Análisis de impacto en requisitos soportado de forma semi-automática y en un marco de desarrollo TDD basado en pruebas de aceptación

María Company[1], Patricio Letelier[1], Mª Isabel Marante[1] y Francisco Suárez[2]
[1] Dept. de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia
{mcompany, letelier, mmarante}@dsic.upv.es

[2] Aphelion Soluciones Informáticas S.L
Torrent, 46900 Valencia
francisco.suarez@addinformatica.com

Abstract. Conseguir un ratio positivo entre el esfuerzo invertido en trazabilidad y la rentabilización en explotación de dicha información es muy difícil. Por un lado, la recolección y mantenimiento manual de relaciones de trazabilidad constituye un gran obstáculo, y por otro, la explotación que usualmente se consigue es muy rudimentaria. Una de las principales motivaciones para mantener trazabilidad en un producto software es explotarla para realizar Análisis de Impacto de cambios en el producto. El estado del arte en Análisis de Impacto en las últimas dos décadas prácticamente no ha cambiado. Si bien existen propuestas del ámbito de investigación, hasta dónde sabemos, la aplicación industrial de Análisis de Impacto es escasa y poco satisfactoria. En este trabajo se presenta un enfoque innovador que gracias a la detección automática de relaciones de interdependencia entre artefactos salva el mayor desafío que presenta la trazabilidad: la recolección y mantenimiento de las relaciones entre artefactos. Aunque nuestro enfoque se restringe sólo al ámbito de artefactos de tipo requisitos y pruebas de aceptación, resulta muy útil en la práctica pues resuelve situaciones usuales en las cuales se necesita conocer el impacto de cambios en el producto. Este enfoque está incorporado en la metodología y herramienta TUNE-UP. Las relaciones entre requisitos son automáticamente registradas y mantenidas actualizadas por TUNE-UP, y se explotan de forma semi-automática para hacer Análisis de Impacto, indicando al equipo de desarrollo aquellos elementos que pudiesen verse afectados por un cambio que se prevé realizar en el producto.

Keywords: Análisis de Impacto, Pruebas de Aceptación, TDD, Ingeniería de Requisitos, Trazabilidad

1 Introducción

El Análisis de Impacto de cambios en los requisitos tiene como propósito prever los efectos que tendrá un cambio propuesto, identificando qué elementos se verán afectados y de qué forma. Por ejemplo, cambiar el tipo de dato de un campo en una IU puede conllevar un cambio en un campo de una tabla y esto a su vez puede afectar a todas las otras IUs que presenten dicho campo. Así, la información necesaria para realizar análisis de impacto son las relaciones entre los artefactos del producto, incluyendo cualquier tipo de especificación y en particular las piezas de código fuente y/o elementos de la base de datos. El Análisis de Impacto es especialmente útil durante el mantenimiento del producto, pero también puede ser de ayuda durante su desarrollo [2]. Los resultados del Análisis de Impacto apoyan decisiones en diferentes ámbitos del desarrollo de software:

- Planificación: ayuda a comprender el alcance de un determinado cambio, lo cual contribuye a una mejor estimación de esfuerzo y evaluación de la conveniencia u oportunidad de dicho cambio.
- Definición del cambio: ayuda en la especificación del cambio para que ésta sea completa y consistente, evitando sorpresas posteriores durante su implementación.
- Programación: Disponer tempranamente de información más abstracta del cambio, aunque sea menos detallada, resulta útil para que el programador pueda hacer un diseño muy preliminar y estimación del esfuerzo de programación asociado al cambio.
- Testeo: conociendo las partes del producto que pueden verse afectadas por un cambio, se puede tomar mejores decisiones respecto de las pruebas de regresión que se deben aplicar después de implementar dicho cambio.

El Análisis de Impacto se basa en las relaciones de trazabilidad existentes entre los artefactos que especifican el producto. La técnica tradicional para establecer relaciones de trazabilidad y explotarlas mediante Análisis de Impacto son las Matrices de Trazabilidad [10, 11], en las cuales se cruzan dos tipos de artefactos marcándose las celdas que corresponden a una relación *trace to* o *trace from*. Por ejemplo, se suelen elaborar matrices entre Features y Casos de Uso, Casos de Uso y Clases, Clases y Componentes, etc. El alto esfuerzo invertido en especificación y mantenimiento de relaciones de trazabilidad es uno de los principales obstáculos para conseguir un ratio positivo entre el esfuerzo invertido y el aprovechamiento de la información resultante [17]. Esto ha motivado la investigación en mecanismos automatizados para descubrir y evaluar relaciones de trazabilidad candidatas [20].

En cuanto a la explotación de información, en las Matrices de Trazabilidad se muestran celdas marcadas *Suspect* (sospechosas), cuando se modifica un artefacto del lado *trace from* de una relación. Cuando se utiliza este enfoque para trazabilidad y Análisis de Impacto los artefactos suelen ser de gránulo muy grueso (IUs, Casos de Uso, Clases, Componentes) con lo cual la información de impacto es demasiado general y se necesita un trabajo manual más detallado para conocer con mayor precisión el impacto del cambio.

