Lenguaje de Especificación para el Framework Tropos

Alicia Martínez	Carmen Velasco	Eliel Morales	Hugo Estrada	Luis A. Gama
CENIDET, México	I.T.Z. México	CENIDET, México	CENIDET, México	I.T.Z. México
amartinez@cenidet.edu.mx	k.rmen.v.v@gmail.com	eliel@cenidet.edu.mx	hestrada@cenidet.edu.mx	lgama_moreno@hotmail.com

Resumen

El framework i surgió como una propuesta para realizar reingeniería de procesos de negocios hace ya casi dos décadas. A partir de entonces han surgido numerosas propuestas y herramientas que han añadido nuevos conceptos y funcionalidades a esta propuesta original. Una de ellas es el framework Tropos, que es una metodología orientada a agentes enfocada al desarrollo de software. Sin embargo, uno de los grandes problemas que se presenta en el uso de esta metodología es que cada propuesta que usa o extiende Tropos o i* utiliza los conceptos dependiendo sus necesidades, y emplean diferentes formatos en el almacenamiento de sus modelos. En este artículo se presenta un lenguaje de especificación para el framework Tropos basado en* XML*, el cual define la estructura de sus diagramas e impone reglas para la utilización de sus primitivas de modelado. Se presenta un DTD (Document Type Definition) que representa la gramática de este lenguaje y que permite hacer la validación de la estructura de los modelos definidos en* XML*. El enfoque propuesto intenta homogeneizar el uso de los modelos organizacionales entre la comunidad que utiliza esta metodología.*

1. Introducción

Hoy en día, el framework *i** [18] ha originado la creación de nuevas propuestas de investigación en torno al modelado organizacional. Una de estas propuestas ha sido la metodología Tropos [3] que ha adoptado los conceptos de *i** y los ha enfocado al desarrollo de software. Existen además otras propuestas que han introducido cambios o extensiones al lenguaje con el objetivo de realizar investigación y desarrollo en las siguientes áreas: seguridad [11] [15], problemas de confianza [7], operadores temporales [6], y construcción de trazabilidad [9], entre muchas otras. En gran parte de estas propuestas de investigación se han desarrollado herramientas propias, que utilizan su propia concepción de los diagramas y primitivas, lo que ha generado incompatibilidad entre ellas; consideramos que el gran potencial que tiene cada una de estas herramientas crecería ampliamente si pudiera contarse con una herramienta base, la cual sea un núcleo donde puedan añadirse nuevos módulos con diferentes funcionalidades. Por otra parte, debido a que la notación y diagramas de Tropos no han sido suficientemente acotados se tienen diversos problemas a nivel de repetitividad. En este contexto, en [5] se realizó una evaluación empírica del framework *i** en un ambiente de generación de software, donde se analizó la construcción de varios modelos. Uno de los problemas encontrados en este análisis fue la falta de repetitividad en la construcción de modelos realizados por diferentes usuarios. Se determinó que, tomando como base una misma semántica a representar, diferentes usuarios utilizaron diferentes diagramas y primitivas para representar la solución, lo cual es un indicativo de la ambigüedad en las primitivas de Tropos.

De esta manera, teniendo como antecedente este estudio y la necesidad de crear una herramienta para la generación de sistemas de información a partir de modelos organizacionales [13] proponemos un lenguaje de especificación y presentamos un DTD (*Document Type Definition*) que representa la gramática de este lenguaje. El DTD ha sido diseñado con base en las reglas para el modelado organizacional del framework Tropos. Consideramos que el uso de esta especificación permitirá una utilización más repetible y consistente de los conceptos de Tropos y de las primitivas que esta

metodología posee. Se busca también poder especificar estos conceptos en un formato que sea independiente de la plataforma de implementación.

Es necesario comentar que se han colocado algunas restricciones en la especificación del lenguaje, con el objetivo de facilitar su uso entre personas que no cuentan con experiencia en la construcción de modelos organizacionales Tropos.

Una propuesta encaminada a esta solución es presentada en [4], donde los autores proponen una solución a la interoperabilidad que existe entre las diferentes herramientas a través de la propuesta de un conjunto de etiquetas especificadas en XML (*eXtensible Markup Language*). En este artículo presentamos una solución alternativa al presentado en [4], ya que el artículo presenta un lenguaje de especificación para el framework Tropos basado en XML, el cual define la estructura de sus diagramas e impone reglas para la utilización de sus primitivas de modelado. La principal diferencia entre estos dos enfoques radica en el nivel de detalle de las etiquetas XML y en la definición de las reglas para utilizar las primitivas y crear los diagramas que son propuestas en nuestro trabajo. Consideramos que la especificación propuesta en este artículo permite un mayor control en la construcción de los diagramas, una semántica clara y una estructura bien definida para cada uno de sus elementos de modelado.

Este artículo se encuentra organizado de la siguiente manera: la sección 2 presenta los conceptos básicos del framework Tropos, en la sección 3 se detallan guías y restricciones para la construcción de los diagramas de Tropos, la sección 4 define la representación del modelo organizacional utilizando XML, en la sección 5 se especifica la gramática utilizada para diseñar el DTD. La sección 6 describe un caso de estudio. Finalmente, la sección 7 presenta las conclusiones y trabajos futuros.

2. El Framework Tropos

El framework Tropos está formado por dos diagramas que nos permiten modelar el ambiente organizacional con un enfoque basado en metas y dependencias [16] [12]:

Diagrama de actores. Es una representación gráfica donde se muestran los actores y sus metas, y las dependencias entre los actores.

Diagrama de metas. Es una representación gráfica donde se analizan con mayor detalle las metas, planes y dependencias de cada actor.

A continuación se presentan los conceptos básicos utilizados en este framework.

