ISSN 1870-4069

Ambiente inteligente de aprendizaje de depuración de errores en programas Java

María Lucia Barrón-Estrada, Ramón Zatarain-Cabada,
Minerva Valencia-Rodríguez, Gloria E. Peralta-Peñuñuri

Tecnológico Nacional de México – Instituto Tecnológico de Culiacán,
División de Estudios de Posgrado e Investigación, Culiacán, Sinaloa, México
{lbarron, rzatarain, mvalencia, gperalta}@itculiacan.edu.mx

Resumen. La tarea de depurar los errores de un programa es una de las más complejas y difíciles para un estudiante que se inicia en el proceso de aprender programación. Los compiladores ofrecen ayudas para realizar dicha tarea, pero ésta se enfoca más a apoyar a programadores expertos que a novatos. En este trabajo se presenta un ambiente inteligente para aprender a depurar errores en el código de programas escritos en el lenguaje de programación Java. La implementación del sistema experto del ambiente se basó en la teoría de seguimiento de ejemplos, para el proceso de aprendizaje se usó un agente pedagógico y técnicas de gamificación. El ambiente fue evaluado exitosamente con estudiantes de nivel básico de ingeniería en sistemas computacionales e ingeniería en tecnologías de la información y comunicaciones.

Palabras clave: java, depuración de errores, ejemplos erróneos, seguimiento de ejemplos, ambiente de aprendizaje, agente pedagógico, gamificación.

Intelligent Learning Environment for Debugging Errors in Java Programs

Abstract. One of the most complex and difficult tasks for a novice programmer as student is to debug a program. Compilers offers tools to carry out this task, which nevertheless focuses more on supporting expert programmers than novices. In this paper we present an intelligent environment to help students learn how to debug code in the Java programming language. Example-tracing theory was used to implement the expert system. In addition, the learning process was supported by a pedagogical agent and gamification techniques. The environment was successfully evaluated with freshmen students of computer systems engineering and engineering in information and communication technologies.

Keywords: java, debugging, incorrect examples, example tracing, learning environment, pedagogical agent, gamification.

1. Introducción

Aprender a desarrollar programas para una computadora, es una tarea compleja que involucra diversas habilidades requeridas durante los pasos de la metodología de desarrollo, que van desde la comprensión del problema y el planteamiento de la solución, la selección del lenguaje de programación y codificación, las pruebas y depuración de errores, finalizando con la ejecución y mantenimiento del programa. La complejidad que representa enseñar y aprender programación [1] ha tomado notoriedad desde sus inicios. Cualquier persona que aprenda a programar debe enfrentarse a la tarea de corregir los diferentes tipos de errores que pueden presentarse en un programa: sintácticos, semánticos, de lógica, etc. Una deficiente habilidad para depurar errores en programas produce frustración en el estudiante y además puede ocasionar la introducción de nuevos errores [2]. La enseñanza a través de ejemplos erróneos [4] se ha utilizado en diferentes áreas como: Matemáticas, Física y Medicina, entre otras, aportando buenos resultados en el aprendizaje de los estudiantes.

En el área de Computación, la enseñanza de la programación se centra principalmente en el desarrollo de la lógica algorítmica y el correcto uso de la sintaxis y semántica del lenguaje de programación usado para la codificación; la depuración de errores, pocas veces o nunca es utilizada por los maestros, libros de programación y software educativo como estrategia central de enseñanza. Un enfoque basado en errores fue desarrollado por Ginat [5], para un curso introductorio a la programación orientada a objetos.

La motivación juega un papel importante en los procesos de aprendizaje y actualmente, en el ámbito de los ambientes de aprendizaje, se utilizan diversas técnicas de gamificación para motivar a los estudiantes. La gamificación se refiere a la inclusión de elementos presentes en los videojuegos, en el diseño y desarrollo de aplicaciones correspondientes a un contexto educativo, con la intención de involucrar, motivar y mejorar el aprendizaje. Las mecánicas de juego se refieren a las reglas que se deben cumplir para producir cambios en el sistema. Generalmente se trata de recompensas virtuales [8], como por ejemplo ganar medallas, puntos, insignias o monedas, o bien, subir de nivel o categoría, así como visualizaciones de tablero de jugadores, entre otras.

En el estudio de Stott y sus colegas [9] hablan sobre las dinámicas subyacentes que hacen que los juegos sean atractivos y cómo éstas han sido utilizadas en prácticas pedagógicas modernas.

Por otra parte y en relación a este trabajo, existen metodologías para la representación del conocimiento de un experto que tienen un enfoque cognitivo, el cual se sustenta en la identificación de los procesos mentales durante el aprendizaje. Una de estas metodologías es llamada seguimiento de ejemplos (*example-tracing*). Este método evalúa el comportamiento del estudiante comparándolo flexiblemente contra ejemplos generalizados de comportamiento de resolución de problemas [6].