TUNE-UP es una metodología y herramienta para la gestión ágil de proyectos de desarrollo y mantenimiento de software. TUNE-UP se ha ido refinando en el trabajo diario de una PYME de desarrollo de software desde hace más de seis años. TUNE-UP incluye un enfoque innovador para la gestión de requisitos denominado TDRE [9] (Test-Driven Requirements Engineering) el cual se fundamenta en el hecho de especificar los requisitos mediante Pruebas de Aceptación (PAs). De esta forma un requisito y sus PAs son parte del mismo artefacto, lo cual conlleva ventajas tales como: se evita el esfuerzo asociado a derivar pruebas de aceptación a partir de requisitos, el mantenimiento de los requisitos se reduce a añadir, modificar o eliminar pruebas de aceptación, y los requisitos son directamente verificables por estar especificados como pruebas. En este contexto, cuando nos planteamos ofrecer un apoyo específico para Análisis de Impacto nos dimos cuenta que podíamos ofrecer una solución novedosa, pragmática, y además con soporte automatizado para la identificación y mantenimiento de las relaciones de trazabilidad. Nuestra motivación no era determinar el impacto de un cambio a través de todos los tipos de artefactos, lo que queríamos conocer en detalle era el efecto que un cambio en un requisito podía tener en el resto de los requisitos. Es decir, pretendíamos apoyar la especificación del cambio, evitando detectar tardíamente que dicha especificación estaba incompleta o era inconsistente.

El objetivo de este trabajo es presentar nuestro enfoque para Análisis de Impacto en el marco de TDRE y de TUNE-UP. Este trabajo está organizado en 4 secciones. En la siguiente sección se presentan los elementos básicos de TUNE-UP y del enfoque TDRE, el marco de trabajo para nuestra propuesta. La sección 3 describe brevemente la infraestructura incorporada en TUNE-UP para soportar Análisis de Impacto. En la sección 4 se detalla nuestro enfoque para realizar Análisis de Impacto. A continuación, en la sección 5 se comentan algunos trabajos relacionados, y finalmente en la sección 6 se presentan las conclusiones.

2 TUNE-UP y TDRE

TUNE-UP mezcla aspectos ágiles y tradicionales ofreciendo una solución metodológica, pragmática y detallada. TUNE-UP permite escalabilidad en la forma de trabajo del equipo desde enfoques más sencillos como Scrum o XP hasta otros más exigentes, por ejemplo, en el ámbito de CMMI. TUNE-UP utiliza un modelo de proceso iterativo e incremental con entregas frecuentes de versiones del producto. Cada iteración está compuesta por Unidades de Trabajo (Work Units – WUs). Una WU es un cambio en el producto (nuevo requisito, mejora o fallo). TUNE-UP incorpora TDRE [9], un enfoque en el cual las Pruebas de Aceptación (PAs) dirigen el proceso de desarrollo. Los requisitos en TUNE-UP son especificados mediante PAs. En TUNE-UP el cliente con la ayuda del analista es el encargado de definir las WUs en términos de PAs. Posteriormente el programador escribe código para satisfacer las PAs, y por último, el Tester establece combinaciones de datos para cada una de las PAs y las aplica para garantizar que el comportamiento del producto cumple con su especificación. En TUNE-UP la estructura de requisitos se representa como un grafo acíclico dirigido cuyos nodos son contenedores de PAs. Como se muestra en la Fig. 1, un cambio en el comportamiento del producto viene dado por una WU, la cual afecta

a uno o más nodos de la estructura de requisitos del producto, añadiendo, modificando o eliminando PAs.

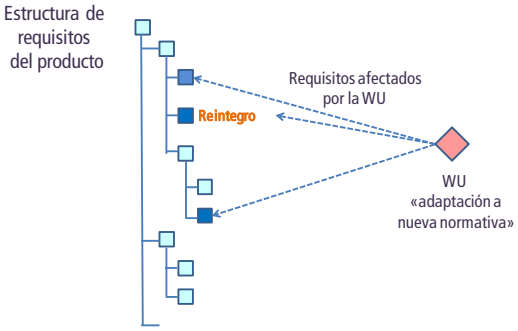

Fig. 1. Estrutura de requisitos del producto

En TUNE-UP el módulo Gestor de Unidades de Trabajo (Work Unit Manager - WUM) ofrece funcionalidad para apoyar a los agentes en su trabajo con una WU. Desde el WUM se puede acceder al Product Change Scope, donde se establece qué nodos se verán afectados y se definen los cambios en las PAs de dichos nodos. En la Fig. 2 se muestra parte del WUM, la pestaña Product Change Scope, donde a la izquierda el treview representa el grafo con la estructura de requisitos del producto. El nodo *Reintegro* está seleccionado y en el panel de la derecha, en la pestaña Acceptance Test se muestran todas las PAs definidas en dicho nodo y en color de fondo oscuro se destacan las PAs que en el contexto de la WU cargada en el WUM, se están (en este ejemplo) añadiendo a dicho nodo.

