Modelado de Requisitos de Sistemas Colaborativos con CSRML

Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero, Pascual González

Grupo de Investigación LoUISE
Departamento de Sistemas Informáticos
Universidad de Castilla - La Mancha
{MiguelAngel.Teruel, Elena.Navarro, VictorManuel.Lopez, Francisco.MSimarro, Pascual.Gonzalez}@uclm.es

Resumen. Un sistema colaborativo es una aplicación software que permite a varios usuarios llevar a cabo tareas de colaboración, comunicación y coordinación. Para realizar dichas tareas, los usuarios tienen que ser conscientes de las acciones de los demás usuarios, usualmente por medio de una serie de técnicas de *awareness*. En trabajos anteriores determinamos, por medio de estudios empíricos, que las técnicas de Ingeniería de Requisitos más adecuadas para especificar los requisitos de este tipo de sistemas son las aproximaciones *Goal-Oriented*, y más concretamente, la aproximación *i**. En este artículo se presenta CSRML (*Collaborative Systems Requirements Modelling Language*): una extensión de *i** para abordar la especificación de requisitos de sistemas en los que la colaboración y la consciencia de la presencia y/o las acciones de los otros usuarios es crucial. Para validar nuestra propuesta, se ha utilizado un caso de estudio en el que se modela un sistema de gestión de congresos en el que la revisión se realiza colaborativamente entre los distintos revisores.

Keywords: Sistemas Colaborativos, awareness, Ingeniería de Requisitos, Goal-Oriented, *i**, CSRML

1 Introducción

La elicitación de requisitos es una de las primeras fases en el Proceso de Desarrollo Software [1,2]. Si esta fase falla en la captura adecuada de los requisitos de la aplicación, es muy probable que el resto del proceso falle también, causando importantes costes en tiempo y dinero. Por lo tanto, la generación de un modelo de requisitos preciso es una cuestión muy importante para cualquier tipo de sistema

Los Sistemas Colaborativos (sistemas CSCW, *Computer Supported Cooperative Work*) no están exentos de esta necesidad. Estos sistemas son un tipo especial de aplicación cuyos usuarios pueden realizar tareas de colaboración, comunicación y coordinación. A diferencia de los sistemas convencionales monousuario, los sistemas CSCW tienen que ser especificados usando un conjunto especial de requisitos, usualmente de naturaleza no funcional, los cuales suelen ser resultantes de la

necesidad de los usuarios de ser conscientes de la presencia y actividad de otros usuarios, tanto remotos como locales, con los que realizar las anteriormente mencionadas tareas colaborativas, o sea, el denominado *Workspace Awareness* [3].

Workspace Awareness (WA) puede definirse como la comprensión en el momento de la interacción de otras personas en un espacio de trabajo compartido. WA implica conocimiento acerca de *dónde* están trabajando los demás, *qué* están haciendo *ahora*, y *qué* van a hacer *a continuación*. Gutwin et al. [3] presentaron un marco conceptual para establecer qué tipo de información constituye el WA. Esta información se obtiene respondiendo las preguntas "quién, qué y dónde". Así, cuando varios usuarios trabajan conjuntamente en un espacio físico compartido, saben con quién están trabajando, qué están haciendo, dónde están trabajando y cuándo y cómo se producen determinados eventos.

En este contexto, una especificación adecuada del sistema, identificando claramente los requisitos del sistema, en especial los relacionados con el *awareness*, es una de las primeras etapas para solventar este problema. Los requisitos de awareness pueden ser considerados requisitos no funcionales (*Non-Functional Requirements*, NFR) o requisitos extra-funcionales (*Extra-Functional Requirements*, EFR) debido a que suelen ser requisitos ligados a la calidad no explícitamente funcional de un producto software, como por ejemplo, a la usabilidad [4].

Podemos definir nuestra metodología de investigación de la siguiente manera: En un trabajo anterior [5], analizamos basándonos en las líneas guía establecidas en DESMET [6], las técnicas Goal-Oriented (GO) [7], Casos de Uso [8,9] y Viewpoints [10] con el objetivo de comprobar cuál era la más apropiada para especificar los requisitos de los sistemas colaborativos. El análisis concluyó que el enfoque GO proporcionaba más facilidades para modelar los requisitos de este tipo de sistemas. Una vez determinamos que GO era la técnica más adecuada, analizamos qué aproximación de GO trataba mejor los requisitos de los sistemas CSCW [11]. Los enfoques analizados fueron *NFR Framework* [12], *i* Framework* [13] y *KAOS Methodology* [7], prestando especial atención a los requisitos de *awareness*. Para llevar a cabo este estudio, especificamos los requisitos de *awareness* de un sistema real (*Google Docs*) [14]. Tras modelar el sistema, se realizó un estudio empírico para comparar las técnicas. Como resultado del experimento, concluimos que los enfoques GO analizados no son completamente apropiados para modelar las características de los sistemas colaborativos, debido a su falta de expresividad en cuanto a mecanismos de colaboración entre usuarios se refiere, así como a la representación de los requisitos de *awareness* y a la no del todo adecuada gestión de actores y roles para sistemas colaborativos.

