CSRML Tool: una Herramienta para el Modelado de Requisitos de Sistemas Colaborativos

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

LoUISE, Instituto de Investigación en Informática, Universidad de Castilla - La Mancha
{miguel, enavarro, victor, fmontero, pgonzalez}@dsi.uclm.es

Resumen. Cada vez más aplicaciones incluyen algún tipo de soporte a la realización de tareas colaborativas. Como para cualquier otro tipo de sistema, una especificación de requisitos precisa es uno de los pilares básicos en el desarrollo de este tipo de sistemas con soporte a la colaboración. Sin embargo, este tipo de sistemas poseen un tipo especial de requisitos difícil de especificar con las técnicas de Ingeniería de Requisitos (IR) tradicionales. Estudios experimentales previos muestran que la especificación del conocimiento o la percepción de una situación o hecho (*awareness*), necesarios para que los usuarios puedan llevar a cabo tareas colaborativas, puede ser excesivamente compleja, o incluso incompleta, cuando se usan la técnicas clásicas de IR. Para solventar este problema, se ha desarrollado CSRML (*Collaborative Systems Requirements Modeling Language*), una extensión del lenguaje orientado a objetivos *i** para especificar requisitos de sistemas colaborativos. En este trabajo se presenta CSRML Tool: una herramienta que da soporte al lenguaje CSRML. Gracias a esta herramienta, se podrán modelar los distintos diagramas que conforman la especificación de los requisitos de un sistema colaborativo utilizando CSRML, así como comprobar la validez de los mismos con respecto al meta-modelo de CSRML.

Palabras clave: ingeniería de requisitos; sistemas colaborativos; workspace awareness; Visualization and Modeling SDK; Visual Studio; CSRML.

1 Introducción

Durante los últimos años, la forma en la que Internet proporciona servicios, aplicaciones, etc. ha cambiado notablemente. Hoy en día, la colaboración está en todas partes. De hecho, si observamos el ranking de las páginas web más visitadas, prácticamente las 100 primeras son webs colaborativas [1]. Redes sociales, editores de texto multiusuario, juegos online… todo tiende a ser colaborativo. Un buen ejemplo para entender cómo funciona la colaboración entre usuarios es Google Docs [2], una aplicación web cada vez más utilizada, que permite a varios usuarios editar un documento de texto de forma simultánea. Este tipo de herramienta es un interesante ejemplo de sistema CSCW (*Computer Supported Cooperative Work*) [3,

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 131-144, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

4]. Gracias a estos sistemas, junto a las funcionalidades que ya proporcionaban las aplicaciones software clásicas, los usuarios pueden realizar tareas relacionadas con colaboración, comunicación y coordinación, también conocidas como tareas *3C*. No obstante, el principal problema cuando se definen los requisitos de un sistema CSCW no es únicamente la especificación de esas tareas 3C. Existen otros requisitos, de naturaleza altamente no funcional, éstos surgen de la necesidad que tienen los usuarios de ser, por ejemplo, conscientes de la presencia o ausencia de otros usuarios con los que realizar las anteriormente mencionadas tareas 3C, siendo un ejemplo de ello el denominado *Workspace Awareness* (WA) [5]. WA incluye conocimiento acerca de, por ejemplo, *quién* hay disponible para colaborar, *qué* están haciendo (o hicieron en el pasado) el resto de usuarios, *cuándo* un artefacto fue modificado o *cómo* ocurrió alguna operación.

Estudios experimentales previos [6, 7] muestran que la especificación del conocimiento o la percepción de una situación o hecho (*awareness*), necesarios para que los usuarios puedan llevar a cabo tareas colaborativas, puede ser excesivamente compleja, o incluso incompleta, cuando se usan la técnicas clásicas de IR. Ello ha motivado la definición del lenguaje CSRML [8] como una extensión del lenguaje orientado a objetivos *i** [9, 10]. No obstante, CSRML no disponía de una herramienta de modelado, haciendo que la especificación de sistemas fuera compleja y sin soporte automático para verificar los modelos creados.

En el contexto de las técnicas orientadas a objetivos una de las herramientas más conocidas es OpenOME [11] que da soporte entre otras metodologías, a *i** [9, 10] en la que se basa CSRML (véase Sección 2). Ésta es una herramienta orientada a objetivos y a agentes que soporta tanto el modelado como el análisis de requisitos. Es de destacar que esta herramienta dispone de dos versiones: aplicación desktop y plug-in para Eclipse [12]. Podemos encontrar otras herramientas que dan soporte a *i** (o a otras metodologías orientadas a objetivos) como por ejemplo IStarTool [13], DesCARTES [14] o J-PRiM [15], pero hasta donde nosotros sabemos, ninguna de ellas ofrece la flexibilidad necesaria para incorporar los elementos expresivos de CSRML. Con el fin de solventar este problema surge la herramienta *CSRML Tool*. En este trabajo se presenta dicha herramienta que da soporte a CSRML permitiendo tanto el modelado de sistemas colaborativos a través de un conjunto de diagramas como la validación automática de los mismos.

Este artículo se estructura de la siguiente manera: después de esta introducción, en la sección 2 se presenta el lenguaje CSRML para, seguidamente, en la sección 3 presentar la herramienta que da soporte al mismo, CSRML Tool, mostrando detalles sobre su implementación e interfaz, así como describiendo su funcionamiento mediante un caso de estudio. Finalmente, la sección 4 documenta nuestras principales conclusiones e identifica trabajos futuros.

