Revisión de Herramientas de Apoyo en el Proceso de Enseñanza-Aprendizaje de Programación

Guerrero M.; GuamánD. S.; Caiza J. C.

Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica y Electrónica, Quito, Ecuador
e-mail: monicaguerrero120@gmail.com; {danny.guaman;julio.caiza}@epn.edu.ec

Resumen: El aprendizaje de programación es un tema de gran interés en investigación académica. Varias investigaciones se han centrado en la creación y uso de herramientas de calificación automática; sin embargo, existen otras de diferente tipo que también pueden ser muy útiles. Este artículo toma un enfoque amplio y hace una revisión de un conjunto extenso de herramientas. Se establece una tipología de ellas, y se provee de información relevante considerando una evolución temporal y organizándolas por cada tipo establecido. Se presenta una discusión en la cual se establecen los parámetros relevantes que se deberían considerar para comparary seleccionar una herramienta determinada. Este artículo pretende ayudar a docentes e investigadores que están interesados en realizar nuevos proyectos para mejorar el proceso de enseñanza-aprendizaje de programación.

Palabras clave:revisión; programación; aprendizaje; herramientas de apoyo; calificación automática.

Abstract: Programming learning is a hot topic in academic research. Although there have been much work focused in building and using automatic grading tools; there are other useful kinds of them. This paper takes a broad focus to give a review of a large set of tools. The main paper's contribution is the proposal of a typology for the reported tools. Considering that and adopting a time evolution perspective, a set of relevant information about the tools is presented. Additionally, it is presented a discussion to define the relevant parameters to be considered to select a tool for implementations. This paper aims to help teachers and researchers who are interested in projects oriented to improve the programming teaching-learning process.

Keywords:review; programming, learning, support tools, automatic grading.

1. INTRODUCCIÓN

Elaprendizaje de programación es un requisito fundamental dentro de la mayoría de ingenierías técnicas, y por ende es muy importante en la formación de un gran número de estudiantes. Adquiere aún más importancia debido a que es un proceso difícil, especialmente para aquellos que se enfrentan al aprendizaje de esta asignatura por primera vez. Además, dado que es un proceso de enseñanza-aprendizaje, también se vuelve un tema relevante para los profesores. El alto grado de dificultad que tiene el aprendizaje de la misma se ve reflejado en una alta tasa de fallos de los estudiantes en las asignaturas orientadas a la enseñanza de programación.

Varios trabajos han afirmado que el incremento en la cantidad de ejercicios que el estudiante realiza, y una rápida retroalimentación, colaboran en su proceso de aprendizaje [1]. También se debe tener en cuenta que el incremento de la cantidad de ejercicios, acompañado del gran número de estudiantes de ingeniería, hacen inviable un proceso de calificación manual. Es por esto que muchos trabajos se han orientado al uso de herramientas de calificación automática [2][3][4][5].

Se debe tener en cuenta que adicionalmente a las herramientas de calificación automática, hay otras reportadas y que se han usado de manera exitosa; así en [6] se reporta el uso screencasts y cuestionarios de autoevaluación, y en [7] se reporta el uso de Scratch. Es por ello que se hace necesario revisar y considerar el resto de herramientas existentes, en conjunto con las de calificación automática, para consolidar un conjunto más amplio y poder escoger aquellas que se puedan acoplar a los casos de las instituciones interesadas en mejorar el proceso enseñanza-aprendizaje de programación.

El resto del artículo está organizado como sigue: en la sección 2 se revisan los trabajos relacionados. En la sección 3 se realiza una revisión de las herramientas usadas en el proceso de enseñanza-aprendizaje estableciendo una tipología y señalando su evolución. . Una discusión considerando diferentes parámetros dependiendo de la tipología establecida es realizada en la sección 4. Finalmente, en la sección 5 se mencionan las conclusiones y posible trabajo futuro.

Artículo recibido el 15 de diciembre de 2014; revisado XX XX de 20XX.
Esta obra ha sido realizada como parte del proyecto de investigación semilla "Propuesta y despliegue de una solución basada en Herramientas Telemáticas (multimedia, herramientas web 2.0, servicios telemáticos) para reducir la tasa de estudiantes reprobados en la asignatura de programación en la Facultad de Ingeniería Eléctrica y Electrónica.", con código PIS – 14 – 27 en la Escuela Politécnica Nacional.
Autor para correspondencia: julio.caiza@epn.edu.ec, Cel. 0984718532, Ext. 2350.

2. TRABAJO RELACIONADO

El apoyo en el proceso enseñanza-aprendizaje de varias asignaturas de las carreras de ingeniería utilizan como soporte un amplio conjunto de herramientas de software. En [8] se presenta una revisión de herramientas informáticas licenciadas bajo el paradigma desoftware libre muypopulares y útilesdentro de las carreras de Ingeniería. Laasignatura de programación,debido a su vital importancia dentro de las carreras de ingeniería sigue siendo la motivación de varios proyectos de investigación, razón por la cual es abordada en el presente artículo. Varias herramientas, sobresaliendo aquellas orientadas a la calificación automática de ejercicios, han sido y están siendo utilizadas en varias instituciones alrededor del mundo.

Debido a la gran cantidad de herramientas existentes, algunos autores han realizado una revisión de herramientas y su evolución en el tiempo.

