Análisis Comparativo de Lenguajes de Modelado Orientados a Objetivos basados en *i**

Claudia P. Ayala, Carlos Cares, Juan P. Carvallo, Gemma Grau, Mariela Haya, Guadalupe Salazar, Xavier Franch, Enric Mayol, Carme Quer

Universitat Politècnica de Catalunya (UPC), Barcelona (Spain)
{cayala, ccares, carvallo, ggrau, mhaya, gsalazar, franch, mayol, cquer}@lsi.upc.es

Resumen. Los modelos orientados a objetivos son usados frecuentemente en disciplinas tales como la ingeniería de requisitos o el modelado de procesos en organizaciones. *i** es una de las notaciones más empleadas para construir este tipo de modelos. Desafortunadamente, no existe una definición única de *i** sino diferentes versiones y variantes que, con frecuencia, no están totalmente definidas dificultando su comprensión y utilización. En este artículo, se presenta un estudio comparativo de las tres variantes más utilizadas de *i**: la propuesta original de Eric Yu, el lenguaje GRL y el lenguaje utilizado en el método TROPOS. A continuación, se formula un modelo conceptual genérico como marco de referencia de las variantes estudiadas, se muestra cómo a partir de este modelo genérico pueden generarse los modelos específicos de cada una, y se evidencia que también puede usarse para clasificar algunas variantes puntuales que encontramos en la literatura.

1. Introducción

En los últimos años, la construcción de modelos orientados a objetivos se ha convertido en una herramienta habitual en distintos ámbitos tales como el de la ingeniería de requisitos y el del modelado de procesos en organizaciones [1, 2, 3].

Existen diversas propuestas de lenguajes para la construcción de modelos orientados a objetivos. Entre ellas destaca poderosamente la notación *i** propuesta por Eric Yu en la primera mitad de la década de los 90 [4, 5]. *i** permite expresar de forma clara y sencilla los objetivos de los actores que aparecen en los modelos y las dependencias entre ellos. Además, *i** cuenta con una notación gráfica que permite tener una visión intuitiva y unificada del entorno modelado mostrando tales actores y dependencias.

No obstante, el uso de *i** entraña un riesgo que se descubre pronto, pues no existe una definición única del lenguaje. Además, las definiciones existentes no son tan precisas como sería deseable, ya que contienen ambigüedades, contradicciones e incompletitudes. Esta laxitud es en cierto modo intencionada, ya que debido a su naturaleza y objetivos, se pretendió dotar al lenguaje de un cierto grado de libertad. No obstante, esta flexibilidad crea una cierta inseguridad en el uso de la notación, y se genera la tendencia de que cada equipo de investigación cree su propio *i** a medida, dificultándose el intercambio de conocimiento en la comunidad interesada. Algunas de las variantes en proceso de consolidación son el lenguaje GRL [6, 7] y el lenguaje de la metodología TROPOS [8, 9, 10]. De hecho, estas dos variantes crean incluso

una confusión mayor: ¿cuándo utilizar una u otra, o la propuesta original de Yu?, y ¿cuáles son las características de cada una de estas tres propuestas? Las respuestas son difíciles, sobre todo teniendo en cuenta que hay grupos y personas involucradas en más de una propuesta, e incluso distintas versiones de una misma propuesta.

Este artículo tiene como objetivo clarificar algunos de estos interrogantes, con la definición de un marco de referencia de ayuda al análisis y clasificación de variantes de *i**. En las secciones 2, 3 y 4 se presentan brevemente las características y el análisis de cada una de las variantes citadas. Además, complementamos este estudio con observaciones siguiendo algunos de los criterios de análisis de especificaciones informales propuestos por Meyer [11]: *ruidos* (presencia de información irrelevante), *silencios* (ausencia de información), *contradicciones* y *ambigüedades*. En la sección 5 se muestra un análisis comparativo de las propuestas. En la sección 6 se propone un modelo conceptual que sirva como marco de referencia común y se estudia una de las variantes analizadas respecto a dicho marco. En la sección 7 se menciona brevemente cómo podrían integrarse en este marco otras variantes propuestas en la literatura. Finalmente, la sección 8 incluye las conclusiones. Queda fuera del ámbito de este artículo detallar las bases del modelado orientado a objetivos.

2. El lenguaje *i**

El lenguaje *i** definido por Eric Yu [2, 4] es la propuesta seminal de la familia de lenguajes orientados a objetivos que nos ocupa. En concreto, la tesis doctoral de Yu [4] puede considerarse como la definición oficiosa del lenguaje y es la que hemos utilizado como referencia en esta sección.

En *i** se propone el uso de dos modelos, cada uno correspondiente a un nivel de abstracción: el nivel intencional representado por el *Strategic Dependency Model* (SD) y el nivel racional representado por el *Strategic Rationale Model* (SR).

