risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Recebido/Submission: 12/07/2019
Aceitação/Acceptance: 18/10/2019

Razonamiento Basado en Casos para asignación óptima de tareas de desarrollo de software

Guillermo Rodríguez[1], Luis Berdun[1], Leonardo Da Rocha Araujo[1]

{guillermo.rodriguez, luis.berdun, leorardo.araujo}@isistan.unicen.edu.ar

[1]ISISTAN (UNICEN-CONICET), Campus Universitario – Paraje Arroyo Seco, 7000, Tandil, Argentina.

DOI: 10.17013/risti.35.116–131

Resumen: La administración de proyectos significa mucho más que dividir el trabajo para asignar las partes a diferentes personas. De hecho, con frecuencia, los proyectos que pudieron haber sido exitosos fracasan debido a los enfoques que dan por sentado que la administración de proyectos solo implica dividir el trabajo en tareas de menor complejidad, dejando de lado otros aspectos importantes. Este trabajo presenta ARDe (Asistente de Recomendación de Desarrolladores), un herramienta basada en Razonamiento Basado en Casos para construir una solución que asista en la toma de decisiones acerca de qué miembros del equipo del proyecto seleccionar para realizar las tareas establecidas durante la planificación del proyecto. Como resultado de los diversos experimentos, se comprobó que las estimaciones se asemejan a las métricas reales obtenidas de la ejecución y que además ARDe aprende a ordenar las recomendaciones según las preferencias del líder del proyecto para seleccionar a un desarrollador. Esto último es un factor importante que ayuda a mitigar los cambios que se producen ante la ida de un líder de proyecto.

Palabras-clave: Razonamiento Basado en Casos; Administración de Proyectos; Planificación; Métricas de Software.

Case Based Reasoning for optimal assignment of software development tasks

Abstract: Project management means much more than dividing the work to assign parts to different people. Often, projects that may have been successful fail because of the assumptions that project management only involves dividing work into less complex tasks, leaving aside other important aspects. This paper presents ARDe (Developer Recommendation Assistant), a tool based on Case-Based Reasoning to build a solution that assists in making decisions about which members of the project team to select to perform the tasks established during project planning. As a result of various experiments, we found that the estimates resemble the actual metrics obtained from the execution and that ARDe also learns to order the recommendations according to the preferences of the team leader to select a developer. The latter is an important factor that helps mitigate the changes that occur before the departure of a team leader.

***Keywords**:* Case Based Reasoning; Project management; Planning; Software Metrics.

1. Introducción

Existen muchas razones por las que los proyectos de software tienen problemas. La cantidad de trabajo que conlleva un proyecto de software requiere grandes esfuerzos de desarrollo, lo que conduce a complejidad, confusión y dificultades significativas en la coordinación de los miembros del equipo. En este contexto general de incertidumbre, surge un torrente continuo de cambios que detienen al equipo de proyecto. Para ello, deben implantarse métodos efectivos a fin de coordinar al personal que hace el trabajo (Morales-Aguilar, 2018).

El líder de proyectos necesita establecer expectativas claras a los miembros del equipo y que todos conozcan la importancia de su papel para lograr el objetivo del proyecto. De esta manera cada miembro del equipo no sólo puede ver las tareas que ya le han sido asignadas, sino también las de los otros y cómo todas se acoplan (Gido & Clements, 1999).

Para decidir qué tarea asignar a cada miembro del equipo es fundamental contar con un soporte para que el líder de proyecto pueda tomar la decisión de manera más precisa. Los sistemas basados en conocimiento cobran importancia y utilidad en aquellos dominios en los que resulta difícil y costoso disponer de expertos humanos para realizar consultas de manera personal, ilimitada e instantánea (Carignano, 2019). En este contexto, Razonamiento Basado en Casos (CBR, Case-Based Reasoning) surge como una alternativa a explorar para entender nuevas situaciones comparándolas y contrastándolas con situaciones similares dadas en el pasado (Wu, 2018).

En este artículo se presenta ARDe, un Asistente de Recomendación de Desarrolladores basado en CBR, que tiene por objetivo facilitar la toma de decisiones acerca de qué miembros del equipo del proyecto seleccionar para realizar las tareas establecidas durante la planificación del proyecto. Para evaluar ARDe, se utilizaron proyectos reales realizados por 120 estudiantes avanzados de la carrera de Ingeniería de Sistemas de la Facultad de Ciencias Exactas entre los años 2014 y 2015.

Como resultado, ARDe permitió disminuir las fallas dentro del proyecto, debido a que el momento de realizar recomendaciones, no solo consideró los casos positivos sino también los negativos. De esta forma, un desarrollador que ejecutó una tarea similar pero los resultados de la ejecución fueron malos es posicionado en las últimas posiciones del ranking. Esto posibilitó al líder de proyecto de seleccionarlo nuevamente pero disminuyendo la probabilidad de ser elegido ya que se espera que el líder de proyecto seleccione primero a los desarrolladores que se encuentran en las primeras posiciones del ranking.

El resto del artículo se organiza de la siguiente manera. En la sección 2, se describen los trabajos relacionados. En la sección 3, se describe ARDe. La sección 4 analiza los resultados de una evaluación del sistema experto propuesto. Finalmente, en la sección 5, se presentan las conclusiones y trabajos futuros.