- **Actor.** Es una entidad que tiene metas estratégicas e intenciones dentro del sistema o dentro del conjunto organizacional. La representación gráfica de un actor es un círculo de líneas punteadas.
- **Hardgoal/Softgoal.** Estas metas representan los intereses estratégicos de un actor. Las *hardgoals* se distinguen de las *softgoals* porque en las segundas no existe un criterio claro para definir si ellas han sido satisfechas. Las *softgoals* son dibujadas como una nube, mientras que las *hardgoals* se muestran como un rectángulo con las puntas redondeadas.
- **Plan.** Representa una forma de hacer algo en un nivel abstracto. La ejecución del plan puede ser una manera de satisfacer una *hardgoal* o una *softgoal*. Los planes son dibujados como hexágonos.
- **Recursos.** Representa una entidad física o de información. Son dibujados como un rectángulo. La Figura 1 ilustra la notación gráfica de estos conceptos de modelado.

Figura 1. Notación gráfica de los elementos básicos de Tropos

- **Dependencias.** Es una relación intencional y estratégica entre dos actores. Este tipo de relación indica que un actor depende de otro actor con el objeto de alcanzar una meta, ejecutar un plan u obtener un recurso. El primer actor es llamado *depender*, mientras que el actor del cual se depende se denomina *dependee*. El objeto alrededor del cual se centra la dependencia se denomina *dependum*. Existen 4 tipos de dependencias:
  - **Dependencia de Hardgoal.** Este tipo de dependencia es una relación en la cual un actor depende de otro para satisfacer una meta, sin prescribir la manera en la cual debe ser llevada a cabo.
  - **Dependencia de Softgoal.** Este tipo de dependencia es muy similar a la dependencia de *hardgoal*, con la diferencia que no es posible definir, en forma precisa, la forma en la cual se satisface la meta suave.
  - **Dependencia de Recurso**. Este tipo de dependencia es una relación en la cual un actor depende de otro actor para entregar un

recurso que puede ser material o informacional.

- **Dependencia de Plan.** Es una relación en la cual existe una dependencia para llevar a cabo una tarea. En este tipo de dependencia se prescribe la manera en la cual debe ser ejecutado el plan.

- **Contribución.** Es una relación entre metas o planes y representa la forma en que ciertas metas o planes pueden contribuir (positiva o negativamente) en el cumplimiento de otra meta.

La representación gráfica de las dependencias se ilustra en la Figura 2.

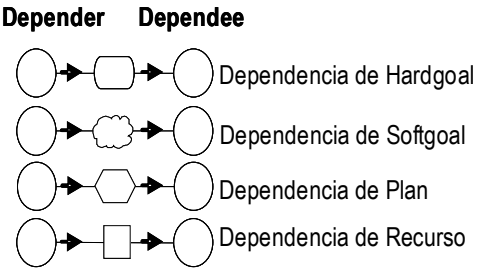

Figura 2. Notación gráfica de las relaciones de dependencia

3. Guías y restricciones para la construcción de los diagramas de Tropos

La definición del lenguaje de especificación presentado en este trabajo nos permitió establecer las posibles formas y combinaciones para utilizar las primitivas de modelado en los diagramas de Tropos.

A continuación se detallan los pasos propuestos para la construcción de cada uno de los diagramas. También se listan las restricciones que surgieron al realizar las reglas de producción de nuestra especificación. Posiblemente, algunas de estas restricciones podrían parecer obvias para el analista experto en Tropos, pero consideramos necesario proponer estas restricciones como un conjunto de requerimientos mínimos que debe contar un modelo organizacional. Además, esto permite tener una semántica mejor definida para cada uno de los elementos de Tropos. Estas restricciones solucionan algunos de los problemas encontrados en [4], donde la notación definida permite que se coloquen en su diagrama (este enfoque sólo considera un tipo de diagrama) metas o tareas sin estar relacionadas a un actor, o colocar un actor sin ningún elemento de modelado, lo cual puede inducir a errores de modelado en usuario inexpertos.

3.1. Diagrama de actores

El primer paso para la construcción del diagrama de actores es la identificación de todos los actores que intervienen en la organización. El segundo paso consiste en insertar las metas de cada actor en el diagrama. El tercer paso consiste en insertar las dependencias entre los actores identificados. Las restricciones que se han colocado en la gramática propuesta para la construcción de los diagramas de Tropos, son las siguientes:

Restricción 1: el diagrama de actores debe contener al menos dos actores organizacionales y una o más dependencias entre ellos.

Restricción 2: Cada actor organizacional identificado debe tener al menos una *hardgoal* asociada a él, lo que indica que todos los actores dentro de una organización tienen al menos una meta que intentan cumplir.

Restricción 3: Una dependencia debe estar formada por tres elementos: el *depender*, *dependee* y el *dependum*. Los primeros dos elementos corresponden a algunos de los actores identificados, mientras que el *dependum* puede ser: una *hardgoal*, una *softgoal*, un recurso o un plan.

Restricción 4: Un actor puede contener atributos que lo describan (este elemento es opcional). Esta condición no es necesaria en la descripción gráfica de una organización, sin embargo cuando los modelos organizacionales se utilizan dentro de un ambiente de modelado para el desarrollo de software, estos atributos se hacen indispensables para la obtención de información. De esta manera, los atributos se han colocado como un elemento opcional en esta gramática, porque su implementación dependerá del uso de los modelos organizacionales.

