UN ENTORNO PARA LA OBTENCIÓN DEL DIAGRAMA DE TAREAS A PARTIR DE ESQUEMAS PRECONCEPTUALES

AN ENVIRONMENT FOR AUTOMATICALLY OBTAINING THE TASK DIAGRAM FROM PRECONCEPTUAL SCHEMAS

[1]Carlos Mario Zapata Jaramillo, [2] Sebastian Osorio Vélez, [3] Roberto Manjarrés

[1] Ph.D en Ingeniería, Profesor Asociado de la Universidad Nacional de Colombia, cmzapata@unal.edu.co , [2] Estudiante de Maestría en Ingeniería de Sistemas la Universidad Nacional de Colombia, sosoriov@unal.edu.co [3] Profesor Asistente, Politécnico Colombiano Jaime Isaza Cadavid ramanjar@elpoli.edu.co

UN ENTORNO PARA LA OBTENCIÓN DEL DIAGRAMA DE TAREAS A PARTIR DE ESQUEMAS PRECONCEPTUALES

AN ENVIRONMENT FOR AUTOMATICALLY OBTAINING THE TASK DIAGRAM FROM PRECONCEPTUAL SCHEMAS

Carlos Mario Zapata Jaramillo, Sebastian Osorio Vélez, Roberto Manjarrés

RESUMEN

El diagrama de tareas constituye el punto de partida de algunas metodologías para el desarrollo de aplicaciones Web. Con él se busca la integración de elementos característicos de estas aplicaciones en la fase de requisitos, tales como la navegación y la construcción de otros diagramas de esas metodologías. Actualmente, los diagramas de tareas se construyen en una etapa avanzada del ciclo de vida del software, sin la participación del interesado y de forma manual, lo que implica más tiempo y esfuerzo. También, existen proyectos que procuran construir los diagramas de tareas desde artefactos técnicos, lejanos a la comprensión y validación del interesado, o desde discursos en lenguaje natural, pero en dominios muy restringidos. En este artículo se propone la obtención automática del diagrama de tareas, a partir de la representación del dominio del problema mediante esquemas preconceptuales. Para ello, se define un entorno que incluye los metamodelos de ambos diagramas y las reglas de transformación en una herramienta metaCASE basada en el GMF (*Graphic Modelling Framework*) de *Eclipse*. El entorno se ejemplifica con un caso de estudio.

Palabras clave: Desarrollo Web, navegación, diagramas de tareas, metamodelo, esquemas preconceptuales.

Recibido 13 de abril de 2011
Aceptado 26 de Mayo de 2011

ABSTRACT

The task diagram is the starting point of some methodologies for the development of Web applications. By using such diagram, some Web application elements (i.e. navigation and other diagrams) are sought to be integrated into the requirements phase. Nowadays, task diagrams are manually built in an advanced stage of software development lifecycle and avoiding the stakeholder participation, in a time and effort consuming fashion. Also, some projects propose the construction of task models from either technical artifacts difficult to stakeholder understanding and validation or natural language discourses (but related to restricted domains). In this paper we propose to automatically obtain the task diagram from a preconceptual schema based representation of the problem domain. For completing this task, an enviroment is defined with both the task diagram and preconceptual schema metamodels. Also, we define transformation rules into a tool based on GMF metaCASE (*Graphic Modelling Framework*) Eclipse. The environment is exemplified by means of a case study.

Keywords: *Web development, navigation, task model, metamodel, pre-conceptual schemas.*

Received: April 13, 2011
Accepted: April 26, 2011

1. INTRODUCCIÓN

El diagrama de tareas es un elemento fundamental para algunas de las metodologías de desarrollo Web existentes. Con su uso se pretende complementar la toma de requisitos, permitiendo modelar no sólo elementos estructurales, sino también aspectos claves de las aplicaciones Web, como la navegación [1]. Además, proporciona un mecanismo de ayuda para la generación automática de prototipos Web.

Actualmente, la construcción del diagrama de tareas se hace de forma manual y sin la participación del interesado. Además, no existe un mecanismo definido para su construcción, lo cual deja abierta la posibilidad de incurrir en fallas a la hora de su construcción. En algunos proyectos se pretende la construcción de estos diagramas empleando como base diagramas orientados a objetos [2] o las denominadas "plantillas de interacción" [3], pero, en ambos casos, se trata de artefactos técnicos que los interesados poco comprenden y, mucho menos, los pueden validar. Existe una propuesta basada en discursos en lenguaje natural [4] pero se emplea sólo para el dominio de los datos médicos, lo cual no es la generalidad de los desarrollos de aplicaciones web. Otros proyectos se incluyen en una revisión sobre el tema [5].