En [9] se adopta una perspectiva de evolución temporal y revisan los trabajos hasta el 2005, clasificando las herramientas reportadas en tres generaciones. La primera generación, se refería a programas de calificación realizados a medida, y que iban desde aquellos que calificaban tarjetas perforadas hasta aquellos que necesitaban integrarse con los compiladores o con el sistema operativo. La segunda generación se refería a aquellos sistemas desarrollados a partir de otras herramientas que venían dentro de los sistemas operativos; este tipo de sistemas podían ser llamados desde consola o a través de una interfaz gráfica. La tercera generación usaba tecnologías web, y se aprovechaba de sus inherentes ventajas, para el envío, administración, procesado, etc.

En [10] se cubrieron los trabajos desarrollados desde el 2005 al 2010. En este caso se puede notar las mejoras obtenidas desde tres perspectivas. Tecnológicamente, se reportó el uso de espacios seguros ante posibles ataques y la integración con LMSs (Sistemas de Administración de Aprendizaje). Pedagógicamente, se reportó el uso de análisis estático y dinámico para establecer calificaciones, yla existencia de herramientas con detectores de plagio. Respecto de la adopción de los sistemas construidos, se reportó proyectos de código abierto que hicieron que la aceptación de las herramientas mejore.

En [5] se hizo una revisión de un conjunto de herramientas que incluían aquellas que ya tenían años de investigación y por lo tanto tenían un esquema consolidado y varias funcionalidades, y aquellas recientes, con nuevas características, que se habían reportado entre 2010 y 2013. La perspectiva adoptada para su análisis les llevo a determinar los diferentes criterios de calificación usados en las herramientas, y se propuso una caracterización de los mismos.

Se debe mencionar que para conseguir mejoras en el proceso enseñanza-aprendizaje de programación no solo las herramientas de calificación automática son de utilidad, sino que hay varios tipos de herramientas que pueden ayudar y por lo tanto su estudio es necesario.

3. HERRAMIENTAS USADAS EN EL PROCESO ENSEÑANZA-APRENDIZAJE DE PROGRAMACIÓN

Hay una gran cantidad de herramientas usadas como ayuda en el proceso enseñanza-aprendizaje de programación, por consiguiente se requiere el establecimiento de una tipología que permita organizarlas y estudiarlas adecuadamente.

3.1 Establecimientode una Tipología

Aunque las diferentes herramientas reportadas persiguen un objetivo similar, que es ayudar en el proceso de enseñanza-aprendizaje de programación; estas presentan una gran diversidad en torno a su naturaleza.

Para establecer una tipología de las herramientas, se procedió a revisar el conjunto de características y se definieron las más notables. Se abstrajeron aquellas que son comunes en un grupo de herramientas y se las estableció como un tipo. Adicionalmente, en los casos en los cuales hubo más de una característica común, se escogió la que resultaba más representativa a la herramienta.

Los tipos determinados son:

- Herramientas de calificación automática.- Son aquellas herramientas orientadas a la automatización del proceso de calificación de ejercicios de programación. Ayudan a que el estudiante pueda realizar una mayor cantidad de ejercicios y pueda recibir una rápida retroalimentación, y también a que el profesor no se vea agobiado por un excesivo número de tareas a calificar.
- Herramientas multimedia.- Son herramientas que ayudan en el proceso de aprendizaje a través del uso de recursos como textos, imágenes, videos, entre otros.

  Además, se puede dar una integración de herramientas multimedia para dar lugar a los Objetos Digitales Educativos, los mismos que siguen una arquitectura modular de jerarquía creciente [11]. El nivel jerárquico se define en función de las características de estructura, funcionalidad y cobertura curricular. En nuestro estudio se distinguen los dos primeros niveles correspondientes a los Objetos Básicos (OB), y a los Objetos de Aprendizaje (OA).
- Sistemas inteligentes de tutoría.- Son herramientas complejas y orientadas al soporte de los estudiantes durante la escritura de sus programas. Incluyen mecanismos por medio de los cuales el estudiante resuelve el ejercicio en base a su nivel de conocimiento.
- Herramientas de aprendizaje visual.- Son herramientas orientadas a representar gráficamente un programa. Se puede observar a través de componentes gráficos el algoritmo del programa y la ejecución del mismo. Son herramientas que permiten un alto grado de interacción con el estudiante.

Adicionalmente, los IDEs (Entornos de Desarrollo Integrado) por sus características como el coloreado de sintaxis, la compilación, la depuración, la ayuda contextual, entre otras, ayudan al proceso de aprendizaje dependiendo de las técnicas de uso.

3.2Herramientas de Calificación Automática

Este tipo de herramientas evalúan de forma automática las tareas de programación, y se centran en calificar el código desarrollado por el alumno. La mayoría de herramientas usa como criterio de calificación únicamente la funcionalidad [5], es decir, luego de la ejecución del código se comprueba si la salida es correcta.

El aparecimiento de este tipo de herramientas ha surgido a la par de los lenguajes de programación [9]. Su importancia dentro del ámbito educativo ha permitido su constante desarrollo. Basándonos en el esquema adoptado en [9] se mencionan:

- Las herramientas de calificación automática que nacieron a partir del uso de código máquina y lenguaje ensamblador.El primer ejemplo de este tipo de sistemas fue dado en [12] usando tarjetas perforadas.
- Las herramientas a modo de scripts, llamadas a través de línea de comandos o incluso a través de interfaces gráficas de usuario, fueron creadas basándose en utilidades o herramientas provistas por el sistema operativo. Estas herramientas comenzaron a incorporar mecanismos para el análisis y evaluación de código, y que permitían a los docentes asignar pesos a las pruebas. Entre estos trabajos se pueden mencionar Ceilidh[13], y BOSS[14].
- Herramientas de calificación automática basadas en tecnologías Web, muchas de las cuales proporcionan algún grado de administración de contenido. A este grupo de herramientas pertenecen CourseMarker[15], Web-CAT [16], BOSS2 [17] y SAC [18].
- Adicionalmente,se puede mencionar nuevas herramientas que permiten una evaluación granular aplicando métodos de regresión y patrones, así tenemos Automata[19], y eGrader[20]. Aquellas que mejoran en aspectos de seguridad como Pythia[21], CAP [22] Y AUTOLEP [19]. Otras que consiguen una integración con LMSs como Virtual ProgrammingLab (VPL) [3], YAP3 + APAC [23], IT VBE [24] y PETCHA [1].