3.2. Diagrama de metas

El primer paso para la construcción del diagrama de metas es colocar los actores identificados en el diagrama de actores. El segundo paso consiste en insertar las metas de cada actor y refinar cada meta en submetas o planes, utilizando las diferentes ligas del framework Tropos. El refinamiento concluye una vez que todas las metas principales se han descompuesto en al menos un plan y este plan a su vez ya no puede ser descompuesto en otros subplanes. El tercer paso consiste en colocar las dependencias identificadas en el diagrama de actores. Las restricciones que se han colocado en este diagrama, son las siguientes:

Restricción 5: El diagrama de metas debe contener los mismos elementos del diagrama de actores. Esto se debe a que este diagrama es una extensión del diagrama de actores.
Restricción 6: Las *hardgoals* deben ser refinadas en otros elementos y uno de ellos debe ser un plan. Esto con el objetivo de operacionalizar las *hardgoals* dentro de un ambiente de desarrollo de software. Por su parte, las *softgoals* podrían no refinarse.
Restricción 7: Cada plan definido en este diagrama puede ser refinado en otros elementos. Es decir un plan podría estar formado por otros subplanes.
Restricción 8: El refinamiento entre los elementos del diagrama de metas puede hacerse mediante las ligas de descomposición AND, OR o *means-end*. El refinamiento AND y OR sólo puede hacerse entre elementos del mismo tipo. El refinamiento *means-end* debe utilizarse cuando se desea refinar elementos de diferente tipo, es decir, con esta liga podemos especificar que una meta (*end*) puede ser satisfecha mediante varios medios (*means*), los cuales podrían ser planes. Esta liga funciona como un OR.
Restricción 9: Otros elementos que pueden utilizarse en un diagrama de metas son las contribuciones, atributos de recursos, *hardgoals* o planes. Todos estos elementos son opcionales.

4. Representación del modelo organizacional utilizando XML

XML es un metalenguaje que permite definir lenguajes para diferentes necesidades. Es por esto que se ha convertido en un estándar para el intercambio de información estructurada entre diferentes plataformas [10].

4.1. Una representación XML de Tropos

En este artículo se propone la utilización de XML para la manipulación y almacenamiento de los diagramas de Tropos. De esta forma, cada documento XML contendrá, de forma estructurada, los diagramas de actores y de metas que representan a un modelo organizacional.

La Tabla 1 muestra todas las etiquetas XML utilizadas para representar los elementos del diagrama de actores. Se muestran además los atributos que pueden tener algunos de estos elementos. Las etiquetas que representan a los elementos del diagrama de metas se muestran en las Tablas 2 y 3.

La Tabla 2 contiene la descripción de los elementos: *actor, hardgoal*, *softgoal y plan*; todos estos elementos contienen a su vez subelementos. En la Tabla 3 se muestran los elementos: contribución (que son los que nos revelan cómo ciertas metas contribuyen al cumplimiento de otras metas), recurso y dependencia. Se ha omitido la descripción de los elementos complementarios por razones de espacio. La versión completa puede ser consultada en [14].

Los documentos XML funcionan como árboles jerárquicos, los cuales contienen un elemento raíz del cual se desprenden los demás elementos. Por esta razón se ha agregado un elemento llamado *organizational-model* el cual contiene los elementos que representan al diagrama de actores y al diagrama de metas. El elemento *organizational-model* representa al modelo organizacional en su totalidad.

Tabla 1. Elementos de un diagrama de actores

Concepto en Tropos	Etiqueta XML	Atributos	Subelementos
Diagrama de actores	**<actor-model>**		**<organizational-actor>** **<actor-dependency>**
Actor	**<organizational-actor>**	**id:** identificador único para el actor **name:** nombre del actor **generic-concep:** agent, rol o position	**<main-hardgoal>** **<attribute>**
Hardgoal	**<main-hardgoal>**	**name:** nombre de la *hardgoal*	**<attribute>**
Dependencia	**<actor-dependency>**		
Depender	**<depender>**	**id-actor:** Identificador del actor	**<depender>** **<dependum>** **<dependee>**
Dependum	**<dependum>**	**type:** *hardgoal*, *softgoal*, *plan*, *resource.* **name:** Nombre del *dependum*	
Dependee	**<dependee>**	**id-actor:** Identificador del actor	

Tabla 2. Elementos de un diagrama de metas

Concepto en Tropos	Etiqueta XML	Atributos	Subelementos
Diagrama de metas	**<goal-model>**		<**actor**> <**dependency**>
Actor	**<actor>**	**name:** Nombre del actor **generic-concep:** agent, rol o position	**<hardgoal>** **<softgoal>** **<attribute>**
Hardgoal	**<hardgoal>**	**id:** Identificador único **goal :** Nombre de la meta	**<contribution>** **<hardgoal-and-decomposition>** **<hardgoal-or-decomposition>** **<hardgoal-means-end>** **<plan-means-end>**
Softgoal	**<softgoal>**	**id:** Identificador de la *softgoal* **goal:** Nombre de la *softgoal*	**<softgoal-and-decomposition>** **<softgoal-or-decomposition>**
Plan	**<plan>**	**id:** Identificador del plan **plan:** Nombre del plan	**<contribution>** **<plan-and-decomposition>** **<plan-or-decomposition>** **<plan-means-end>**

Tabla 3. Elementos contribución, recurso y dependencia de un diagrama de metas

Concepto en Tropos	Etiqueta XML	Atributos	Subelementos
Contribución	**<contribution>**	**type:** almacena el valor de la contribución: ***p*** **(**positiva), ***fp*** **(**fuertemente positiva), ***n*** **(**negativa), ***fn*** **(**fuertemente negativa).	**<softgoal>**
Recurso	**<resource>**	**id:** Identificador del recurso **name:** Nombre del recurso **type:** almacena el tipo de recurso: material, de información, etc.	**<attribute>**
Dependencia	**<dependency>**		**<id-hardgoal>** **<id-softgoal>** **<id-plan>** **<depender-element>** **<dependum-element>** **<dependee-element>**

5. Gramática del Framework Tropos

Con el objeto de facilitar la construcción de los diagramas se ha desarrollado una gramática para el lenguaje Tropos, la cual ha sido delimitada con las restricciones citadas en la sección 3 de este artículo. Esta gramática es especificada utilizando la notación BNF (Backus-Naur Form) [1], la cual denominamos gramática G compuesta por la cuádrupla: G = (N, T, S, P)

Donde:

- N es el conjunto de símbolos no terminales y se escriben en mayúsculas.
- T representa al conjunto de símbolos terminales y se escriben con minúsculas.
- S es el símbolo inicial y se considera un elemento no terminal.
- P es el conjunto de reglas de producción.

