CONVERSIÓN DE DIAGRAMAS DE PROCESOS EN DIAGRAMAS DE CASOS DE USO USANDO AToM^3

CONVERSION OF PROCESSES DIAGRAMS IN USE CASE DIAGRAMS USING AToM^3

CARLOS M. ZAPATA J.
Grupo de Investigación UN-INFO. Escuela de Sistemas. Facultad de Minas. Universidad Nacional de Colombia.
cmzapata@unalmed.edu.co

CARLOS ALBERTO ÁLVAREZ.
Grupo de Investigación UN-INFO. Escuela de Sistemas. Facultad de Minas. Universidad Nacional de Colombia.

Recibido para revisar 26 de Abril de 2004, aceptado 19 de Mayo de 2004, versión final 20 de Mayo de 2004

RESUMEN: Toda pieza de software se origina en el modelo verbal, con el cual se pueden definir los diferentes modelos conceptuales que acerquen el problema a una solución. Las herramientas convencionales para la construcción de los modelos conceptuales no toman en consideración las diferentes reglas de consistencia que se pueden presentar entre los diferentes modelos. En este artículo se emplea el AToM^3 como herramienta para la definición de los meta-modelos del diagrama de procesos y el diagrama de casos de uso, con el fin de reexpresar el primero para obtener algunos elementos básicos del segundo.

PALABRAS CLAVE: Meta-modelos, AToM^3, Gramática de Grafos, consistencia entre modelos.

ABSTRACT: Every software piece has its origins in the verbal model. With this model, it's possible to define several conceptual models that allow reaching a solution closer to the problem. Conventional tools for conceptual model building don't take into account the different consistency rules between different models. In this paper we use AToM^3 as a tool for the meta-models' definition of process diagram and use case diagrams, trying to redefine the first to obtain some basic elements from the second.

KEYWORDS: Meta-models, AToM^3, Grammars Graphs, consistency between models.

1. INTRODUCCIÓN

La creación de una pieza de software implica una serie de etapas que permiten la simplificación sucesiva de los problemas del mundo real. En esta simplificación, los diferentes mecanismos de abstracción juegan un papel fundamental, puesto que son los que permiten extraer las características relevantes en cada una de las etapas para generar los diferentes modelos asociados con el problema y su posible solución.

Entre los diferentes tipos de modelos enfocados a la construcción de software, el UML (Unified Modeling Language) ha suministrado desde sus inicios una serie de modelos que constituyen un lenguaje para la representación de los problemas de la realidad con miras a su solución mediante piezas de software (OMG, 2004).

La construcción de los diferentes modelos se realiza empleando una sintaxis gráfica completamente definida para cada uno de los modelos, lo que facilita la visualización e interpretación de los diferentes diagramas por

parte de los analistas y usuarios familiarizados con la sintaxis de los mismos. Además, existen lenguajes especiales de programación que posibilitan la definición de cada uno de los modelos para la representación de la realidad; algunos de estos lenguajes como el OCL facilitar la definición de los diferentes modelos. Estos últimos, denominados Lenguajes Visuales Específicos del Dominio (Sprinkle y Karsai, 2004) permiten la definición de meta-modelos, que posibilitan la expresión de las características de los modelos que pueden posteriormente traducir las características de un problema del mundo.

En la literatura se pueden encontrar diferentes productos de meta-modelado, tales como el Generic Model Environment – GME (Ledeczi et al. 2001), el DOME (DOME, 2004) y el AToM^3 (De Lara y Vangheluwe, 2002) (ATOM3, 2004), entre otros. Estos productos se caracterizan por la definición de meta-modelos mediante un entorno gráfico que también se emplea para generar modelos con la sintaxis expresada en el meta-modelo, tal como se hace en las herramientas CASE convencionales.

Si bien las tres herramientas mencionadas permiten la elaboración de meta-modelos mediante una sintaxis de tipo gráfico, existen algunas diferencias apreciables que fundamentan la elección de AtoM3 como herramienta para la conversión entre modelos planteada en este artículo, a saber:

