Ingeniería de Software I

1er cuatrimestre de 2006

Modelo Conceptual

de una

Fábrica Automatizada de Productos

Departamento de Computación
Facultad de Ciencias Exactas
Universidad de Buenos Aires

Enunciado

1. Objetivos

El objetivo de este ejercicio es efectuar un modelo conceptual de la siguiente fábrica automatizada de productos. Los invariantes y otras restricciones (constraints) que no se desprendan del o los diagramas de clases conceptuales deberán ser especificados en lenguaje coloquial y en OCL.

Se debe detallar cualquier suposición sobre el funcionamiento de la fábrica que no esté explícita en el enunciado y cualquier decisión tomada que se considere relevante.

2. Funcionamiento de la fábrica

La fábrica que se pide modelar genera productos manufacturados a partir de otros productos. La fabricación está totalmente automatizada, en el sentido de que la fábrica no necesita que se le indique qué productos manufacturados se deben fabricar. La fábrica misma es la que determina la fabricación, en función de los productos (clasificados en materias primas y productos manufacturados) que tiene en su depósito.

2.1 Ingreso de materias primas

Desde el exterior sólo se puede ingresar a materias primas, es decir, no es posible ingresar productos manufacturados. Al efectuar el ingreso, la fábrica ubica lo ingresado temporalmente en su depósito.

2.2 Depósito

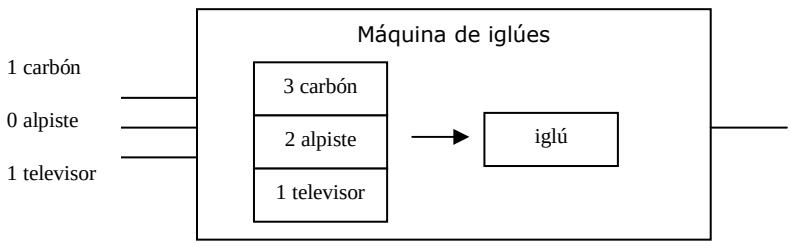
La fábrica cuenta con un depósito donde se almacenan tanto las materias primas (del exterior) como los productos manufacturados producidos por la misma fábrica. Frente al incremento de productos (de cualquiera de ambos tipos) el depósito debe informarle del hecho a la fábrica para que éstos sean considerados en el proceso de fabricación general.

El depósito admite una cantidad ilimitada de productos.

2.3 Fabricación

Para fabricar productos manufacturados, la fábrica cuenta con un conjunto de máquinas. Cada una de las máquinas necesita una cantidad de productos determinados como entrada para producir una unidad de **un único** producto manufacturado.

La fábrica debe ser la encargada de ir distribuyendo los productos que tiene en el depósito entre las máquinas, e indicarles que produzcan. Cada máquina recibe los productos que la fábrica le entrega hasta el momento en que se llena y no recibe más productos. Una máquina está llena cuando alcanza el conjunto de productos de entrada necesarios para producir el tipo de producto manufacturado determinado por su fórmula. Un ejemplo de una posible máquina es el siguiente:



El diagrama de productos que figura en el interior de la máquina es lo que llamaremos la *fórmula* de la máquina. Una máquina sabe a través de su fórmula qué y cuántos productos de entrada se necesitan para generar un producto manufacturado. Cuidado, no confundir la fórmula de producción de una máquina con los productos con los que cuenta la máquina en un momento dado. Notar que una máquina puede tomar como entrada productos manufacturados y materias primas.

La fábrica puede indicarle a una máquina, en un momento dado, que produzca su producto. Como mencionamos antes, la máquina sólo va a poder producirlo cuando tenga todos los productos necesarios para ello. En el caso de no poseerlos, debe ignorar el pedido de producción de la fábrica.

Cuando una máquina produce un nuevo producto manufacturado debe avisarle a la fábrica, quien debe tomarlo e ingresarlo al depósito.

2.4 Distribución de productos

Cuando el depósito le avisa a la fábrica que un nuevo producto ha sido ingresado, la fábrica debe encargarse de distribuir el contenido del depósito entre las máquinas. Para ello debe egresarlos del depósito e ingresarlos en las máquinas haciendo uso de alguna **estrategia de distribución** que determina el criterio de distribución de los productos en las máquinas. Al finalizar la distribución de los productos, la fábrica debe indicarle a todas sus máquinas que produzcan, en el caso que puedan.

3. Otras consideraciones

1. Todos los productos tienen un costo (al público) determinado. Las materias primas tienen un costo fijo y los productos manufacturados tienen un costo que es la suma directa de sus componentes. Los productos manufacturados saben su fecha de producción y qué máquina los produjo.
2. Una máquina sabe la cantidad de productos que produjo y cuándo produjo su último producto.
3. Durante el período de vida de una fábrica pueden agregarse desde el exterior tanto materias primas como nuevas máquinas, el proceso de fabricación debe tener en cuenta estos cambios. Se supone que una vez ingresada una máquina a la fábrica, su fórmula no cambia. La fábrica puede aceptar cualquier tipo de máquina independientemente de su fórmula.
4. Se debe poder consultar el stock de productos que la fábrica tiene en un momento determinado.
5. La estrategia de distribución de la fábrica debe poder intercambiarse con facilidad. Por ahora las estrategias son:

    5.1. **Golosa**: se asigna un producto a la primera máquina que lo necesita.

