Trazabilidad de Requisitos Adaptada a las Necesidades del Proyecto: Un Caso de Estudio Usando Alternativamente RUP y XP

Víctor Anaya, Patricio Letelier
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n
E-46071 Valencia – España
{vanaya | letelier}@dsic.upv.es

Resumen

Facilidad de mantenimiento y alineamiento con las necesidades reales de los usuarios son desafíos comunes en el proceso de desarrollo de software de calidad. La gestión de requisitos es un proceso clave para cumplir dichos objetivos. Para realizar una exitosa gestión de requisitos, la trazabilidad de requisitos es una actividad clave. Es imprescindible disponer de herramientas, métodos y ontologías que den soporte a esta actividad. Actualmente no existe un consenso acerca de los tipos de enlaces y tipos de artefactos que el proceso de trazabilidad ha de considerar. Este artículo muestra *SharpTrace,* una herramienta que implementa un profile UML de trazabilidad basado en un metamodelo de trazabilidad. Dicho profile UML es adaptable a las necesidades específicas del proyecto, para lo cual se sigue un proceso de configuración de trazabilidad. Este artículo muestra la configuración, especificación y visualización de trazabilidad entre los artefactos de un proyecto usando *SharpTrace***.** Se incluye un caso de estudio desarrollado usando un proceso RUP y otro XP.

Palabras Clave: Trazabilidad de requisitos, gestión de requisitos, UML, proceso software

1. Introducción

La trazabilidad de requisitos es considerada un proceso imprescindible para la adecuada gestión de requisitos. El principal problema de dicho proceso es la falta de consenso en cuanto a los tipos de enlace de trazabilidad que han de considerarse, la semántica de cada uno de ellos y los tipos de artefactos entre los que se establecen dichos enlaces ([Ram98]). Otro factor clave a tener en cuenta es la adaptabilidad de la trazabilidad a las necesidades específicas de un proyecto. Con el fin de dar solución a estos problemas en [Let02] se presenta un metamodelo de trazabilidad. Dado que actualmente UML es la notación OO más popular, en dicho trabajo se define un profile UML de trazabilidad que incorpora los conceptos del metamodelo de trazabilidad. Así se consigue extender UML para integrar artefactos no-UML y enlaces de trazabilidad. La implementación de dicho metamodelo mediante un profile UML de trazabilidad permite beneficiarse de los mecanismos de extensión de UML para adaptar el marco de trazabilidad a las necesidades específicas del proyecto y además ofrecer el soporte de herramientas CASE que incluyan UML. Mediante el profile de trazabilidad se pueden definir nuevos tipos de enlaces y tipos de artefactos entre los que establecer trazabilidad. En [Ana02] se hace una primera implementación del profile extendiendo la herramienta CASE Rational Rose. Adicionalmente se definió un proceso de configuración de trazabilidad que mostraba la metodología de adaptación del proceso de trazabilidad a las necesidades específicas de un proyecto.

Un proceso de trazabilidad se divide en dos subprocesos: a) de configuración de la trazabilidad y b) de especificación y explotación de la información de trazabilidad. En este trabajo se muestra la configuración y especificación de la trazabilidad en un caso de estudio de una tienda virtual. Para ello, se ha extendido la herramienta presentada en [Ana02]. El trabajo muestra los enlaces de trazabilidad que pueden ser necesarios entre artefactos generados en el proceso de desarrollo. El enfoque descrito en este artículo es independiente de la metodología de desarrollo utilizada. El artículo ilustra la trazabilidad tanto para el desarrollo del sistema

mediante *Rational Unified Process* (RUP)[1], como para un proceso de desarrollo ágil como es *Extreme Programming* (XP)[2].

Este trabajo se estructura en cinco secciones. En la siguiente sección se presenta *SharpTrace*, una herramienta para trazabilidad de requisitos. En la tercera sección se muestra un ejemplo de seguimiento de los artefactos generados en el proceso de desarrollo haciendo uso de *SharpTrace*. En la sección cuarta se muestra el estado del arte. Finalmente, en la sección quinta se resumen las conclusiones y se definen los trabajos futuros.