- DOME utiliza una forma ampliada del diagrama de clases adicionando algunos símbolos propios, que le entregan gran versatilidad para la definición de los meta-modelos, y se puede combinar con un lenguaje de programación similar al LISP llamado ALTER, especialmente para la definición de ciertas restricciones difíciles de explicar mediante la simbología gráfica. Cuando las restricciones son tan complejas que ni siquiera mediante el ALTER se puedan expresar, se puede recurrir al lenguaje en el cual está basado el DOME, que es el Smalltalk. Estas características hacen del DOME una herramienta muy adecuada para la realización de meta-modelos y chequeos de consistencia entre los mismos; sin embargo, cualquier transformación entre modelos debe ser programada en ALTER o incluso en Smalltalk, dependiendo del grado de dificultad, pues no cuenta con una herramienta de traducción que posibilite la transformación de un esquema en otro.
- GME igualmente utiliza una simbología similar al diagrama de clases para la definición de los meta-modelos, e incorpora la posibilidad de interpretación de restricciones utilizando OCL, que es el lenguaje de meta-modelamiento de UML, y en el cual se han ya definido algunas reglas de consistencia para la especificación de UML. Esto hace del GME una excelente herramienta para la elaboración de modelos basados en UML; sin embargo, el modelamiento conceptual posee diagramas que no pertenecen al UML y que son igualmente importantes en el análisis. Por ejemplo, en este artículo se pretende la construcción inicial de los casos de uso a partir del Diagrama de Procesos, que no pertenece a UML. Si bien es posible aún definir diagramas diferentes a UML en GME, la expresión de las reglas de consistencia debe ser programada completamente en OCL, pues GME tampoco cuenta con un mecanismo que facilite la elaboración de dichas reglas. La transformación entre modelos también se podría hacer, pero requiere un conocimiento muy a fondo del manejo de OCL.
- AtoM3, como se verá más detalladamente en la sección 2, basa la definición de sus meta-modelos en el modelo entidad – relación y, además de interpretar posibles restricciones escritas en OCL o Phyton, posee una gramática de grafos que permite definir fácilmente las reglas de transformación para dos modelos dados. Estas características hacen del AtoM3 una herramienta muy versátil para la transformación entre diferentes modelos, tales como los definidos para este artículo.

(Object Constraint Language) (Warmer y Kleppe, 2003) y el MAUDE (MAUDE, 2004) poseen expresiones sintácticas en forma de texto, mientras que otros lenguajes toman partido de ciertas características visuales para

- Queda fuera del alcance de este artículo la definición de cuál de las tres herramientas podría manejar de mejor manera las reglas de consistencia entre dos meta-modelos definidos.

Ahora, tomando como partida el modelo verbal del proceso, el grupo de desarrollo se enfrenta con la realización de los diferentes modelos que apoyan la definición, el análisis y el diseño de la pieza de software particular que se va a construir. Cada modelo que se incorpora al análisis contribuye con una vista parcial del mismo problema, surgida del modelo verbal, el cual se expresa a veces en forma ambigua e inexacta. La construcción de cada modelo, sin embargo, debe estar exenta de las ambigüedades propias del lenguaje natural y posibilitar una expresión formal del dominio de que trata el problema. Ese punto de partida comúnmente ocasiona la aparición de inconsistencias entre los modelos empleados en la definición y el análisis del problema, que se facilitan por la carencia de productos comerciales de modelamiento con mecanismos de control que impidan la introducción de inconsistencias entre los diferentes diagramas que hacen parte de un mismo proyecto. En la mayoría de esas herramientas de modelado es posible, por ejemplo, introducir en el diagrama de colaboración de un proyecto un conjunto de clases que no han sido definidas en el diagrama de clases del mismo proyecto y viceversa; estos productos tampoco controlan reglas sencillas de consistencia que validen, por ejemplo, la existencia de métodos asociados con una clase del modelo de clases para verificar los mensajes que viajan a través de los enlaces en el diagrama de colaboración.

Como una primera aproximación a la solución de este tipo de situaciones, el AToM^3 como herramienta de meta-modelado permite la introducción y utilización de gramáticas de grafos (De Lara et al., 2003), que permiten la evolución de los modelos mediante la reescritura de los mismos para cambiar su apariencia.

En este artículo se analiza específicamente la utilización de las gramáticas de grafos en el entorno AToM^3 para la generación de elementos de casos de uso a partir de la definición del Diagrama de Procesos de la organización.

El artículo está organizado así: en la sección 2 se muestran las principales características del AToM^3 y de las gramáticas de grafos que se pueden emplear para la construcción y reexpresión de modelos y meta-modelos; en la sección 3 se introduce la notación de los Diagramas de procesos y de los Casos de Uso; en la sección 4 se muestra la definición de estos modelos en el entorno AToM^3, al igual que las reglas definidas mediante gramáticas de grafos para la generación de algunos elementos del diagrama de casos de uso a partir del diagrama de procesos; en la sección 5 se muestra un caso de aplicación; en la sección 6 se discuten los trabajos futuros que se pueden generar; finalmente, en la sección 7 se muestran las conclusiones del trabajo.

2. AToM^3 Y LAS GRAMÁTICAS DE GRAFOS

AToM^3.(A Tool for Multi-Formalism Modelling and Meta-Modelling) es una herramienta escrita en el lenguaje de programación Phyton. Esta herramienta posee un procesador que incluye un meta-meta-modelo inicial basado en el modelo entidad relación extendido con restricciones, que permite la definición de los diferentes meta-modelos en un entorno gráfico con las mismas características que emplea el usuario en la construcción de los diferentes modelos. De esta manera, se puede definir cualquier tipo de meta-modelo en términos de las "entidades" que hacen parte del mismo y sus posibles interconexiones o "relaciones". Una vez definido el meta-modelo, se puede emplear su definición para construir los modelos pertinentes a un problema específico del mundo (De Lara y Vangheluwe, 2002).

