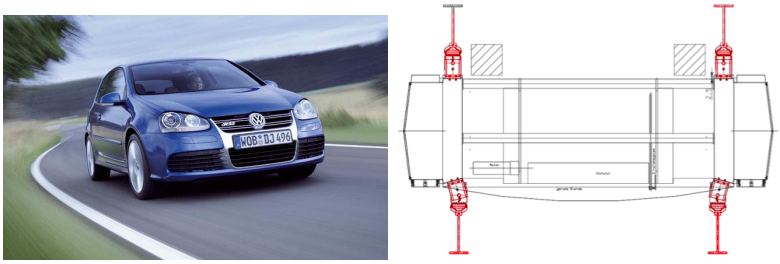
La noción fundamental en esta definicin informal es la de *observación*: observamos que el sistema está compuesto de partes, que hay ciertas regularidades (comportamiento) y también que el sistema se transforma.

Como es en general difícil o a veces imposible, experimentar directamente sobre el sistema, la manera que tiene la ingeniería para tratar de comprenderlo es *representarlo* en algún contexto diferente en el que se pueda experimentar con él. Estas representaciones se llaman *modelos*. Los modelos son *maquetas* del sistema sobre las que el ingeniero puede intervenir para determinar propiedades. Una maqueta es en general muy diferente al sistema que representa, pero se puede identificar con precisión en ella cuáles aspectos o características del sistema son *igualmente* observables en la maqueta y en el sistema real.

Los modelos pueden tomar diversas formas. Pueden ser representaciones físicas, como las maquetas de construcciones que hacen los arquitectos o los ingenieros civiles, en los que lo que los diferencia del sistema real es el material del que se componen. La maqueta de un puente, por ejemplo, puede estar construida con cartón, madera o plástico. Las maquetas pueden también ser dibujos, como los bocetos de los ingenieros mecánicos o los circuitos de los ingenieros electrónicos. En la figura 1 se muestra un sistema físico (un carro) y un modelo particular de él que se centra en la observación del sistema de ejes. En este modelo se han eliminado muchos aspectos del sistema que no se consideran relevantes. Por ejemplo,

no aparece la forma del carro, ni el motor. El modelo se dice que *hace abstracción*, es decir, ignora, esos aspectos. En este mismo sentido se dice que un modelo es *abstracto* cuando representa solamente unos pocos aspectos del sistema, que se consideran particularmente importantes.

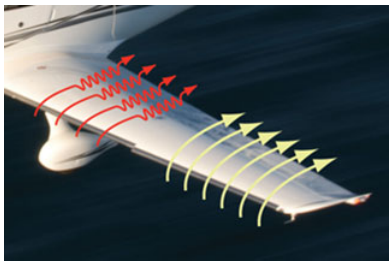
Estas dos formas de representación, física y de bocetos, son muy útiles para observar comportamientos generales de un sistema. Sin embargo, cuando el sistema es complejo (muchas partes y relaciones) es muy difícil determinar con estos modelos si éste cumple o no ciertas propiedades. En estos casos una mejor forma de representación es un *lenguaje*.



(a) (b)

Figura 1: Un sistema y su modelo. (a) sistema, (b) modelo

Cada dominio de la ingeniería ha desarrollado varios lenguajes particulares para expresar sus modelos. La industria aeronáutica, por ejemplo, usa el lenguaje del cálculo y de las ecuaciones diferenciales para representar el comportamiento de las alas ante el flujo del aire. En la figura 2, el lenguaje (ecuaciones diferenciales) expresa la diferencia de presión (p) entre la parte superior e inferior del ala con respecto a la curvatura del ala (R), la velocidad (v) y la densidad del aire (ρ). La igualdad (ecuación) que se muestra en la parte (b) de la figura 2 define una *propiedad* que debe cumplir siempre el modelo, independientemente de los valores particulares de lo que se haya escogido observar del sistema, en este caso, R, v, ρ La aeronáutica usa también el lenguaje de la físisca de materiales para representar el comportamiento de la estructura del fuselaje ante distintas condiciones.