3.3Herramientas Multimedia

Lasherramientas multimedia (incluyendo texto, imágenes, audio, animación, video, entre otros) pretenden enriquecer los contenidos y favorecer al aprendizaje de los estudiantes, estimulando su percepción.

En [6]se hace uso de este tipo de herramientas dentro del LMS Moodle para explicar conceptos referentes a programación y el uso de aplicaciones de software. Usan screencasts y cuestionarios autoevaluados como medio de apoyo en cursos introductorios de programación, persiguiendo como objetivo el aumentar la motivación y el dinamismo en los estudiantes.

Othman et al. [25], Sharp et al. [26] y Fenwick et al. [27] reportan el uso de videos y screencast para ayudar a mejorar la comprensión de los estudiantes en la resolución de problemas de programación.

Naz et al.[28] reporta el uso de imágenes diseñadas de tal forma que permitan mostrar paso a paso la ejecución de un programa, ayudando a la comprensión de la lógica y el flujo de programa. Alhazbi[29] reporta el uso de un diario incorporado en el LMS Blackboard para que el alumno escriba durante la semana sobre su proceso de estudio para su clase de programación. Esta metodología tenía por objetivo el reforzar en los alumnos habilidades tales como la reflexión, análisis y autoevaluación.

En [30] y en el programa de apoyo a estudiantes en materias de programación LTI 2012-B de la Universidad de Guadalajara [31]hacen uso de OBs constituidos por wikis. Este tipo de herramientas promueven el aprendizaje colaborativo, ya que su principal ventaja es la comunicación e interacción entre los alumnos para la consulta y resolución de problemas.

En [32], [33] y [34] se reporta el uso de OAs para la enseñanza de programación. Estas herramientas se han diseñado para que se ajusten a las necesidades de los estudiantes, lo que les permite desarrollar sus habilidades mediante la interacción con ellas, y también para que les proporcionen actividades de evaluación.

3.4Sistemas Inteligentes de Tutoría

Una herramienta de este estilo utiliza técnicas procedentes de la IA (Inteligencia Artificial) y está diseñado para brindar una enseñanza personalizada.

Su aparición se dio en los años setenta en paralelo con el desarrollo de los sistemas EAO (Enseñanza Asistida por Ordenador) y apoyados con el uso de técnicas de inteligencia artificial. Este tipo de herramientas tenía como objetivo lograr la interacción con el alumno, incorporando un componente que simula a un tutor. Los primeros STI que aparecieron fueron LispTutor[35], PROUST [36] y MENO II [37] cuyas características fueron: su orientación a la enseñanza introductoria de programación, empleando los lenguajes Lisp y Pascal, su entorno de trabajo fue el de una aplicación independiente que sea simple de usar centrados en el manejo de bugs y errores que se generan en la escritura de código.

Los progresos en la IA, la psicología cognitiva y los paradigmas de programación dieron origen a la evolución de los STI hacia sistemas adaptables a los alumnos y que querían proporcionar un entorno de enseñanza altamente personalizable [38]. Sistemas que pertenecen a esta generación son ELM-PE [39], ELM-ART [40].

Los STI evolucionaron a los Sistemas Adaptativos Inteligentes basados en la web para la enseñanza, que precisamente tomaban ventaja de los beneficios de las

tecnologías web. A este tipo de sistemas pertenecen M-PLAT [41], CPP-Tutor [42], y C++ STL [43]y Altuna J. y Guibert L [44].
Es necesario mencionar herramientas como Prog-Tool [45] que hacen uso de agentes como soporte a las actividades a desarrollarse en la misma.

