# SmarTTrace: Una Herramienta para Trazabilidad de Requisitos en Proyectos basados en UML*

**Víctor Anaya y Patricio Letelier**
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, España
{vanaya, letelier}@dsic.upv.es

**Abstract.** La trazabilidad de requisitos es un proceso clave para la exitosa gestión de los requisitos de un sistema de información. A pesar de ello, no existe un consenso acerca de los tipos de especificaciones y enlaces entre éstos que deben usarse en un proceso de trazabilidad. Además, a pesar de la existencia de herramientas específicas para la gestión de requisitos, éstas no proven mecanismos adecuados para la configuración de la trazabilidad de acuerdo con las necesidades específicas del proyecto. En este artículo se presenta SmarTTrace, una herramienta que aprovecha los mecanismos de extensión proporcionados por la CASE Rational Rose para incorporar a ésta las tareas de configuración, especificación y explotación de trazabilidad en un proceso de desarrollo. Para ilustrar la funcionalidad de SmarTTrace respecto a dichas tareas se ha propuesto un pequeño ejemplo basado en el proceso *Rational Unified Process*.

Keywords: Gestión de requisitos, trazabilidad de requisitos, UML.

## 1 Introducción

La trazabilidad de requisitos es clave para conseguir una exisota gestión de requisitos. Dicha importancia no se ve refleja en un consensorespecto de las prácticas con el que el proceso de trazabilidad ha de llevarse a cabo [7]. No existen estándares asociados al proceso de trazabilidad que ayuden a determinar qué tipos de artefactos y de enlaces se han de considerar. Esto provoca la paradoja de que a pesar de la importancia de este proceso y de las múltiples herramientas de gestión de requisitos, no se provean soluciones adecuadas para configurar la trazabilidad de acuerdo a las necesidades específicas del proyecto. El presente artículo define un marco de trabajo que sirve como base para la especificación de los tipos de artefactos y enlaces de interés de la trazabilidad de un proceso de desarrollo software.

* Este trabajo ha sido financiado por el proyecto *DOLMEN-SIGLO* de la Comisión Interministerial de Ciencia y Tecnología, TIC2000-1673-C06-01.

La trazabilidad de requisitos se define como la habilidad para describir y seguir la vida de un requisito en ambos sentidos, hacia sus orígenes o hacia su implementación, a través de todas las especificaciones generadas durante el proceso de desarrollo de software.

Para ello el proceso de trazabilidad ha de considerar dos subprocesos: a) configuración de la trazabilidad de acuerdo con las necesidades concretas del proyecto [1], para así conseguir un resultado positivo respecto del costo-beneficio asociado, b) especificación de la trazabilidad en el proyecto y la posterior explotación de dicha información.

En este trabajo se presenta SmarTTrace, una herramienta que hace uso de los mecanismos de extensión proporcionados por Rational Rose para incoporar configuración y especificación de trazabilidad en dicha herramienta. SmarTTrace hace de Rational Rose un único entorno de trabajo en el que se integran tanto especificaciones UML como no UML principalmente. Para ello integra los tipos de artefactos de trazabilidad considerados en el *framework* que no considera Rational Rose (especificaciones textuales y algunos tipos de artefactos UML no considerados en Rational Rose) en el metamodelo de Rational Rose. Al proporcionar un único entorno de trabajo se evitan problemas de sincronización y de cambio de contexto para el analista, es decir, puede realizar la gestión de requisitos y el modelado UML de la aplicación en el mismo entorno. Adicionalmente, SmarTTrace provee un marco de trazabilidad extensible con tipos de artefactos y de enlaces de trazabilidad definidos por el usuario. SmarTTrace permite disponer de diferentes tipos de enlaces de trazabilidad de acuerdo a la naturaleza de los tipos de artefacto involucrados. El marco de trazabilidad provisto por SmarTTrace se incorpora a Rational Rose extendiendo su metamodelo mediante un profile UML detallado en [6].

Este artículo se organiza en ocho secciones. Tras la introducción, la siguiente sección describe el metamodelo para trazabilidad utilizado como marco de trabajo. La sección tercera explica cómo las especificaciones textuales y los enlaces de trazabilidad se pueden definir en UML consiguiendo un marco homogéneo para toda la información de trazabilidad. A continuación, en la sección cuarta se presenta un proceso para la configuración de la trazabilidad. En la sección quinta se explica la integración de SmarTTrace en Rational Rose. En la sección sexta se ilustra un caso práctico de configuración de trazabilidad con SmarTTrace, utilizando un pequeño ejemplo basado en el proceso RUP (Rational Unified Process). La sección séptima describe algunas herramientas representativas, específicas para la gestión de requisitos. Finalmente la sección octava presenta las conclusiones.

