UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Revista Científica

http://revistas.udistrital.edu.co/ojs/index.php/revcie/index

CIDC

Artículo de investigación científica y tecnológica

Herramientas usadas para la evaluación formativa automatizada en cursos de programación asistidos por computadora

Tools Used for Automated Formative Assessment in Computer-Assisted Programming Courses

Ferramentas utilizadas para a geração de avaliação formativa automatizada em cursos de programação assistida por computador

Ginna-Viviana Leytón-Yela[1]
Víctor-Andrés Bucheli-Guerrero[2]
Hugo-Armando Ordoñez-Erazo[3]

Recibido: mayo de 2022 **Aceptado:** agosto de 2022

Para citar este artículo: Leytón-Yela, G. V., Bucheli-Guerrero, V. A. y Ordoñez-Erazo, H. A. (2022). Herramientas usadas para la evaluación formativa automatizada en cursos de programación asistidos por computadora. *Revista Científica*, *45*(3), 358-368. https://doi.org/10.14483/23448350.19662

Resumen

Este estudio presenta el despliegue de herramientas para verificación del funcionamiento de la realimentación automatizada implementada en cursos de programación. Los entornos educativos ofrecen la experiencia de evaluación sumativa y formativa de programas informáticos dirigida a estudiantes. En este tipo de herramientas, los estudiantes resuelven una tarea de programación, la cual es validada de manera automática para generar calificaciones y realimentación. Con respecto a la evaluación sumativa, se genera una calificación numérica o porcentual sobre si es correcta o no la solución de una tarea. Para el caso de la evaluación formativa, se genera información sobre los errores o sugerencias a incorporar en los programas con el fin de mejorar el aprendizaje. Las herramientas utilizadas son UNCode, Ask-Elle y Nbgrader. Adicionalmente, se hacen menciones importantes sobre algunas de las herramientas usadas para la comparación de programas y validación de diferencias.

Palabras clave: aprendizaje de programación; curso de programación; evaluación formativa; herramientas de programación; realimentación automatizada.

Abstract

This study presents the use of tools for verifying the operation of automated feedback as implemented in programming courses. Educational environments offer the experience of summative and formative evaluation of computer programs, which is aimed at students. In this type of tool, students solve a programming task, which is automatically validated in order to generate grades and feedback. Regarding

1. Universidad del Valle (Cali, Colombia). Contacto: ginna.leyton@correounivalle.edu.co
2. Universidad del Valle (Cali, Colombia). Contacto: victor.bucheli@correounivalle.edu.co
3. Universidad del Cauca (Popayán, Colombia). Contacto: hugoordonez@unicauca.edu.co

summative evaluation, a numerical or percentage grade is generated on whether the solution of a task is correct. In the case of formative evaluation, information is generated on errors or suggestions to be incorporated to the programs in order to improve learning. The employed tools are UNCode, Ask-Elle, and Nbgrader. In addition, some important remarks are made about some of the tools used for comparing programs and validating differences.
Keywords: automated feedback; formative assessment; learning programming; programming course; programming tools.

Resumo

Atualmente, muitos estudos já estão disponíveis sobre o conhecimento teórico da população sobre as mudanças climáticas. Os resultados de alguns desses estudos mostram a existência de uma séria confusão em relação a certos fenômenos e conceitos básicos de grande importância, apesar de o problema das mudanças climáticas ter adquirido uma relevância crescente em diferentes meios de comunicação nos últimos anos e também estar presente nos currículos escolares como parte fundamental da Educação para a Sustentabilidade. Algumas dessas confusões constituem verdadeiras ideias alternativas. Este artigo analisa, em primeiro lugar, a incidência e a força de cinco dessas ideias em alunos do Ensino Secundário Obrigatório (ESO) e em vários grupos de professores em formação. Em seguida, é feita uma proposta didática para alcançar a mudança de uma dessas ideias, que envolve vários conceitos e princípios da Física.
Palavras-chaves: aprendizagem de programação; avaliação formativa; curso de programação; feedback automatizado; ferramentas de programação.