2 El Lenguaje CSRML

El lenguaje CSRML (*Collaborative Systems Requirements Modeling Language*) es un lenguaje de IR Goal-Oriented (GO) [16], basado en *i** [9, 10]. CSRML se propone con el objetivo de facilitar la especificación de sistemas colaborativos en los cuales la

definición de ciertos requisitos ligados a la colaboración entre usuarios es difícil (o incluso imposible) de representar con las técnicas clásicas de IR. Es importante señalar que la definición de CSRML, así como de la herramienta que da soporte al mismo, no pueden considerarse trabajos aislados, sino que han sido definidos por un proceso guiado por evaluaciones empíricas:

- En un trabajo inicial [6], se analizaron tres técnicas de IR (Use Cases [17, 18], Viewpoints [19] y Goal-Oriented [16]) con el fin de identificar cuál de ellas era la más adecuada para especificar los requisitos especialmente necesarios para la descripción de un sistema colaborativo, concluyendo que Goal-Oriented era la técnica más adecuada para este fin.
- A continuación, en [7], se realizó un análisis de algunos de los enfoques Goal-Oriented más conocidos (NFR Framework [20], KAOS [16] e *i** [9, 10]) para averiguar cuál de ellos podía ser aplicado en sistemas colaborativos de la forma más adecuada. A pesar de que ninguno de estos enfoques satisfizo todas las necesidades de expresividad requeridas, la notación *i** fue la mejor posicionada para la realización de esta tarea.
- El resultado anterior nos condujo a justificar la propuesta de CSRML [8, 21] como una extensión de *i** que resolvía los problemas de expresividad de este último. Entre esas limitaciones cabe destacar la representación de la colaboración y el *awareness* entre usuarios, así como la gestión de actores / roles.
- Finalmente, esta primera versión de CSRML fue evaluada mediante una familia de experimentos [22, 23] que pese a demostrar que CSRML era adecuado para el modelado de las distintas características de un sistema colaborativo, nos hizo descubrir ciertos problemas en la representación del *awareness* y de la importancia entre tareas colaborativas, lo que condujo al desarrollo de la segunda versión de CSRML, la cual solventa estos problemas mediante la inclusión / modificación de nuevos elementos expresivos.

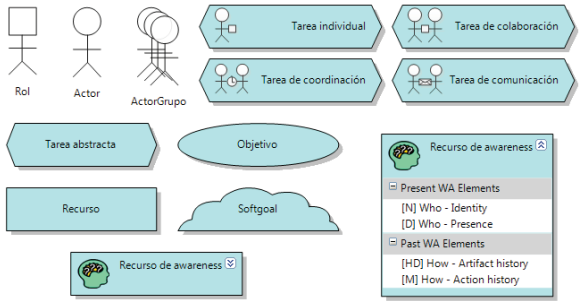

Fig. 1. Elementos de CSRML

Una vez vistos los trabajos que condujeron a CSRML, y antes de presentar la herramienta que da soporte al mismo, haremos un recorrido por los distintos elementos que forman parte de este lenguaje (Fig. 1):

- Un *rol* (*role*) representa el conjunto de conocimientos, destrezas y habilidades que debe poseer un actor para desarrollar con competencia una serie de tareas. 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).

- Un *actor* es un usuario, programa o entidad con ciertas capacidades adquiridas que puede interpretar un rol responsable de ciertas acciones. Un actor tiene que representar un rol (lo cual se especifica mediante un enlace de representación) para participar en el sistema.
- Un *actorgrupo* (*groupactor*) representa un grupo compuesto de uno o más actores cuyo objetivo es lograr uno o más objetivos específicos del grupo.
- Una *tarea* (*task*) especifica una forma particular de hacer algo. La importancia de una tarea se define mediante un código de colores: verde (menos importante), amarillo, naranja, rojo (más importante). 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 (*concrete task*): 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 (denominadas tareas 3C).
- Un *objetivo* (*goal/hardgoal*) responde a las preguntas "¿qué desea alcanzar el usuario?" "¿a qué aspira el usuario?". Describe un cierto estado que un actor desearía alcanzar. No obstante, un objetivo no describe cómo debería ser alcanzado.
- 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.
- 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 *recurso de awareness* (*awareness resource*) es una especialización del recurso que representa una necesidad de percepción que facilita y en algunos casos posibilita a un rol realizar una tarea proporcionándole el *feedback* requerido. Este elemento representa un conjunto de atributos relacionado con un *enlace de participación* entre un rol y una tarea. Sobre un diagrama, este elemento puede mostrarse de forma contraída o expandida (véase Fig. 1) para facilitar su legibilidad de las especificaciones y reducir la sobrecarga de información. En su forma expandida, este recurso muestra las características del *Workspace Awareness* identificadas por Gutwin [5] que pueden ser establecidas (si son necesarias) mediante el grado de contribución al cumplimiento de la tarea a la que el recurso

va asociado. La importancia puede ser *bueno* (N*, nice to have*), *deseable* (D, *desirable*)*, altamente deseable* (HD, *highly desirable*) u *obligatorio* (M, *mandatory*).

