novática
Revista de la Asociación de Técnicos de Informática

Novática, revista fundada en 1975 y decana de la prensa informática española, es el órgano oficial de expresión y formación continua de **ATI** (Asociación de Técnicos de Informática), organización que edita también la revista **REICIS** (Revista Española de Innovación, Calidad e Ingeniería del Software).

<http://www.ati.es/novatica/>
<http://www.ati.es/reicis/>

ATI es miembro fundador de **CEPIS** *(Council of European Professional Informatics Societies)* y es representante de España en **IFIP** *(International Federation for Information Processing)*; tiene un acuerdo de colaboración con **ACM** *(Association for Computing Machinery)*, así como acuerdos de vinculación o colaboración con **AdaSpain**, **AI2**, **ASTIC**, **RITSI** e **Hispalinux**, junto a la que participa en **ProInnova**.

Coordinación Editorial, Redacción Central y Redacción ATI Madrid
Plaza de España 6, 2ª planta, 28008 Madrid
Tlfn.914029391; fax.913093685 <novatica@ati.es>
Composición, Edición y Redacción ATI Valencia
Av. del Reino de Valencia 23, 46005 Valencia
Tlfn. 963740173 <novatica_prod@ati.es>
Administración y Redacción ATI Cataluña
Calle Avila 50, 3a planta, local 9, 08005 Barcelona
Tlfn.934125235; fax 934127713 <secregen@ati.es>
Redacción ATI Aragón
Lagasca 9, 3-B, 50006 Zaragoza.
Tlfn./fax 976235181 <secreara@ati.es>
Redacción ATI Andalucía <secreand@ati.es>
Redacción ATI Galicia <secregal@ati.es>
Suscripción y Ventas <novatica.subscripciones@atinet.es>
Publicidad Plaza de España 6, 2ª planta, 28008 Madrid
Tlnf.914029391; fax.913093685 <novatica@ati.es>
Imprenta: Derra S.A., Juan de Austria 66, 08005 Barcelona.
Depósito legal: B 15.154-1975 -- ISSN: 0211-2124; CODEN NOVAEC
Portada: "Heuristically" - Concha Arias Pérez / © ATI
Diseño: Fernando Agresta / © ATI 2003

sumario

Nº 226, noviembre-diciembre 2013, año XXXIX

Tema del próximo número:
"Eficiencia energética en centros de procesos de datos"

David Aguilera Moncusí,
Cristina Gómez Seoane,
Antoni Olivé Ramon
Universitat Politècnica de Catalunya, BarcelonaTech

<daguilera@essi.upc.edu>,
<antoni.olive@upc.edu>,
<cristina@essi.upc.edu>

Guía de estilo completa para nombrar los elementos de un esquema conceptual en UML/OCL

Resumen: *Uno de las objetivos principales a la hora de especificar los requisitos funcionales de un sistema de información es el facilitar su comprensión por parte de los usuarios finales. Así, cuando se usan lenguajes de modelización conceptual como es el caso de UML, resulta muy importante usar sistemáticamente una nomenclatura apropiada, rigurosa y coherente para los distintos elementos de los esquemas. El propósito de este artículo es proporcionar una guía de estilo completa para denominar los distintos elementos que pueden aparecer en los esquemas conceptuales en UML.*

Palabras clave: *Calidad pragmática, calidad semántica, calidad sintáctica, comprensibilidad, esquema conceptual, forma gramatical, frase patrón, guía de estilo, modelización conceptual, nomenclatura, UML.*

1. Introducción

La especificación de los requisitos funcionales constituye una de las tareas principales en el desarrollo de sistemas de información, independientemente del método de desarrollo (en cascada, ágil, etc.) usado. Esta especificación consta básicamente del esquema conceptual del sistema, que describe la estructura de los conceptos procesados por el sistema, y el comportamiento del sistema en respuesta a los eventos que se producen.

En la actualidad, el lenguaje de modelización conceptual más utilizado es el UML (*Unified Modeling Language*). Se trata de un lenguaje propuesto por OMG (*Object Management Group*) a finales de los años 90, de aplicación general en el desarrollo de software y que se ha convertido en un estándar de facto.

Los modelos definidos en UML deben tener la calidad mínima que se requiera en cada caso, y seguir las mejores prácticas profesionales. Naturalmente, esto también se aplica a los modelos conceptuales.

Se acostumbra a distinguir tres tipos de calidad en los esquemas conceptuales: sintáctica, semántica y pragmática. El objetivo de la calidad pragmática es que todas las partes interesadas comprendan las partes del esquema que les son relevantes. En este sentido, los nombres de los elementos de los esquemas son muy importantes, porque tienen una gran influencia en su comprensión. Los esquemas son más fáciles de comprender si se han dado nombres adecuados a sus elementos.

La guía que proponemos a continuación trata cada uno de los elementos de los esquemas conceptuales que pueden ser nombrados en UML[1]. Para cada elemento, la guía proporciona una forma gramatical y una frase patrón. Un nombre de un elemento cumple con la guía si tiene la forma gramatical indicada, y si la frase generada a partir de la frase patrón y el nombre es sintácticamente correcta y semánticamente significativa.

El uso sistemático de la guía que se propone debería contribuir a mejorar la calidad de los esquemas conceptuales en UML.