2. *SharpTrace*: una herramienta para la trazabilidad de requisitos

Con la finalidad de integrar artefactos UML y no-UML se ha desallorado *SharpTrace*. *SharpTrace* es una herramienta implementada como un add-in de Rational Rose que extiende dicha CASE. *SharpTrace* implementa el profile UML de trazabilidad definido en [Let02] permitiendo a Rational Rose integrar especificaciones UML y no-UML. Con tal finalidad, *SharpTrace* proporciona un marco común de interpretación para la información de trazabilidad. Dicho marco puede ser adaptado, pudiendo definirse nuevos tipos de enlaces y artefactos de trazabilidad a partir de los ya especificados en el profile. Las extensiones se hace siguiendo un proceso de configuración de trazabilidad bien definido. Adicionalmente, *SharpTrace* dota a Rational Rose de la capacidad de definir tipos de artefactos no-UML. Así, la CASE extendida es capaz de trabajar con todo tipo de artefactos en el mismo contexto, por lo que guarda todos los artefactos de un proyecto en un mismo repositorio (fichero .mdl). La tarea de adaptación del marco de trazabilidad proporcionada por *SharpTrace* se corresponde con el subproceso de configuración de trazabilidad, que junto al de especificación y explotación de la información de trazabilidad componen el proceso de trazabilidad.

El marco de trazabilidad, que *SharpTrace* implementa mediante nuestro profile UML de trazabilidad, contiene los cuatro tipos esenciales de información para la trazabilidad de requisitos: i) especificaciones textuales, ii) especificaciones de modelado UML, iii) especificaciones de test y iv) fundamentos (ver Figura 1). A su vez, se registran las contribuciones hechas por los stakeholder (estructuras de contribución [Got97]), mediante las asociaciones *modifies* y *reponsible of*. Las asociaciones entre artefactos trazables representan distintos tipos de enlaces de trazabilidad (*traceTo*, *validatedBy*, *verifiedBy*, *assignedTo* y *rationaleOf*). La relación de agregación entre *traceable specifications* permite disponer de jerarquías de composición de tipos de artefactos.

En dicho metamodelo se registran la pre-trazabilidad y post-trazabilidad. La pre-trazabilidad está constituida por los enlaces que se establecen entre artefactos generados con anterioridad a la especificación de requisitos. La post-trazabilidad se corresponde con los enlaces entre los requisitos y los artefactos generados en posteriores fases de desarrollo (diseño, implementación, etc).

A continuación se indican los tipos de información asociada a la trazabilidad de requisitos y sus posibles usos (adaptado desde [Döm98]):

1. Los enlaces de trazabilidad entre diferentes tipos de especificaciones permiten:
   - Validar que la funcionalidad del sistema reune las expectativas del cliente y que no se ha implementado funcionalidad supérflua.
   - Realizar análisis de impacto de cambios en los requisitos. Los enlaces permiten determinar las especificaciones que se verían afectadas.
2. Las estructuras de contribución, es decir, los enlaces entre participantes en el proyecto (stakeholders) y las especificaciones permiten:
   - Mejorar la comunicación y cooperación entre los participantes del proyecto. En caso de una solicitud de cambio, los participantes que deberían estar involucrados y/o informados pueden ser determinados fácilmente.
   - Asegurar que la contribución de cada individuo es considerada y registrada.

[1] http://www.rational.com
[2] http://www.extremeprogramming.org

3. Los fundamentos asociados a las especificaciones, incluyendo alternativas, decisiones, suposiciones, etc. contribuyen a:
   - Mejorar la comprensión del sistema por parte del usuario y así su aceptación
   - 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.

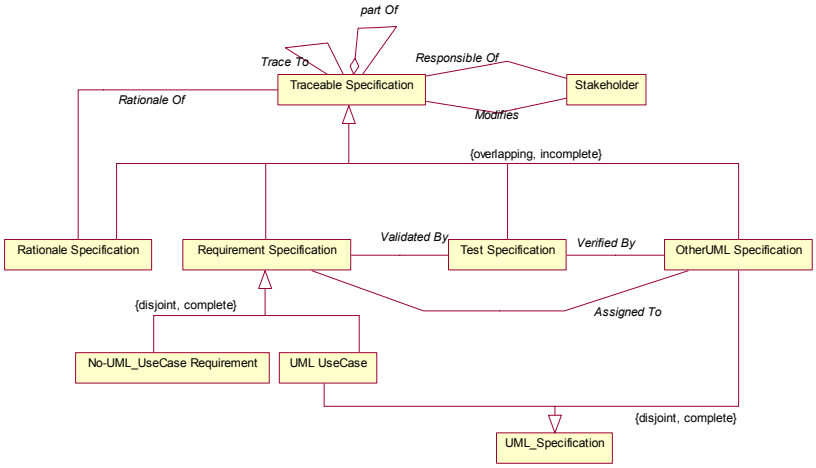

Figura 1. Metamodelo de Trazabilidad

El proceso de configuración de trazabilidad consta de cinco tareas:

1. Definir los tipos de artefactos usados en el proyecto de acuerdo con la metodología utilizada.
2. Definir las relaciones de agregación entre tipos de artefactos que pueden venir dadas por la metodología utilizada.
3. Seleccionar los tipos de artefactos que son relevantes para trazabilidad.
4. Establecer tipos de enlaces de trazabilidad relevantes para el proyecto, de entre los posibles de acuerdo a los artefactos de interés seleccionados en la tarea 3. En caso de ser necesario, definir nuevos tipos de enlaces de trazabilidad.
5. Definir criterios de derivación automática de enlaces de trazabilidad.

