RISI 6(2), 27 - 36 (2011)
REVISTA DE INVESTIGACIÓN DE SISTEMAS E INFORMÁTICA
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

ISSN 1815-0268 (versión impresa)
ISSN 1816-3823 (versión electrónica)

Un modelo de Razonamiento Basado en Casos para la captación de requisitos en el desarrollo de proyectos de software

A model of case-based reasoning to capture requirements in the development of software projects

[1] Lenis R. Wong, [2] David S. Mauricio, [3] Erik A. Papa

Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos

lwongp@yahoo.es[1], dms_research@yahoo.com[2], erikpapa@gmail.com[3]

RESUMEN

Los estudios muestran que una de las causas de los atrasos de los proyectos de software se encuentra en la Captación de Requisitos. También se sabe que el costo para reparar un error en los requisitos es 5 a 10 veces menos que en la codificación y 200 veces menos que en el mantenimiento. Por ello existen muchos esfuerzos para mitigar este problema. Como por ejemplo: herramientas y técnicas que facilitan el trabajo en la administración y gestión de requisitos, tales como *RUP, REM, Entrevistas, Encuestas, Casos de usos, Win Win*, etc. sin embargo según los últimos reportes siguen los problemas. es por ello que en el presente trabajo proponemos un *Modelo de razonamiento basado en casos para la captación de requisitos en el desarrollo de proyecto de software.* Con esta técnica de inteligencia artificial, se aprovecha los requisitos funcionales de proyectos de software desarrollados anteriormente, para resolver o identificar requisitos funcionales de un nuevo proyecto. Para este fin los *requisitos funcionales* son representados como "*casos".* en el diseño CBR se ha contemplado las tareas: representar el caso, modelar la Librería de Casos (LC), definir la técnica de indexación y llenar la LC con requisitos de anteriores proyectos. El modelo también considera los procesos de recuperar, reutilizar, revisar y retener un requisito. Los estudios de casos sobre 20 requisitos de dos proyectos muestran que el modelo propuesto reduce en 79.84% el tiempo de captación de requisito obtenido por la metodología RUP.

Palabras clave: Razonamiento basado en casos, Requisito, Captación de Requisitos, Librería de Casos.

ABSTRACT

studies show that one of the causes of delay in software projects is in the Collection Requirements. We also know that the cost to repair a requirements error is 5 to 10 times less than in the coding and 200 times less than in maintenance. Therefore, there are many efforts to mitigate this problem. For example: tools and techniques that make working in the administration and management requirements, such as RUP, REM, Interviews, surveys, Case uses, Win Win, and so on. However according to the latest reports are the problems. That is why in this paper we propose a model case-based reasoning to capture requirements in the software development project. With this artificial intelligence technique, is used the functional requirements of software projects previously developed to solve or identify functional requirements of a new project. For this purpose the functional

requirements are represented as “cases”. The design has provided CBR tasks: represent the case, model Case Library (LC), define the indexing technique and fill the requirements of LC with previous projects. The model also considers the processes of recovery, reuse, revise and retain a requirement. The 20 case studies of two projects requirements show that the proposed model reduces to 79.84% the time required to capture obtained by the RUP.

Keywords: Case-based reasoning, Requirement, Requirements elicitation, Library cases.

1. INTRODUCCIÓN

según el reporte del caos del año 1995 [11], no todos los proyectos de software son terminados con éxito. solo el 16 % de los proyectos son terminados con éxito y el 31 % de estos son cancelados. existen muchos factores de los fracasos de estos proyectos. Una de ella es *Requerimientos incompletos*. Según la figura 1 los *Requerimientos incompletos*. Representan el 13.1 % de la causa de los fracasos.

Fig. 1. The standish Group Report, “The Chaos Report”, 1995.

Para mitigar este problema existen muchas herramientas y técnicas que facilitan el trabajo en la administración de requisitos, tales como *Entrevistas* [1], *Encuestas*[1], *Brainstorming* [8], *Prototipación* [1], *RUP* [10], *JAD* [5] *y Win Win* [4], *etc*. sin embargo según el reporte del caos del año 2009 siguen los problemas [12].

es por ello que proponemos un *Modelo de razonamiento basado en casos para la captación de requisitos en el desarrollo de proyecto de software.* Con esta técnica de inteligencia artificial, se aprovecha los requisitos funcionales de proyectos de software desarrollados anteriormente, para resolver o identificar requisitos funcionales de un nuevo proyecto.