2. Tipos de entidad

En esta sección ofrecemos una propuesta para nombrar los tipos de entidad en UML (ver **tabla 1**). La regla que presentamos sirve para clases UML, clases asociativas y tipos de datos (los cuales incluyen las enumeraciones).

Está claro que nombres como *Persona*, *Silla*, *Factura* o *Categoría* (clases), *Trabajo* o *Afiliación* (clases asociativas), *Fecha* o *CantidadDeDinero* (tipos de datos) y *Sexo* o *Color* (enumeraciones), siguen la norma que hemos presentado.

Un ejemplo más complejo de nombre que también cumple la guía es *VíaParaVehículosAMotor*. El núcleo del sintagma nominal (*Vía*) es un nombre contable y singular, y la frase que se obtiene en aplicarlo a $\boldsymbol{G_{1s}}$ es:

Una instancia de este tipo de entidad es una vía para vehículos a motor.

la cual es gramaticalmente correcta y tiene sentido.

Fíjense que la norma $\boldsymbol{G_{1f}}$ establece que "el núcleo del sintagma nominal tiene que ser un nombre contable y singular". Esto quiere decir que nombres como *Agua* o *Plata* son inválidos, ya que no son contables. De hecho, lo que suele pasar con instancias de estos conceptos es que, en realidad, son "porciones" o "cantidades". Para estos casos atípicos, podríamos ofrecer una frase distinta que incluyese esta noción de "porción" o "cantidad":

Una instancia de este tipo de entidad es una porción de *minúsculas**(E)***.

pero creemos que es mejor que sea el modelador quien le asigne un prefijo que convierta el nombre en contable. Así, por ejemplo,

Tipos de entidad

Sea ***E*** un tipo de entidad denominado, precisamente, ***E***, entonces

$\boldsymbol{G_{1f}}$: El nombre de un tipo de entidad ***E*** tiene que ser un sintagma nominal cuyo núcleo sea contable y singular. El nombre debe escribirse con la "forma Pascal"[1]; es decir, la primera letra de cada palabra debe estar en mayúscula y el nombre no puede contener espacios.

$\boldsymbol{G_{1s}}$: La siguiente frase tiene que ser gramaticalmente correcta y tener sentido:

Una instancia del tipo de entidad denominado ***E*** es [un|una] *minúsculas**(E)***.

donde *minúsculas*(***E***) es una función que devuelve el nombre ***E*** en minúsculas, separando las palabras con un espacio.

Cuando el tipo de entidad es implícito por el contexto donde estamos, la frase es, sencillamente:

Una instancia de este tipo de entidad es [un|una] *minúsculas**(E)***.

Tabla 1. Nomenclatura para tipos de entidad UML.

tendríamos *BotellaDeAgua* o *PorciónDePlata*. Esto es lo que hemos hecho antes con el concepto *Dinero*, el cual hemos nombrado como *CantidadDeDinero*.

Si el nombre del tipo de entidad contiene acrónimos, éstos tienen unas reglas específicas (ver **tabla 2**).

Algunos ejemplos del uso de acrónimos son (1) *bdActiva* y *cssPorDefecto*; (2) *LectorDeCD* y *UsuarioDeLaBD*; y (3) *PáginaHtml* y *DocumentoXml*.

3. Atributos

En esta sección presentamos las reglas para nombrar atributos. Éstas sirven para atributos de una clase, de una clase asociativa, o de un tipo de datos.

Sea ***E***::***A***:***T*** [*min*..*max*] un atributo denominado ***A*** de tipo ***T*** de un tipo de entidad ***E*** y cuyas multiplicidades mínima y máxima son *min* y *max*. Para simplificar esta guía de estilo, supondremos que el mínimo es 0 ó 1 y el máximo es 1 ó * (sin límite), ya que el caso general es fácilmente derivable.

La guía G_2 que proponemos para los atributos distingue aquellos que son *booleanos* de los que no lo son. En ambos casos, la frase que generaremos tiene en cuenta las multiplicidades. Sea G_2 la regla para atributos no booleanos y G_2' la regla para los booleanos.

3.1. Regla para atributos no booleanos

Si *T* no es el tipo de datos *booleano*, entonces la regla que aplicaremos para el atributo es la expresada en la **tabla 3**.

El ejemplo de la **figura 1** muestra cuatro atributos no *booleanos*, los cuales siguen la anterior regla. Las frases que generan son:

Una oferta tiene un precio.
Una oferta puede tener una fecha de caducidad.
Una oferta puede tener cero o más imágenes.
Una cantidad de dinero tiene un valor.
Un producto tiene uno o más colores.
Una tienda tiene un nombre.

3.2. Regla para atributos booleanos

Si ***T*** es *booleano*, entonces suponemos que la multiplicidad mínima es 0 ó 1 y la máxima 1 (ver **tabla 4**).

El ejemplo de la **figura 1** muestra tres atributos *booleanos*, los cuales siguen la anterior norma. Las frases que generan son:

Una tienda abre o no abre todo el día, o no se sabe.
Una tienda es o no es económica.
Un producto caduca o no caduca.