Estas conclusiones, junto con los resultados de [5], apoyan nuestra hipótesis inicial: una técnica de Ingeniería de Requisitos (IR) es necesaria para solventar los problemas detectados durante el estudio. Esta técnica debería adoptar algunas características de los enfoques GO estudiados y cubrir la falta de expresividad en ciertos aspectos que presentan las actuales técnicas GO, como son la especificación de requisitos no funcionales y la representación del *awareness*. Éste constituye el objetivo principal de este trabajo: adaptar/extender una notación GO para este tipo de sistemas. Concretamente, y de acuerdo con las conclusiones de nuestro estudio previo

[11], el enfoque más apropiado para tratar este tipo de sistemas es *i**. Por lo tanto, en este artículo se describe la notación CSRML (*Collaborative Systems Requirements Modelling Language*), la cual extiende la notación de *i** con el fin de ofrecer la expresividad necesaria para modelar las características especiales de los requisitos de los sistemas CSCW. Además, se presentarán y aplicarán un conjunto de heurísticas y restricciones a la notación *i** original con el objetivo de mejorar la facilidad de uso del lenguaje gráfico que proponemos y la comprensibilidad de los modelos creados mediante CSRML.

Este artículo se estructura de la siguiente manera: después de esta introducción, en la Sección 2, ofrecemos una descripción de la técnica GO, centrándonos especialmente en el enfoque *i**. En la Sección 3 presentamos CSRML, nuestra propuesta para tratar con los requisitos de los sistemas CSCW. En la Sección 4, presentamos nuestro caso de estudio: un sistema de gestión de congresos con aspectos de colaboración que será modelado en la Sección 5 usando CSRML. Finalmente, en la Sección 6 recogemos nuestras conclusiones y trabajos futuros.

2 Trabajos relacionados: Técnicas *Goal-Oriented* y el enfoque *i**

Esta sección proporciona una introducción a las técnicas GO, centrándose en el enfoque i*, el cual constituye la base principal de nuestra propuesta.

2.1 Ingeniería de Requisitos *Goal-Oriented*

En el contexto de la IR, el enfoque GO ha demostrado ser útil para elicitar y definir requisitos. Las técnicas más tradicionales, como los Casos de Uso [8], solo se centran en el establecimiento de las características (por ejemplo, actividades y entidades) que el sistema debería soportar. No obstante, las propuestas GO se centran en porqué los sistemas se construyen, proporcionando la motivación y la lógica para justificar la especificación de los requisitos del software. Por otra parte, las propuestas GO no son sólo útiles para analizar objetivos (*goals*[1]), sino también para elaborarlos y refinarlos.

Un modelo GO puede ser especificado en varios formatos, usando una notación definida más o menos formalmente. Se construye como un grafo dirigido por medio de un refinamiento de los objetivos del sistema. Este refinamiento acaba cuando los objetivos tienen la suficiente granularidad y detalle para ser asignados a un agente (software o entorno). Entonces, son verificables en el sistema a construir. Este proceso de refinamiento es realizado usando relaciones de refinamiento AND/OR/XOR. Hay un gran número de propuestas que van desde las actividades de elicitación hasta las de validación dentro del proceso de IR (véase [15] para un estudio exhaustivo). Sin embargo, algunos conceptos son comunes a todas ellas:

- Un *objetivo* (goal) describe el porqué un sistema está siendo o ha sido desarrollado, desde el punto de vista del negocio, la organización, o el sistema propiamente

[1] En lo sucesivo, se utilizará la denominación inglesa para facilitar su correspondencia con otras propuestas GO y con *i** en particular.

dicho. Para especificar el sistema, deben determinarse tanto los objetivos funcionales (*functional goals*) concernientes con los servicios que se esperan del sistema, como los *softgoals*, relacionados con la calidad de servicio, restricciones en el diseño, etc.

- Un *agente* (agent) es cualquier componente activo, ya sea del propio sistema o del entorno, cuya cooperación es necesaria para definir la operacionalización de un objetivo, es decir, la forma en la que el objetivo va a ser proporcionado por el sistema. Esta operacionalización de los objetivos es aprovechada para mantener la trazabilidad a través del Proceso de Desarrollo Software.
- *Relaciones de refinamiento* (*refinement relationships*): Son relaciones AND/OR/XOR que permiten la construcción del modelo de objetivos como un grafo dirigido. Estas relaciones se aplican por medio de un proceso de refinamiento (desde objetivos genéricos hasta subobjetivos) hasta que los objetivos tienen la suficiente granularidad para ser asignados a una operacionalización específica.

Hay que tener en cuenta que una de las principales ventajas de esta aproximación es que introduce mecanismos para razonar acerca de la especificación. Esto facilita el proceso de evaluación de diseños o especificaciones alternativas del sistema [16].