Además del formalismo descrito, AToM^3 tiene la posibilidad de expresar ciertas restricciones en términos de Gramáticas de Grafos, que están incorporadas en su entorno. Las Gramáticas de Grafos tienen similitudes con las gramáticas basadas en texto en el sentido de que pueden ser usadas para describir las transformaciones a un grafo determinado; la diferencia radica en que las reglas que hacen parte de este tipo de gramáticas se expresan de manera gráfica y no

a modo de texto. Las gramáticas de grafos se definen como un conjunto de reglas que poseen un lado izquierdo (left-hand side o LHS) que contiene las precondiciones (expresadas de forma gráfica) que deben ser cumplidas para activar una determinada regla y un lado derecho (right-hand side o RHS) que contiene el grafo que reemplazará el que equipare el lado izquierdo de la regla. Para las reglas expresadas de esta manera también se deben definir unas condiciones y unas acciones para ejecutar cuando la regla se active. La Gramática de Grafos de AtoM^3 posee también un mecanismo que va rescribiendo el modelo a medida que las diferentes reglas se van activando hasta que no haya reglas que se puedan ejecutar (De Lara et al., 2003).

3. DIAGRAMAS DE PROCESOS Y DIAGRAMAS DE CASOS DE USO

El modelo verbal de un problema suministra los elementos necesarios para trazar el diagrama de procesos de la organización, que se constituye en una descripción de las secuencias de actividades que describen el quehacer de la organización. Este diagrama ha sido descrito en los textos clásicos de administración de negocios, tales como (Harrington, 1991) y se ha retomado su utilización en herramientas actuales como el Oracle® Designer (Anderson y Wendelken, 1996). En el diagrama se incluyen varios elementos relevantes al problema, tales como:

- Los *actores* (denominados también *Unidades Organizacionales* por su notación en el Oracle® Designer), que son responsables de cada uno de los procesos que ocurren en la organización.
- Los *procesos*, que son secuencias de pasos que se ejecutan en la organización.
- Los *eventos*, que son los "detonadores" que inician una determinada secuencia de procesos.
- Los *almacenamientos*, que son los sitios donde se guarda la información generada.
- Los *condicionales*, que permiten la bifurcación de los procesos en diferentes caminos dependiendo del cumplimiento de una condición especificada.
- Los *finales de procesos*, que marcan sitios o momentos especiales donde mueren los procesos.
- Los *flujos*, que son los conectores que se presentan entre los diferentes elementos del diagrama y que representan físicamente el paso de algún tipo de información entre los elementos que unen.

Las responsabilidades de los procesos definidos se asocian con cada uno de los actores o unidades organizacionales mediante unos carriles o rectángulos horizontales que representan el área de influencia o responsabilidad del actor. Por simplicidad, y para efectos de este artículo, se sustituyó el carril correspondiente al actor por un dibujo del actor mismo. La simbología básica del diagrama de procesos se puede apreciar en la Figura 1.

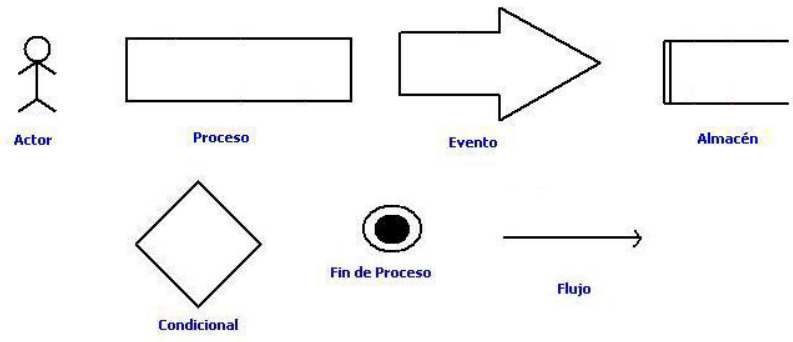

Figura 1. Simbología básica del Diagrama de procesos.
Figure 1. Basic simbology of process diagrams.

En AToM^3 la construcción de los meta-modelos se realiza mediante el formalismo correspondiente al diagrama entidad-relación, en el cual las entidades se representan como rectángulos y los rombos representan las relaciones. La construcción del meta-modelo del diagrama de procesos en AToM^3 se puede visualizar en la Figura 2. Para cada una de las entidades se puede definir la imagen que la representa dentro de las propiedades de la entidad; la representación del carril correspondiente a la zona de influencia de un actor o unidad organizacional, en el meta-modelo se coloca al interior de las entidades "Eventos" y "Procesos" mediante una propiedad llamada "Actores", en la cual se pueden listar los actores o unidades organizacionales que tienen responsabilidad sobre el proceso.