Uso de mayúsculas y minúsculas en acrónimos

1. Si el acrónimo es la primera palabra de un nombre y, además, el nombre tiene que seguir la "forma Camel" (es decir, la primera palabra se escribe en minúsculas, las demás empiezan con mayúscula y no puede contener espacios), entonces escriban el acrónimo todo en minúsculas.
2. Respetando la primera regla, escriban siempre en mayúsculas todas las letras de un acrónimo de sólo dos letras.
3. Respetando la primera regla, escriban sólo la primera letra en mayúscula para cualquier acrónimo de tres o más letras.

Tabla 2. Reglas específicas si el tipo de entidad contiene acrónimos.

Atributos no *booleanos*

Sea ***E***::***A***:***T*** [*min*..*max*] un atributo denominado ***A*** de tipo ***T*** de un tipo de entidad ***E***, entonces

G_{2f}: El nombre ***A*** tiene que ser un sintagma nominal cuyo núcleo sea contable y singular, escrito utilizando la "forma Camel".

G_{2s}: Una de les siguientes frases tiene que ser gramaticalmente correcta y tener sentido:

- Si $min = 0$ y $max = 1$:

[Un|Una] *minúsculas*(***E***) puede tener [un|una] *minúsculas*(***A***).

- Si $min = 0$ y $max > 1$:

[Un|Una] *minúsculas*(***E***) puede tener cero o más *minúsculas*(plural(***A***)).

- Si $min = max = 1$:

[Un|Una] *minúsculas*(***E***) tiene [un|una] *minúsculas*(***A***).

- Si $min > 0$ y $max > 1$:

[Un|Una] *minúsculas*(***E***) tiene [un|una] o más *minúsculas*(plural(***A***)).

donde *plural*(***A***) es una función que devuelve la forma plural de ***A***.

Tabla 3. Regla para atributos no booleanos.

Atributos *booleanos*

Sea ***E***::***A***:*Boolean* [*min*..*max*] un atributo booleano denominado ***A*** de un tipo de entidad ***E***, entonces

G_{2f}: El nombre ***A*** tiene que ser un sintagma verbal en tercera persona del singular, escrito con la "forma Camel".

G_{2s}: La siguiente frase tiene que ser gramaticalmente correcta y tener sentido:

[Un|Una] *minúsculas*(***E***) *minúsculas(normalONegado*(***A***)) [, o no se sabe].

donde la última parte es opcional y se incluye si la multiplicidad mínima es cero.

La función *normalONegado*(***A***) expande ***A*** añadiéndole su forma negada. Por ejemplo:

normalONegado(esDerivado) = esONoesDerivado
normalONegado(tieneHijos) = tieneONoTieneHijos

Tabla 4. Regla para atributos booleanos.

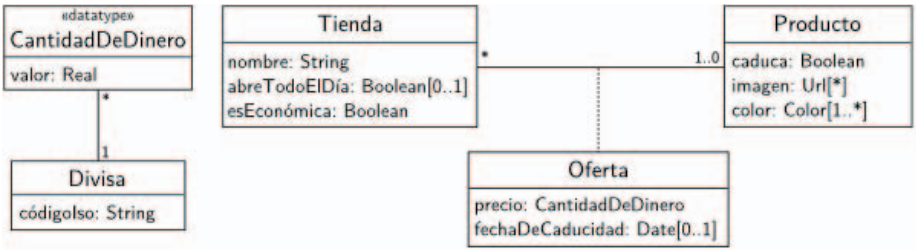

Figura 1. Ejemplos de atributos booleanos y no booleanos.

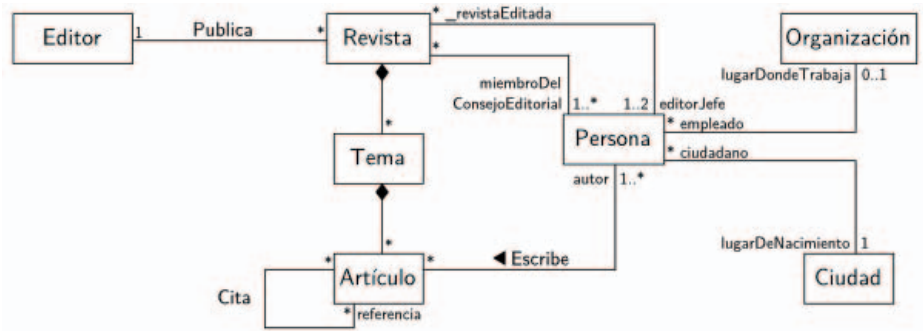

Figura 2. Ejemplos de asociaciones y roles.

Asociaciones binarias

Sea $\boldsymbol{R}(\boldsymbol{p}_1{:}\boldsymbol{E}_1\ [min_1, max_1], \boldsymbol{p}_2{:}\boldsymbol{E}_2\ [min_2, max_2])$ una asociación binaria, entonces

$\boldsymbol{G}_{3f}{}^{a}$: El nombre de la asociación $\boldsymbol{R}$ tiene que ser un sintagma verbal en tercera persona del singular, escrito utilizando la "forma Pascal".

$\boldsymbol{G}_{3s}{}^{a}$: Una de les siguientes frases tiene que ser gramaticalmente correcta y tener sentido:

- Si $min_2 = 0$ y $max_2 = 1$:

  [Un|Una] *minúsculas*($\boldsymbol{E}_1$) *minúsculas*($\boldsymbol{R}$), como mucho, [un|una] *minúsculas*($\boldsymbol{E}_2$).