A continuación se detalla cada uno de estos elementos:

N = {ORGANIZATION-MODEL, ACTOR-MODEL, GOAL-MODEL, ACTOR-ORG, ACTOR, HARDGOAL, MAIN-HARDGOAL, SOFTGOAL, PLAN, HG-AND-DESCOMP, HG-OR-DESCOMP, HG-MEANS-END, SG-AND-DESCOMP, SG-OR-DESCOMP, PLAN-AND-DESCOMP, PLAN-OR-DESCOMP, RESOURCE, DEPENDUM, PLAN-MEANS-END, CONTRIBUTION, DEPENDENCY, ACTOR-DEPENDENCY, DEPENDUM-ELEMENT, NAME}

T= {*organization*, *actor-model*, *goal-model*, *actor*, *hardgoal*, *softgoal*, *plan*, *attribute*, *decomposition*, *and*, *or*, *means-end*, *contribution*, *p*, *n*, *fn*, *fp*, *resource*, *dependency*, *depender*, *dependum*, *dependee*}

S= {ORGANIZATIONAL-MODEL}

Las reglas de producción P se explicarán a detalle en las siguientes subsecciones, mostrando la regla de producción, su correspondiente diagrama sintáctico y su equivalente en el DTD.

5.1. El modelo organizacional

La Figura 3 muestra la primera regla de producción, la cual indica que el modelo organizacional debe contener el nombre de la organización, seguido del diagrama de actores y del diagrama de metas. Por motivos de espacio no se detallan algunos elementos no terminales de las reglas de producción, que resultan obvios, tales como, la construcción del nombre de la organización que es representada como una secuencia de caracteres.

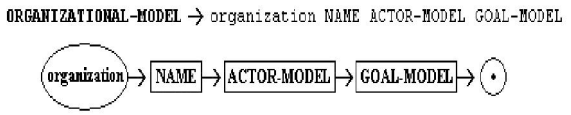

Figura 3. Regla de producción: ORGANIZATIONAL-MODEL

5.1.1. El Diagrama de actores. La Figura 4 muestra la regla de producción del diagrama de actores. En esta regla puede verse representada la restricción 1 del diagrama de actores detallada en la sección 3. La declaración del elemento *actor-model* en el DTD, es la siguiente:

<!ELEMENT actor-model (organizational-actor, organizational-actor+, actor-dependency+)>.

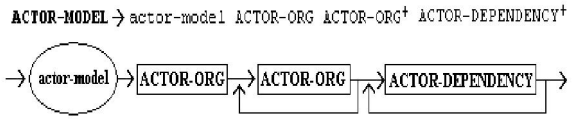

Figura 4. Regla de producción: ACTOR MODEL

a) **Actores.** Cada actor organizacional de este diagrama debe contar con un nombre y al menos una meta (*hardgoal*) principal (restricción 2 del diagrama de actores, detallada en la sección 3). La Figura 5 muestra la regla de producción para la definición de los actores (a) y para la definición de la meta principal de cada actor (b).

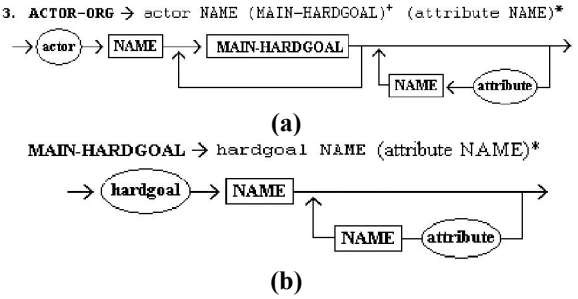

Figura 5. Definición de actores organizacionales

La declaración del actor organizacional y de sus metas (*hardgoal*) principales en el DTD, es el siguiente:

<!ELEMENT organizational-actor (main-hardgoal+, attribute)>*
<!ATTLIST organizational-actor
id ID #REQUIRED
name CDATA #IMPLIED
generic-concep (agent | rol | position) "agent" >
<!ELEMENT main-hardgoal (attribute)>*
<!ATTLIST main-hardgoal
name CDATA "" >

b) **Dependencias.** La Figura 6 muestra la regla de producción para una dependencia. En esta regla puede verse representada la restricción 3 del diagrama de actores detallada en la sección 3. La definición del DTD para la dependencia y para el elemento *dependum* es el siguiente:

<!ELEMENT actor-dependency (depender, dependum, dependee)>
<!ELEMENT depender EMPTY>
<!ELEMENT dependum EMPTY>
<!ATTLIST dependum
type (hardgoal |softgoal | plan | resource) "hardgoal"
name CDATA "" >

<!ELEMENT dependee EMPTY>
<!ATTLIST dependee
id-actor IDREF #REQUIRED >
<!ATTLIST depender
id-actor IDREF #REQUIRED >

5.1.2. El Diagrama de metas. El diagrama de metas es una extensión del diagrama de actores, por lo que este diagrama debe contener los mismos elementos

(restricción 5 del diagrama de metas, detallada en la sección 3).

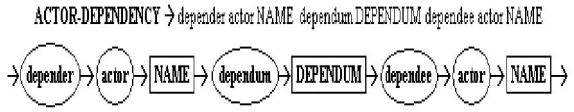

Figura 6. Regla de producción: ACTOR-DEPENDENCY

La regla de producción de este diagrama es mostrada en la Figura 7 y la definición en el DTD es:

<!ELEMENT goal-model (actor, actor+, dependency+)>

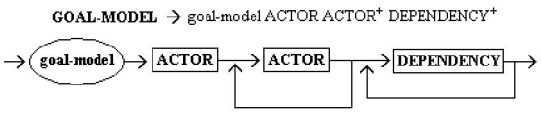

Figura 7. Regla de producción: GOAL-MODEL

a) **Actores del diagrama de metas.** Cada actor de este diagrama es detallado a profundidad. Los actores cuentan con un nombre y al menos una *hardgoal*. Cada actor también puede o no contener una o más *softgoals* y/o atributos (Figura 8). La declaración del actor organizacional en el diagrama de metas en el DTD, es el siguiente:

<!ELEMENT actor (hardgoal+, softgoal, attribute*)>*
<!ATTLIST actor
id IDREF #REQUIRED
name CDATA #IMPLIED
generic-concep (agent | rol | position)"agent" >

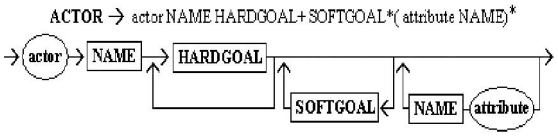

Figura 8. Regla de producción: ACTOR

b) **Hardgoals y Softgoals** (restricción 6 del diagrama de metas, mostrada en la sección 3). Las *hardgoals* deben ser refinadas en otros elementos (Figura 9-a) mientras que las *softgoals* podrían no refinarse (Figura 9-b). El refinamiento de cada *hardgoal* puede ser de tipo: *or, and, means-end*. Cada *hardgoal* puede o no contener contribuciones. Las metas pueden también contener los atributos que sean necesarios para describirlas. La declaración de las metas *hardgoal* y *softgoal* en el diagrama de metas del DTD, es el siguiente:

<!ELEMENT hardgoal ((hardgoal-and-decomposition | hardgoal-or-decomposition | hardgoal-means-end | plan-means-end), attribute, contribution*)>*
<!ATTLIST hardgoal
id ID #REQUIRED
goal CDATA "" >

<!ELEMENT softgoal ((softgoal-and-decomposition | softgoal-or-decomposition)?)>
<!ATTLIST softgoal
id ID #REQUIRED
goal CDATA #IMPLIED
level (composite | final) #IMPLIED >

c) **Planes.** Cada plan definido en el diagrama de metas está representado por un nombre y podría refinarse en otros elementos, además de contar con alguna contribución (restricción 7 del diagrama de metas, mostrada en la sección 3). La Figura 10 muestra el diagrama sintáctico de esta regla de producción. La declaración del elemento *plan* del diagrama de metas en el DTD, es el siguiente:

<!ELEMENT plan ((plan-and-decomposition | plan-or-decomposition | plan-means-end)?,
attribute, contribution*)>*
<!ATTLIST plan
id ID #REQUIRED
plan CDATA #IMPLIED
level (composite | final) #IMPLIED
automatize (true | false) "false" >

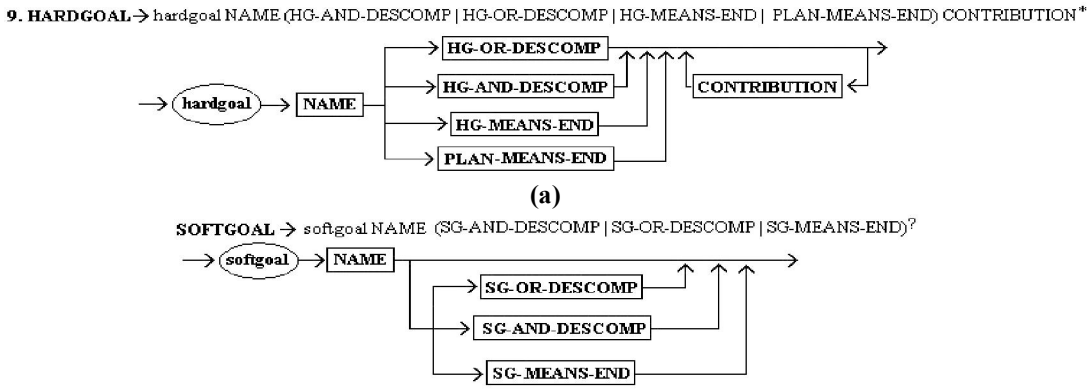

Figura 9. Reglas de producción: HARDGOAL Y SOFTGOAL

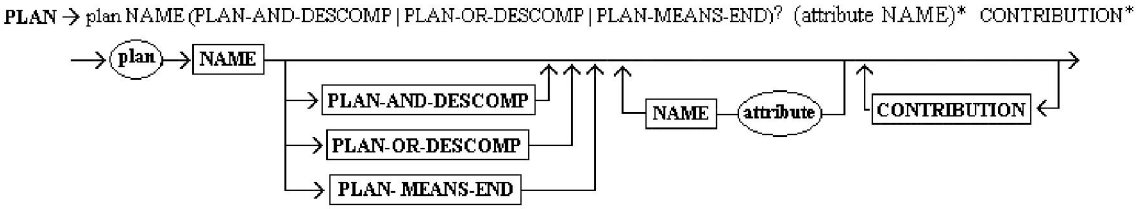

Figura 10. Regla de producción: PLAN

d) **Liga de descomposición AND.** El refinamiento entre los elementos del diagrama de metas puede hacerse mediante la liga de descomposición AND. Este refinamiento sólo puede realizarse entre elementos del mismo tipo (restricción 8 del diagrama de metas, mostrada en la sección 3), es decir, una meta puede refinarse en una o varias submetas mediante una liga de descomposición AND, o en uno o varios planes, pero sin combinar metas con planes dentro de la misma descomposición. La Figura 11 muestra la regla de producción para el refinamiento de un elemento en submetas. La declaración de las ligas de descomposición AND en el DTD es la siguiente:

<!ELEMENT hardgoal-and-decomposition (hardgoal+)>
<!ELEMENT plan-and-decomposition (plan+)>
<!ELEMENT softgoal-and-decomposition (softgoal+)>

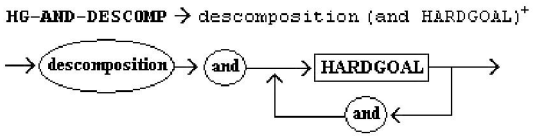

Figura 11. Regla de producción para la liga de descomposición AND

e) **Liga de descomposición OR.** El refinamiento entre los elementos del diagrama de metas puede realizarse mediante la liga de descomposición OR. Este refinamiento incluye sólo elementos del mismo tipo (restricción 8 del diagrama de metas, mostrada en la sección 3), es decir, una meta puede refinarse en dos o más submetas o en uno o más planes, evitando combinar metas y planes dentro de la misma descomposición. La Figura 12 muestra la regla de producción para el refinamiento de un elemento en metas utilizando la descomposición OR. La declaración de las ligas de descomposición OR en el DTD es la siguiente:

<!ELEMENT hardgoal-or-decomposition ((hardgoal, hardgoal+))>
<!ELEMENT softgoal-or-decomposition (softgoal, softgoal+)>
<!ELEMENT plan-or-decomposition (plan, plan+)>

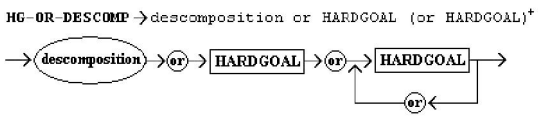

Figura 12 Regla de producción para la liga de descomposición OR

f) **Descomposición *Means-end.*** El refinamiento entre elementos de diferente tipo en el diagrama de metas puede hacerse mediante la liga de descomposición *means-end*. Con esta liga podemos tener que una meta (*End*) puede ser satisfecha mediante varios medios (*means*), los cuales pueden ser planes (restricción 8 del diagrama de metas, mostrada en la sección 3). Esta liga funciona como un OR a diferencia de que en este tipo de descomposición puede haber al menos un elemento organizacional. La Figura 13 muestra este tipo de descomposición. La declaración de la liga de descomposición *means-end* en el DTD es la siguiente:

<!ELEMENT hardgoal-means-end (hardgoal+)>
<!ELEMENT plan-means-end (plan+)>

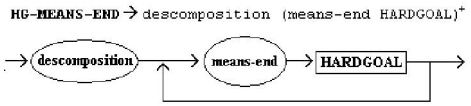

Figura 13. Regla de producción para la liga de descomposición *means-end*

g) **Contribuciones.** A las contribuciones en el diagrama de metas se les ha colocado el valor de *p* si se trata de una contribución positiva, de *n* si se trata de una contribución negativa, de *fp* si resulta fuertemente positiva, o de *fn* si corresponde a fuertemente negativa. A este valor le sigue una *softgoal* quien realiza la contribución. La Figura 14 ilustra estos valores. La declaración del elemento *contribution* en el DTD es la siguiente:

<!ELEMENT contribution (softgoal)>
<!ATTLIST contribution
type (p | n | fp | fn) #REQUIRED >

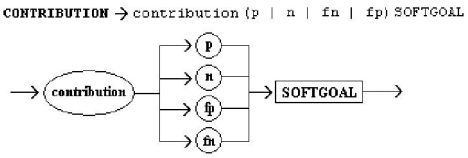

Figura 14. Regla de producción: CONTRIBUTION

h) **Recursos.** Los recursos en el diagrama de metas tienen un nombre que los identifica; cada recurso puede o no contener atributos. Los atributos de un recurso contienen un nombre y un tipo. La Figura 15 muestra la regla de producción de este elemento. La declaración del elemento *resource* en el DTD es la siguiente:

<!ELEMENT resource (attribute)>*
<!ATTLIST resource
id ID #REQUIRED
name CDATA #IMPLIED
type (material | informational | other) #REQUIRED >

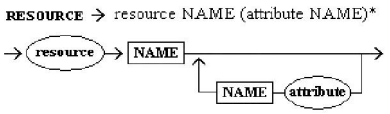

Figura 15. Regla de producción: RESOURCE

i) **Dependencias en el diagrama de metas.** Las dependencias entre los actores definidas en el diagrama de actores deben estar presentes en el diagrama de metas (restricción 1 del diagrama de actores, mostrada en la sección 3), sin embargo en este diagrama deben definirse los elementos que están ligados por la dependencia. En nuestra especificación propuesta se han definido todas las alternativas que pueden presentarse en los elementos que forman una dependencia (Figura 16). Por ejemplo, los elementos *depender* y *dependee* pueden ser: *hardgoal*s, *softgoal*s o planes (Figura 17 y Figura 19), mientras que el elemento *dependum* sólo podría tratarse de un plan o un recurso (Figura 18). La declaración del elemento dependency en el DTD es la siguiente:

<!ELEMENT dependency ((denpender-element, dependum-element, denpendee-element)
| (id-hardgoal,(id-hardgoal | id-softgoal),id-hardgoal)
|(id-softgoal,id-softgoal,id-softgoal)) >

La declaración del elemento *depender-element* en el DTD es:
<!ELEMENT denpender-element (id-plan | id-hardgoal)>

El elemento *dependum-element* se declara en el DTD de la siguiente manera:

<!ELEMENT dependum-element (id-plan|resource)>

Declaración del elemento *dependee-element* en el DTD:
<!ELEMENT denpendee-element (id-plan | id-hardgoal)>

En esta propuesta se han diseñado elementos complementarios para las dependencias, las cuales son colocadas en el diagrama de metas. Estos elementos son: *id-hardgoal*, *id-plan*, *id-softgoal*, *depender-element*, *dependum-element* y *dependee-element*, lo anterior con el objetivo de enriquecer la gramática y cumplir con ciertas restricciones con las que cuenta el framework Tropos.