En seguimiento de ejemplos, los ejemplos generalizados, contienen las rutas de solución aceptables para un problema en particular (ruta óptima y sub-óptimas) además, también es posible registrar comportamiento erróneo (correspondiente a errores comunes) para ofrecer tutoría contextualizada a un tipo de error en particular y tienden a ser más fáciles de crear que las reglas de producción o las restricciones. Los

tutores basados en el método seguimiento de ejemplos pueden exhibir un comportamiento sofisticado de tutoría, ya que es posible proporcionar orientación a nivel paso, es decir, orientación durante la resolución de problemas complejos y no solo orientación hasta la finalización del ejercicio lo que hace que se cumpla con el criterio mínimo para considerase como un Sistema Tutor Inteligente (STI) según Vanlehn [7].

En este artículo, se presenta un ambiente inteligente de aprendizaje de depuración de errores en programas llamado *Find Error Java*, el cual utiliza un agente pedagógico llamado Lucy [3] para presentar al estudiante ejemplos erróneos con el fin de ayudarlo a comprender el error que se presenta en el código del programa y la forma cómo debe resolverlo. El ambiente está apoyado por un sistema experto el cual se construyó usando la metodología seguimiento de ejemplos y técnicas de gamificación para motivar al estudiante en su proceso de aprendizaje.

Este artículo está estructurado de la siguiente forma: en la Sección 2 se presentan los trabajos relacionados en las áreas de depuración de errores, sistemas tutores inteligentes y ambientes de aprendizaje, gamificación y agentes pedagógicos; en la Sección 3 se muestra el ambiente inteligente de aprendizaje desarrollado. La Sección 4 expone las pruebas realizadas y el análisis de las mismas y para finalizar, en la sección 5 se presentan las conclusiones y trabajos futuros.

2. Trabajos relacionados

En esta sección se abordan los temas relacionados con el trabajo que se presenta, tales como: el proceso de depuración de errores en programas, los sistemas tutores inteligentes y ambientes de aprendizaje, la gamificación y los agentes pedagógicos, los cuales que constituyen teoría y estado del arte.

2.1. Depuración de errores

Diversas investigaciones abordan el tema de la depuración de errores de programación partiendo de enfoques diferentes como: identificar el tipo de errores que comúnmente cometen los programadores novatos [10], visualizar dinámicamente la ejecución de programas [11], pronosticar la navegación en la web para la depuración de programas con el objetivo de ampliar herramientas que ofrecen los entornos de desarrollo [12], o abordar factores afectivos como frustración ante la depuración [2]. En estos trabajos se considera a la depuración de errores como una actividad de recuperación, posterior a la codificación y no como un tema central para la enseñanza.

Ahmadzadeh y sus colegas [9] proponen que la comprensión de patrones de depuración en errores de compilación y lógicos ofrece elementos que permiten mejorar el método de enseñanza, mientras que Murphy y otros [2] plantean que las habilidades de depuración consisten en aplicar simultáneamente la comprensión del funcionamiento previsto del programa, entender la ejecución real defectuosa del programa, tener experiencia en programación en general, comprender el lenguaje de programación y el dominio de la aplicación y tener conocimiento de errores y métodos de depuración.

Jackson, Cobb y Carver [13] lograron identificar los errores comunes que comenten los programadores novatos en lenguaje Java, y mencionan que durante su revisión a la literatura encontraron que los métodos más usados para la identificación de errores comunes de programación son el uso de la experiencia, la aplicación de encuestas a maestros, el conteo manual y categorización de errores o bien, el uso de las compilaciones proporcionadas por los mismos alumnos.

McCall y Kölling [14] proponen un método de categorización de errores comunes cometidos por programadores novatos al utilizar el lenguaje de programación Java, basado en los mensajes de diagnóstico que produce el compilador. Los investigadores clasificaron en 10 categorías todos los errores, relacionando los mensajes de error con el código que lo produjo.

En otras investigaciones [4] se ha demostrado que el uso adecuado de ejemplos erróneos conduce a buenos resultados. Aplicarlos en un entorno tecnológico educativo incluye, entre otros aspectos, el diseño de la instrucción, que se refiere al proceso o método de enseñanza, tal como la definición de las habilidades que deben ser fortalecidas, la decisión sobre la manera y el momento para dar retroalimentación al estudiante y la elección o innovación de los recursos pedagógicos-tecnológicos tales como el uso de agentes pedagógicos o recursos de gamificación.

2.2. Sistemas tutores inteligentes y ambientes de aprendizaje

No existe una definición universal aceptada del concepto Sistema Tutor Inteligente. Puede decirse que los Sistemas Tutores Inteligentes (STI) son aquellos que ofrecen una enseñanza diferencial, aquellos que adaptan su respuesta de enseñanza después de realizar un razonamiento sobre las necesidades de los estudiantes y el conocimiento del dominio [15].