Con el fin de solucionar parcialmente estos problemas, en este artículo se define un entorno para la construcción de esquemas preconceptuales [6] y la obtención automática del diagrama de tareas a partir de dicho esquema. Para ello, se definen los metamodelos de ambos diagramas y las reglas de transformación en una herramienta metaCASE basada en el GMF (*Graphic Modelling Framework*) de *Eclipse* y en MOFScript (*Meta-Object Facility Script*). Al utilizar los esquemas preconceptuales, se mejora la comunicación con el interesado, dado que estos son muy cercanos al lenguaje natural y no requieren capacitación previa para su comprensión. Además, los esquemas preconceptuales se pueden desarrollar para cualquier dominio de conocimiento.

Este artículo se organiza de la siguiente manera: en la Sección 2 se define el marco teórico que agrupa los conceptos de este dominio; en la Sección 3 se resumen algunas metodologías de desarrollo que hacen uso de los diagramas de tareas; en la Sección 4 se plantean los elementos de la propuesta de implementación basada en metamodelos; en la Sección 5 se plantea un caso de estudio para ejemplificar el uso de la herramienta. Las conclusiones y el trabajo futuro se incluyen en las secciones 6 y 7, respectivamente.

2. MARCO TEÓRICO

2.1. Esquemas preconceptuales

Son representaciones de la terminología de un dominio en forma gráfica, que facilita la traducción a diferentes esquemas conceptuales [6]. En la Figura 1 se muestran los elementos de los esquemas preconceptuales, los cuales se describen a continuación:

- Concepto: es un sustantivo o un sintagma nominal del discurso del interesado.
- Relación estructural: es una relación que asocia dos conceptos y permite el uso de dos verbos: "es" y "tiene".
- Relación dinámica: se asocia con los verbos de actividad.
- Condicional: es una relación de causalidad que indica las restricciones que los conceptos deben cumplir.
- Marco: es un límite que agrupa uno o varios elementos del esquema preconceptual [7].
- Implicación: sirve para unir relaciones dinámicas o para unir condicionales con relaciones dinámicas, estableciendo entre ellas una relación causa-efecto.
- Conexión: permite enlazar los conceptos con las relaciones y las relaciones con los conceptos.
- Especificación de proceso: sirven para especificar las operaciones de una relación dinámica. En su interior, se definen las secuencias del proceso de la relación dinámica indicada.

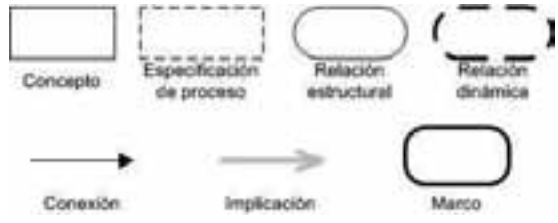


Figura 1. Elementos esquemas Preconceptuales [6].

2.2. Diagramas de tareas

Los diagramas de tareas constituyen alternativas para el

desarrollo de aplicaciones interactivas, ya que, a partir de su construcción, se pueden identificar elementos clave de diseño y de la interfaz gráfica de la aplicación [8].

En la construcción de software, los diagramas de tareas se usan para modelar requisitos de una forma más amigable, permitiendo que el interesado comprenda fácilmente la información y, de esta manera, pueda realizar una validación en fases tempranas del ciclo de vida del software. Además, en algunos casos son elementos de ayuda en la generación de las interfaces gráficas y la documentación del usuario final [2]. Algunas de las ventajas que trae consigo el uso del diagrama de tareas son:

- Los modelos son muy expresivos y de fácil comprensión [9].
- Los diagramas de tareas se pueden usar para disminuir la complejidad de algunas actividades, descomponiéndolas en otras más simples, con algún nivel de jerarquía [10].

La representación jerárquica de tareas facilita la concepción del problema, debido a que una tarea principal ("compleja") se descompone en otras tareas "más simples", lo que se conoce como tareas y subtareas. Este tipo de representación es la que más se utiliza en actividades de toma de requisitos [11].

2.2.1. Elementos del diagrama

El diagrama de tareas se compone de dos elementos principales: la tarea y las relaciones entre las tareas. Existen diversas formas de representarlos, que van desde tablas explicativas hasta objetos gráficos. Para este trabajo se emplea la notación de Paternò *et al.* [8].

En la Figura 2 se muestra la representación gráfica de los elementos principales del diagrama de tareas. En la parte izquierda de la imagen se encuentra una tarea, la cual se representa mediante un óvalo y un nombre representativo. En la parte derecha de la imagen se encuentra un conjunto de tareas representadas de forma jerárquica, desde la más general hasta la más específica.