## 2 Un Metamodelo para Trazabilidad de Requisitos

Antes de presentar nuestro metamodelo de trazabilidad resumiremos las necesidades de información para la gestión de requisitos. A continuación se indican los tipos de información asociada a la trazabilidad de requisitos y sus posibles usos (adaptado desde [1][8]):

1. Los enlaces de trazabilidad entre diferentes tipos de especificaciones permiten: validar que la funcionalidad del sistema reúne las expectativas del cliente y que no se ha implementado funcionalidad superflua, y realizar análisis de impacto de cambios en los requisitos.
2. Las estructuras de contribución [2], es decir, los enlaces entre participantes en el proyecto (*stakeholders*) y las especificaciones permiten: mejorar la comunicación y cooperación entre los participantes del proyecto, y asegurar que la contribución de cada individuo es considerada y registrada.
3. Los fundamentos asociados a las especificaciones, incluyendo alternativas, decisiones, suposiciones, etc. contribuyen a: mejorar la comprensión y aceptación del sistema por parte de los *stakeholders*, y a mejorar la gestión de los cambios evitando reconsiderar cuestiones ya antes descartadas, pues las soluciones y sus fundamentos, así como las alternativas descartadas son accesibles.

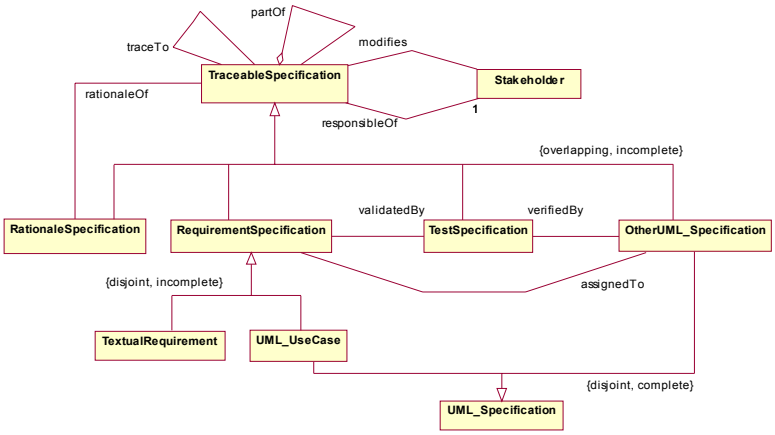


**Fig. 1.** Un metamodelo para trazabilidad de requisitos

El metamodelo que proponemos se muestra en la Figura 1 mediante un diagrama de clases. Las clases representan los tipos de entidades y las asociaciones los tipos de enlaces de trazabilidad. Se utilizan nombres de roles (*rolnames*) para distinguir los distintos tipos de enlaces de trazabilidad.

En términos generales interesan dos tipos de entidades *TraceableSpecification* y *Stakeholders*. Los *Stakeholders* son responsables de crear y modificar especificaciones. Una *TraceableSpecification* es una especificación de software con un determinado nivel de granuralidad, es decir, puede ser un documento, un modelo, un diagrama, un apartado de un documento, un texto especificando un requisito no-funcional, un caso de uso, una clase, un atributo, etc. Esta granularidad para

una *TraceableSpecification* se define mediante la agregación con el nombre de rol *partOf*.

El tipo de entidad *TraceableSpecification* es una generalización de *RationaleSpecification*, *RequirementSpecification*, *TestSpecification*, y *OtherUML_Specification*. Una *TraceableSpecification* puede clasificarse en más de uno de estos subtipos, por ejemplo, cuando se trata de un documento que incluye distintos tipos de especificaciones (mediante *partOf*). Una *RequirementSpecification* es un requisito o grupo de requisitos. Los requisitos, según cómo se expresan, puede clasificarse en *TextualRequirements* (requisitos expresados mediante un texto) o *UML_UseCase* (elemento de modelado usado en UML para representar un requisito funcional). Una *RationaleSpecification* establece, por ejemplo: fundamentos, alternativas o suposiciones asociadas a una *TraceableSpecification*. Finalmente, una *TestSpecification* define una prueba, ya sea para validar un requisito o para verificar un elemento de modelado UML (por ejemplo: para verificar un fichero con código fuente que es la implementación de una clase o un componente). Las generalizaciones cuyas clases padre son *TraceableSpecification* y *RequirementSpecification* están definidas como "incompletas" para permitir otros tipos de especificaciones que puedan ser de interés para trazabilidad. Por ejemplo, algunos tipos de especificación no textuales ni UML son: vídeos, imágenes, voz, etc. Éstos corresponden usualmente a fundamentos, útiles durante la revisión y evaluación de modelos de análisis y diseño [3], sin embargo, podría tratarse simplemente de otro tipo de medio, por ejemplo, una *RequirementSpecification* registrada en un vídeo.