Los STI se caracterizan por poseer mecanismos de inteligencia artificial que permiten adaptar la tutoría, pero generalmente no brindan al estudiante la posibilidad de elegir los temas o ejercicios del dominio. Los ambientes de aprendizaje a diferencia de los STI, ponen a disposición del estudiante todo el contenido educativo ofreciéndole la posibilidad de elegir los temas y secuencia de sus actividades de estudio, sin embargo, no cuentan con características de inteligencia artificial con el fin de guiar la tutoría.

Los Entornos Inteligentes de Aprendizaje (EIA) incorporan características de los ambientes de aprendizaje (flexibilidad de navegabilidad en los temas de instrucción) y los Sistemas Tutores Inteligentes (adaptabilidad de las experiencias de tutoría conforme al modelo del estudiante). Estas características que anteriormente eran consideradas como contradictorias hoy se consideran como complementarias [16].

2.3. Gamificación y agentes pedagógicos

Los videojuegos se han vuelto muy populares en los últimos años. Aunque el término gamificación, surgió en el ámbito digital y tuvo una aceptación generalizada después del 2010 [17], es importante mencionar que desde los años ochenta los investigadores han estudiado los beneficios de los enfoques basados en juegos en la educación [18,19].

Por otra parte, uno de los elementos importantes en los sistemas educativos es el tutor o profesor que determina la estrategia de enseñanza a utilizar para que los

estudiantes adquieran el conocimiento. En los sistemas usados en el aprendizaje electrónico, los tutores son llamados agentes pedagógicos (AP) y frecuentemente se representan a través de personajes que pueden ser reales o animados; ellos se encargan de establecer la comunicación con los estudiantes interactuando de diferentes formas para proporcionarles el apoyo cognitivo que requieren.

3. Ambiente inteligente de aprendizaje

3.1. Descripción general

Find Error Java es un ambiente inteligente de aprendizaje en el que se provee al estudiante una forma sencilla y ágil de aprender a evitar o superar errores comunes que se cometen durante las primeras etapas de aprendizaje de programación Java. El sistema está diseñado para ser usado por estudiantes novatos de nivel escolar bachillerato o universidad que se encuentren estudiando cursos introductorios de programación Java. Básicamente el estudiante debe identificar, entender y corregir el error sintáctico, semántico o lógico que contiene un pequeño programa o fragmento de código presentado.

Para desarrollar el ambiente de aprendizaje *Find Error Java* fue necesario establecer los elementos clave que permitirán su correcto funcionamiento para proporcionar asistencia a los estudiantes al momento de resolver los ejercicios planteados y además poder medir su aceptación y eficacia.

La Figura 1 muestra los 5 elementos clave definidos para el desarrollo del sistema.

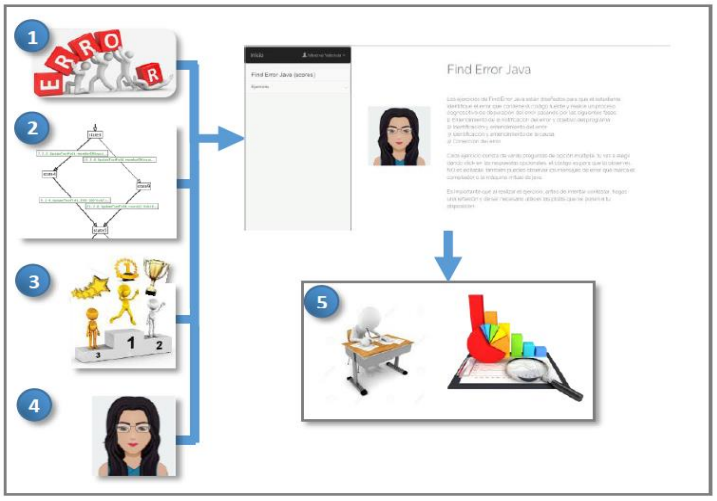

Fig. 1. Elementos que fundamentan el desarrollo de *Find Error Java*.

A continuación, se presenta una breve descripción de los elementos clave.

1. Identificación de los errores comunes que cometen los estudiantes novatos al aprender a programar y creación de banco de ejercicios.

2. Estrategia de instrucción de depuración de errores mediante la adaptación a la técnica de seguimiento de ejemplos (*example-tracing*).
3. Dinámicas y mecánicas de gamificación que ayuden a motivar al estudiante a la reflexión.
4. Agente pedagógico Lucy [3] integrado a la estrategia de instrucción basada en ejemplos erróneos.
5. Plan de pruebas experimentales mediante evaluaciones de usabilidad y ganancias de aprendizaje.

3.2. Proceso de instrucción

Partiendo de los errores comunes identificados, se realizó un Análisis de Tareas Cognitivas (CTA por sus siglas en inglés) que permitió identificar los diferentes recursos cognitivos que un experto utiliza consciente o inconscientemente durante la depuración de estos errores, con el fin de definir los lineamientos y el proceso de instrucción para la depuración.