El conjunto anterior de elementos se relacionará entre sí mediante los siguientes enlaces (Fig. 2):

- Una *dependencia* (*dependency*) documenta una relación entre dos roles. Un rol depende de otro para conseguir un objetivo, realizar una tarea o usar un recurso.
- 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 pasos necesarios para realizar una tarea. Este enlace relaciona una tarea con sus componentes, los cuales pueden ser cualquier combinación de sub-objetivos, 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*, *negativa* o *desconocida* de ciertos *softgoals* sobre otros *softgoals* o tareas.
- 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. A este enlace podrá ir asociado un *recurso de awareness.*
- 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.

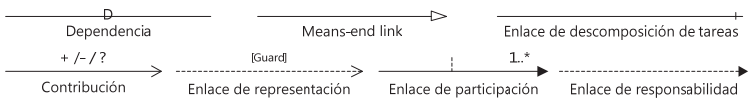

Fig. 2. Relaciones en CSRML v2

Finalmente, para estructurar una especificación de requisitos en CSRML de una forma ordenada y complementaria, se definen cinco tipos de diagramas. Algunos de ellos se usarán sólo una vez en la especificación del sistema, pero otros serán usados cuantas veces sean necesarios, dependiendo de las necesidades específicas del sistema. El conjunto completo de diagramas en CSRML y sus relaciones entre ellos se muestran en la siguiente figura (ver Fig. 3). Así, una especificación de requisitos en

CSRML se estructura por medio de los siguientes diagramas, que se detallan mediante un ejemplo en la sección 3.3:

- *Diagrama de jerarquía de grupo* (*Group Hierarchy Diagram, GHD*): éste es el primer diagrama que será empleado cuando se especifica un sistema colaborativo con CSRML (Fig. 8). En este diagrama se especificarán los grupos de *actores* que participarán en el sistema mediante los correspondientes *actorgrupo*. En un GHD se usan *enlaces de participación* para denotar qué actores forman cada grupo (con su correspondiente cardinalidad). Además, si un grupo tiene un líder (actor responsable de coordinar el grupo y ser el delegado en la interacción con otros grupos) su nombre se mostrará subrayado.

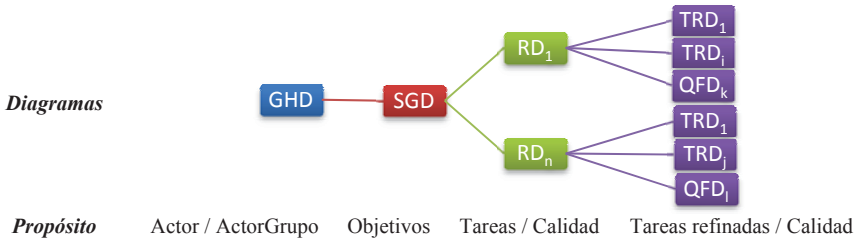

Fig. 3. Estructura de diagramas de CSRML

- *Diagrama de objetivos del sistema* (*System Goals Diagram, SGD*): en este diagrama (Fig. 9) se especifican los objetivos de actores y grupos de actores, así como la tarea principal del (sub)sistema. Estos actores y grupos de actores tendrán una cardinalidad (entre corchetes) que representará cuantas ocurrencias de los mismos participarán en el cumplimiento de un objetivo.
- *Diagramas de responsabilidades* (*Responsibility Diagrams, RDs*): en este diagrama (Fig. 10) se representa la descomposición de la tarea principal identificada en el diagrama anterior. Además, se usarán *enlaces de responsabilidad* para denotar qué actores (representando un rol) son responsables de cada objetivo, tarea o softgoal. Cada tarea representada en este diagrama se refinará por medio de un *diagrama de refinamiento de tareas.*
- *Diagramas de refinamiento de tareas* (*Task Refinement Diagrams, TRDs*): en estos diagramas (Fig. 11) se refinan las tareas identificadas en el RD en tareas más concretas y/o nuevos objetivos hasta que se especifiquen tareas concretas (individuales o colaborativas). Pueden haber tantos TRD como sean necesarios.
- *Diagramas de factores de calidad* (*Quality factors diagrams, QFDs*): en este último tipo de diagrama (Fig. 12) se especifican los factores de calidad que contribuyen a lograr los principales softgoals (factores de calidad) identificados en los RD. Estos softgoals se relacionan con el softgoal principal de calidad mediante *contribuciones.*

3 CSRML Tool: dando soporte al lenguaje CSRML

Una vez presentados los principales elementos del lenguaje CSRML, se procede a la presentación de una herramienta CASE que proporciona soporte al mismo. Para su

implementación se decidió la utilización de *Microsoft Visualization and Modeling SDK* (VMSDK, anteriormente conocido como DSL tools) [24], que permite crear herramientas de desarrollo basadas en modelos integradas con Visual Studio, entorno de desarrollo ampliamente utilizado de Microsoft.