Los sistemas de Razonamiento Basado en Casos se han podido aplicar con éxito a gran cantidad de dominios, desarrollando gran cantidad de tareas, tanto en el campo académico como en la industria. Tales como: en la planificación, en los diagnósticos, en el diseño, etc. También se han aplicado *Razonamiento Basado en Casos* con éxito, en la *ingeniería de software*, como por ejemplo: en la optimización de la calidad del software [3], en la estimación de los esfuerzos en el desarrollo de proyectos de software [7], etc.

el presente trabajo está organizado de la siguiente manera. en la sección 2 se presenta un resumen del estado de Arte. La sección 3 describe el modelo propuesto. en la sección 4 se describe la implementación del CBR. y en la sección 5 se presente dos casos de estudio y sus resultados. Finalmente las conclusiones son descritas en la sección 6.

2. ESTADO DE ARTE

Hoy en día existen varias técnicas utilizadas en la captación de requisitos, entre las más usadas se encuentran: *Entrevistas* [1], *Encuestas*[1], *Brainstorming* [8], *Prototipación* [1], *RUP* [10], *JAD* [5] *y Win Win* [4]. Con el transcurso de los tiempos y el empleo de cada una estas técnicas se puede observar que cada una tiene puntos muy relevantes.

Por ejemplo con las *Entrevistas* se obtiene una gran cantidad de información adecuada a través del usuario ya que estas son flexibles. Las *Encuestas* pueden ser reutilizadas en nuevos proyectos ya que permite obtener información de una gran cantidad de personas de una manera rápida y esta información puede ser reutilizada en nuevos proyectos. La técnica *Brainstorming* o *Tormenta de Ideas*, ayuda a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo, cuando los requerimientos son todavía muy difusos. Los *Casos de Uso* representan los requerimientos desde el punto de vista del usuario, por ello son fácilmente comprensibles. Además, pueden servir de base en las pruebas y documentación del sistema. Con la técnica *JAD*, la información obtenida se puede contrastar *in situ* porque están todos los interesados involucrados, tanto desarrolladores y usuarios finales. La Prototipación es útil cuando el prototipo puede ser construido rápidamente.

Usando la técnica Win Win el desarrollo de la captación de requisitos es continuo y colaborativo, donde la refinación de los requisitos se da en cada ciclo.

Analizando cada una de estas técnicas podemos observar que la información que ellas obtienen la registran de diferentes maneras. Por ejemplo la información obtenida con las entrevistas es registrada en los apuntes que realiza el entrevistador. en cambio en las encuestas, dicha información se encuentra en los resultados de los cuestionarios.

Usando la técnica Brainstorming, la información obtenida es registrada en alguna plantilla o documento. Mientras que en los Casos de Uso, toda se registrada en los gráficos y notaciones que esta técnica propone.

sin embargo todas estas técnicas están usando Razonamiento Basado en Casos indirectamente, ya que ellas almacenan la información obtenida y luego lo usan para resolver un nuevo problema. Pero esto no está automatizado. Por ejemplo, los expertos que usan la técnica Brainstorming, en sus reuniones usan experiencias vividas anteriormente para resolver el problema específico de la reunión.

El Razonamiento Basado en Casos (CBR) es una técnica de Inteligencia Artificial que permite aprovechar la experiencia acumulada en la solución de nuevos problemas. Con esta técnica se almacenan casos con la solución que se ha dado anteriormente y cuando se presenta un nuevo problema esta información o experiencia acumulada es empleada para resolverlo [6].

se han aplicado razonamiento basado en casos con éxito en la ingeniería de software, como por ejemplo: Adam Brady y Tim Menzies [3], realizan una comparación entre dos herramientas aplicado en la *optimización de la calidad del software*.

3. MODELO PARA LA CAPTACIÓN DE REQUISITOS APLICANDO RAZONAMIENTO BASADO EN CASOS

en esta sección proponemos un modelo de Razonamiento Basado en Casos, que esta compuesto por las siguientes partes: estudio del dominio, Actividades del desarrollo de requisitos, Diseño CBR, Ciclo CBR tal como se muestra en la figura 2.

