Improving Use Case Specifications by Means of Refactoring

C. Marcos, A. Rago and J. A. D. Pace

Abstract— **This work presents a semi-automatic tool for use case refactoring called RE-USE. This tool discovers existing quality problems in use cases and suggests a prioritized set of candidate refactorings to functional analysts. The analyst then reviews the recommendation list and selects the most important refactoring. The tool applies the chosen refactoring and returns an improved specification. The tool effectiveness in detecting existing quality problems and recommending proper refactorings was assessed using a set of case studies related to real-world systems obtaining encouraging results.**

Keywords— **requirements engineering, early aspects, use case refactoring, use case specification, duplicate functionality.**

I. INTRODUCCIÓN

PARA que un proyecto de software sea exitoso es necesario comprender de una manera clara los problemas del cliente que deben ser resueltos. Una buena especificación de requerimientos es un aspecto primordial para lograr el éxito del proyecto. La ingeniería de requerimientos continúa hoy en día siendo uno de los temas principales de investigación relativos al desarrollo de software. A pesar de la experiencia adquirida por la industria en las últimas décadas en dicha materia, es común que las organizaciones tengan dificultades para elicitar, documentar y administrar los requerimientos de sus clientes. Muchos proyectos de software fracasan al momento de entregar la funcionalidad dentro del tiempo y presupuesto acordado. Las razones principales que provocan esta falla son insuficiente información provista por el usuario, documentación incompleta de los requerimientos y gran volatilidad de las necesidades del cliente [1].

En este trabajo se presenta un enfoque semi-automático para la mejora de especificaciones de casos de uso, facilitando la comprensión y especificación de requerimientos. Este trabajo realiza varias contribuciones al estado del arte. En primer lugar, define un enfoque iterativo (semi-)automático para la refactorización de casos de uso. En segundo lugar, diseña e implementa un componente para la detección de secciones de funcionalidad duplicada entre casos de uso. Por último, plantea un sistema de puntuación que permite priorizar las refactorizaciones candidatas halladas, asistiendo al analista funcional en la toma de decisiones.

Con el objetivo de determinar la efectividad de la herramienta desarrollada, este trabajo se centra en describir la evaluación del prototipo utilizando un conjunto de casos de estudio. Los resultados preliminares obtenidos en las pruebas son alentadores. El prototipo demuestra que posee la capacidad de detectar correctamente la gran mayoría de los problemas de calidad presentes en las especificaciones sin introducir una cantidad excesiva de falsos positivos. La cantidad de falsos positivos, es decir la cantidad de casos reportados erróneamente como problemas, varía de acuerdo a las particularidades de cada caso de estudio y es uno de los puntos a mejorar en los futuros desarrollos. Respecto a la sugerencia de refactorizaciones que permitan solucionar los problemas detectados, la herramienta tuvo un excelente desempeño identificando correctamente casi la totalidad de las refactorizaciones requeridas.

Este trabajo está organizado de la siguiente manera: en la Sección II se describen los trabajos relacionados a la mejora de la especificación de requerimientos. La Sección III presenta brevemente el enfoque RE-USE. La Sección IV describe detalladamente los experimentos realizados para la validación de RE-USE. Finalmente, la Sección V presenta las conclusiones.

II. TRABAJOS RELACIONADOS

En la actualidad, es posible encontrar una gran diversidad de trabajos que proponen estrategias para mejorar la calidad de los documentos de especificación de requerimientos y de casos de uso, cada uno enfocando la mejora en diversos aspectos de la documentación.

Existen una serie de trabajos recientes que basan el proceso de mejora en la definición de catálogos de refactorización aplicados a la etapa de requerimientos. Algunos de estos trabajos formalizan el proceso de análisis de la documentación mediante la técnica Goal Question Metrics (GQM) [2]. GQM prescribe la construcción de un plan de evaluación y de mejoras, definiendo un modelo de métricas que permite determinar la calidad de los artefactos de software. Ramos y otros han instanciado el framework GQM en el contexto de casos de uso [3]. Su enfoque, llamado AIRDoc, permite mejorar la calidad de los casos de uso mediante evaluaciones y refactorizaciones progresivas. Una evaluación permite determinar dónde yace un problema, mientras que una refactorización ayuda a resolver el problema. Para cada problema definido, el enfoque describe mecanismos que permiten solucionarlos. Sin embargo, la identificación de los defectos debe ser realizada por un analista de forma manual.