2. Trabajos relacionados

Particularmente, CBR ha sido ampliamente usado para la estimación de costos de proyectos de software. La estimación basada en conocimiento previo es una de las técnicas más atractivas en estimación de costos. En este campo, Wu y otros han estudiado el uso de CBR en combinación con Particle Swarm Optimization para estimar el costo de proyectos de software, alcanzado resultados prometedores (Wu, 2018). Kim ha abordado la incertidumbre en la estimación de costos mediante CBR y Analytic Hierarchy Process (Kim, 2013). Simultáneamente, CBR ha sido aplicado para identificar riesgos y mitigarlos a lo largo de un proyecto de software (Cordero Morales, 2013).

Siguiendo la misma línea, Fan y otros han propuesto a CBR como una técnica adecuada para generar estrategias preventivas y controladoras de riesgos en un proyecto de subterráneos de China (Fan, 2015). Stamelos estableció un enfoque basado en CBR, entre otras técnicas de IA, para capturar, almacenar y propagar y evitar anti-patrones en el manejo de proyectos de software, con el objetivo de educar a futuros managers de proyectos (Stamelos, 2010). En esta línea, Koo y otros han propuesto un modelo de CBR para predecir la duración y el costo de un proyecto en su etapa temprana en el contexto de construcción de viviendas, combinando análisis de regresión múltiple, redes neuronales artificiales, algoritmos genéticos, y simulación Monte-Carlo (Koo, 2010). Jin y otros han trabajado sobre la mejora en modelos de predicción de costos en el contexto de la construcción, mediante la combinación de CBR y regresión lineal múltiple (Jin, 2012). Siguiendo en el campo de la construcción, Ji y otros han abordado CBR, en combinación con algoritmos genéticos, para mejorar la precisión en la estimación de costos en la construcción (Ji, 2013).

Si bien el uso de CBR se ha aplicado satisfactoriamente en el campo de la estimación de costos de proyectos, poco se ha hecho para planificar un proyecto teniendo en cuenta el conocimiento previo basado en el desempeño y aptitudes del recurso humano de la organización. Este problema queda explícito cuando los responsables de administrar los proyectos son novatos y/o carecen de suficiente experiencia en el contexto tecnológico del proyecto. En este punto es donde el uso de CBR puede ser clave para optimizar la asignación de tareas a desarrolladores de software.

3. ARDe: Asistente de Recomendación de Desarrolladores

ARDe tiene como objetivo extraer las métricas de análisis de cada desarrollador obtenidas mediante la integración de distintas herramientas utilizadas en entornos de desarrollo de software, entre las que se encuentran un gestor de incidencias, un sistema de control de versiones y un analizador de código estático, con el fin de recomendar un desarrollador basándose en el desempeño del mismo de acuerdo a las tareas que haya realizado.

En la Figura 1 se observa el enfoque adoptado, el cual toma como entrada una tarea específica que desea ser asignada a un desarrollador, un conjunto de métricas con sus valores deseados para la tarea y un conjunto opcional de habilidades que debería poseer el desarrollador para llevar a cabo la misma.

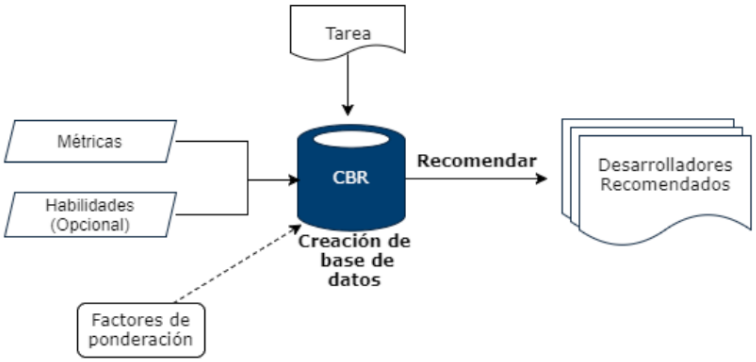

Figura 1 – Esquema general del enfoque de ARDe

Dentro del proceso de CBR, se buscan tareas similares a la tarea a ser asignada teniendo en cuenta la similitud entre la descripción de las tareas y las habilidades necesarias para desarrollar la misma. Se obtienen los desarrolladores correspondientes a esas tareas similares, y se estiman métricas para los mismos. Para la estimación, se utiliza la regresión lineal entre las métricas de tareas ya realizadas por el desarrollador y los valores de métricas deseados por el usuario, solicitados en la interfaz de la recomendación. Una vez que se obtengan las métricas reales, se comparan con las valores de las métricas estimadas. Se listan las tareas aún no asignadas presentes en el sistema, permitiendo al líder del proyecto elegir una tarea del listado.

A partir de CBR, ARDe recomienda miembros del equipo como posibles desarrolladores de la misma, adjuntando a cada uno un conjunto de métricas estimadas que reflejan las consecuencias de que ese desarrollador lleve a cabo la tarea. El líder del proyecto selecciona alguno de los desarrolladores recomendados, teniendo en cuenta el criterio de negocio que más se ajuste al momento de la recomendación. Para la asistencia de la elección se presentan en la interfaz diagramas que representan las métricas estimadas de cada desarrollador, dando la posibilidad de que la elección sea objetiva y dependa solo del desempeño del desarrollador. ARDe lleva a cabo una heurística para determinar bajo qué criterio se realizó la elección de un desarrollador, y en las consultas futuras la lista se presentará ordenada por ese criterio.