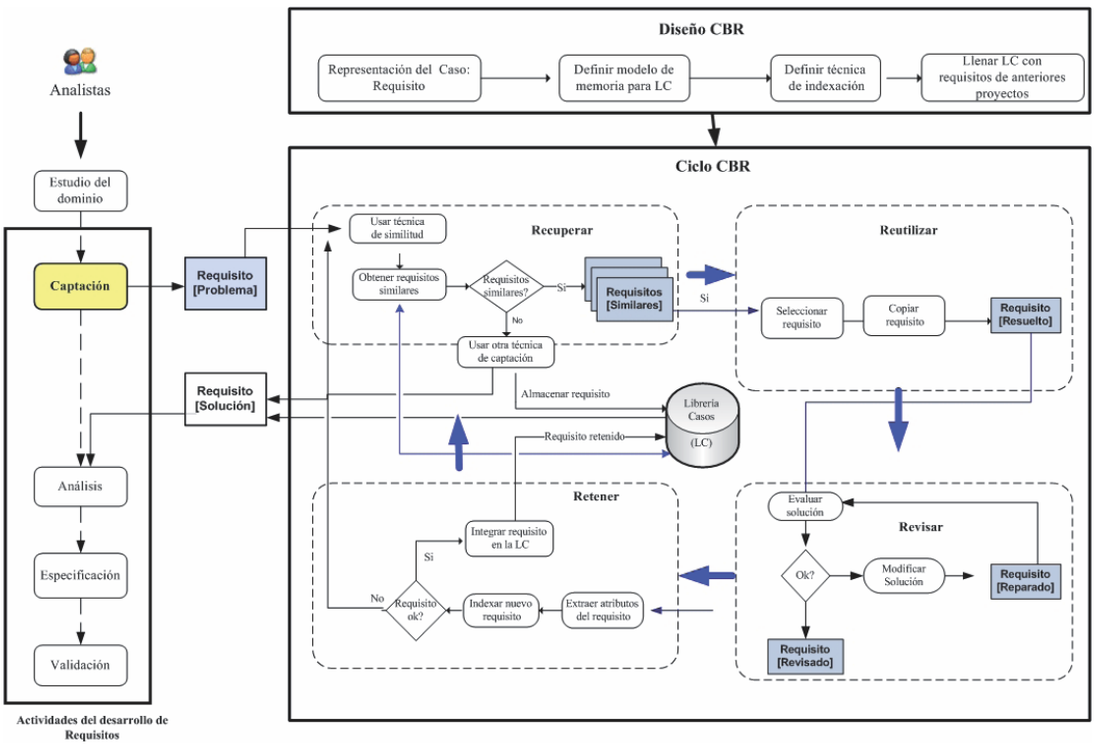

Fig. 2. Modelo propuesto.

en el estudio del dominio, el analista deberá hacer un estudio del negocio según el tipo de proyecto que va desarrollar. Se propone usar BPMN para definir los procesos de negocio. BPMN es una notación gráfica estandarizada basada en diagramas de flujo para definir procesos de negocio. Modela tanto la secuencia de actividades como los datos o mensajes intercambiados entre los distintos participantes de un proceso [9].

Tal como se muestra en la figura 2, las actividades del desarrollo de requisitos son: captación, análisis, especificación y validación. Siendo la primera el centro de atención del modelo propuesto.

En la Captación el analista deberá identificar el requisito, este requisito o caso es el problema que el analista debe resolver. en el modelo que se propone, la captación de requisitos se realizará usando la técnica de Razonamiento Basado en Casos si se encuentran algún requisito igual o similar en la Librería de Casos (LC), de lo contrario se deberá usar una usual técnica para la captación de requisitos del nuevo caso (recomendamos usar Win Win) y luego deberá ser registrado en LC para su futuro uso.

en el Diseño CBR se realizaran todas las tareas relacionadas con la construcción de CBR tales como: Representar el caso (requisito), definir el modelo de memoria para la LC, definir la técnica de indexación y llenar la LC con requisitos de anteriores proyectos.

en el ciclo CBR, se realizan todas las tareas y técnicas involucradas en cada uno de los procesos del ciclo de vida de CBR, por ello se toma como base el modelo de procesos del ciclo de vida CBR propuesto por [2], donde se usan las cuatro etapas (Recuperar, Reutilizar, Revisar y Retener) y el modelo de descomposición Tareas-Métodos.

en el presente trabajo nos concentraremos en el diseño de CBR y su ciclo de vida CBR aplicado a la captación de requisitos.

3.1 Representación de los casos

La representación del caso estará compuesta por atributos del problema y la solución, cada atributo tienen asociados nombre, valor y pesos, los pesos indican la importancia del atributo en el caso. el caso se compone de la siguiente manera: Caso = (atributos 1, atributos 2, ..., atributo n), donde el atributo n representa la solución de nuestro caso [6]. en el presente modelo, el caso estará representado por el Requisito funcional del sistema.