Algunos de los lineamientos son:

1. Presentar conceptos básicos específicos para entender el funcionamiento del lenguaje o un error en particular, con el fin de ofrecer una solución.
2. Orientar al estudiante sobre el significado de los mensajes que arroja el compilador y la máquina virtual de Java según el tipo de error encontrado.
3. Para errores lógicos, ofrecer al estudiante orientación que le permita entender o interpretar el error con base en las entradas o salidas del programa.
4. Durante la realización de todos los ejercicios ofrecer al estudiante orientación sobre la diferencia entre errores sintácticos, semánticos o lógicos.

El proceso de instrucción para la depuración se definió con base a los principios cognitivos, los errores comunes detectados y los lineamientos, con el fin de promover el aprendizaje meta cognitivo al propiciar que el estudiante realice un razonamiento consciente sobre cada una de las fases del proceso de depuración. Además se utilizó un enfoque constructivista en el sentido que los ejercicios consideran conocimiento previo el cual pudiera ser impreciso.

El proceso de instrucción consta de 4 pasos:

1. Notificación del error (instrucciones del ejercicio).
2. Identificación del error en el código (línea o fragmento de código erróneo).
3. Entendimiento de la causa (conceptos, reglas o principios relacionados).
4. Corrección del error (instrucciones correctas).

3.3. Arquitectura del ambiente inteligente

Para el diseño del sistema se eligió el modelo arquitectónico en capas que permite una organización clara e independiente de los componentes (ver figura 2).

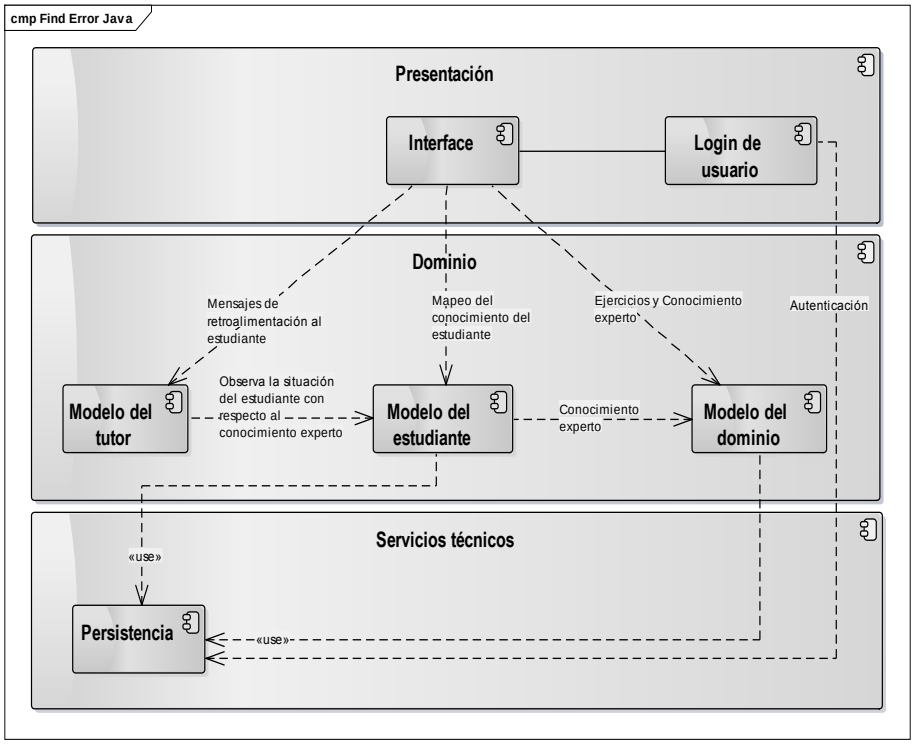

Fig. 2. Arquitectura lógica del sistema Find Error Java.

Capa de presentación. Contiene las interfaces con las que el usuario va a interactuar con el sistema, tales como registro de usuario, iniciar sesión, menú principal, realización de ejercicios y consulta de puntuación.

Capa de dominio. Integrada por 3 componentes principales: Modelo del tutor, Modelo del estudiante y Modelo del Dominio. Este último contiene las representaciones del conocimiento experto respecto a los ejercicios, estrategia de depuración, conceptos y procesos de pensamiento que deben transferirse al estudiante. Este componente tiene la responsabilidad de implementar una adaptación de la técnica de seguimiento de ejemplos (example-tracing) para procesar un grafo de comportamiento que incluye acciones incorrectas que pudiera cometer un estudiante durante el proceso de instrucción en depuración; así como también comportamiento óptimo, mismo que se pretende transferir al estudiante. Las acciones o comportamiento sub-óptimo de la técnica original, no fueron consideradas ya que el objetivo es mostrar un ejemplo de depuración de errores de manera óptima. El grafo de comportamiento experto del modelo del dominio contiene la estrategia de instrucción en la que se conduce al estudiante a un razonamiento consciente sobre el entendimiento de la notificación del error (instrucciones), identificación del error en el código, entendimiento de la causa y corrección del error. La figura 3 presenta el grafo de comportamiento experto utilizado por el modelo del dominio.