3.1. Descripción de los componentes de ARDe

ARDe consta de tres etapas. La primera de ellas es la etapa de *Planificación*, que incluye la obtención y la manipulación de los datos de entrada, el mecanismo de CBR y la recomendación de desarrolladores. La etapa *Ejecución* comprende la realización de la tarea por parte de los desarrolladores seleccionados para llevarla a cabo, así como también el uso de las herramientas del entorno de desarrollo por parte de los mismos. La etapa de *Resultados* incluye la obtención de métricas reales y la comparación de las mismas con las estimaciones provistas por ARDe. Dichos resultados son utilizados dentro del mecanismo de CBR de la primera etapa, retroalimentando el enfoque.

En la implementación de ARDe se tomó como referencia las siguientes herramientas SVN[1], JIRA[2], Jenkins [3]y SonarQube[4]. SVN es una herramienta de control de versiones open source basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de archivos. JIRA es una aplicación para el seguimiento de errores, de incidencias y para la gestión operativa de proyectos. Jenkins es una herramienta de integración continua que trabaja con herramientas de control de versiones como SVN o Git[5]. Mientras que SonarQube es una plataforma para evaluar código fuente utilizando diversas herramientas de análisis estático de código fuente como Checkstyle[6], PMD[7] o FindBugs[8] para obtener métricas que pueden ayudar a mejorar la calidad del código.

En la Figura 2 se puede observar un esquema conceptual de ARDe que involucra las tres etapas antes mencionadas. En la primera etapa, a medida que los desarrolladores utilizan las herramientas dentro de los entornos de desarrollo, como por ejemplo, JIRA, Git y SonarQube, se generan datos que son capturados por la herramienta Tesys (Lezcano & Kiehr, 2016). Tesys es capaz de obtener estimaciones de métricas de cada desarrollador y generar gráficos para cada uno de ellos en base a las mismas. La información procesada por Tesys persiste en una base de datos que contiene el histórico de las tareas y las métricas pertenecientes a cada desarrollador. Dicha información será utilizada por el mecanismo del recomendador basado en CBR de la Figura 2.

El líder de proyecto, selecciona una tarea del conjunto de tareas no ejecutadas dentro del proyecto hasta el momento, para que sea asignada a un desarrollador. La tarea seleccionada junto con otro conjunto de datos, conforman los datos de entrada para el recomendador. El proceso de CBR utiliza, entre otras cosas, la tarea de entrada, la base de datos y las estimaciones de la herramienta Tesys. Dicha información es necesaria para la construcción y manipulación de la base de casos y para devolver el conjunto de desarrolladores recomendados para la tarea elegida.

A continuación, el líder de proyecto selecciona un desarrollador del conjunto resultante. Tanto el desarrollador elegido como los desarrolladores recomendados son almacenados en la base de casos, y esta información es utilizada para recomendaciones futuras. Finalmente, se llevarán a cabo las etapas de Ejecución y Resultados. Una vez que la tarea haya sido ejecutada por el desarrollador asignado para llevarla a cabo, se podrán ingresar resultados reales asociados a la ejecución de la misma. Con esta información se evalúa si el asistente ofreció una buena o una mala recomendación. La información disponible es almacenada en la base de casos construida en la etapa de planificación, ya que será útil para el mecanismo del recomendador basado en CBR.

[1] https://subversion.apache.org/

[2] https://www.atlassian.com/es/software/jira

[3] https://jenkins.io/

[4] https://www.sonarqube.org/

[5] https://git-scm.com/

[6] https://checkstyle.sourceforge.io/

[7] https://pmd.github.io/

[8] http://findbugs.sourceforge.net/

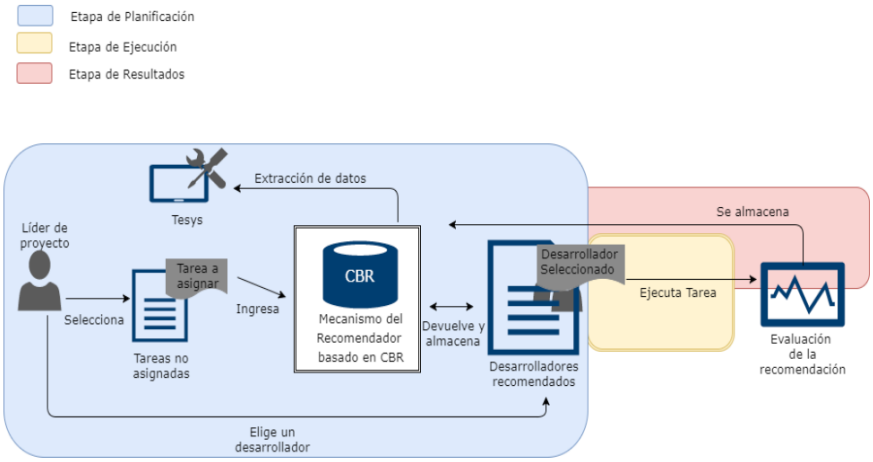

Figura 2 – Esquema conceptual del asistente para la recomendación de desarrolladores

3.2. Ejemplo motivador

Para facilitar el entendimiento del funcionamiento de ARDe, se presenta un caso de estudio ficticio en el cual una organización lleva adelante un proyecto de software llamado “Nirvana”. El equipo de desarrollo está compuesto por un líder de proyecto y sus desarrolladores. El equipo trabaja sobre un entorno de desarrollo compuesto por tres herramientas: JIRA, Git y SonarQube. En dicho proyecto el líder, desea conocer cuáles son los desarrolladores más aptos para llevar a cabo la tarea *NIR-24*. Por este motivo es que utilizará la herramienta ARDe, la cual le recomendará un conjunto de desarrolladores de su equipo con métricas estimadas para cada uno de ellos.