Un caso nuevo será representado por r'. Mientras que un caso que se encuentre en la librería de casos será representado por r. Los nombres de los atributos del caso (requisito) son los siguientes: Proyecto, Dominio, Modulo, Verbo, Sustantivo, Actor 1, Actor 2, Área y SR (Especificación del requisito). Siendo este último la solución del problema, ver figura 3.

Por ejemplo, si nuestro caso es un requisito del sistema para registrar datos de una persona, los atributos del requisito serían: sistema de Planillas, Comercial, Recursos Humanos, Planillas, Registrar, persona, encargado de Planillas, Jefe de Recursos Humanos, "?". Donde el símbolo "?" representa el problema.

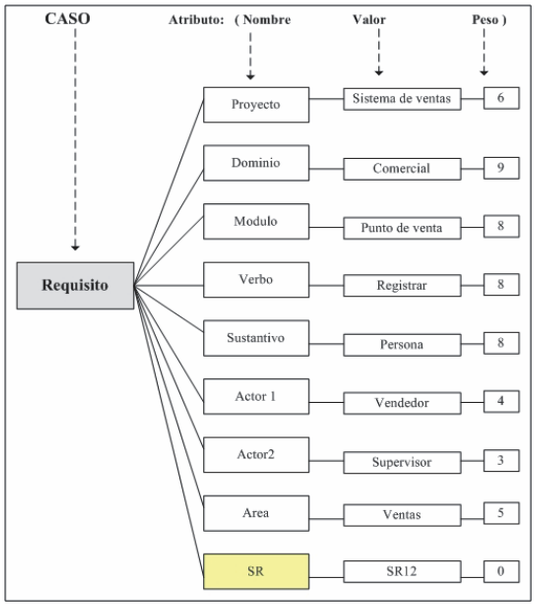

Fig. 3. estructura del Caso

3.2 Indexación de Casos

Para indexar los casos se propone el algoritmo que se muestra en la tabla 1. este algoritmo se usará en el proceso de Recuperación. Para tal tarea se organizan jerárquicamente los casos de manera que sólo un pequeño subconjunto de ellos tiene que ser buscado durante la recuperación.

Primero se debe seleccionar el atributo más importante como índice: en este caso es "Dominio". este atributo es considerado como importante porque los requisitos van a ser diferentes de acuerdo al tipo de sistema. Por ejemplo un requisito de un sistema comercial no será igual a un requisito de un sistema bancario. Luego se debe clasificar los casos en clases o categorías de acuerdo a este atributo: Dominio. Donde las categorías de casos se denotarán de la siguiente manera: CC = {C1,C2,C3,…}. La entrada de un caso nuevo, estará compuesta por un conjunto de atributos denotadas de la siguiente manera: r' = {f1,f2,f3,f4,f5,f6,f7,…,f10}, ver tabla 1.

3.3 RECUPERAR REQUISITOS SIMILARES

en esta etapa se aplica el algoritmo que se muestra en la tabla 2. este proceso comienza cuando hay como entrada un problema, es decir un requisito r' (caso nuevo). Primero se debe indexar los requisitos, para poder recuperar los requisitos similares a nuestro problema r', es por ello que se propone usar el algoritmo *Indexar_Requisito* que se muestra en la tabla 1. Luego se usa una medida de similitud para seleccionar los casos similares a nuestro problema r'. Para ello se utiliza la formula de la Distancia euclideana Ponderada [7]. Finalmente se obtiene los requisitos más similares de acuerdo al grado de similitud. sino existen casos simi-

Tabla 1. Algoritmo para indexar un requisito.

Indexar _ Requisito
Inicio
Entrada: r' = {f1, f2, f3, f4, f5, f6, f7,…f10} //Caso nuevo: Requisito
salida: Ci // Categoría de casos
1. Desde i = 1, seleccionar un caso rj al azar de la categoría Ci
2. Comparar el valor de la característica "Dominio" de r' y rj
3. si los dos valores son idénticos
   entonces r'□ Ci,
   Si no i = i+1, ir al paso 1
4. Repetir pasos 1, 2 y 3 hasta que r' pertenezca a algunas categoría Ci.
Fin

Tabla 2. Algoritmo para recuperar un requisito

Recuperar _ Requisito
Inicio
//entrada: **r'** = {f1, f2, f3, f4, f5, f6, f7,..f10} Requisito o caso nuevo
//salida: *CS:* Conjunto de Casos similares

1. ejecutar el algoritmo de la tabla 1 para tener una categoría de casos CC
   // Calcular el grado de similitud entre *r' y rj*
2. Para j=1 hasta k
   SIM(r', rj)
   //Donde:
   // i: contador
   // k: es el número de casos que hay en la clase de casos CC
   // sIM: es la medida de similitud
3. si existen casos similares
   - Ordenar los casos de acuerdo a la medida de similitud (SIM) calculados en el paso 2.
   - escoger los casos mas similares de acuerdo al grado de similitud
4. sino existen casos similares
   - Usar otra técnica de captación de requisitos

Fin

lares, es decir el caso es totalmente nuevo, se propone usar otra técnica de captación, podría ser RUP.