Ahora, para analizar cómo será la interacción entre los diferentes actores que van a hacer parte de la solución al problema propuesto y el sistema que servirá como solución, en (Jacobson et al., 1992) se propuso el modelo de "Casos de Uso"; los elementos fundamentales de este modelo son:

- El *actor*, que es el responsable de la iniciación o culminación de un caso de uso.
- Las *funciones o procesos* que se pueden ejecutar dentro del caso de uso.
- Las *asociaciones*, que representan intercambios de información entre los actores y las funciones.

Por simplicidad de uso de este modelo, se han obviado otros elementos que hacen parte de la descripción de un caso de uso como son las asociaciones tipo <<extend>> y las asociaciones tipo <<include>>. La simbología básica del diagrama de casos de uso se puede apreciar en la Figura 3.

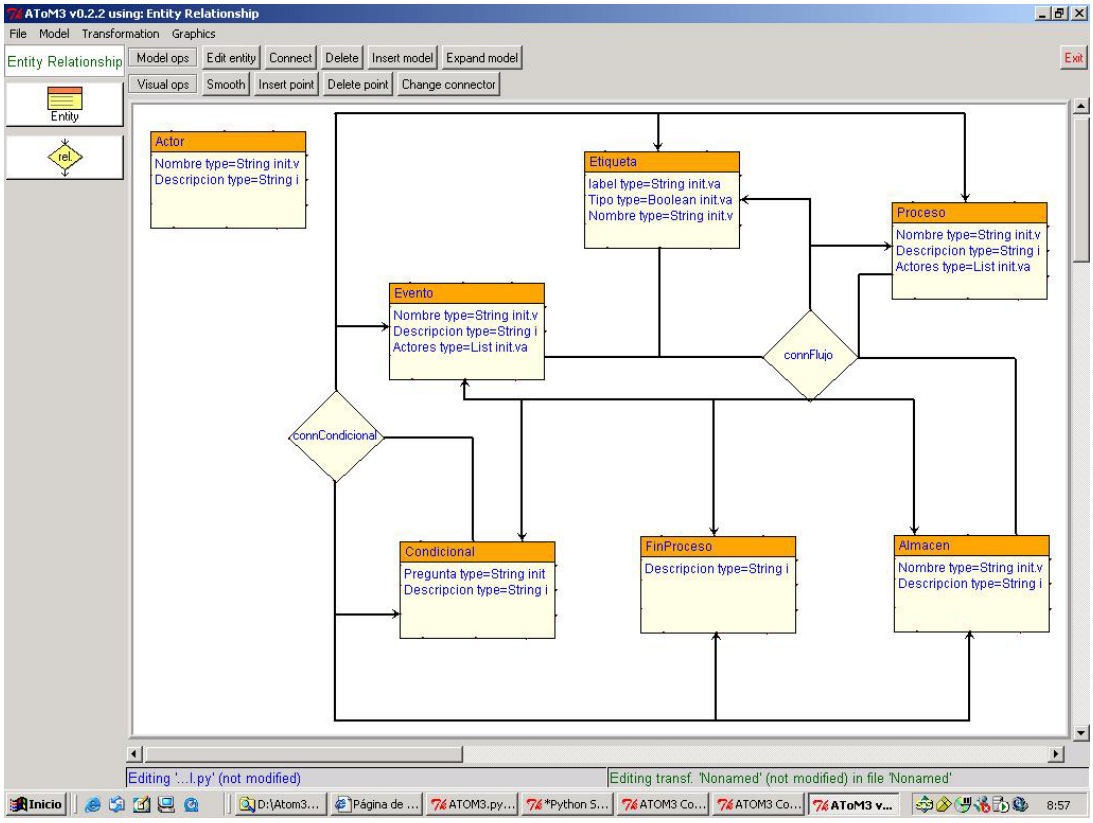

Figura 2. Meta-modelo en AToM^3 para el Diagrama de procesos.
Figure 2. Meta-model in AToM^3 for process diagrams.

Figura 3. Simbología Básica del Diagrama de Casos de Uso.
Figure 3. Basic simbology of use case diagrams.

La construcción del meta-modelo del Diagrama de casos de uso en AToM^3, con las entidades definidas "actor" y "función o proceso" y los flujos como las conexiones, se puede apreciar en la Figura 4.

4. CONVERSIÓN DE DIAGRAMAS DE PROCESOS EN CASOS DE USO

Existen ciertas similitudes que pueden ser explotadas entre los diagramas de procesos y los de casos de uso:

- Los actores están presentes en ambos modelos.
- Los procesos del diagrama de procesos se pueden asimilar a funciones o procesos del caso de uso.
- Los procesos pertenecen al carril de responsabilidad de un actor en el diagrama de procesos. Además, en los casos de uso los actores son los que inician funciones o procesos.