Otros trabajos brindan indicios de problemas que resultan útiles para guiar y enfocar el proceso de mejora de documentos de requerimientos. Por ejemplo el trabajo de Ciemniewska y otros [4] provee técnicas para identificar defectos en especificaciones de casos de uso. Los defectos se organizan en

C. Marcos, Fac. de Cs. Exactas de la UNCPBA e investigadora de CIC, cmarcos@exa.unicen.edu.ar

A. Rago, Fac. de Cs. Exactas de la UNCPBA y becario de CONICET, arago@exa.unicen.edu.ar

J. A. D. Pace es investigador del ISISTAN, docente de la Fac. de Cs. Exactas de la UNCPBA e investigador de CONICET, adiaz@exa.unicen.edu.ar

tres niveles: a nivel de especificaciones, de casos de uso, y de pasos. El nivel de especificaciones considera la duplicación de comportamiento. El nivel de casos de uso considera casos de uso muy cortos o largos, o extensiones complejas, entre otros. El nivel de pasos considera estructuras sintácticas complejas, u omisión de actores, entre otros. Aunque este trabajo no propone refactorizaciones para solucionar los defectos, es relevante porque provee heurísticas sencillas para identificar los defectos.

Otros trabajos se enfocan más en la mejora propiamente dicha. Rui y otros estudiaron la aplicación de técnicas de refactorización en casos de uso [5]. Las refactorizaciones son organizadas usando un meta-modelo de casos de uso. Sin embargo, la tarea de determinar dónde y cómo una refactorización debe ser aplicada es incierto, y se deja esta responsabilidad a los analistas. En la misma línea, Yu y otros sugieren que para asegurar la construcción de casos de uso con una granularidad y nivel de abstracción adecuada, es necesario tener en cuenta la funcionalidad común, eliminando el comportamiento redundante y motivando el reuso de funcionalidad [6].

III. RE-USE: UNA HERRAMIENTA DE REFACTORIZACIÓN DE CASOS DE USO

El enfoque propuesto para la refactorización de casos de uso es iterativo, y se basa en la aplicación de sucesivas evaluaciones y refactorizaciones que permiten una mejora incremental de la especificación de requerimientos (Fig. 1). Este enfoque toma como entrada la *especificación de casos de uso* del sistema a analizar. Primero, se analizan los casos de uso y se identifican los posibles problemas. Luego, en cada iteración, el analista selecciona el problema considerado más crítico y aplica la refactorización apropiada, obteniendo una especificación de casos de uso parcialmente mejorada a ser utilizada como entrada de la siguiente iteración. El enfoque propuesto se encuentra materializado como un conjunto de plugins de Eclipse, los cuales permiten a los analistas interactuar durante todo el proceso iterativo del enfoque.

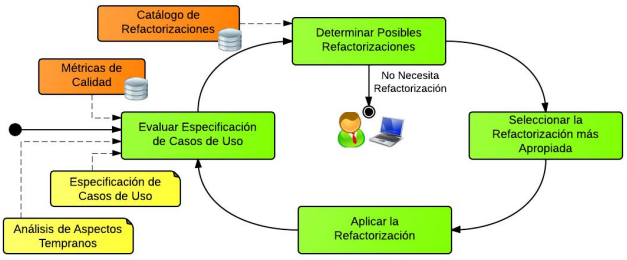

Figura 1. Enfoque para la Refactorización de Casos de Uso.

Para la primera actividad denominada *Evaluar Especificaciones de Casos de Uso*, que involucra evaluar los defectos de la especificación, se desarrolló un conjunto de métricas de calidad que permiten recuperar indicios de problemas de calidad. Para dichos problemas las métricas reportan principalmente secciones de funcionalidad duplicada, requerimientos funcionales no modularizados, casos de uso excesivamente complejos, relaciones entre casos de uso mal utilizadas y actores o casos de uso sin sentido dentro de la especificación. Los procesos para la recolección de estos datos están implementados en distintos componentes de la herramienta. El componente de detección de secciones duplicadas permite descubrir funcionalidad que presenta gran similitud en los documentos. Esto se realiza mediante la combinación de técnicas de Procesamiento de Lenguaje Natural (NLP) [7] y Aprendizaje de Maquina (ML) [8] [9] con técnicas de Alineamiento de Secuencias (usadas generalmente en el área de investigación genética) [10] [11] [12].

