Revista Cubana de Ciencias Informáticas
ISSN: 2227-1899 | RNPS: 2301
Vol. 8, No. 4, Octubre-Diciembre, 2014
Págs. 100-115
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Inteligencia artificial
Recibido: 23/06/2014 | Aceptado: 25/08/2014

Método para la construcción del modelo de dominio en un Tutor Inteligente de Programación

Method for the construction of the domain model in an Intelligent Programming Tutor

Enrique José Altuna Castillo [1*], Lisandra Guibert Estrada [1], Vivian Estrada Sentí [2]

[1] FORTES. Centro de Tecnologías para la Formación. Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños km 2 ½, Reparto Torrens, Boyeros, La Habana, Cuba. C.P.: 19370
[2] Centro de Postgrado Internacional. Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños km 2 ½, Reparto Torrens, Boyeros, La Habana, Cuba. C.P.: 19370

[*] Autor para correspondencia: ejaltuna@uci.cu

Resumen

La generación de retroalimentación es un elemento fundamental para garantizar las ganancias de aprendizaje que se obtienen del uso de Sistemas Tutores Inteligentes. En la enseñanza de la programación los métodos tradicionales necesitan de un exhaustivo análisis de las soluciones correctas e incorrectas, produciendo tiempos de autoría de las actividades entre 200 y 300 horas. Las alternativas reducen el tiempo a cambio de una disminución en la efectividad de la retroalimentación. El objetivo del presente trabajo es desarrollar un método para la construcción del modelo de dominio para un Sistema Tutor Inteligente de programación que contribuya a la reducción del tiempo de autoría de forma que se genere retroalimentación efectiva para las actividades de ejercitación de tipo implementación. Se realiza la exposición de la propuesta de solución y se realiza un estudio para comprobar que esta cumple con las características deseadas.

Palabras clave: autoría, efectividad, programación, retroalimentación, Sistema Tutor Inteligente.

Abstract

The feedback generation is an essential element to ensure the learning gains that are obtained from the application of Intelligent Tutoring Systems. In programming teaching traditional methods require a thorough analysis of correct

Revista Cubana de Ciencias Informáticas
ISSN: 2227-1899 | RNPS: 2301
Vol. 8, No. 4, Octubre-Diciembre, 2014
Págs. 100-115
http://rcci.uci.cu

and incorrect solutions, producing authoring time for activities between 200 and 300 hours. The alternatives reduce the time in return of a decrease in the feedback´s effectiveness. The aim of this work is to develop a method for the construction of the domain model for an Intelligent Tutoring System for programming that contributes to reduce authoring time and the generation of effective feedback for activities of implementation type. The proposed solution is presented and a study is conducted to check that it meets the expected characteristics.
Keywords: *authoring, effectiveness, feedback, Intelligent Tutoring System, programming.*

Introducción

La enseñanza de la programación presenta deficiencias que se reflejan en los bajos índices de retención (Hakimzadeh, Adaikkalavan *et al.*, 2011, Junco Vázquez, 2012). Trabajos como (Jurado, 2010) proponen como alternativa a esta situación complementar los conocimientos teóricos con actividades prácticas. En estas es importarte la cantidad de apoyo que recibe el estudiante durante su realización (Watson, Li *et al.*, 2011), elemento que está normalmente acotado por la relación de estudiantes por profesor.

Los Sistemas Tutores Inteligentes (STI) surgen como una evolución de la enseñanza asistida por computadoras, al combinar conocimientos de pedagogía, psicología, ciencias cognitivas con avances en la ciencia de la computación, en especial de Inteligencia Artificial (IA). Se usan diversas técnicas de IA en la generación de rutas de aprendizaje, selección de actividades, soporte durante las actividades, evaluación, presentación de los contenidos, entre otras (Woolf, 2009). Tienen como meta reproducir el hacer de un profesor humano competente, que adapta técnicas para el aprendizaje de un dominio de enseñanza en función del perfil del estudiante (Conati, 2009).

Como resultado del uso de STI se reportan ganancias significativas en el aprendizaje con respecto a otros tipos de sistemas que se usan para apoyar el proceso de enseñanza-aprendizaje (Brawner, Holden *et al.*, 2011). Autores como (Aleven, 2010, Eagle y Barnes, 2013) destacan como causa principal de estos resultados positivos la retroalimentación que recibe el estudiante durante la realización de actividades. Incluso se apunta que existe una relación directa entre la cantidad de retroalimentación y estas ganancias (Chi, VanLehn *et al.*, 2010).

Para realizar la tutoría adaptativa se usan componentes que, aunque varían de un sistema a otro, responden a la representación de tipos de conocimiento similares: conocimiento sobre el dominio o materia objeto de estudio, para saber qué enseñar; conocimiento sobre la persona que aprende, para saber a las características de quien adaptar; conocimiento sobre el proceso de tutoría, para saber cómo enseñar y conocimiento sobre cómo debe ser la interacción entre sistema y usuario. Estos son conocidos como modelos del dominio, del estudiante, del profesor y de interacción

respectivamente. El término modelo es usado en este contexto por los investigadores para conceptualizar y operacionalizar las funciones y variables que intervienen en la tutoría (Bourdeau y Grandbastien, 2010).

El modelo del dominio contiene las formas de representación y razonamiento que le permiten funcionar como una fuente de conocimiento y un estándar para evaluar las acciones del estudiante dentro de la temática que se aprende. La forma para enfocar el desarrollo de un modelo de dominio está directamente relacionada con la estructura de sus conocimientos y la idoneidad para soportar los procesos de tutoría.