Estas similitudes posibilitan la definición de ciertos elementos a partir de los diagramas de procesos que pueden facilitar la construcción de los casos de uso. Por ejemplo, utilizando la primera similitud se puede concluir que un actor o unidad organizacional en el diagrama de procesos se podrá convertir directamente en un actor del diagrama de casos de uso. De igual manera, se pueden emplear las demás similitudes para generar diferentes reglas en gramática de grafos para la conversión del diagrama de procesos en el punto de partida del diagrama de casos de uso. Se habla de "punto de partida" porque no todos los actores que se encuentran definidos en el Diagrama de Procesos se involucrarán de manera definitiva en los Diagramas de Casos de Uso; sin embargo, deberá estar disponible en la "versión inicial" del diagrama de casos de uso generado a partir del diagrama de procesos.
Diagrama de Casos de Uso se definieron nueve reglas, cuyo listado se puede apreciar en la Figura 5.
El número que está ubicado a la derecha de cada regla corresponde a la prioridad de ejecución de la misma. Las reglas que inician con "delete" realizan la eliminación de las entidades que no se van a utilizar en el diagrama de casos de uso, tales como eventos, almacenes, condicionales, finales de procesos, etiquetas, flujos y flujos de condicionales. Para estas reglas, el lado izquierdo de la regla (LHS) posee la imagen del elemento correspondiente, mientras que el lado derecho (RHS) se encuentra vacío (esto le indica al entorno que el elemento del LHS no aparecerá en el modelo resultante) porque lo que se busca es que estos elementos no estén presentes en el diagrama de casos de uso que se está generando.

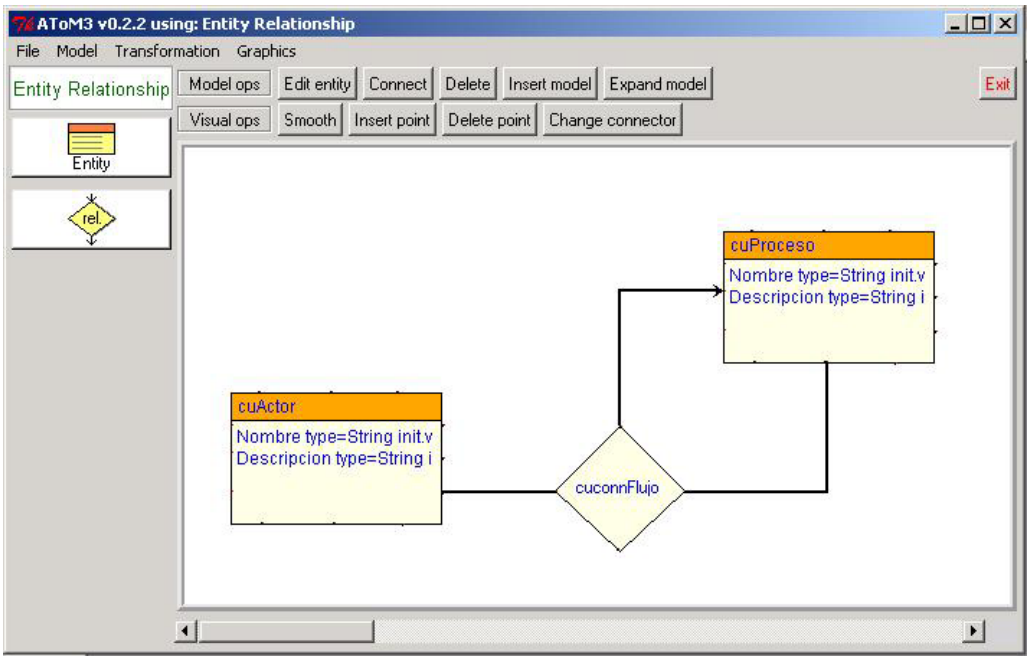

Figura 4. Meta-modelo en AToM^3 para el Diagrama de Casos de Uso.
Figure 4. Meta-model in AToM^3 for use case diagrams.

Para la conversión del Diagrama de Procesos en Las reglas que comienzan con "change", asociadas con una prioridad de 8 y 9 respectivamente, realizan inicialmente el borrado de los elementos actores y procesos, para luego realizar su construcción en los equivalentes a actores y funciones (o procesos) del diagrama de casos de uso. Para estas reglas, el RHS también está vacío, ya que inicialmente esos elementos deben ser borrados del diagrama resultante; su creación posterior se realiza mediante un código Phyton que se coloca en la parte de "acciones" asociadas con la regla.

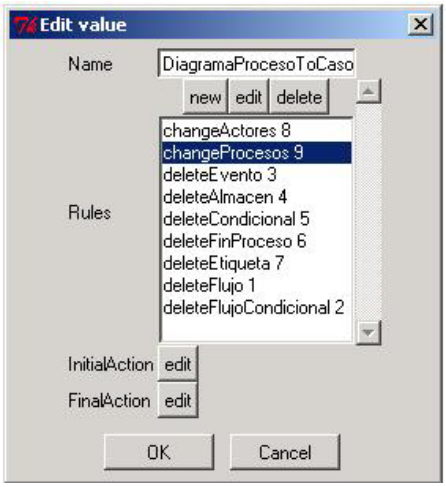

Figura 5. Listado en AToM^3 de Reglas para la conversión de Diagramas de Procesos en Diagrama de Casos de Uso.