El tipo de enlace más genérico es la asociación con nombre de rol *traceTo* el cual permite establecer enlaces de trazabilidad entre *TraceableSpecifications*. El resto de los tipos de enlace (*modifies*, *responsibleOf*, *rationaleOf*, *validatedBy*, *verifiedBy* y *assignedTo*) son más específicos. El tipo de enlace *modifies* permite establecer una relación entre los *Stakeholders* y las *TraceableSpecifications* que éstos modifican. Del mismo modo, *responsibleOf* determina el *Stakeholder* que es responsable de la definición y mantenimiento de una *TraceableSpecification*. El tipo de enlace *validatedBy* relaciona a las *RequimementsSpecifications* con las correspondientes *TestSpecification* que las validan. Correspondientemente, el tipo de enlace *verifiedBy* determina las *TestSpecifications* que verifican una especificación UML. Por último, el tipo de enlace *assignedTo* determina a qué elementos de modelado UML se les asigna la realización de un determinado requisito, por ejemplo, qué clases realizan un Caso de Uso.

## 3 El Metamodelo en el contexto de UML

Considerando que: (a) las especificaciones UML están definidas con mayor precisión y aceptación que los otros tipos de especificaciones incluidas en el metamodelo, (b) que UML provee mecanismos de extensión (*stereotypes*, *tagged values* y *constraints*) para incorporar nuevos tipos de especificaciones y (c) que las especificaciones UML tienen un amplio soporte en las herramientas CASE hemos considerado conveniente integrar todos los tipos de especificaciones del meta-

modelo dentro del contexto de UML. Así, para cada tipo de entidad y tipo de enlace presentes en el metamodelo de trazabilidad se establece una correspondencia con un elemento de modelado en UML. En cada caso, se selecciona una metaclase del metamodelo de UML que se utiliza como clase base para establecer un estereotipo. Para aquellos tipos de entidad y de enlace del metamodelo que coincidan semánticamente con una metaclase de UML se utiliza directamente dicha metaclase para representarlos, sin definir un nuevo estereotipo. El resultado de este análisis es un *profile* UML asociado a nuestro metamodelo de trazabilidad. A continuación se detalla cómo se realiza dicha integración.

*Entidades en UML* Para los tipos de entidad del metamodelo el elemento de modelado UML que los represente debe permitir asociaciones para poder establecer relaciones de agregación entre especificaciones. De acuerdo con esto, la elección debe estar entre los elementos de modelado UML que son especialización de *Classifier*. Para la entidad *Stakeholder* la elección fue directa; el elemento de modelado *Actor* se ha utilizado como clase base para el estereotipo. Para las entidades correspondientes a especificaciones no-UML estándar se ha seleccionado el *Classifier* llamado *Artifact* (añadido en la versión 1.4 de UML). Nótese que *Artifact* ya tiene definidos algunos estereotipos, entre ellos «document» que es el que utilizaremos para representar documentos y secciones de documentos. Para los tipos de entidad que se corresponden directamente con elementos de modelado de UML (*UML_ UseCase* y *OtherUML_ Specification*) se utiliza el correspondiente elemento de modelado UML para representarlas. Por otra parte, para la agrupación y organización de artefactos UML y *Stakeholders* utilizamos el elemento de modelado UML provisto para este fin, denominado *Package* y opcionalmente añadiremos los estereotipos predefinidos «model» o «subsystem» , según estemos definiendo un modelo de un sistema/subsistema o estemos dividiendo un sistema en subsistemas, respectivamente.

*Enlaces en UML* Los tipos de enlaces están indicados como asociaciones en nuestro metamodelo de trazabilidad y son representados como elementos de modelado UML del tipo *Abstraction*, excepto para el caso de la relación *partOf*, la cual se representa mediante la agregación o composición entre especificaciones usando como clase base la metaclase *Association*. Aunque los distintos tipos de enlaces son modelados por diferentes asociaciones en el metamodelo de trazabilidad, éstas no son independientes, de hecho, el tipo de enlace *traceTo* es la generalización de todos los otros tipos de enlace. Así, *traceTo* se hace coincidir con el estereotipo «trace», ya definido de UML. Una dependencia «trace» indica una relación histórica o de proceso entre dos elementos que representan el mismo concepto sin especificar reglas de derivación entre ellos [9]. Exceptuando el tipo de relación *partOf*, todos los tipos de enlace de nuestro metamodelo de trazabilidad son especializaciones del estereotipo predefinido «trace».

De acuerdo a lo comentado en los apartados previos, en la Figura 2 y Figura 3 se muestra la representación UML de los tipos de entidades y los tipos de enlaces

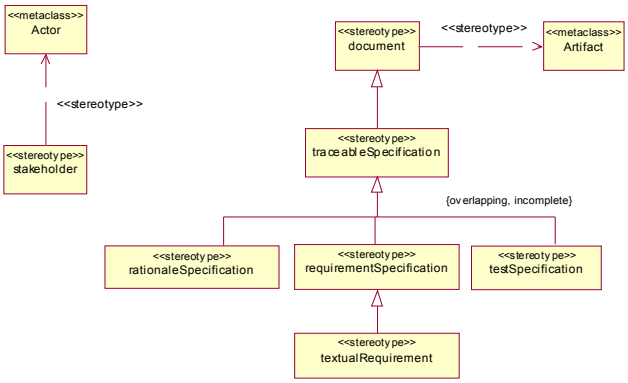


**Fig. 2.** Estereotipos para *stakeholder* y para especificaciones textuales básicas

del metamodelo de trazabilidad. Esta representación constituye un *profile* UML para trazabilidad de requisitos.