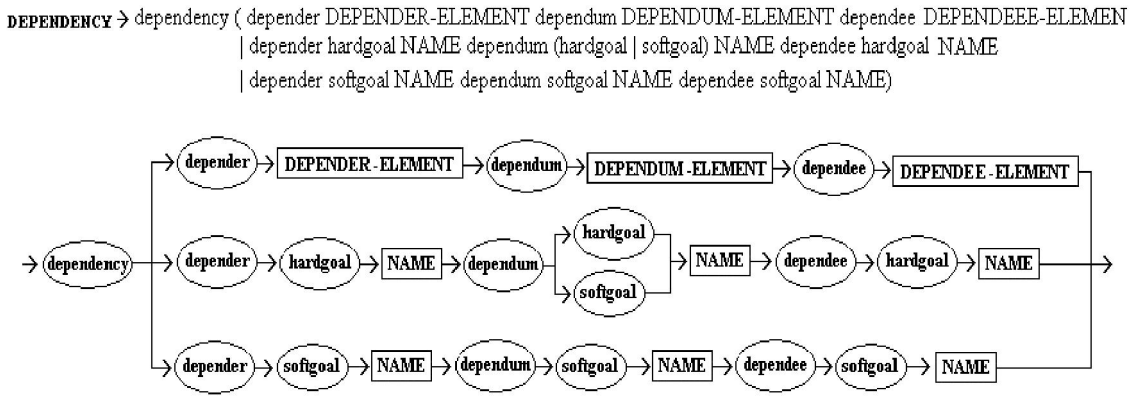

Figura 16. Regla de producción: DEPENDENCY

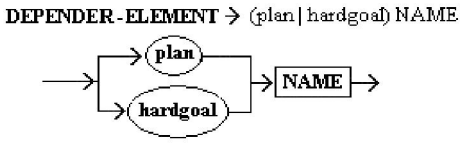

Figura 17. Regla de producción: DEPENDER-ELEMENT

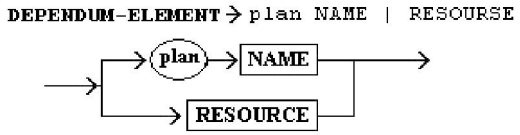

Figura 18. Regla de producción: DEPENDUM-ELEMENT

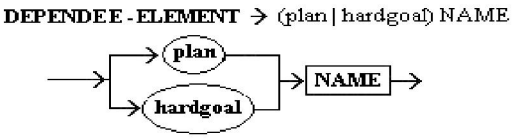

Figura 19. Regla de producción: DEPENDEE-ELEMENT

6. Caso de estudio

Con el objetivo de comprobar nuestro lenguaje de especificación, se desarrollaron varios casos de estudio. En este artículo se detalla una vista parcial del caso de estudio utilizado en [12], el cual muestra el modelo organizacional de la empresa *RentaCar*, dedicada a la renta de autos en la región de Alicante, España.

Por razones de espacio se muestran sólo fragmentos, tanto del diagrama de metas, como del documento XML. La figura 20 detalla gráficamente al actor cliente del diagrama de metas.

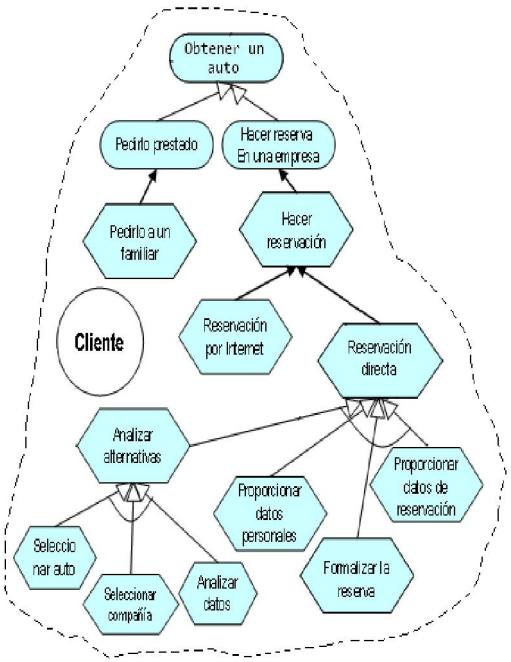

Figura 20. Vista parcial del diagrama de metas del caso de estudio *RentaCar*. Tomado de [12]

La Tabla 4 muestra la representación del actor "Cliente" en XML.

Tabla 4. Representación del actor "Cliente" en XML

```
<actor name="Cliente" generic-concep ="agent">
 <hardgoal id="goal101" goal="Obtener un auto">
 <hardgoal-or-decomposition>
 <hardgoal id="goall02" goal="Pedirlo prestado" >
 <plan-means-end>
 <plan id="plan101" plan="Pedirlo a un familiar"
          level="final" automatize="false">
 </plan>
 </plan-means-end>
 </hardgoal>
 <hardgoal id="goal103"
       goal="Hacer reserva en una empresa">
 <plan-means-end>
 <plan id="plan102" plan="Hacer reservacion"
          level="composite" automatize="false">
 <plan-means-end>
 <plan id="plan103" plan="Reservacion por internet"
           level="final" automatize="false"></plan>
 <plan id="plan104" plan="Reservacion directa"
           level="composite" automatize="false">
 <plan-and-decomposition>
 <plan id="plan105" plan="Analizar alternativas"
           level="composite" automatize="false">
 <plan-and-decomposition>
 <plan id="plan110" plan="Seleccionar auto"
                    level="final"
                    automatize="false"></plan>
 <plan id="plan111" plan=">Seleccionar
           compañia" level="final"
                    automatize="false"></plan>
 <plan id="plan112" plan="Analizar datos"
           level="final" automatize="false"></plan>
 </plan-and-decomposition>
 </plan>
 <plan id="plan106" plan="Proporcionar datos
           personales" level="final"
                automatize="false"></plan>
 <plan id="plan107" plan="Formalizar la reserva"
           level="final" automatize="false"></plan>
 <plan id="plan108"
                plan="Proporcionar datos de reservacion"
                level="final" automatize="false"></plan>
 <plan id="plan109" plan="Proporcionar datos
              de servicios" level="final"
              automatize="false"></plan>
 </plan-and-decomposition>
 </plan>
 </plan-means-end>
 </plan>
 </plan-means-end>
 </hardgoal>
 </hardgoal-or-decomposition>
 </hardgoal>
  </actor>
```

7. Conclusiones y trabajos futuros

En este artículo se presenta un lenguaje de especificación para el framework Tropos basado en XML, el cual ha sido descrito con el suficiente nivel de detalle para definir las reglas de creación de los modelos de actores y metas.