La segunda actividad denominada *Determinar Posibles Refactorizaciones* identifica las refactorizaciones que solucionan los problemas de calidad hallados, para lo cual se desarrolló un catálogo de refactorizaciones de casos de uso. Este catálogo detalla cada refactorización disponible, describiendo el contexto de aplicación, el problema a solucionar y los pasos requeridos para solucionarlo. Esta actividad toma cada problema de calidad relevado anteriormente y busca en el catálogo una refactorización que lo solucione. Cada refactorización que provee una solución es candidata a ser aplicada en esta iteración.

La tercer actividad denominada *Seleccionar la Refactorización más Apropiada* tiene como objetivo identificar la refactorización más importante. Para ello, se desarrolló un sistema de puntuación que permite priorizar las refactorizaciones candidatas en base a distintos aspectos. En este ordenamiento se tienen en cuenta distintos aspectos de las refactorizaciones candidatas: la importancia del defecto, la confianza de la detección y la severidad del problema encontrado. Mediante la combinación de los valores mencionados, se obtiene una puntuación para cada refactorización candidata. La salida de la herramienta presenta al desarrollador una lista con las refactorizaciones candidatas ordenadas de acuerdo a la puntuación calculada. Por último, si bien la herramienta sugiere la aplicación de la refactorización con puntuación más alta, el analista funcional puede evaluar la lista y seleccionar manualmente otra refactorización que considere más importante.

La cuarta actividad denominada *Aplicar la refactorización* implementa los mecanismos de aplicación descriptos en el catálogo para solucionar el problema encontrado. En este caso, se desarrolla la secuencia de pasos correspondiente a la refactorización elegida. En caso que la aplicación requiera de alguna información adicional, por ejemplo para ingresar el nombre de un nuevo caso de uso, dicha información es solicitada al analista funcional.

Una vez aplicada la refactorización, la iteración actual finaliza presentando como resultado una especificación de casos de uso parcialmente mejorada. La próxima iteración comienza con la evaluación de los problemas de calidad presentes en esta versión mejorada, y continua sucesivamente con los pasos anteriormente descriptos. El procedimiento iterativo finaliza cuando no se detectan problemas de calidad al evaluar la especificación, o el analista así lo decide. Una descripción más detallada del enfoque es descripta en [13].

IV. EVALUACIÓN DE RE-USE

De manera tal de evaluar el enfoque se analizaron especificaciones de casos de uso de sistemas reales. Los analizados fueron: dLibra CRM [4], que describe un software para la gestión de los clientes que adquirieron el sistema para creación de librerías digitales dLibra; Mobile News [4], un sistema de noticias para dispositivos móviles; WebJSARA [4], el cual provee funcionalidades para la gestión de cuentas de usuario, administración, upload y download de archivos de datos, búsqueda de información, ejecución de simulaciones y gestión de publicaciones; HWS [14], una especificación de casos de uso que describe un sistema web intermediario entre los ciudadanos y el gobierno municipal para atender cuestiones relacionadas con la salud; y CRS [15], que describe un sistema distribuido para administrar cursos e inscripciones en una universidad. Los primeros tres sistemas están compuestos por un conjunto de especificaciones de casos de uso desarrolladas como parte del proyecto Software Development Studio (SDS). La documentación de dLibra CRM, Mobile News y WebJSARA consiste de 15 (aprox. 13 páginas), 15 (aprox. 12 páginas) y 29 (aprox. 22 páginas) casos de uso, respectivamente. El cuarto sistema es el Course Registration System (CRS), y consiste de 8 casos de uso (aprox. 20 páginas). El quinto y último sistema es el Health Watcher System (HWS), el cual contiene 9 casos de uso (aprox. 19 páginas).

Para determinar el desempeño del enfoque se utilizan métricas del área de Recuperación de Información, como *precision* y *recall*. La *precisión* mide la proporción de material recuperado realmente relevante del total de lo recuperado, mientras que *recall* es la proporción de material recuperado con respecto al total de lo identificado en la solución de referencia. Las fórmulas correspondientes son las siguientes:

$$precisión = TP / (TP + FP)$$

$$recall = TP / (TP + FN)$$

Donde TP (true positive) representa aquellos casos donde la respuesta del enfoque es el resultado esperado; FP (false positive) representa aquella respuesta del enfoque que no es un resultado esperado; FN (false negative) representa aquellos casos donde el enfoque no encuentra el resultado esperado.