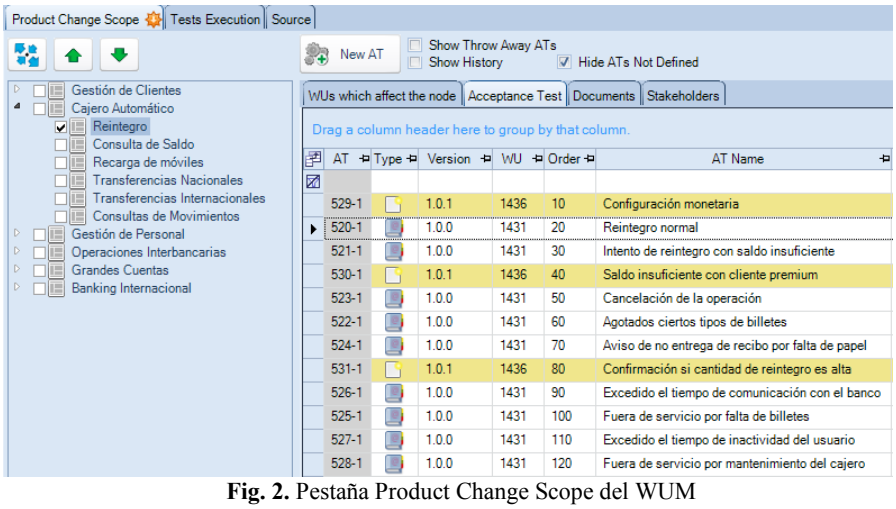

Fig. 2. Pestaña Product Change Scope del WUM

2.1 Pruebas de Aceptación y referencias a Entidades

Una PA tiene como propósito demostrar al cliente el cumplimiento parcial o total de un requisito del software. Una PA describe un escenario de ejecución o de uso del

sistema desde la perspectiva del cliente. Un requisito puede contener una o más PAs y éstas pueden estar asociadas tanto a requisitos funcionales como a no funcionales. A continuación se presenta como ejemplo la definición de la PA "Intento de reintegro con saldo insuficiente" (se trata del contexto de la funcionalidad Reintegro en un cajero automático).

Condición
Cliente del tipo normal
Cliente con saldo positivo
Acceder a ventana de reintegro
Pasos
Introducir cantidad mayor que el saldo
Resultado esperado
Se muestra el mensaje "Saldo insuficiente"
Se ofrece nueva introducción

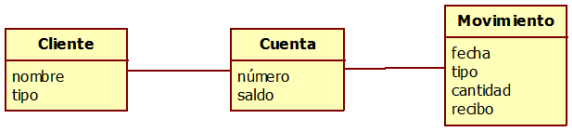

Fig. 3. Modelo de dominio de ejemplo

En TUNE-UP la definición de una PA se compone de tres apartados: Condiciones, Pasos y Resultado. El cuerpo de cada apartado se escribe en lenguaje natural pero referenciando entidades del modelo de dominio del producto. Esto nos permite establecer automáticamente relaciones entre PAs que tienen referencia a las mismas entidades. Una entidad puede contener atributos o relaciones con otras entidades, el modelo de dominio que vamos a utilizar en este trabajo se muestra en la Fig. 3.

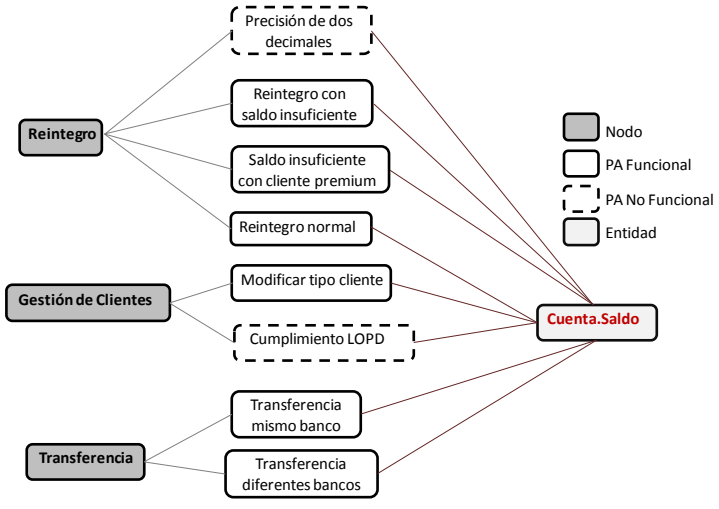

Fig. 4. Representación gráfica de PAs, Nodos y Entidades

Más adelante, en la Fig. 5 veremos el mismo ejemplo de la PA antes mostrada pero gestionada por nuestra herramienta e incluyendo referencias a entidades del modelo de dominio. En la sección 3 se mostrará cómo se referencian las entidades en la definición de una PA con la ayuda de *intellisense*, el cual nos ofrece facilidades para navegar por la estructura del modelo de dominio, introduciendo las referencias correspondientes.

En la Fig. 4 se muestra un diagrama que ilustra las relaciones entre el atributo Saldo de la entidad Cuenta y las PAs de los nodos Reintegro, Gestión de Clientes y Transferencia. La relación entre una PA y una entidad surge cuando en la especificación de dicha PA, en alguno de sus tres apartados, se ha referenciado dicha entidad.

3 Infraestructura para Análisis de Impacto

En esta sección se describe la infraestructura software que se ha implementado en TUNE-UP para dar soporte a la especificación de PAs con referencias a entidades.