Esta propuesta presenta varios beneficios, algunos de ellos son:

- Se establecieron restricciones para el uso de las primitivas del framework Tropos. Esto constituye una ventaja contra el resto de herramientas de modelado con Tropos donde no se cuenta con reglas ni restricciones para la creación de modelos, lo cual, como fue ya demostrado en evaluaciones de Tropos, no es de utilidad para usuarios no experimentados.
- Se cuenta con una gramática para representar los modelos organizacionales del framework Tropos, especificados en un DTD. La definición de una gramática requirió de un esfuerzo importante de modelado para determinar todas las posibles combinaciones y restricciones en el uso de primitivas para modelar en Tropos.
- Es posible verificar que los modelos generados respeten la gramática propuesta por los modelos. Si bien es cierto que la gramática impone restricciones, también es cierto que esta es una ventaja significativa para aquellos usuarios inexpertos en modelado con Tropos.

Al representar los modelos organizacionales en XML, se cuenta con un formato independiente de la plataforma de implementación.

Esta propuesta surge de la necesidad de utilizar los modelos organizacionales Tropos como punto de partida para la generación semi-automática de un modelo de requisitos [13]. De esta manera se analizaron varias herramientas desarrolladas, por ejemplo: OME/openOME [17], TAOM4E [2], J-PRiM [8], etc. Estas herramientas son sólo algunas que se han generado en torno a la metodologías de *i** y Tropos.

Muchas de estas herramientas utilizan la metodología Tropos o i* según sus necesidades ya que no se ha estandarizado el uso de las primitivas y la forma de llevar a cabo el refinamiento entre los elementos de sus diagramas.

En nuestra propuesta la gramática ha sido delimitada y se han establecido algunas restricciones para ayudar a los analistas en la uniformidad de esta metodología.

Actualmente nos encontramos desarrollando una herramienta para la generación de sistemas de información a partir de modelos organizacionales, la cual debe utilizar la especificación XML que es generada por la herramienta desarrollada en este trabajo de investigación, para generar el correspondiente modelo conceptual orientado a objetos.

Reconocimientos: Esta investigación ha sido financiada por el Consejo de Ciencia y Tecnología en México, con el proyecto de investigación No. 90859.

Referencias

[1] Aho, A., R. Sethi, J. Ullman, *Compiladores. Principios, técnicas y herramientas*, Massachusetts, Pearson Educación, 1998.

[2] D. Bertolini, A. Novikau, A. Susi, y A. Perini, "TAOM4E: an Eclipse ready tool for Agent-Oriented Modeling. Issue on the development process", *Technical Report, Automated Reasoning Systems Division*, ITC-IRST, Italia, 2005.

[3] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, y J. Mylopoulos, "Tropos: An agent-oriented software development methodology", *Autonomous Agents And Multi-Agent Systems*, vol. 8, pp. 203-236, 2004.

[4] C. Cares, X. Franch, A. Perini, y A. Susi, "iStarML. The *i** Mark-up Language", *3rd International i* Workshop*, Recife, Brazil, Febrero 11-12, 2008, CEUR ISSN 1613-0073.

[5] H. Estrada, A. Martinez, O. Pastor, J. Mylopoulos, "An experimental evaluation of the *i** Framework in a Model-based Software Generation Environment", *18th Conference on Advanced Information Systems Engineering (CAISE 06). Luxembourg, Grand-Duchy of Luxembourg*, Junio 2006, pp. 513-527.

[6] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, y P. Traverso, "Specifying and analyzing early requirements in Tropos", *Requirements Engineering*, vol. 9, pp. 132-150, 2004.

[7] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, "Requirements Engineering Meets Trust Management", *Model, Methodology, and Reasoning. Lecture Notes in Computer Science*, vol. 2995, pp. 176-190, 2004.

[8] G. Grau, X. Franch, "J-PRiM: A Java Tool for a Process Reengineering *i** Methodology", *Requirements Engineering, 14th IEEE International Conference (RE'06)*, pp. 359-360.

[9] G. Grau, X. Franch, E. Mayol, C. Ayala, C. Cares, J. Carvallo, M. Haya, F. J. Navarrete, P. Botella, y C. Quer, "RiSD: A Methodology for Building *i** Strategic Dependency Models", *17th International Conference on Software Engineering and Knowledge Engineering (SEKE'05)*, Taipei, Taiwan; China, 2005.

[10] Gutiérrez, A., R. Martínez, *XML a través de ejemplos*, Ra-ma, 2001.

[11] L. Liu, E. Yu, J. Mylopoulos, "Security and Privacy Requirements Analysis within a Social Setting", *International Conference on Requirements Engineering (RE'03)*, Monterey, California, USA, 2003.

[12] A. Martínez, H. Estrada, y L. A. Gama, "Una guía rápida de la metodología Tropos", *Gerencia Tecnológica e Informática*, Vol. 7, Nº 19, ISSN 1657-8236, Diciembre 2008.

[13] A. Martinez, O. Pastor, J. Mylopoulos, H. Estrada, "From Organizational models to Software Requirements", *21st International Conference on Software Engineering & Knowledge Engineering (SEKE2009)*, Boston, Massachussetts, 2009. ISBN 1-891706-24-1, pp. 61-66.

[14] A. Martínez, C. Velasco, E. Morales, "Generación de Modelos Conceptuales OO a partir de Modelos Organizacionales", *Reporte Técnico presentado a CONACYT*, México, Diciembre 2009, pp. 1-46.

[15] H. Mouratidis, M. Weiss, P. Giorgini, "Security Patterns Meet Agent Oriented Software Engineering: A Complementary Solution for Developing Secure Information Systems", *Conceptual Modeling – ER 2005: 24th International Conference on Conceptual Modeling*, pp. 225-240.

[16] F. Sannicoló, y otros. "The Tropos Modeling Language. A user guide", *Technical Report #DIT-02-0061*, Universidad de Trento, Italia, 2002.

[17] OpenOME, an open-source requirements engineering tool. (online). http://www.cs.toronto.edu/km/openome/

[18] Yu, E., *Modelling Strategic Relationships for Process Reengineering*, PhD Thesis, University of Toronto, Toronto, 1995.