    5.2. **Equilibrada**: entre todas las máquinas que necesitan un mismo producto, se le entrega a la máquina que lleva más tiempo sin producir nada.

    5.3. **Exitista**: entre todas las máquinas que necesitan un mismo producto, se le entrega a la máquina que más productos produjo en total.

    5.4. **Productiva:** entre todas las máquinas que necesitan un mismo producto, se le entrega a la máquina que le falta menos unidades de productos para estar llena.

    Es posible que se agreguen nuevas estrategias en el futuro y esto también debe poder hacerse con facilidad.

Resolución

Conceptos clave:

dominio del problema,
explorar,
observador,
concepto, ente, entidad,
clase conceptual,
clase conceptual de asociación,
atributo, propiedad,
asociación simple,
agregación, agregación compartida,
composición , agregación compuesta,
multiplicidad,
rol,
herencia,
jerarquía de clases,
generalización, especialización,
superclase, subclase,
diagrama de clases conceptuales,
OCL, invariantes, restricciones,
modelo conceptual, modelo de dominio, modelo de objetos del dominio,
refinamiento, iteración.

Algunos de los conceptos mencionados pueden no aparecer en esta clase.

Respecto a la forma de resolución

Iteraremos en nuestro modelo conceptual con el objetivo de ir refinándolo.

Aclaraciones

La ausencia de multiplicidad en los diagramas de clases conceptuales debe ser interpretada como multiplicidad 1.

Los guiones, '-', que aparecen delante cada atributo de una clase conceptual y delante de cada rol en los diagramas de clases no deben ser considerados.

Observación

El enunciado no dice nada sobre la remoción de máquinas ni tampoco sobre la extracción de productos manufacturados del depósito desde el exterior. Si esto fuera posible habría que pensar en ciertas cuestiones tales como que un producto manufacturado podría no conocer la máquina que lo fabricó.

Asumiremos que ni las máquinas, ni los productos manufacturados pueden ser extraídos de la fábrica.

Comenzaremos la exploración del dominio del problema identificando conceptos y relaciones entre estos que plasmaremos en un diagrama mediante clases conceptuales, asociaciones simples y relaciones de generalización/especialización (conocidas como relaciones "es-un"). Proseguiremos añadiendo multiplicidades y roles en los extremos de las asociaciones.

Asumimos que un producto manufacturado se compone de los productos usados por la máquina productora para producirlo. Es decir, que si por ejemplo, una fórmula de una máquina dice que debe usarse carbón, éste formará parte del producto manufacturado que produzca la máquina. No podremos pensar, por ejemplo, que el carbón fue usado como combustible para la producción del producto manufacturado no formando parte de éste.

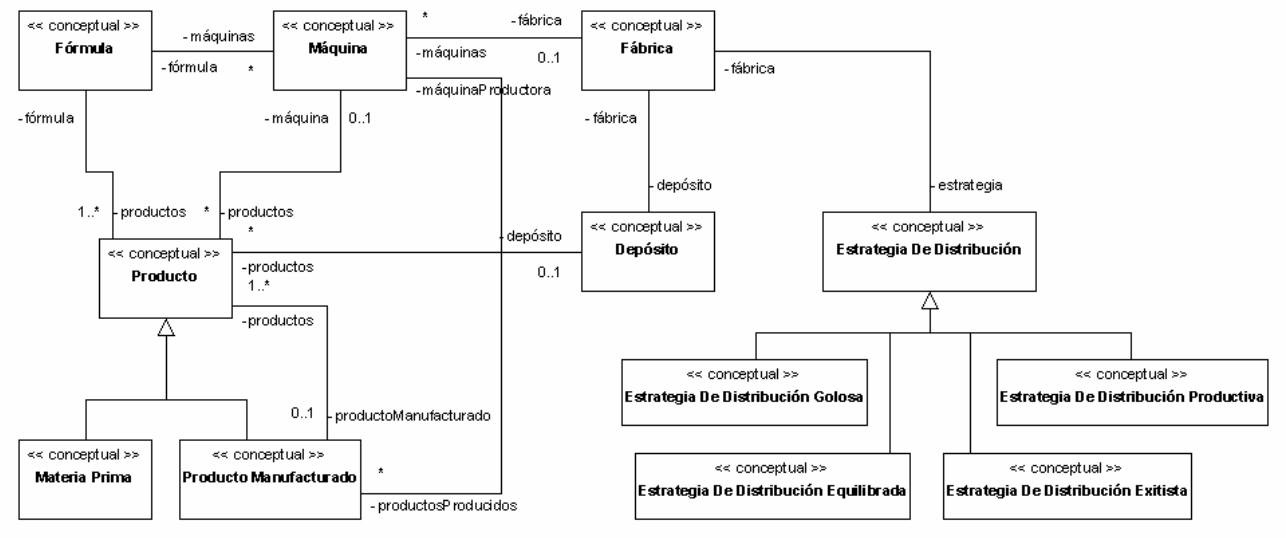


Figura 1

Por enunciado sabemos que "Una máquina sabe a través de su fórmula qué y cuántos productos de entrada se necesitan para generar un producto manufacturado" es por ello que relacionamos una fórmula con productos.