Aunque la definición de enlaces de interés del paso 4 acota el trabajo en cuanto a establecimiento de enlaces de trazabilidad, el esfuerzo para especificación puede ser considerable. Por esto, es interesante dotar de mecanismos de generación automática de enlaces de trazabilidad. Dichos mecanismos se basan en un conjunto de fórmulas, denominadas criterios. Dichos criterios pueden estar predefinidos en *SharpTrace* o definir nuevos criterios por el usuario en la etapa de configuración. Ejemplos de criterios básicos para derivar trazabilidad implícita son:

- **Convención de nombre**
  Sea X e Y tipos de artefactos, x, y artefactos tales que $x \in X$ e $y \in Y$. Si $X \rightarrow Y$ es un tipo de enlace de interés y algún criterio de convención de nombre se satisface entre estos artefactos, entonces se deriva el enlace de trazabilidad $x \rightarrow y$.

- **Transitividad:**
  Sea X, Y y Z tipos de artefactos, x, y, z artefactos tales que $x \in X$, $y \in Y$ y $z \in Z$. Si $X \rightarrow Y$, $Y \rightarrow Z$ y $X \rightarrow Z$ son tipos de enlaces relevantes para el proyecto, de acuerdo con el criterio de transitividad: $(x \rightarrow y) \wedge (y \rightarrow z) \Rightarrow (x \rightarrow z)$.

SharpTrace permite definir y trazar artefactos con cualquier nivel de granularidad, al contrario que otras propuestas en las que la trazabilidad sólo se establece entre tipos de artefactos a un alto nivel de granularidad (por ejemplo en [Spe98] y [Tor99]) lo cual no suele ser de gran utilidad. Por ejemplo, la relación de

trazabilidad entre un diagrama de casos de uso y un diagrama de clases no aporta mucho respecto de trazabilidad.

3. Un Caso de Estudio: configuración, definición y seguimiento mediante *SharpTrace*

Se ha desarrollado un caso de estudio de una pequeña tienda virtual. Dicha empresa tiene como objetivo vender productos electrónicos, juguetes, libros, música, videos y productos informáticos. Debido a la amplia difusión de las tecnologías de la información y al creciente uso de internet por parte de grandes comunidades de usuarios, el fin de la empresa es el de desarrollar las aplicaciones necesarias para informatizar y dinamizar los servicios y soluciones de los que dispone, así como de incrementar en la medida de lo posible los mismos. A su vez, la empresa no sólo desea vender productos, sino que también subcontratar los servicios de envíos y de proveedores. Para el desarrollo de dicho caso de estudio se han seguido dos procesos de desarrollo de software distintos, que dan lugar a ciclos de vida distintos y por lo tanto a artefactos diferentes. Los procesos de desarrollo estudiados son RUP y XP[3].

Proceso RUP de desarrollo de la Tienda Virtual

RUP es un proceso de desarrollo que se caracteriza por la gran cantidad de tipos de artefactos generados por las actividades que lo componen. Sin embargo, en RUP se hace hincapié en que no es necesario utilizar todos los artefactos y provee un subproceso específico que se encarga de la selección de los artefactos de acuerdo con las características del proyecto (disciplina *Environment*).

En primer lugar se han determinado los tipos de artefactos que van a ser usados o producidos en el proceso de desarrollo del sistema de la Tienda Virtual. A partir de dicha lista se determina, para cada uno de los tipos de artefactos que la componen, si existe una forma de representar cada concepto con los constructores que proporciona el metamodelo UML básico. En caso contrario, se comprueba si algunos de los estereotipos definidos por el profile UML de trazabilidad representan dicho concepto. Si dicho concepto no es representable por el metamodelo UML "extendido" (metamodelo UML y profile UML), se extiende el profile UML de trazabilidad definiendo un nuevo tipo de artefacto. Para ello se define el nuevo contructor como una subclase a partir del estereotipo del profile UML de trazabilidad que se asemeje más respecto al concepto que se quiere representar. En la siguiente tabla se muestra la lista de tipos de artefactos usados en el ejemplo y el estereotipo del profile UML de trazabilidad a partir del cual se ha extendido en caso de que ser necesario.