En la Figura 3, se puede observar un ejemplo de cuáles son los componentes principales que va a presentar la estructura del caso (según CBR) asociado a la tarea NIR-24. El caso va a estar conformado por tres partes principales: el problema, la solución y el resultado. El problema estará compuesto por la tarea a ser asignada, que a su vez va a almacenar las etiquetas que la describen, las habilidades que debe tener el desarrollador para realizarla y un conjunto de valores de métricas que el líder del proyecto desea que tenga la tarea.

La solución estará formada por los desarrolladores recomendados para realizar la tarea, por el desarrollador elegido por el líder del proyecto para llevarla a cabo y por un criterio de orden, bajo el cual se supone que fue elegido el desarrollador seleccionado por el líder. Los desarrolladores recomendados tendrán asociadas estimaciones de métricas para la nueva tarea.

El resultado estará formado por una calificación de la recomendación que va a indicar si la misma fue una buena o una mala recomendación. Este resultado va a estar basado en lo que resulte de la comparación entre los valores de las métricas estimadas y los valores de las métricas reales que van a ser obtenidas luego que la tarea haya sido ejecutada por el desarrollador elegido. Los valores de las métricas reales van a ser almacenadas dentro de la tarea asociada al desarrollador seleccionado por el líder del proyecto.

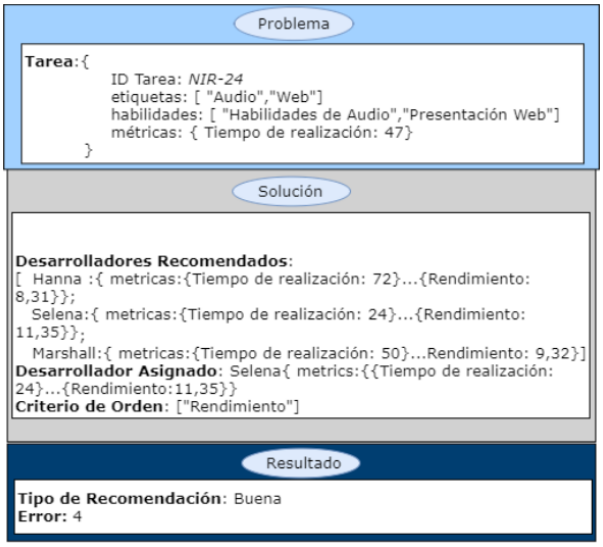

Figura 3 – Caso definido para la tarea NIR-24

3.3. Ciclo de CBR en ARDe

Los componentes detallados en la subsección anterior se van a ir obteniendo a lo largo del proceso de CBR que va a involucrar la realización de las siguientes etapas: recuperar, adaptar, revisar y almacenar. La Figura 4, proporciona un esquema general de la instanciación del ciclo de CBR para la recomendación de desarrolladores. Como se puede observar, la etapa adaptar va a tomar el nombre de recomendar, ya que dicha definición se ajusta mejor a nuestro enfoque. En primer lugar, se presentará al líder del proyecto un listado de tareas identificadas por su nombre, que aún no han sido asignadas, para que pueda seleccionar una de ellas. También aquí puede ingresar el conjunto de valores estimados para ciertas métricas que considere para la tarea y opcionalmente seleccionar aquellas habilidades que se espera que presenten los desarrolladores recomendados. Para el caso de estudio presentado, el líder de proyecto va a seleccionar la tarea NIR-24 del listado de las tareas aun no asignadas y va a ingresar un valor que desea para la métrica de tiempo de realización. También va a seleccionar del conjunto de habilidades a *Habilidades de Audio* y a *Presentación Web*.

Por cada tarea seleccionada distinta, se va a generar un Caso Nuevo. Este caso va a almacenar la tarea no asignada que haya sido seleccionada. Dicha tarea va a contener el conjunto de etiquetas, las habilidades seleccionadas para los desarrolladores y los valores de métricas que líder de proyecto ingresó. Esta información será proporcionada a la etapa Recuperar.

Esta etapa consiste en la recuperación de casos o tareas similares en la base de casos y en la base de datos respectivamente. En ambas situaciones el mecanismo se basa en la búsqueda de tareas o casos que representen a una tarea, que contengan etiquetas y habilidades similares a la nueva tarea a asignar. Las etiquetas, comprenden a las palabras

claves que aparecen en la descripción de una tarea, mientras que las habilidades, corresponden a las capacidades y aptitudes en el área de software que poseen los desarrolladores.

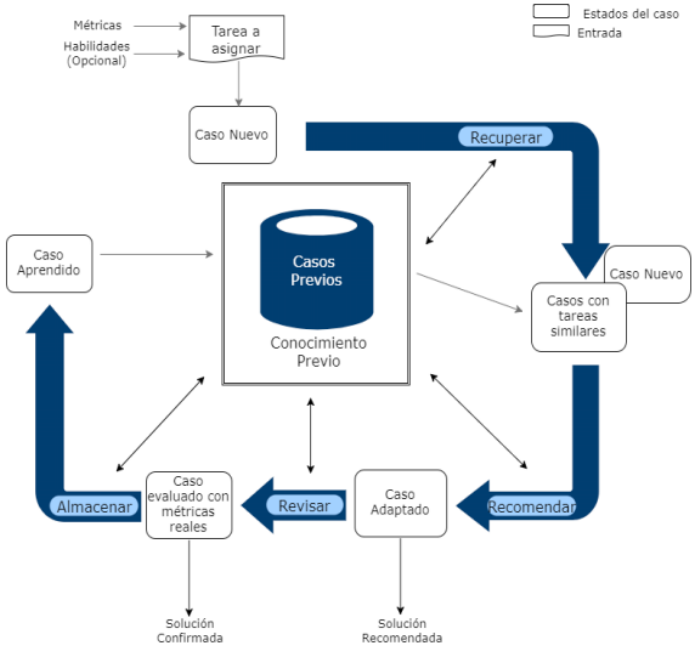

Figura 4 – Ciclo de Razonamiento Basado en Casos para la recomendación de desarrolladores