3.1 BD orientada a grafos

La capa de persistencia en nuestra versión actual está soportada por una base de datos orientada a grafos (BDOG). Una característica de las BDOG es que la información más importante no se encuentra en los elementos, sino en las relaciones entre estos mismos. Las BDs Relacionales dan prioridad a los datos, y las relaciones son una forma de navegar entre ellos. Las BDOG permiten gestionar grandes volúmenes de datos de diferentes tipos y relaciones, recuperándolos de forma sencilla (sin tener que hacer una gran cantidad de Joins) gracias a la teoría de grafos. Además, las BDOG son muy flexibles respecto a los cambios que se pueden realizar en la estructura, no imponen una estructura común a ítems que normalmente en una BD Relacional tendrían que compartir atributos, y los registros pueden ser de longitud variable evitando tener que definir un tamaño. De momento sólo relacionamos entidades-nodos-pruebas, pero en un futuro podríamos relacionar más tipos de términos, es decir, desconocemos cómo evolucionará nuestro metamodelo. Otra motivación adicional para usar una BDOG es la futura visualización que queremos ofrecer para el Análisis de Impacto, la cual se ajustaría de forma más natural a dicha estructura.

Las BDOG más populares son: Neo4j (neo4j.org), Dex (sparsity-technologies.com/dex.php), HyperGraphDB (hypergraphdb.org) e Infogrid (infogrid.org). Elegimos Neo4j porque es open source, tiene una comunidad bastante activa, entrega versiones continuamente y especialmente porque permite la comunicación con otras tecnologías mediante servicios REST. Así, aunque Neo4j está en su totalidad implementada en Java, se puede interactuar con ella desde otros lenguajes (en nuestro caso interactuamos desde C#).

3.2 Gestión del Modelo de Dominio

Para gestionar el modelo de dominio (almacenado en la BDOG) se ha implementado un sencillo Gestor de Entidades (Fig. 5). En él se pueden gestionar las Entidades, sus Atributos y establecer relaciones entre entidades.

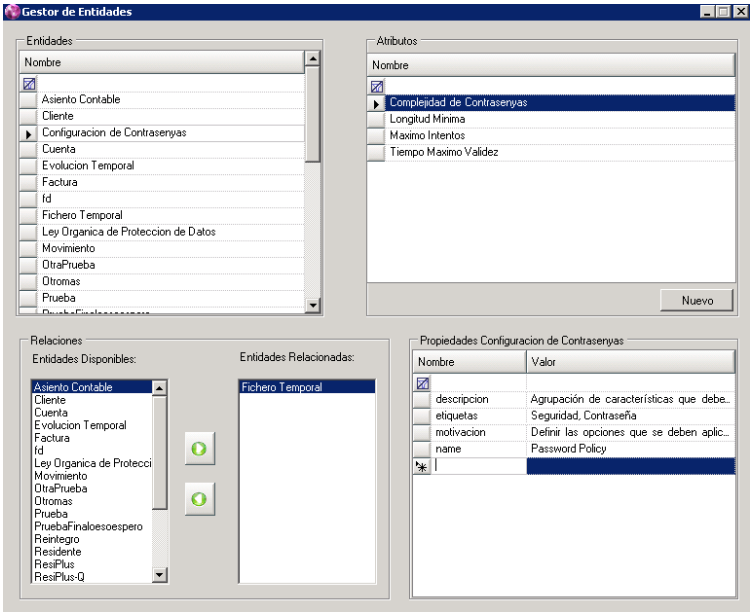

Fig. 5. Gestor de Entidades del Modelo de Dominio

3.3 Parser para Pruebas de Aceptación

El mecanismo *intellisense* permite fácilmente acceder a las entidades del modelo de dominio al introducir texto en cualquiera de los apartados de una PA. Hemos implementado un *intellisense* tal como se ofrece en los entornos de desarrollo: se activa con ctrl + space, se quita con ESC, al escribir se filtra el contenido y con el punto se navega al nivel de atributos de la entidad o hacia las entidades relacionadas. Cuando se selecciona una entidad automáticamente se introduce en el cuadro de texto con un formato diferente (itálica y color de fuente azul). Para detectar las entidades en el texto de los diferentes apartados de la PA se ha implementado un parser del contenido de cada apartado de la PA (en formato rtf), en el cual se identifican las entidades y atributos referenciados.

Cuando una PA contiene en su especificación alguna referencia a una entidad, automáticamente al aceptar los cambios se crea la relación PA-Entidad en la BDOG. Utilizamos atributos en las relaciones para establecer el apartado en el cual se referencia la entidad dentro de la PA. Esto nos permite identificar dependencias más específicas entre PAs, lo cual veremos en detalle en la siguiente sección.

En la Fig. 6 se muestra cómo se introduciría en TUNE-UP la definición de la prueba "Intento de reintegro con saldo insuficiente" referenciando las entidades del modelo de dominio. En itálica se observan dichas referencias. Puede observarse que aunque se ha modificado la especificación original incluyendo las referencias a entidades, la especificación mantiene prácticamente el mismo grado de flexibilidad y legibilidad del lenguaje natural.