## 4 Configuración de la trazabilidad

En trazabilidad de requisitos podemos identificar dos actividades: (a) configurar la trazabilidad respecto de las necesidades del proyecto y (b) especificar y explotar la información de trazabilidad durante el desarrollo y mantenimiento del software. Nos centraremos en la actividad de configuración haciendo uso del metamodelo de UML extendido con nuestro profile para trazabilidad (definido por el metamodelo de trazabilidad). El profile de trazabilidad actuará como marco para establecer los tipos de artefacto[1] entre los que se desea registrar trazabilidad y los tipos de enlaces de trazabilidad entre ellos. A su vez, se permite extender el profile para indicar los tipos de artefactos de interés para trazabilidad en cada proyecto. A cada nuevo tipo de artefacto se le asocia un estereotipo especializando uno de los estereotipos del profile de trazabilidad (excepto para aquellos tipos de artefacto que se corresponden directamente con un elemento de modelado UML). Al hacer esto, se está extendiendo el profile de trazabilidad en lo que respecta a tipos de artefacto. Consecuentemente, los enlaces de trazabilidad establecidos durante el modelado del sistema se verifican con respecto de la configuración de trazabilidad (por ejemplo, para un proyecto específico ciertos tipos de enlaces son válidos sólo entre determinados tipos de artefactos).

[1] A partir de aquí utilizaremos el témino "artefacto" en un sentido más amplio al dado en UML, tal como lo hacen la mayoría de los procesos de software (por ejemplo RUP). Así, consideraremos como artefactos a todos los documentos, ficheros y otros elementos físicos generados y/o usados durante el proyecto, pero además, llamaremos también artefactos a cualquier elementos de modelado UML.

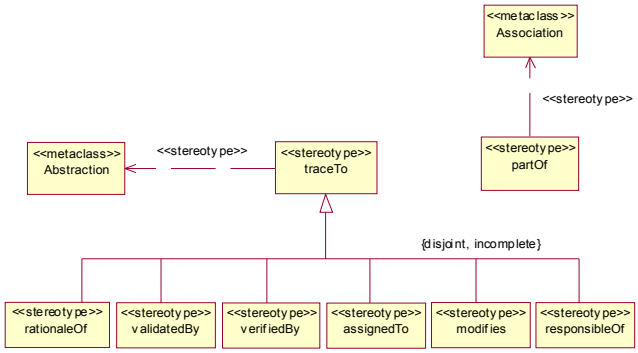


**Fig. 3.** Estereotipos para tipos de enlaces de trazabilidad

La configuración de trazabilidad para un proyecto incluye las siguientes tareas:

1. Extender el marco de trabajo con la definición de aquellos tipos de artefactos del proceso de desarrollo software que son usados en el proyecto y que no esten considerados en el metamodelo extendido con el profile.
2. Definir las relaciones de agregación entre los tipos de artefactos del proyecto. Esta tarea puede que no sea necesaria si dicha información está predefinida como parte de la descripción de cada tipo de artefacto en el metamodelo.
3. Seleccionar los tipos de artefactos para los cuales se realizará un seguimiento de trazabilidad. Se trata de un subconjunto del total de tipos de artefactos que se utilizarán en el proyecto.
4. Establecer los tipos de enlaces de trazabilidad de interés para el proyecto. Éstos han de establecerse entre pares de tipos de artefactos seleccionados en la tarea 3. En este caso también puede ser necesario extender el *profile* de trazabilidad definiendo tipos de enlaces de trazabilidad más específicos.
5. Definir criterios para la derivación de enlaces de trazabilidad implícita y establecer qué enlaces de trazabilidad, entre los indicados en la tarea 4, se desean generar por alguno de los criterios definidos. El esfuerzo asociado a la introducción de los enlaces de trazabilidad puede ser considerable, con lo cual es deseable que se provean mecanismos que permitan derivar automáticamente parte de los enlaces de trazabilidad. Denominaremos trazabilidad explícita a aquellos enlaces de trazabilidad que son introducidos manualmente. Consecuentemente, llamaremos trazabilidad implícita a aquellos enlaces de trazabilidad que se derivan automáticamente de acuerdo con ciertos criterios.

Las tareas 1 y 2 no son consideradas tareas propias del proceso de configuración de trazabilidad, debiendo realizarse a priori. Siendo éstas tareas clave de todo proceso de desarrollo de software.