Figure 5. List in AToM^3 of rules to change process diagrams in use case diagrams.

Al seleccionar la regla “ChangeActores” se activa la pantalla que se muestra en la Figura 6, en la cual se muestran las propiedades que hacen parte de una regla de transformación.

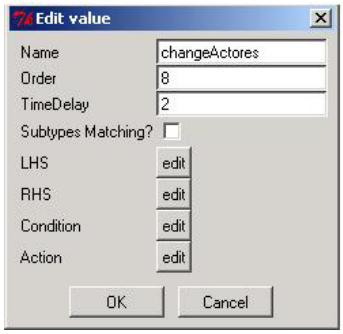

Figura 6. Propiedades de la regla “ChangeActores”.
Figure 6. Properties of the rule “ChangeActores”.

Al seleccionar LHS en la Figura 6 se puede apreciar la sintaxis correspondiente al lado izquierdo de la regla (ver Figura 7), que incluye la palabra reservada <ANY> para significar que cualquier entidad del tipo definido (con cualquier valor en la propiedad Nombre) que se encuentre en el modelo activará la regla.

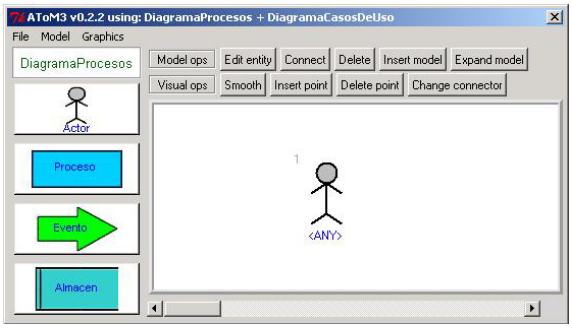

Figura 7. LHS de la regla “ChangeActores”
Figure 7. LHS of the rule “ChangeActores”.

El LHS de la regla “ChangeProcesos” es similar a la Figura 7 pero en lugar de la imagen del actor tiene la imagen del proceso.
Para que el actor aparezca en el diagrama de Casos de Uso resultante, es necesario definir una acción que cree ese actor con características similares al que le dio origen en el diagrama de procesos, pero con el tipo y características del diagrama de casos de uso. El código Phyton correspondiente a esta acción es el siguiente:

```
from DiagramaCasosDeUso_MM import
createNewcuActor
n1 = self.getMatched( graphID,
self.LHS.nodeWithLabel(1))
n2 = createNewcuActor(atom3i,
n1.graphObject_.x, n1.graphObject_.y, 1)
n2.Nombre = n1.Nombre
n2.Descripcion =n1.Descripcion
n2.graphObject_.ModifyAttribute("Nombre",
n2.Nombre.toString())
```

De manera similar, para la generación de las funciones, en la regla “ChangeProcesos” se debe adicionar el siguiente código en las acciones:

```
from DiagramaCasosDeUso_MM import
createNewcuProceso, createNewcuconnFlujo
n1 = self.getMatched( graphID,
self.LHS.nodeWithLabel(1))
n2 = createNewcuProceso(atom3i,
n1.graphObject_.x, n1.graphObject_.y, 1)
n2.Nombre = n1.Nombre
n2.Descripcion =n1.Descripcion
n2.graphObject_.ModifyAttribute("Nombre",
n2.Nombre.toString())
for name in n1.Actores.getValue():
    for sem_objTo in
atom3i.ASGroot.listNodes["cuActor"]:
        if sem_objTo.Nombre.toString().upper()
== name.toString().upper():
            dx = sem_objTo.graphObject_.x -
n2.graphObject_.x
            dy = sem_objTo.graphObject_.y -
n2.graphObject_.y
            x = n2.graphObject_.x + dx/2
            y = n2.graphObject_.y + dy/2
            flujo =
createNewcuconnFlujo(atom3i, x, y, 1)
            atom3i.drawConnections((sem_objTo,
flujo), (flujo, n2))
            break
```

Con la aplicación de estas reglas no se obtienen todos los elementos que hacen parte del diagrama de casos de uso, pero, por lo menos, permiten tener un punto de partida que facilite la realización del diagrama de casos de uso y que, adicionalmente, genere elementos que son totalmente consistentes con aquellos definidos

en el diagrama de procesos, dado que se originaron a partir del mismo.

5. APLICACIÓN DE LA CONVERSIÓN A UN CASO DE EJEMPLO

En la Figura 8 se presenta una parte del modelo de procesos de una empresa de Consultoría para la definición de encuestas de satisfacción. En este modelo se presentan dos actores (la secretaria y el cliente) y los eventos, procesos, flujos y almacenamientos asociados con sus funciones en relación con la organización.

Al aplicar las reglas de transformación definidas para la generación del diagrama de casos de uso a partir de este modelo se obtiene la imagen de la Figura 9, la cual se constituye en un punto de partida para la elaboración del diagrama de casos de uso completo para la organización.