Debido a la naturaleza comparativa de la experimentación con casos de estudio, y a la inexistencia de análisis similares en la literatura, se pidió a analistas experimentados que realizaran un análisis de los casos de uso para tener una solución de referencia. Ellos estuvieron a cargo de inspeccionar los documentos textuales de forma manual, identificando las porciones defectuosas de las especificaciones y determinando alternativas que permitirían mejorar dichos casos de uso. El resultado de este ejercicio permitió elaborar una solución de referencia, la cual representa el razonamiento de los analistas y permite hacer comparaciones con los resultados producidos por la herramienta.

TABLA I. PROBLEMAS DETECTADOS POR LOS ANALISTAS.

Problemas de calidad	dLibra CRM	Web JSARA	Mobile News	CRS	HWS	Total
Funcionalidad duplicada	6	15	4	0	11	**36**
Requerimientos funcionales no modularizados	3	1	0	0	0	**4**
Casos de uso extensos	0	2	2	3	2	**9**
Casos de uso sin sentido	0	0	1	0	0	**1**

TABLA II. SOLUCIONES ESTABLECIDAS POR LOS ANALISTAS.

Refactoring candidato	dLibra CRM	Web JSARA	Mobile News	CRS	HWS	Total
Generar Relación de Inclusión	4	10	4	0	0	**18**
Generar Relación de Extensión	2	0	0	0	11	**13**
Generar Relación de Generalización	0	5	0	0	0	**5**
Unificar Casos de Uso	3	1	0	0	0	**4**
Extraer Caso de Uso	0	2	2	3	2	**9**
Eliminar Caso de Uso	0	0	1	0	0	**1**

La inspección manual de los casos de estudio realizada por los analistas funcionales permite establecer una base de comparación útil para los experimentos, permitiendo establecer fehacientemente si el enfoque es capaz de descubrir los defectos ocultos en los casos de uso, como así también su eficacia cuando se utiliza en especificaciones provenientes de sistemas reales. Como resultado de dichas inspecciones, los analistas identificaron un número considerable de defectos en las especificaciones. La Tabla I presenta el conjunto de los problemas detectados para cada caso de estudio. Además de la detección, los analistas sugirieron un conjunto de refactorizaciones candidatas para solucionar los defectos encontrados (Tabla II). La Fig. 2, describe la totalidad de los defectos hallados, su distribución, y el conjunto de refactorizaciones candidatas sugeridas por los analistas para las especificaciones evaluadas.

Al observar estos gráficos, se destaca que más del 70% de los problemas de calidad detectados corresponden a secciones de funcionalidad duplicada. Las refactorizaciones sugeridas para solucionar este tipo de defectos son generar relaciones de inclusión, extensión y de generalización. El 30% restante está formado principalmente por problemas de calidad relacionados con la extensión de los casos de uso. Demasiados detalles sobre la interfaz de usuario, bajo nivel de abstracción de eventos, y descripciones de información no relacionada al comportamiento resultan en casos de uso muy extensos y difíciles de leer. Las buenas prácticas recomiendan casos de uso con más de tres eventos y menos de diez, lo cual es en general suficiente para describir un caso de uso correctamente.

Dado estos resultados esperados se procedió a realizar la evaluación de los casos de estudio. Dicha evaluación se realizó en tres etapas: análisis de la identificación de problemas de

calidad, análisis del proceso de selección de refactorizaciones candidatas y análisis del ordenamiento sugerido acorde a la importancia de estas refactorizaciones. Los resultados obtenidos son comparados con los arrojados por el proceso manual, y se analizan en detalle de forma tal de poder arribar a conclusiones generales de la efectividad de la herramienta.

A. Identificar problemas de calidad

La primera etapa definida es la de identificación de problemas de calidad. En la misma, se verificó la capacidad de la herramienta para detectar los problemas presentes en las especificaciones de casos de uso. Para llevar a cabo esta verificación, se compararon los resultados manuales obtenidos por los analistas con los arrojados por la herramienta, considerando correcta una identificación automática cuando coincide con la realizada de forma manual. Los resultados obtenidos en esta comparación fueron utilizados para calcular los valores de los operadores definidos anteriormente en la matriz de difusión binaria, como se muestra en la Tabla III.