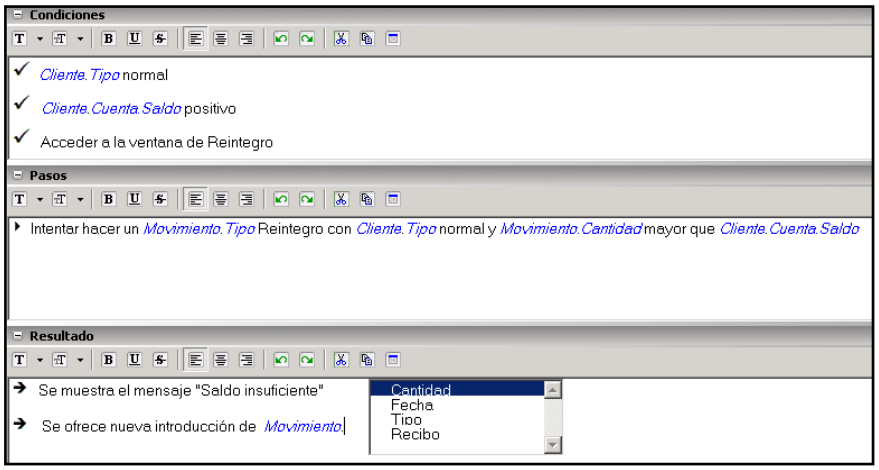

Fig. 6. Ejemplo de PA con referencias a entidades

4 Análisis de Impacto

En TUNE-UP el Análisis de Impacto está centrado en el comportamiento externo del producto, es decir, queremos determinar el conjunto completo de requisitos que pueden verse afectados o involucrados por un cambio. En TUNE-UP las PAs definen el comportamiento actual del producto, pero también un cambio asociado a una WU se especifica en términos de PAs añadidas, modificadas o eliminadas.

En nuestra propuesta gracias a las relaciones creadas de forma semiautomática del tipo PA-Entidad podemos hacer Análisis de Impacto de dos tipos, los cuales hemos denominado Interdependencia Simple y Dependencia Causa-Efecto. Estos dos tipos de Análisis de Impacto se explican a continuación.

4.1 Interdependencia Simple

En TUNE-UP los requisitos se corresponden con nodos de la estructura del producto y el comportamiento asociado a cada nodo está determinado por el conjunto de PAs contenidas en el nodo. Cuando se van a añadir, modificar o eliminar PAs (que tendrán ciertas referencias a entidades), todas las otras PAs (del mismo nodo o de otros nodos) que tengan referencias hacia las mismas entidades deberían ser revisadas para verificar si se verán afectadas. Si posteriormente se determina que efectivamente el cambio afecta a una o varias PAs, puede que dicho cambio deba propagarse y dichas

PAs afectadas también tengan que ser modificadas, o que la evaluación del impacto nos lleve a desistir del cambio y buscar una alternativa.

En la Fig. 7 se muestra un ejemplo genérico de cómo funciona la interdependencia simple. En este ejemplo tenemos dos nodos (N1 y N2) con sus respectivas PAs. Las PAs: PA1, PA2 y PA3 referencian a la Entidad (E), por lo tanto estas tres pruebas están relacionadas entre ellas. Si se hiciera un cambio en cualquiera de ellas sería conveniente que se revisarán las otras dos. Además, como las PAs se encuentran en diferentes nodos (N1 y N2) se deduce que dichos nodos están relacionados, con lo cual de forma más global se podría deducir que un cambio en uno de dichos nodos podría afectar al otro.

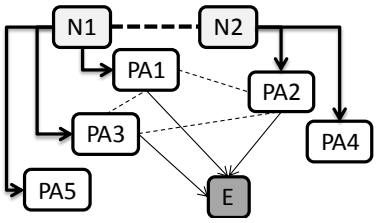

Fig. 7. Ejemplo de Interdependencia Simple

Resulta evidente que lo más adecuado para visualizar las relaciones es un grafo, similar al de la Fig. 7. Sin embargo, como primera versión hemos presentado la información en grids. Se ofrecen dos vistas de explotación, en la primera, dada una PA se muestran otras PAs que se relacionan con ella a través de una entidad. En la otra vista (Fig. 8), teniendo seleccionado un nodo, se muestra qué otros nodos del producto se relacionan con él, desplegando además las PAs que hacen que los nodos se relacionen.

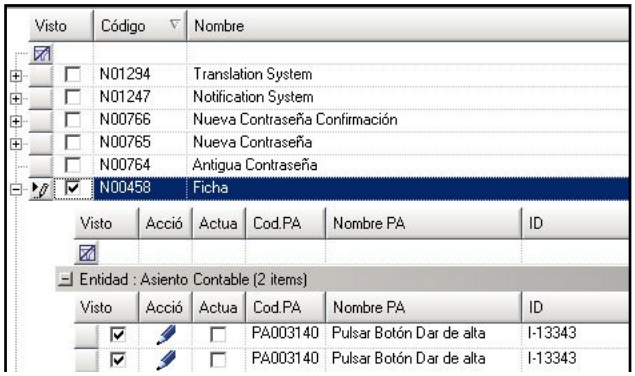

Fig. 8. Explotación Análisis de Impacto usando Interdependencia Simple

En la Fig.8, en el segundo nivel del grid se muestran las PAs que dada su referencia a una misma entidad hacen que se relacionen los nodos. Además, se proveen casillas para que se marque el nodo como revisado. Cuando esto se hace, automáticamente se marcan como revisadas todas las PAs del nodo. También se puede ir prueba a prueba revisándolas y cuando estén todas las pruebas del nodo marcadas se marcará

automáticamente el nodo como revisado. Haciendo doble clic en una PA se accede a un formulario con todos los detalles de la especificación de dicha PA.