Asumimos que:
- una fábrica sólo puede tener un depósito,
- una fábrica podría no tener máquinas y
- un depósito podría no tener productos

Proseguimos explorando el dominio para ahora identificar atributos o propiedades, de las clases conceptuales. Surge entonces la idea de tipo para producto, por ejemplo, en nuestra fábrica podrá haber 50 patitos de hule que son 50 objetos conceptuales todos de la misma índole. A esta idea de índole la llamaremos tipo de un producto. Una fórmula será de un tipo de producto y a través de ésta se sabrá qué tipos de productos y qué cantidades son necesarias para producir un producto manufacturado con tipo el que indica la fórmula.

Llamaremos ítem de una fórmula a aquello que relaciona un tipo de producto con una cantidad que indicará la cantidad de productos de este tipo.

Por lo comentado anteriormente, una fórmula deja de estar relacionada con un producto ya que entró en juego la idea de tipo de producto.

Notamos que el costo de un producto manufacturado es un atributo derivado.

Por ahora nuestro modelo conceptual tiene la siguiente pinta:

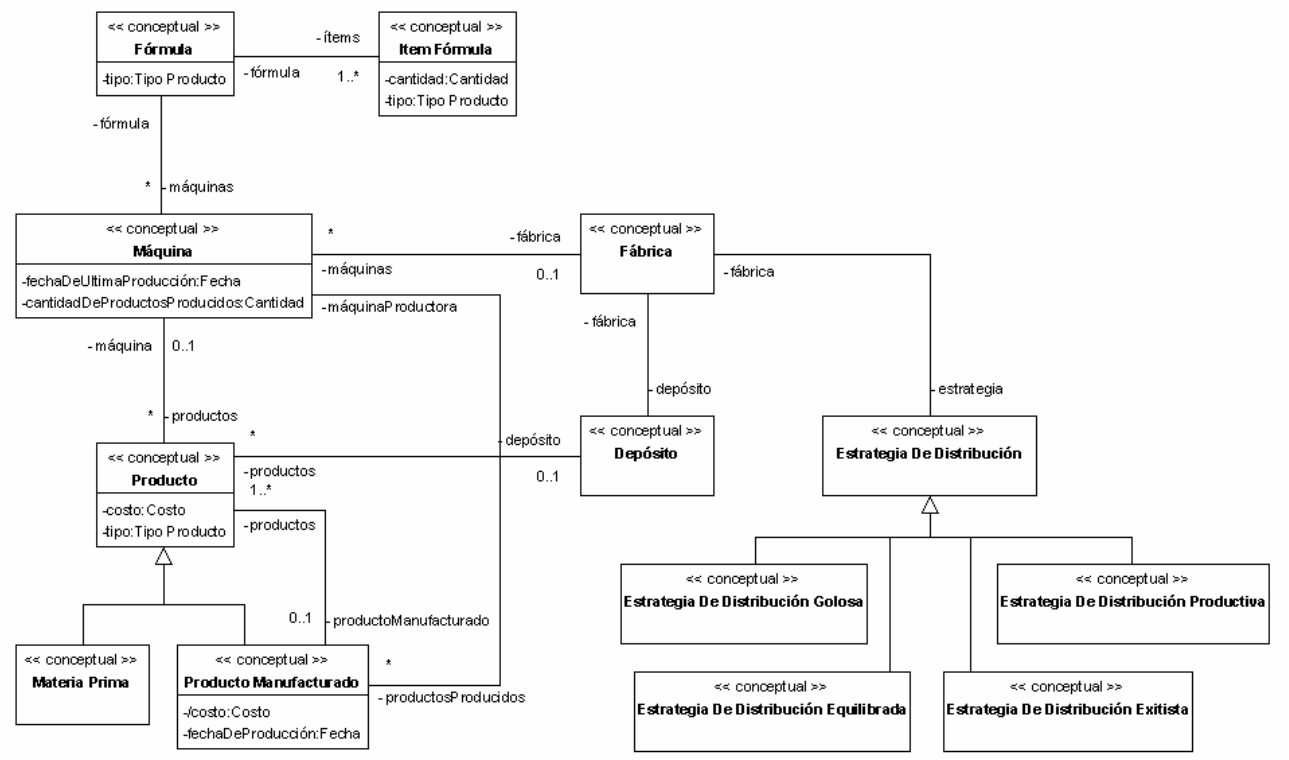


Figura 2

Trataremos ahora de identificar agregaciones y composiciones.

Asumimos que una fórmula es responsable de la existencia de sus ítems, es decir, los ítems de una fórmula existen a partir de la existencia de ésta última y si una fórmula dejara de existir sus ítems también, además, un ítem fórmula sólo puede ser parte de una única fórmula; con lo cual tenemos una relación de composición de una fórmula a sus ítems. Podemos aplicar la misma idea para la relación entre la fábrica y el depósito.

Respecto a los productos, su existencia no depende de la existencia de una máquina ni de la del depósito por lo cual usaremos una agregación.

Nos queda analizar el caso de la relación entre un producto manufacturado y sus productos componentes que a priori pareciera ser una relación de composición. Ya sabemos que en nuestro dominio del problema un mismo producto no puede ser componente de dos productos manufacturados a la vez, sin embargo, la existencia de los productos componentes de un producto manufacturado no depende de éste con lo cual no corresponde que usemos composición para este caso. Por otra parte, si un producto manufacturado dejara de existir sus productos componente también deberían dejar de existir pero no nos preocuparemos de esto por ahora dada la suposición hecha al comenzar la resolución. Deberemos entonces usar una agregación y restringir que un mismo producto sólo puede ser componente de un único producto manufacturado, esto último lo haremos luego usando OCL.