El modelo SD consiste en un conjunto de nodos que representan actores *(actors)* y un conjunto de relaciones entre estos actores. Las relaciones representan dependencias (*dependencies*) que permiten expresar que un actor (*depender*) depende de otro actor (*dependee*) para conseguir un determinado propósito (*dependum*). El *dependum* es un *elemento intencional* que puede ser de tipo: recurso (*resource*), tarea a realizar (*task*), objetivo (*goal*) o requisito no funcional (*softgoal*). El modelo permite definir la importancia (*strength*) de la dependencia para cada uno de los actores que intervienen en ella. Existen tres grados de importancia: abierta (*open*), comprometida (*commited*) o crítica (*critical*).

El modelo SR permite visualizar los elementos intencionales dentro de los límites (*boundary*) de un actor para refinar el modelo SD y añadirle capacidad de razonamiento. Para ello, se asignan las relaciones de dependencia del SD a elementos intencionales internos al actor y se añaden relaciones medio-fin (*means-end*) y descomposición de tarea (*task-decomposition*):

- Una relación *means-end* se establece entre dos elementos intencionales de un mismo actor. Uno de los elementos intencionales de la relación es el medio (generalmente una tarea) que contribuye a la realización de un fin. Cuando existen varios medios para un determinado fin existe una relación *OR*, la cual indica las diferentes maneras de lograr un fin. Las relaciones posibles son: *Goal-Task*,

Resource-Task, *Task-Task*, *Softgoal-Task*, *Softgoal-Softgoal* y *Goal-Goal*. En las relaciones *means-end* en las que aparece un *softgoal* como fin se indica si la contribución del medio a la consecución del *softgoal* es positiva o negativa.

- Una relación *task-decomposition* descompone una tarea en elementos intencionales de cualquier tipo. Cuando son varios los elementos intencionales que descomponen una tarea, existe una relación *AND* entre ellos, pudiéndose completar con restricciones (*constraints*) para refinar esta relación. Finalmente, se puede marcar la importancia del elemento intencional en la realización de la tarea, de la misma forma que en las dependencias del modelo SD.

La notación gráfica de estos conceptos se muestra en la figura 1. El ejemplo presentado ilustra el modelado de un proceso de tutoría académica. A la izquierda se muestra el SR del tutor y las relaciones que se establecen entre sus elementos intencionales internos. A la derecha se muestra el SD del alumno con sus dependencias respecto al tutor.

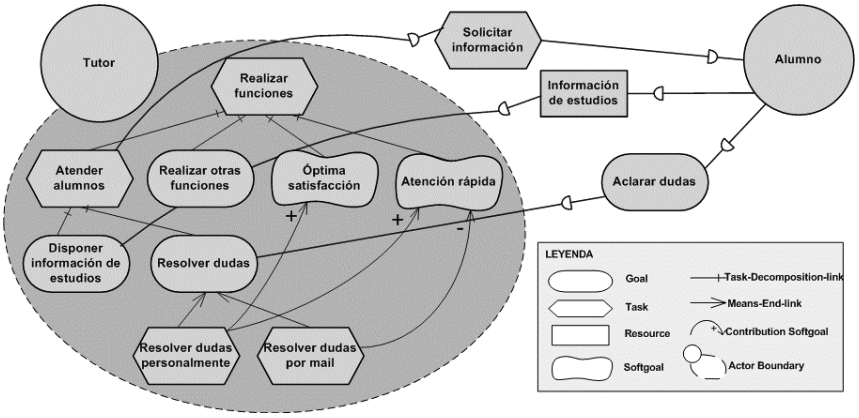

Figura 1. Ejemplo de modelo *i** de Yu para un proceso de tutoría académica.

Los actores pueden especializarse en agentes (*agents*), roles (*roles*) y cargos (*positions*). Un cargo puede cubrir varios roles. Los agentes representan instancias particulares de personas, máquinas o software que ocupan un determinado cargo dentro de la organización y que, por lo tanto, pueden jugar varios roles. Los actores y sus especializaciones pueden desagregarse en otros actores mediante la relación *Is-Part-Of*.

Los modelos SR constan de elementos de razonamiento adicionales como son las rutinas (*routines*), reglas (*rules*) y creencias (*beliefs*). Una rutina es una de las opciones que aparecen en una relación *means-end* e indica un determinado curso de acción a seguir. Se deben considerar las reglas y creencias como información complementaria de condiciones de aplicabilidad de dichas rutinas.

En la figura 2 se expone un extracto del modelo conceptual correspondiente a *i** en lenguaje UML [12] que integra la mayor parte de los conceptos descritos, exceptuando los relacionados con elementos de razonamiento. Debe observarse que se ha generalizado como *Dependable Node* aquellos elementos de los modelos para los que puede definirse dependencias, es decir, los actores y los elementos intencionales

de los SR. Por otro lado, se han definido dos asociaciones derivadas que permiten representar la raíz de la descomposición SR de un actor y la equivalencia entre las dependencias SD y sus asignaciones al modelo SR.