2.2 *i* Framework*

El framework *i** [2,13] hace distinción entre dos tipos de elementos: objetos y relaciones. Los objetos (Fig. 1) considerados en *i** son:

- Un *actor* es una persona o sistema que tiene una relación con el sistema a desarrollar [2]. *i** refina este actor en tres tipos:
  - Un *agente* (*agent*) es un actor que tiene una representación física concreta, por ejemplo una persona o un sistema.
  - Un *rol* (*role*) define el comportamiento de un actor dentro de un contexto específico. Un actor puede tener varios roles, y un rol puede ser asignado a múltiples actores.
  - Una *posición* (*position*) es un conjunto de roles que pueden ser representados típicamente por un agente. Un agente puede representar varias posiciones.
- Un *objetivo* (*goal*) responde a "¿por qué?". Describe un cierto estado que un actor desearía alcanzar. No obstante, un objetivo no describe cómo debería ser alcanzado.
- Una *tarea* (*task*) especifica una forma particular de hacer algo. Normalmente, una tarea consiste en un número de pasos (o sub-tareas) que un actor debería realizar para ejecutar dicha tarea.
- Un *recurso* (*resource*) es una entidad (física o de información) que un actor necesita para conseguir un objetivo o realizar una tarea. El mayor interés sobre un recurso es si está disponible y para quién lo está.
- Un *softgoal* es una condición que a un actor le gustaría conseguir, aunque al contrario que para el concepto de objetivo (*hardgoal*), la condición para conseguirlo no está claramente definida.

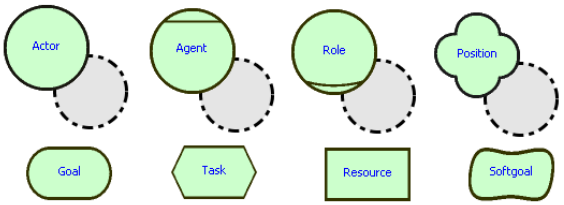

Fig. 1. Objetos en *i**

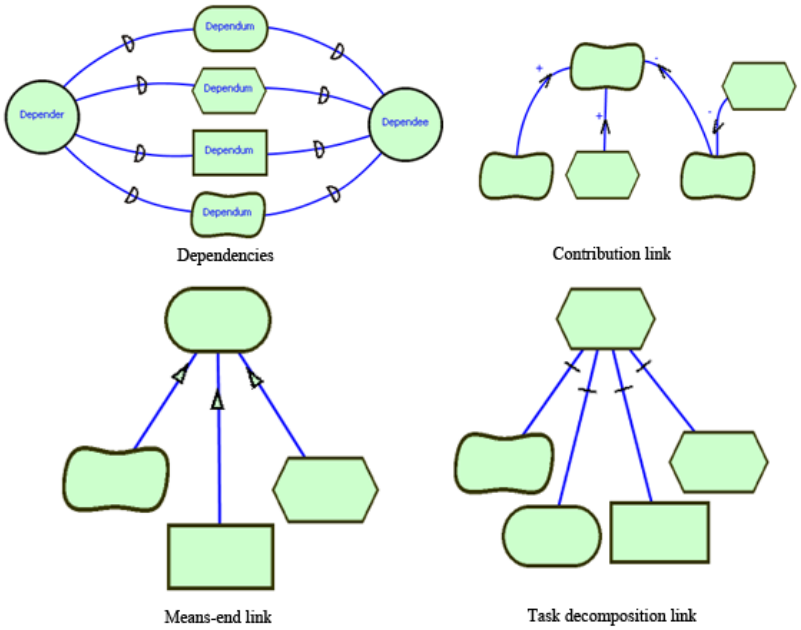

Fig. 2. Relaciones en *i**

Los objetos anteriores pueden relacionarse entre sí mediante el siguiente conjunto de relaciones (Fig. 2):

- Una *dependencia* (*dependency*) en *i** documenta una relación entre dos actores. Un actor depende de otro para conseguir un objetivo, realizar una tarea o usar un recurso. *i** distingue cuatro tipos de dependencias: de objetivo, tarea, recurso o *softgoal*.
- Un *means-end link* documenta qué *softgoals*, tareas y/o recursos contribuyen a conseguir un objetivo. Estos enlaces facilitan la documentación y la evaluación de las distintas formas de satisfacer un objetivo, por ejemplo, realizando varias descomposiciones de un objetivo en distintos *softgoals*, tareas y recursos.
- Un *enlace de descomposición de tareas* documenta los elementos esenciales de una tarea. Este enlace relaciona una tarea con sus componentes, los cuales pueden ser

cualquier combinación de subobjetivos, sub-tareas, recursos o *softgoals*. La descomposición de una tarea abarca las sub-tareas que pueden realizarse, los sub-objetivos que deberían cumplirse, los recursos que se necesitan y los softgoals que típicamente definen objetivos de calidad para la tarea.
- Una *contribución* (*contribution*) documenta una influencia positiva o negativa de ciertos *softgoals* sobre otros *softgoals* o tareas.