3.3REUTILIZAR REQUISITO

en esta etapa se aplica el algoritmo que se muestra en la tabla 3. Primero, se selecciona el requisito con grado de similitud alta y se copia esta información como parte de la solución de r'. Teniendo así el requisito resuelto para ser analizado en la siguiente etapa.

Tabla 3. Algoritmo para Reutilizar requisitos

Reutilizar _ Requisito
Inicio
// entrada: Cs = r1 ,r2, r3... rk Conjunto de Casos similares
// salida: ri requisito resuelto ó
1. seleccionar el requisito *ri* con grado de similitud alta
2. Copiar la solución del requisito ri a r'
3. Ir a la siguiente etapa

Fin

3.4REVISAR REQUISITO

en esta etapa se aplica el algoritmo que se muestra en la tabla 4. Después de tener la solución propuesta o requisito resuelto para nuestro problema, es muy importante saber si la solución propuesta en la etapa anterior es la adecuada, es decir, su validez debe ser probada con la realidad. Por ejemplo, si en muestro problema r' se plantea la solución de un requisito r1, donde el valor de su solución es el la especificación de requisito SR6, esta SR debe ser probado por el analista de sistema.

es por ello que primero se evalúa el valor de la solución del requisito obtenido como solución en la etapa Reutilizar. Por ejemplo, se va evaluar si el valor del atributo SR es una solución para nuestro problema r'. si es conforme a la realidad del problema, se pasará a la siguiente etapa o proceso del ciclo de vida de CBR. si no es conforme el valor del atributo "SR", se puede reparar este valor y se continuar al paso 1 según el algoritmo. Finalmente, si después de reparar el valor del atributo "SR" y aun así no es conforme la solución a nuestro problema, se descarta el requisito.

3.5 RETENER REQUISITO

en esta etapa se aplica el algoritmo que se muestra en la tabla 5. Primero se extraer los atributos del requisito revisado en la etapa anterior, se analiza que información que proporciona el requisito revisado ri se usará para nuestro problema *r'*, es decir que atributos del requisito *ri* tomaremos como parte de nuestra. Luego, se debe indexar el nuevo caso r' para que el futuro se pueda acceder a él sin problemas. Para esta tarea se usará el algoritmo que se muestra en la tabla 1. Luego se verifica nuevamente si la información de nuevo caso r' (ya incorporado con la solución de ri) es conforme con la realidad. Finalmente si la solución de r' es conforme,

Tabla 4. Algoritmo para Revisar requisitos

Revisar _ Requisito
Inicio
// entrada : *ri* = {f *i* 1, f *i* 2, f *i* 3, f *i* 4, f *i* 5,.. f*i* 10} requisito resuelto
// salida: *ri* revisado ó , 0: requisito descartado

evaluar la el valor de solución de requisito *ri:*
Si *fi 10* esta conforme a la realidad
entonces *ri* será la solución *de r' //*Ir a la siguiente etapa
Sino
Reparar *fi 10*
Si *fi 10 reparado es conforme*
entonces *Ir al paso 1*
Sino
Descartar Requisito
Fin

Tabla 5. Algoritmo para Retener Requisitos

Retener _ Requisito
Inicio
// entrada : *ri* = {f *i* 1, f *i* 2, f *i* 3, f *i* 4, f *i* 5,.. f*i* 10} Requisito revisado
// salida: *r'* retenido ó para la siguiente etapa.

1. extraer atributos del requisitos *ri*
2. Copiar estos atributos al requisito *r'*
3. Indexar nuevo requisito *r' //* usar algoritmo de indexación
4. Revisar requisito *r'*
5. **Si** *r'* esta conforme
   entonces incorporarlo a la LC
   Sino
6. Ir a la etapa *Recuperar*

Fin

se integra a la librería de casos. si el requisito r' no es conforme, vuelve nuevamente a la etapa Recuperar, para seguir el ciclo CBR nuevamente.