2.2.2. Tipos de tareas

Para realizar una descomposición de cada una de las tareas con el fin de obtener una estructura jerárquica, se definen dos tipos de relaciones: estructural y temporal.

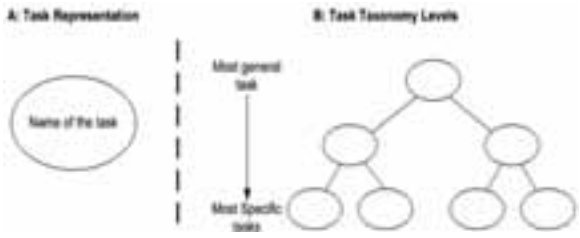


Figura 2. Representación gráfica del modelo de tareas [11].

La relación estructural, representada por medio de una línea sólida, se utiliza cuando cada una de las subtareas, asociadas con una tarea general, se puede realizar de forma independiente de las otras. Generalmente, se utiliza para representar actividades que realizan usuarios con diferentes roles en la aplicación.

Dentro de las relaciones temporales, se encuentran las relaciones múltiples y las relaciones unitarias. Paternò *et al.* [8] clasifican las relaciones múltiples en varios tipos, cada una de ellas con su propia representación:

- Habilitada (T1 >> T2): la tarea T2, se debe realizar siempre después de la tarea T1.
- Habilitada con información requerida (T1 [ ]>> T2): la tarea T2, se debe realizar siempre después de la tarea T1, pero es necesario el resultado de realizar la tarea T1 para ejecutar la tarea T2.
- Suspender-reanudar (T1 |> T2): la tarea T1, se puede interrumpir un número indefinido de veces para realizar la tarea T2. Una vez finalice dicha actividad, se retoma la tarea T1.
- Tareas independientes (T1 |=| T2): ambas tareas se pueden realizar en cualquier orden. Sin embargo, cuando una tarea inicie debe terminar antes de comenzar con la otra.
- Deshabilitada (T1 [> T2): la tarea T2 interrumpe completamente la tarea T1.

Además, clasificaron las relaciones unitarias en tres tipos:

- Tarea iterativa (*): tarea que se realiza varias veces.
- Tarea con iteración definida (T(n)): tarea que se realiza un número n veces.
- Tarea opcional (T[ ]): tarea cuyo desarrollo no es obligatorio.

Estos son los tipos de tareas que se utilizarán en el desarrollo de este artículo.

2.3. Metamodelado

Según De Lara y Vangheluwe [12], el metamodelado proporciona una serie de formalismos adecuados para describir los diferentes aspectos de los modelos. El OMG (*Object Management Group*) propone una arquitectura dividida en cuatro capas, con el fin de estandarizar la construcción de los modelos (véase la Figura 3): M3, M2, M1, M0. Se parte desde el punto más abstracto (M3) hasta el más concreto (M0). De este modo, cada uno de ellos se puede ver como una instancia de su antecesor. El nivel M3 representa la parte más abstracta de la cadena, el cual, en UML (*Unified Modeling Language*), se genera a partir del lenguaje MOF (*MetaObject Facility*), el cual permite definir modelos más concretos [13].

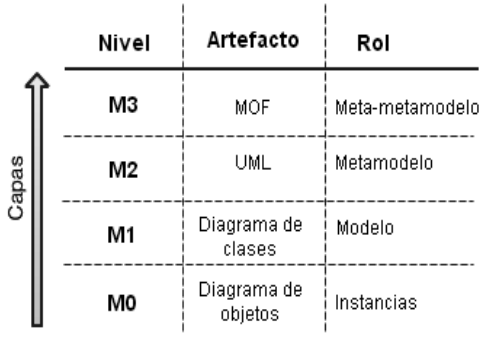


Figura 3. Arquitectura metamodelado [13].

2.4. EMF (Eclipse® Modeling Framework)

Es una estructura de modelado propia de *Eclipse*, la cual posee un lenguaje de metamodelado (*Ecore*) para definir metamodelos propios y utilizarlos en alguna herramienta específica [14].

2.5. GMF (Graphic Modeling Framework)

GMF provee una completa infraestructura de desarrollo para editores gráficos como GEF (*Graphical Editing Framework*). GEF es un *plug-in* de *Eclipse* que permite realizar de forma gráfica modelos mediante un conjunto amplio de herramientas que facilitan su desarrollo [14].

2.6. MOFScript

MOFScript es un *plug-in* de *Eclipse*, que permite la generación de texto a partir de metamodelos, mediante un lenguaje de programación de fácil uso [15].