3 CSRML: un lenguaje de modelado de requisitos para sistemas colaborativos

Para tratar con el tipo especial de requisitos de los sistemas colaborativos, y basándonos en los dos estudios anteriormente mencionados [5,11], se ha desarrollado CSRML (*Collaborative Systems Requirements Modelling Language*). Este lenguaje es una extensión de *i** que incluye algunos elementos para modelar las características especiales de los sistemas colaborativos que puede ser aplicado a cualquier tipo de dominio que implique colaboración entre usuarios. Los elementos de CSRML, excluyendo aquellos cuyo significado es el mismo que en *i**, son:

- Un *rol* (*role*) representa el conjunto de tareas que pueden ser realizadas por el actor que asuma dicho rol. La diferencia entre *i** y CSRML es que un actor que representa un rol puede participar en tareas individuales y colaborativas (mediante enlaces de participación) y puede ser responsable de la consecución de un objetivo (a través de enlaces de responsabilidad). Además, la notación gráfica también varía respecto al rol de *i**. Finalmente, CSRML no comparte el concepto de límite (*boundary*) de actor/rol de *i**, a fin de facilitar la descripción de tareas colaborativas.
- Un *actor* es un usuario, programa o entidad con ciertas capacidades adquiridas que puede interpretar un rol responsable de ciertas acciones [17]. Un actor tiene que representar un rol (lo cual se especifica mediante un enlace de representación, ver Fig. 3) para participar en el sistema.
- Una *tarea* (*task*) en CSRML tiene el mismo significado que en *i**. No obstante, se diferencia en la notación introducida para definir la importancia de una tarea: uno, dos o tres signos de exclamación, dependiendo de la importancia de la tarea. Además, CSRML identifica dos tipos especiales de tareas:
  - Tarea abstracta (*abstract task*): Este tipo de tarea es una abstracción de un conjunto de tareas concretas y, posiblemente, de otros elementos, como objetivos, recursos o *softgoals*. No pueden asignarse enlaces de participación directamente a este tipo de tareas.
  - Tarea concreta: Estas son las tareas en las que están involucrados los participantes, los cuales serán asignados a estas tareas mediante los enlaces de participación. Las tareas abstractas se refinan en estas tareas. A su vez, se subdividen en cuatro tipos: (i) *tareas individuales*, que los actores pueden realizar sin ningún tipo de interacción con otros actores; (ii) *tareas de*

colaboración ; (iii) *tareas de comunicación*; y (iv) *tareas de* coordinación. Los últimos tres tipos responden a tareas en las que dos o más actores participan.

- Un *softgoal de awareness* (*awareness softgoal*) es una especialización del concepto de *softgoal* en *i**, que representa una necesidad especial de percepción de la presencia y/o acciones de otros usuarios, sin la cual, la tarea que el usuario desea realizar se vería negativamente afectada, o incluso no podría realizarse.
- Un *recurso de awareness* (*awareness resource*) es un tipo especial de recurso correspondiente a una implementación o solución de diseño para lograr en *softgoal* de *awareness*.
- Un *enlace de representación* (*playing link*) sirve para representar cuándo un actor asume un rol. Este enlace tiene una condición de guarda, que establece la condición que debe cumplirse para que el actor interprete el rol.
- Un *enlace de participación* (*participation link*) indica quién está involucrado en una tarea. Este enlace tiene un atributo para especificar su cardinalidad, es decir, el número de usuarios que pueden estar involucrados en dicha tarea.
- Un *enlace de responsabilidad* (*responsability link*) asigna un rol (interpretado por un actor) a un objetivo, *softgoal* o tarea. Este enlace representa quién es el *stakeholder* responsable del cumplimiento de una tarea u objetivo. No es necesario que un *stakeholder* esté involucrado en las sub-tareas del objetivo. No obstante, si el rol es responsable de una tarea u objetivo, este rol es también responsable de los elementos en los que éstos se dividen, a menos que un nuevo enlace de responsabilidad llegue a uno de estos elementos.

La representación gráfica de los elementos anteriormente mencionados puede verse en la siguiente figura (Fig. 3):

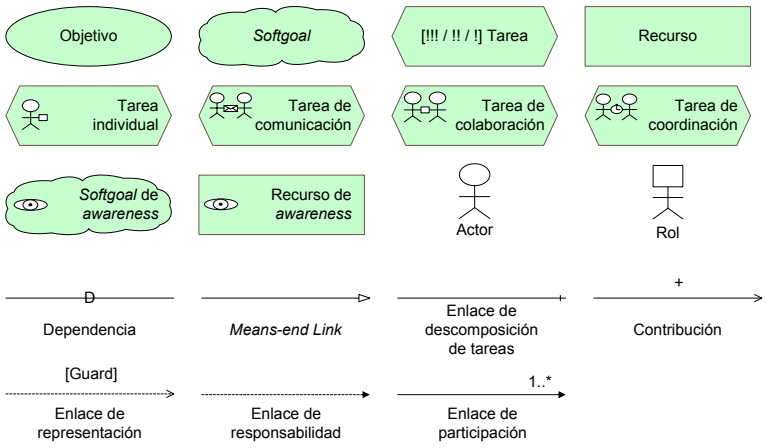

Fig. 3. Elementos de CSRML

La razón para introducir nuevos elementos a la notación original fue que, a pesar de que *i** tenía suficientes elementos expresivos, estos adolecían de una sobrecarga expresiva importante, sobre todo cuando se aplica a sistemas colaborativos.

Otra diferencia entre CSRML e *i** es que el primero es prácticamente jerárquico (ver Fig. 4 a Fig. 10), favoreciendo de esta manera la escalabilidad del modelo creado mediante el uso de esta notación. En un primer nivel, tenemos el diagrama de objetivos del sistema que permite definir los objetivos principales que el sistema debería alcanzar. A continuación, aparece el diagrama de responsabilidades, en el cual, el diagrama anterior se descompone en las tareas principales. Además, en este diagrama se definen las responsabilidades de los objetivos y las tareas.