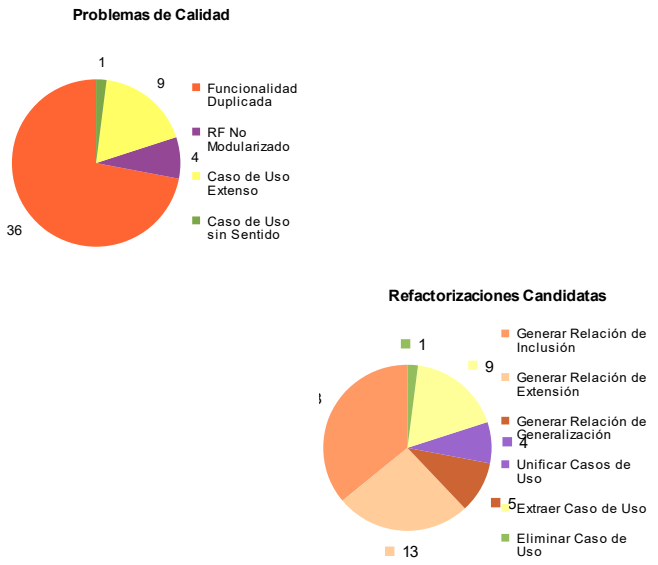

Figura 2. Distribución de problemas y refactorizaciones por el analista.

Al analizar los resultados, se observa que la herramienta recuperó la mayoría de los problemas de calidad existentes (90% *recall*), sin cometer un número significativo de errores en el proceso (66% *precisión*). En 3 de los 5 casos de estudio, la herramienta reconoció la totalidad de los defectos presentes en las especificaciones, logrando un *recall* del 100%. Estos resultados de *recall* dan la pauta de que la herramienta es capaz de detectar una gran parte de los defectos en las especificaciones evaluadas. En los casos en que el valor de *recall* es menor, los causantes de este resultado fueron las particularidades del caso de estudio evaluado. Por ejemplo, en el caso de estudio WebJSARA, en donde el *recall* cayó a un 66%, se observó que uno de los problemas detectados por los expertos que no fue recuperado por la herramienta estaba relacionado con la modularización de los requerimientos funcionales. El caso particular fue la modularización que involucraba a los casos de uso "Login" y "Logoff". Ambos casos de uso son extremadamente cortos (2 eventos cada uno) y el análisis experto sugiere que deberían unificarse dado que el contenido está relacionado a la misma funcionalidad y no se justifica el mantenimiento de ambos artefactos. Sin embargo, la herramienta falla en la detección ya que en descripciones tan cortas (sólo 12 términos relevantes) el análisis de contenido no puede determinar si refieren a una funcionalidad similar.

TABLA III. PROBLEMAS DE CALIDAD IDENTIFICADOS.

	dLibra CRM	Web JSARA	Mobile News	CRS	HWS	**Total**
TP	9	12	6	3	13	**43**
FP	6	10	1	1	7	**25**
FN	0	6	1	0	0	**7**
precisión	**0.60**	**0.54**	**0.85**	**0.75**	**0.65**	**0.66**
recall	**1.00**	**0.66**	**0.85**	**1.00**	**1.00**	**0.90**

Con respecto a la *precisión*, la herramienta obtuvo sin grandes dificultades y de manera automática valores superiores al 75% aunque, en algunos casos en particular, el resultado fue notablemente menor, obteniendo resultados del 60%. Esta disminución en la *precisión* se atribuye a dificultades en la herramienta para analizar especificaciones que presentan la misma estructura sintáctica, y utilizan un vocabulario similar para describir las secuencias de acciones de los casos de uso. Por ejemplo, en el caso de estudio dLibra CRM, en el cual se obtuvieron los resultados más bajos de *precisión*, se apreció que la herramienta recuperó en total un 40% de falsos positivos. Este problema ocurrió en menor medida para el resto de los casos de estudio.