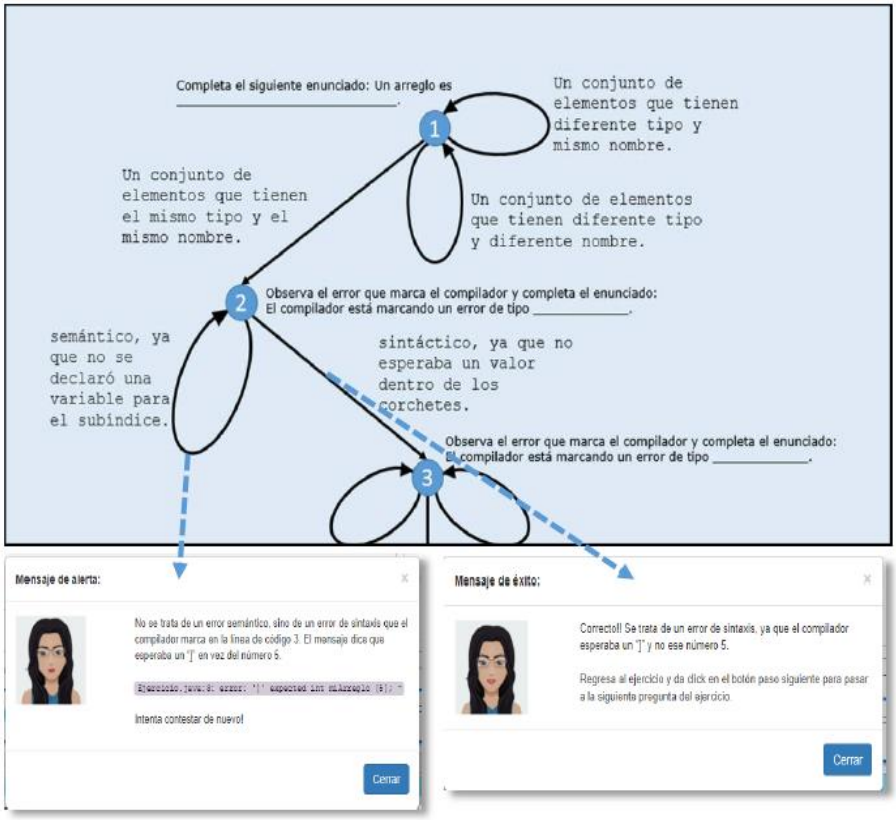

Fig. 3. Grafo de comportamiento experto.

C**apa de servicios técnicos. S**e encarga de realizar la lectura y almacenamiento de la información. Está conformada por modelos que el ORM de laravel (llamado Eloquent) permite utilizar. Los modelos tienen una correspondencia con cada una de las tablas que integran la base de datos de *FindErrorJava* y son los encargados de realizar la conexión a la base de datos y ejecutar las consultas de lectura y actualización.

3.4. Principales componentes del ambiente inteligente

En esta sección se describen los principales componentes con los que interactúa el usuario en el ambiente de aprendizaje FindErrorJava.

Interfaz de inicio. En la Figura 4 se presenta la interfaz de inicio, donde el agente pedagógico Lucy explica brevemente el proceso de instrucción que siguen los ejercicios. Se presenta el menú principal, con las ligas a los ejercicios que el estudiante puede realizar en orden ascendente, de menor a mayor grado de complejidad.

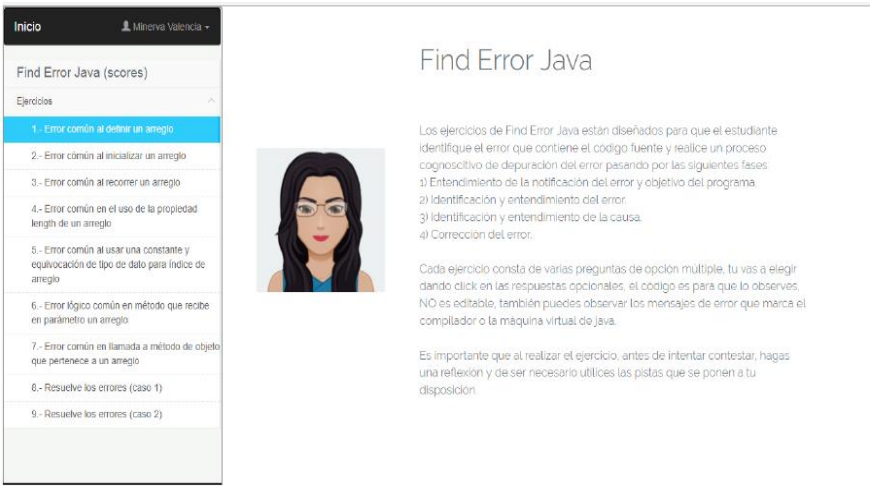

Fig. 4. Interfaz de inicio.