## 5 SmarTTrace y su integración en Rational Rose

SmarTTrace es una herramienta que extiende y adapta Rational Rose para incorporar a éste el proceso de configuración de trazabilidad y capacidades de definición de aquellos tipos de artefactos no considerados en su metamodelo. SmarTTrace se implementa como un add-in de Rational Rose que hace uso de las funciones provistas por la interfaz REI (*Rose Extensibility Interface*) para acceder al metamodelo de Rational Rose. Los insuficientes mecanismos de adaptación y extensibilidad de Rational Rose han dificultado el desarrollo del marco de trabajo de trazabilidad. A eso se añaden las limitaciones mostradas por el metamodelo de Rational Rose. Éste carece de algunas de las metaclases UML a partir de los cuales se define el *profile* UML de trazabilidad, como por ejemplo las metaclases *Actor*, *Artifact* y *Abstraction,* o el estereotipo *Document.* Además, Rational Rose no permite definir jerarquías de estereotipos, necesarias para el *profile* UML.

La adaptación y extensión de la interfaz de Rational Rose es muy reducida. No se puede modificar libremente la estructura del *browser*, la cual está sujeta al sistema de vistas 4+1 [5]. Adicionalmente, por las limitaciones de los add-in, no se pueden definir *browsers* alternativos. Al tener que definir el metamodelo de trazabilidad a nivel modelo, todos los tipos de artefactos no considerados en Rational Rose se muestran en una subcategoría de la vista lógica.

Las capacidades de definición de nuevos tipos de artefactos provistas por SmarTTrace se integran con las de Rational Rose. Por consiguiente, la especificación de artefactos disponibles en Rational Rose se realiza de forma independiente a SmarTTrace. La definición de los artefactos asociados a los tipos definidos en el profile base o introducidos por el usuario se realiza mediante los mecanismos provistos por SmarTTrace. En la Figura 4 se muestra la integración de las opciones de configuración proporcionadas por SmarTTrace en Rational Rose.

## 6 Configuración de la trazabilidad de un proyecto RUP en SmarTTrace

Tanto SmarTTrace, como el metamodelo de trazabilidad subyacente son independientes del proceso de desarrollo. Sin embargo, para ilustrar su aplicación hemos elegido RUP como proceso, por disponer en él de un gran detalle respecto de los tipos de artefactos. RUP es un producto de Rational Software basado en el Proceso Unificado de Desarrollo de Software [4].

A continuación se ilustran las tareas que componen la configuración de la trazabilidad de requisitos en un proyecto pequeño basado en RUP haciendo uso de SmarTTrace. La tarea 5 de dicho proceso no se ha indicado ya que aún no ha sido considerada en la herramienta.

***Tarea 1*** Definir los tipos de artefactos usados en el proyecto. Los tipos de artefacto de RUP que utilizaremos en nuestro ejemplo se muestran en la siguiente tabla de la siguiente página.

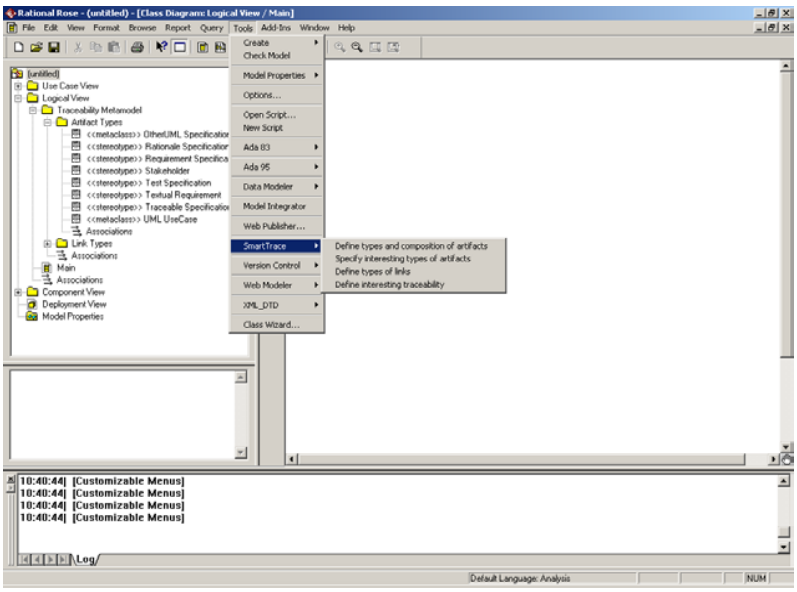


**Fig. 4.** Opciones de Configuración de Trazabilidad de SmarTTrace

Cuando en la tabla no aparece el estereotipo es porque se utiliza exactamente el mismo elemento de modelado definido en UML, con lo cual el usuario no ha tenido que extender el marco de trabajo base. El estereotipo «model» se establece a partir de la metaclase "package" de UML.