Como se puede apreciar en la Figura 2, la entidad "Proceso" del meta-modelo del Diagrama de procesos tiene asociada una propiedad que es "Actores" en la cual se consigna una lista de los nombres de las instancias de la entidad "Actor" que están asociados con ese proceso. Esto se realizó debido a que no se pudieron generar con el formalismo de AToM3 los "carriles" descritos para el diagrama de procesos. Con esta propiedad, una vez se han creado tanto los actores como las funciones en el diagrama de casos de uso, la aplicación de las regla "ChangeProcesos" genera las conexiones entre actores y funciones.

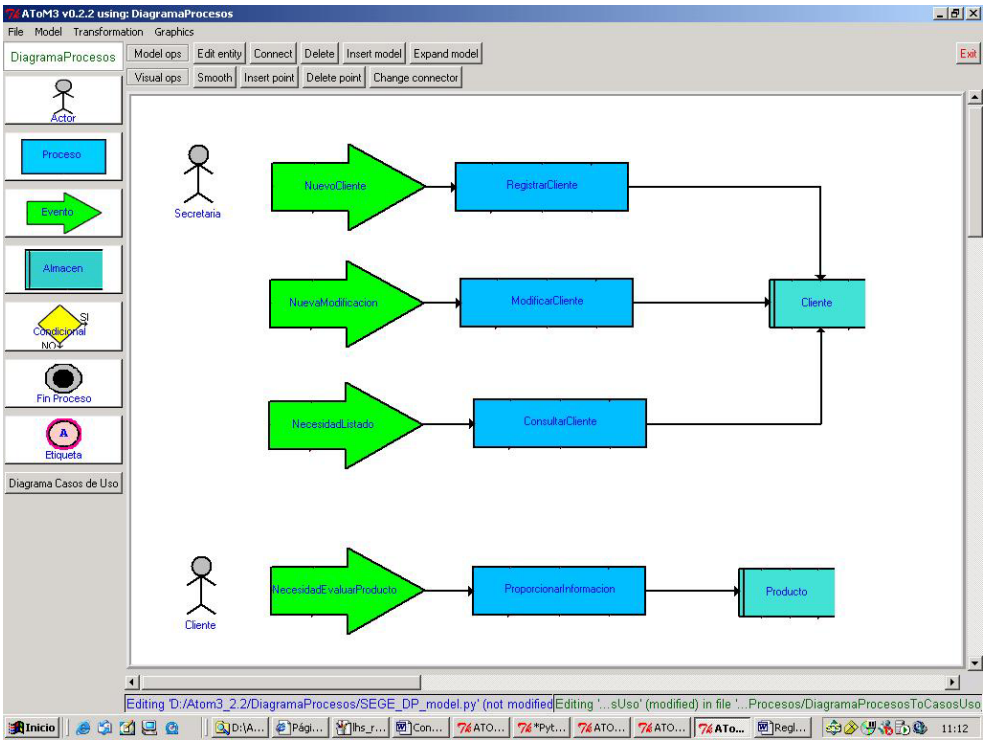

Figura 8. Modelo del Diagrama de Procesos para una Empresa de Consultoría de Definición de Encuestas de Satisfacción (Parcial).
Figure 8. Process diagram model for a consultancy company of definition of satisfaction surveys (Partial).