Interfaz de ejercicios. Esta es la interfaz principal del sistema donde el estudiante realiza los ejercicios de depuración de errores de programación y donde se combinan diferentes técnicas de instrucción con el objetivo de darle tutoría al estudiante para que no solo aprenda a corregir el tipo de error que contiene el código del ejercicio sino también que adquiera conceptos, principios, razonamientos y habilidades que le servirán incluso para resolver otro tipo de errores. Un ejercicio puede ser resuelto en *n* pasos, donde *n* puede ser mayor o igual que 1.

Un ejercicio está conformado por:

a) Título del ejercicio
b) Indicaciones
c) Código del ejercicio
d) Mensajes del compilador
e) Mensajes de la Máquina Virtual de Java (MVJ)
f) Pregunta o petición de interacción
g) Acciones optativas

En la Figura 5 se puede observar la correspondencia entre los atributos descritos y la interfaz de ejercicios.

Agente pedagógico. El agente pedagógico Lucy se utilizó para realizar intervenciones con el estudiante durante la realización de los ejercicios. Las intervenciones del agente pedagógico pueden ser de 3 tipos:

1) Lucy proporciona pistas bajo demanda.
2) Retroalimentación a nivel paso: Lucy ofrece una retroalimentación inmediatamente después de que el estudiante comete algún error, con el fin de intentar corregir la idea errónea que haya provocado su respuesta incorrecta (ver Figura 6) en el momento justo cuando ocurrió

3) Intervención por notificación: Cuando el agente pedagógico considera que el estudiante no está interesado en contestar correctamente, le envía una notificación cuyo objetivo es motivarlo a leer las pistas y realizar reflexiones más profundas. Los mensajes de notificación enviados no siempre son los mismos.

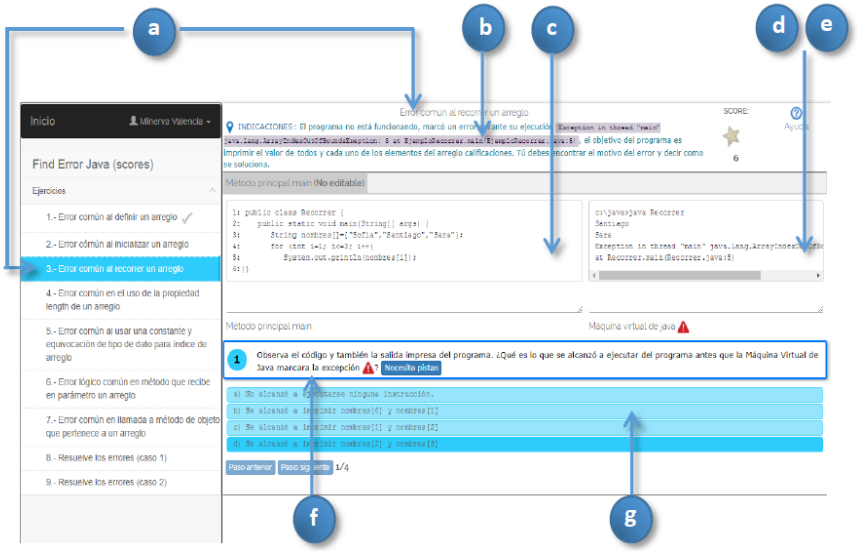

Fig. 5. Interfaz de ejercicios.

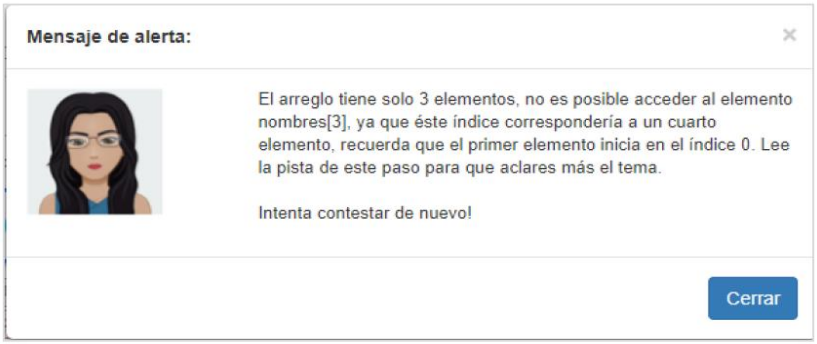

Fig. 6. Retroalimentación inmediata respuesta incorrecta del estudiante.

4. Pruebas

En esta sección se presentan las evaluaciones realizadas al sistema con el fin de dictaminar su eficacia y medir el grado de aceptación por parte de los usuarios.