En un tercer nivel aparecen los diagramas de refinamiento de tareas, en los que las tareas principales del sistema se descomponen en nuevos objetivos, *softgoals*, tareas y recursos. Debido a que CSRML ha sido pensado para su uso en sistemas colaborativos, los límites de los actores/roles de *i** han sido descartados, usándose en su defecto enlaces de participación en estos diagramas. Además, el diagrama de factores de calidad completa la especificación del sistema mostrando los *softgoals* de calidad y los elementos que contribuyen a su cumplimiento.

En la Sección 5 pueden apreciarse algunas líneas guía aplicadas directamente sobre el caso de estudio definido en la Sección 4. No obstante, consideramos como trabajo futuro la definición de una metodología que defina formalmente las líneas guía para modelar sistemas colaborativos con CSRML.

4 Caso de estudio: Sistema de gestión de congresos con revisiones colaborativas

Para comprobar cuán adecuado es CSRML para el modelado de requisitos de sistemas colaborativos, se ha recurrido a un caso de estudio clásico: un sistema de gestión de congresos. En nuestro caso, a diferencia de la gestión de congresos original, se dotará al sistema de aspectos de colaboración entre usuarios. Nuestro sistema constará de cuatro objetivos principales:

1. Creación del congreso: El *chair* del congreso dará de alta el congreso en el sistema y pre-registrará a los miembros del comité de programa (CP), los cuales habrán de confirmar ese pre-registro.
2. Envío de artículos: El autor principal de cada artículo se deberá registrar en el congreso y a su vez, pre-registrará a los co-autores que, como en el caso de los miembros de CP, deberán confirmar el pre-registro. Tras ello se podrá enviar (o re-enviar) los artículos.
3. Asignación de artículos a revisores: El *chair* del CP realizará una reunión virtual en la que se irán proponiendo los artículos a los miembros del CP, los cuales elegirán qué artículo desean revisar. Esta tarea concluirá cuando cada artículo sea asignado a tres revisores.
4. Revisión de artículos: Cada artículo será revisado colaborativamente por tres revisores mediante una aplicación web. Deberá saberse en cada momento que párrafo está siendo revisado, y por quién, así como los comentarios efectuados sobre cada uno de ellos. Tras la revisión se comunicará el resultado a los autores.

Además de estas tareas, el sistema deberá adecuarse a ciertos factores de calidad del estándar SQuaRE [18], como la eficiencia, fiabilidad, operabilidad, o seguridad.

5 Modelando el sistema con CSRML

En esta sección se modela el caso de estudio descrito anteriormente, haciendo uso de la notación CSRML con el fin de ilustrar su capacidad expresiva en el modelado de los requisitos de los sistemas colaborativos. Para empezar, en la Fig. 4, se muestra el *diagrama de objetivos del sistema*, en el que se define el objetivo global que el sistema debería alcanzar. Como se muestra, este objetivo se logrará por medio de la tarea principal: preparación de un congreso usando técnicas de colaboración entre usuarios.

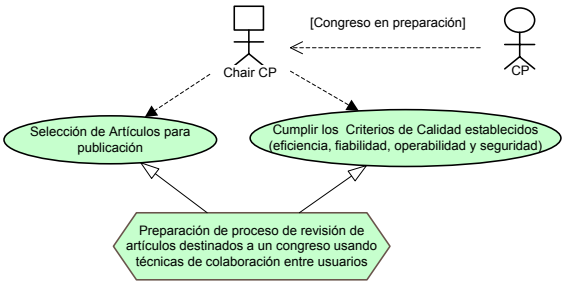

Fig. 4. Diagrama de objetivos del sistema

La Fig. 5 muestra el *diagrama de responsabilidades* con la tarea principal del sistema y su descomposición en *softgoals* de calidad y sub-tareas. En este diagrama se muestra quién es el responsable de cada tarea u objetivo mediante el uso de *enlaces de responsabilidad*. Nótese que si un rol es responsable de un objetivo o tarea, también es responsable de los elementos en que se divide, a menos que un nuevo enlace de responsabilidad llegue a uno de los elementos en los que se divide. Por ejemplo, el CP*chair* es responsable de la preparación del congreso (tarea principal) y de la asignación de artículos (sub-tarea), pero no del envío de los artículos, tarea de la cual es responsable el autor principal. Además, también se muestra el uso de los *enlaces de representación* utilizados para indicar la condición que debe cumplirse para que un actor interprete un rol. Para aumentar la legibilidad del modelo, la descomposición de las sub-tareas identificadas se realiza en las figuras mostradas a continuación (figuras desde la Fig. 6 a la Fig. 10).

La Fig. 6 representa el *diagrama de refinamiento* de la tarea de creación del congreso. En esta figura, las tareas se refinan en otras más específicas y en nuevos objetivos hasta que se especifiquen tareas individuales o colaborativas (colaboración, coordinación y comunicación).