Esta plataforma de creación de DSLs (*Domain Specific Languages*) de Microsoft fue seleccionada para crear CSRML Tool por su potencia gráfica necesaria para representar los elementos de CSRML (bastante complejos en algunos casos como el de los recursos de *awareness*), así como por la experiencia previa del equipo de desarrollo de la aplicación en entornos Microsoft .NET, entre otras razones. Además, dada su integración con Visual Studio, esta herramienta resulta fácil de utilizar por cualquier usuario con un mínimo de experiencia en este entorno. Dadas las facilidades que proporciona el entorno, las características de CSRML Tool son las siguientes:

- Especificación completa de un sistema colaborativo en CSRML mediante el uso de los distintos diagramas que componen el mismo.
- Validación automática de modelos respecto al meta-modelo de CSRML (Fig. 4 y Fig. 5), impidiendo la especificación de modelos no consistentes. Esta validación se lleva a cabo tanto mediante las restricciones establecidas en el metamodelo como utilizando el motor de reglas proporcionado por VMSDK.
- Interfaz de usuario sencilla integrada dentro de Visual Studio, con distintas áreas de trabajo como explorador de modelos o los cuadros de herramientas para los distintos diagramas de CSRML (Fig. 7)
- Generación de código para el enlace de CSRML con otras metodologías para el desarrollo de interfaces de sistemas colaborativos [25] (actualmente en desarrollo).
- Integración con Windows y Visual Studio mediante un sencillo programa instalador el cual integra los ficheros con extensión **.csrml* dentro del sistema operativo.

En el resto de esta sección se documentan detalles de esta herramienta respectivos a su implementación (Sección 3.1) e interfaz de usuario (Sección 3.2). Finalmente, se utilizará la herramienta para el modelado de un sistema colaborativo basado en una actividad de e-learning multiusuario cooperativa (Sección 3.3).

3.1 Detalles de implementación de CSRML Tool

Sin ahondar en exceso en los detalles de implementación de esta herramienta, en esta sección se hará una breve presentación a nivel gráfico a través de su meta-modelo (Fig. 4 y Fig. 5). El principal problema que se identificó inicialmente, es que VMSDK no usa un sistema de meta-modelado basado en UML, sino que usa un sistema propietario mediante el que, si bien resulta más sencillo el desarrollo del DSL, fue necesaria una conversión de nuestro meta-modelo original. Véase como ejemplo la parte gráfica de la definición de un recurso de *awareness* (Fig. 6).

Tras la definición del meta-modelo dentro de VMSDK, se procedió a la definición de los elementos gráficos de CSRML más complejos (tareas, softgoals, recursos de awareness), dado que este entorno sólo permite la inclusión de imágenes estáticas, además de cuatro formas geométricas básicas. Así, estos elementos complejos han

sido implementados mediante los servicios de *.net printing* [26], permitiendo de esta forma la inclusión de elementos no soportados de forma nativa por el entorno.

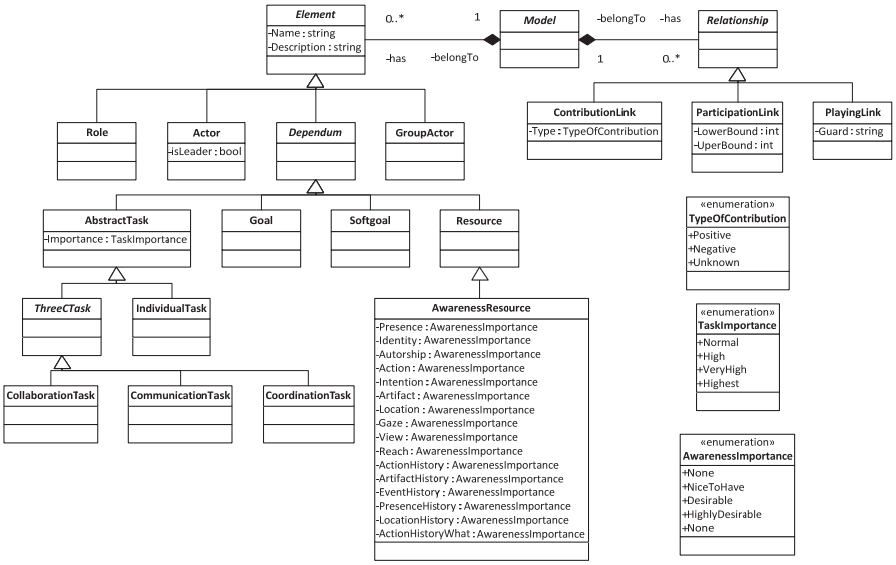

Fig. 4. Meta-modelo de CSRML (primera parte)

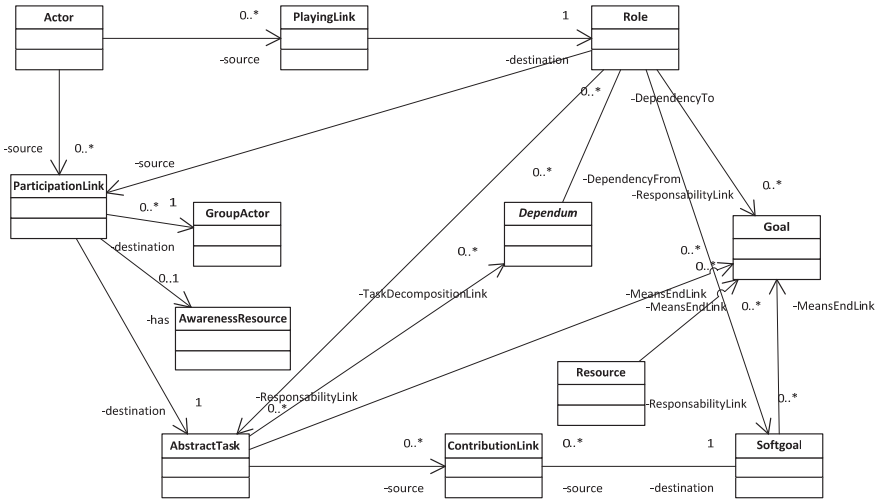

Fig. 5. Meta-modelo de CSRML (segunda parte)

Una vez que CSRML Tool soporta todos los elementos gráficos de CSRML, se procedió al desarrollo del código correspondiente con las restricciones necesarias para evitar todo tipo de inconsistencias en los modelos. Por ejemplo, cuando el usuario intente guardar un modelo en que más de un rol participe en una tarea individual, o bien exista algún tipo de recursividad en la descomposición de alguna tarea, el entorno advertirá al usuario y le indicará dónde está el error dentro del modelo.