4.2 Dependencia Causa-Efecto

La Interdependencia Simple no considera dirección en las relaciones entre PAs o entre nodos, así cualquier elemento podría verse afectado si alguno de los elementos relacionados se modifica. El segundo tipo de Análisis de Impacto que soporta nuestra propuesta lo hemos llamado Dependencia Causa-Efecto pues conlleva una direccionalidad desde uno de los elementos relacionados.

Al etiquetar con atributos las relaciones (facilidad que ofrece Neo4j) podemos diferenciar desde qué apartados de la PA se hace referencia a una Entidad. En la Fig. 9 se ilustra la Dependencia Causa-Efecto. En este ejemplo tenemos dos pruebas (PA1 y PA2) con sus tres apartados. En el apartado Resultado de la PA1 se referencia a la entidad (E) y en el apartado condiciones de la PA2 se referencia también esa entidad. Así, al ejecutarse el comportamiento de la PA2 debe darse una condición que está relacionada con el resultado de la ejecución de la PA1. Por lo tanto, hay una mayor probabilidad de que si se modifica la PA1 esto va a tener un efecto en la PA2 (PA1 puede afectar PA2). Además, como dichas PAs se encuentran en nodos diferentes podemos deducir que el cambio en el nodo N1 puede afectar al nodo N2. En la Fig. 10 se ilustra esta dependencia con un ejemplo.

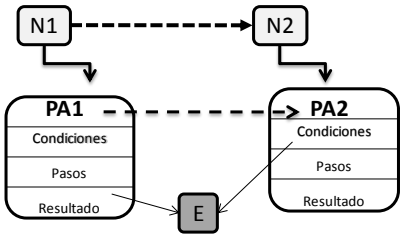

Fig. 9. Dependencia Causa-Efecto

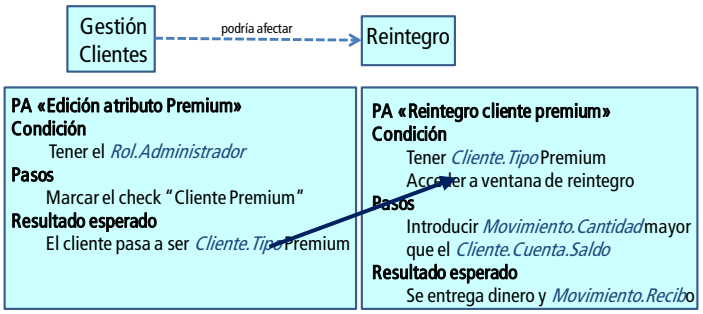

Fig. 10. Ejemplo con Dependencia Causa-Efecto

Un caso típico de Dependencia Causa-Efecto es el que se establece entre nodos (y sus PAs) en los cuales se realiza la configuración de un producto, con respecto de otros nodos (y sus PAs) sobre los cuales actúan los parámetros de configuración.

Las relaciones que obtenemos por Dependencia Causa-Efecto son un subconjunto de las que se obtienen mediante Interdependencia Simple, con lo cual en la explotación de la información las primeras son remarcadas, puesto que tienen mayor probabilidad de evidenciar la necesidad de propagación de cambios a otros requisitos.

5 Trabajos Relacionados

La mayoría de trabajos que abordan el análisis de impacto con automatización para el mantenimiento de los enlaces de trazabilidad se centran en un análisis a nivel de diseño y código. A continuación se describen algunos de estos trabajos.

En [4] se presenta el prototipo *What-If Impact Analysis tool* que se ha implementado para analizar a priori el análisis de impacto a través de las dependencias del código, aunque se ha desarrollado el prototipo no se ha integrado en ningún entorno de desarrollo y tampoco ha sido testeado en ningún proyecto real. En [8] se presenta la técnica *PathImpact,* la cual mientras el sistema se está ejecutando, se construye una representación del comportamiento del sistema (utilizando un grafo acíclico dirigido propuesto por Larus en [14]) y la utiliza para estimar el impacto. En dicho trabajo los autores comparan su técnica con otras técnicas que predicen el impacto del cambio. Lee y Offut en [1] analizan las características del software orientado a objetos para definir cómo la composición, la herencia y el polimorfismo pueden influenciar en el Análisis de Impacto. Para ello categorizan los diferentes tipos de cambios que podrían ser aplicados en un sistema orientado a objetos y asignan a cada tipo un atributo que indica la influencia del cambio en otros objetos del sistema.

Las aproximaciones presentadas en [1,4,8,14] tienen como objetivo evaluar a priori el impacto de un cambio en el software, es decir, hacen un análisis predictivo para trabajar con un plan antes de tratar con las consecuencias.

Por otra parte, gran parte de los trabajos asociados a Análisis de Impacto intentan abarcar todos los posibles artefactos del proceso, a continuación comentamos dos que nos parece interesante mencionar.