4.1. Población y planificación de las pruebas

El estudio se realizó en el Instituto Tecnológico de Culiacán, y participaron 46 estudiantes (9 mujeres y 37 varones) en un rango de 18-21 años, pertenecientes a las carreras Ingeniería en Sistemas Computacionales e Ingeniería en Tecnologías de la Información y Comunicaciones.

La lista de actividades de este estudio se muestra en la tabla 1.

Tabla 1. Listado de actividades para la realización del estudio.

Orden	Actividad
1	*Selección de la muestra (estudiantes que participan en el estudio).*
2	*Distribución de la muestra en dos grupos: experimental y control.*
3	*Aplicación de examen Pre-Test a ambos grupos.*
4	*Utilización de Find Error Java en grupo experimental.*
5	*Aplicación de encuesta TAM a grupo experimental.*
6	*Aplicación de examen Post-Test a ambos grupos.*

4.2. Efectividad y ganancias de aprendizaje significativo en depuración de errores

Para la evaluación de la efectividad en ganancia de aprendizaje, se decidió abordar con *Find Error Java* el tema de arreglos, ya que este tópico es generalmente un problema para los estudiantes durante los cursos básicos de programación, persistiendo estos problemas incluso en cursos posteriores a los básicos.

Para medir y comparar la ganancia de aprendizaje entre alumnos que usan *Find Error Java* y los que no lo usan, la muestra de 46 estudiantes se dividió en dos grupos, el primero denominado grupo experimental y el segundo denominado grupo de control.

A los 46 estudiantes, previo a la utilización de *Find Error Java*, se les aplicó una primera evaluación denominada Pre-Test relacionada a la depuración de errores. Posteriormente, se realizó una sesión de 45 minutos con el grupo experimental para que resolvieran los ejercicios de depuración de errores utilizando *Find Error Java* y para finalizar, se aplicó a ambos grupos una segunda evaluación denominada Post-Test.

En la Figura 7 se presenta un comparativo de las calificaciones obtenidas en el Pre-Test y Post-Test correspondientes al grupo experimental, el cual utilizó *Find Error Java*. La mayoría de los estudiantes (78.2%) del grupo experimental, presentaron un incremento en su calificación en el examen posterior al uso de la herramienta.

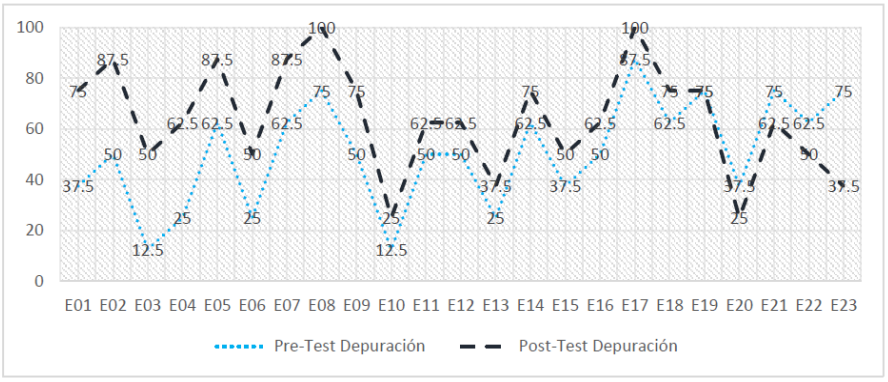

Fig. 7. Comparativo grupo experimental en relación a la depuración.

En la Figura 8 se presenta un comparativo de las calificaciones obtenidas en el Pre-Test y Post-Test correspondientes al grupo de control, el cual no utilizó *Find Error Java*. En este caso fue el 26% del grupo de control, el que presentó un incremento en su calificación.

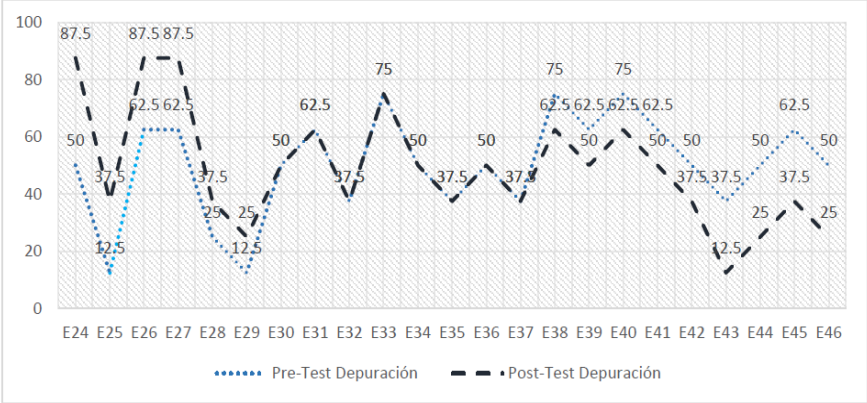

Fig. 8. Comparativo grupo de control en relación a la depuración.