3.5 Herramientas de Aprendizaje Visual

Lasherramientas para aprendizaje visual pretenden ayudar en el proceso de aprendizaje de programación a través de representaciones gráficas de los algoritmos de solución y/o de la ejecución de un programa.
Durante la época de 1960 se desarrollaron las primeras herramientas en esta área. Una de estas herramientas fue Logo [46] que surgió en el año de 1967 y utilizaba el lenguaje de programación Lisp. Uno de los ambientes consistía en una tortuga que yace en el piso y cuyos movimientos eran dirigidos a través de los comandos escritos por los alumnos; permitía implementar operaciones de E/S, iteración, estructuras de datos, y algoritmos.
El Robot Karel20 [47], desarrollado en la Universidad Carnegie Mellon se enfocaba en enseñar los conceptos de programación estructurada. El estudiante escribía pequeños programas que resultaban en una serie de acciones que le permitían al robot desplazarse en un simulador que contiene calles y avenidas. Su estructura fue cambiada dando lugar al aparecimiento de JKarel Robot [48].
En [49] se reporta Turingal, que se componía de una máquina de Turing y una cinta magnética. El estudiante escribía instrucciones usando un lenguaje basado en la sintaxis de Pascal, que le permitía a la máquina de Turing escribir y leer símbolos a través de la cinta magnética.
El Media Lab. del MIT (2003-2007) en convenio con la Universidad de California Los Ángeles desarrollaron la herramienta Scratch[7], que es entorno 2D que apoya el aprendizaje a través de la creación de animaciones, juegos, e historias interactivas. Su entorno presenta funcionalidades *drag and drop*, y permite usar objetos llamados sprites y bloques que definen una operación lógica a realizar. Permite mejorar el aprendizaje de conceptos tales como: variables, operadores relaciones y lógicos, y tipo de datos.
También Greenfoot[50] está considerado dentro de este tipo y se presenta como un mundo virtual 2D. Requiere de la introducción de código Java a un nivel básico y permite el aprendizaje de los conceptos fundamentales tales como herencia, abstracción, polimorfismo a un mayor nivel que lo introducido en otros entornos como Scratch o Alice.
La Universidad Carnegie Mellon desarrolló Alice, misma que trabaja en un entorno de animación de un mundo virtual en 3D. Cuenta con protagonista virtual llamado Alice que ejecuta las acciones programadas por los alumnos. Las instrucciones se definen usando un editor gráfico, especificando las acciones a realizar con la ayuda de una paleta de herramientas y arrastrando y soltando los componentes. Alice aporta al aprendizaje de los fundamentos de programación orientada a objetos [51][52].
Existen herramientas derivadas de las anteriores generaciones y que mejoran su interactividad. Así se tiene PLM [53] que permite visualizar la ejecución de código paso a paso para diferentes lenguajes de programación.
También se reporta el uso combinado de herramientas de aprendizaje visual en Londoño S. y Orozco A. [54] donde se emplea simultáneamente Alice y el Robot Scribbler para fomentar la capacidad de análisis algorítmico y mejorar las habilidades para resolver problemas en los estudiantes. Jonas[55] expone el uso de Greenfoot [50] en combinación con el uso de estrategias de juegos de mesa, y a través del flujo del programa se pretende reforzar la comprensión de las diferencias entre clases y objetos variables, bucles, etc.

4. DISCUSIÓN

4.1Parámetros Relevantes

Todas las herramientas tienen un conjunto amplio de características, y por ende se hace necesario definir cuáles de ellas tienen mayor importancia y se convierten en parámetros relevantes a considerar antes de su posible uso. Se tienen:

- Lenguajes/paradigma de programación soportado.- Es importante debido a que cada interesado podría usar diferentes lenguajes de programación dentro de su curso.
- Modo de trabajo.- Se refiere a la manera en la cual la herramienta se implementa para su funcionamiento, resulta ser importante especialmente si se va a integrar con otros sistemas.
- Método de evaluación de código.- Indica las métricas o mecanismos usados para evaluar el código.
- Retroalimentación.- Hace referencia a la forma en la cual el estudiante recibe comentarios u observaciones sobre su trabajo.
- Tecnologías usadas.- Se refiere a las tecnologías usadas y sobre las cuales trabaja la herramienta, importante en integración de sistemas.
- Interactividad.- Hace referencia a la forma en la cual la herramienta interactúa con el estudiante.
- Tipo de recurso multimedia.- Indica el tipo de elemento multimedia usado o el tipo de información que representa.
- Colaboración.- Indica si la herramienta permite la comunicación o cooperación entre alumnos.
- Licencia de Software.- Parámetro que proporciona información acerca del tipo de licenciamiento de la herramienta.

Los parámetros a considerar variarán dependiendo del tipo de herramienta usada. Esto se verá en la siguiente sección.

4.2Herramientas de Calificación Automática

Este tipo de herramientas automatizan el proceso de evaluación y pueden ayudar a establecer una calificación, por lo que es necesario conocer el dominio de lo que evalúan (lenguajes), y como lo hacen (criterios usados). Así mismo, la

calidad y cantidad de retroalimentación que ofrezcan ayudarán en mayor o menor grado al estudiante, por lo que también es un parámetro fundamental a considerar.

Además, una institución, docente, o investigador interesado en utilizar la herramienta necesitará información respecto a su modo de trabajo y la tecnología sobre la que se soporta la herramienta debido a que necesitará instalar la misma, esto adquirirá más importancia cuando se requiera integración con otros sistemas.Finalmente, características como la forma en la cual interactúa el estudiante con la herramienta y el acceso al código fuente de la aplicación son relevantes. En el primer caso porque se busca una interacción que facilite el envío de tareas, o que anime al estudiante a seguir usando la herramienta; y en el segundo por políticas organizacionales, o porque podrán acceder al código de la herramienta para modificarlo según sus necesidades. El conjunto de herramientas estudiadas puede verse en la Tabla 1.

.