3. ANTECEDENTES

Algunas de las metodologías Web actualmente existentes abarcan todas las etapas de desarrollo de software, desde la concepción (toma de requisitos) hasta la construcción. Sin embargo, muchas propuestas no tienen en cuenta características esenciales de toda aplicación Web, como la usabilidad, la navegabilidad y la accesibilidad. Actualmente, una de las metodologías que intenta integrar estas características durante el ciclo de vida de la aplicación es OOWS (*Object Oriented Web Solution* [16]). OOWS es una extensión del método de producción de software *OO-Method* [17], el cual introduce la captura de requisitos de navegación y presentación para las aplicaciones Web. Los esquemas conceptuales de *OO-Method* permiten la especificación de la estructura y el comportamiento de la aplicación, mediante el uso de tres modelos:

- Un modelo estructural, el cual define la estructura de clases y la relación entre ellas.
- Un modelo dinámico, el cual describe la validez de cada uno de los objetos.
- Un modelo funcional, el cual captura la semántica de los cambios de estado para definir sus efectos, utilizando una especificación formal.

La metodología OOWS [18] introduce dos nuevos modelos para abarcar la navegación y la definición de las interfaces de usuario:

- Modelo navegacional, el cual captura la estructura de navegación de la aplicación Web.
- Modelo de presentación, el cual define la presentación de los objetos, sus propiedades y la forma en que estos se deben mostrar en la aplicación.

Además, la toma de requisitos, en esta metodología, se hace mediante un discurso del interesado acompañado de diagramas de tareas, lo cual facilita la compresión del problema y sirve de base para la construcción del diagrama navegacional de la aplicación. Este proceso se hace de forma manual, lo que conlleva a que el analista debe emplear una gran cantidad de tiempo y esfuerzo en esta actividad.

Para la construcción del diagrama de tareas, Lu *et al.* [2] proponen el uso de diagramas orientados a objetos, los cuales se obtienen en fases muy avanzadas del ciclo de vida de una aplicación y, por lo general, se refieren a un lenguaje

técnico que los interesados poco entienden y difícilmente pueden validar. Algo similar ocurre con la propuesta de Paquette y Schneider [3], quienes emplean las denominadas "plantillas de interacción", que son diagramas y tablas que sólo se pueden construir en una fase muy avanzada del ciclo de vida del desarrollo de una aplicación. Existe una propuesta para la obtención del diagrama de tareas desde discursos en lenguaje natural [4], pero se desarrolló, únicamente, para el dominio de los datos médicos y requiere para su utilización una alta participación de los analistas y de expertos del dominio. Paris *et al.* [5] muestran un conjunto de propuestas que exhiben los mismos problemas definidos. En la construcción de otros diagramas, los esquemas preconceptuales [6] demuestran ser útiles, pues, debido a la simplicidad de su sintaxis, se pueden construir en las fases iniciales del ciclo de vida del software y, además, las pueden validar los interesados.

4. SOLUCIÓN PROPUESTA

Para una solución parcial del problema anteriormente descrito, se propone la obtención automática del diagrama de tareas a partir de un entorno gráfico basado en el metamodelo de los esquemas preconceptuales.

4.1. Metamodelo esquema preconceptual

La Figura 4 muestra el diagrama de clases que representa el metamodelo de los esquemas preconceptuales, en el cual, además de los elementos básicos, se añaden las capas de diseño y de proceso.

4.2. Metamodelo diagrama de tareas

La Figura 5, muestra el diagrama de clases que representa el metamodelo de un diagrama general de tareas.

El diagrama se compone de varios elementos, que se describen a continuación:

- El principal elemento es la clase tarea. Existen dos tipos de tareas: tarea atómica y tarea compuesta. Una tarea atómica se refiere a aquellas tareas que no requieren descomposición y son las de más baja jerarquía dentro del diagrama. Por otro lado, las tareas compuestas son aquellas que requieren descomposición en dos o más subtareas. Su descomposición puede ser de tipo temporal o estructural.

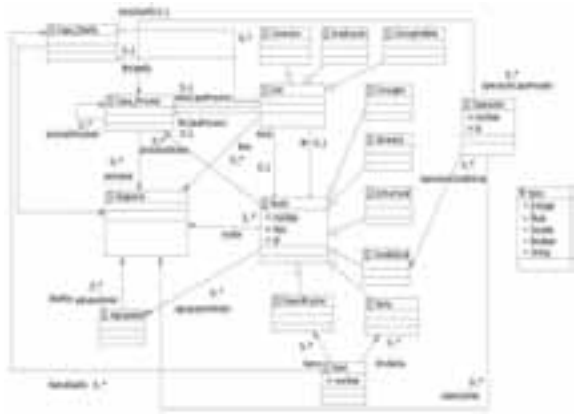


Figura 4. Metamodelo del esquema preconceptual, representado mediante el diagrama de clases.