La herramienta SmarTTrace permite definir aquellos tipos de artefactos RUP no disponibles en Rational Rose a partir del metamodelo de trazabilidad subyacente a SmarTTrace. En la Figura 5 se muestra cómo se desea crear el tipo de artefacto *UseCase Specification* como subtipo del tipo de artefacto *Traceable Specification*.

| Tipo de Artefacto RUP | Estereotipo desde el cual se especializa |
|---|---|
| Vision | «traceableSpecification» |
| Software Feature | «textualRequirement» |
| Supplementary Specification | «traceableSpecification» |
| Non-Functional Requirement | «textualRequirement» |
| Assumption | «rationaleSpecification» |
| Use Case Specification | «traceableSpecification» |
| Flow of Events | «traceableSpecification» |
| Use Case Model | «model» |
| Use Case | |
| Analysis & Design Model | «model» |
| Class | |
| Implementation Model | «model» |
| Component | |
| Data Model | «model» |
| Table | |
| Test Case | «testSpecification» |

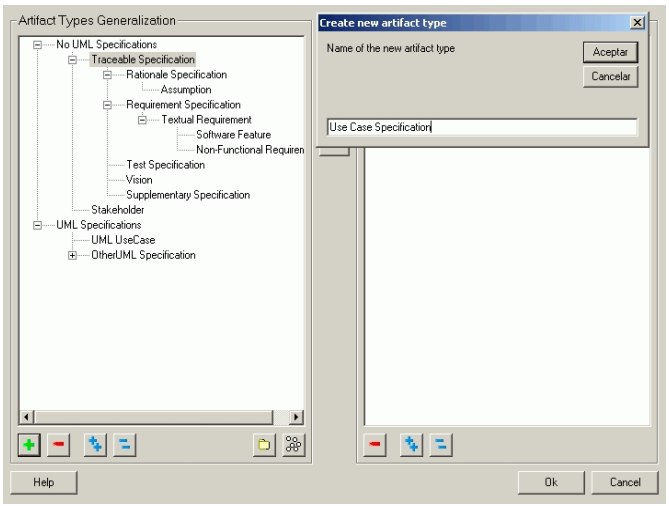


**Fig. 5.** Definición de tipos de artefactos no contemplados en UML

***Tarea 2*** Se definen las relaciones de composición para los tipos de artefacto de nuestro ejemplo:

Vision ◇— Software Feature
Vision ◇— Assumption
Supplementary Specification ◇— Non-Functional Requirement
Use Case Specification ◇— Flow of Events
Use Case Model ◇— Use Case
Analysis & Design Model ◇— Class
Implementation Model ◇— Component
Data Model ◇— Table

En SmarTTrace esta tarea se realiza a partir de la opción de definición de tipos de artefacto. El usuario, a partir del conjunto de tipos de artefacto provistos por el metamodelo al final de la tarea 1, indica la composición de los mismos estableciendo relaciones "*part of*" entre ellos. Se genera un árbol de composición de tipos de artefactos como el mostrado en la Figura 6. En la Figura 6 se observa como el usuario se dispone a definir la composición existente entre los tipos de artefactos *"Flow of Events"* y *"Use Case Specification"*.

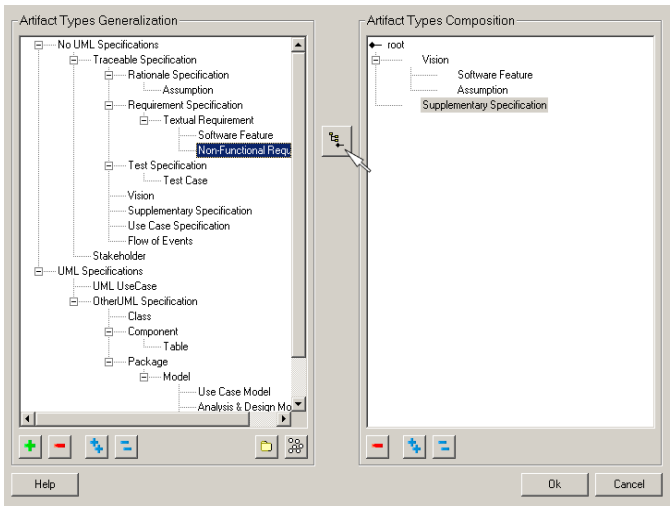


**Fig. 6.** Definición de la composición de tipos de artefactos de interés

***Tarea 3*** A partir del conjunto de tipos de artefactos usados en el proyecto, resultantes de la tarea 1, se seleccionan los tipos de artefactos entre de los que se desea registrar trazabilidad. En la Figura 7 se observa la forma de realizar esta tarea en SmarTTrace.

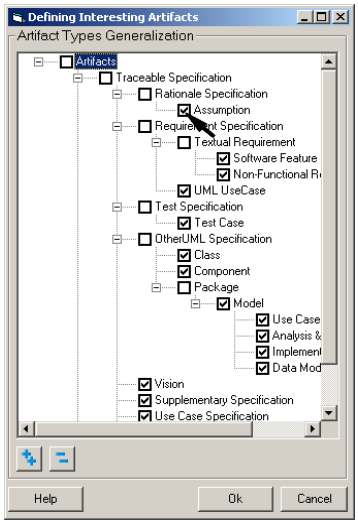


**Fig. 7.** Selecting interesting artifact types

***Tarea 4*** Se establecen los tipos de enlace de trazabilidad que son de interés para el proyecto. En nuestro ejemplo supondremos que son de interés los siguientes:

Stakeholder $\xrightarrow{\text{«responsibleOf»}}$ RUP Artifact
Stakeholder $\xrightarrow{\text{«modifies»}}$ RUP Artifact
Software Feature $\xrightarrow{\text{«traceTo»}}$ Use Case
Assumption $\xrightarrow{\text{«supports»}}$ Software Feature
Use Case $\xrightarrow{\text{«traceTo»}}$ Use Case Specification
Use Case $\xrightarrow{\text{«validatedBy»}}$ Test Case
Flow of Events $\xrightarrow{\text{«traceTo»}}$ Class
Class $\xrightarrow{\text{«traceTo»}}$ Component
Class $\xrightarrow{\text{«traceTo»}}$ Table
Class $\xrightarrow{\text{«verifiedBy»}}$ Test Case
Component $\xrightarrow{\text{«verifiedBy»}}$ Test Case

Hemos utilizado *RUP Artifact* para referirnos a cualquiera de los tipos de artefacto RUP seleccionados en la tarea 3. El estereotipo «supports» se ha introducido como un nuevo estereotipo, siendo éste una clase hija del estereotipo «rationaleOf».

A partir del conjunto de tipos de artefactos y enlaces se define la trazabilidad de interés cuya información se someterá a un posterior proceso de explotación.

En SmarTTrace se permiten definir tipos de enlaces de interés entre distintos tipos de artefactos, siempre teniendo en cuenta las restricciones impuestas por el *profile. Éste,* restringe los tipos de artefactos entre los que se puede establecer dicho tipo de enlace. Si se desea añadir un tipo de enlace de interés, SmarTTrace limita el tipo de artefactos entre los que se puede asociar el tipo de enlace. Para ello se tiene en cuenta el metamodelo de trazabilidad. En la Figura 8 se observa como el usuario añade a la lista de enlaces de interés el tipo *"verified by"* entre los tipos de artefactos *"Component"* y *"Test Case"*.

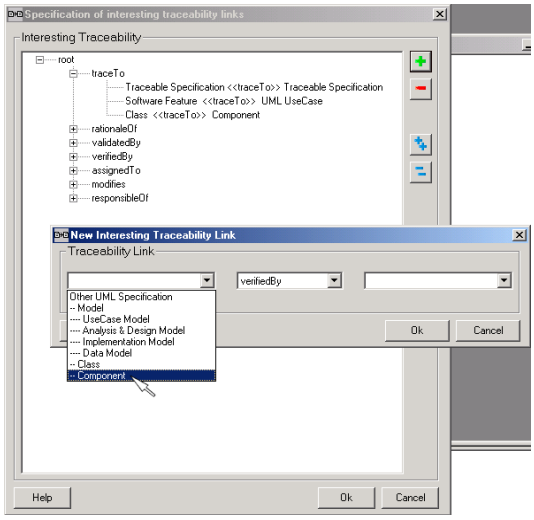


**Fig. 8.** Definición de tipos de enlaces de interés para el proyecto

## 7 Trabajos Relacionados

Las propuestas actuales de herramientas para la trazabilidad de requisitos son muy excasas, proviniendo mayoritariamente del ámbito universitario. Entre las más importantes se encuentra TOOR (*Traceability of Object-Oriented Requirements*), una herramienta presentada por Pinheiro y Goguen [11] basada en el lenguaje formal orientado a objetos FOOPS. Dicha herramienta presenta un tratamiento de artefactos especialmente orientado a la gestión de requisitos textuales. En un posterior estudio [10], Pinheiro proporciona un implementación en Java de TOOR cuyo objetivo principal es la fácil integración con otras herramientas.

Toranzo y Castro en [12] presentan una herramienta de modelado la cual proporciona facilidades para la gestión de requisitos mediante enlaces de tra-

zabilidad. Si bien, el proceso de trazabilidad que proveen esta limitado al conjunto de tipos de artefactos proporcionados por la herramienta, siendo inviable la configuración de la trazabilidad por no proveer mecanismos de extensión y adaptación del metamodelo que proponen. A su vez, la herramienta proporciona deficientas en el nivel de granularidad de las especificaciones con las que trabaja (en el ámbito de documentos y diagramas).

En el contexto de herramientas comerciales, las propuestas provistas están orientadas al tratamiento de especificaciones textuales. Pero las herramientas no proporcionan marcos homogeneos para trabajar con todos los tipos de artefactos asociados al proceso de desarrollo de software. Rational RequisitePro permite únicamente una trazabilidad básica entre especificaciones textuales y casos de uso especificados en Rational Rose, haciendo para ello uso del repositorio de este último. Cualquier otro tipo de trazabilidad con otros tipos de artefactos es imposible. Además, Rational RequisitePro provee un único tipo de enlace que se corresponde con la dependencia genérica *trace*. Otra propuesta es la provista por Telelogic DOORS. Esta herramienta permite la conexión con diversas herramientas CASE para importar los elementos de los modelos en el repositorio de la CASE hacia el entorno de gestión de requisitos. En este caso está el inconveniente añadido de tener que cambiar de entorno según se quiera gestionar requisitos o modelar y construir el software. Ambas herramientas industriales carecen de la fase de configuración de la trazabilidad del proyecto.