Lista de Artefactos RUP	**Estereotipo a partir del cual se define**
User Need	*<<No UML-UseCase requirement>>*
Software Feature	*<<No UML-UseCase requirement>>*
Actor	
Use Case	
Step	*<<traceable specificationt>>*
Precondition	*<<No UML-UseCase requirement>>*
Non-Functional Requirement	*<<traceable specificationt>>*
Change Request	*<<textual requirement>>*
Class	
Table	*<<traceable specification>>*
Component	
Node	
Test Case	*<<test specification>>*
Justification	*<<rationale specification>>*
Assumption	*<<rationale specification>>*

SharpTrace permite realizar dicha adaptación del profile (ver Figura 2).

[3] El artículo no se centra en la comparativa de ambos procesos. Una comparativa puede verse en [Smi01].

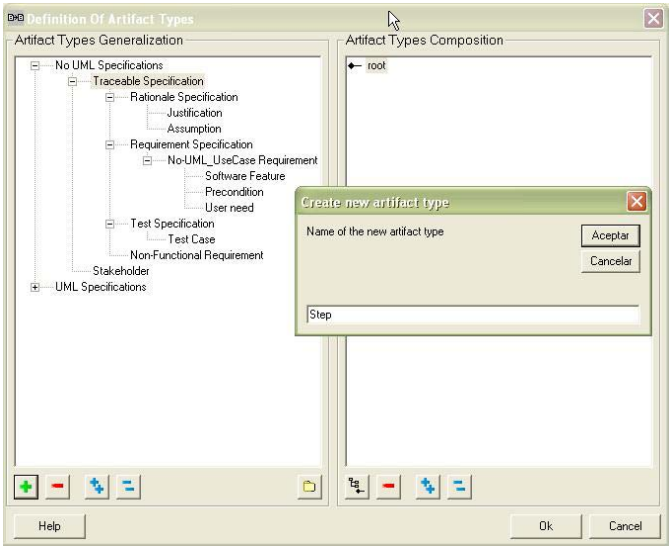

Figura 2. Adaptación del Profile UML de Trazabilidad

Para la gestión de los artefactos se pueden definir un conjunto de atributos asociados a los mismos basándose en los provistos ya por RUP. Por ejemplo, para una *software feature* algunos de los atributos que se proponen en RUP 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).

Una vez extendido el marco de trazabilidad, se definen los artefactos que componen el sistema. Para los artefactos UML se usan los métodos proporcionados por Rational Rose. Para definir los artefactos no-UML se usan los métodos proporcionados por *SharpTrace*, que se integran con la propia *CASE*. En la Figura 3 se observa cómo se han definido un conjunto de necesidades de usuario (*User Needs*).

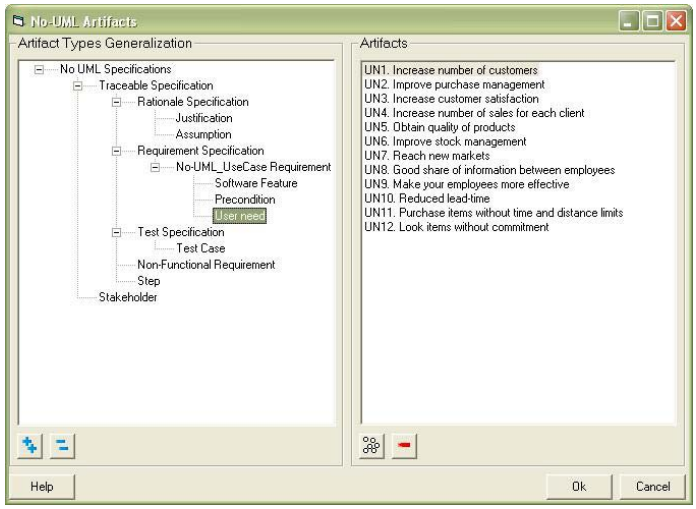

Figura 3. Definición de Artefactos No-UML

A partir de las necesidades de usuarios se definen el conjunto de características de software (*software feature*):