	NIR-10	*NIR-14*	*NIR-22*
Etiquetas de la tarea	*audio*	*imagen*	*video, web*
Nombre de usuario	*Hanna*	*Tom*	*Selena*
Tiempo de realización	*65h*	*23 hs*	*42 hs*
Número de líneas	*739*	*253*	*456*
Calidad del código	*63,5 %*	*12,9%*	*59,8%*

Figura 5 – Tareas almacenadas en la base de datos

Una tarea será similar a la tarea aún no asignada, si tienen en común un cierto número de etiquetas entre sus descripciones y/o si sus habilidades coinciden con un cierto número de las habilidades ingresadas como deseadas para la tarea. En nuestro ejemplo en la

tarea NIR-24 las etiquetas son web y audio. El histórico de tareas de la organización almacenada hasta el momento en la base de datos se puede observar en la Figura 5. El algoritmo devolverá como tareas similares a NIR-10 y NIR-22, dado que al menos una de sus etiquetas coincide con las etiquetas de NIR-24.

Para buscar casos similares al nuevo caso, solo bastará con llevar a cabo el algoritmo de búsqueda de tareas similares mencionado con anterioridad, dado que cada caso tiene definido una tarea específica. La base de casos de ARDe hasta el momento tiene almacenados los casos de la tabla de la Figura 6. Por lo tanto, el algoritmo también devolverá como caso similar de la tarea NIR-24 al caso CASE1 dado que la tarea definida en el mismo también tiene como etiqueta una de las etiquetas de la tarea a asignar. Por lo tanto los desarrolladores asociados a las tareas similares van a ser Hanna, Selena y Marshall.

La siguiente etapa (Recomendar) consiste en presentar al líder del proyecto, una lista con los nombres de desarrolladores que se adaptan a la tarea a asignar, junto a un conjunto de métricas estimadas sobre cada desarrollador para la tarea dada. Para ello, se van a utilizar las tareas similares y/o casos similares resultantes de la primera etapa.

En el caso de estudio las métricas estimadas para la tarea NIR-24 para cada uno de los desarrolladores similares se muestran en la Figura 7. Para el cálculo de las mismas el algoritmo utilizó el valor de la métrica tiempo de realización que ingresó el líder del proyecto en el asistente. Si el líder de proyecto hubiese ingresado valores de otras métricas que deseaba para la tarea, se tendría que haber usado el indicador de similitud mencionado anteriormente, para obtener el mejor conjunto de métricas estimadas para cada desarrollador similar.

ID Caso	*CASE 1*	*CASE 2*
ID Tarea	*NIR-8*	*NIR-19*
Etiquetas de la tarea	*web*	*texto*
Nombre de usuario	*Marshall*	*Lebron*
Tiempo de realización	*45 hs*	*19 hs*
Número de líneas	*578*	*335*
Calidad del código	*78,4%*	*90,2%*
Rendimiento	*9,85*	
Tipo de Recomendación	*Buena*	-
Error	*0,3*	-
Criterio de Orden de mayor peso	*Rendimiento*	-

Figura 6 – Casos almacenados en la base de casos

En el caso que no existan tareas similares en la base de datos y solo existan casos similares obtenidos de la base de casos, se devolverá una lista con los desarrolladores de los casos similares. Sí ocurre que solo existen tareas similares en la base de datos y no hay casos similares en la base de casos, sólo se devolverán los desarrolladores de aquellas tareas similares a la tarea a asignar.

Si hay casos similares en la base de casos y tareas similares en la base de datos, se deben adaptar los desarrolladores del nuevo caso con los desarrolladores recomendados por ambas bases. Si no hay casos similares, ni tareas similares en la base de datos, entonces el asistente devolverá una lista de desarrolladores vacía. Luego de esto, se le agrega al nuevo caso información sobre los desarrolladores recomendados. Cada desarrollador recomendado, tendrá asociado la nueva tarea a asignar y el mejor conjunto de métricas estimadas para esa tarea.

Desarrolladores recomendados	*Hanna*	*Selena*	*Marshall*
Tiempo de realización	*72 hs*	*24 hs*	*50 hs*
Número de líneas	*889*	*570*	*654*
Calidad del código	*68,3%*	*49,8%*	*72,7%*
Esfuerzo Real	*607,187*	*283,86*	*475,478*
Rendimiento	*8,31*	*11,35*	*9,32*

Figura 7 – Métricas estimadas para cada desarrollador similar para la tarea NIR-24

Luego de esto, se le agrega al nuevo caso información sobre los desarrolladores recomendados. Cada desarrollador recomendado, tendrá asociado la nueva tarea a asignar y el mejor conjunto de métricas estimadas para esa tarea (Figura 7).

Para el caso de estudio la heurística determinará que la lista de desarrolladores presentada será ordenada por la métrica Rendimiento, como se puede observar en la Figura 8. Esto se debe a que el caso similar CASE-1 presenta como criterio de orden a dicha métrica. El líder de proyecto podrá seleccionar a algunos de los tres desarrolladores recomendados. En este caso seleccionará a Selena para resolver la tarea NIR-24.