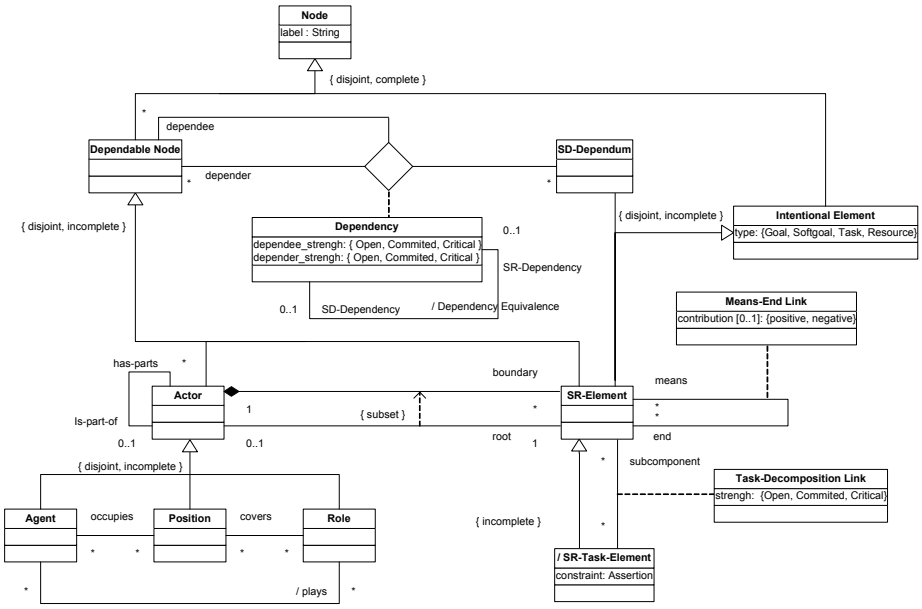

Figura 2. Extracto del modelo conceptual de *i** de Yu.

El análisis de [4] ha mostrado diversas situaciones anómalas debidas principalmente a que en varios puntos de la documentación se hace referencia a una especificación en TELOS [13], que en realidad está incompleta. Esta falta de formalización ha hecho necesario un estudio intenso de las descripciones textuales y de los gráficos para completar ciertos aspectos y ha conducido a las siguientes observaciones:

- **Ruido.** El concepto de rutina se define en la formalización y descripción del modelo SD. Este hecho induce a confusión, ya que en realidad su uso se hace visible como elemento de razonamiento únicamente en el modelo SR. Por otro lado, la relación *IS-A* (generalización/especialización) no forma parte del lenguaje pero se utiliza profusamente en los gráficos como una simplificación de la notación.
- **Silencio.** Se han detectado las siguientes situaciones: no se indica si se permite más de una raíz en la descomposición interna de un actor; no está explícito si cualquier tipo de elemento intencional puede ser raíz de una descomposición; no se especifica si un actor puede ser parte-de varios actores; no se detalla si un *dependum* puede estar relacionado con más de un *dependee* y/o más de un *depender*; la formalización y descripción de *constraints* de las relaciones *task-decomposition* es incompleta; finalmente, a pesar de que se dan definiciones de los distintos tipos de nodos (actores y elementos intencionales), únicamente se pueden deducir sus criterios de utilización de los ejemplos incluidos en el texto.

- **Ambigüedad.** La importancia (*strengh*) de la dependencia se interpreta de forma diferente según se encuentre en el extremo del *depender* o del *dependee*, dando a entender que una misma dependencia puede tener diferente importancia para actor participante. Sin embargo, no existe ningún ejemplo que lo clarifique.

3. El lenguaje GRL

GRL *(Goal-oriented Requirement Language)* es un lenguaje para apoyar el modelado orientado a objetivos y de razonamientos con requisitos no funcionales. Este lenguaje tiene una fuerte influencia de *i** así como también del *NFR framework* para la especificación de los requisitos no funcionales [14]. GRL forma parte de la notación URN (*User Requirements Notation*) [6], que ha sido propuesta como estándar de la ITU-T (*International Telecommunication Union -Telecommunication Standardization Sector*) [15]. GRL distingue, al igual que *i**, tres principales categorías de conceptos: elementos intencionales, relaciones intencionales y actores (que no admiten especializaciones). Las principales diferencias respecto a *i** son que GRL ofrece constructores para establecer relaciones con elementos externos al modelo (elementos no-intencionales y atributos de conexión), y que GRL posee elementos adicionales de justificación y/o contextualización como creencias (*beliefs*), correlaciones, tipos de contribuciones y etiquetas de evaluación para especificar los estados de satisfacción, ampliando así los tipos y rango de calificaciones de las relaciones intencionales existentes en *i**.

Las observaciones derivadas del resultado del análisis de GRL según los criterios establecidos, son las siguientes:

- **Ruido.** La existencia de una triple especificación sintáctica (BNF gráfica, BNF textual y especificación XML), impide una certeza inmediata de que las fórmulas estén bien formadas para GRL e incrementa innecesariamente el esfuerzo de comprensión de esta variante.
- **Silencio.** La especificación sintáctica permite una variedad de fórmulas que no son cubiertas totalmente en las explicaciones realizadas en lenguaje natural.
- **Ambigüedad.** La especificación sintáctica formal, acompañada de explicaciones concisas y ejemplos sencillos, evita ambigüedades en la construcción de expresiones GRL. Sin embargo, existen ambigüedades semánticas respecto a las contribuciones, dándose el caso de que existen contribuciones que tienen una calificación referida a operadores binarios (*AND*, *OR*), pero que permiten su construcción con un solo operando.
- **Contradicción.** Se ha detectado una contradicción entre las especificaciones sintácticas de GRL: la BNF textual determina un conjunto delimitado de valores para tipos de contribuciones con 11 elementos terminales; sin embargo, la especificación XML correspondiente, especifica como elemento terminal un tipo de dato básico (*CDATA*), lo cual no necesariamente correspondería a un valor dentro del conjunto delimitado de tipos. Otra contradicción aparece al estudiar la herramienta OME [16] para la edición de modelos GRL, donde se observa que se permite usar la especialización de roles, agentes y posiciones, a pesar de que la especificación formal de GRL no hace esta consideración.