En la búsqueda realizada se encontraron múltiples ejemplos de STI que han sido desarrollados y aplicados con éxito para la programación, conocidos como Tutores Inteligentes de Programación (TIP). El interés de esta investigación se encuentra en aquellos que apoyan la realización de actividades prácticas de tipo implementación. Se estudian los enfoques que se han utilizado para generar retroalimentación, que permiten al estudiante trabajar sobre la base de los errores cometidos.

Dentro de este ámbito se encontraron trabajos que tienen como meta generar retroalimentación para el lenguaje estructurado de consultas (en inglés SQL) y lenguajes de programación lógica (Weber y Brusilovsky, 2001, Mitrovic, 2003). Los autores del presente trabajo consideran que en ambos casos se trabaja sobre condiciones más simples como consecuencia del reducido número de construcciones del lenguaje. Además, un lenguaje de programación lógica puede ser entendido como el cálculo relacional enriquecido con recursión y símbolos funcionales (Le, 2011). Por tanto, en este artículo se analizan exclusivamente los relacionados con lenguajes de alto nivel de paradigma imperativo. Se trabaja con un enfoque similar al usado para el entrenamiento de estudiantes para competencias (Urbancic y Trampus, 2012), (Zhao, Wang *et al.*, 2013). Se enuncia el problema y se especifican formatos rígidos de entrada y salida. El estudiante deberá implementar el programa completamente, usando las librerías básicas del lenguaje de programación.

En los trabajos consultados se encontraron dos grupos de enfoques para realizar la generación de retroalimentación en este ámbito. El primero está compuesto por las técnicas tradicionalmente usadas para los dominios mejor definidos. En el sistema PHP ITS (Weragama, 2013) se usan reglas escritas en lógica de predicados de primer orden y en J-LATTE (Holland, Mitrovic *et al.*, 2009) se usan restricciones con ese propósito. Sin embargo, estos métodos requieren de la completitud del modelo desarrollado. En la programación, incluso los más simples problemas, al ser diseñada su solución suelen tener varias vías posibles, producto de diferencias en las estrategias, algoritmos y decisiones de implementación tomadas (Le, Menzel *et al.*, 2010, Le, Loll *et al.*, 2013). Se necesita de un análisis exhaustivo de las actividades, que algunos autores indican que representa un costo en tiempo de entre 200 y 300 horas por hora de instrucción (Barbhuiya, Mustafa *et al.*, 2011). La omisión de variantes correctas o incorrectas, da lugar a

conocimiento incompleto y potencialmente incorrecto, que provoca errores y falta de apoyo por parte del sistema durante su uso.

El segundo grupo está compuesto por propuestas alternativas que sacrifican aspectos relacionados con la efectividad de la retroalimentación para disminuir el tiempo de autoría. Las técnicas de caja negra (Enström, Kreitz *et al.*, 2011), comparación con soluciones ideales (Sykes y Franek, 2003, Jurado, 2010, Naudé, Greyling *et al.*, 2010) y presentación de información estática (Truong, Bancroft *et al.*, 2005, Watson, Li *et al.*, 2012) y dinámica (Orehovacki, Radosevic *et al.*, 2012) del programa han sido aplicadas con resultados exitosos. Sin embargo, al limitarse la efectividad no se obtienen las ganancias observadas en otras áreas de aplicación.

Tomando en consideración estos elementos, este artículo tiene como objetivo presentar un método para la construcción del modelo de dominio para un TIP que contribuya a la reducción del tiempo de autoría de forma que se genere retroalimentación efectiva para las actividades de ejercitación de tipo implementación.

Metodología computacional

Para la definición de la propuesta se emplearon diversos métodos científicos de investigación entre los que se encuentran el análisis y síntesis, entrevistas y encuestas. Se realizó un análisis documental relacionado con los TIP y la efectividad de la retroalimentación, así como las técnicas utilizadas. Se realizó un estudio sobre técnicas de análisis estático y dinámico de programas y formas de representación de las características de las soluciones. A continuación se describen brevemente los resultados de este estudio y las decisiones tomadas para la generación de la propuesta.

Efectividad de la retroalimentación

La retroalimentación es una de las partes esenciales de los STI (Lazar y Bratko, 2014) por lo que es necesario un estudio para alcanzar sus potencialidades en cuanto a efectividad. Sin encontrarse un criterio definitivo sobre los elementos que debe cumplir, se analizan las características identificadas como deseables.

En las actividades prácticas, donde el estudiante debe solucionar problemas aplicando conocimientos, es importante la utilidad de la retroalimentación. Información oportuna sobre los errores cometidos ayuda al estudiante a salir de estancamiento o baches en el conocimiento. Esta información resulta un factor importante para evitar el abandono como producto de la desmotivación (Dominguez, Yacef *et al.*, 2010), (Lazar y Bratko, 2014).

Un elemento de importancia en la calidad de la retroalimentación se encuentra en la capacidad del sistema del sistema para relacionar los errores cometidos por el estudiante con elementos del dominio. Esto depende en gran medida de

determinar en la evaluación conocimientos correctos aún no aprendidos, así como elementos aprendidos de forma incorrecta (VanLehn, 2011). Durante este trabajo se llamará a esta característica capacidad cognitiva.

En (Stamper, Eagle *et al.*, 2013) se presenta la especificidad de la retroalimentación como un factor determinante para su efectividad. En ese trabajo se discute sobre la relevancia de incluir información directamente relacionada con el contexto donde se encuentra el estudiante y en menor medida con aspectos generales del área de conocimiento que se trabaja.