Figura 5. Metamodelo del diagrama de tareas, representado mediante un diagrama de clases. Elaboración propia de los autores.

- Otro elemento fundamental es la clase enlace, la cual determina el tipo de tarea que se genera. Las tareas se identifican con el símbolo que tiene la relación que las conecta.

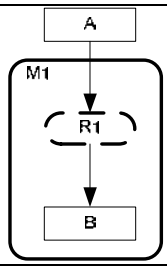
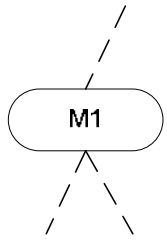
- La descomposición de las tareas se puede hacer mediante una relación de tipo temporal. Ésta, a su vez, posee dos clases, una unitaria y una múltiple.

4.3. Reglas de transformación

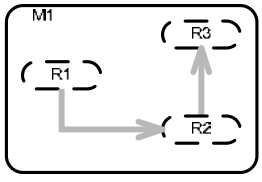
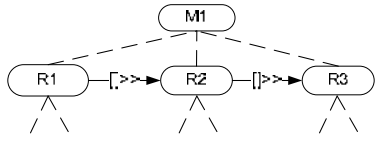
En la tabla 1 se propone un conjunto de reglas para la conversión entre esquemas preconceptuales y sus equivalencias representadas en los diagramas de tareas.

Tabla 1. Reglas de transformación (parte 1/2).

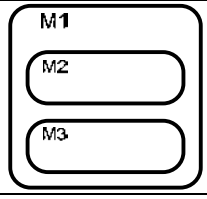
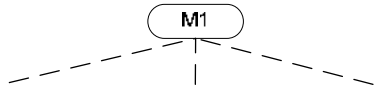
Regla 1. Un marco de nombre M1, el cual agrupa una o más relaciones dinámicas (R1), se transforma en una tarea de nivel intermedio de nombre M1, que tendrá una o más subtareas.

Precondición	Código	Resultado
	*self.nodos->forEach(ag: modelo.Agrupacion)* *{agrupaciones.put(ag.id, ag.nombre)}* . . . *var nombreAgrupacion:String =* *agrupaciones.get(id)* *mapTarea(nombreAgrupacion)*	

Regla 2. Un marco de nombre M1, que agrupa un número n relaciones dinámicas (R1, R2, R3) conectadas mediante implicaciones, se transforma en una tarea de nivel intermedio de nombre M1. Ésta se divide en un número n de subtareas (R1, R2, R3), a las cuales se asociará el nombre de cada relación dinámica implicada. Dichas subtareas son de tipo "habilitada con información requerida" y, a su vez, tendrán otras subtareas asociadas.

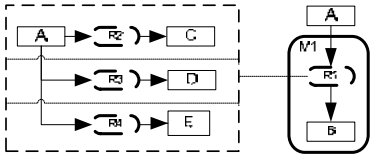
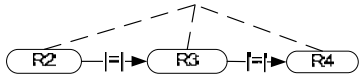
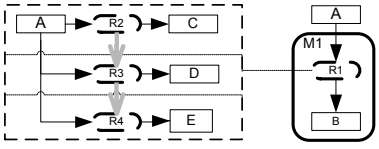
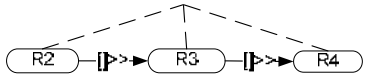
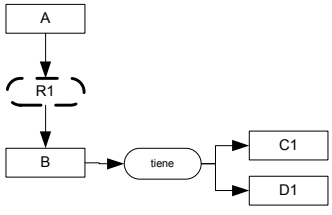
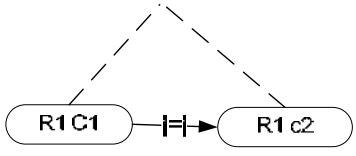
Precondición	Código	Resultado
	*self.nodosAgrupacion->forEach(rd:* *modelo.Dinamica)* *{ dinamicasA.put(rd.id, rd.nombre) }* . . *var nombreDin: String = dinamicasA.get(id)* *var nombreTarea: String = nombreDin + " "* *+ nombreConcepto*	

Regla 3. Un marco de nombre M1, que agrupa uno o más marcos (M2, M3), se transforma en la tarea principal del diagrama de tareas, la cual corresponde a la tarea con mayor jerarquía dentro del diagrama. El nombre de dicha tarea, será el nombre asociado con el marco más general dentro del esquema preconceptual.

Precondición	Código	Resultado
	*var alds:list = agrupaciones.keys()* *alds->forEach(ald)* *{ var marcos:list = elementosA.get(ald)* *marcos->forEach(value)* *{mapMarcos(ald,value)}* *}.* .	