Nos queda lo siguiente:

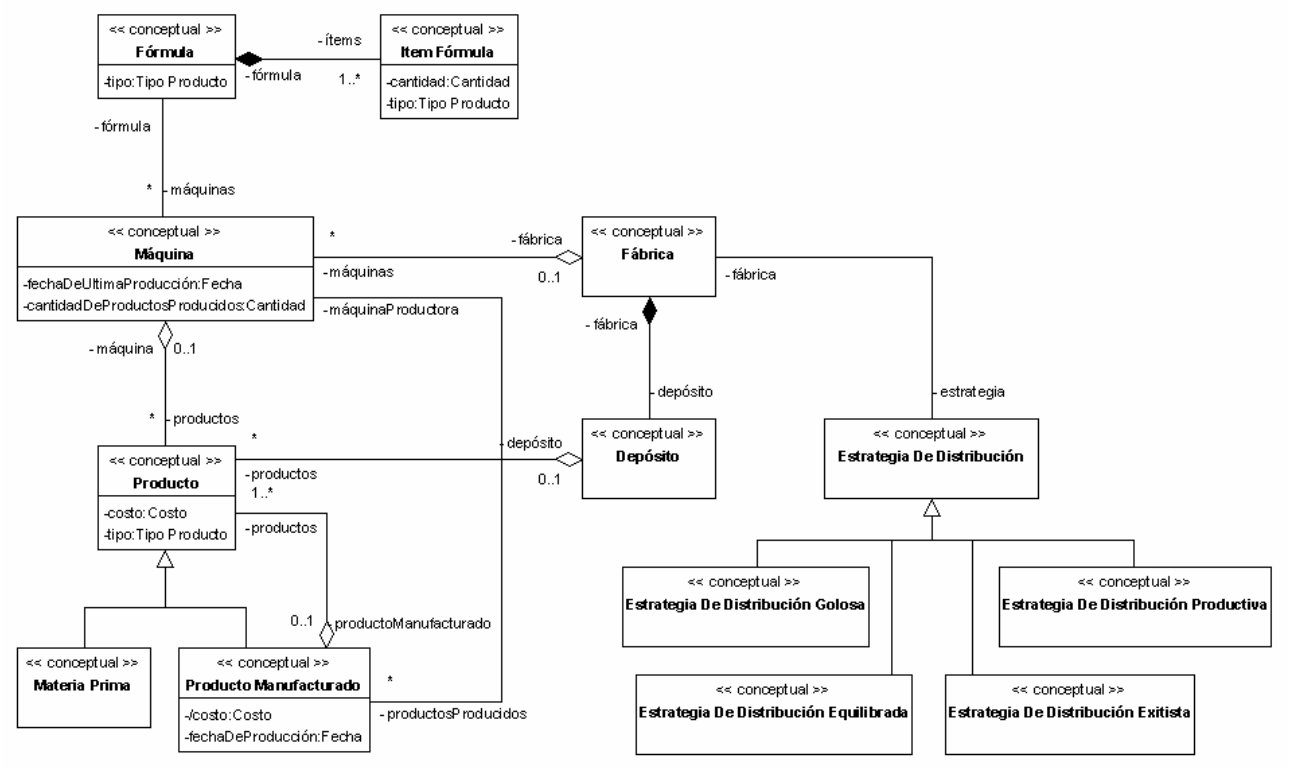


Figura 3

Reflexionamos y nos damos cuenta que el concepto de tipo de producto es relevante en el dominio del problema representando al mismo como una clase conceptual. La pregunta que nos surge entonces es ¿cuándo modelamos un concepto como una clase conceptual o como un atributo? Pensemos lo siguiente, las cantidades y fechas son conceptos, sin embargo, no poseen la suficiente importancia como para ser modelados como clases conceptuales. La clave está en saber identificar aquellos conceptos que sí son relevantes en un determinado dominio.