En trabajos revisados (Beck, Chang *et al.*, 2008), (Razzaq y Heffernan, 2009), (Stamper, Eagle *et al.*, 2013), (Mitrovic, Ohlsson *et al.*, 2013) se manifiesta una falta de criterio definitivo sobre el momento en que debe ser provista la retroalimentación o si esta se produce a demanda de forma proactiva por el sistema. En los estudios realizados se aprecia un consenso acerca que la cantidad de soporte que se sea capaz de entregar puede resultar un factor decisivo indican que existe una relación directa entre el volumen de interacción que ocurre entre sistema-estudiante y la ganancia en el aprendizaje que ocurre del proceso (Chi, VanLehn *et al.*, 2010). Por este motivo, es una característica deseable la posibilidad de retroalimentación en cualquier momento que se necesite en función de una estrategia pedagógica (Woolf, 2009), (Muldner y Conati, 2012).

Aspectos generales del método propuesto

Para cumplir con los objetivos trazados se enuncian dos premisas que deben ser cumplidas por la propuesta:

1. La inclusión de nuevas actividades en el sistema no requieran intensivos procesos de autoría, donde sea necesario encontrar todos los programas correctos y los principales tipos de errores.
2. En la retroalimentación que se genera al estudiante se incluyen aspectos semánticos, relativos a elementos específicos de la actividad que se intenta solucionar.

En la figura 1 aparece una representación de alto nivel del sistema, donde se puede observar el modelo de dominio propuesto en relación con otros componentes y con los procesos fundamentales sobre los que incide la investigación.

El sistema provee al estudiante de actividades para ser resueltas en herramientas externas, normalmente en entornos integrados de desarrollo (IDE). El estudiante codifica, en un lenguaje de programación admitido por el sistema, la respuesta y la envía para la evaluación de la actividad. El código fuente de la respuesta se recibe por el tutor a través de la interfaz y se hace llegar al modelo de dominio, después de pasar por el modelo del tutor. En el modelo de dominio se genera la retroalimentación de la evaluación de la actividad que se pasa al modelo pedagógico para su adecuación y se entrega al estudiante a través de la interfaz.

Revista Cubana de Ciencias Informáticas
ISSN: 2227-1899 | RNPS: 2301
Vol. 8, No. 4, Octubre-Diciembre, 2014
Págs. 100-115
http://rcci.uci.cu

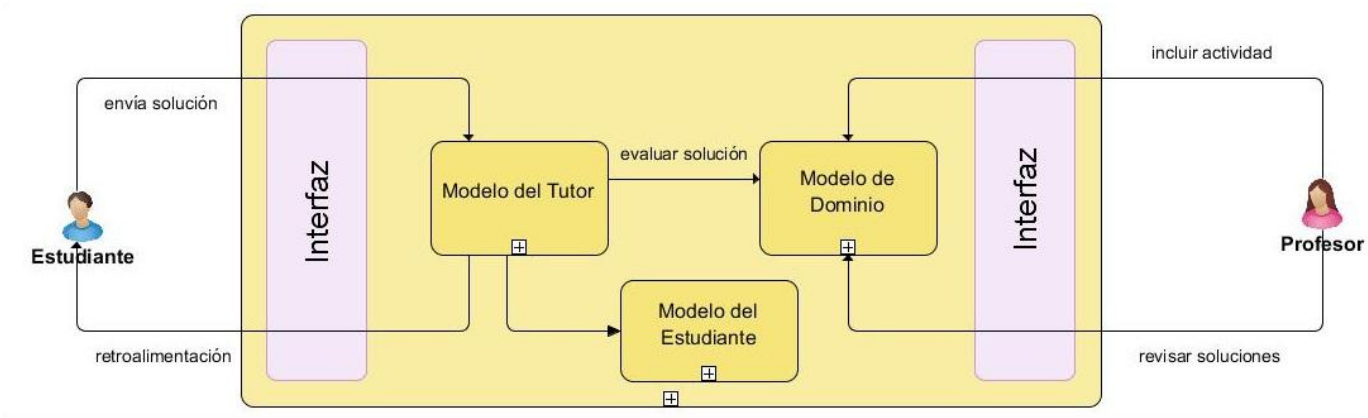

Figura 1. Procesos y relaciones entre modelos

Generación de retroalimentación

Se hace uso de técnicas de caja negra, análisis estático y análisis dinámico para la generación de retroalimentación durante las actividades prácticas de tipo implementación. El modelo toma como entrada el código fuente del estudiante que pasa por los componentes internos generando la retroalimentación en cada uno de los niveles. Se diseña siguiendo un estilo arquitectónico secuencial en lote, que se caracteriza por un número finito de etapas que se conectan de forma lineal. Este estilo favorece la extensibilidad del sistema al limitar el impacto de los cambios.

El modelo de dominio toma como entrada el nivel hasta el que se genera la retroalimentación en cada caso, que es determinado por otros componentes del sistema en función de alguna estrategia pedagógica. Se brinda al estudiante la posibilidad de corregir los errores antes de proporcionar la ayuda, permitiendo que solucione errores tipográficos u otros de similar naturaleza provocados por el descuido, lo que limita el impacto en la evaluación de evidencias que no se corresponde con el estado real de conocimiento del estudiante. Cada problema puede ser enviado un número ilimitado de oportunidades y en cada ocasión el estudiante puede mantenerse en el nivel actual de ayuda, solicitar el próximo nivel, suspender o finalizar la actividad. La retroalimentación que se entrega se divide en 6 niveles, donde para cada uno se muestran las ayudas correspondientes a este y a los anteriores:

- Nivel 0: El resultado de la evaluación automática mediante técnicas de caja negra.
- Nivel 1: Información sobre la estructura del programa que se genera con técnicas de análisis estático.
- Nivel 2: Información general sobre el comportamiento del programa en tiempo de ejecución. Se obtiene mediante técnicas de análisis dinámico.