Desarrollador	**Tiempo de realización**	**Número de líneas**	**Calidad del código**	**Esfuerzo Real**	**Rendimiento**
Selena	*24 hs*	*570*	*49,8%*	*283,86*	*11,35*
Marshall	*50 hs*	*654*	*72,7%*	*475,478*	*9,32*
Hanna	*72 hs*	*889*	*68,3%*	*607,187*	*8,31*

Figura 8 – Métricas estimadas para cada desarrollador similar

Una vez que se obtenga esta información, se procede a guardar en dicho caso, el desarrollador elegido y el criterio por el cual se eligió al mismo.

Finalmente, en la etapa de Revisar y Retener, la tarea se va a ejecutar, y se podrán obtener resultados reales acerca de los valores de las métricas que se habían estimado. Se van a comparar los valores de las métricas reales con los valores de las métricas estimadas almacenadas en el caso, y se evaluará si la solución dio buenos resultados. Si es así, el caso se tomará como aceptable y llevará una marca que indique que es una buena recomendación, y si no, se indicará lo contrario. Esto será útil para no tener en cuenta esa combinación en casos futuros.

4. Evaluación y resultados

El experimento se realizó sobre proyectos reales de la carrera de Ingeniería de Sistemas de la Facultad de Ciencias Exactas (UNICEN 2014 y 2015). El presente trabajo evaluó 2 cohortes formadas por aproximadamente 60 estudiantes avanzados de la carrera de Ingeniería de Sistemas. En cada cohorte se simuló una organización de desarrollo de software con el objetivo de incrementar y mantener un proyecto de software real, mediante la implementación de prácticas de software previamente enseñadas (Rodríguez, 2016).

4.1. Diseño de casos de estudio

El objetivo del primer caso de estudio es demostrar que cada desarrollador que haya ejecutado una determinada tarea presente la mínima diferencia entre el resultado que había dado su ejecución y el resultado estimado por ARDe. Además, el resto de los desarrolladores que se encuentren en la lista de recomendados deben poseer mayores diferencias entre los valores de sus métricas estimadas y las reales obtenidas de la ejecución de dicha tarea.

Para este caso de estudio, se utiliza el error cuadrático medio (ECM), que mide el promedio de los errores al cuadrado, es decir, la diferencia entre lo estimado y lo real. La fórmula del ECM se define como:

$$\mathrm{ECM} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

Donde n es la cantidad de métricas, $\hat{Y}_i$ son los valores de métricas estimadas e Y_i son los valores de métricas reales. Además, para obtener las métricas estimadas de cada desarrollador se utilizó el cálculo de estimaciones provisto por la herramienta Tesys, la cual utiliza el coeficiente de correlación de Pearson para obtener dichas métricas. El cálculo de correlación de Pearson se define como:

$$\rho_{X,Y} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_X)]}{\sigma_X \sigma_Y}$$

Donde, σ_{XY} es la covarianza de X e Y, σ_X es la desviación típica de la variable X y σ_Y es la desviación típica de la variable Y.

El objetivo del segundo caso de estudio es demostrar que a medida que se van realizando recomendaciones de tareas similares, la herramienta va a aprendiendo en qué orden

recomendar a los desarrolladores teniendo en cuenta las selecciones de los líderes de proyecto en recomendaciones anteriores.

Para medir este caso de estudio se utiliza el coeficiente de correlación de ranking de Spearman. El coeficiente de correlación de Spearman, ρ (rho) es una medida de la correlación (la asociación o interdependencia) entre dos variables aleatorias continuas. Para calcular ρ, los datos son ordenados y reemplazados por su respectivo orden. La fórmula viene dada por la expresión:

$$\rho = 1 - \frac{6\sum D^2}{N(N^2 - 1)}$$

Donde D es la diferencia entre los correspondientes estadísticos de orden de x - y, y N es el número de parejas de datos.

En cuanto al segundo caso de estudio, el objetivo es demostrar que a medida que se van realizando recomendaciones de tareas similares, la herramienta va a aprendiendo en qué orden recomendar a los desarrolladores teniendo en cuenta las selecciones de los líderes de proyecto en recomendaciones anteriores.

4.2 Resultados obtenidos

Para el primer caso de estudio, cuando se solicita la recomendación se genera una lista de desarrolladores recomendados para la tarea seleccionada. En este caso, se seleccionó la tarea POC-66, la cual retorna un total de doce desarrolladores recomendados como se puede observar en la Figura 9. La lista está ordenada según el criterio de orden obtenido por una recomendación anterior similar (TOM-34). El criterio de orden de mayor a menor importancia está determinado por *reopened issues*, *function complexity*, *duplicated lines density* y *duplicated files*.

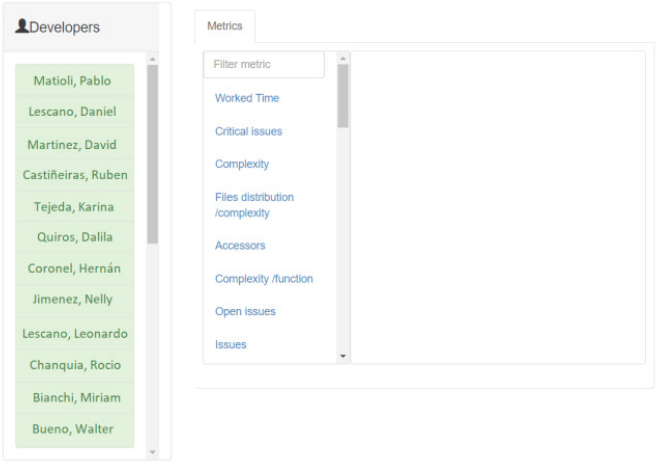

Figura 9 – Desarrolladores recomendados para la tarea POC-66

Se calcula el ECM para 4 situaciones específicas. Con relación a la fórmula, lo estimado está representado por las métricas estimadas de cada desarrollador para la tarea POC-66, mientras que lo real está representado por las métricas reales correspondientes a la ejecución de la tarea por parte de H. Coronel.