Regla 4 (Opción 1). Una relación dinámica asociada con una especificación de proceso, se transforma en una sucesión de tareas independientes, las cuales serán las de mas bajo nivel dentro del diagrama, por lo tanto, éstas no tienen subtareas asociadas. El orden en que se generan las tareas dentro del diagrama, es el mismo en el que se define la especificación de proceso en el esquema preconceptual.

Tabla 1. Reglas de transformación (parte 2/2).

Precondición	Código	Resultado
	*self.nodosEspecificacion->forEach(es: modelo.Especificacion)* *{ especificaciones.put(es.id, es.nombre) }* *.* *.var nombreDin: String = especificaciones.get(id)* *var nombreTarea: String = nombreDin + " " + nombreConcepto* *mapIndependientes(id,nombreTarea)*	
Regla 4 (Opción 2). Una relación dinámica asociada con una especificación de proceso, en la cual se encuentre una sucesión de relaciones dinámicas conectadas mediante implicaciones, se transforma en una sucesión de tareas de tipo "habilitadas con información requerida", las cuales serán las de mas bajo nivel dentro del diagrama, por lo tanto, éstas no tienen subtareas asociadas. El orden en que se generan las tareas dentro del diagrama, es el mismo en el que se define la especificación de proceso en el esquema preconceptual.		
Precondición	Código	Resultado
	*self.nodosEspecificacion->forEach(es: modelo.Especificacion)* *{ especificaciones.put(es.id, es.nombre) }* *.* *.var nombreDin: String = especificaciones.get(id)* *var nombreTarea: String = nombreDin + " " + nombreConcepto* *mapHabilitada(id,nombreTarea)*	
Regla 5. Una relación dinámica aplicada sobre un concepto que tiene uno o más atributos, se transforma en una o más tareas de tipo independiente. Dichas tareas serán del nivel más bajo del diagrama y su nombre se forma con la unión entre la relación dinámica y el atributo asociado al concepto afectado. Generalmente, estas relaciones se encuentran encerradas en un marco, el cual determina el padre de las subtareas resultantes. El número de tareas resultante es igual al número de atributos que posea el concepto sobre el cual se realiza la acción.		
Precondición	Código	Resultado
	*if((self.inicio.oclIsTypeOf(modelo.Estructural) && nombreIni = "tiene") || (self.fin.oclIsTypeOf(modelo.Estructural) && nombreFin = "tiene"))* *{value.add(self.fin.id)* *conexionesE.put(self.inicio.id,value) }* *.* *.* *.* *dinamicas.get(id)* *mapTarea(id, conexionesE.nombre)*	

5. CASO DE ESTUDIO

Con el fin de ejemplificar las reglas de transformación definidas en la sección anterior, se propone un caso de estudio, que se basa en la construcción de una aplicación Web a partir del discurso de un interesado, el cual desea tener un sistema de reserva de hotel. Realizada la entrevista con el interesado, se logró extraer el siguiente fragmento, en donde se expresan los requisitos fundamentales del sistema a desarrollar:

La aplicación Web a desarrollar es un sistema de reserva de hotel. El objetivo principal de la aplicación es proveer una herramienta que permita realizar reservas en línea a cualquiera de los destinos disponibles de la compañía. Para esto, el usuario debe poder consultar los destinos disponibles y realizar búsquedas de acuerdo con sus necesidades para cada uno de ellos. Una vez el usuario se encuentre conforme con los resultados de la búsqueda, tendrá la opción de realizar su reserva, por medio de un formulario en el cual ingresa su información y realiza el pago según el método seleccionado. En la Figura 6 se muestra el esquema preconceptual que representa el discurso anterior.

Aplicando las reglas de transformación definidas anteriormente, se obtiene el diagrama de tareas, como lo muestra la Figura 7, equivalente al esquema preconceptual presentado anteriormente.