- Nivel 3: Información específica sobre el comportamiento en tiempo de ejecución sobre situaciones sospechosas y se incluye un análisis sobre el flujo de datos para la conformación de las salidas del programa.
- Nivel 4: Los juegos de datos con los que se comprueba la solución en el sistema y se muestran las diferencias entre la salida generada por el estudiante y la esperada según lo especificado por el creador del problema.
- Nivel 5: Se muestran las soluciones propuestas y comentarios realizados por el profesor.

Para el análisis estático se recomienda delegar en herramientas externas que existen con ese propósito. Para varios de los lenguajes de programación más conocidos existen herramientas que han demostrado buenos resultados encontrando errores, tanto en ámbitos académicos como en la industria. Algunas de ellas de código abierto y licencias libres (Altuna y Guibert, 2013). A pesar que estos programas brindan flexibilidad en la configuración, los resultados que se obtienen deben ser personalizados para el ámbito educativo teniendo en cuenta la claridad de las explicaciones que se muestran. En los casos donde sea posible también se deben incluir enlaces a información adicional sobre los conceptos que se manejan. Como resultado de este análisis se provee información acerca de ciclos infinitos, errores en la conversión dinámica de tipos, secciones de código inalcanzables, problemas de rendimiento, variables no usadas; así como fomentar el uso de buenas prácticas de codificación que aumentan la calidad del código que se genera al hacerlo menos propenso a errores.

En el análisis dinámico se utiliza información sobre la ejecución del programa. Para obtener los datos se usa el proceso de instrumentación. Se modifica el programa original insertando sentencias que en tiempo de ejecución generan información sobre los eventos que ocurren. La manipulación del programa original es posible en sus diferentes formas: código fuente, representaciones intermedias o código objeto. Para los principales lenguajes de programación en función de sus características existen herramientas que facilitan la navegación y manipulación sobre las estructuras del programa. Usualmente trabajan sobre representaciones abstractas, especialmente sobre el árbol de sintaxis abstracta (en inglés AST). Se recorre la estructura y se realizan las modificaciones que adicionan las sentencias para registrar la ejecución de las sentencias originales de interés.

Debido a la importancia de incluir información semántica en la retroalimentación (Altuna y Guibert, 2013), se trabajó en una forma de introducir conocimiento en el análisis sobre el programa que se evalúa. Esta permite hacer análisis dinámico informado, al tiempo que el impacto en esfuerzo para la obtención de esta característica favorece su uso en entornos reales. El estudio realizado permitió identificar cómo, con la información sobre la estructura y tipos de datos que conforman las entradas y salidas se introduce la capacidad para analizar aspectos semánticos que anteriormente no podían ser vistos. El análisis del flujo de datos adquiere mayor potencialidad al posibilitar encontrar la correspondencia entre entradas y salidas, así como trabajar con datos más reales en las operaciones matemáticas para

Revista Cubana de Ciencias Informáticas
ISSN: 2227-1899 | RNPS: 2301
Vol. 8, No. 4, Octubre-Diciembre, 2014
Págs. 100-115
http://rcci.uci.cu

eliminar falsos positivos en los resultados. Para almacenar y razonar sobre esta información fue necesario desarrollar un lenguaje de especificación y los algoritmos que la manejan, que constituyen dos de los principales aportes de este trabajo. Apoyados en estos elementos y con el soporte de instrumentación anteriormente descrito se realiza el análisis dinámico específico.

El algoritmo es capaz de determinar la correspondencia entre las operaciones de entrada/salida realizadas por el programa y lo especificado en el patrón. También sigue el flujo de datos desde las variables que asimilan datos de entrada, pasando por variables intermedias, hasta las variables que almacenan datos que se envían a la salida. Su funcionamiento sigue el diseño propuesto por el patrón publicador subscritor. En cada caso se ejecuta la parte correspondiente al tipo de sentencia ejecutada por el programa original (invocación y fin de procedimiento, inicio y fin de bloque, sentencia de entrada de datos, operación sobre variables, sentencia de salida).

Revisión de soluciones

La revisión de las soluciones es otro de los elementos distintivos de la propuesta y contribuye a la reducción del tiempo de autoría. Posibilita incrementar el conocimiento codificado sobre las actividades a través de las interacciones de los estudiantes. Debido a esta característica es posible retrasar la inclusión de soluciones de ejemplo usadas en el nivel 5 de retroalimentación.

El profesor incluye como soluciones de ejemplo aquellas que fueron enviadas por los estudiantes y que difieren en mayor medida de las incluidas para la actividad. Por lo general contienen otros algoritmos, vías solución o diferencias de implementación significativas. Las soluciones son filtradas y se presentan al profesor las que más difieren de las incluidas, entonces el profesor certifica si son realmente significativas e incluye comentarios adicionales. El propósito es reducir el número de soluciones que el profesor debe revisar manualmente.

La parte más significativa del proceso se encuentra en la actividad filtrar soluciones. Con su resultado se posibilita que el profesor revise sólo soluciones candidatas a ser significativas y no inspeccione de forma manual el espacio de solución completo. Para presentar al profesor se seleccionan aquellas soluciones donde la suma de las diferencias con soluciones escogidas y descartadas es máxima.