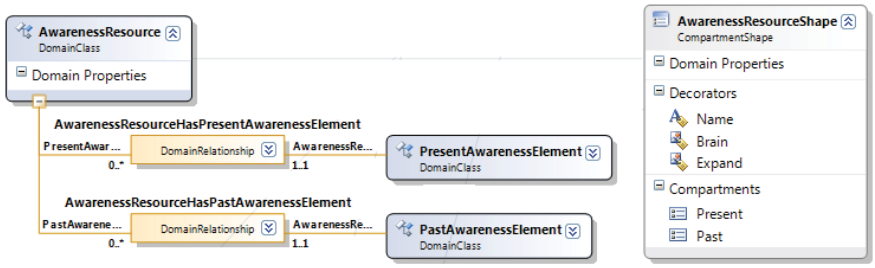

Fig. 6. Vista parcial del meta-modelo implementado dentro de VMSDK

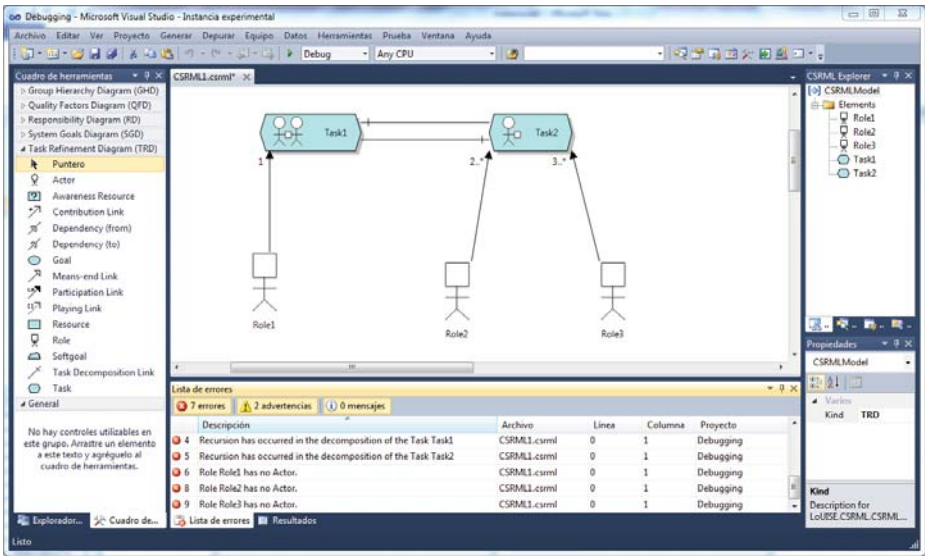

Fig. 7. Interfaz de usuario de CSRML Tool

Por último, se procedió a la mejora de la interfaz de usuario, creando el explorador de modelos CSRML en el cual el usuario puede navegar dentro una estructura en forma de árbol. Además, se crearon distintas cajas de herramientas para facilitar a los usuarios la definición de los distintos tipos de diagramas. Finalmente, también se creó un programa instalador que integra de forma automática CSRML con Windows y Visual Studio.

3.2 La interfaz de usuario de CSRML Tool

Una vez comentados distintos detalles relativos a la implementación de CSRML Tool, en esta sección se describe su interfaz de usuario (véase Fig. 7) en la que podemos contemplar la edición de un modelo con errores de validación.

La interfaz de CSRML Tool cuenta con los siguientes apartados:

- Área de modelado (parte superior central): permite editar y validar modelos.
- Cuadro de herramientas (parte izquierda): contiene todos los elementos y relaciones de CSRML agrupados por tipo de diagrama.

- Explorador del modelo (parte superior derecha): facilita la navegación y edición de un modelo dentro de una estructura con forma de árbol.
- Lista de errores de validación (parte inferior central): muestra los errores cometidos en el modelo activo y conduce al usuario a la localización exacta del error.
- Propiedades de los elementos (parte inferior derecha): permite cambiar las propiedades de modelos y elementos (importancias de tareas, cardinalidades, etc.)

3.3 Aplicando CSRML Tool: Jigsaw, una Actividad E-Learning Cooperativa

En esta sección se mostrará el uso de la aplicación CSRML Tool mediante el modelado de una actividad de *jigsaw* (puzzle) [27]. Esta es una técnica de aprendizaje cooperativo en la que los estudiantes realizan algún estudio a nivel individual sobre un problema propuesto, tras lo cual, enseñarán al resto de los integrantes de su grupo lo que han aprendido, compartiendo cada uno su punto de vista individual sobre el problema. Antes de realizarse esta actividad, se establecen los grupos (de entre 4 y 6 alumnos) y a cada uno de los alumnos se les proporciona una "pieza del puzzle" sobre la que aprender o investigar. Por ejemplo, una tarea de lectura considerablemente larga podría ser dividida en seis partes más pequeñas, convirtiéndose cada alumno en el *experto* de su parte asignada. Finalmente, las "piezas del puzzle" se juntarán cuando el grupo vuelva a reunirse y los alumnos compartirán su conocimiento.