Göknil en [2] presenta una técnica basada en la formalización de relaciones de requisitos en el contexto de Model Driven Depevelopment. Allí se presenta un metamodelo que incluye los conceptos comunes extraídos de las aproximaciones más importantes de Ingeniería de Requisitos. Los cuatro tipos de relaciones que se han considerado en el metamodelo (Requires, Refines, Conflicts y Contains) son utilizadas para hacer la traza y obtener los requisitos que se tendrían que modificar. En dicho trabajo se presentan estas cuatro relaciones formalizadas junto con unas restricciones que hay que tener en cuenta y basándose en esta formalización se definen las reglas de impacto para los requisitos. En aproximaciones como ésta, en las cuales se propone trabajar con diferentes tipos de relaciones de trazabilidad entre los artefactos, aparece como problema añadido el diferenciar semánticamente dichos tipos de relaciones. Un caso extremo ocurre en [18] donde se proponen alrededor de 50 diferentes tipos de relaciones entre artefactos. En nuestro contexto y centrados en Análisis de Impacto hasta ahora sólo hemos consideramos el tipo de relación *Afecta*, es decir, una PA puede afectar otra PA, o un requisito (nodo) puede afectar a otro.

En [3] se presenta un enfoque para Análisis de Impacto considerando diferentes niveles de profundidad del alcance del cambio. De manera similar en [19] se

introduce la medida Distancia para ser capaz de evaluar el impacto según su probabilidad de ocurrencia y basándose en la distancia entre los elementos modificados y los elementos impactados.

En propuestas como las descritas en [3, 2] las relaciones de trazabilidad entre los artefactos se definen manualmente, lo cual es inabordable en proyectos medianos o grandes, a menos que dicha trazabilidad sea fácilmente configurable para adaptarla a las necesidades del proyecto, restringiendo los tipos de artefactos y de relaciones que se identificarán y mantendrán actualizados [15,16].

Todos los enfoques antes descritos para Análisis de Impacto son teóricos, no están soportadas por una herramienta, o tienen un prototipo, pero hasta donde llega nuestro conocimiento, en ningún caso han sido utilizados industrialmente.

Las herramientas comerciales específicas para gestión de requisitos ofrecen funcionalidad muy similar respecto de Análisis de Impacto, las cuáles normalmente se encuentran descritas en la literatura y están basadas en Matrices de Trazabilidad [10,11]. Por ejemplo, en RequisitePro (ibm.com/software/awdtools/reqpro/), DOORS (ibm.com/software/awdtools/doors/), IRQA (visuresolutions.com/irqa-requirements-tool) y Jama (jamasoftware.com) se establecen y mantienen manualmente Matrices de Trazabilidad. Las celdas se marcan *Suspect* (sospechosas) cuando el elemento de la parte *from* de la relación es modificado.

Finalmente, desde el punto de vista de la especificación textual de requisitos y de los inconvenientes que conlleva utilizar lenguaje natural, existen muchos trabajos en los cuales, utilizando restricciones de sintaxis o incluso lenguajes formales, intentan evitar o reducir dichos inconvenientes. La motivación de estas propuestas normalmente es generar otros artefactos a partir de especificaciones de requisitos (usando técnicas de transformación de modelos) o realizar análisis de consistencia de la especificación de requisitos. En nuestro caso, el interés está centrado en el Análisis de Impacto, y sólo en el ámbito de los requisitos y pruebas de aceptación, sin pretender limitar o restringir el uso del lenguaje natural en la especificación textual de las PAs.

6 Conclusiones

El Análisis de Impacto es un tema de gran interés y una de las principales motivaciones para definir y mantener trazabilidad entre artefactos software. Esta importancia es reconocida por toda la literatura de Ingeniería de Requisitos, sin embargo, el estado del arte en la teoría y en la práctica continúa siendo el mismo de hace dos décadas. Existen dos factores que explican esta situación; por un lado resulta evidente que disponer de relaciones de trazabilidad entre artefactos en un producto software requiere mucho esfuerzo, tanto en su definición como en su mantenimiento. El esfuerzo invertido no es fácil de rentabilizar cuando la explotación de dicha información es rudimentaria y poco detallada, tal como sucede al utilizar Matrices de Trazabilidad como soporte para dicha información. Por otra parte, los enfoques clásicos para Análisis de Impacto son demasiado ambiciosos en cuanto a intentar cubrir todo tipo de artefactos, desde requisitos hasta el código, lo cual incrementa el problema de esfuerzo antes mencionado. Otros enfoques para Análisis de Impacto ofrecen automatización, salvando así el inconveniente del esfuerzo en definición y

mantenimiento de la trazabilidad, pero están centrados sólo en artefactos del ámbito del código. Además, a nivel de código esto no representa una significativa contribución teniendo en cuenta todas las facilidades que suelen incluir los IDEs actuales para determinar relaciones entre piezas de código.

Nuestro enfoque salva los impedimentos asociados al esfuerzo invertido mediante automatización en la definición y mantenimiento de la información necesaria para Análisis de Impacto. Además, nuestra propuesta está principalmente dirigida a facilitar el trabajo de definición de requisitos, ámbito en el cual el Análisis de Impacto resulta muy apreciado. Aunque en nuestro caso el alcance de la trazabilidad sólo son los requisitos y las pruebas de aceptación resulta muy pragmático e integrado en el marco TDRE de TUNE-UP.

TUNE-UP, con el enfoque TDRE incluido, está disponible para ser descargado desde www.tuneupprocess.com. La funcionalidad asociada a Análisis de Impacto ha empezado actualmente a ser aplicada y mejorada en contexto de mantenimiento de un producto de gran envergadura (un ERP del sector socio-sanitario) y en el desarrollo de otros productos más pequeños. En breve la funcionalidad para Análisis de Impacto estará incorporada en la versión descargable de TUNE-UP.