Tabla 1. Herramientas de Calificación Automática

Herramienta	**Lenguajes de programación soportados**	**Modo de Trabajo**	**Métodos de Evaluación de Código**	**Retroalimentación**	**Tecnologíasusadas**
Ceilidh	C, Pascal, C++	Aplicación de escritorio	Test dinámicos Tipografía Uso adecuado de estructuras	Información y notas acerca de donde se ocasiona la pérdida de puntuación Lecturas sobre resolución de ejercicios	Programas de sistema .act
BOSS	C	Software Package	Comparación carácter a carácter	Mensajes de error	Java
Course Marker	Java, C++	Standalone	Tipografía Uso adecuado de estructuras y objetos	Comentarios de explicación del código	Java Java RMI DATsys
Web-CAT	Java C++ y Pascal	Aplicación Web	Análisis estático	Salida de casos de prueba	Java
BOSS2	Pascal y Java	Aplicación Web	Análisis Dinámico Detección de plagio, JUnit	Salida de casos de prueba	Java JAva RMI
SAC	Java	Aplicación web	Análisis dinámico	Salida de casos de prueba Estadísticas acerca del desempeño	Java, JSP, Apache Tomcat 5.5 y PostgreSQL
Automata	Probada usando Lenguaje C	Aplicación Online	Rúbricas basadas en Modelos de Regresión	Reportes detallados de las características recogidas de los programas	-
eGrader	Java	AplicaciónGráfica	Análisis Estático y Análisis Dinámico	Reportes basados en los errores cometidos	Junit, Software Engineering Metrics (SEM)
Pythia	Soporta varios lenguajes de Programación/Probada usando Phyton	Aplicación Web	Métodos de evaluación usando scripts	Gráficos usando google chart tools que muestran, el número de accesos a los métodos.	XML Sandbox Linux
CAP	Java	Aplicación Web	Análisis Estático y Análisis Dinámico	Lista de errores del programa evaluado. Código de solución y estadísticas sobre los trabajos presentados	Applet Java
VPL	Variosparadigmas de programación	Plugin para Moodle	Detección de Plagio	Comentarios, lista de test fallidos y casos de prueba	Java Applet PHP
AUTOLEP	C	AplicaciónDistribuida	Análisis estático y testeo dinámico	Mensajes de advertencia ante error de semántica o estructura.	-
YAP3 + APAC	Soporta varios lenguajes de programación como C, C++, Java, PHP	Módulo de Moodle Aplicación con Servelets	Pruebas de entrada/salida	Salida de casos de prueba. Lista de resultados de la ejecución delprograma.	Java EE 7, API REST, Servelets
IT VBE	Pascal y C++	Plugin	Análisis Dinámico con Pruebas de Caja Blanca	Devuelve las correcciones del programa evaluado	-
PETCHA	Lenguajes soportados por Eclipse y Visual	Web	Mooshak	Salida de casos de prueba. Patrones de error	PExIL, IMS,bLTI LMS, JAWS

Tabla 2. Herramientas Multimedia

Herramienta	Lenguajes/Paradigma de programación soportado	Retroalimentación	Interactividad	Modo de trabajo	Proyecto referido
Screencast	Lenguajes soportados por Eclipse	Explicación de la resolución de un programa.	Medio de recuperación de la información	Integrado en Moodle	[6]
Diario	Programación Estructurada/C++	Escritura y lectura de los conceptos aprendidos en clase.	Medio de recuperación de la información	Integrado en Blackboard	[29]
Wiki	Estructurada, Funcional e Imperativa	Soluciones a los programas posteados.	Herramienta Colaborativa	Aplicación Web	[30]
Wiki	Programación Estructurada, POO y Desarrollo de Sofware	Escritura y lectura de los conceptos aprendidos en clase.	Herramienta Colaborativa	Aplicación Web	[31]
Imágenes	Algoritmos	Diagramas de flujo y Cuestionarios.	Medio de recuperación de la información	Integrado en Moodle y Claroline	[32]
Cuestionarios	C	Errores de compilación Revisión de conceptos.	Herramienta constructiva	Aplicación Web	[33]
Juegos, Texto Animaciones Imágenes	Python. C y Java	Salida de los casos de prueba Errores de compilación.	Herramienta constructiva	Aplicación Web	[34]
Video, Simulaciones y Animaciones	C,Matlab,Java	Explicación de la resolución de un programa.	Medio de recuperación de la información	Puede ser integrado en sistemas e-learning	[26]
Video	C#	Explicación de la resolución de un programa.	Medio de recuperación de la información	Online	[27]
eBook	Prolog	Explicación de conceptos de programación	Medio de recuperación de la información	Se integra con CMS y Sistemas de Calificación Automática	[28]
Imágenes	C/C++	Explicación paso a paso de la ejecución de un programa.	Medio de recuperación de la información	App Gráfica	[29]

4.3 Herramientas Multimedia

Debido a que hay una variedad de herramientas multimedia, es necesario establecer una comparativa definida por los tipos de herramientas existentes y que han sido usados para ayudar el proceso de aprendizaje de programación.Aunque el tipo de recurso referido tendrá ventajas inherentes a su naturaleza, denotando la interactividad, también es necesario considerar elementos útiles como la retroalimentación, el paradigma de programación soportado y el modo de trabajo. Los dos primeros pueden ser de mucha utilidad para definir un plan de trabajo dependiendo del lenguaje de programación que se usa en un curso.
El modo de trabajo con el cual se lo ha usado permitiría saber si es compatible con nuestro esquema de trabajo para obtener resultados exitosos. El resumen de las características de las herramientas se puede ver en la Tabla 2, cuya última columna muestra los proyectos de investigación reportados que han hecho uso de este tipo de herramientas.

4.4Sistemas de Tutoría Inteligentes

En este tipo de herramientas la resolución de ejercicios planteados retroalimentan al sistema y al alumno. La finalidad de estos sistemas es ayudar al estudiante en el desarrollo de programas a través de un tutor virtual que es el núcleo del sistema. Esta característica hace que varios de los parámetros relevantes aplicados en las herramientas de calificación automática, también sean aplicables aquí; así se mencionan: los criterios o métodos de evaluación, la forma de retroalimentación, el paradigma o los lenguajes de programación soportados, y las tecnologías usadas. La Tabla 3, muestra las herramientas de este tipo.