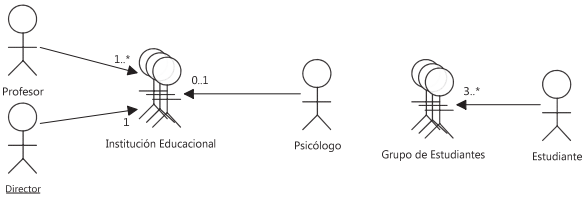

Fig. 8. Diagrama de jerarquía de grupo (GHD)

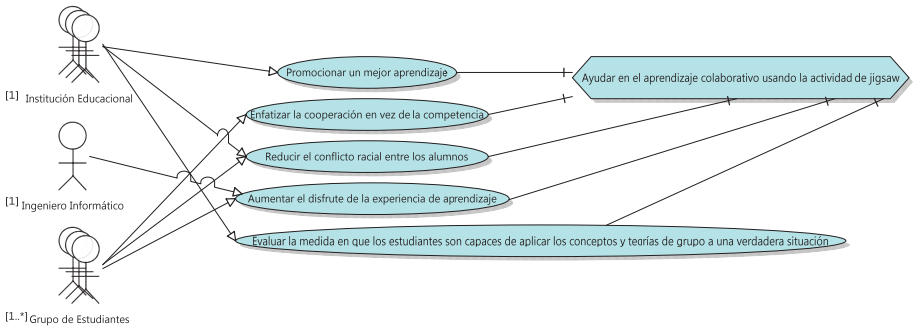

Fig. 9. Diagrama de objetivos del sistema (SGD)

Siguiendo la secuencia de diagramas mostrada en la Fig. 3, en primer lugar debe definirse el GHD (Fig. 8) para identificar los actores y actoresgrupo, así como para asignar actores a sus correspondientes grupos por medio de los enlaces de participación. Como puede observarse, una *Institución Educacional* consta de un

Director, uno o más *Profesores* y opcionalmente, un *Psicólogo*. Además, este grupo tiene un líder (*Director*, subrayado en el diagrama), pero no el *Grupo de Estudiantes*.

Después del GHD, debemos definir el SGD (Fig. 9) para especificar los objetivos del sistema. Estos objetivos son logrados mediante la tarea principal del sistema *Ayudar en el aprendizaje colaborativo usando la actividad de jigsaw* y la participación de varios actores (o grupo de actores). Por ejemplo, el objetivo *Aumentar el disfrute de la experiencia de aprendizaje* será logrado gracias a la participación de los *Grupos de Estudiante* y del *Ingeniero Informático*, quien podría añadir características multimedia a la interfaz con este fin. Como puede observarse en la cardinalidad establecida entre corchetes, uno o más grupos de estudiantes participarán en el logro de este objetivo, pero sólo participará un ingeniero informático.

Una vez definidos el GHD y el SGD, la tarea principal del sistema debe ser refinada por medio del RD (Fig. 10). En este diagrama se descompone la tarea principal en sub-tareas y softgoals de calidad. Además, se asigna las responsabilidades por medio de los enlaces de responsabilidad. Por ejemplo, el rol *Gestor del jigsaw* (representado por un *Profesor* mientras la actividad de jigsaw está realizándose) es responsable tanto de la tarea principal, como de *Crear Grupos*.

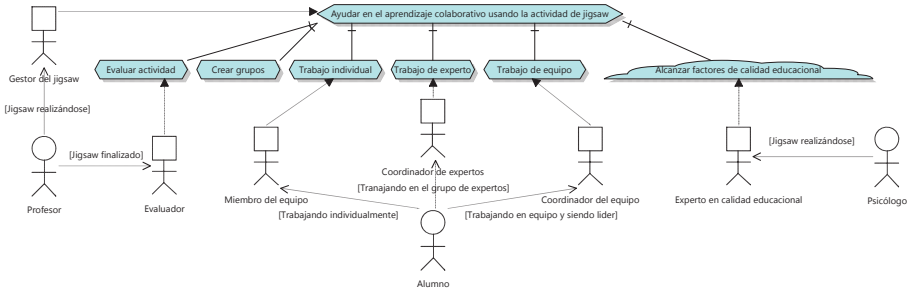

Fig. 10. Diagrama de responsabilidades (RD)

Cada una de las cinco sub-tareas (véase Fig. 10) en las que se descompone la tarea principal se refinará en un TRD diferente (debido las limitaciones de espacio mostraremos solamente uno de ellos). Por ejemplo, en la Fig. 11 se muestra cómo la tarea abstracta *Trabajo de equipo* se descompone en nuevos objetivos y sub-tareas (tanto individuales como 3C). Aquí, la involucración de roles (interpretados por actores) en tareas se realiza por medio de enlaces de participación. En este caso, para realizar la tarea *Hacer informe*, se necesitan dos o más *Miembros del equipo*. Como puede observarse, este enlace de participación está relacionado con un recurso de awareness debido a que para realizar dicho informe, los alumnos deben ser conscientes de la actividad de los otros miembros de su equipo (el comportamiento típico de un editor de textos colaborativo). Analizando el recurso de awareness de manera expandida se puede comprobar que para la realización de esta tarea, los usuarios tienen que ser conscientes de qué están haciendo los otros usuarios y con qué artefacto (párrafo, imagen, tabla, etc.) están trabajando (*What-Action* y *What-Artifact*). Además, los usuarios deberían también percibir con quién están colaborando y dónde están trabajando (*Who-Identity* y *Where-Location*).