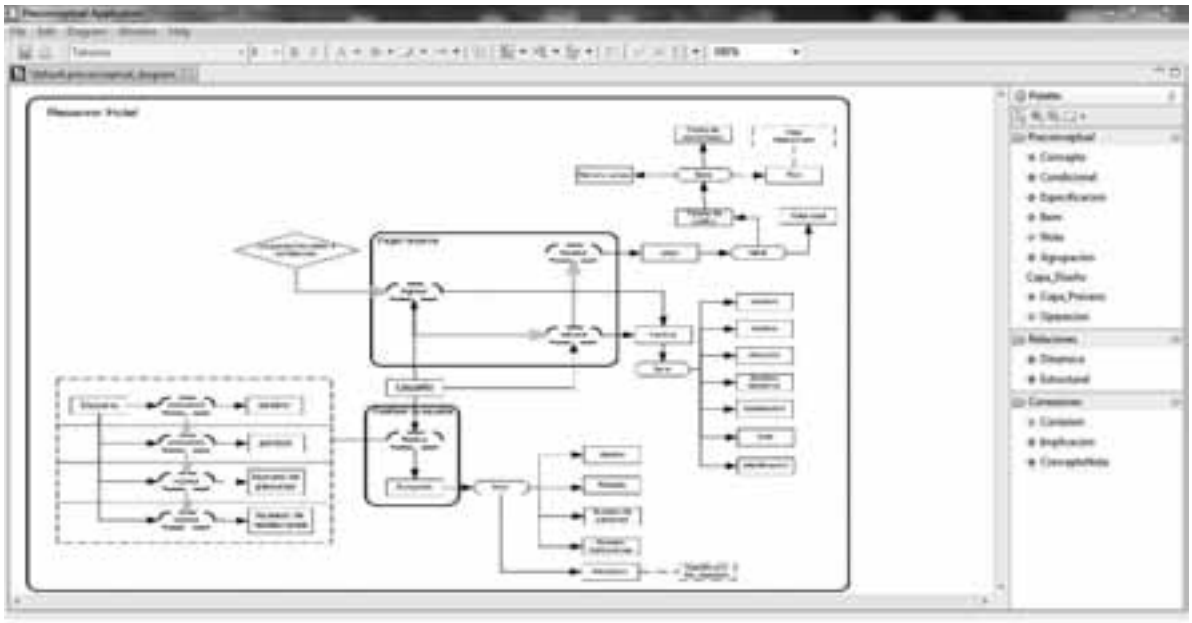


Figura 6. Esquema preconceptual del sistema de reserva de hotel. Elaboración propia de los autores.

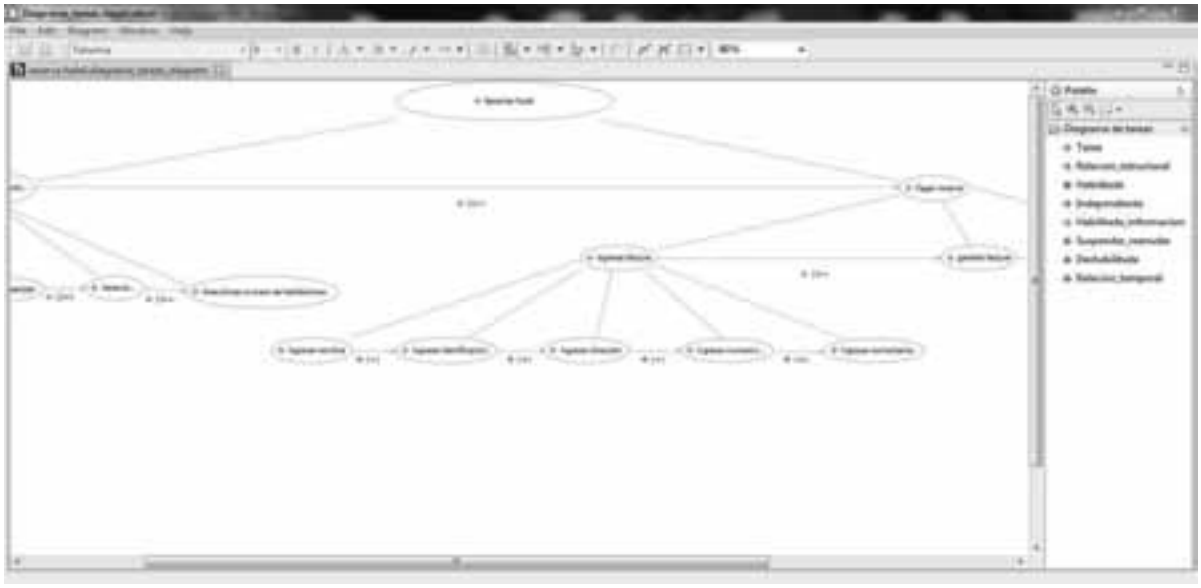


Figura 7. Diagrama de tareas del sistema de reserva de hotel, descrito anteriormente. Elaboración propia de los autores.

6. CONCLUSIONES

En este artículo se definió un conjunto de reglas heurísticas que permiten la obtención automática del diagrama de tareas, a partir de esquemas preconceptuales. Los principales aportes de este artículo son:

- Reducción del tiempo de desarrollo, lo cual permite una disminución de los costos.
- Automatización en el proceso de toma de requisitos.
- Completitud de la toma de requisitos con la generación de los diagramas de tareas.
- Obtención de diagramas a partir del diagrama de tareas de una forma más rápida.
- Simulación de las posibles interfaces de una aplicación Web desde etapas tempranas de su desarrollo.