- Si $min_2 = 1$ y $max_2 = 1$:

  [Un|Una] *minúsculas*($\boldsymbol{E}_1$) *minúsculas*($\boldsymbol{R}$) [un|una] *minúsculas*($\boldsymbol{E}_2$).

- Si $min_2 = 0$ y $max_2 = *$:

  [Un|Una] *minúsculas*($\boldsymbol{E}_1$) *minúsculas*($\boldsymbol{R}$) cero o más *minúsculas*(plural($\boldsymbol{E}_2$)).

- Si $min_2 = 1$ y $max_2 = *$:

  [Un|Una] *minúsculas*($\boldsymbol{E}_1$) *minúsculas*($\boldsymbol{R}$) [un|una] o más *minúsculas*(plural($\boldsymbol{E}_2$)).

Tabla 5. Regla para las asociaciones binarias en UML.

4. Asociaciones

En esta sección presentamos cómo nombrar asociaciones en UML. Distinguimos entre asociaciones binarias y $\boldsymbol{n}$-arias porque, como veremos, las respectivas reglas son bastante diferentes.

4.1. Asociaciones binarias

Sea $\boldsymbol{R}(\boldsymbol{p}_1{:}\boldsymbol{E}_1\ [min_1, max_1], \boldsymbol{p}_2{:}\boldsymbol{E}_2\ [min_2, max_2])$ una asociación binaria entre dos tipos de entidad $\boldsymbol{E}_1$ y $\boldsymbol{E}_2$, cuyos roles son $\boldsymbol{p}_1$ y $\boldsymbol{p}_2$ con multiplicidades $[min_1, max_1]$ y $[min_2, max_2]$, respectivamente. En UML, el orden de los participantes en la asociación se muestra con una pequeña flecha sólida al lado del nombre R. Cuando el orden es de arriba a abajo o de izquierda a derecha, puede omitirse.

UML permite tres nombres explícitos en una asociación, todos ellos opcionales: el nombre de la asociación $\boldsymbol{R}$ y los dos nombres de rol $\boldsymbol{p}_1$ y $\boldsymbol{p}_2$. Los roles siempre tienen un nombre implícito cuando no se les da un nombre explícito, el cual coincide con el nombre del tipo de entidad al que están vinculados, pero empezando en minúscula.

En general, no hay nombres implícitos para las asociaciones. Sin embargo, UML incluye dos tipos especiales de asociación, las agregaciones y las composiciones, los cuales sí tienen nombres predefinidos y, por lo tanto, no debemos darles ninguno.

La regla $\boldsymbol{G}_3$ que proponemos para las asociaciones binarias en UML tiene dos partes: una $\boldsymbol{G}_3{}^{a}$ para el nombre de la asociación $\boldsymbol{R}$ (ver **tabla 5**) y otra para los nombres de rol $\boldsymbol{G}_3{}^{r}$ (ver **tabla 6**).

Como ya hemos dicho, los nombres de rol pueden ser implícitos o explícitos. Distinguimos entre dos tipos de nombre de rol explícitos: los *externos* y los *internos*. En UML, hay casos en que tenemos que asignar un nombre rol a un miembro de una asociación y, por tanto, no queremos que se utilicen en la verbalización del esquema. Para esos casos diferenciamos entre ambos tipos de nombre y distinguimos los internos añadiendo un guión bajo como prefijo.

La regla que proponemos para los nombres de rol externos se basa en la idea que "un rol se puede ver como un atributo no booleano del tipo de entidad del otro extremo" y, por consiguiente, la regla que deberíamos aplicar es la misma que la que hemos descrito para esos atributos. En particular, si $\boldsymbol{R}(\boldsymbol{p}_1{:}\boldsymbol{E}_1\ [min_1, max_1], \boldsymbol{p}_2{:}\boldsymbol{E}_2\ [min_2, max_2])$ es una asociación binaria, entonces los roles $\boldsymbol{p}_1$ y $\boldsymbol{p}_2$ se pueden ver como los siguientes atributos:

$$\boldsymbol{E}_2{::}\boldsymbol{p}_1{:}\boldsymbol{E}_1[min_1, max_1] \text{ y } \boldsymbol{E}_1{::}\boldsymbol{p}_2{:}\boldsymbol{E}_2[min_2, max_2]$$

y, así, tienen que seguir la regla $\boldsymbol{G}_2$. La regla en cuestión, adaptada a los nombres de rol, se muestra en la **tabla 6**.

Como ejemplos, consideremos las nueve asociaciones que se muestran en la **figura 2**. Hay dos asociaciones que son composiciones (*Revista–Tema* y *Tema–Artículo*) y que, por consiguiente, no necesitan nombre. Hay cuatro asociaciones con nombres explícitos que siguen la regla $\boldsymbol{G}_{3f}{}^{a}$, generando las siguientes frases:

Un editor puede publicar cero o más revistas.
Un artículo puede citar cero o más artículos.
Una persona puede escribir cero o más artículos.
Una persona nació en una ciudad.