4. Tropos

Tropos es un proyecto [10] en el que participan distintas universidades que tiene como objetivo definir una metodología de desarrollo de sistemas software orientados a agentes y que utiliza como lenguaje de modelado una variante de *i** [8, 9].

La metodología propuesta abarca desde la etapa de análisis de requisitos hasta la de implementación. En cada una de ellas se utilizan los conceptos propios de *i** (actor, dependencia, etc.) para mostrar una vista del modelo teniendo en cuenta la etapa.

En las vistas correspondientes a las etapas de análisis de requisitos, los actores se utilizan para modelar los *stakeholders* del dominio y el nuevo sistema a construir. De esta manera, las dependencias representadas en estas vistas serán dependencias entre *stakeholders*, y entre éstos y el sistema software.

En las vistas correspondientes a las etapas de diseño, los actores modelan los componentes de la arquitectura del sistema y los agentes software que finalmente se implementaran. Las dependencias, representan pues, el intercambio de datos y control entre componentes y agentes, y definen las habilidades o responsabilidades asignadas a cada uno y que deberán ser implementadas.

Las diferencias en cuanto a lenguaje respecto a *i** son relativas a la sintaxis de algunos conceptos. Una primera diferencia radica en que Tropos no distingue entre modelos SD y SR. En cambio, propone vistas de un único modelo para cada etapa de desarrollo. Así pues, Tropos modela explícitamente de forma separada aspectos relacionados con el dominio de aspectos relacionados con el sistema software.

Las observaciones derivadas del resultado del análisis de Tropos según los criterios establecidos, son las siguientes:

- **Ruido**: Al llevar a cabo el análisis del lenguaje utilizado en Tropos se constata que la mayoría de las publicaciones consultadas hacen más énfasis en la metodología de desarrollo que en el lenguaje a utilizar.
- **Silencio**: Aunque existe una guía de usuario de la versión *i** de Tropos [9] (escrita en italiano), en la que se define en detalle el lenguaje y se dan reglas y guías de su utilización, se desconoce la existencia de una versión en inglés.
- **Ambigüedades:** En artículos como [17] se utiliza directamente la versión de *i** de Yu o la de GRL en lugar de su propia versión del lenguaje, lo que induce a confusión.
- **Contradicciones**: Tropos parte de la hipótesis de que todo objetivo y toda tarea (a la que en Tropos llaman *plan*) del modelo deben estar asignados a un actor. En cambio, en el modelo conceptual que representa los conceptos usados por Tropos y que se incluye en [9] esta asignación es opcional. Por otro lado, los elementos intencionales contribuyentes y contribuidos en las relaciones de contribución no coinciden en los metamodelo y explicaciones de las fuentes principales [8] y [9], e incluso en la [9], de éstos con los ejemplos que usan contribuciones.

5. Análisis comparativo

En esta sección se presenta un análisis comparativo de las tres variantes estudiadas. Para ello identificamos un total de 14 criterios, agrupados en dos grandes categorías, estructurales y no estructurales:

- Los criterios estructurales se refieren a las características propias de las construcciones del lenguaje, es decir, tipos de, y relaciones entre, actores y elementos intencionales, así como tipos de modelos y posibilidad de desagregación de los mismos. También consideramos relevante la posibilidad de referenciar elementos externos a los modelos orientados a objetivos.
- Los criterios no estructurales analizan cuestiones relativas principalmente a la definición y uso de los lenguajes. Así, nos interesamos por el tipo de descripción sintáctica y semántica, las etapas del ciclo de vida del desarrollo de software a las que se dirige, el grado de diseminación, la existencia de herramientas y otros elementos de interés que se relacionan con estas propuestas, así como lenguaje de especificación y herramientas disponibles.

A continuación se detallan y se comentan las diferencias y similitudes de las tres variantes respecto a cada uno de estos 14 criterios (ver tabla 1).