A continuación, la Fig. 7 presenta la descomposición de la tarea de envío de artículos en la que se ilustran los cuatro grados de prioridad que pueden asignarse a las tareas: normal, alta ([!]) , muy alta ([!!]) y la más alta ([!!!]).

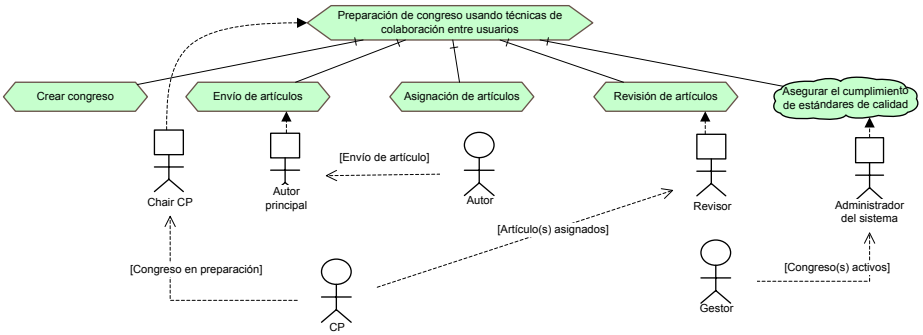

Fig. 5. Diagrama de responsabilidades

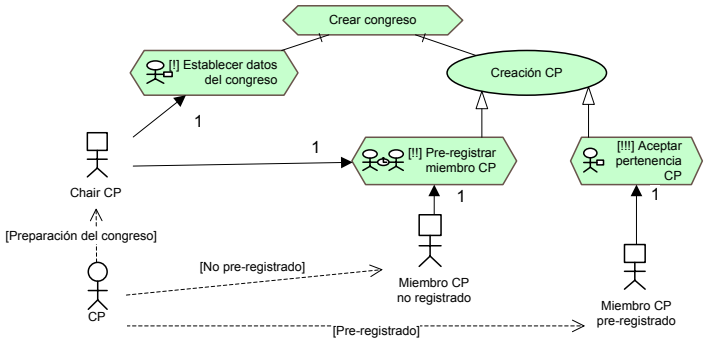

Fig. 6. Diagrama de refinamiento de la tarea *Crear congreso*

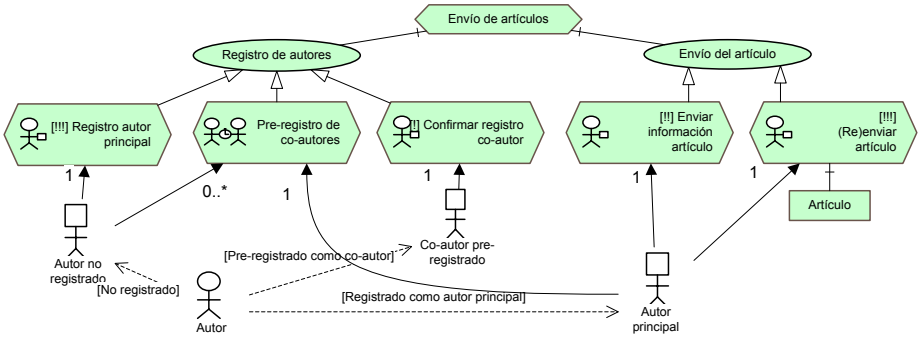

Fig. 7. Diagrama de refinamiento de la tarea *Envío de artículos*

La Fig. 8 ilustra el refinamiento de la tarea de asignación de artículos. En esta figura se usan diferentes cardinalidades para los *enlaces de participación*. Por

ejemplo, para la tarea de proposición de artículos al CP en la que participa el CP *chair* y tres o más miembros del CP. Esta última cardinalidad se especifica como *3..**. Además, en este diagrama se especifica un *softgoal* de *awareness* llamado *Ser consciente del resto de miembros y de sus artículos asignados*, relacionado con la tarea principal del diagrama. Este *softgoal* se incluye porque para poder asignar artículos, se debe ser consciente de quién está participando en la reunión, así como de los artículos asignados a cada miembro del CP. Para lograrlo, se introduce el recurso de *awareness* llamado *Lista de participantes con estado de las asignaciones*.

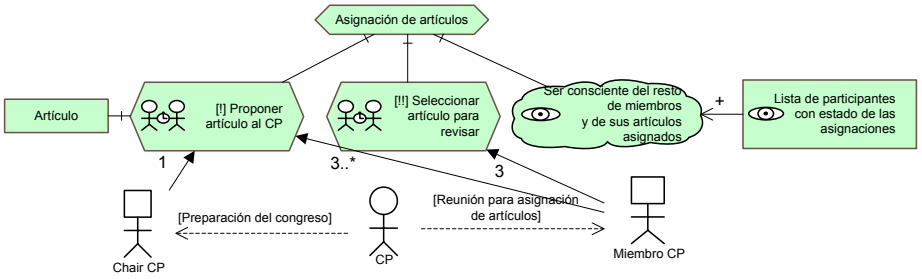

Fig. 8. Diagrama de refinamiento de la tarea *Asignación de artículos*

La especificación de las tareas principales del sistema concluye con el refinamiento de la tarea de revisión de artículos (Fig. 9). En esta figura se incluyen dos nuevos *softgoals* de *awareness* que harán posible el seguimiento de las revisiones de los demás revisores y el trabajo colaborativo gracias al uso de tele-punteros [19].