Hay ocho nombres de rol explícitos, los cuales siguen la regla $\boldsymbol{G}_{3f}{}^{r}$:

Un artículo tiene cero o más referencias.
Un artículo tiene uno o más autores.
Una ciudad tiene cero o más ciudadanos.
Una persona tiene un lugar de nacimiento.
Una revista tiene uno o más editores jefe.
Una revista tiene uno o más miembros del consejo editorial.
Una organización tiene cero o más empleados.
Una persona tiene, como mucho, un lugar donde trabaja.

Fíjense que el nombre de rol *_revistaEditada* es interno. En este ejemplo, necesitamos asignarle un nombre porque hay una restricción

Nombres de rol externos en asociaciones binarias

Sea $\boldsymbol{R}(\boldsymbol{p}_1{:}\boldsymbol{E}_1\ [min_1, max_1], \boldsymbol{p}_2{:}\boldsymbol{E}_2\ [min_2, max_2])$ una asociación binaria, entonces

$\boldsymbol{G}_{3f}^{r}$: Suponiendo que $\boldsymbol{p}_i$ es un nombre de rol externo, $\boldsymbol{p}_i$ tiene que ser sintagma nominal cuyo núcleo sea contable y singular, escrito utilizando la "forma Camel".

$\boldsymbol{G}_{3s}^{r}$: Una de las siguientes frases tiene que ser gramaticalmente correcta y tener sentido:

- Si $min_i = 0$ y $max_i = 1, \boldsymbol{i} \neq \boldsymbol{j}$:

  [Un|Una] *minúsculas*($\boldsymbol{E}_j$) puede tener [un|una] *minúsculas*($\boldsymbol{p}_i$).

- Si $min_i = 0$ y $max_i > 1, \boldsymbol{i} \neq \boldsymbol{j}$:

  [Un|Una] *minúsculas*($\boldsymbol{E}_j$) puede tener cero o más *minúsculas*(plural($\boldsymbol{p}_i$)).

- Si $min_i = max_i = 1, \boldsymbol{i} \neq \boldsymbol{j}$:

  [Un|Una] *minúsculas*($\boldsymbol{E}_j$) tiene [un|una] *minúsculas*($\boldsymbol{p}_i$).

- Si $min_i > 0$ y $max_i > 1, \boldsymbol{i} \neq \boldsymbol{j}$:

  [Un|Una] *minúsculas*($\boldsymbol{E}_j$) tiene [un|una] o más *minúsculas*(plural($\boldsymbol{p}_i$)).

plural($\boldsymbol{p}_i$) es una función que devuelve la forma plural de $\boldsymbol{p}_i$.

Tabla 6. Regla para los nombres de rol de las asociaciones binarias en UML.

Asociaciones *n*-arias

Sea $\boldsymbol{R}(\boldsymbol{p}_1{:}\boldsymbol{E}_1\ [min_1, max_1], \ldots, \boldsymbol{p}_n{:}\boldsymbol{E}_n\ [min_n, max_n])$ una asociación $\boldsymbol{n}$-aria, entonces

$\boldsymbol{G}_{4f}$: El nombre de la asociación $\boldsymbol{R}$ tiene que ser un sintagma verbal en tercera persona del singular, utilizando la "forma Pascal". El nombre debe incluir $\boldsymbol{n}-1$ subcadenas de texto de la forma <$\boldsymbol{p}_i$>, una por cada rol $\boldsymbol{p}_2, \ldots, \boldsymbol{p}_n$, donde $\boldsymbol{p}_i$ es el nombre de rol explícito (o implícito si no tiene). Los nombres de rol se escriben usando la "forma Camel".

En UML, los participantes de una asociación están ordenados, pero para las asociaciones $\boldsymbol{n}$-arias este orden no se muestra gráficamente. Fíjense que el primer participante $\boldsymbol{p}_1$ de la asociación es el que no aparece en $\boldsymbol{R}$.

$\boldsymbol{G}_{4s}$: La siguiente frase tiene que ser gramaticalmente correcta y tener sentido

[Un|Una] *minúsculas*($\boldsymbol{E}_1$) *minúsculas*(expanded($\boldsymbol{R}$))

La función *expanded*($\boldsymbol{R}$) devuelve $\boldsymbol{R}$ en minúsculas y utilizando el espacio como delimitador y substituye cada <$\boldsymbol{p}_i$> por "[un|una] ($\boldsymbol{E}_i$)". Fíjense que los nombres de rol no aparecen en el resultado de *expanded*($\boldsymbol{R}$).

Tabla 7. Regla para los nombres de rol de las asociaciones binarias en UML.

en UML que impide que un tipo de entidad (como *Persona*) pueda participar en dos asociaciones tales que los demás participantes tengan el mismo nombre de rol.

4.2. Asociaciones n-arias

Sea $\boldsymbol{R}(\boldsymbol{p}_1{:}\boldsymbol{E}_1\ [min_1, max_1], \ldots, \boldsymbol{p}_n{:}\boldsymbol{E}_n\ [min_n, max_n])$ una asociación $\boldsymbol{n}$-aria ($\boldsymbol{n} > 2$) entre los tipos de entidad $\boldsymbol{E}_1, \ldots, \boldsymbol{E}_n$, cuyos roles son $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$, respectivamente. En UML hay hasta $\boldsymbol{n} + 1$ nombres explícitos vinculados a esta asociación: el nombre de la asociación $\boldsymbol{R}$ y los nombres de rol $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$.