- **Tipos de modelos.** Los tipos de modelos de *i** (SD y SR) no existen en GRL ni en Tropos. Sin embargo, se puede conseguir los mismos niveles de abstracción mediante la representación de los límites de los actores. En Tropos se construyen vistas del modelo que varían según la etapa de desarrollo de software.
- **Tipos de actores.** Se distingue GRL por no permitir la especialización de actores, aunque como se ha indicado anteriormente existen contradicciones entre distintas fuentes consultadas. En Tropos algunas de las especializaciones son específicas para la fase de diseño.
- **Elementos intencionales.** No hay diferencias en cuanto a tipos de elementos intencionales más que un pequeño cambio de nomenclatura: el concepto de tarea se denomina *task* en *i** y GRL, y *plan* en Tropos; asimismo, Tropos denomina *hard goals* a los objetivos, y el término *goal* generaliza los de *hard* y *soft goal*. Nótese que, aunque GRL define las creencias (*beliefs*) como un tipo de elemento intencional, el uso que se hace de ellas provoca que en este estudio sean consideradas como un elemento de razonamiento.
- **Relaciones entre actores.** Se distingue entre dependencias y otros tipos de relaciones. La diferencia principal está en GRL que no permite otros tipos de relaciones. Por otra parte, la relación *is-part-of* existe únicamente en *i**.
- **Relaciones entre elementos intencionales.** Existen cuatro tipos básicos recurrentes: la dependencia, la relación medio-fin, la descomposición y la contribución. Las dependencias funcionan de la misma forma en los tres lenguajes estudiados (los 4 tipos de *dependum* correspondientes a los tipos de elementos intencionales). No obstante, los otros tres tipos de relaciones difieren en: la nomenclatura usada; los elementos intencionales permitidos como origen y destino de las relaciones; la capacidad expresiva de los tipos de contribuciones; la combinación de los contribuyentes. En la tabla 2, se muestran estas diferencias: para cada relación y variante de *i**, vemos el constructor correspondiente del lenguaje, las combinaciones válidas de elementos intencionales (izquierda, destino; derecha, origen), la forma de combinación y en el caso de contribución, los valores posibles. Destacamos que el concepto de contribución en GRL tiene dos construcciones, la contribución misma y también la correlación. Al establecer las combinaciones válidas de elementos intencionales, usamos letras para abreviar: **O**bjetivo, requisito **N**o-**F**uncional, **T**area, **R**ecurso, **C**reencia y re**L**ación, representando este último concepto los tres tipos mismos de relación, que GRL

contempla como posible participante en las contribuciones; el símbolo '*' es una abreviatura de **O**, **NF**¸ **T** y **R**. Por ejemplo, puede observarse que solamente Tropos permite definir relaciones medio-fin en las que el contribuyente (el medio) es un objetivo o requisito no funcional.

- **Niveles de desagregación.** En el caso del *i** las relaciones *is-part-of* entre actores facilitan la desagregación. En el caso de Tropos la desagregación es posible dado que dentro de los límites de un actor pueden aparecer nuevos actores.
- **Elementos de razonamiento adicional.** Los tipos varían de un lenguaje a otro, así como también los elementos posibles de cada tipo. En el caso de Tropos algunos elementos son específicos de las etapas de diseño.
- **Relaciones con elementos externos al modelo.** Aunque parece una capacidad necesaria para cualquier lenguaje de modelado de sistemas, estas relaciones así como los elementos externos existen únicamente en GRL.
- **Etapas del ciclo de vida cubiertas.** En este caso se han hecho explícitas las etapas que en la literatura estudiada se indican como objetivo de cada lenguaje. Sin embargo, desde nuestro punto de vista los tres lenguajes pueden ser usados en cualquiera de las etapas.
- **Diseminación y estandarización.** Para evaluar la consolidación de cada lenguaje se ha considerado principalmente la cantidad de citas bibliográficas de las publicaciones relacionadas con cada lenguaje. Por otra parte, el único lenguaje que está en progreso de ser adoptado como estándar es GRL [15].
- **Herramientas.** No hemos encontrado ninguna herramienta que soporte Tropos, aunque existe la herramienta T-Tool que soporta una evolución de éste (Formal Tropos [18]). OME [16] ofrece la posibilidad de editar modelos *i** y GRL, aunque permite añadir elementos que no existen en las publicaciones estudiadas. Lo mismo pasa con la herramienta REDEPEND [19] para *i**.
- **Acompañamiento del lenguaje.** Se ha explicitado en esta categoría los aspectos adicionales que complementan cada lenguaje.
- **Descripción sintáctica.** El único de los tres lenguajes que da una formalización de su sintaxis es GRL.
- **Descripción semántica.** Existe una descripción semántica incompleta de *i** en Telos [13]. En el caso de Tropos existe una definición rigurosa en UML [12] de su modelo, meta-modelo y meta-metamodelo [17].