B. Seleccionar factorizaciones candidatas

Se evaluó la selección de refactorizaciones candidatas provistas por la herramienta. Para realizar esta evaluación se buscó obtener, del total de refactorizaciones candidatas sugeridas por la herramienta, el porcentaje de las mismas que es considerado correcto, representado por el valor de *precisión*. Utilizando un método similar al de la primera etapa, se compararon las refactorizaciones sugeridas automáticamente con las identificadas manualmente por los analistas para calcular los valores de los operadores. Al igual que en la etapa anterior, se recurre a la matriz de difusión binaria, representada en la Tabla IV, para calcular el valor de *precisión* del enfoque en la selección de refactorizaciones, utilizando los resultados de la comparación entre el proceso manual y el automático. Ya que esta etapa parte del conjunto de problemas identificados previamente, el valor de *recall* no aporta información de gravedad, por lo que no es incluido en la evaluación. De esta forma, la suma de los valores de TP y FP para cada columna se corresponde con el total de TP detectados en la etapa anterior para el mismo caso de estudio.

TABLA IV. REFACTORIZACIONES CANDIDATAS.

	dLibra WCM	Web JSARA	Mobile News	CRS	HWS	Total
TP	9	9	6	3	13	**40**
FP	0	3	0	0	0	**3**
FN	0	3	0	0	0	**3**
precisión	**1.00**	**0.75**	**1.00**	**1.00**	**1.00**	**0.93**

Las refactorizaciones sugeridas por la herramienta incluyeron: "Generar Relación de Inclusión", "Generar Relación de Extensión", "Generar Relación de Generalización", "Unificar Casos de Uso", "Extraer Caso de Uso", y "Eliminar Caso de Uso". Al analizar estos resultados, se observa que en, prácticamente, la totalidad de los casos la herramienta sugiere la refactorización recomendada por los expertos. Igualmente, la herramienta cometió algunos errores en la selección de refactorizaciones candidatas. En el caso de estudio WebJSARA, las tres refactorizaciones de tipo "Generar Relación de Generalización" presentes en este caso de estudio fueron erróneamente identificadas por la herramienta como tres refactorizaciones de tipo "Generar Relación de Inclusión".

C. Priorización de refactorizaciones candidatas

Para realizar esta tarea, se tomó como ejemplo el ordenamiento de resultados en motores de búsquedas web, donde los resultados más relevantes son incluidos en la primera página. En el análisis de este proceso de ordenamiento en particular se calcula, para cada caso de estudio, el porcentaje de sugerencias correctas tomando distintos subconjuntos de la lista de recomendación: los primeros cinco resultados, los primeros diez, y la lista completa; considerando correctos (verdaderos positivos) aquellos resultados donde tanto la identificación del problema como la sugerencia de refactorización provista por la herramienta coincidieron con lo reportado por el análisis manual. A su vez, se consideraron como error (falsos positivos) aquellos casos donde la identificación del problema o la sugerencia provista fueron incorrectos.

Para determinar la posición de una refactorización candidata en la lista, se utiliza una función de scoring. La función utilizada para este ordenamiento plantea tres variables en una refactorización candidata: prioridad, propia de cada tipo de refactorización; confiabilidad de la detección, calculada comparando términos clave entre casos de uso; y un valor adicional que representa la severidad del problema, obtenido observando características propias de cada refactorización en particular.

La Fig. 3 muestra la lista ordenada de problemas y refactorizaciones para el caso de estudio WebJSARA. Al evaluar dicha lista comparando los resultados con los provistos por el análisis manual, es posible calcular la *precisión* de los resultados obtenidos. Considerando los primeros cinco puestos de la lista, sólo uno de estos es incorrecto (ID = 2), por lo tanto la *precisión* es la mayor, alcanzando el 80%. Si se consideran los primeros diez el recuento da tres resultados incorrectos (ID = 2, 6 y 9), obteniendo una *precisión* del 70%. Al incrementar el tamaño del subconjunto considerado, se encuentran cada vez más recomendaciones incorrectas, disminuyendo la *precisión* hasta llegar al 40%. De estos valores, es posible deducir que la herramienta hace un buen trabajo al reconocer los problemas más importantes, pero su efectividad decae a medida que se amplía el subconjunto de resultados analizados. Si bien los resultados satisfacen las expectativas, el algoritmo de ordenamiento puede ser analizado y mejorado en trabajos futuros.