4.5 Herramientas de Aprendizaje Visual

Las herramientas de aprendizaje visual permiten asimilar, a través de una representación gráfica, conceptos de programación. Durante la creación de este tipo de sistemas se ha tenido claramente definido el tipo de lenguaje o paradigma sobre el cual se brindaría la ayuda, por lo que es un parámetro relevante a considerar.
Aunque la mayoría de este tipo de herramientas se ha desarrollado para trabajar como una aplicación independiente, hay casos en los que trabajan a través de interfaces web; por lo tanto su modo de trabajo es un parámetro relevante.
Debido a que la mayoría de ellas son aplicaciones instalables, es necesario conocer también la licencia de uso.
Finalmente, la visualización de retroalimentación es a través de animaciones y todas se han pensado para cursos introductorios de programación. El resumen de características relevantes se observan en la Tabla 4.

Tabla 3. Sistemas de Tutorías Inteligentes (página 1 de 2)

Herramienta	Lenguaje/Paradigma de Programación	Modo de Trabajo	Métodos de Evaluación	Retroalimentación	Tecnologíasusadas
LispTutor	Lisp	AplicaciónIndependiente	Reglas de Resolución de Problemas	Mensajes de diagnóstico por pieza de código.	-
PROUST	Pascal	AplicaciónIndependiente	Emplea métodos de análisis de bugs.	Mensajes de explicación acerca de bugs encontrado en el código	-
MENO II	Pascal	AplicaciónIndependiente	Componentes de la solución del programa a través de PDL	Sugerencias sobre el código erróneo generado por el estudiante	-
ELM-PE	Lisp OOP	AplicaciónIndependiente	Análisisdinámico	Explicación de los errores en el código	-
ELM-ART	Lisp OOP	Web	Casos de prueba	Lecturas Mensajes enviados por los tutores	Common Lisp WWW CL-HTTP
M-PLAT	OOP	Web	AnálisisDinámico	Explicación de los errores en el código	Web-Service, Java, SOAP, C#, UDDI, WSDL, XML
CPP-TUTOR	C++, ProgramaciónEstructurada	Web	Algoritmo de Programacióndinámica	Proporciona retroalimentación inteligente apoyada en la compilación del código, el enunciado del programa, el código desarrollado, entre otros	Web-services
C++ STL	Usada frecuentemente para Programación Estructurada	Web	Casos de prueba	Explicación de soluciones a los problemas planteados	Java, Mysql
Generación de pistas durante el aprendizaje de la programación para concursos usando el análisis estático y dinámico de las soluciones.	ProgramaciónOrientada a Objetos	Web	Análisisestático y dinámico	Proporciona soluciones y comentarios dados por el profesor. Información suministrada acerca de los errores cometidos.	Java EE JavaServerPages Apache Tomcat
Prog-Tool	Programación Orientada a Objetos	Aplicación Multi-Agente	Cumplimiento de criterios de evaluación.	Mensajes que indican si la opción seleccionada es correcta o incorrecta	Java Tilab Jade

Tabla 4. Herramientas de Aprendizaje Visual

Herramienta	Paradigma de Programación	Modo de Trabajo	Licencia
Logo	Lisp OOP	App. Independiente	Software Libre
Karel20	Estructurada	App. Independiente	Software Libre
JKarel Robot	Estructurada	App. Independiente	Software Libre
Turingal P	Imperativa	App. Independiente	Software Libre
Scratch	POO	App. Independiente Aplicación web.	Propietario
Greenfoot	POO	App. Independiente	GNU
Alice	POO	App. Independiente	BSD
PLM	Java, Python y Scala	App. Independiente	Software Libre

5. CONCLUSIONES Y TRABAJO FUTURO

Se ha realizado una revisión de un amplio conjunto de herramientas de apoyo en el proceso enseñanza-aprendizaje de programación, con la intención de adoptar un enfoque más amplio que permita considerar muchos tipos de herramientas reportadas, y no limitarse únicamente a aquellas que se han orientado a la calificación automática de tareas.

Debido a la gran cantidad de herramientas reportadas y a su distinta naturaleza, se ha propuesto una tipología de clasificación, la misma que tiene como objetivo el ayudar a acceder de manera más ordenada y rápida a la información.

Considerando cada uno de los tipos establecidos, se ha realizado una revisión de las distintas herramientas considerando una perspectiva temporal; esto ha permitido

estudiar la problemática y avances encontrados en los trabajos que reportan las herramientas.

Dentro de cada tipo establecido, se ha definido un conjunto de parámetros relevantes y que se deberían tener en cuenta ante posibles desarrollos o implementaciones de soluciones futuras. Esta información se la ha sintetizado a través de tablas comparativas.

El presente trabajo pretende ayudar a los docentes e investigadores a través de un acceso rápido a información de las herramientas desarrolladas, mismas que han sido clasificadas dentro de tipos establecidos. Esto les permitirá colocarse rápidamente en el contexto de este ámbito de investigación.

Se pretende que esta tipología sea usada por los docentes e investigadores para que definan soluciones o planes de trabajo que usen las herramientas reportadas para su aplicación sobre el proceso de enseñanza-aprendizaje de programación.

Debido a la gran cantidad de herramientas y de diverso tipo, se puede pensar en definir arquitecturas que integren las mismas, en lugar de seguir construyendo nuevas.

REFERENCIAS

[1] R. A. P. Queirós and J. P. Leal, "PETCHA: A programming exercises teaching assistant," in *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, 2012, pp. 192-197.

[2] S. Gupta and S. K. Dubey, "Automatic assessment of programming assignment," *Computer Science & Engineering: An International Journal (CSEIJ)*, vol. 2, 2012.