Desarrollador	Ranking
Matioli	1
Quiros	6
Coronel	7
Jiménez	8

Figura 10 – Ranking de desarrolladores

Se observa en la Figura 10 las posiciones de los desarrolladores analizados que dio la recomendación de la tarea POC-66, basada en el criterio de orden obtenido. En la Figura 11 se puede observar el cálculo del ECM entre las métricas reales obtenidas de la ejecución de la tarea POC-66 realizada por Coronel y las métricas estimadas para D. Quiros, N. Jiménez, H. Coronel y P. Matioli. Como se esperaba, se puede observar que el ECM entre las métricas estimadas y reales de Coronel es el mínimo de los 4 presentados.

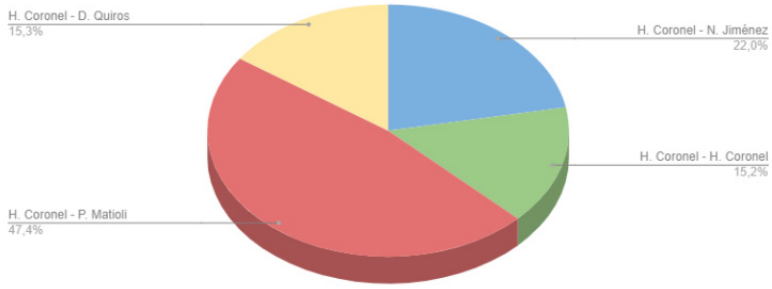

Figura 11 – Errores Cuadráticos Medios

Estos resultados refuerzan la idea de que debido a que las estimaciones de las métricas se calculan en base a experiencias anteriores, la realización de la tarea por parte del mismo desarrollador devuelve valores de métricas cercanos a los obtenidos anteriormente. Por otro lado, el mayor ECM es el obtenido entre las métricas estimadas por P. Matioli y las reales de H. Coronel. Esto se debe a que los desarrolladores están ordenados por valor de métricas, y dichos valores difieren a medida que se alejan de las posiciones cercanas a H. Coronel dentro del ranking de la recomendación. Es decir la estimación realizada para P.Matioli fue muy diferente a la de H.Coronel; por este motivo, si la estimación es próxima a la real es lógico pensar que se aleje de la estimación real u obtenida por H.Coronel. De esta forma, el ECM aumenta a medida que se aleja del desarrollador que ejecutó realmente la tarea.

En cuanto al segundo caso de estudio, para comprobar que ARDe aprende a ordenar los desarrolladores a medida que avanzan las distintas recomendaciones de tareas

similares, primero se observa cómo van variando los rankings de los desarrolladores desde la primera tarea hasta la última que se solicitó recomendar. Para esto se analizan las posiciones de los desarrolladores a lo largo de las recomendaciones de las distintas tareas. En cada análisis se representan los rankings de los desarrolladores de una tarea y de su consecutiva, siguiendo el orden de ejecución establecido. Como es de esperar, al principio ARDe no tiene información de contexto y le resulta difícil ordenar a los desarrolladores recomendados. Sin embargo, la Figura 12 muestra que al ir eligiendo los desarrolladores ARDe va cambiando las posiciones hasta que se consigue emular lo que elige el líder del proyecto. Esto quiere decir que en este punto (Tom-38) ARDe comienza a aprender un criterio de orden y ordena a los desarrolladores recomendados en base a ese criterio.

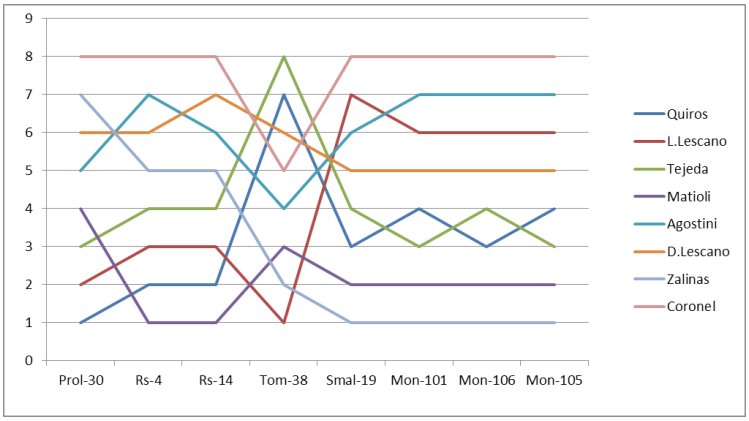

Figura 12 – Evolución del ranking de los desarrolladores a medida que se realizan las tareas (R1 a R8)

En la Figura 13 se pueden visualizar cuáles fueron las posiciones en el ranking para cada uno de los desarrolladores a lo largo de las recomendaciones de las distintas tareas (R1 a R8).