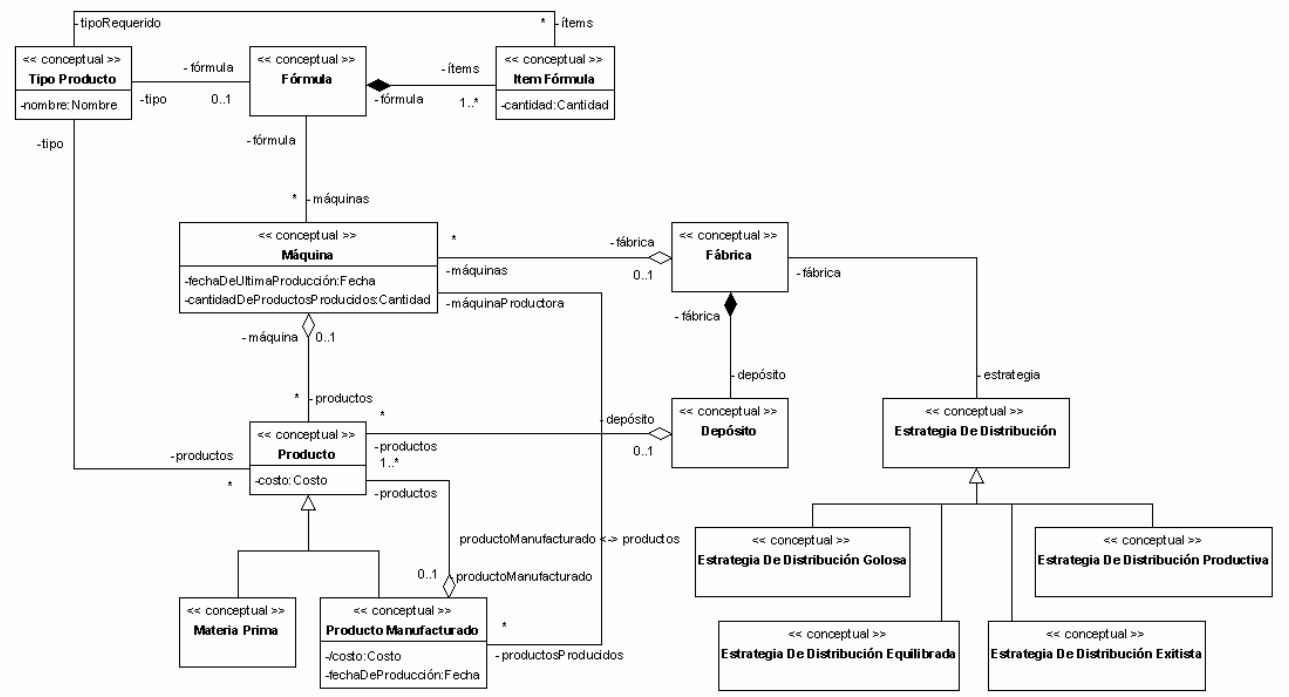


Figura 4

Se nos ocurre otra forma de expresar que una fórmula sepa qué tipos de productos y qué cantidades son necesarias para producir un producto manufacturado con tipo el que indica la fórmula:

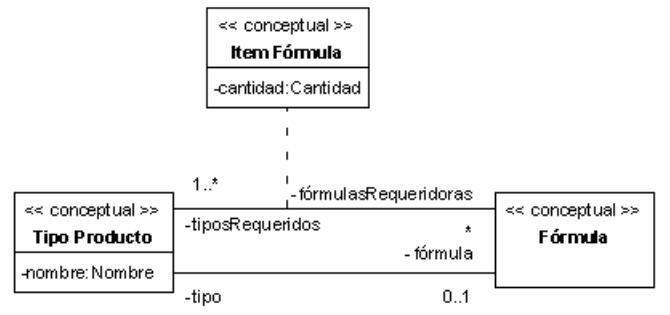


Figura 5

Usaremos esta forma para que en nuestro diagrama de clases conceptuales aparezca una clase de asociación y podamos usarla en restricciones que luego escribamos en OCL a modo de ejemplificar cómo notar esto.

Notamos que en nuestro dominio no podrá haber objetos conceptuales que sean un producto sin ser o bien una materia prima o bien un producto manufacturado por lo cual la clase conceptual Producto será *abstracta*. Lo mismo ocurre entre la clase conceptual Estrategia De Distribución y sus especializaciones. Cada una de ambas subclaseaciones, la de Producto y la de Estrategia De Distribución, es *completa* y *disjunta*.

Nos queda entonces lo siguiente:

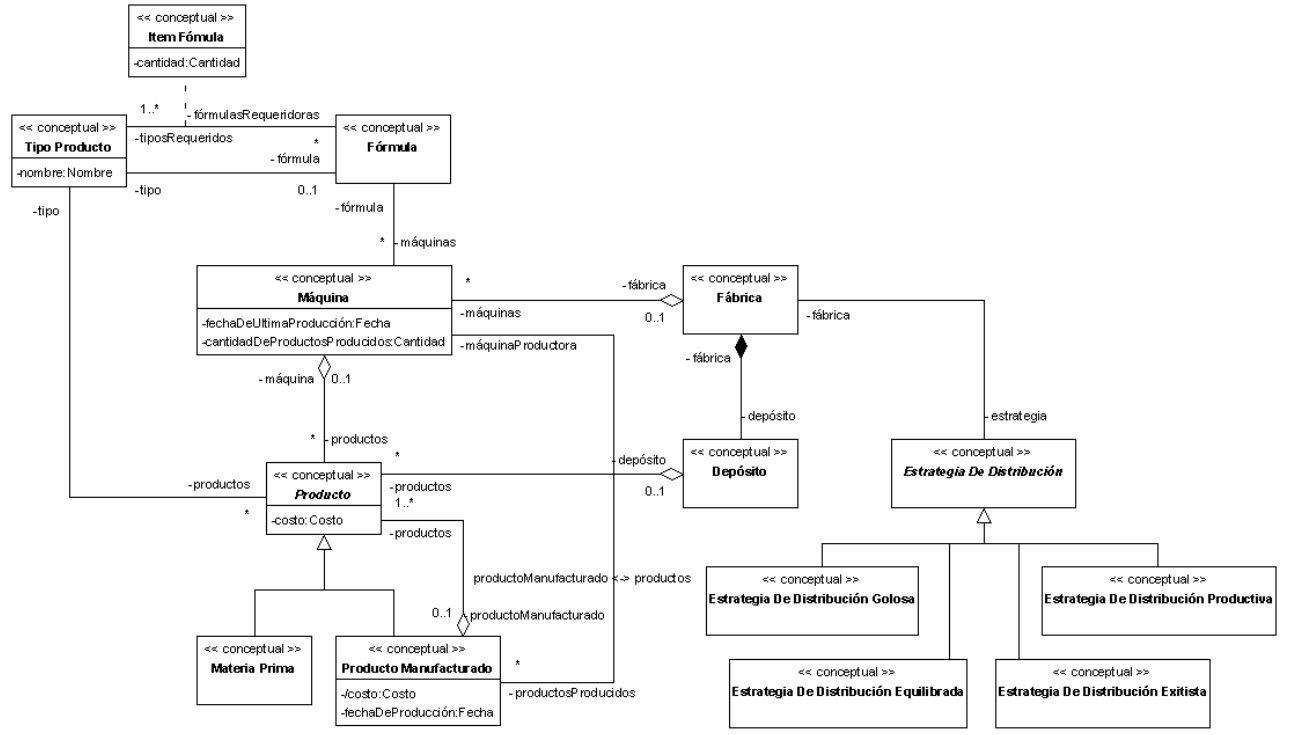


Figura 6

Parece que llegamos a una versión final de nuestro modelo.

Pensamos un poco más y nos preguntamos si el costo de un producto podría ser algo que sepa un tipo producto. Si asumiéramos que todos los productos de un mismo tipo tienen el mismo costo podríamos entonces proponer que un producto sepa su costo a través de su tipo. Del enunciado no se desprende nada sobre cómo un producto sabe su costo, por ejemplo, si lo "consulta" a su tipo, sólo sabemos que su forma de calcularlo variará dependiendo de si es materia prima o producto manufacturado. La pregunta es entonces ¿a quién le corresponde conocer el costo de un producto, al producto mismo o a su tipo? Sabemos, por ejemplo, que el precio de dos patitos de hule cualesquiera es el mismo con lo cual pareciera ser algo que debería saber el tipo de producto Patito de Hule en lugar de cada uno de todos los patitos. Nos quedamos entonces con que el costo de un producto puede ser un atributo de la clase conceptual Tipo Producto. Sabemos también que el costo de un tipo de producto de un producto manufacturado es calculable a través de la suma de los costos de sus productos componente mientras que el del tipo de producto de una materia prima es fijo. Mostremos esto en el diagrama a través de una jerarquía de clases y un atributo derivado. Paremos un segundo y formulémonos las siguientes preguntas: ¿nos interese realmente modelar esta jerarquía?, ¿se desprende de lo que conocemos? y ¿podemos asumirla? El punto es que si llegamos a la conclusión de que el costo de un producto debe ser conocido por su tipo pues entonces deberemos mostrar tal jerarquía dada la diferencia de cálculo del costo (dependiendo de si el tipo de producto es de materia prima o de producto manufacturado).

A continuación mostramos cómo nos termina quedando el modelo conceptual mediante dos diagramas de clases conceptuales:

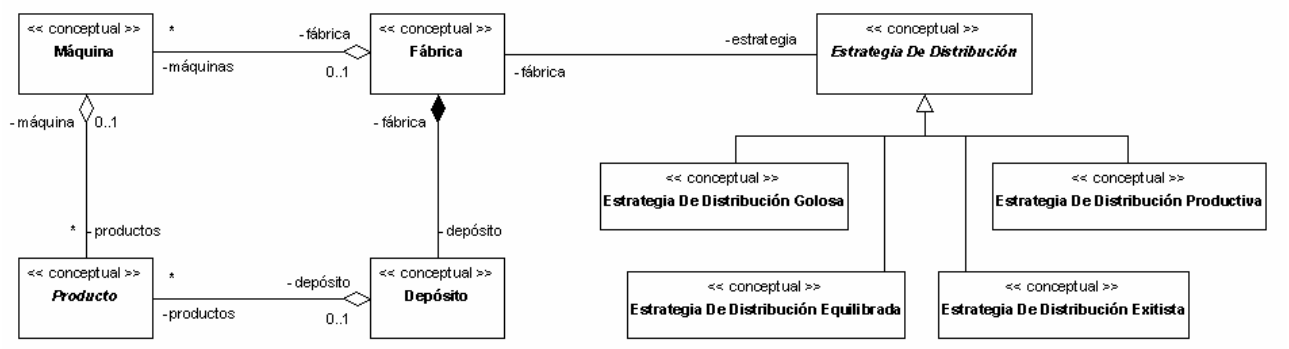


Figura 7 - Diagrama 1

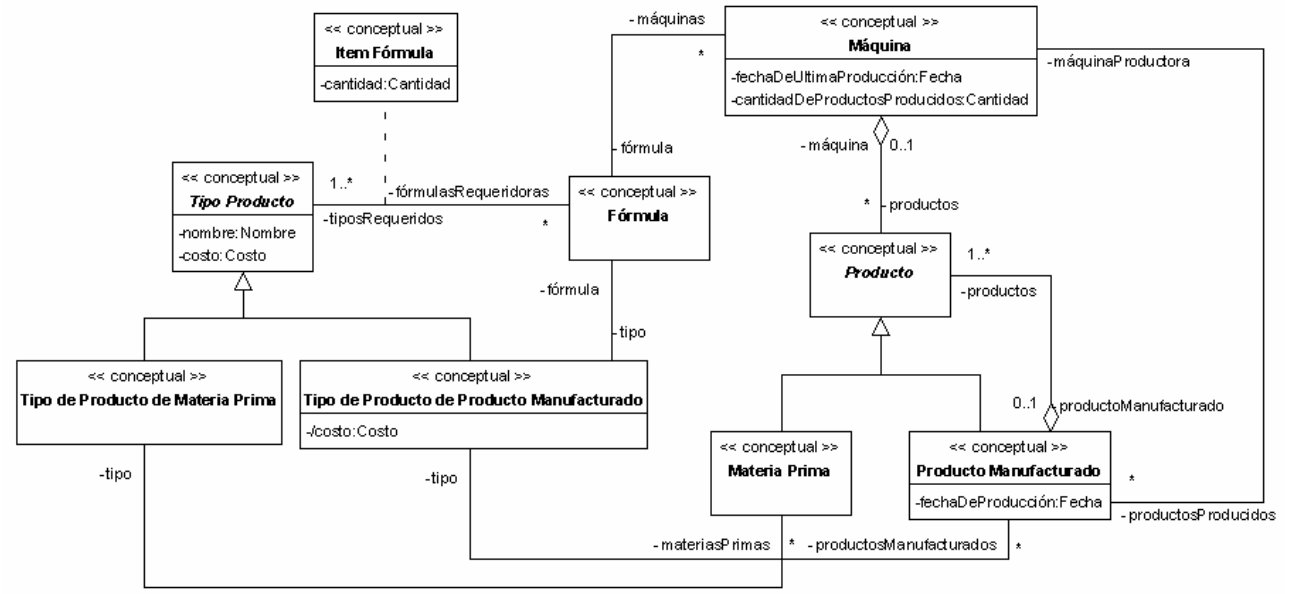


Figura 7 - Diagrama 2

Observamos que ahora Fórmula se relaciona con Tipo Producto De Producto Manufacturado y que cada subclase de Tipo Producto se relaciona con la subclase de Producto que corresponda. Esto nos evitará escribir restricciones en OCL.

Podremos tomar como modelos conceptuales válidos los mostrados en las figuras 6 y 7.

Para completar nuestro modelado escribiremos aquellas restricciones que no se desprenden del o los diagramas de clases conceptuales en lenguaje coloquial y en OCL. Proseguiremos trabajando sobre el diagrama mostrado en la figura 6 ya que es el más parecido al mostrado en clase.

Invariantes y otras restricciones

A continuación mostraremos algunas restricciones que deberán cumplirse. En algunos casos se mostrará más de una forma de escribir una misma restricción en OCL.

Item Fórmula

IF1: Para todo ítem de una fórmula, su cantidad debe ser mayor a 0.

context Item Fórmula
inv: self.cantidad > 0

Máquina

M1: Ninguna máquina puede tener más cantidad de productos de un mismo tipo que aquella que indica su fórmula.

context Máquina
inv: self.fórmula.ítem Fórmula -> forAll(if | self.productos -> select (p | p.tipo = if.tiposRequeridos) -> size() <= if.cantidad)

Nota 1: para navegar hacia la clase de asociación Item Fórmula se usó el nombre de la clase con su primera letra en minúscula.

Nota 2: en el invariante recién escrito if.tiposRequeridos es el tipo asociado al ítem fórmula if.

Lo mismo podría decirse con contexto en un ítem de una fórmula:

context Item Fórmula
inv: self.fórmulasRequeridoras.máquinas -> forAll(m | m.productos -> select (p | p.tipo = self.tiposRequeridos) -> size() <= self.cantidad)

Nota: en el invariante recién escrito self.fórmulasRequeridoras es la fórmula asociada al ítem fórmula self.

M2: Una máquina no tiene ningún producto cuyo tipo no esté en la fórmula de la máquina.

context Máquina
inv: self.productos -> forAll(p | self.fórmula.tiposRequeridos -> exist(tr | tr = p.tipo))

Este invariante también podría escribirse:

inv: self.productos -> forAll(p | self.fórmula.tiposRequeridos -> includes (p.tipo))

M3: Una máquina siempre produce productos manufacturados de un mismo tipo.

context Máquina
inv: self.productosProducidos -> forAll (pm1, pm2 | pm1.tipo = pm2.tipo)

Este invariante también podría escribirse:

inv: ProductosManufacturados.allInstances() - > forAll(pm1, pm2 | pm1.máquinaProductora = pm2.máquinaProductora implies pm1.tipo = pm2.tipo)

M5: La cantidad de productos manufacturados producidos por una máquina es igual a la cantidad de productos que esa máquina tiene registrada.

context Máquina
inv: self.cantidadDeProductosProducidos = self.productosProducidos -> size()

Este invariante también podría escribirse:

inv: self.cantidadDeProductosProducidos = Producto Manufacturado.allInstances() - > select (pm | pm.máquinaProductora = self) -> size()

Producto Manufacturado

PM1: *Para todo producto manufacturado su fecha de producción es menor o igual que la fecha de última producción de la máquina que lo produjo.*

context Producto Manufacturado
inv: self.fechaDeProducciónProducto <= self.máquinaProductora.fechaDeUltimaProducción

PM2: *Todo producto manufacturado debe cumplir con la fórmula de su tipo.*

Se entenderá que un producto manufacturado cumple la fórmula de su tipo si sólo tiene lo que ésta indica. Este invariante tendrá dos partes que llamaremos *tieneTodo* y *noHayTiposDeMás*:

tieneTodo: todo producto manufacturado tiene lo que la fórmula de su tipo indica.

noHayTiposDeMás: los tipos de los productos componente de un producto manufacturado están en la fórmula del tipo de éste.