Por tanto, dado un conjunto de soluciones inspeccionadas por un profesor y clasificadas como significativas que llamaremos soluciones revisadas C_r. Un conjunto de soluciones inspeccionadas por un profesor y clasificadas como muy similar a una solución significativa catalogada anteriormente que llamaremos soluciones desechadas C_d. Un conjunto de soluciones enviadas por lo estudiantes que no se encuentran en C_r o en C_d que llamaremos soluciones enviadas C_e. Es necesario resolver el problema de encontrar un conjunto de soluciones filtradas $C_f \subset C_e$ tal que $\forall f \in C_f$,

Revista Cubana de Ciencias Informáticas
ISSN: 2227-1899 | RNPS: 2301
Vol. 8, No. 4, Octubre-Diciembre, 2014
Págs. 100-115
http://rcci.uci.cu

$\forall e \in C_e \cap C_f : p(f) > p(e)$. Donde p es una función de puntuación que determina el cumplimiento de las características deseadas. Por lo que para la propuesta actual se define como:

$$p(i) = \sum_{d=1}^{|C_d|} 1 - s(i,d) + \sum_{r=1}^{|C_r|} 1 - s(i,r) \quad (1)$$

En la ecuación 1 ***s*** es una medida de similitud entre dos soluciones. Se usa como medida de similitud la variación de *neighbor matching* propuesta por (Vujošević-Janičić, Nikolić *et al.*, 2013). Este método usa la representación del programa conocida como grafo de flujo de control. Define que dos nodos ***i*** y ***j*** de los grafos ***A*** y ***B*** son considerados similares si los nodos vecinos de ***i*** pueden ser correspondidos a vecinos de ***j*** similares. Entonces, se calcula la similitud los dos nodos mediante un proceso iterativo que hace uso de la siguiente función de actualización:

$$x_{ij}^{k+1} \leftarrow \sqrt{y_{ij} \cdot \frac{s_{in}^{k+1}(i,j) + s_{out}^{k+1}(i,j)}{2}} \quad (2)$$

Donde Y_{ij} es la similitud del contenido de los nodos i y j. Además:

$$s_{in}^{k+1}(i,j) \leftarrow \frac{1}{m_{in}} \sum_{l=1}^{n_{in}} x_{f_{ij}^{in}(l) g_{ij}^{in}(l)}^{k} \qquad s_{out}^{k+1}(i,j) \leftarrow \frac{1}{m_{out}} \sum_{l=1}^{n_{out}} x_{f_{ij}^{out}(l) g_{ij}^{out}(l)}^{k}$$

$$m_{in} = \max(id(i), id(j)) \qquad m_{out} = \max(od(i), od(j))$$

$$n_{in} = \min(id(i), id(j)) \qquad n_{out} = \min(od(i), od(j)) \quad (3)$$

Donde f_{ij}^{in} y g_{ij}^{in} son las funciones de enumeración de la correspondencia óptima de los vecinos de entrada (si un grafo contiene una arista de ***i*** a ***j***, el nodo ***i*** es llamado vecino de entrada del nodo ***j***) para los nodos ***i*** y ***j***, de forma análoga para f_{ij}^{out} y g_{ij}^{out}. Los términos ***id(i)*** y ***od(i)*** se refieren respectivamente a los grados de entrada y salida del nodo i. Encontrar la correspondencia de los elementos de ***A*** y ***B*** de mayor peso es un problema de asignación, que puede ser resuelto a través del algoritmo húngaro, aunque existen otros más eficientes. La matriz de similitud $[x_{ij}]$ refleja la similitud de los nodos de dos grafos ***A*** y ***B***. La similitud de los grafos se define como el peso de la correspondencia óptima dividido por el número de pares en la correspondencia:

$$s(G_A, G_B) = \frac{1}{n} \sum_{l=1}^{n} x_{f(l)g(l)} \quad (4)$$

Revista Cubana de Ciencias Informáticas
ISSN: 2227-1899 | RNPS: 2301
Vol. 8, No. 4, Octubre-Diciembre, 2014
Págs. 100-115
http://rcci.uci.cu

Resultados y discusión

Se realizó la implementación del modelo de dominio con el método descrito como un componente de Instructor: un Tutor Inteligente de Programación desarrollado en la Universidad de las Ciencias Informáticas. El desarrollo se realizó en forma de aplicación web sobre la plataforma Java Edición Empresa (Java EE) y Java como lenguaje de programación.

Para evaluar la propuesta se aplicaron dos métodos científicos, un cuasi experimento para medir el tiempo de autoría y una encuesta a expertos para medir su percepción sobre la existencia de los indicadores de efectividad de la retroalimentación. En ambos casos se aplicaron los métodos a la propuesta y a las desarrolladas para el mismo propósito, anteriormente descritas. Se analizaron los resultados obtenidos para conocer el comportamiento de la variable tiempo de autoría con respecto a la efectividad de la retroalimentación y comprobar que se cumplió con el objetivo trazado.

Tiempo de autoría de las actividades

Se diseñó un cuasi experimento usando profesores y actividades reales con el objetivo de determinar si existen diferencias significativas entre los tiempos de autoría para cada propuesta. Se seleccionaron 12 profesores universitarios con más de 5 años de experiencia en la disciplina de programación. En ellos se encontraron representadas las cuatro categorías docentes principales de la educación superior cubana, por lo que se considera una muestra representativa. Dado que no se encontraron implementaciones provistas por los autores de las propuestas, el estudio se realizó con implementaciones del autor del presente trabajo siguiendo las indicaciones descritas en los trabajos originales.

El cuasi experimento se realizó a lo largo de un semestre y en este periodo cada participante trabajó sobre 10 actividades de similar nivel de dificultad (las mismas para todos los participantes) y las 10 propuestas, exactamente una vez. Se realizó una distribución aleatoria sobre el orden en que los individuos trabajaron sobre las propuestas. De esa forma se redujo la posible influencia del factor orden sobre las mediciones del tiempo.