Los roles $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ siempre tienen un nombre implícito que se utiliza cuando no se les ha dado uno explícito. Este nombre implícito es el mismo nombre que el tipo de entidad al que están vinculados, empezando en minúscula. Cuando $\boldsymbol{E}_i = \boldsymbol{E}_j$, entonces $\boldsymbol{p}_i$ o $\boldsymbol{p}_j$ o ambos roles tienen que tener un nombre explícito. No hay nombres implícitos para las asociaciones $\boldsymbol{n}$-arias. Dos asociaciones distintas pueden tener el mismo nombre.

La regla $\boldsymbol{G}_4$ que proponemos para las asociaciones UML $\boldsymbol{n}$-arias es la mostrada en la **tabla 7**.

Los ejemplos de la **figura 3** producen el siguiente resultado:

Suministra<*producto*>*A*<*proyecto*> genera:

> Un proveedor suministra un producto a un proyecto.

Vende<*producto*>*De*<*proveedor*>*En*<*pais*> genera:

> Un vendedor vende un producto de un proveedor en un país.

PuedeSustituir<*sustituto*>*En*<*proyecto*> genera:

> Un producto puede sustituir un producto en un proyecto.

5. Invariantes

En UML, un invariante es una restricción estática vinculada a un tipo de entidad. La restricción puede estar definida como una expresión OCL cuyo contexto es el tipo de entidad. Esta expresión debe ser cierta para todas las instancias de ese tipo. Los invariantes pueden tener nombre. Por ejemplo:

```
context Departamento inv
tieneSuficientesEmpleados:
 self.numeroDeEmpleados > 50
```

es un invariante denominado tieneSuficientesEmpleados.

En UML, las restricciones *untargeted* deben definirse como invariantes cuyo contexto sea un tipo de entidad *singleton*. Por ejemplo, un sistema que gestione congresos podría tener una restricción indicando que "debe haber como mínimo una sala grande", la cual podría expresarse como el siguiente invariante:

```
context GestorDeCongresos inv
tieneComoMinimoUnaSalaGrande:
 Sala.allInstances()->select(tamaño =
Tamaño::grande)->size() > 0
```

donde se supone que *GestorDeCongresos* es un tipo de entidad *singleton*, que *tamaño* es un atributo de *Sala* y que *Tamaño* es un tipo de datos enumerado.

La regla que proponemos para denominar un invariante $\boldsymbol{I}$ es la mostrada en la **tabla 8**.

Los siguientes ejemplos muestran la aplicación de nuestra regla:

```
context Persona inv seIdentificaPorSuNom-
bre:
```

```
context Matrimonio inv es una relación entre
un hombre y una mujer:
```

```
context Seccion inv seIdentificaPorElCurs-
oYElNombre:
```

context Curso inv noPuedeSerSucesorDeSi-Mismo:

context Curso inv consta de secciones cuyos profesores son
expertos en el tema del curso:

los cuales generan las siguientes frases:

Una persona se identifica por su nombre.
Un matrimonio es una relación entre un hombre y una mujer.
Una sección se identifica por el curso y el nombre.
Un curso no puede ser sucesor de sí mismo.
Un curso consta de secciones cuyos profesores son expertos en el tema del curso.

6. Tipos de evento y operaciones de sistema

A continuación nos centraremos en la parte de comportamiento de los esquemas conceptuales. En general, un esquema de comportamiento se basa en un conjunto de tipos de evento u operaciones de sistema, y/o un conjunto de diagramas de transición de estados.

En esta sección veremos los tipos de evento (u operaciones de sistema) y, en la siguiente, los diagramas de transición de estados.

6.1. Tipos de evento

Sea *Ev* un tipo de evento. La regla que proponemos para nombrarlo es la mostrada en la **tabla 9**.

La **figura 4** muestra un ejemplo con tres tipos de evento: *NuevaVenta*, *NuevoItemDeVenta* y *FinDeVenta*, los cuales generan las siguientes frases:

Una instancia de este tipo de evento es un evento de nueva venta.
Una instancia de este tipo de evento es un evento de nuevo item de venta.
Una instancia de este tipo de evento es un evento de fin de venta.

6.2. Operaciones de sistema

Sea *Op* una operación de sistema. La regla que proponemos para las operaciones de sistema es la mostrada en la **tabla 10**.

Fíjense que la frase incluye simplemente *Op*, en lugar de *minúsculas*(*Op*). El motivo es que los nombres que se dan a la mayoría de operaciones son, en la práctica, descripciones compactas de lo que harán cuando se invoquen. Por lo tanto, no parece posible dar una regla general para su correcta verbalización.