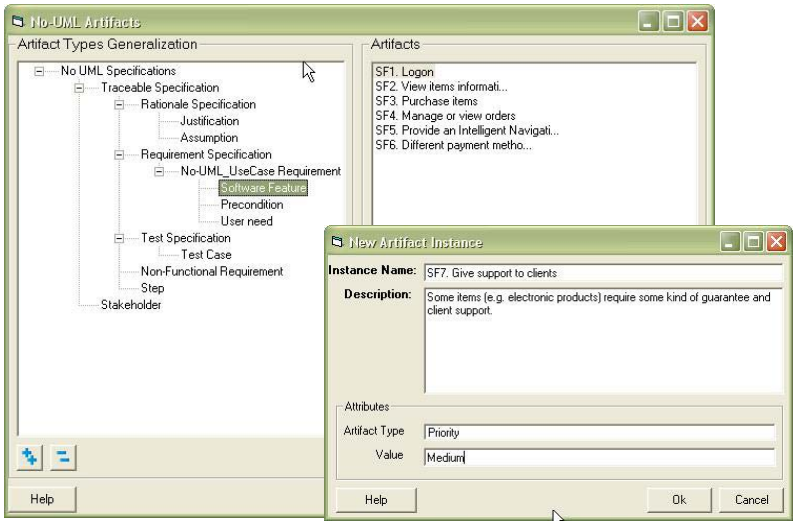

Figura 4. Definición de un Artefacto de Tipo *Software Feature*

A partir de las características del software se define la especificación de requisitos (SRS) la cual esta compuesta de requisitos funcionales y no-funcionales. Un pequeño extracto de los requisitos funcionales se muestra en la Figura 5.

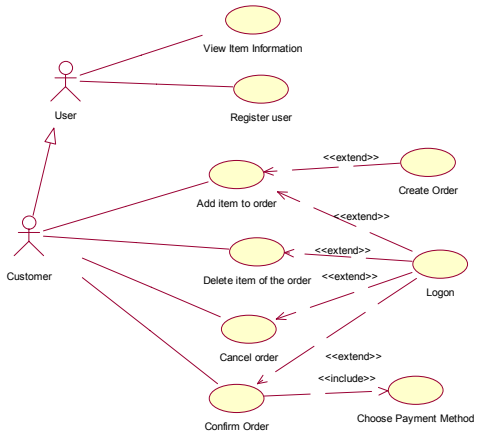

Figura 5. Conjunto de Casos de Uso de la Tienda Virtual

Tras especificar el resto de artefactos (requisitos no-funcionales, casos de test, pruebas unitarias, componentes, nodos, etc), y siguiendo con el proceso de configuración de trazabilidad, se establecen los tipos de artefactos de interés para trazabilidad entre todos los usados en el proyecto. Estos van a determinar los artefactos a partir de los cuales se van a poder establecer enlaces de trazabilidad (ver Figura 6).

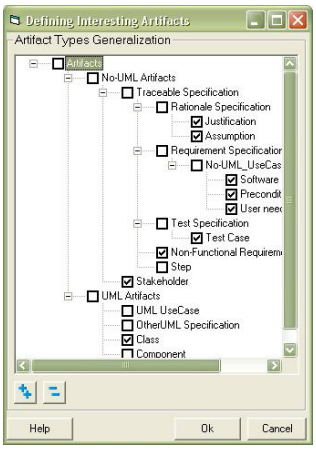

Figura 6. Tipos de Artefactos de Interés para Trazabilidad en la Tienda Virtual usando RUP

Posteriormente se establecen los tipos de enlaces de trazabilidad de interés y los pares de tipos de artefactos de interés entre los que se pueden establecer dichos enlaces. Dicha información proporciona unas guías que determinan qué tipos de enlace de trazabilidad se van a poder establecer desde los artefactos de interés. En la Figura 7 se muestra cómo se indica en *SharpTrace* los tipos de enlaces que se consideran relevantes para el proyecto, se han dejado los proporcionados por *SharpTrace* por defecto (los indicados en el metamodelo).

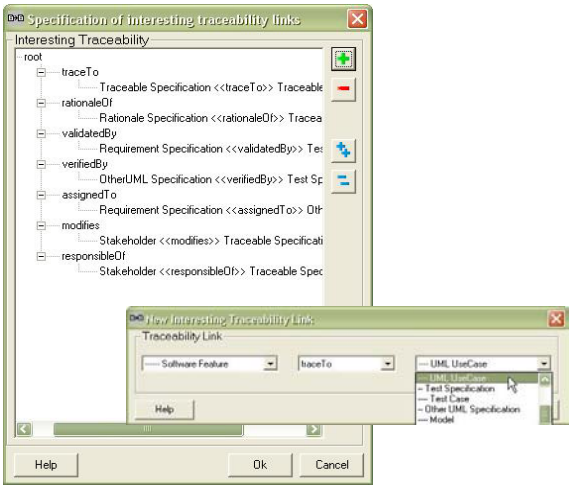

Figura 7. Tipos de Enlaces de Trazabilidad de Interés en la Tienda Virtual

A su vez, es posible definir nuevos tipos de enlaces de trazabilidad a partir de los predefinidos en el profile UML de trazabilidad. Los nuevos tipos de enlaces de trazabilidad heredan las restricciones de los tipos de enlaces de trazabilidad a partir de los cuales se definen, pudiendo establecerse restricciones más severas. La variedad de tipos de enlaces de trazabilidad permite realizar un análisis de la información de trazabilidad más rica que la que proporciona el enlace clásico *traceto*.