4. IMPLEMENTACIÓN

A continuación describimos el inicio del aporte práctico de nuestra investigación, presentando los pasos a seguir para la implementación del modelo propuesto en la sección anterior. Para la creación de la LC, la representación de los casos y el desarrollo del ciclo de vida CBR usaremos una herramienta que hemos implementado para la representación del presente modelo.

a. Creación de la Librería de Casos

Para poder diseñar nuestro modelo lo primero que se debe hacer es crear la Librería de Casos, para ello se debe escoger la opción Librería de Casos de la herramienta.

b. Representación del Caso

Para la representación del *caso*, es decir de nuestro *requisito*, debemos configurar todos sus atributos: etiquetas, pesos y valor. Tal como se explicó en la propuesta de nuestro modelo. Para ello seleccionamos la opción *Representación del Caso* (ver figura 3).

Un ejemplo se muestra en la figura 4, donde se ingresan todos los atributos de nuestro caso. Para una búsqueda eficiente se debe ingresar los pesos de cada uno de los atributos, ya que esta herramienta usa la formula de *Distancia Euclidean Ponderada* para calcular la similitud de cada uno de los requisitos, tal como se propone en la sección anterior del presente trabajo.

Fig. 4. Representación del caso

c. Recuperar y Reutilizar el Caso

Para recuperar y luego reutilizar los requisitos, se debe entrar a la opción *Recuperar-Reutilizar* de la herramienta. Como se explico en el punto anterior, para recuperar los requisitos similares que se encuentran en nuestra LC, primero se debe ingresar el *requisito* o *caso* que se desea resolver.

Un ejemplo se muestra en la figura 5, donde se quiere hallar la solución del requisito cuyos atributos son: sistema de Ventas, Comercial, Punto de Venta, Registrar, Persona, Vendedor, supervisor,

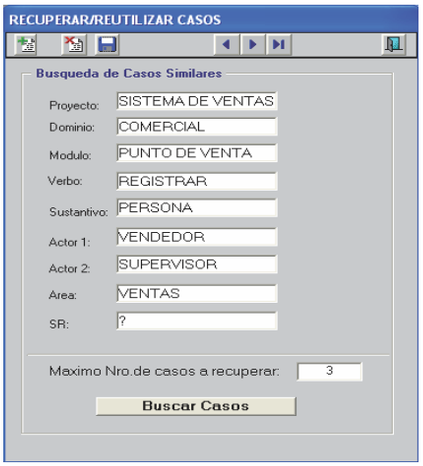

Fig. 5 Recuperar

y Venas. Tal como explicamos en la propuesta de nuestro modelo, la solución del requisito estará representada por el atributo SR (Especificación del Requisito), es por ello que este atributo se denota con el carácter "?". Luego de ingresar los atributos del requisito o problema que se desea resolver, se debe ingresar el número de casos similares que se quiere mostrar. Para ello ingresamos este número en el campo: *Máximo Nro. de casos ha recuperar* y presionamos el botón *Buscar Casos*.

Después del paso anterior, la herramienta mostrará la ventana dada en la figura 6, con los resultados de la búsqueda, en orden a la similitud de cada uno de los requisitos almacenados en la librería de casos.

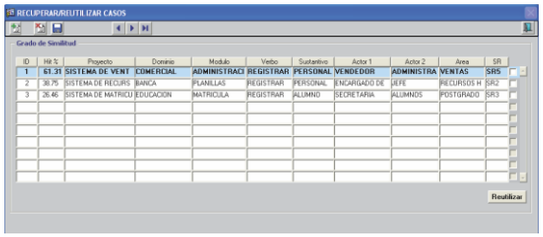

Fig. 6 Resultados de Recuperar (reutilizar el caso)

Para reutilizar la información del caso o requisito que es similar o igual a nuestro problema, se debe seleccionar el requisito que tenga un buen porcentaje de similitud (Hit %), para ello se debe marcar el requisito y presionar el botón Reutilizar. Un ejemplo se muestra en la figura 6. Luego de presionar el botón Reutilizar, el sistema preguntará si esta seguro de reutilizar este *caso*. Finalmente la información de este *caso* se utilizará como parte de la solución de nuestro *requisito*.

d. Revisar el Casos

Para que la posible solución del *requisito* sea revisado por un analista se debe cambiar el estado de Reutilizado a Revisado. Para ello se ingresará a la opción *Revisar* de la herramienta.