La **figura 4** muestra también tres operaciones de sistema, denominadas *CreaNuevaVenta*, *CreaItemDeVenta* y *FinalizaVenta*, las cuales se corresponden a los tipos de evento que ya hemos comentado. Las frases generadas son:

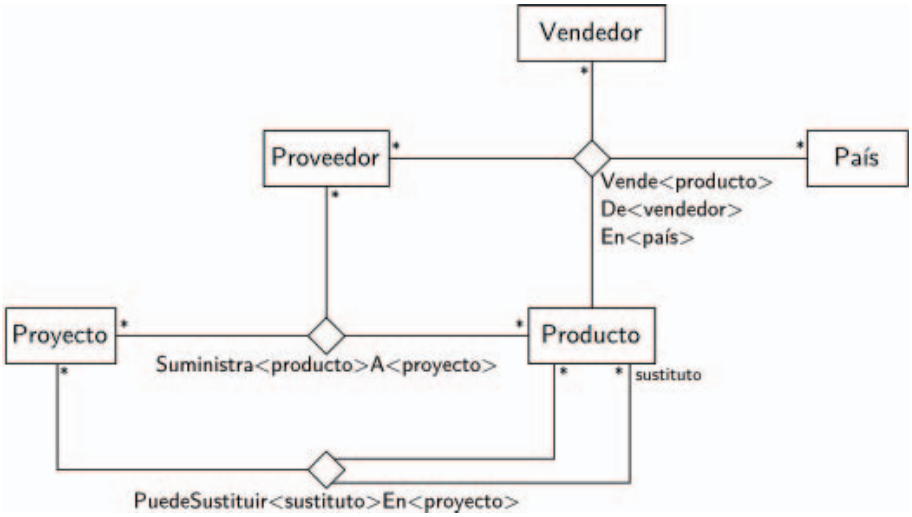

Figura 3. Ejemplos de asociaciones n-arias.

Invariantes

G_{5f}: El nombre de un invariante tiene que ser un sintagma verbal en tercera persona del singular. El nombre tiene que estar escrito usando la "forma Camel" o, si no hay ninguna palabra que pueda generar ambigüedad con OCL, usando la forma normal (es decir, usando minúsculas y espacios).

G_{5s}: Si ***I*** es el nombre del invariante y ***E*** su contexto, entonces la siguiente frase tiene que ser gramaticalmente correcta y tener sentido:

[Un|Una] *minúsculas*(***E***) *minúsculas*(***I***).

Tabla 8. Regla para denominar un invariante.

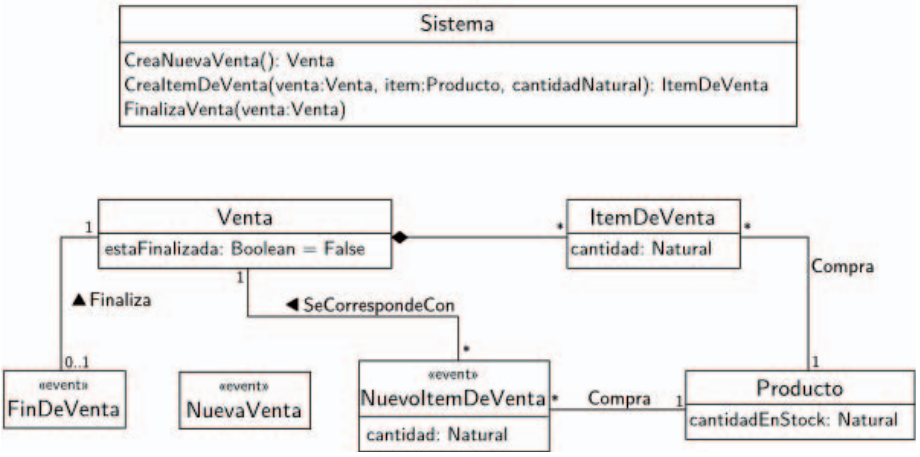

Figura 4. Fragmento de un esquema estructural y la especificación de su comportamiento utilizando tipos de evento y operaciones de sistema.

Una invocación a esta operación pide al sistema que ejecute el comando CreaNuevaVenta.
Una invocación a esta operación pide al sistema que ejecute el comando CrealtemDeVenta.
Una invocación a esta operación pide al sistema que ejecute el comando FinalizaVenta.

6.3. Precondiciones

Una precondición es una condición que tienen que satisfacer las características del evento y los parámetros de la operación y/o el sistema de información cuando el evento o la operación va a producirse. En UML, las precondiciones pueden tener un nombre y se especifican formalmente en OCL.

Sea *Pre* una precondición del tipus de evento *Ev* o de la operación de sistema *Op*. La regla que proponemos para darle nombre es la mostrada en la **tabla 11**.

En el ejemplo de la **figura 4**, el tipo de evento *NuevoItemDeVenta* (o la operación de siste-

Tipos de evento

G_{6f} : El nombre de un tipo de evento tiene que ser un sintagma nominal cuyo núcleo sea contable y singular. El nombre tiene que escribirse usando la "forma Pascal". Fíjense que esta regla es igual a la propuesta para los tipos de entidad (G_{1f}).

G_{6s} : Si *Ev* es el nombre de un tipo de evento, entonces la siguiente frase tiene que ser gramaticalmente correcta y tener sentido:

Una instancia del tipo de evento denominado *Ev* es un evento de *minúsculas*(*Ev*).

Cuando el tipo de un evento es implícito por el contexto, la frase es simplemente:

Una instancia de este tipo de evento es un evento de *minúsculas*(*Ev*).

Tabla 9. Regla para nombrar tipos de evento.

Operaciones de sistema

G_{7f}: El nombre de una operación de sistema tiene que ser un sintagma verbal, en imperativo. El nombre debe escribirse usando la "forma Pascal".

G_{7s}: Si *Op* es el nombre de la operación de sistema, entonces la siguiente frase tiene que ser gramaticalmente correcta y ten er sentido:

Una invocación a la operación denominada *Op* pide al sistema que ejecute el comando *"Op"*.