Una vez finalizado el proceso de configuración de trazabilidad se establecen los enlaces de trazabilidad entre artefactos generados en el proceso de desarrollo. A partir de aquellos artefactos cuyo tipo es de interés, *SharpTrace* habilita la opción de especificación de enlaces de trazabilidad. A partir del artefacto seleccionado se muestran los tipos de enlaces de interés para trazabilidad que se pueden establecer de acuerdo al profile UML de trazabilidad (ver Figura 8). Seleccionando cada tipo de enlace se visualizan los artefactos alcanzables de acuerdo con el profile UML. Los enlaces de trazabilidad poseen una direccionalidad. Por lo tanto, los enlaces de un artefacto se pueden clasificar en entrantes o salientes (*from* y *to* respectivamente). Los enlaces entrantes son enlaces que se realizan desde un artefacto determinado al artefacto seleccionado. Los enlaces salientes son los enlaces que se realizan desde el artefacto seleccionado hacia otros artefactos. A partir

de la lista de artefactos con los que se puede establecer un determinado tipo de enlace de trazabilidad, el usuario selecciona los artefactos con los que desea establecer enlace. Así, en el ejemplo de la tienda virtual, el gestor de ventas es el responsable de requerir las necesidades de usuario 1, 2, 3 y 4 (ver dichas necesidades en la Figura 8).

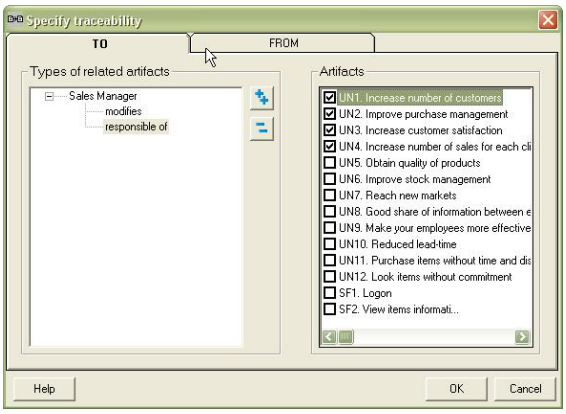

Figura 8. Enlaces salientes de tipo *responsible of* desde *Sales Manager*

Con el fin de guiar más el proceso de trazabilidad, el usuario ha de especificar de forma más detallada los tipos de enlaces de interés entre artefactos. En este ejemplo, el usuario sólo quiere establecer desde los *stakeholders* enlaces de trazabilidad *responsible of* con las necesidades de usuario, puede indicarlo en la tarea 4 del proceso de configuración. En la Figura 8 solamente se hubieran visualizado las necesidades de usuario, y no las software feature, ni el resto de artefactos de interés traceables.

Proceso XP de desarrollo de la tienda Virtual

XP es un proceso ágil que se caracteriza por el minimalismo en cuanto a artefactos y actividades en el desarrollo de software. XP es un proceso que remarca la importancia del trabajo en equipo y la participación del cliente, y que centra esfuerzos en la programación. XP es adecuado principalmente para proyectos cuyos requisitos son muy dinámicos, aunque este proceso tiene importantes restricciones de aplicabilidad [Smi01].

En el ejemplo de la tienda virtual manteniendo las características del proyecto hemos establecido un desarrollo XP usando los artefactos: *user story*, *task*, *component*, *acceptance test*, *unit test* y *user*.

El proceso de configuración de la trazabilidad es análogo al explicado anteriormente en el caso de RUP. En primer lugar se extiende el profile UML de trazabilidad con aquellos tipos de artefactos no representables por el profile UML de trazabilidad junto con el metamodelo UML básico. Tras dicha tarea se configuran los tipos de artefactos de interés para trazabilidad. En la Figura 9 se puede observar el profile UML de trazabilidad extendido con los nuevos artefactos, así como la selección de aquellos artefactos interesantes para el proceso de trazabilidad.