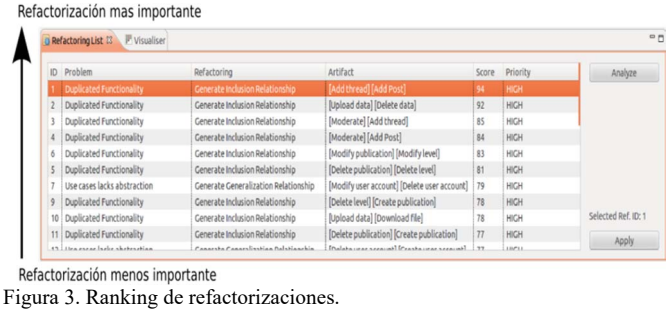

Figura 3. Ranking de refactorizaciones.

El mismo proceso de análisis fue aplicado a todos los casos de estudio, obteniendo resultados similares. Debido a esta similitud, no se muestran en esta evaluación los resultados individuales. Promediando estos resultados, se calculó el valor de *precisión* general de la herramienta en el ordenamiento de las refactorizaciones candidatas. El promedio de los valores de *precisión* obtenidos para cada subconjunto se muestra a continuación en la Fig. 4.

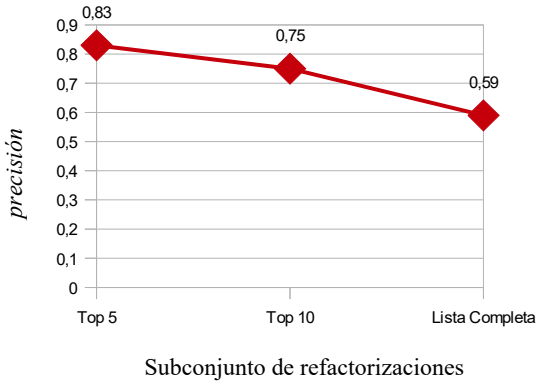

Figura 4. Ranking de refactorizaciones de todos los casos de estudio.

Al analizar el promedio de los ordenamientos en los cinco casos de estudio, se apreció que la *precisión* es mayor cuando se consideran los primeros 5 resultados de la lista de recomendación. En general, aproximadamente el 80% de estos resultados fueron detectados correctamente por la herramienta, la cual reportó el mismo problema de calidad que el experto y coincidió en la sugerencia de refactorización. Considerando los primeros 10 resultados la *precisión* decae a 0,75 y disminuye a 0,59 al considerar la lista completa. Teniendo en cuenta los resultados obtenidos con la función actual, se concluye que el ordenamiento provisto por la herramienta aporta el beneficio de presentar una mayor proporción de resultados correctos al inicio de la lista de recomendación, facilitando el trabajo de los analistas al elegir la refactorización correcta en la iteración actual. Si bien estos resultados cumplen con las expectativas,

cabe destacar que la función que rige el ordenamiento puede ser objeto de futuras evaluaciones para mejorar los resultados, o adaptarlos a distintas situaciones.

V. CONCLUSIONES

En este trabajo se presentó un enfoque semi-automático para la refactorización de casos de uso, que tiene como objetivos principales solucionar los inconvenientes asociados a la detección de relaciones entre casos de uso, y asistir al analista funcional en el proceso de mejora de especificaciones de casos de uso.

Para poder validar el enfoque se desarrollaron cinco casos de estudio, cuyos resultados fueron alentadores. El punto más destacado es que la herramienta fue capaz de recuperar la mayoría de los problemas detectados por el análisis manual realizado expertos. Otro aspecto surge al considerar la efectividad en la obtención de refactorizaciones señaladas por los expertos. Respecto al sistema de puntuación que permite generar la lista de recomendaciones ordenadas en base a su criticidad, se llegó a la conclusión de que este ordenamiento tiende a presentar en las primeras posiciones de la lista un porcentaje alto de resultados correctos.

Como trabajo futuro consideramos necesario realizar más experimentación y contar con más casos de estudio para confirmar los resultados. Adicionalmente, estamos analizando alternativas para refinar la jerarquía de acciones de dominio para describir mejor la semántica de las especificaciones.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por la Comisión de Investigaciones Científicas (CIC) de la provincia de Buenos Aires mediante el Programa de Subsidios Proyectos de Investigación Científica y Tecnológica (Convocatoria 2013 – Resolución Nº 813/13).

REFERENCIAS

[1] K.E. Wiegers and J. Beatty, “Software Requirements”, Microsoft Press, Redmond, USA, 3rd edition, 2013, ISBN 0735679665, 9780735679665.