(a)

$$\frac{\mathrm{d}\,p}{\mathrm{d}\,R} = \rho\frac{v^2}{R}$$

(b)

Figura 2: Flujo de aire en el ala. (a) modelo, (b) lenguaje

La ciencia de la computación (y algunas ramas de la ingeniería electrónica) utiliza lenguajes de representación adaptados a la descripción de un sistema como algo que transforma, no cosas materiales, sino *información*. En cada momento dado, lo que se observa del sistema es cierta información. Por ejemplo, el comportamiento del ala ante el flujo de aire podría estar descrito en cada momento por información que incluya cuál es la velocidad del aire en el borde frontal del ala, cuál es en el borde posterior, cuál es la presión en distintos puntos de la superficie del ala. A lo largo del vuelo del avión esta información va cambiando. El lenguaje del modelo para el ala permite observar de manera compacta y precisa cómo evoluciona paso a paso esa información. Para la computación, en este ejemplo simplificado, el sistema ala-viento *es* el proceso de transformación de esa información.

2. Lenguaje de modelado

Como se dijo, las ciencias de la computación proponen lenguajes para expresar modelos. Lo que es común a muchos de estos lenguajes es que representan un sistema como un *proceso de transición de estados*. Se denomina *estado* la colección de observaciones del sistema en un momento dado. Supongamos el sistema de una bicicleta (ver figura 3). Este

Figura 3: Sistema de una bicicleta

es un sistema complejo, con muchos componentes que se relacionan para hacer posible que la bicicleta avance a distintas velocidades y con distintos grados de esfuerzo por parte del pedalista. La primera tarea en la construcción de un modelo consiste en enfocarse en ciertos aspectos del sistema, que son los que se desea entender. Esta primera tarea tiene como consecuencia la eliminación de muchos detalles del sistema, que la persona que construye el modelo considera irrelevantes para lo que quiere entender de él. En el caso de la bicicleta, supongamos que nos interesa entender las relaciones entre el tamaño del engranaje del cambio en la rueda trasera, el tamaño del plato en los pedales, el diámetro de la rueda trasera y la distancia recorrida, para un número de pedalazos dado. Esta selección, que se muestra mediante las flechas en la figura 3, determina lo que interesa *observar* del sistema.

Del sistema de la bicicleta nos interesa entonces observar lo siguiente:

1. El número de dientes del plato. Esto da una idea de su circunferencia.

2. El número de dientes del piñón correspondiente al cambio en el que está la bicicleta.
3. El diámetro de la llanta trasera.
4. El número de pedalazos que se da.
5. La distancia recorrida.

El siguiente paso es identificar cada cosa que vamos a observar mediante un nombre. Por ejemplo, como se muestra en el cuadro 1:

Observación	Nombre
número de dientes del plato	*plato*
número de dientes del cambio	*cambio*
diámetro de la llanta trasera	*diam*
número de pedalazos	*ped*
distancia recorrida	*dist*

Cuadro 1: Nombre de las observaciones

El nombre que se da a cada aspecto de observación es arbitrario, pero una buena práctica que debe seguir el ingeniero consiste en dar nombres que permitan identificar claramente lo que se observa. Estos nombres se denominan *variables* del sistema.

Una observación del sistema está compuesta de los *valores* que tienen en un momento dado cada uno de las variables que se escogieron. Por ejemplo, dos observaciones, hechas en momentos diferentes, podrían ser:

`Observación 1`	`Observación 2`
$plato = 42$	$plato = 42$
$cambio = 17$	$cambio = 35$
$diam = 28$	$diam = 28$
$ped = 1$	$ped = 2$
$dist = 207$	$dist = 201$

En principio, el *comportamiento* de un sistema es simplemente la colección de todas las observaciones posibles. Sin embargo, tener un listado completo de ellas, como en el caso de la bicicleta, es a veces imposible. Pero aun si fuera posible, el listado no explica *por qué* esas observaciones son válidas y otras no. Explicar la razón de la validez de una observación requiere que hayamos encontrado una *relación* entre los valores de las variables en toda observación. Es decir, una explicación precisa que establezca cómo cambian los valores de una variable con respecto a los valores de las otras. Por ejemplo, podemos inferir que la distancia recorrida será mayor cuando el piñón del cambio sea más pequeño, pero también cuando el plato sea más grande. Es decir, debe ser proporcional a la razón entre el tamaño

del plato y el piñón del cambio. Esta razón indica simplemente el número de giros que da el piñón del cambio por cada giro del plato. Si se escoge un plato grande y un cambio pequeño, se va a recorrer más distancia en cada pedalazo (aunque el esfuerzo del pedalista será mayor). Es decir, una primera aproximación podría ser:

$$dist = K \times (plato/cambio)$$

En esta relación K es una cantidad que falta determinar. Obviamente, la distancia recorrida (en una cierta unidad de tiempo) también es proporcional al número de pedalazos: entre más pedalazos, más se avanza. Finalmente, también es proporcional al diámetro de la rueda. Cada giro del engranaje del cambio hace dar un giro a la rueda trasera. Si la rueda es más grande, ese giro avanzará más distancia. Cada giro de la rueda trasera recorre una distancia igual a la circunferencia de la rueda, que es igual a π multiplicado por su diámetro. La relación podría entonces ajustarse de la siguiente manera:

$$dist = ped \times diam \times \pi \times (plato/cambio)$$

Todas las observaciones deben obedecer esta relación. Es decir, la colección de todas las observaciones válidas del sistema está descrita por la relación. Una relación que describe todas las observaciones válidas de un sistema se denomina un *invariante* del sistema.

2.1. Especificación de sistemas en el lenguaje Event B

El lenguaje Event B está diseñado para facilitar la expresión organizada de un modelo computacional de un sistema. Se basa en la división del sistema en *componentes*. Hay fundamentalmente dos tipos de componentes, el *contexto* y la *máquina*.

Un contexto describe todo lo que es *constante* del sistema: las cosas que no cambian, lo que permanece igual a lo largo de todas las observaciones. Qué es constante o no de un sistema depende de nuestro punto de vista sobre él. En el ejemplo de la bicicleta, el diámetro de la llanta trasera puede o no ser constante. Si estamos interesados en observar la bicicleta andando, no es razonable pensar que el pedalista vaya a cambiar la llanta trasera mientras avanza! Bajo este punto de vista, el diámetro de la rueda trasera permanecerá constante. Sin embargo, si nos interesa el sistema desde el punto de vista del entrenador en una competencia completa de bicicletas, será posible observar que antes de alguna carrera se selecciona un tipo de llanta trasera, con un cierto diámetro. En este caso, el diámetro no sería una constante para ese sistema.

2.1.1. El contexto

En Event-B, la forma de un contexto es la siguiente:

```
context biciCtx
constants
...
axioms
...
end
```

El nombre que se da al contexto, en este ejemplo "biciCtx", es arbitrario. La sección `constants` le da un nombre a cada constante del sistema. La sección `axioms` es para escribir las *propiedades* de las constantes. Una propiedad importante, que es necesario especificar siempre, es el *tipo* de la constante. El tipo indica qué clase de valores toma la constante. Por ejemplo, pueden ser valores numéricos enteros, o pueden ser colores, o la constante puede también representar objetos (un carro, por ejemplo), etc.

El concepto de *tipo* es una de las nociones fundamentales de la computación. Permite clasificar lo que se observa en grupos cuyas propiedades se consideran similares. El tipo *número*, por ejemplo, comprende elementos que pueden sumarse, restarse, multiplicarse, etc. También admiten preguntarse si uno de ellos es más grande que otro. El tipo *color*, en cambio, no admite estas operaciones. Uno puede preguntarse si dos colores son iguales, o si uno de ellos es primario o si es el resultado de mezclar otros. También si es brillante u opaco. Pero no se pueden sumar dos colores, así como tampoco se puede uno preguntar si un número es más brillante que otro.

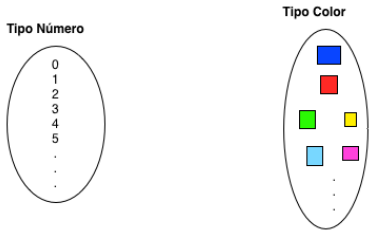


Figura 4: Tipos de valores

Para el modelo de la bicicleta, un contexto podría ser el siguiente:

```
context biciCtx
constants
  diam
axioms
 ax1 :     diam ∈ ℕ
end
```