Adicionalmente, es deseable ser consciente de la intención del resto de usuarios (*What-Intention*), y sería interesante tener información de lo que están viendo exactamente (*Where-View*) y de los elementos que tienen a su alcance (*View-Reach*). Además, en este TRD podemos observar cómo se han especificado las distintas importancias entre tareas, teniendo por ejemplo *Enviar informe* una importancia normal y *Hacer informe* una importancia muy alta.

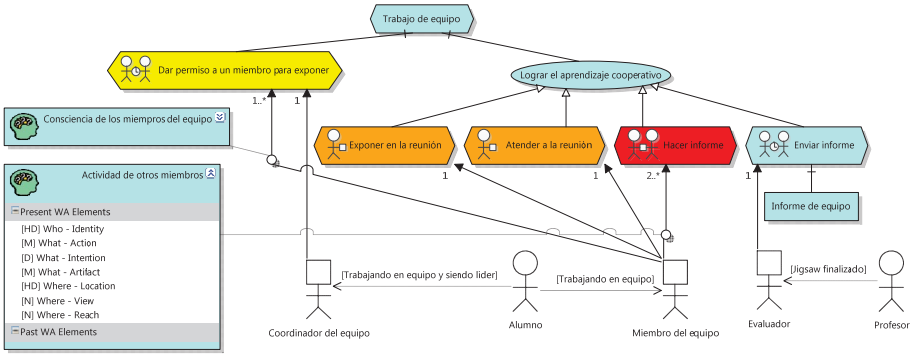

Fig. 11. Diagrama de refinamiento de tareas (TRD)

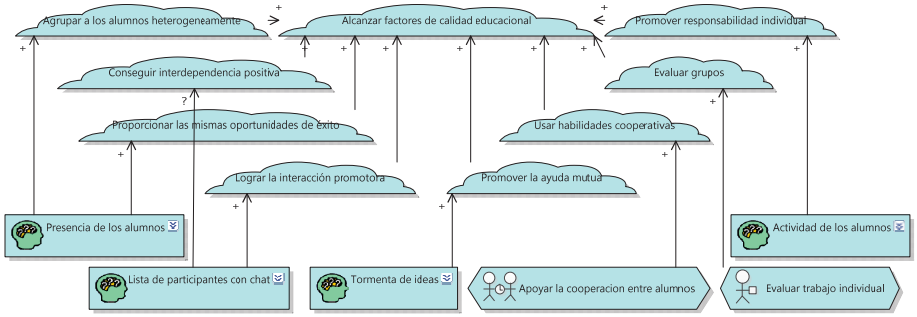

Fig. 12. Diagrama de factores de calidad (QFD)

Finalmente, la Fig. 12 muestra el último de los diagramas propuestos en CSRML, el QFD. En este modelo se muestran los factores de calidad que contribuyen a realizar la actividad de *jigsaw* con un nivel de calidad elevado. Estos factores se representan como softgoals y se relacionan con el factor de calidad principal mediante contribuciones. El logro de estos softgoals de calidad se alcanzará de diferentes formas. Por ejemplo, el softgoal *Agrupar a los alumnos heterogéneamente* será logrado por medio del recurso de awareness *Presencia de los alumnos*, que podría ser implementado con un *video embodiment widget*.

4 Conclusiones y trabajo futuro

En trabajos anteriores, basándonos en un proceso guiado por evaluaciones empíricas se definió el lenguaje CSRML [8, 21] como una extensión del lenguaje Goal-Oriented *i** [9, 10] que permitirá la especificación de los requisitos de un sistema colaborativo,

que son difíciles con las técnicas clásicas de IR [6, 7]. Esto es debido a la necesidad especial de percepción (*Workspace Awareness*) que requieren los usuarios de un sistema colaborativo acerca de, por ejemplo, quién está disponible para colaborar, qué están haciendo el resto de usuarios, quién y cuándo se realizó determinada acción.

En este trabajo se presenta CSRML Tool, la herramienta que permite la especificación de requisitos utilizando el lenguaje CSRML que ha sido creado mediante la implementación del meta-modelo de CSRML en Visualization and Modeling SDK (VMSDK). CSRML Tool permite la especificación de los requisitos de un sistema colaborativo por medio de los cinco tipos de diagramas que forman CSRML basada en una interfaz de usuario sencilla y totalmente integrada con Windows y Visual Studio. Esta herramienta permite validar los modelos creados, evitando errores e incoherencias y facilitando la futura modificación de los mismos.

Nuestro trabajo futuro consiste en la creación de una nueva versión de CSRML que permita la creación de distintas vistas sobre un mismo modelo, ya que actualmente VMSDK carece de esa funcionalidad de forma nativa. Además, otra característica que será añadida a la herramienta será la trazabilidad con técnicas de desarrollo de interfaces de usuario para sistemas colaborativos, como CIAM [25], obteniéndose así una metodología completa para el desarrollo de este tipo de sistemas. Finalmente, también forma parte de nuestro trabajo futuro la realización de experimentos con usuarios a fin de validar su usabilidad.

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.