[3] J. C. Rodríguez-del-Pino, E. Rubio-Royo and Z. J. Hernández-Figueroa, "A virtual programming lab for moodle with automatic assessment and anti-plagiarism features," in *Proceedings of the 2012 International Conference on e-Learning, e-Business, Enterprise Information Systems, & e-Government*, 2012.

[4] J. López Reguera, C. Hernández Rivas and Y. Farran Leiva, "Una plataforma de evaluación automática con una metodología efectiva para la enseñanza/aprendizaje en programación de computadores," *Ingeniare Revista Chilena De Ingeniería*, vol. 19, pp. 265-277, 2011.

[5] J. C. Caiza and J. M. del Alamo, "Programming assignments automatic grading: Review of tools and implementations," in *ICERI2013 Proceedings [Ref_]*, Valencia, Spain, 2013.

[6] B. San Miguel, S. Aguirre, J. del Alamo and M. Cortés, "A proposal for enhancing the motivation in students of computer programming," *ICERI2012 Proceedings*, pp. 1157-1164, 2012.

[7] D. J. Malan and H. H. Leitner, "Scratch for budding computer scientists," in *ACM SIGCSE Bulletin*, 2007, pp. 223-227.

[8] R. Herrera, " Herramientas de Software Libre paraAplicaciones en Ciencias e Ingeniería," in Revista Politécnica, vol. 32, 2013, pp. 1-8.

[9] C. Douce, D. Livingstone and J. Orwell, "Automatic test-based assessment of programming: A review," *Journal on Educational Resources in Computing (JERIC)*, vol. 5, pp. 4, 2005.

[10] P. Ihantola, T. Ahoniemi, V. Karavirta and O. Seppälä, "Review of recent systems for automatic assessment of programming assignments," in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, 2010, pp. 86-93.

[11] *Objetos educativos digitales*. Available: http://cefire.edu.gva.es/file.php/1/Comunicacion_y_apertura/B4_RecursosEducativos/5objetos_educativos_digitales.html.

[12] J. Hollingsworth, "Automatic graders for programming classes," *Commun ACM*, vol. 3, pp. 528-529, 1960.

[13] S. Benford, E. Burke, E. Foxley, N. Gutteridge and A. M. Zin, "Ceilidh as a course management support system," *J. Educ. Technol. Syst.*, vol. 22, pp. 235-250, 1993.

[14] M. Joy and M. Luck, "Effective electronic marking for on-line assessment," in *ACM SIGCSE Bulletin*, 1998, pp. 134-138.

[15] C. Higgins, T. Hegazy, P. Symeonidis and A. Tsintsifas, "The coursemarkercba system: Improvements over ceilidh," *Education and Information Technologies*, vol. 8, pp. 287-304, 2003.

[16] S. H. Edwards and M. A. Perez-Quinones, "Web-CAT: Automatically grading programming assignments," in *ACM SIGCSE Bulletin*, 2008, pp. 328-328.

[17] M. Joy, N. Griffiths and R. Boyatt, "The boss online submission and assessment system," *Journal on Educational Resources in Computing (JERIC)*, vol. 5, pp. 2, 2005.

[18] B. Auffarth, M. López-Sánchez, J. Campos iMiralles and A. Puig, "System for automated assistance in correction of programming exercises (SAC)," in *Proceedings of CIDUI 2008*, 2008, pp. 104-113.

[19] S. Srikant and V. Aggarwal, "A system to grade computer programming skills using machine learning," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 1887-1896.

[20] F. Al Shamsi and A. Elnagar, "An Intelligent Assessment Tool for Students' Java Submissions in Introductory Programming Courses," *Journal of Intelligent Learning Systems and Applications*, vol. 4, pp. 59, 2012.

[21] S. Combéfis and V. de Saint-Marcq, "Teaching programming and algorithm design with pythia, a web-based learning platform," *Olympiads in Informatics*, vol. 6, pp. 31-43, 2012.

[22] Ó. Sapena, M. Galiano, N. Prieto and M. Llorens, "Aprender, enseñar y evaluar con CAP, un Corrector Automático de tareas de Programación,"," *XIX Jornadas De Enseñanza Universitaria De La Informática (JENUI 2013), Castellón*, 2013.

[23] D. Pohuba, T. Dulik and P. Janku, "Automatic evaluation of correctness and originality of source codes," in *Microelectronics Education (EWME), 10th European Workshop on*, 2014, pp. 49-52.

[24] B. Skūpas, A. Čaplinskas, J. Augutis, E. Bareiša, G. Kulvietis, V. Marcinkevičius, D. Dzemydienė and R. Šeinauskas, "A Method for Semi-Automatic Evaluation and Testing of Programming Assignments," *A Method for Semi-Automatic Evaluation and Testing of Programming Assignments*, 2013.

[25] A.Othman, C. Pislaru and A. Impes, "Improving the Quality of Technology-Enhanced Learning for Computer Programming Courses," *International Journal of Information and Education Technology*, vol. 4, pp. 83-88, 2014.

[26] J. H. Sharp and L. A. Schultz, "An exploratory study of the use of video as an instructional tool in an introductory C# programming course," *Information Systems Education Journal*, vol. 11, pp. 33, 2013.

[27] J. B. Fenwick Jr, B. L. Kurtz, P. Meznar, R. Phillips and A. Weidner, "Developing a highly interactive ebook for CS instruction," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 2013, pp. 135-140.

[28] S. Naz, S. H. Shirazi, T. Iqbal, D. Irfan, M. Junaid and Y. Naseer, "Learning Programming through Multimedia and Dry-run," 2014.