Note que cada axioma está identificado con un nombre (arbitrario), en este caso "ax1". En el axioma $ax1$ del ejemplo, se quiere especificar que la constante *diam* (el diámetro) tiene como valor algún número. El valor preciso de este número no es importante en esta etapa del modelo. El modelo quiere representar a toda bicicleta posible. Cada una de estas puede tener su propio valor para el diámetro. La manera de decir que *diam* es un número es especificando en cuál conjunto se encuentran todos sus valores posibles. Este es el conjunto de los números *naturales*. Los números naturales en el lenguaje Event-B corresponden al conjunto $\{0, 1, 2, 3, 4, ...\}$, que se denota como $\mathbb{N}$. Lo que expresa $diam \in \mathbb{N}$ se lee "el valor de la constante *diam* es un número natural cualquiera". Como el conjunto de los números naturales incluye el cero, el diámetro podría ser igual a cero. Una llanta con diámetro cero sería bastante extraña, por lo que agregamos una propiedad (otro axioma) que elimine el cero como valor posible para el diámetro. El axioma $ax2$ especifica esta condición afirmando que el valor del diámetro es *estrictamente* mayor a cero:

```
context biciCtx
constants
  diam
axioms
 ax1 :  diam ∈ ℕ
 ax2 :  diam > 0
end
```

En la definición del contexto el ingeniero expresa algo muy importante sobre el modelo que va a construir. En nuestro ejemplo, dice que este modelo representa el sistema de una bicicleta cualquiera *siempre y cuando el diámetro de su rueda trasera no sea igual a cero*. Es en este sentido en el que debe entenderse la noción de *contexto*: es la expresión precisa de las restricciones sobre la clase de sistemas que el modelo es capaz de representar.

2.1.2. La máquina

La parte *dinámica* de un modelo, que expresa lo que cambia con la evolución del sistema, se especifica en un componente que se denomina *máquina*. Lo que cambia cuando un sistema evoluciona son las observaciones. Por ejemplo, en un momento dado puede observarse un incremento en los pedaleos y también (por consiguiente) observar un aumento en la distancia recorrida. Tanto la observación de pedalazos, como la de distancia, pertenecen entonces a la parte dinámica del modelo.

La forma de la parte dinámica del modelo es la siguiente:

```
machine biciMaq sees biciCtx
variables
...
invariants
...
events
...
end
```

La primera línea da un nombre al componente, en este caso "biciMaq" (puede ser cualquier otro), y dice que este componente puede "ver" (*sees*, en inglés) el contexto de nombre "biciCtx". Ver un contexto significa que el componente dinámico del modelo asume las propiedades definidas en ese contexto. Por lo tanto, puede referirse a las constantes que se definieron en él.

En la segunda línea se lista cada una de las variables que representa observaciones. La sección de *invariants* es el análogo de *axioms*: describe las propiedades de las observaciones (por ejemplo, el tipo de sus valores). En nuestro ejemplo, una de las observaciones es el número de pedalazos. Esta observación se llamó *ped*, que entonces hace parte de las variables del modelo. El valor de cada observación de la variable *ped* es un número. Entonces, en la sección de *invariants* se especifica que los valores de la variable *ped* pertenecen al conjunto de los números (o sea, $ped \in \mathbb{N}$).

Finalmente está la sección que describe los *eventos* del sistema. En un sistema se pueden observar, en general, muchísimas acciones. Por ejemplo, que el pedalista quitó las manos del manubrio, o que inclinó un poco la bicicleta. Lo que se llama *evento* es una de esas acciones, pero solamente de aquellas que afectan lo que hemos escogido observar. Soltar las manos del manubrio es una acción, pero no es un evento porque no afecta el número de pedalazos ni la distancia recorrida.

Un evento es entonces cualquier acción que cause modificaciones en las observaciones. Por ejemplo, dar un pedalazo más va a cambiar la observación sobre el número de pedalazos y también sobre la distancia recorrida. La forma de un evento es la siguiente:

```
event pedalear
when ⟨condición⟩
then ⟨acción⟩
end
```

Cada evento consta de una condición (llamada "guarda del evento") y de una acción. La condición indica cuándo es posible realizar la acción. Por ejemplo, levantar las manos del manubrio requiere de una velocidad mínima si se quiere evitar que el pedalista se caiga, o reducir los pedalazos requiere que la bicicleta no esté ya detenida. Cuando su condición

se cumple, la acción puede realizarse, pero no es absolutamente necesario que se haga. Por ejemplo, si la bicicleta va a 30 kilómetros por hora, el pedalista podría levantar las manos del manubrio, pero también es posible que elija no hacerlo. La condición *posibilita* la acción, pero no la *obliga*.