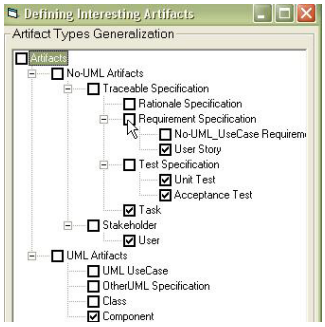

Figura 9. Tipos de Artefactos de Interés para Trazabilidad en la Tienda Virtual usando XP

Para finalizar la configuración se indican los tipos de enlaces de interés entre tipos de artefactos (ver Figura 10). En este caso de estudio se considera relevante la trazabilidad entre *User Stories* con *Tasks*, *Components* y *Acceptance Tests*, así como el enlace de trazabilidad entre *Components* y *Unit Tests*.

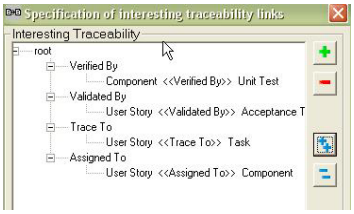

Figura 10. Tipos de Enlaces de Trazabilidad de Interés usando XP

Tras la configuración de la trazabilidad, se desarrolla el software requerido. Como resultado del empleo de XP como proceso de desarrollo se obtienen un conjunto de artefactos instancias de algunos de los tipos definidos previamente. A partir de los artefactos cuyo tipo es de interés para el proceso de trazabilidad se pueden establecer enlaces de trazabilidad. Dichos enlaces serán del tipo indicado en la configuración. En la Figura 11 se muestran los enlaces establecidos entre una *User Story* y sus *Components*. Hay que tener en cuenta que *User Story* → *Component* es un enlace de trazabilidad de interés.

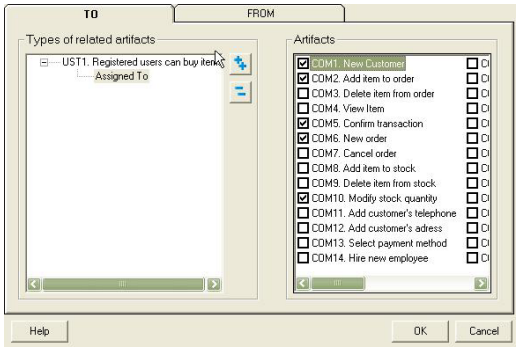

Figura 11. Trazabilidad desde una *User Story*

4. Trabajos relacionados

El marco de trazabilidad que proporciona *SharpTrace* es un marco esencial a partir del cual se puede derivar un conjunto de modelos referenciales de trazabilidad orientados a las necesidades específicas de cada uno de los *stakeholders,* proporcionando un símil a los modelos referenciales de Ramesh y Jarke en [Ram01] o de Toranzo y Castro en [Tor02], los cuales no ofrecen mecanismos específicos para adaptación. Además, con nuestro marco se proporcionan guías semánticas basadas en la definición de un metamodelo de trazabilidad implementado como un profile UML.

Existen herramientas de trazabilidad, como la presentada en [Pin96] y [Pin99]. En el primero de estos trabajos se presenta TOOR (*Traceability of Object-Oriented Requirements*) una herramienta cuyo principal objetivo es el seguimiento de artefactos. El principal inconveniente de TOOR es su limitado uso, siendo necesario emplear otras herramientas para realizar tareas de elicitación o desarrollo del software. Al contrario, *SharpTrace* se beneficia de Rational Rose para integrar las fases de análisis, diseño y gestión de artefactos en un único contexto. A pesar de que TOOR proporciona un lenguaje formal, FOOPS, para la definición del marco de trazabilidad, no proporciona una propuesta de metamodelo básico, ni un proceso de configuración de trazabilidad.

RequisitePro es una herramienta desarrollada por Rational Software para la gestión de requisitos, que proporciona trazabilidad. *RequisitePro* dota al usuario de un conjunto de plantillas para la gestión de requisitos. En *RequisitePro* a todos los artefactos se les denomina requisitos (incluso a los elementos de etapas de diseño que el usuario puede definir), lo cual es confuso. *RequisitePro* permite adaptar dichas plantillas a las necesidades del proyecto, pero carece de una semántica subyacente. No se proporcionan ni

restricciones, ni mecanismos para indicar restricciones que permitan guiar al usuario en la especificación de trazabilidad. *RequisitePro* solamente proporciona un tipo de enlace de trazabilidad, el tradicional *trace to*. La herramienta permite una pobre integración con la CASE Rational Rose, permitiendo establecer únicamente enlaces de trazabilidad entre los requisitos definidos *en RequisitePro* y los casos de usos definidos en Rational Rose, no siendo posible establecer enlaces con otros tipos de artefactos de análisis y diseño, ni tampoco definir niveles de granularidad más finos.