	Criterio	*i** de Yu	GRL	TROPOS
Estructurales	Tipos de modelos	SD y SR	Sin tipos	Sin tipos (vistas)
	Tipos de actores	1 genérico 3 específicos: rol, cargo y agente	1 genérico	1 genérico 3 específicos: rol, cargo y agente
	Elementos intencionales	Objetivo, requisito no fun., tarea, recurso	Objetivo, requisito no fun., tarea, recurso	Objetivo, requisito no func., tarea, recurso
	Relaciones entre actores	Dependencias entre actores mediante elementos intencionales Relaciones entre tipos específicos de actores: "occupies", "covers" y "plays" Relación "is part of" entre actores	Dependencias entre actores mediante elementos intencionales	Dependencias entre actores mediante elementos intencionales Relaciones entre tipos específicos de actores: "occupies", "covers" y "plays"
	Relaciones entre elementos intencionales (v. tabla 2)	Dependencias entre actores Relación medio-fin Relación de descomposición Relación de contribución	Dependencias entre actores Relación medio-fin Relación de descomposición Relación de contribución	Dependencias entre actores Relación medio-fin Relación de descomposición Relación de contribución
	Niveles de desagregación	De actores, ilimitado	De actores, restringido	De actores, ilimitado
	Elementos de razonamiento adicionales	Explícitos: Strength, Contribution, Constraints Razonamiento dinámico: Routine, Rule, Belief Propiedades: Workability, Ability, Viability, Believability	Explícitos: Belief, Contribution Types, Correlation Types, Evaluation Labels, Criticality	Explícitos: Belief, Contribution Types, Mode Comportamiento dinámico: Capability, Events
	Relaciones con elementos externos al modelo	No existen	Atributos Elementos de modelos externos Tópico de un softgoal	No existen
No estructurales	Etapas del ciclo de vida cubiertas	Early requirements Late requirements	Early requirements	Todas (de Early requirements a Implementation)
	Diseminación y estandarización	Amplia consolidación Diversidad de fuentes de información	Poca consolidación Adopción como estándar por la ITU-T en progreso	Consolidación emergente Existencia de una guía de usuario (en italiano)
	Herramientas	OME, REDEPEND	OME	Ninguna
	Acompañamiento del lenguaje	Ejemplos de casos de aplicación en diversos dominios	Lenguaje funcional complementario (UCM) Metodología de uso complementaria en el contexto de URN	Metodología de Ingeniería de Software Formal Tropos (con herramienta FTT)
	Descripción sintáctica	Lenguaje natural, notación gráfica	BNF Textual, BNF Gráfica, XML	Lenguaje Natural, Notación Gráfica
	Descripción semántica	Lenguaje natural, Telos	Lenguaje Natural	Lenguaje Natural, UML (modelo, metamodelo, meta-metamodelo)

Tabla 1: Análisis comparativo de las tres variantes principales de *i**

	i de Yu*	**GRL**	**Tropos**
Medio-fin	*means-end*	*means-end*	*means-end*
	* – T	(T, O, R) – T	(O, NF) – * T – (T, R)
	OR	OR	OR
Descomposición	*task-decomposition*	*decomposition*	*AND/OR decomposition*
	T – *	T – *	(G, NF) – (G, SG) T – T
	AND	AND	AND, OR
Contribución	*means-end*	*contribution, correlation*	*contribution*
	O – O NF – NF	*contribution*: (NF, C, L) – (NF, T, C, L) *correlation*: (NF, T) – (NF)	O, NF – *
	no existe	*contribution*: AND, OR *correlation*: no existe	no existe
	+, -	Make, Break, Help, Hurt, Some+, Some-, Equal, Unknown	++, +, -, --

Tabla 2: Análisis comparativo de las relaciones entre elementos intencionales en *i**

6. Una propuesta de modelo conceptual de referencia para *i**

En esta sección se formula un modelo conceptual con el objetivo de establecer un marco de referencia para el estudio de las diferentes variantes de *i** presentadas y para otras que puedan aparecer. Este modelo conceptual de referencia de *i** contiene los elementos que son iguales en las tres variantes de *i** estudiadas y, a partir de éstos, se pueden determinar las diferencias de cada variante respecto al modelo. Estas diferencias se describen mediante operaciones de transformación del modelo conceptual similares a las propuestas en [20] en el contexto de *refactoring*. De esta forma, se permite indicar las transformaciones necesarias para convertir el modelo conceptual de referencia en el modelo conceptual específico de una variante de *i**.

Para la construcción del modelo conceptual de referencia (ver figura 3) se ha tenido en cuenta todas las variantes descritas en este artículo. En él se intenta representar los conceptos comunes a todas las variantes, así cómo aquellos elementos que no son comunes pero que, por sus características, aportan grandes facilidades en el modelado orientado a objetivos.

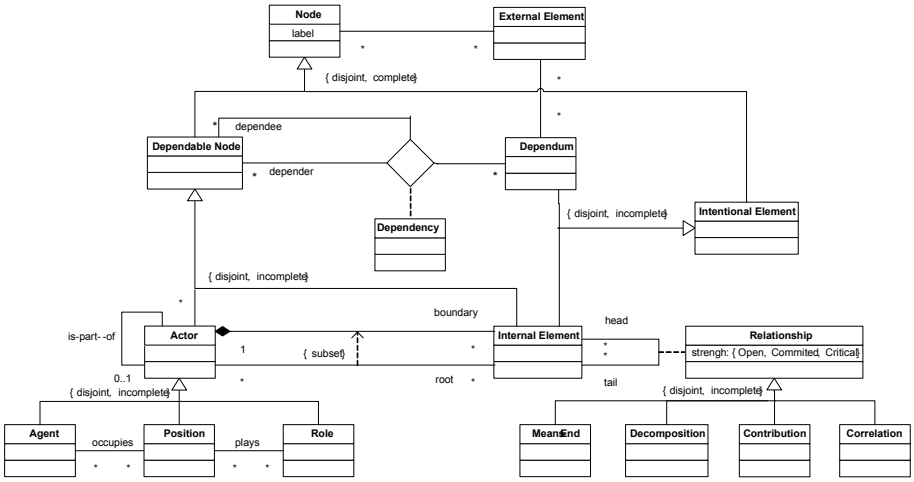

Figura 3. Modelo conceptual de referencia de *i**

En el caso particular de *i** de Yu [4], el modelo conceptual específico de la figura 2 puede obtenerse a partir del modelo conceptual de referencia aplicando las siguientes operaciones[1]:

- **Añadir.** Se añaden los los atributos *depender_strengh* y *dependee_strengh* en la classe *Dependency*. Se añaden las asociaciones derivadas */dependency_equivalence* (con sus nombres de roles correspondientes) y */plays*.
- **Suprimir.** Se suprime la clase *External Element* y sus relaciones con *Dependum* y *Node*.
- **Renombrar.** Se renombran las clases *Dependum* a *SR-Dependum* e *Internal Element a SR-Element*. También se da nombre al rol *has-parts* de *is-part-of*.
- La operación más compleja es la transformación de la clase asociativa *Relationship* del modelo de referencia en las relaciones *Means-End* y *Task-Decomposition* del modelo de *i** pues implica un colapso de la jerarquía. Para dicha transformación se aplican los pasos siguientes: se suprime las subclases *Contribution* y *Correlation*; se añade la subclase derivada *SR-Task-Element* a la clase *SR-Element* (con el correspondiente atributo *constraint*); se suprime la subclase *Decomposition* y se añade una clase asociativa *Task-Decomposition Link* entre las clases *SR-Element* y *SR-Task-Element* con el atributo *strength*; se generaliza la clase *Means-End* pasando a sustituir la clase *Relationship* y se renombra dicha clase como *Means-End Link*, añadiéndosele el atributo *contribution*.

7. Uso del modelo conceptual para el análisis de otras variantes

La existencia de un modelo conceptual de referencia también permite analizar y comparar con más facilidad nuevas propuestas de lenguajes basadas en *i**, nuevas

[1] Por simplicidad, se omiten las restricciones OCL que se expresarían mediante notas.

versiones de los lenguajes ya existentes, e incluso, extensiones puntuales en usos específicos del lenguaje *i** para dominios concretos.

Un primer ejemplo de estos casos podría ser la propuesta de *i** implementada en la herramienta REDEPEND [19] en que se extiende el *i** con nuevos tipos de relaciones *Means-End*, introduce la relación *Contribution* y otras extensiones menores. La mayor parte de estas extensiones están ya consideradas en GRL y Tropos y por tanto cubiertas por el modelo conceptual de referencia.

Un segundo ejemplo lo encontramos en el propio proyecto Tropos [10] que define el lenguaje Formal Tropos [18] con el objetivo de permitir la aplicación de técnicas de validación de modelos. Formal Tropos incorpora a *i** primitivas de especificación temporal, cuyos nuevos elementos se incorporan al lenguaje, tal y como se muestra en [21]. Por un lado se admite la especificación de restricciones de cardinalidad en las dependencias entre elementos intencionales. Por el otro, se define una nueva dependencia entre elementos intencionales (*prior-to*) para especificar un orden temporal entre estos elementos. Estas dos extensiones son fácilmente incorporables a nuestro marco conceptual de referencia, por un lado, definiendo estas multiplicidades como atributos de una dependencia y por el otro, definiendo un nuevo tipo de la clase asociativa *Dependency*.

En el área del modelado de procesos de organizaciones y de la ingeniería de requisitos hay trabajos utilizan la notación de *i**, pero con extensiones y/o adaptaciones según sus necesidades particulares. Por ejemplo, en el trabajo de [22] se analizan las interacciones entre un sistema software y los usuarios que lo utilizan. Los autores proponen nuevos tipos de dependencias entre actores y elementos intencionales: dependencias de responsabilidad entre un agente y un objetivo o tarea; dependencias de autoridad entre dos agentes; dependencias de auditoria entre un agente y un objetivo o tarea; y dependencias de habilidades de un agente respecto a un objetivo o tarea. Todas estas nuevas dependencias son fácilmente incorporables a nuestro modelo conceptual de referencia como nuevos subtipos de la clase asociativa *Dependency* con las restricciones de integridad oportunas.

8. Conclusiones

El propósito de este artículo ha sido analizar en profundidad las variantes más importantes de una de las propuestas más difundidas en la actualidad para el modelado orientado a objetivos, el lenguaje *i**. Las aportaciones más relevantes del artículo son las siguientes.

- La realización de un estudio comparativo de las tres propuestas más populares de modelado usando *i** o sus variantes. Este estudio se ha llevado a cabo de manera rigurosa mediante el uso de modelos conceptuales de datos en UML e identificando un total de 14 criterios de comparación.
- La enumeración de los ruidos, silencios, ambigüedades y contradicciones que existen en las definiciones disponibles de estas variantes.
- La definición de un modelo conceptual de referencia que engloba los conceptos presentes en las diferentes variantes estudiadas y que sirve para contextualizarlas.
- La constatación empírica de que el modelo de referencia permite también capturar otras variantes puntuales existentes en la literatura.