Al iniciar el estudio se entregó a cada profesor una actividad para ser insertada usando una propuesta (indicada en cada caso), cuyo prototipo se encontraba disponible a través de una interfaz accesible mediante un navegador. Al completar la asignación le fue entregada la siguiente y así sucesivamente, hasta completar todas las programadas según la distribución.

Para lograr mayor uniformidad se pidió a los profesores que todo el trabajo de mesa de la autoría se realizara en formularios impresos en papel, que le fueron entregados, registrando el tiempo de inicio y de fin de cada sesión de

Revista Cubana de Ciencias Informáticas
ISSN: 2227-1899 | RNPS: 2301
Vol. 8, No. 4, Octubre-Diciembre, 2014
Págs. 100-115
http://rcci.uci.cu

trabajo realizada. Al completar esta etapa, en presencia del investigador, transcribieron la información que desarrollaron en papel hacia el sistema, tiempo que también fue registrado.

En la tabla 1 se presenta un resumen de las mediciones realizadas, presentado en horas. Para comprobar si existían diferencias significativas en la media del tiempo de autoría entre propuestas se realizó una prueba de hipótesis. Al aplicar la prueba Kolmogorov-Smirnov se desechó la hipótesis de normalidad.

Tabla 1. Resultados de la medición para la variable tiempo de autoría

Método	Mean	N	Std. Deviation
PHP-ITS (Weragama, 2013)	98.55	12	14.81
J-LATTE (Holland, Mitrovic *et al.*, 2009)	96.54	12	15.01
Enström, Kreitz *et al.* 2011	2.31	12	0.63
JITS (Sykes y Franek, 2003)	32.24	12	4.78
Naudé, Greyling *et al.* 2010	31.83	12	3.85
ELP (Truong, Bancroft *et al.*, 2005)	33.92	12	5.52
Jurado 2010	33.21	12	5.54
Verificator (Orehovacki, Radosevic *et al.*, 2012)	2.10	12	0.59
BlueFix (Watson, Li *et al.*, 2012)	1.01	12	0.22
Instructor (propuesta del autor)	3.40	12	0.80

En la tabla 2 se exponen los valores de significación asintótica de los resultados obtenidos con la aplicación por pares del test de Mann Whitney. En los casos donde la significación es menor a 0.05 se rechaza la hipótesis de igualdad de medias y es posible afirmar que existen diferencias entre las propuestas en cuestión. Como se puede observar, la propuesta del autor presenta uno de los menores tiempos de autoría. Existen otras tres para las que se obtuvieron tiempos menores. La explicación a este comportamiento se encuentra en los elementos que son necesarios definir para la inclusión de actividades. Estos elementos son los que posibilitarán las diferentes estrategias para la generación de la retroalimentación y están directamente relacionados con la efectividad.

Tabla 2. Resultados de la aplicación por pares del test de Mann Whitney

	Método	1	2	3	4	5	6	7	8	9	10
1	PHP-ITS		.713	.000	.000	.000	.000	.000	.000	.000	.000
2	J-LATTE	.713		.000	.000	.000	.000	.000	.000	.000	.000
3	Enström, Kreitz *et al.* 2011	.000	.000		.000	.000	.000	.000	.378	.000	.002
4	JITS	.000	.000	.000		.843	.514	.671	.000	.000	.000
5	Naudé, Greyling *et al.* 2010	.000	.000	.000	.843		.242	.551	.000	.000	.000
6	ELP	.000	.000	.000	.514	.242		.843	.000	.000	.000
7	Jurado 2010	.000	.000	.000	.671	.551	.843		.000	.000	.000
8	Verificator	.000	.000	.378	.000	.000	.000	.000		.000	.000
9	BlueFix	.000	.000	.000	.000	.000	.000	.000	.000		.000
10	Instructor (propuesta del autor)	.000	.000	.002	.000	.000	.000	.000	.000	.000	

Revista Cubana de Ciencias Informáticas
ISSN: 2227-1899 | RNPS: 2301
Vol. 8, No. 4, Octubre-Diciembre, 2014
Págs. 100-115
http://rcci.uci.cu

Efectividad de la retroalimentación

La comparación con las propuestas encontradas en la literatura se realizó a través de una encuesta a expertos que expresaron su criterio según la escala de Likert. Se registró su percepción sobre la existencia en las propuestas de las características utilidad, especificidad, completitud y capacidad cognitiva.

Para la encuesta se requirieron especialistas con al menos cinco años de experiencia en la enseñanza de la programación, teniendo en ese periodo relación con tecnologías para la enseñanza a distancia y la evaluación automática de actividades para esta disciplina. De 28 expertos inicialmente identificados, 7 declinaron participar y 5 fueron descartados debido a su índice de competencia, finalmente siendo procesados 16 cuestionarios.

Para conformar el cuestionario por cada propuesta se incluyeron cuatro afirmaciones, correspondientes a los indicadores de efectividad. Las afirmaciones usadas fueron:

1. La retroalimentación generada contribuye a que los estudiantes puedan encontrar la causa de sus errores cometidos en el programa.
2. En la retroalimentación se incluyen elementos específicos del problema o la implementación en cuestión y no únicamente elementos generales correspondientes al lenguaje de programación.
3. Para todos los tipos de errores cometidos se muestra retroalimentación útil al estudiante.
4. Al generarse la retroalimentación, la misma está relacionada con elementos del dominio, permitiendo que el sistema pueda identificar elementos no aprendidos.