5. Conclusiones y trabajo a futuro

Los errores de programación pueden utilizarse naturalmente como ejemplos erróneos, los cuales a su vez ofrecen una gran oportunidad de aprendizaje cuando son bien utilizados en el desarrollo de temas y actividades de aprendizaje. El uso de ejemplos erróneos genera conflicto cognitivo y siembran curiosidad en el estudiante (animan a la reflexión e investigación).

Find Error Java implementa un ambiente de aprendizaje para la depuración de errores que integra a la estrategia de instrucción de un agente pedagógico y elementos de gamificación. El objetivo de la gamificación fue motivar al estudiante para que intentara hacer reflexiones profundas antes de responder cada paso de los ejercicios;

además se requería propiciar un ambiente de competencia con la finalidad de ofrecer diversión.

Los resultados obtenidos en los experimentos donde se midió la efectividad en ganancias de aprendizaje mediante evaluaciones de conocimiento Pre-Test y Post-Test demuestran que el uso de una herramienta como *Find Error Java*, puede ser de gran utilidad para que los estudiantes de programación aprendan a depurar los errores del código que se generan cuando se diseña un programa para resolver un problema.

Como trabajo futuro se considera realizar más evaluaciones con temas diferentes del lenguaje Java. También buscaremos implementar el ambiente con otros lenguajes como Python y otros lenguajes en el campo de pensamiento computacional donde el enfoque es hacia lenguajes visuales orientados a niños y jóvenes.

Referencias

1. Jenkins, T.: On the difficulty of learning to program. In: Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences, 4(2002), pp. 53–58 (2002)
2. Murphy, L., Lewandowski, G., McCauley, R., Simon, B., Thomas, L., Zander, C.: Debugging: the good, the bad, and the quirky - a qualitative analysis of novices' strategies. In: ACM SIGCSE Bulletin 40(1), pp. 163–167 (2008)
3. Sosa, C.: Agente Pedagógico con estrategia “Aprendiz Cognoscitivo” para un ambiente de programación colaborativo. Tesis de maestría en Instituto Tecnológico de Culiacán (2016)
4. Borasi, R.: Capitalizing on errors as "springboards for inquiry": A teaching experiment. Journal for Research in Mathematics Education, pp. 166–208 (1994)
5. Ginat, D., Shmalo, R.: Constructive use of errors in teaching CS1. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education, pp. 353–358 (2013)
6. Aleven, V., Mclaren, B., Sewall, J., Koedinger, K.: A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors. International Journal of Artificial Intelligence in Education 19(2), pp. 105–154 (2009)
7. Vanlehn, K.: The behavior of tutoring systems. International Journal of Artificial Intelligence in Education 16(3), pp. 227–265 (2006)
8. Chorney, A.: Taking the Game Out of Gamification. Dalhousie Journal of Interdisciplinary Management, 8(1) (2012)
9. Stott, A., Neustaedter, C.: Analysis of gamification in education. Surrey, BC, Canada, 8(36) (2013)
10. Ahmadzadeh, M., Elliman, D., Higgins, C.: An analysis of patterns of debugging among novice computer science students. In: ACM SIGCSE bulletin 37(3), pp. 84–88 (2005)
11. Cross II, J. H., Hendrix, T. D., Barowski, L. A.: Combining dynamic program viewing and testing in early computing courses. In: Proceeding of IEEE 35th Annual Computer Software and Applications Conference (COMPSAC), pp. 184–192 (2011)
12. Lawrance, J., Bogart, C., Burnett, M., Bellamy, R., Rector, K., Fleming, S. D.: How programmers debug, revisited: An information foraging theory perspective. IEEE Transactions on Software Engineering 39(2), 197–215 (2013)
13. Jackson, J., Cobb, M., Carver, C.: Identifying Top Java Errors for Novice Programmers. In: Proceedings of 35th Annual Conference Frontiers in Education (FIE 2005), pp. 24–27 (2005)
14. McCall, D., Kölling, M.: Meaningful Categorisation of Novice Programmer Errors. In: Proceedings of 2014 Frontiers in Education Conference (FIE 2014), pp. 2589–2596 (2014)

15. Woolf, B. P.: Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning. Morgan Kaufmann (2010)
16. Brusilovsky, P., Pesin, L., Zyryanov, M.: Towards an adaptive hypermedia component for an Intelligent Learning Environment. In: Proceedings of International Conference on Human-Computer Interaction, Springer, Berlin, Heidelberg. pp. 348–358 (1993)
17. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From game design elements to gamefulness: defining "gamification". In: Proceedings of the 15th International Academic MindTrek Conference: Envisioning future media environments, pp. 9–15 (2011)
18. Malone, T.: Making Learning Fun: A taxonomy of intrinsic motivations for learning. Aptitude, learning, and instruction 3, pp. 223–253 (1987)
19. Gee, J.: What Video Games Have to Teach Us About Learning and Literacy? Computers in Entertainment, 2nd ed., Palgrave Macmillan, USA (2007)