[2] R. Van Solingen and E. Berghout. “The Goal/Question/Metric Method: a practical guide for quality improvement of software development”, McGraw-Hill, 1999.

[3] R. Ramos, J. Castro, F. Alencar, J. Araújo, A. Moreira, C. de Engenharia da Computacao and R Penteado, “Quality improvement for use case model”, In Software Engineering, 2009. SBES’09. XXIII Brazilian Symposium on, pp. 187–195, IEEE, 2009.

[4] A. Ciemniewska and J. Jurkiewicz, “Automatic detection of defects in use cases”, Master’s thesis, Poznan University of Technology, Faculty of Computer Science and Management, Institute of Computer Science, 2007.

[5] K. Rui and G. Butler, “Refactoring use case models: the metamodel”, In Proceedings of the 26th Australasian Computer Science Conference, vol. 16, pp. 301–308, Australian Computer Society, Inc., 2003.

[6] W. Yu, J. Li and G. Butler, “Refactoring use case models on episodes”, In Proceedings of Automated Software Engineering, 19th International Conference on, pp. 328–335, IEEE, 2004.

[7] C.D. Manning, H. Schutze and MITCogNet, “Foundations of statistical natural language processing”, vol. 59, MIT Press, 1999.

[8] T.M. Mitchell, “Machine Learning”, McGraw-Hill series in Computer Science, McGraw-Hill, 1997, ISBN 9780071154673.

[9] C. Manning, P. Raghavan and H. Schuutze, “Introduction to Information Retrieval”, Cambridge University Press, 2008.

[10] J. A. Moustafa, “Jaligner: Open source java implementation of smith-waterman”, http://jaligner.sourceforge.net.

[11] T. Smith and M. Waterman, “Identification of common molecular subsequences”, Journal of Molecular Biology, vol. 147, no. 1, pp. 195–197, 1981.

[12] O. Gotoh, “An improved algorithm for matching biological sequences”, Journal of molecular biology, vol. 162, no. 3, pp. 705–708, 1982, ISSN 0022-2836.

[13] A. Rago, P. Frade, M. Ruival and C. Marcos, “Un Enfoque para Automatizar la Refactorización de Casos de Uso”, Proceedings of Argentine Symposium on Software Engineering (JAIIO'13), pp. 183-197, Córdoba, Argentina, Septiembre 2013.

[14] P. Greenwood, “Tao - A testbed for aspect oriented software development”, http://www.comp.lancs.ac.uk/greenwop/tao/, 2011.

[15] R. Bell, “Course Registration System”, http://sce.uhcl.edu/helm/RUP_course_example/courseregistrationproject/indexcourse.htm, 2011.

Claudia Marcos es profesora e investigadora de la Fac. de Ciencias Exactas de Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina) desde 1991. Ella obtuvo el título de Ingeniero de Software (UNCPBA) en el año 1993 y el título de Doctora en Ciencias de la Computación (UNCPBA) en el año 2001. Además, es investigadora de CIC. Sus principales áreas de investigación son Evolución de Sistemas, Ingeniería de Requerimientos, UML y Desarrollo Ágil. Es profesora de varios cursos de grado y posgrado y ha dirigido varias tesis de grado y posgrado.

Alejandro Rago es estudiante de doctorado en Ciencias de la Computación en la Fac. de Ciencias Exactas de Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina). Sus principales áreas de investigación son al automatización de las actividades de la ingeniería de software, específicamente en las primeras etapas del ciclo de vida. Alejandro obtuvo si título de grado en Ingeniería de Sistemas y maestría en Ciencias de la Computación en Fac. de Ciencias Exactas de Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina).

J. Andrés Díaz-Pace es profesor de la Fac. de Ciencias Exactas de Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina) e investigador del Consejo Nacional de Investigaciones Científicas y Técnicas de Argentina (CONICET). Del 2007 al 2010, fue miembro del Instituto de Ingeniería de Software (SEI, Pittsburgh, USA). Obtuvo el título de Dr. en Ciencias de la Computación en el 2004 en la Fac. de Ciencias Exactas de Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina). Sus principales áreas de interés son diseño de arquitecturas dirigidas por la calidad, técnicas de IA aplicadas al diseño. Es profesor de varios cursos de grado y posgrado y ha dirigido varias tesis de grado y posgrado.