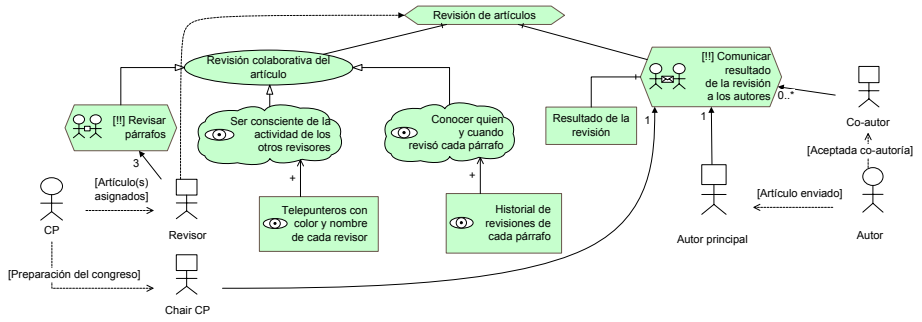

Fig. 9. Diagrama de refinamiento de la tarea *Revisión de artículos*

Finalmente, se describe el *diagrama de factores de calidad* que permite presentar los factores de calidad del estándar SQuaRE [18] que deberán cumplirse por la aplicación que contribuya a lograr el objetivo principal: la Selección de Artículos para publicación de acuerdo a los Criterios de Calidad establecidos. Estos factores son representados como *softgoals* y se relacionan con el principal *softgoal* de calidad mediante *enlaces de contribución* con contribuciones positivas, aunque en algún caso son negativas o indeterminadas. Por ejemplo, el *softgoal* relacionado con los

telepunteros, *Ser consciente de la actividad de los otros revisores*, contribuye positivamente a la utilidad del sistema, pero negativamente a la utilización de recursos, ya que hará posible el conocimiento en tiempo real del trabajo colaborativo del resto de revisores, pero aumentará el uso de la infraestructura de red debido al constante trasiego de información sobre el posicionamiento del cursor de los colaboradores.

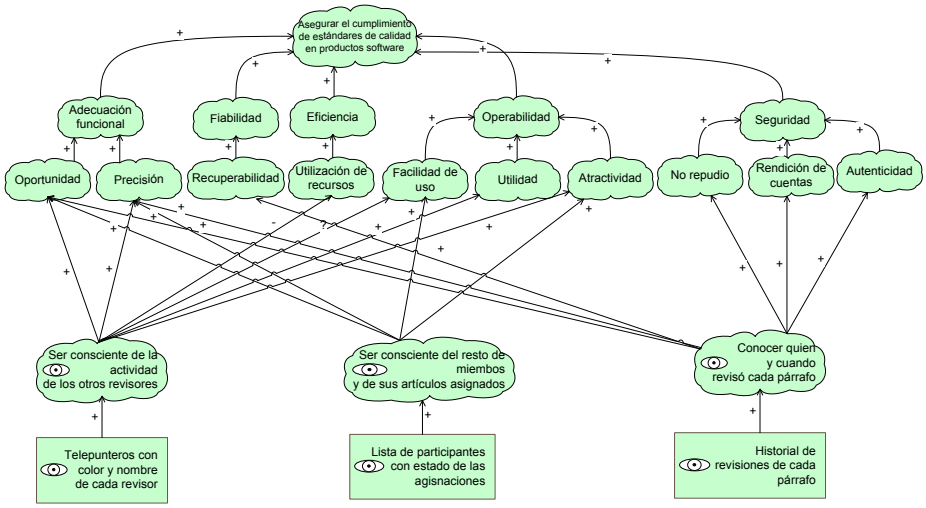

Fig. 10. Diagrama de factores de calidad

Con este último diagrama quedaría definido el sistema completo. Tal y como se permite apreciar CSRML permite la especificación de sistemas colaborativos, cubriendo las deficiencias encontradas en *i** para este tipo de sistemas.

6 Conclusiones y trabajos futuros

En trabajos anteriores [5,11], concluimos que las técnicas GO (y en especial *i**) pueden usarse para modelar requisitos de sistemas colaborativos. No obstante, también resaltamos que este tipo de especificaciones adolecen de una falta de expresividad respecto a ciertas características relacionadas con la colaboración entre usuarios, la representación del *awareness* y los factores de calidad. Para solucionar estos problemas, proponemos CSRML como una extensión de *i** para modelar los requisitos de los sistemas colaborativos. De este modo, CSRML solventa la representación de la colaboración entre usuarios con el uso de los enlaces de participación y la distinción de varios tipos de tareas colaborativas. A su vez, La representación del *awareness* es posible con esta notación gracias a los nuevos elementos de *awareness* incorporados. Además, la complejidad y falta de legibilidad de los diagramas *i** queda mitigada mediante la división jerárquica del modelo en

varios diagramas en los que los modelos tienden a descomponerse en forma de árbol, permitiendo una mayor legibilidad y extensibilidad.