Un evento se dice que se puede "disparar" cuando se cumple su condición. Disparar un evento consiste en realizar su acción. Para el ejemplo de la bicicleta, el evento "desacelerar", que disminuye el número de pedalazos (por unidad de tiempo), podría escribirse de la siguiente manera:

```
event desacelerar
when
 grd1 :    ped > 0
then
 acc1 :    ped := ped − 1
end
```

La condición del evento es $ped > 0$, que verifica que el número de pedalazos sea mayor a cero. Un evento puede tener más de una condición y más de una acción. Note que hay que darle un nombre a cada condición (en el ejemplo, $grd1$) y a cada acción (en el ejemplo, $acc1$). Este evento dice que cuando se esté dando al menos un pedalazo (condición $ped > 0$), se puede disminuir en uno el número de pedalazos (acción $ped := ped - 1$). El símbolo ":=" en la acción $ped := ped - 1$ quiere decir "la variable que está a la izquierda del símbolo := toma el valor de la expresión que está a la derecha de ese símbolo". Es decir, la variable ped toma el valor de $ped - 1$, es decir, uno menos que el valor que esa variable tenía antes. El símbolo ":=" solamente se puede utilizar en las acciones de los eventos, nunca en las condiciones ni en las propiedades.

La especificación completa del modelo (simplificado) de la bicicleta en el lenguaje de

event B es como sigue:

```
machine biciMaq sees biciCtx
variables
   ped
invariants
 inv1 :     ped ∈ ℕ
events
event initialisation
then
 acc1 :     ped := 0
end

event pedalear
when
 grd1 :     ped < 1000
then
 acc1 :     ped := ped + 1
end

event desacelerar
when
 grd1 :     ped > 0
then
 acc1 :     ped := ped − 1
end

end
```

```
context biciCtx
constants
  diam
axioms
 ax1 :     diam ∈ ℕ
x2 :     diam > 0
end
```

En el ejemplo anterior el evento *pedalear* tiene una condición que limita el número de pedalazos a 1000. Note que uno de los eventos de la máquina es siempre la *inicialización*. El evento de inicialización corresponde a la escogencia del momento en que se quiere empezar a observar el sistema. La acción de este evento da a cada variable un valor compatible con las propiedades establecidas en el invariante para esa variable. En el ejemplo, el invariante dice que el valor de la variable *ped* debe ser un número natural. La inicialización puede entonces asignar el valor cero, que es un número natural, pero no podría, por ejemplo, asignar el valor -1, que no lo es. El evento de inicialización solamente puede tener acciones. Nunca puede tener condiciones.

2.2. Expresiones

En la discusión del ejemplo anterior usamos de manera informal dos conceptos fundamentales de la computación, "expresión" y "condición". El término "expresión" se refiere a algo con lo que se asocia un valor. Por ejemplo, cuando se dice "la estatura de Juan más 5 centímetros", nos referimos a un valor concreto. Por ejemplo, si Juan mide un metro con 80, la frase anterior se refiere al valor 1,85. Las constantes (por ejemplo, el número 7) son expresiones. También lo son las variables. El resultado que se asocia con la expresión que es una sola variable es el valor mismo de la variable.

Cuando las variables tienen valores numéricos, como en el ejemplo de la bicicleta, las expresiones tienen también valores numéricos. Las expresiones en este caso se construyen usando números, variables y las operaciones aritméticas "+" (suma), "∗" (multiplicación) y "/" (división entera), junto con otras que veremos más adelante. Por ejemplo,

$$(ped + 1) * cambio/7$$

es una expresión cuyo valor depende de los valores de las variables *ped* y *cambio*. Si $ped = 2$ y $cambio = 14$, entonces la expresión tiene valor $(2 + 1) * 14/7$ que es igual a 6.

Como se dijo, las expresiones se usan *solamente* en las acciones de los eventos. Cada acción est compuesta de una *asignación* de la forma $var := exp$, en donde *var* es una variable cualquiera y *exp* es una expresión. Por ejemplo, una acción de un evento podría ser: $x := (x - 2) * 3 + y$. Esto se lee "la variable x toma un nuevo valor que es igual al que tenía antes menos dos, multiplicado por 3, más el valor de la variable y". La parte izquierda del símbolo := representa entonces la variable a la que se va modificar su valor, y la parte derecha la expresión que calcula el valor que se le va a asignar. Cada acción involucra un "antes" y un "después". La expresión a la derecha del símbolo ":=" se refiere a los valores que las variables tenían "antes". La variable a la izquierda de ":=" toma el valor que resulta "después" de calcular esa expresión. Por ejemplo,