Existen algunos trabajos relacionados con el análisis comparativo, evaluación y crítica de los modelos orientados a objetivos [23, 24], pero no conocemos ninguno que esté enfocado a clarificar o guiar al usuario en torno a las confusiones de uso por la existencia de las diferentes variantes ni a formalizar el marco general de uso de una propuesta tan extendida como lo es *i**.

Agradecimientos

Este trabajo se ha realizado en el marco del proyecto de investigación TIC2001-2165 subvencionado por el Ministerio de Ciencia y Tecnología. Además, C. Ayala disfruta de una beca subvencionada por el Consejo Nacional de Ciencia y Tecnología (CONACyT) de México y G. Grau disfruta de una beca de investigación de la propia Universidad Politècnica de Catalunya.

References

[1] E. Yu. "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering". *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering, RE'97*. January 6-8, 1997, Washington, USA.

[2] E. Yu, J. Mylopoulos. "An actor dependency model of organizational work: with application to business process reengineering". *Proceedings of the Conference on Organizational Computing System, COOCS 1993*, November 1-4, 1993, Milpitas, California, USA.

[3] A. van Lamsweerde. "Goal-Oriented Requirements Engineering: A Guided Tour". *Proceedings of the 5th IEEE International Symposium on Requirements Engineering, RE´01*, Toronto, Canada.

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

[5] Pàgina web de *i**: http://www.cs.toronto.edu/km/istar/. Fecha de consulta: Julio 2004.

[6] D. Amyot, G. Mussbacher. "URN: Towards a New Standard for the Visual Description of Requirements". *Telecommunications and beyond: The Broader Applicability of SDL and MSC. Third International Workshop (SAM 2002)*. June 24-26, 2002, Aberystwyth, UK.

[7] Página web de GRL: http://www.cs.toronto.edu/km/GRL/. Fecha de consulta: Julio 2004.

[8] P. Bresciani, A. Perini , P. Giorgini , F. Giunchiglia , J. Mylopoulos. "Tropos: An Agent-Oriented Software Development Methodology". In *Journal of Autonomous Agents and Multi-Agent Systems*, Kluwer Academic Publishers Volume 8, Issue 3, Pages 203 - 236, May 2004.

[9] F. Sannicoló, A. Perini, and F. Giunchiglia. "The Tropos modeling language. a User Guide". *Technical report DIT-02-0061*, University of Trento, February 2002. (in Italian).

[10] Página web del proyecto TROPOS: http://www.troposproject.org. Fecha de consulta: Julio 2004.

[11] B. Meyer. "On Formalisms in Specifications". *IEEE Software*, 2(1), January 1985.

[12] *Unified Modelling Language (UML) 1.4 Specification*. OMG document formal/(formal/2001-09-67). September 2001.

[13] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis. "Telos: Representing Knowledge about Information Systems". *ACM Transactions Information Systems*, 8 (4). October, 1990.

[14] L. Chung, B.A. Nixon, E. Yu, J. Mylopoulos *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[15] Z.151 (GRL). International Telecommunication Union (ITU). September 2003.

[16] Página web de la herramienta OME3: http://www.cs.toronto.edu/km/ome. Fecha de consulta: Julio 2004.
[17] J. Mylopoulos, M. Kolp, J. Castro, "UML for Agent-Oriented Software Development: The Tropos Proposal". *Proceedings of 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools* , Toronto, Canada, October 1-5, 2001.
[18] Página web de Formal Tropos: http://dit.unitn.it/~ft/doc. Fecha de consulta: Julio 2004.
[19] N. Maiden, P. Pavan, A. Gizikis, O. Clause, H. Kim, X. Zhu. "Integrating Decision-Making Techniques into Requirements Engineering". *Proceedings of the eighth International Workshop on Requirements Engineering: Foundation for Software Quality*. September 09-10th, 2002, Essen Germany
[20] G. Sunyé, D. Pollet, Y. Le Traon, J.M. Jézéquel. "Refactoring UML Models". *Proceedings of the 4th International Conference <<UML>> 2001 – The Unified Modeling Language*, Toronto, Canada, 2001, LNCS 2185, pp. 134-148.
[21] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, P. Traverso. "Specifying and analizing early requirements in Tropos". *Requirements Engineering Journal, vol. 9, num. 2, 2004*, pp. 132-150.
[22] A. Sutcliffe, S. Minocha. "Linking Business Modelling to Socio-Technical System Design". *Proceedings of Advanced Information Systems Engineering, 11th International Conference CAiSE'99*, Heidelberg, Germany, June 14-18, 1999, pp. 73-87.
[23] E. Kavakli. "Modeling organizational goals: analysis of current methods". *Proceedings of the 2004 ACM symposium on Applied computing*. Pages: 1339 – 1343. ISBN:1-58113-812-1.
[24] C. Silva, R. Pinto, J. Castro, P. Tudesco. "Requirements for Multi-Agent Systems". *Proceedings of the Workshop em Engenharia de Requisitos*, Piracicaba-SP, Brasil, November 27-28, 2003. ISBN 8587-926071. pp 198-212.