En el procesamiento de los resultados se consideró la frecuencia de aparición para cada categoría de la escala de Likert definida para la encuesta (muy de acuerdo (MA), de acuerdo (A), ni de acuerdo ni en desacuerdo (SN), en desacuerdo (ED), completamente en desacuerdo (CD)). A partir de esos datos se calculó para las características de cada propuesta el índice porcentual (IP), para integrar en un único valor numérico el criterio de los expertos sobre su existencia. Los valores superiores a 85 indican que existe un consenso a favor de la afirmación, para los inferiores a 35 hacia el rechazo de la afirmación y los valores en el rango intermedio indican la inexistencia de un consenso.

$$IP = \frac{5(\%\text{ de MA}) + 4(\%\text{ de A}) + 3(\%\text{ de SN}) + 2(\%\text{ de ED}) + 1(\%\text{ de CD})}{5} \quad (5)$$

En la tabla 3 se muestran los resultados interpretados a partir del índice porcentual. La propuesta del autor presenta tres de los cuatro indicadores de efectividad, nivel que sólo es alcanzado por otras dos propuestas que tienen tiempos de autoría superiores.

Tabla 3. Indicadores de efectividad de la retroalimentación por propuesta.

Propuesta	Utilidad	Especificidad	Completitud	Capacidad cognitiva

PHP ITS	X	X		X
J-LATTE	X	X		X
Enström, Kreitz *et al.* 2011			X	
JITS	X	X		
Naudé, Greyling *et al.* 2010		X	X	
ELP	X	X		
Jurado 2010	X	X		
Verificator	X			
BlueFix	X			
Instructor (propuesta del autor)	X	X	X	

Discusión

Del análisis de los resultados obtenidos, se puede afirmar que la propuesta de la presente investigación reduce los tiempos de autoría de las actividades en comparación con otras que tienen un nivel de efectividad similar. Con respecto a otros métodos, con nivel de efectividad bajo, el sistema tiene tiempos de autoría superiores. La propuesta representa una reducción del tiempo de autoría de las actividades en relación con los métodos que necesitan de la especificación de variantes correctas e incorrectas. En relación con las propuestas alternativas, se puede afirmar que a pesar que se renuncia a una de las características que contribuyen a la efectividad de la retroalimentación, se logra incorporar la capacidad para utilizar características específicas del problema en el análisis.

A pesar de que se necesitan realizar pruebas con estudiantes para medir las ganancias en el aprendizaje que se obtienen de la aplicación, los resultados alcanzados sugieren que la propuesta puede contribuir de forma positiva a la enseñanza de la programación.

Conclusiones

A partir de la propuesta se obtiene un método para la construcción del modelo de dominio de un Tutor Inteligente de Programación. Con ella se posibilita que se puedan incluir actividades en el sistema sin elevados tiempos de autoría y que se produzca retroalimentación efectiva para actividades prácticas de tipo implementación. La existencia de esta característica está soportada en la combinación de técnicas para el análisis estático y dinámico de soluciones con la especificación de entradas/salidas y el algoritmo desarrollado para su procesamiento. Esta condición representa una novedad con respecto a las propuestas encontradas en la bibliografía, donde la inclusión de características del problema implica un deterioro en la efectividad o un aumento considerable del tiempo de autoría.

Como líneas de trabajo futuro para continuar la presente investigación se plantea realizar un estudio donde se evalúe los resultados de la propuesta a través de su aplicación en el contexto de diferentes marcos metodológicos. Además,

se buscará profundizar desde la perspectiva teórica en integrar métodos de evaluación automática para otros tipos de actividades, por ejemplo las que hacen uso de interfaces mediante formularios.

Referencias

- ALEVEN, V. Rule-Based Cognitive Modeling for Intelligent Tutoring Systems. En: R. Nkambou, J. Bourdeau y R. Mizoguchi. (editores). Advances in Intelligent Tutoring Systems. Springer-Verlag, 2010, p. 33-62.
- ALTUNA, E. J. y GUIBERT, L. Un modelo para la evaluación y recomendación de interacciones en el ámbito educativo. En: Memorias del Congreso Internacional COMPUMAT 2013. La Habana: 2013, p. 101-113.
- BARBHUIYA, R. K.; MUSTAFA, K., et al. Design specifications for a generic Intelligent Tutoring System. En: Proceedings of the International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government. Las Vegas, USA: CSREA Press, 2011, p. 399-409.
- BECK, J.; CHANG, K., et al. Does Help Help? Introducing the Bayesian and Assessment Methodology. En: Proceedings of the International Conference on Intelligent Tutoring Systems. Springer, 2008, p. 383–394.
- BOURDEAU, J. y GRANDBASTIEN, M. Modeling Tutoring Knowledge. En: R. Nkambou, J. Bourdeau y R. Mizoguchi. (editores). Advances in Intelligent Tutoring Systems. Springer-Verlag, 2010, p. 123-143.
- BRAWNER, K. W.; HOLDEN, H. K., et al. Understanding the Impact of Intelligent Tutoring Agents on Real-Time Training Simulations. En: Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2011. 2011, p. 170-182.
- CHI, M.; VANLEHN, K., et al. Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics. En: Proceedings of the International Conference on Intelligent Tutoring Systems. 2010, p. 224-234.
- CONATI, C. Intelligent Tutoring Systems: New Challenges and directions. En: 20th International Joint Conference on Artificial Intelligence. San Francisco: 2009, p.
- DOMINGUEZ, A. K.; YACEF, K., et al. Data Mining for Individualised Hints in eLearning. En: Proceedings of the International Conference on Educational Data Mining. Pittsburgh, PA, USA: Carniege Learning, 2010, p. 91-100.
- EAGLE, M. J. y BARNES, T. Evaluation of Automatically Generated Hint Feedback. En: Proceedings of the International Conference on Educational Data Mining. Memphis, Tennessee, USA: International Educational Data Mining Society, 2013, p. 372-381.