Si bien de momento no tenemos resultados definitivos respecto de la validación del enfoque, tenemos garantizado que representará una mejora significativa pues los futuros usuarios de la herramienta han participado activamente en su concepción.

Aunque la propuesta se acaba de implantar ya hemos detectado algunos aspectos importantes para mejorar y que constituyen nuestros próximos desafíos. Ellos son los siguientes:

- Se generan muchísimas relaciones entre PAs (y por consiguiente entre nodos). Aunque el estudio de las Dependencias Causa-Efecto permite centrarse en un subconjunto más relevante, aún resulta laborioso estudiar una por una las PAs posiblemente afectadas. Habría que incorporar otros mecanismos que permitan determinar un subconjunto menor en el cual concentrar el estudio. Por ejemplo, se podrían utilizar pesos para las PAs y/o para las relaciones de forma que aquellas con mayor peso se remarquen. De forma similar podríamos utilizar la información histórica asociada a artefactos efectivamente afectados y a falsos positivos para destacar el conjunto de artefactos candidatos para ser revisados.
- Asociado a lo anterior, es muy importante ofrecer una forma de presentación más visual de los elementos y sus relaciones. Debería ser lo más cercano a un grafo, tal como está estructurada la información en la BDOG.
- Sería conveniente tener un mecanismo para realizar análisis predictivo (*what if*), sin tener que modificar el estado actual de la especificación, puesto que a veces debido a los resultados del Análisis de Impacto se decide explorar otras alternativas de cambio o simplemente descartar la propuesta de cambio debido al impacto que puede tener.

Referencias

1. Lee, M., Offutt, J., and Alexander, R. Algorithmic Analysis of the Impacts of Changes to Object-Oriented Software. 34th International Conference on Technology of Object-Oriented Languages and Systems, 2000, pp. 61-70.
2. Göknil, A. and Kurtev, I. and van den Berg, K.G. Change Impact Analysis based on Formalization of Trace Relations for Requirements. European Conference on Model Driven Architecture (ECMDA) Traceability Workshop 2008, pp. 59-75
3. Gupta, C. Singh, Y. Chauhan, D. Dependency based Process Model for Impact Analysis: A Requirement Engineering Perspective. International Journal of Computer Applications 2010 , vol. 6, issue 6, pp. 28-33
4. Ajila, S. Software Maintenance: An Approach to Impact Analysis of Objects Change. Software: Practice and Experience. Volume 25, Issue 10, pp. 1155–1181, October 1995
5. Turver, R. Munro, M. An early impact analysis technique for software maintenance. Journal of Software Maintenance: Research and Practice. Volume 6, Issue 1, pp. 35–52, 1994
6. Jun Han. Supporting Impact Analysis and change propagation in SW Engineering Environments. 8th International Workshop on Software Technology and Engineering Practice incorporating Computer Aided Software Engineering 1997, pp. 172-182.
7. James S. O'Neal, Doris L. Carver. Analyzing the Impact of Changing Requirements. 17th IEEE International Conference on Software Maintenance (ICSM) 2001, pp. 190-195
8. Law, J. Rothermel, G. Whole program Path-Based dynamic impact analysis. 25th International Conference on Software Engineering (ICSE) 2003, pp. 308-318
9. Marante, M. Company, M. Letelier, P. Suarez, F. Gestión de requisitos basada en pruebas de aceptación: Test-Driven en su máxima expresión. XV Jornadas de Ingeniería del Software y Bases de Datos 2010.
10. Sommerville, I. Sawyer, P. Requirements Engineering: A Good Practice Guide. Chichester, England: John Wiley & Sons. 1997
11. Wiegers, K. Software Requirements. Microsoft Press 2003.
12. Marante, M. Letelier, P. Suarez, F. TUNE-UP: Un enfoque pragmático para la planificación y seguimiento de proyectos de desarrollo y mantenimiento de software. I Congreso Iberoamericano SOCOTE- Soporte al Conocimiento con la Tecnología. 2009.
13. Bohner, S and Arnold, R. *Software Change Impact Analysis* IEEE Computer Society Press, Los Alamitos, CA, USA 1996.
14. J. Larus. Whole Program Paths. ACM SIGPLAN 1999 conference on Programming language design and implementation, pp. 259-269, Atlanta, GA, May 1999. ACM.
15. 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. 2002 , pp. 32-41.
16. Letelier, P. Navarro, E. Anaya, V. Customizing Traceability in a Software Development Process. Information System Development: Advances in Theory, Practice and Education. Springer. 2005 pp. 137-148.
17. Ramesh, B. Factors Influencing Requirements Traceability Practice. Communications of the ACM. 1998, pp. 37-44.
18. Ramesh, B., Jarke, M. Toward Reference Models for Requirements Traceability. IEEE Transactions on Software Engineering, Vol. 27, No. 1, January 2001.
19. Briand, L.C. Labiche, Y. O'Sullivan, L. Impact analysis and change management of UML models. Proceedings of the International Conference on Software Maintenance (ICSM'03). 2003, pp. 256-266
20. Hayes, J.H. Dekhtyar, A. Sundaram, S. Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods. IEE Transactions on Software Engineering, Vol. 32,No. 1, January 2006.