Un ejemplo se muestra en la figura 7, en esta ventana se observa que el caso que queremos resolver, se encuentra en el estado *Reutilizado* y esta asignado el ID del caso que estamos usando como posible solución. Para cambiar el estado del requisito a *Revisado* solo se debe escoger la opción *Estado* y seleccionar el estado *Revisado*.

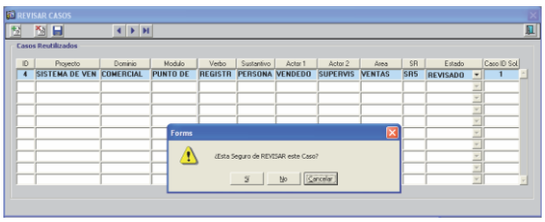

Fig. 7. Revisión del caso

Como ahora el *requisito* se encuentra en estado *Revisado*, el analista podrá verificar la información del SR del caso escogido como posible solución. si esta información es la correcta, este atributo se usará como posible solución sin ningún cambio. en caso contrario, si se tiene que hacer algunas modificaciones al SR, este atributo se guardará como parte de la solución del requisito pero con otro nombre. Luego el requisito podrá pasar a la siguiente etapa.

c. Retener el Caso en la LC

Finalmente si el requisito ya fue revisado por el analista podrá pasar a esta etapa, se debe cambiar el estado de *Revisado* a *Retenido*. Para ello se ingresará a la opción *Revisar* de la herramienta.

Un ejemplo se muestra en la figura 8, en esta ventana se observa que el caso que queremos resolver, se encuentra en el estado *Revisado*. Para cambiar el estado del requisito a *Retenido* solo se debe escoger la opción *Estado* y seleccionar el estado *Retenido*. De este modo la información del requisito se copiará en la LC, con la solución obtenida.

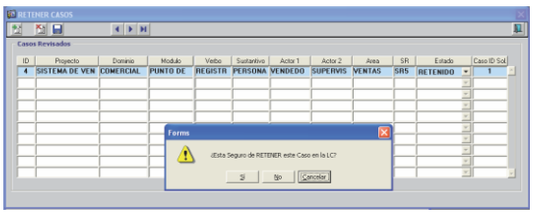

Fig. 8. Retención del Caso

si escogemos la opción *Casos* de la herramienta, podemos consultar todos los casos de nuestra LC. Como ejemplo (ver figura 9 podemos observar que el caso resuelto se encuentra con el *ID* 4, este caso tiene como solución sR5 del caso 1.

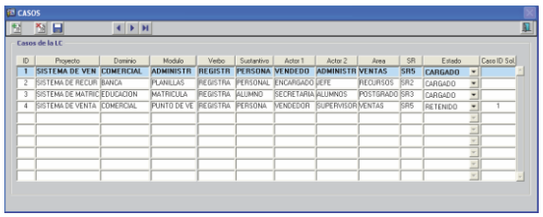

Fig. 9. Casos de la LC

5. PRUEBAS DEL MODELO

Caso de Estudio 1: Sistema de Recursos Humanos

el siguiente caso de estudio trata sobre la captación de requisitos del sistema de Recursos Humanos para una entidad comercial. Tal como se muestra en la tabla 6, se han considerado 10 requisitos: registrar empleado, aperturar planilla, cerrar planilla, generar planilla, generar boleta, generar liquidaciones, registrar licencias, registrar vacaciones, imprimir planilla y generar PDT. Los resultados obtenidos usando la metodología RUP es de 650 minutos y aplicando el modelo propuesto basado en CBR es de 123.50 minutos.

Tabla 6. Cuadro comparativo RUP vs CBR del caso de estudio 1.

Requisitos	RUP (*)	CBR (*)
R1. Registrar empleado	50.00	12.15
R2. Aperturar Planilla	30.00	7.15
R3. Cerrar Planilla	30.00	7.15
R4. Generar Planilla	120.00	17.15
R5. Generar Boleta	60.00	14.15
R6. Generar Liquidaciones	80.00	16.15
R7. Registrar Licencias	70.00	12.15
R8. Registrar Vacaciones	90.00	14.15
R9. Imprimir Planilla	20.00	5.15
R10.Generar PDT	100.00	18.15
Totales	650.00	123.50

(*) Tiempo en minutos

Caso de Estudio 2: Sistema de Punto de Venta Delivery

el segundo caso de estudio trata sobre la captación de requisitos del sistema de Punto de Venta Delivery, para una cadena de farmacias. Tal como se muestra en la tabla 7, también se han considerado 10 requisitos: registrar cliente, registrar empleado, registrar venta, generar comprobante, apertura caja, consultar ventas, consultar clientes, consultar empleados e imprimir comprobantes. Los resultados obtenidos usando la metodología RUP es de 535 minutos en total y aplicando el modelo propuesto basado en CBR es de 115.5 minutos.