`Valores antes`	`Asignación`	`Valores después`
$y = 2, x = 5$	$x := x + 3 - 2 * y$	$y = 2, x = 4$
$y = 3, x = 7$	$y := x$	$y = 7, x = 7$

La parte izquierda de la asignación := debe siempre ser una variable. Las siguientes acciones son entonces *erróneas*:

$y + 3 := z + 1$ ⟶ `error! y + 3 no es una variable`
$7 := x - 1$ ⟶ `error! 7 no es una variable`

2.3. Predicados

Las condiciones en los eventos y las propiedades (por ejemplo, la especificación del tipo de las variables) *no* son expresiones, son *predicados*. Contrariamente a las expresiones, con

un predicado no se asocia ningún valor. Un predicado es una afirmación, que puede ser verdadera o falsa. Por ejemplo, si el valor de la variable x es 3, entonces la afirmación $x > 1$ es un predicado verdadero, mientras que la afirmación $x > 7$ es falsa. Note que una expresión no puede ser un predicado. De la expresión $x + 5$ no se puede preguntar si es verdadera o falsa, solamente se puede calcular su valor, que es igual a 8. Cuando las variables son numéricas, los predicados se construyen con las *comparaciones*, "=" (es igual a), ">" (es mayor a), "<" (es menor a), ">=" (es mayor o igual a), "<=" (es menor o igual a). Por ejemplo,

$$(x - 1) * 2 + 1 > y + 3$$

es un predicado. Si $x = 3, y = 1$, el predicado es verdadero, mientras que si $x = 2, y = 1$, el predicado es falso. El predicado $x >= 4$ es verdadero cuando el valor de x es cualquier número igual o mayor a 4. Por ejemplo, cuando $x = 7$ el predicado $x \geq 4$ es verdadero. También lo es el predicado $x >= 7$. Note que en los predicados se puede usar el símbolo "=", que es diferente a ":=". El predicado "$x = y + 1$" afirma que el valor actual de la variable x es igual al valor actual de la variable y más uno. Por ejemplo, si $x = 3, y = 2$, esa afirmación es verdadera. La asignación $x := y + 1$, por el contrario, no afirma nada, sino que representa una acción: hace que el valor de la variable "x" *cambie*, tomando como valor el resultado de sumarle uno al valor de la variable "y". Si $x = 7, y = 2$, el predicado $x = y + 1$ es falso, mientras que la asignación $x := y + 1$ simplemente modifica el valor que tenía "x" para volverlo igual a 3.

En los predicados que aparecen en la especificación de las propiedades del sistema, se usa también el símbolo ":", que se puede leer "es de tipo". Por ejemplo, $x : NAT$ quiere decir que los valores que toma la variable x son de tipo "número natural", es decir, cualquier número mayor o igual a cero. En matemáticas es usual escribir lo anterior de la forma, $x \in \mathbb{N}$. Para valores numéricos, existen las siguientes definiciones de tipo:

Declaración	En matemáticas	Significado
$x : NAT$	$x \in \mathbb{N}$	x es un número natural (mayor o igual a cero)
$x : NAT1$	$x \in \mathbb{N}_1$	x es un número natural, pero no cero
$x : INT$	$x \in \mathbb{Z}$	x es un número entero (puede ser negativo)

2.4. Expresiones, predicados, asignaciones en un modelo

En el modelo simplificado del ejemplo de la bicicleta que se mostró antes, no se usa el diámetro de la rueda ni se actualiza la distancia recorrida en la unidad de tiempo en que se dan los pedalazos. Tampoco se establece como propiedad que no se puedan dar más de 1000 pedalazos por unidad de tiempo. Para tener en cuenta estos detalles, hay que agregar una variable "*dist*" que tenga como valor la distancia recorrida y definir propiedades para el máximo número de pedalazos y para la relación entre la distancia recorrida, los cambios y los pedalazos. Para simplificar, en este modelo hemos utilizado el valor 3 para π. Hemos determinado también que el plato es una constante. Es decir, el pedalista no puede

cambiarlo.