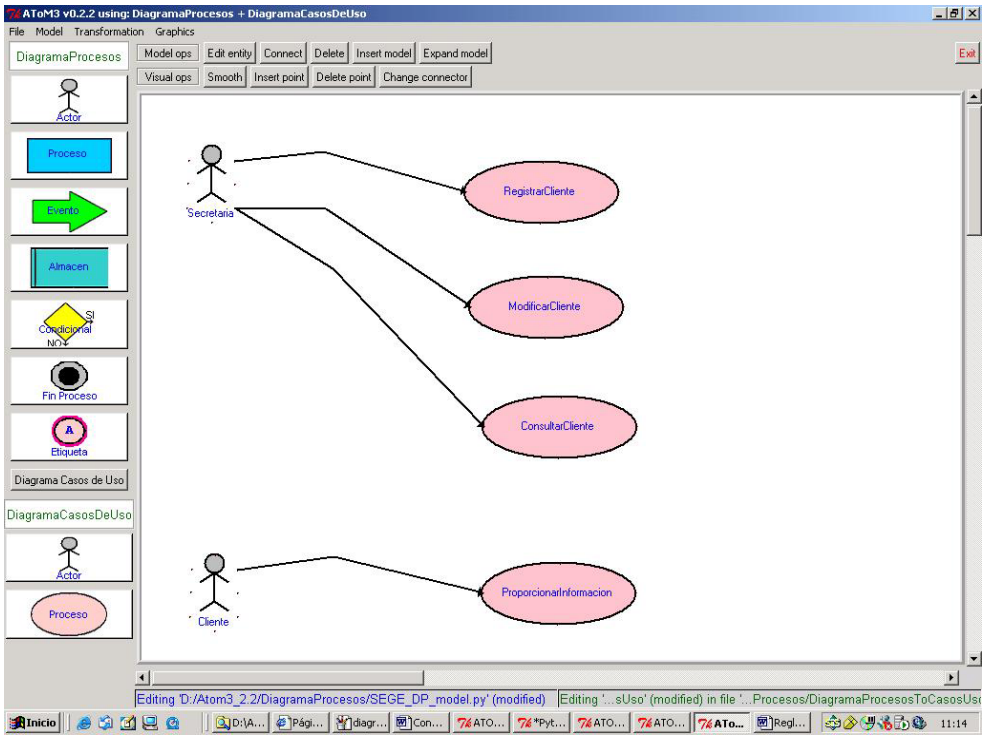

Figura 9. Modelo del Diagrama de Casos de Uso generado a partir del diagrama de procesos de la Figura 8.
Figure 9. Use case diagram model generated from figure 8 process diagram.

6. TRABAJOS FUTUROS

- En este artículo sólo se definieron algunas de las reglas que permiten la conversión de los diagramas de procesos en diagramas de casos de uso. Se podrían definir posteriormente otras reglas que puedan asociarse con la consistencia entre los modelos planteados.
- Cuando se realiza el modelamiento conceptual se utilizan otros modelos que podrían generar información complementaria. Por ejemplo, los diagramas de clases y de los diagramas de colaboración presentan similitudes que son evidentes y que también podrían manejarse como reglas para la transformación de los primeros en los segundos.
- No se maneja plenamente la consistencia con lo que se definió en este artículo, ni siquiera entre los dos modelos que sirvieron de base para este trabajo. Aquí se empleó el diagrama de procesos para generar parte del diagrama de casos de uso y no a la inversa. Ello implica que, si al terminar la construcción del caso de uso generado con la transformación definida se presentan cambios radicales en el diagrama, esos cambios no se podrán reflejar mediante conversiones en el diagrama de procesos que le dio origen. Para el manejo de la consistencia se podrían emplear los formalismos de los dos meta-modelos y verificar, mediante algunas reglas definidas en gramáticas de grafos, la equivalencia entre los elementos de uno y del otro que deban ser iguales.

7. CONCLUSIONES

- En AToM^3 las gramáticas de grafos posibilitan la expresión de reglas de conversión que se pueden utilizar entre diferentes tipos de modelos conceptuales, dependiendo de sus similitudes.
- Un punto de partida para el manejo de la consistencia entre modelos pertenecientes

al mismo problema consiste en la reexpresión de las características de un modelo en las características de otro. En el caso de este artículo, las similitudes entre el diagrama de procesos y el de casos de uso posibilitaron la conversión del primero en la primera versión del segundo, con los consecuentes beneficios para el proyecto como tal.

REFERENCIAS

[1] ANDERSON, C., y WENDELKEN, D. *The Oracle® Designer/2000 Handbook*. Addison-Wesley. 1996

[2] ATOM3. MSDL – ATOM3. Página Web de la herramienta ATOM3. Available: http://atom3.cs.mcgill.ca/. [Citado 22 de Marzo de 2004]

[3] DE LARA, J., y VANGHELUWE, H.. *AToM^3: A tool for Multi-Formalism and Meta-Modelling*. Proceedings of the Fifth Ingernational Conference on Fundamental Approaches to Software Engineering. Pp. 174-188. 2002.

[4] DE LARA, J., VANGHELUWE, H y ALFONSECA, M. *Using Meta-Modelling and Graph-Grammars to Create Modelling Environments.* Electronic Notes in Theoretical Computer Science, Vol. 72 No. 3. 2003.

[5] DOME. What is Dome. Available: http://www.htc.honeywell.com/dome/description.htm. [Citado 22 de Marzo de 2004].

[6] HARRINGTON, H. J. *Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness*. McGraw-Hill. 1991.

[7] JACOBSON, I. et al. *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley. 1992.

[8] LEDECZI A., MAROTI M., BAKAY A., KARSAI G., GARRETT J., THOMASON IV C., NORDSTROM G., SPRINKLE J. y VOLGYESI P. *The Generic Modeling Environment*. Proceedings of the Workshop on Intelligent Signal Processing, Budapest. 2001.

[9] MAUDE. The Maude System. Stanford Research Institute. Available: http://maude.cs.uiuc.edu/. [Citado 22 de Marzo de 2004]

[10] OMG. OMG Unified Modeling Language Specification. Object Management Group. Available: http://www.omg.org/UML/. [Citado 22 de Marzo de 2004]

[11] SPRINKLE J. Y KARSAI G. *A Domain-Specific Visual Language For Domain Model Evolution.* Journal of Visual Languages and Computing, vol. 15, no. 2. 2004.

[12] WARMER, J., KLEPPE, A. *The Object Constraint language: Getting your models ready for MDA.* Addison – Wesley. 2003.