Aplicaremos este lenguaje al modelado de un sistema colaborativo basado en un sistema de gestión de congresos con revisiones colaborativas. Este caso de estudio fue elegido debido a que poseía un conjunto de características que serían difíciles o incluso imposibles de representar usando la notación *i** original, como la asignación y la revisión colaborativa de artículos. Estas características fueron adecuadamente descritas en los modelos añadiendo un conjunto de nuevos elementos y relaciones a la notación *i**. La representación de la calidad y el *awareness* fue posible por medio de nuevos elementos de *awareness* y por la inclusión de un conjunto de nuevos diagramas que dotan de cierta estructura a la especificación.

En resumen, CSRML ayuda a mejorar los modelos de requisitos de sistemas colaborativos permitiendo considerar nuevos elementos y relaciones a *i**. Estos nuevos elementos facilitan la especificación de los requisitos de *awareness*, los cuales son un factor clave en el desarrollo de los sistemas colaborativos.

Actualmente, se encuentran en desarrollo diferentes trabajos relacionados con CSRML. Uno de ellos está relacionado con patrones para sistemas colaborativos. Gracias al trabajo desarrollado por el grupo de investigación LoUISE en este tipo de sistemas [20] en los últimos años, se han documentado una serie de patrones relacionados con la interacción para la colaboración. Sin embargo, uno de los principales problemas que se plantean para su utilización es que estos patrones han sido especificados de manera informal, de manera que no pueden ser fácilmente reutilizados para la especificación de diversos sistemas. Debido a ello, estamos estudiando cómo puede utilizarse CSRML para mejorar la especificación de dichos patrones.

Además, estamos trabando en la definición de un conjunto de líneas guía formales sobre el uso de nuestra propuesta, así como evaluando la comprensibilidad de los modelos CSRML mediante una familia de experimentos empíricos, cuyos resultados están siendo analizados.

Agradecimientos

Este trabajo ha sido parciamente financiado por el proyecto PEII09-0054-9581 de la Junta de Comunidades de Castilla-La Mancha y por el proyecto DESACO (TIN2008-06596-C02-01) del Ministerio de Ciencia e Innovación.

Referencias

[1] R.S. Pressman, *Software engineering: a practitioners approach*, McGraw-Hill Science/Engineering/Math, 2009.

[2] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, Springer, 2010.

[3] C. Gutwin and S. Greenberg, "A Descriptive Framework of Workspace Awareness for Real-Time Groupware," *Computer Supported Cooperative Work*, vol. 11, 2002, pp. 411-446.
[4] H. Hochmuller, "Towards the Proper Integration of Extra-Functional Requirements," *Australasian Journal of Information Systems*, vol. 6, 1999, pp. 98-117.
[5] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, and P. González, "An Empirical Evaluation of Requirement Engineering Techniques for Collaborative Systems," *15th International Conference on Evaluation and Assessment in Software Engineering*, Durham, UK: 2011.
[6] B. Kitchenham, "A methodology for evaluating software engineering methods and tools," *Experimental Software Engineering Issues: Critical Assessment and Future Directions*, H. Rombach, V. Basili, and R. Selby, eds., Springer Berlin / Heidelberg, 1993, pp. 121-124.
[7] A. van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour," *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, IEEE Computer Society, 2001, pp. 249-263.
[8] A. Cockburn, *Writting Effective Use Cases*, Addison-Wesley, 2000.
[9] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, 2005.
[10] A. Finkelsetin, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke, "Viewpoints: A Framework for Integrating Multiple Perspectives in System Development," *International Journal of Software Engineering and Knowledge Engineering*, vol. 2, 1992, pp. 31-57.
[11] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, and P. González, "A Comparative of Goal-Oriented Approaches to Modelling Requirements for Collaborative Systems," *6th International Conference on Evaluation of Novel Software Approaches to Software Engineering*, Beijing, China: 2011.
[12] L.M. Cysneiros and E. Yu, *Non-Functional Requirements Elicitation (Perspectives on Software Requirements)*, Springer, 2003.
[13] J. Castro, M. Kolp, and J. Mylopoulos, "A requirements-driven development methodology," *In Proc. of the 13th Int. Conf. On Advanced Information Systems Engineering (CAiSE'01)*, 2001, pp. 108-123.
[14] Google, "Google Docs," 2001.
[15] E. Kavakli and P. Loucopoulos, "Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods," *Information Modeling Methods and Methodologies*, 2004, pp. 102-124.
[16] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *No Non-Functional Requirements in Software EngineeringTitle*, Kluwer Academic Publishers, 1999.
[17] M. Noguera, M. González, J.L. Garrido, V. Hurtado, and M. Rodríguez, "System Modeling for Systematic Development of Groupware Applications," *International Conference on Software Engineering Research and Practice*, 2006, pp. 750-756.
[18] ISO/IEC JTC1/SC7, "Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) Quality model," 2008.
[19] C. Gutwin, S. Greenberg, and M. Roseman, "Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation," *Proceedings of HCI on People and Computers XI*, Springer-Verlag, 1996, pp. 281-298.
[20] H. Fardoun, F. Montero, and V. López-Jaquero, "eLearniXML: Towards a model-based approach for the development of e-Learning systems considering quality," *Advances in Engineering Software*, vol. 40, 2009, pp. 1297-1305.