Introducción

Existen muchas herramientas para el aprendizaje de la programación, que brindan entornos de aprendizaje dinámicos para apoyar a los estudiantes, además de experiencia en los entornos de desarrollo y ejecución de tareas de programación, puesto que es una de las habilidades más demandadas en la actualidad por los estudiantes de ingeniería (McBroom et al., 2020). Muchas de las herramientas se han implementado como apoyo en las clases tradicionales de programación y otras como apoyo y factor clave en el desarrollo de cursos netamente virtuales. En la mayoría de los cursos, en los que se han implementado estas herramientas el aforo se encuentra entre 60 y 90 estudiantes de cursos presenciales asistidos por computadora (Razeeth *et al.*, 2019).

Los estudiantes no solo necesitan la evaluación sumativa de las tareas de programación resueltas, necesitan saber si es posible mejorar la presentación enviada o saber que errores se cometieron en la entrega. A pesar, de que los esfuerzos y experimentos realizados con estas herramientas es eficaz para mejorar las habilidades en programación, se quedan cortos como herramientas de evaluación formativa debido a la limitada realimentación que ofrecen a los estudiantes, puesto que en su mayoría lo que se genera es la calificación a modo numérico o porcentual (Yong and Bedoya, 2022; Asadullah, 2019). En este contexto, se realiza un análisis comparativo de tres herramientas con realimentación automatizada utilizadas en entornos educativos para el aprendizaje de la programación. Las herramientas fueron desplegadas para determinar las características relevantes de estas, principalmente, características referentes a la evolución formativa.

Para el análisis del tipo de realimentación que genera la herramienta, se ha tomado como referencia el estudio (Keuning *et al.*, 2018), donde se especifican cinco tipos de realimentación aplicados en tareas de programación. Estas se han agrupado de acuerdo con el conocimiento sobre restricciones de las tareas (*Knowledge About Task Constraints,* KTC), sobre los conceptos relacionados a las tareas (*Knowledge About Concepts,* KC), los errores encontrados en estas (*Knowledge About Mistakes,* KM) y sobre cómo proceder para solventar un error o mejorar una tarea (*Knowledge About How to Proceed,* KH). Además, de determinar el tipo de realimentación, se explora como se genera la misma, si es configurada por el docente o es soportada mediante técnicas computacionales.

Otros estudios relacionados que han desplegado las herramientas de realimentación automatizada buscan realizar comparativos sobre las características y funcionamiento de estas. Como es el caso de la revisión sistemática de literatura (Keuning *et al.*, 2018), donde se realiza un comparativo de diferentes herramientas para aprender a programar mediante la solución de tareas, con el análisis del contenido de la realimentación, la tecnología usada y la adaptabilidad de la herramienta.

Para el análisis del contenido de la realimentación, se realiza una categorización de los tipos de comentarios, según (Narciss, 2008) se describe una clasificación de componentes de realimentación relacionada con el contenido para entorno de aprendizaje asistidos por computadora. Las etiquetas definidas para clasificar las herramientas y usadas para categorizar el contenido de los comentarios es implementada para clasificar las 101 herramientas seleccionadas en la SLR. Se definen 5 tipos de componentes de realimentación elaborada, conocimiento sobre restricciones (*Knowledge About Task Constraints*, KTC), conocimiento sobre conceptos (*Knowledge About Concepts*, KC), conocimiento sobre errores (*Knowledge About Mistakes*, KM), conocimiento sobre cómo proceder (*Knowledge About How to Proceed*, KH) y conocimiento sobre metacognición (*Knowledge About Meta-cognition*, KMC). Por cada subcategoría se clasifican las herramientas bajo estudio y se obtiene los porcentajes correspondientes a cada caso, analizando que cada una de las herramientas y la forma en cómo se genera el contenido de la realimentación.