- ENSTRÖM, E.; KREITZ, G., et al. Five Years with Kattis – Using an Automated Assessment System in Teaching. En: Proceedings of the 41st ASEE/IEEE Frontiers in Education Conference. Rapid City: IEEE, 2011, p. 31-44.
- HAKIMZADEH, H.; ADAIKKALAVAN, R., et al. Successful Implementation of an Active Learning Laboratory in Computer Science. En: Proceedings of the SIGUCCS'11,. California, EEUU: ACM, 2011, p.
- HOLLAND, J.; MITROVIC, A., et al. J-LATTE: a Constraint-based Tutor for Java. En: Proceedings of the 17th International Conference on Computers in Education. Hong Kong: Asia-Pacific Society for Computers in Education, 2009, p. 142-146.
- JUNCO VÁZQUEZ, T. O. Un juez en línea ajustado a las necesidades de la docencia, Universidad de las Ciencias Informáticas. Tesis de Maestría, 2012.
- JURADO, F. Proposal for Evaluating Computer Programming Algorithms to Provide Instructional Guidance and Give Advice. Tesis Doctoral, 2010.
- LAZAR, T. y BRATKO, I. Data-Driven Program Synthesis for Hint Generation in Programming Tutors. En: Proceedings of the International Conference on Intelligent Tutoring Systems. Springer, 2014, p. 306-311.
- LE, N.-T. Using weighted constraints to build a tutoring system form logic programming. Informatik. Hamburg, Hamburg University. Tesis Doctoral, 2011.
- LE, N.-T.; LOLL, F., et al. Operationalizing the Continuum between Well-Defined and Ill-Defined Problems for Educational Technology. IEEE Transactions on Learning Technologies, 2013, 6(3): p. 258-270.
- LE, N. T.; MENZEL, W., et al. Considering Ill-Definedness Of Problem Tasks Under The Aspect Of Solution Space. En: Proceedings of the 23st International Conference of the Florida Artificial Intelligence Research Society (FLAIRS). AAAI Press, 2010, p. 534–535.
- MITROVIC, A. An intelligent SQL tutor on the web. International Journal of Artificial Intelligence in Education (IJAIED), 2003, 13(2): p. 173-197.
- MITROVIC, A.; OHLSSON, S., et al. The effect of positive feedback in a constraint-based intelligent tutoring system. Computers & Education, 2013, 60(1): p. 264-272.
- MULDNER, K. y CONATI, C. A Decision-Theoretic Tutor for Analogical Problem Solving. En: L. E. Sucar, E. F. Morales y J. Hoey. (editores). Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions. IGI Global, 2012, p. 219-247.
- NAUDÉ, K. A.; GREYLING, J. H., et al. Marking student programs using graph similarity. Computers & Education, 2010, 54(2): p. 545-561.

- OREHOVACKI, T.; RADOSEVIC, D., et al. Acceptance of Verificator by Information Science Students. En: Proceedings of the International Conference on Information Technology Interfaces. IEEE, 2012, p.
- RAZZAQ, L. y HEFFERNAN, N. To Tutor or not to Tutor: That is the Question. En: Proceedings of the Artificial Intelligence in Education. Amsterdam: IOS Press, 2009, p.
- STAMPER, J. C.; EAGLE, M. J., et al. Experimental Evaluation of Automatic Hint Generation for a Logic Tutor. International Journal of Artificial Intelligence in Education, 2013, 22(1): p. 3-17.
- SYKES, E. R. y FRANEK, F. An Intelligent Tutoring System Prototype for Learning to Program Java. 2003: p.
- TRUONG, N.; BANCROFT, P., et al. Learning to program through the web. En: Proceedings of the SIGCSE conference on Innovation and technology in computer science education. ACM, 2005, p.
- URBANCIC, J. y TRAMPUS, M. Putka - A Web Application in Support of Computer Programming Education. International Journal Olimpiads in Informatics, 2012, 6(1): p. 205-211.
- VANLEHN, K. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educational Psychologist, 2011, 46(4): p. 197-221.
- VUJOŠEVIĆ-JANIČIĆ, M.; NIKOLIĆ, M., et al. Software verification and graph similarity for automated evaluation of students' assignments. Information and Software Technology, 2013, 55(6): p. 1004-1016.
- WATSON, C.; LI, F. W. B., et al. BlueFix: Using Crowd-Sourced Feedback to Support Programming Students in Error Diagnosis and Repair. En: Proceedings of the International Conference on Web-Based Learning - ICWL Sinaia, Romania: Springer Berlin Heidelberg, 2012, p.
- WATSON, C.; LI, F. W. B., et al. Learning Programming Languages through Corrective Feedback and Concept Visualisation. En: Proceedings of the International Conference on Web-Based Learning - ICWL. Hong Kong: Springer Berlin Heidelberg, 2011, p.
- WEBER, G. y BRUSILOVSKY, P. ELM-ART: An adaptive versatile system for Web-based instruction. International Journal of Artificial Intelligence in Education, 2001, 12(4): p. 351-384.
- WERAGAMA, D. S. Intelligent Tutoring System for learning PHP. School of Electrical Engineering & Computer Science, Queensland University of Technology. Tesis doctoral, 2013.
- WOOLF, B. P. Building Intelligent Interactive Tutors, Morgan Kaufmann, 2009.
- ZHAO, Q.; WANG, F., et al. Arbiter: the Evaluation Tool in the Contests of the China NOI International Journal Olimpiads in Informatics, 2013, 7(1): p. 180-185.