Bibliografía

1. Google Inc.: Google DoubleClick Ad Planner 1000 most-visited sites on the web, http://www.google.com/adplanner/static/top1000/index.html.
2. Google Inc.: Google Docs, https://docs.google.com.
3. Schmidt, K., Bannon, L.: Taking CSCW seriously. Computer Supported Cooperative Work. 1, 7-40 (1992).
4. Ellis, C.A., Gibbs, S.J., Rein, G.: Groupware: some issues and experiences. Communications of the ACM. 34, 39-58 (1991).
5. Gutwin, C., Greenberg, S.: A Descriptive Framework of Workspace Awareness for Real-Time Groupware. Computer Supported Cooperative Work. 11, 411-446 (2002).
6. Teruel, M.A., Navarro, E., Lopez-Jaquero, V., Montero, F., Gonzalez, P.: An empirical evaluation of requirement engineering techniques for collaborative systems. 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE'11). pp. 114-123. IET, Durham, UK (2011).
7. Teruel, M.A., Navarro, E., López-Jaquero, V., Montero, F., González, P.: A Comparative of Goal-Oriented Approaches to Modelling Requirements for Collaborative Systems. 6th International Conference on Evaluation of Novel Software Approaches to Software Engineering (ENASE'11). pp. 131-142. SciTePress, Beijing, China (2011).
8. Teruel, M.A., Navarro, E., López-Jaquero, V., Montero, F., González, P.: CSRML: A Goal-Oriented Approach to Model Requirements for Collaborative Systems. 30th

International Conference on Conceptual Modeling (ER'11). pp. 33-46. Springer Berlin Heidelberg, Brussels, Belgium (2011).
9. Castro, J., Kolp, M., Mylopoulos, J.: A requirements-driven development methodology. 13th Int. Conf. On Advanced Information Systems Engineering (CAiSE'01). pp. 108-123. Springer-Verlag, London, UK (2001).
10. López, L., Franch, X., Marco, J.: Making Explicit Some Implicit i* Language Decisions. 30th International Conference on Conceptual Modeling (ER'11). pp. 62-77 (2011).
11. Ernst, N., Yu, Y., Mylopoulos, J.: Visualizing Non-Functional Requirements. 2006 First International Workshop on Requirements Engineering Visualization (REV'06 - RE'06 Workshop). pp. 2-2. IEEE (2006).
12. Gronback, R.C.: Eclipse Modelling Project: A Domain-Specific Languaje Toolkit. Addison-Wesley (2009).
13. Malta, Á., Soares, M., Santos, E., Paes, J., Alencar, F., Castro, J.: iStarTool: Modeling requirements using the i* framework. CEUR Proceedings of the 5th International i* Workshop (iStar 2011). pp. 163-165 (2011).
14. Kolp, M., Faulkner, S., Wautelet, Y., Nguyen, T., Coyette, A., Achbany, Y., Hoang, H., Do, T.: DesCARTES Architect. Business Driven Software Design. Forum des Recherches Doctorales. , Institut d'administration et de gestion (IAG) (2005).
15. Grau, G., Franch, X., Avila, S.: J-PRiM: A Java Tool for a Process Reengineering i* Methodology. 14th IEEE International Requirements Engineering Conference (RE'06). pp. 359-360. IEEE (2006).
16. van Lamsweerde, A.: Goal-oriented requirements engineering: a guided tour. Fifth IEEE International Symposium on Requirements Engineering (RE'01). pp. 249-262. IEEE Comput. Soc, Washington DC, USA (2001).
17. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley Professional (2000).
18. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley Professional (2005).
19. Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., Goedicke, M.: Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. International Journal of Software Engineering and Knowledge Engineering. 2, 31-57 (1992).
20. Cysneiros, L.M., Yu, E.: Non-Functional Requirements Elicitation. In: do Prado Leite, J.C.S. and Doorn, J.H. (eds.) Perspectives on Software Requirements. Kluwer (2003).
21. Teruel, M.A., Navarro, E., López-Jaquero, V., Montero, F., González, P.: Modelado de Requisitos de Sistemas Colaborativos con CSRML. XVI Jornadas de Ingeniería del Software y Bases de Datos (JISBD'11). pp. 639-652. , A Coruña, Spain (2011).
22. Teruel, M.A., Navarro, E., López-Jaquero, V., Montero, F., González, P.: Assessing the Understandability of Collaborative Systems Requirements Notations: an Empirical Study. 1st Int. Work. on Empirical Requirements Engineering (EmpiRE'11). pp. 85-92, Trento, Italy (2011).
23. Teruel, M.A., Navarro, E., López-Jaquero, V., Montero, F., Jaen, J., González, P.: Analyzing the Understandability of Requirements Engineering Languages for CSCW Systems: A Family of Experiments. Information and Software Technology. (2012).
24. Özgür, T.: Comparison of Microsoft DSL Tools and Eclipse Modeling Frameworks for Domain-Specific Modeling In the context of the Model-Driven Development, (2007).
25. Molina, A.I., Redondo, M.A., Ortega, M., Hoppe, U.: CIAM: A methodology for the development of groupware user interfaces. JUCS. 14, 1435–1446 (2008).
26. Aitken, P.G.: .Net Graphics and Printing. Optimax Pub (2003).
27. Pozzi, F.: Using Jigsaw and Case Study for supporting online collaborative learning. Computers & Education. 55, 67-75 (2010).