Recordemos el ejemplo del enunciado de este ejercicio, si tuviéramos un iglú con tres carbones, dos de alpiste, un televisor y un tomate diríamos que iglú cumple con el invariante *tieneTodo* pero no con *noHayTiposDeMás*, no cumpliendo en consecuencia con la fórmula de su tipo con lo cual no podría existir tal iglú en nuestra fábrica.

context Producto Manufacturado
inv *tieneTodo*: self.tipo.fórmula.ítem Fórmula -> forAll (if | if.cantidad = self.productos -> select(p | p.tipo = if.tiposRequeridos) -> size())

Nota: en el invariante recién escrito if.tiposRequeridos es el tipo asociado al ítem fórmula if.

inv *noHayTiposDeMás*: self.productos.tipo -> asSet() -> size() = self.tipo.fórmula.tiposRequeridos -> size()

El último invariante podría escribirse de la siguiente forma también:

inv *noHayTiposDeMás*: self.productos -> forAll (p | self.tipo.fórmula.tiposRequeridos -> includes (p.tipo))

o bien:

inv *noHayTiposDeMás*: self.tipo.fórmula.tiposRequeridos -> includesAll (self.productos.tipo -> asSet())

PM3: *El costo de un producto manufacturado es la suma de los costos de sus componentes directos.*

context Producto Manufacturado::costo: Costo
derive**:** self.productos -> collect(costo) -> sum()

Notar que usamos ***derive*** en lugar de ***inv*** ya que el atributo costo es derivado.
Haciendo uso de la forma corta para collect el invariante se escribirá de la siguiente forma:

derive**:** self.productos.costo -> sum()

PM4: *Para todo producto manufacturado debe existir una fórmula para su tipo.*

context Producto Manufacturado
inv**:** self.tipo.fórmula -> notEmpty()

PM5: *Un producto manufacturado siempre conoce qué máquina lo fabricó y la fecha en que esto ocurrió.*

context Producto Manufacturado
inv :self.máquina productra -> notEmpty() and not self.fechaProducción.oclIsUndefined()

Nota: en el invariante recién mostrado se usó not self.fechaProducción.oclIsUndefined() para asegurar que el atributo fechaProducción de self esté definido.

Tipo Producto

TP1: *Todos los tipos producto tienen distinto nombre.*

context Tipo Producto
inv**:** Tipo Producto.allInstances() -> forAll(tp1, tp2 | tp1 <> tp2 implies tp1.nombre <> tp2.nombre)

A continuación se muestra un ejemplo de cómo escribir más de un invariante en un mismo contexto, para ello usaremos parte de los invariantes escritos en el contexto de Máquina:

context Máquina

-- M1: Ninguna máquina puede tener más cantidad de productos de un mismo tipo que aquella
-- que indica su fórmula.
inv: self.fórmula.ítem Fórmula -> forAll(if | self.productos -> select (p | p.tipo = if.tipo) -> size() <= if.cantidad)

-- M2: Una máquina no tiene ningún producto cuyo tipo no esté en la fórmula de la máquina.
inv: self.productos -> forAll(p | self.fórmula.tiposRequeridos -> includes (p.tipo))

-- M3: Una máquina siempre produce productos manufacturados de un mismo tipo.
inv: self.productosProducidos -> forAll (pm1, pm2 | pm1.máquinaProductora = pm2.máquinaProductora implies pm1.tipo = pm2.tipo)
...

De esta forma evitamos aclarar el contexto para cada invariante.

Ejercicios

Ya que están en contexto proponemos que escriban las siguientes restricciones en OCL:

Fórmula

F1: *Una fórmula es de un tipo producto de un producto manufacturado.*

F2: *Una fórmula no puede requerir un tipo producto que sea igual a su tipo.*

Máquina

M6: *Una máquina produce productos manufacturados con tipo igual al tipo de la fórmula de la máquina.*

Tipo Producto

TP2: *Si un tipo producto es tipo de una materia prima ent. no tiene fórmula asociada.*

Producto

P1: *Un producto está sólo en alguno de los siguientes lugares: el depósito, alguna máquina o algún producto manufacturado.*

P2: *Un producto manufacturado pm no puede contener en sus productos componente a pm ni otro producto que contenga a pm directa o indirectamente.*

Si tratamos de imaginar gráficamente este último invariante estamos pretendiendo que la estructura que se forma como consecuencia de las agregaciones sea un árbol orientado. Veamos un ejemplo, una casa tiene un dormitorio que contiene un ropero donde se encuentra un perfume; el perfume no podrá contenerse a sí mismo ni al ropero que lo contiene directamente ni al dormitorio o a la casa que lo contienen indirectamente.