## 8 Conclusiones

La trazabilidad de requisitos es la clave para lograr una gestión de requisitos exitosa. Sin embargo, no existe un marco de trabajo consensuado para trazabilidad de requisitos. La ausencia de estandares conlleva la deficiencia de herramientas de gestión de requisitos que provean mecanismos adecuados para la configuración de la trazabilidad específica de un proyecto.

Este trabajo presenta SmarTTrace, una herramienta que extendiende y adapta Rational Rose mediante los mecanismos de extensibilidad provistos por dicha CASE para dar soporte a la configuración de trazabilidad. SmarTTrace hace de Rational Rose un entorno único que integra tipos de especificaciones textuales y UML, resolviendo problemas de sincronización y duplicación. Para ello, SmarTTrace extiende el metamodelo de trazabilidad de Rational Rose mediante el *profile* UML presentado en este artículo y detallado en [6]. Dicho metamodelo provee un marco de trazabilidad adaptable y extensible a las necesidades específicas de un proyecto, permitiendo definir nuevos tipos de artefactos y enlaces de trazabilidad. SmarTTrace es adaptable e independiente de cualquier proceso de desarrollo de software.

Un aspecto importante en la configuración de la trazabilidad del proyecto es el establecimiento de atributos de trazabilidad (y sus posibles valores) para cada uno de los artefactos seleccionados. Intencionalmente este tema no ha sido abordado porque consideramos que dicha tarea no presenta especial dificultad. Puede suponerse que la herramienta de gestión de requisitos ofrecerá un conjunto de

atributos predefinidos para seleccionar y asociar a cada artefacto, y que también permitirá definir nuevos atributos. Por ejemplo, en RUP para una *software feature* algunos de los atributos que se proponen son: estado (propuesta, aprobada o incorporada), beneficio (crítica, importante o útil), esfuerzo estimado, riesgo y estabilidad (para estos últimos atributos se suelen usar valores tales como: alto, medio o bajo).

Actualmente se esta trabajando en la mejora de SmarTTrace para que considere la definición de trazabilidad implícita. Otro objetivo es el estudio detallado de técnicas de explotación de la información de trazabilidad especificada durante el modelado del sistema. Dicho proceso de explotación será incorporado a SmarTTrace. La aplicación de dicha herramienta a casos prácticos abrirá nuevos retos en el proceso configuración de trazabilidad definido y en la potenciación de su utilización.

## References


1. R. Dömges and K. Pohl. Adapting Traceability Environments to Project-Specific Needs. Communications of ACM, Vol. 41, No 21, December 1998.
2. O. Gotel and A. Finkelstein. Extended Requirements Traceability: Results of an Industrial Case Study. I'n Proceedings of 3rd International Symposium on Requirements Engineering (RE97), IEEE Computer Society Press, pp. 169-178, 1997.
3. P. Haumer, M. Jarke, K. Pohl and K. Weidenhaupt. Improving reviews of conceptual models by extended traceability to captured system usage. Interacting with Computers, Elsevier Science, 13 (1) pp. 77-95, 2000. `ftp://sunsite.informatik.rwth-aachen.de/pub/CREWS/CREWS-99-16.ps.gz`
4. I. Jacobson, G. Booch and J. Rumbaugh. The Unified Software Development Process. Addison-Wesley, 1999.
5. P. Kruchten: The 4+1 View Model of Architecture. IEEE Software 12(6): 42-50 (1995).
6. P. Letelier. A framework for Requirement Traceabiliaty in UML-based Projects. Workshop on Traceability in Engineering Forms of Software Engineering, Edinburgh, UK, December 2002. (Aceptado)
7. B. Ramesh. Factors influencing requirements traceability practice. Communication of the ACM, Vol. 41, No. 12, pp. 37-44, December 1998.
8. B. Ramesh and M. Jarke. Toward Reference Models for Requirements Traceability. IEEE Transactions on Software Engineering, Vol. 27, No. 1, pp.58-93, January 2001.
9. OMG Unified Modeling Language Specification. UML 1.4 with Action Semantics, Final Adopted Specification, January 2002. `www.omg.org`
10. F. Pinheiro, An Object-Oriented Library for Tracing Requirements, WER99 - II Workshop on Requirements Engineering, event of XXVIII JAIIO - Jornadas Argentinas de Informática e Investigación Operativa, Buenos Aires, Argentina, pp 187-197, September 1999.
11. F. Pinheiro and J. Goguen. An Object-Oriented Tool for Tracing Requirements. IEEE Software, pp. 52-64, March 1996.
12. M. Toranzo and J. Castro. A Comprehensive Traceability Model to Support the Design of Interactive Systems. ECOOP Workshops 1999, pp. 283-284, Lecture Notes in Computer Science 1743, Springer-Verlag, 1999.