7. TRABAJO FUTURO

Las líneas de trabajo futuro en esta área contemplan:

- La inclusión de nuevas reglas que permitan obtener, automáticamente, los otros tipos de tareas a partir de los esquemas preconceptuales.
- La definición de un mecanismo que permita determinar el orden de los procesos dentro del mismo esquema.
- La consolidación de una nueva metodología Web que propugne por la automatización del proceso completo de desarrollo.

8. AGRADECIMIENTOS

Los autores desean expresar su gratitud al Ingeniero Jhon Jairo Chaverra quien aportó el metamodelo de los esquemas preconceptuales para realizar este artículo.

9. REFERENCIAS

[1] Lauesen, S., Task descriptions as functional requirements, IEEE., 20(2),58-65, 2003.

[2] Lu, S., Paris, C., Vander Linden, K. Toward the Automatic Construction of Task Models from Object-Oriented Diagrams. En: Proceedings of VII Working Conference on Engineering for Human-Computer Interaction. Heraklion, Grecia, 169-188, septiembre 1998.

[3] Paquette, D. y Schneider, A. Task Model Simulation Using Interaction Templates. Lecture Notes in Computer Science, Vol. 3941, 78-89, 2006.

[4] Chung-Man Tam, R., Maulsby, D. y Puerta, A. U-Tel: a tool for eliciting user task models from domain experts. In Proceedings of the 1998 International Conference on Intelligent User Interfaces. San Francisco, CA: Association for Computing Machinery. 77-80, 1998.

[5] Paris, C., Lu, S. y Vander Linden, K. Environments for the Construction and Use of Task Models. En The Handbook of Task Analysis for Human-Computer Interaction. Dan Diaper & Neville Stanton (Eds). Lawrence Erlbaum Associates, 467-482.

[6] Zapata, C. M., Gelbukh, A. y Arango, F. Pre-conceptual Schemas: a Conceptual-Graph-like Knowledge Representation for Requirements Elicitation. Lecture Notes in Computer Sciences, Vol. 4293, 17-27, 2006.

[7] Zapata, C. M., Manjarrés, R. y González, G., Representación Gráfica de Restricciones Lógicas y Aritméticas en Esquemas Preconceptuales. Memorias del CISCI 2010, Orlando, EEUU, Julio de 2010.

[8] Paternò, F. Mancini, C y Meniconi, S. ConcurTaskTree: a Diagrammatic Notation for Specifying Task Models. In Proceedings: INTERACT, 362-369. 1997.

[9] Johnson, P. Tasks and situations: considerations for models and design principles in human computer interaction. In Proceedings. of HCI International, 1199–1204. 1999.

[10] Card, S.K., Moran, T.P. y Newell, A., The Psychology of Human Computer Interaction, Lawrence Erlbaum Associates, Hillsdale, 1983.

[11] Valderas, P. A requirements engineering approach for the development of web applications [PhD Thesis]. Camino

de Vera, Valencia: Universidad Politécnica de Valencia, 2007.

[12] De Lara, J. y Vangheluwe, H. Using Meta-Modelling and Graph Grammars to Create Modelling Environments, Electronic Notes in Theoretical Computer Science, Vol 72, 36-50, 2003.

[13] Mora, B., García, F., Ruiz, F., Piattini, M., Boronat, A. y Gómez, A. Marco de Trabajo basado en MDA para la Medición Genérica del Software. Documento presentado en la XII Jornada de Ingeniería del Software y Bases de Datos, Zaragoza, España, septiembre de 2007.

[14] Moore, B., Dean, D., Gerber, A., Wangenknecht G. y Vanderheyden, P. Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework, IBM International Technical Support Organization, 2004

[15] Oldevik, J. MOFScript Eclipse Plug-In: Metamodel-Based Code Generation, En: Eclipse Technology Workshop (EtX) at ECOOP. 2006

[16] Rojas, G. Modelling adaptive web applications in OOWS [PhD Thesis]. Camino de Vera, Valencia: Universidad Politécnica de Valencia, 2008.

[17] Pastor, O., Fons, J., Pelechano, V. y Abrahão, S. Conceptual modelling of Web applications: the OOWS approach, En: Web Engineering: Theory and Practice of Metrics and Measurement for Web Development (Eds. E. Mendes y N. Mosley), Springer-Verlag, reading 277-301, 2005.

[18] Fons, J., Pelechano, V., Albert, M. y Pastor, O. Development of Web Applications from Web Enhanced Conceptual Schemas. Lecture Notes in Computer Science, Vol 2813, 232-245, 2003.