6.CONCLUSIONES y FUTUROS TRABAJOS

este trabajo hace una propuesta de un *Modelo de razonamiento basado en casos para la captación de requisitos en el desarrollo de proyecto de software.* Aplicando *CBR* se aprovechó los requisitos funcionales de proyectos de software desarrollados anteriormente, para resolver o identificar requisitos funcionales dos nuevos proyectos. Analizando los resultados obtenidos en los casos de estudios, se puede

Tabla 7. Cuadro comparativo RUP vs CBR del caso de estudio 2.

Requisitos	**RUP (*)**	**CBR (*)**
R1. Registrar Cliente	40.00	9.15
R2. Registrar empleado	50.00	12.50
R3. Registrar Venta	110.00	20.15
R4. Generar Comprobante	100.00	18.15
R5. Apertura Caja	30.00	7.15
R6. Cerrar Caja	30.00	7.15
R7. Consultar Ventas	40.00	9.15
R8. Consultar Clientes	45.00	10.15
R9. Consultar empleados	50.00	12.15
R10.Imprimir Comprobante	40.00	10.15
Totales	535.00	115.5

(*) Tiempo en minutos

observar la factibilidad del empleo de esta propuesta, en las pruebas realizadas hasta el momento. Los trabajos futuros que podemos considerar son los siguientes: agregar pesos a los casos que han dado solución a problemas, en la etapa de *Recuperación* del ciclo de vida CBR. e integrar este modelo con las otras tareas de ingeniería de Requisitos.

REFERENCIAS

[1] [Andriano 2006] Andriano N. "Comparación del Proceso de elicitación de Requerimientos en el desarrollo de software a Medida y empaquetado. Propuesta de métricas para la elicitación". Tesis Magistral, Universidad Blas Pascal Córdoba, Argentina, (2006).

[2] [Aamodt+ 1994] Aamodt A., Plaza E.,1994 "Case-Based Reasoning: Foundational Issues, Methodological Variations, and system Approaches", Artificial Intelligence Communications, Vol. 7, (1994), pp.39-52.

[3] [Brady+ 2010] Brady A., Menzies T., "Case-Based Reasoning vs Parametric Models for software Quality Optimization", ACM, (2010), pp. 1-10.

[4] [Chao+ 2009] Chao D., Ming W. "Distributed Requirement elicitation and Negotiation Based on the Hall for Workshop of Meta-synthetic engineering". ACM, (2009), pp. 1-5. http://dl.acm.org/citation.cfm?id=1656193

[5] [Dube+ 2010] Dube R., Dixit S. "Process-oriented Complete Requirement engineering Cycle for Generic Projects". International Conference and Workshop on emerging Trends in Technology. ACM. India, (2010).

[6] [Juárez+ 2005] Juárez J., Palma J., "Inteligencia Artificial": Razonamiento Basado en Casos, Universidad de Murcia, Vol. 74, (2005), Murcia-España.

[7] [Mendes+ 2002] Mendes E., Mosley N., Watson I. "A comparison of case-based reasoning approaches to Web hypermedia Project Cost estimation". ACM, (2002), pp. 1-9. http://portal.acm.org/citation.cfm?id=511446.511482&coll=DL&dl=GUIDe&CFID=31271263&CFTOKeN=12937670

[8] [Raghavan+ 1994] S. Raghavan, G. Zelesnik, y G. Ford. Lecture Notes on Requirements elicitation. educational Materials CMU/seI–94–eM–10, software engineering Institute, Carnegie Mellon University, 1994.http://www.sei.cmu.edu.

[9] [Dijkman+ 2008] Dijkman R., Dumas M., Ouyang CH. "semantics and analysis of business process models in BPMN". ACM, Volume 50 Issue 12, (2008).

[10] [Rational 1998] Rational. "Rational Unified Process Best Practices for software Development Teams". IBM (Rational Software Company), (1998). http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[11] [standishGroup 1995] standish Group "The Chaos Report 1995". (10/10/2011). https://cs.nmt.edu/~cs328/reading/standish.pdf

[12] [standishGroup 2009] standish Group "The Chaos Report 2009". (10/10/2011). http://www1.standishgroup.com/newsroom/chaos_2009.php