machine $biciMaq$ **sees** $biciCtx$
variables
ped
$dist$
$cambio$
invariants
$inv1:$ $ped \in \mathbb{N}$
$inv2:$ $dist \in \mathbb{N}$
$inv3:$ $cambio \in \mathbb{N}$
$inv4:$ $ped \leq 1000$
$inv5:$ $dist = ped \times diam \times 3 \times (plato/cambio)$
events
event initialisation
then
$acc1:$ $ped := 0$
$acc2:$ $dist := 0$
$acc3:$ $cambio := 7$
end

event $pedalear$
when
$grd1:$ $ped < 1000$
then
$acc1:$ $ped := ped + 1$
$acc2:$ $dist := (ped + 1) \times diam \times 3 \times (plato/cambio)$
end

event $desacelerar$
when
$grd1:$ $ped > 0$
then
$acc1:$ $ped := ped - 1$
$acc2:$ $dist := (ped - 1) \times diam \times 3 \times (plato/cambio)$
end

end

context $biciCtx$
constants
$diam$
$plato$
axioms
$ax1:$ $diam \in \mathbb{N}_1$
$ax2:$ $plato \in \mathbb{N}_1$
end

Los tres primeros invariantes de la máquina expresan que todas las variables, ped, $dist$, $cambio$, son de tipo "número natural". El invariante "inv4" establece una propiedad del sistema, que dice que todas las observaciones de pedalazos en cada momento son números menores

o iguales a 1000. El invariante "inv5" describe la relación que existe entre los pedalazos y la distancia recorrida en una cierta unidad de tiempo. Como se vió antes, la distancia es proporcional al número de pedalazos, al diámetro y a la razón entre el plato y el cambio. Estos dos invariantes determinan cuáles son observaciones posibles del sistema y cuáles no. Toda observación posible debe obedecer esas dos propiedades.

Una observación particular del sistema es la que se realiza con la inicialización. En ella se define la observación inicial. Esta observación, como todas las demás, debe obedecer las propiedades establecidas en el invariante. Es decir, cada predicado del invariante debe resultar verdadero para los valores de las variables en la inicialización. Verifiquemos esto. Si $ped = 0$, los invariantes "inv1" e "inv4" son verdaderos. Si $dist = 0$, el invariante "inv2" es verdadero. Además, como $ped = 0$, entonces el valor de la expresión $ped \times diam \times 3 \times (plato/cambio)$ es igual a cero y, como $dist = 0$, entonces la propiedad "inv5" se reduce a $0 = 0$, que es verdadero.

La acción $acc2$ del evento *pedalear* se puede reducir de la siguiente manera:

$$\begin{aligned}&(ped + 1) \times diam \times 3 \times (plato/cambio)\\ &= \ ped \times diam \times 3 \times (plato/cambio) + diam \times 3 \times (plato/cambio)\end{aligned}$$

Usando el invariante $inv5$ de la máquina, podemos reemplazar en esta última expresión la subexpresión $ped \times diam \times 3 \times (plato/cambio)$, por su igual, que es la variable $dist$. Entonces,

$$\begin{aligned}&ped \times diam \times 3 \times (plato/cambio) + diam \times 3 \times (plato/cambio)\\ &= \ dist + diam \times 3 \times (plato/cambio)\end{aligned}$$

El evento *pedalear* queda entonces,

`event` *pedalear*
`when`
$grd1:\quad ped < 1000$
`then`
$acc1:\quad ped := ped + 1$
$acc2:\quad dist := dist + diam \times 3 \times (plato/cambio)$
`end`

3. Recapitulación: aspectos esenciales de un modelo

De un modelo se deben identificar sus componentes esenciales:

1. Las constantes: son caractersticas del sistema que *no varían* con la evolución del sistema.
   Hay que anotar que lo que es o no constante depende de los límites del modelo. Cada modelo escoge una frontera entre lo que se considera parte del sistema y lo que se

considera parte del entorno. Por ejemplo, un sistema de transporte en buses puede considerar solamente lo que sucede con la flota de buses que haya en un momento dado. No se ocupa entonces de compras o retiros de buses. En un modelo de este estilo, el número total de buses sería una constante.

Ejercicio 1 *Realice una transformación similar para el evento desacelerar.*

Ejercicio 2 *Escriba la máquina completa de la bicicleta en Rodin. Incluya dos eventos nuevos llamados subirCambio y bajarCambio. El primero chequea si el cambio está en el valor* 7 *y en ese caso lo sube a* 14*. El segundo hace lo contrario.*