[29] S. Alhazbi, "Using e-journaling to improve self-regulated learning in introductory computer programming course," in *Global Engineering Education Conference (EDUCON), 2014 IEEE*, 2014, pp. 352-356.

[30] R. A. Lotufo, R. C. Machado, A. Körbes and R. G. Ramos, "Adessowiki on-line collaborative scientific programming platform," in *Proceedings of the 5th International Symposium on Wikis and Open Collaboration*, 2009, pp. 10.

[31] Universidad de Guadalajara, "Programa de apoyo a estudiantes en materias de programación LTI 2012-B,"

[32] K. Olmos, C. Morales, T. Rojas and L. Fernández, "Objetos de aprendizaje enfocados a la resolución de problemas para facilitar la enseñanza de la programación,".

[33] V. J. Carrasco, J. H Guevara J, "Implementación de un conjunto de objetos de aprendizaje y una aplicación web de cuestionarios dinámicos integrados en una herramienta de soporte para el autoaprendizaje de los conceptos de programación básica dirigido a estudiantes de la Facultad de Ingeniería en Electricidad y Computación", Proyecto Fin de Carrera, ESPOL, Guayaquil, 2009. Disponible en: http://www.cib.espol.edu.ec/Digipath/D_Tesis_PDF/D-39238.pdf.

[34] J. A. Villalobos, N. A. Calderon and C. H. Jiménez, "Developing programming skills by using interactive learning objects," *ACM SIGCSE Bulletin*, vol. 41, pp. 151-155, 2009.

[35] R. Anderson and B. J. Reiser, "The LISP tutor," *Byte*, vol. 10, pp. 159-175, 1985.

[36] W. L. Johnson and E. Soloway, "PROUST: Knowledge-based program understanding," *Software Engineering, IEEE Transactions on*, pp. 267-275, 1985.

[37] E. M. Soloway, B. Woolf, E. Rubin and P. Barth, "Meno-ii: An intelligent tutoring system for novice programmers," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence-Volume 2*, 1981, pp. 975-977.

[38] *Sistemas Tutores Inteligentes*. Available: http://prezi.com/ll3znnerauut/copy-of-sistemas-tutores-inteligentes-inteligencia-artificial/.

[39] G. Weber and A. Mollenberg, "ELM-PE: A Knowledge-based Programming Environment for Learning LISP." 1994.

[40] G. Weber and P. Brusilovsky, "ELM-ART: An adaptive versatile system for Web-based instruction," *International Journal of Artificial Intelligence in Education (IJAIED)*, vol. 12, pp. 351-384, 2001.

[41] A. Nuez, J. Fernández, J. D. Garcia, L. Prada and J. Carretero, "M-PLAT: Multi-programming language adaptive tutor," in *Advanced Learning Technologies, 2008. ICALT'08. Eighth IEEE International Conference on*, 2008, pp. 649-651.

[42] S. S. A. Naser, "Developing an intelligent tutoring system for students learning to program in C," *Information Technology Journal*, vol. 7, pp. 1055-1060, 2008.

[43] C. Lee and M. S. Baba, "The Intelligent Web-Based Tutoring System using the C Standard Template Library," *Malaysian Online Journal of Instructional Technology (MOJIT)*, vol. 2, pp. 34-42, 2005.

[44] E. J. Altuna Castillo and L. Guibert Estrada, "Generación de pistas durante el aprendizaje de la programación para concursos usando el análisis estático y dinámico de las soluciones," *Ingeniare.Revista Chilena De Ingeniería*, vol. 21, pp. 205-217, 2013.

[45] M. Goyal, "Development of agent-based intelligent tutoring system for teaching object-oriented programming concepts," in Proceedings of the 9th International Conference on Education and Information Systems, Technologies and Applications (EISTA 2011), 2011, pp. 17-22.

[46] S. Papert, Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, Inc., 1980.

[47] R. M. Leeman and D. H. Glass, "Teaching Java with robots and artificial life," *Innovation in Teaching and Learning in Information and Computer Sciences*, vol. 6, pp. 23-34, 2007.

[48] D. Buck and D. J. Stucki, "JKarelRobot: a case study in supporting levels of cognitive development in the computer science curriculum," *ACM SIGCSE Bulletin*, vol. 33, pp. 16-20, 2001.

[49] P. Brusilovsky, "Turingal-the language for teaching the principles of programming," in *Proceedings of Third European Logo Conference 1991*, 1991.

[50] M. Kölling, "The greenfoot programming environment," *ACM Transactions on Computing Education (TOCE)*, vol. 10, pp. 14, 2010.

[51] S. Cooper, W. Dann and R. Pausch, "Alice: A 3-D tool for introductory programming concepts," in *Journal of Computing Sciences in Colleges*, 2000, pp. 107-116.

[52] S. Cooper, W. Dann and R. Pausch, "Teaching objects-first in introductory computer science," *ACM SIGCSE Bulletin*, vol. 35, pp. 191-195, 2003.

[53] M. Quinson and G. Oster, "The Programmer's Learning Machine: A Teaching System To Learn Programming," 2014.

[54] S. L. Salcedo and A. M. O. Idrobo, "New tools and methodologies for programming languages learning using the scribbler robot and alice," in *Frontiers in Education Conference (FIE), 2011*, 2011, pp. F4G-1-F4G-6.

[55] M. Jonas, "Teaching introductory progamming using multiplayer board game strategies in Greenfoot," *Journal of Computing Sciences in Colleges*, vol. 28, pp. 19-25, 2013.