Cuando la operación es implícita por el contexto, la frase es:

Una invocación a esta operación pide al sistema que ejecute el comando *"Op"*.

Tabla 10. Regla para denominar operaciones de sistema.

Precondiciones

G_{8f}: El nombre de una precondición tiene que ser un sintagma verbal. El nombre tiene que estar escrito en "forma Camel" o usando la forma "normal" (ver G_{7f}).

G_{8s}: Si *Pre* es el nombre de la precondición, entonces la siguiente frase tiene que ser gramaticalmente correcta y tener sentido:

[Antes que suceda el evento de *minúsculas*(Ev) | Antes que se ejecute la operación Op], *minúsculas*(Pre).

Cuando el tipo de evento o la operación de sistema es implícito por el contexto, la frase es:

[Antes que suceda este evento | Antes que se ejecute esta operación], *minúsculas*(Pre).

Tabla 11. Regla para nombrar las precondiciones.

ma equivalente *CreaItemDeVenta*) tiene dos precondiciones con los nombres: "la venta no está finalizada" y "hay suficiente stock":

context NuevoItemDeVenta::effect() pre la venta no esta finalizada:
not venta.estaFinalizada pre hay suficiente stock: cantidad >= producto.cantidadEnStock

el cual genera las siguientes frases:

Antes que suceda este evento, la venta no está finalizada.
Antes que suceda este evento, hay suficiente stock.

6.4. Postcondiciones

El efecto de los eventos o las operaciones de sistema se describe mediante una o más postcondiciones.

Una postcondición es una condición que el sistema de información tiene que satisfacer después que el evento haya sucedido (o que la operación se haya ejecutado). En general, una postcondición también puede incluir información sobre las salidas generadas (*queries* o *communications*). En UML, las postcondiciones pueden tener nombre y se especifican formalmente en OCL.

Sea *Post* una postcondición del tipo de evento *Ev* o de la operación de sistema *Op*. La regla que proponemos para denominar las postcondiciones se muestra en la **tabla 12**.

En el ejemplo de la **figura 4**, el tipo de evento *NuevaVenta* (o la operación de sistema equivalente *CreaNuevaVenta*) tiene una postcondición denominada "se ha creado una venta":

context NuevaVenta::effect() post se ha creado una venta:
venta.oclIsNew() and venta.oclIsTypeOf(Venta)

la cual genera la siguiente frase:

Después que suceda este evento, se ha creado una venta.

De la misma forma, el tipo de evento *NuevoItemDeVenta* (o la operación de sistema equivalente *CreaItemDeVenta*) tiene dos postcondiciones denominadas: "se ha creado un nuevo item de venta" y "se ha reducido el stock del producto", las cuales generan:

Después que suceda este evento, se ha creado un nuevo ítem de venta.
Después que suceda este evento, se ha reducido el stock del producto.

7. Estados

En UML, un estado es una "condición o situación durante la vida de un objecto en donde satisface una cierta condición, ejecuta una cierta actividad, o espera algún evento".

Sea ***S*** un estado de un tipo de entidad ***E***. La regla que proponemos para denominar ***S*** es la mostrada en la **tabla 13**.

A continuación tienen unos cuantos ejemplos:

- *Microondas*: *ListoParaCocinar*, *PuertaAbierta*, *Cocinando*, …
- *Paciente*: *Entrando*, *Admitido*, *BajoObservación*, *DadoDeAlta*.
- *LlamadaTelefónica*: *NoDisponible*, *SeñalDeLlamada*, *Llamando*, …

Algunas de las frases que se generarían con los anteriores ejemplos son:

Un microondas está cocinando.
Un paciente está entrando.
Un paciente está admitido.

Postcondiciones

G_{9f}: El nombre de una postcondición tiene que ser un sintagma verbal. El nombre tiene que estar escrito en "forma Camel" o usando la forma "normal" (ver G_{7f}).

G_{9s}: Si *Post* es el nombre de la postcondición, entonces la siguiente frase tiene que ser gramaticalmente correcta y tener sentido:

[Despúes que suceda el evento de *minúsculas*(Ev) | Después que se invoque la operación Op], *minúsculas*(Post).

Cuando el tipo de evento o la operación de sistema es implícito por el contexto, la frase es:

[Después que suceda este evento | Después que se invoque esta operación], *minúsculas*(Post).

Tabla 12. Regla para denominar las postcondiciones.

Estados

G_{10f}: El nombre de un estado tiene que ser un adjetivo o complemento nominal, escrito utilizando la "forma Pascal".

G_{10s}: Si ***S*** es el nombre de un estado de un tipo de entidad ***E***, entonces una de las siguientes frases como mínimo tiene que ser gramaticalmente correcta y tener sentido:

[Un|Una] minúsculas(***E***) [es|está] minúsculas(***S***).
[Un|Una] minúsculas(***E***) está en el estado de minúsculas(***S***).

Tabla 13. Regla para denominar estados.

Nota

[1] El texto completo de la guía puede verse en el artículo de los mismos autores: *A complete set of guidelines for naming UML conceptual schema elements*. Data and Knowledge Engineering vol 88, pp. 60-74 (2013). Hay una versión de la guía para las lenguas catalana, castellana e inglesa.