Tareas / Desarrolladores	PROL-30 (R1)	RS-4 (R2)	RS-14 (R3)	TOM-38 (R4)	SMAL-19 (R5)	MON-101 (R6)	MON-106 (R7)	MON-105 (R8)
D. Quiros	1	2	2	7	3	4	3	4
L. Lescano	2	3	3	1	7	6	6	6
K. Tejeda	3	4	4	8	4	3	4	3
P. Matioli	4	1	1	3	2	2	2	2
D. Agostini	5	7	6	4	6	7	7	7
D. Lescano	6	6	7	6	5	5	5	5
A. Zalinas	7	5	5	2	1	1	1	1
H. Coronel	8	8	8	5	8	8	8	8

Figura 13 – Ranking de los desarrolladores para tareas similares

Las celdas con color verde indican que ese desarrollador fue seleccionado para llevar a cabo la tarea indicada en la columna. Como se puede observar a lo largo de todas las

recomendaciones, A. Zalinas está en la posición 7, luego en las recomendaciones R2 y R3 el desarrollador se ubica en la posición 5, en la recomendación R4 se ubica en la posición 2 y finalmente en la recomendación R5 se posiciona en el primer puesto. En las recomendaciones siguientes mantiene la posición. Es decir, ARDe aprende por qué criterio ordenar y las métricas por las cuales va ordenando a medida que avanza cada recomendación tienen cada vez mayor correlación con las métrica utilizadas.

5. Conclusiones y trabajos futuros

En este artículo se presentó ARDe, un enfoque de CBR para asistir en la planificación de proyectos a la hora de elegir a un desarrollador para llevar a cabo una tarea. Luego de evaluar ARDe con 2 casos de estudio, se comprobó que las estimaciones se asemejan a las métricas reales obtenidas de la ejecución. Con lo cual se puede esperar que al seleccionar a un desarrollador de la lista para ejecutar una tarea, se obtengan de la misma, métricas similares a las estimadas por la herramienta. Por otro lado, se comprobó que la herramienta aprende a ordenar las recomendaciones según las preferencias del líder del proyecto para seleccionar a un desarrollador, permitiendo facilitar la búsqueda del desarrollador mejor calificado para llevar adelante la tarea.

Como principal limitación de ARDe, se necesita de un soporte de otra herramienta que centralice métricas, en este caso Tesys, la cual tenga la capacidad de obtener y almacenar las diferentes métricas obtenidas de las herramientas utilizadas en los proyectos de software. Como trabajo futuro se podría mejorar ARDe, dándole la capacidad de obtener y almacenar las métricas, sin tener que depender de otra herramienta. Finalmente, proponemos probar ARDe sobre un entorno de desarrollo real y competitivo, en el cual exista un líder de proyecto responsable de la asignación de las tareas.

Referencias

Carignano, M. C., Gonnet, S., & Leone, H. (2019). KE-SER: Un sistema basado en el conocimiento y la experiencia para dar soporte a arquitectos de software en aspectos de seguridad. *RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação, (32)*, 97–112. DOI: 10.17013/risti.32.97–112

Cordero Morales, D., Ruiz Constanten, Y., & Torres Rubio, Y. (2013). Sistema de Razonamiento Basado en Casos para la identificación de riesgos de software. *Revista Cubana de Ciencias Informáticas, 7(2)*, 222–239.

Fan, Z. P., Li, Y. H., & Zhang, Y. (2015). Generating project risk response strategies based on CBR: A case study. *Expert Systems with Applications, 42(6)*, 2870–2883. https://doi.org/10.1016/j.eswa.2014.11.034

Gido, J., & Clements, J. (1999). *Successful project management*. Cincinnati: SouthWestern College Pub.

Ji, S. H., Park, M., & Lee, H. S. (2011). Cost estimation model for building projects using case-based reasoning. *Canadian Journal of Civil Engineering, 38(5)*, 570–581. https://doi.org/10.1139/l11-016

Jin, R., Cho, K., Hyun, C., & Son, M. (2012). MRA-based revised CBR model for cost prediction in the early stage of construction projects. *Expert Systems with Applications, 39(5)*, 5214–5222. https://doi.org/10.1016/j.eswa.2011.11.018

Kim, S. (2013). Hybrid forecasting system based on case-based reasoning and analytic hierarchy process for cost estimation. *Journal of Civil Engineering and Management, 19(1)*, 86–96. https://doi.org/10.3846/13923730.2012.737829

Koo, C., Hong, T., Hyun, C., & Koo, K. (2010). A CBR-based hybrid model for predicting a construction duration and cost based on project characteristics in multi-family housing projects. *Canadian Journal of Civil Engineering, 37(5)*, 739–752. https://doi.org/10.1139/L10-007

Lezcano, L., & Kiehr, A. (2016). Asistente para la planificación inteligente de proyectos. *RiDAA*- Universidad Nacional del Centro de la Provincia de Buenos Aires , retrieved from: https://www.ridaa.unicen.edu.ar/xmlui/handle/123456789/784

Morales-Aguiar, N., & Vega-Zepeda, V. (2018). Factores Humanos y la Mejora de Procesos de Software: Propuesta inicial de un catálogo que guíe su gestión. *RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação*, (29), 30–42. http://dx.doi.org/10.17013/risti.29.30-42

Rodríguez, G., Soria, Á., & Campo, M. (2016). Measuring the impact of agile coaching on students' performance. *IEEE Transactions on Education, 59(3)*, 202–209. DOI: 10.1109/TE.2015.2506624

Stamelos, I. (2010). Software project management anti-patterns. *Journal of Systems and Software, 83(1)*, 52–59. https://doi.org/10.1016/j.jss.2009.09.016

Wu, D., Li, J., & Bao, C. (2018). Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation. *Soft Computing, 22(16)*, 5299–5310. https://doi.org/10.1007/s00500-017-2985-9