5. Conclusiones

En este trabajo se ha presentado *SharpTrace*, una herramienta que implementa mediante un profile UML una propuesta de metamodelo de trazabilidad presentada en [Let02]. En dicho metamodelo, teniendo en cuenta el extenso uso de UML como notación OO, se marca como objetivo la integración de artefactos UML y no-UML. *SharpTrace* se beneficia de los mecanismos de extensión UML, permitiendo adaptar el marco de trazabilidad a las necesidades específicas del proyecto. Con tal finalidad, la herramienta proporciona un proceso de configuración de trazabilidad compuesta de un conjunto de tareas que guían al usuario. *SharpTrace* permite definir nuevos tipos de artefactos y enlaces de trazabilidad. La herramienta es independiente del proceso de desarrollo empleado. Para demostrar la adaptabilidad e independencia respecto al proceso de desarrollo de software, en el artículo se ha mostrado el uso de *SharpTrace* en el desarrollo de una tienda de venta online siguiendo, alternativamente, un proceso de desarrollo RUP o XP. Se ha ilustrado como *SharpTrace* es fácilmente adaptable a las necesidades específicas del proyecto. En el ejemplo se muestran los mecanismos para establecer enlaces de trazabilidad entre artefactos de una forma intuitiva y guiada gracias al profile UML de trazabilidad. *SharpTrace* dota a Rational Rose de mecanismos para crear tipos de artefactos no UML y trabajar junto con los UML en un mismo repositorio, proporcionando un único contexto. Otro factor clave es la posibilidad de definir y seguir la vida de artefactos con cualquier nivel de granularidad, permitiendo realizar análisis de la información de trazabilidad más precisos.

Estamos trabajando en la implementación de mecanismos que permitan al usuario introducir criterios para la derivación automática de enlaces de trazabilidad. A su vez, es necesario hacer un estudio que permita dotar a la herramienta de mecanismos de análisis de la información de trazabilidad capturada más complejos que los simples análisis matriciales que dota *Rational RequisitePro*. Para ello se consideran los múltiples tipos de enlace de trazabilidad, así como los atributos de calidad de cada tipo de artefacto.

Esperamos tener a mediano plazo una versión completamente operativa de la herramienta y ofrecerla gratuitamente mediante difusión, especialmente en los foros de Rational. A su vez, con el propósito de recibir comentarios, tenemos contemplada la aplicación de la herramienta en proyectos reales para conseguir su validación.

6. Referencias

[Ana02] Anaya, V., Letelier, P., SmarTTrace: Una Herramienta para Trazabilidad de Requisitos en Proyectos basados en UML, Proceedings of the V Workshop on Requirements Engineering, pp. 210-224, Valencia, España, Noviembre 2002.

[Döm98] Dömges, R., Pohl, K., Adapting Traceability Enviroments to Project-Specific Needs. Communications of ACM, Vol 41, No 21, December 1998.

[Got97] Gotel, O., Finkelstein, A., Extended Requirements Traceability: Results of an Industrial Case Study. Proceedings of 3rd International Symposium on Requirements Engineering (RE97), IEEE Computer Society Press, pp.169-178, 1997.

[Let02] Letelier, P., A Framework for Requirements Traceability in UML-based Projects. 1st International Workshop on Traceability in Emerging Forms of Software Engineering. In conjunction with the 17th IEEE International Conference on Automated Software Engineering, U.K. http://ase.cs.ucl.ac.uk/, Septiembre 2002

[Pin99] Pinheiro, F., Goguen, J., An Object-Oriented Tool for Tracing Requirements. IEEE Software, pp. 52-64, March 1996.

[Pin96] Pinheiro, F., 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.

[Ram01] Ramesh, B., Jarke, M., Toward Reference Models for Requirements Traceability. IEEE Transactions on Software Engineering, Vol. 27, No. 1, pp.58-93, January 2001.

[Ram98] Ramesh, B., Factors influencing requirements traceability practice. Communication of the ACM, Vol. 41, No 12, pp. 37-44, December 1998.

[Smi01] Smith, J., A comparison of RUP and XP, Rational Software White Paper, 2001.

[Spe98] Spence, I., and Probasco, L., "Traceability Studies for Managing Requirements with Use Cases". Rational Software White Paper, 1998. www.rational.com/products/whitepapers/022701.jsp

[Tor02] Toranzo, M., Castro, J., Uma Proposta para Melhorar o Rastreamento de Requisitos, Proceedings of the V Workshop on Requirements Engineering, pp. 194-209, Valencia, España, Noviembre 2002.

[Tor99] Toranzo, M., Castro, J., 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.