Las tecnologías usadas también fueron clasificadas en dos grupos, sistemas de tutoría inteligente (ITS) y técnicas específicas para el dominio de la programación (Keuning *et al.*, 2016; Keuning *et al.*, 2018). Cada una con sus subcategorías. En la primera; se definen las herramientas de rastreo de modelos, basados en restricciones y tutorías basadas en análisis de datos. En la segunda, se definen el análisis de código dinámico mediante pruebas automatizadas, análisis estático básico, transformación de programa, diagnostico basado en intención y las herramientas externas. Como resultado se genera el recuento del número de herramientas, el porcentaje de herramientas que emplean la tecnología y las subcategorías especificas por cada tecnología, de manera general y especifica. Adicionalmente, se encuentra que por tecnología son nombradas las herramientas clasificadas y la razón del porque se encuentran dentro de esa categoría.

Y finalmente, la adaptabilidad donde se permite a los docentes crear ejercicios e influir en la realimentación generada en las herramientas, como son; las plantillas de solución, la soluciones modelo, los datos de prueba y error. Estas clasificaciones se han realizado según los diferentes tipos de información que los docentes pueden proporcionar para alimentar a la herramienta con modelos de error, reglas y restricciones para buscar las soluciones correctas (Keuning *et al.*, 2016). Por ejemplo, herramientas, donde el docente especifica los mensajes de pistas personalizados y controlar que parte del código deben revelarse a los estudiantes por niveles.

En la mayoría de los curso en línea, las clases con gran número de estudiantes son frecuentes, la proporción de estudiantes por docente o tutor no es la mejor, ocasionando que la aplicación de modelos de tutoría sea imposible de implementar, como por ejemplo, el modelo de Oxford (English and English, 2019) que reúne tanto a estudiante como docente de manera semanal para revisar las evaluaciones de tareas, identificando y corrigiendo conceptos errores de la temática evaluada.

Por lo tanto, el tiempo disponible y proporción de tareas que debe revisar un tutor es limitado, lo que significa que el número de tareas puede reducirse a causa de esto, además de que las fechas de entrega de todas las actividades tienen un plazo hasta finalizar el curso, lo que provoca que los estudiantes no cuenten con mucha realimentación sobre su progreso en las tareas y curso en general (Vittorini *et al.*, 2021).

Una de las propuestas para entregar realimentación oportuna a los estudiantes es a través de la automatización de la evaluación sumativa y formativa en los cursos en línea, aunque no es factible para algunas temáticas realizar la evaluación, aunque si se han realizado trabajos preliminares al respecto logrando avances, como por ejemplo Shermis (2014) o Annamaa *et al.* (2017) realizaron investigaciones sobre cómo generar realimentación automatizada en un curso de programación, mediante rúbricas que verifican si un fragmento de código se compila, si un fragmento de código produce las salidas esperadas en el tiempo definido, entre otros.

Por otro lado, se han implementado herramientas (Ramirez-Echeverry *et al.*, 2018; Angelone and Vittorini, 2020; Gerdes *et al.*, 2017) que con ayuda de módulos que le permiten al docente la configuración de casos de prueba, mediante los cuales es posible definir las salidas esperadas en un problema de programación, así como la configuración de comentarios a cada caso de prueba, para generarse a modo de realimentación cuando un caso de prueba no tenga la salida requerida.

Existen otras herramientas que se enfatizan en la generación de pistas para una tarea de programación. Una forma de hacerlo es a través de un módulo que le permita al estudiante solicitar una pista a cambio de una penalidad en la evaluación sumativa de la tarea, como por ejemplo, si se utiliza la escala Likert de 5 puntos, se reduciría la evaluación a 4 puntos, es decir, que por la pista proporcionada se disminuye un punto para la evaluación (Jeuring *et al.*, 2012; Gerdes *et al.*, 2017; Marin *et al.*, 2017).

Las herramientas anteriormente mencionadas se han implementado con diferentes técnicas computacionales, así como también existen herramientas que han incorporado la inteligencia artificial (AI) para la evaluación automatizada en cursos de programación (Srikant and Aggarwal, 2014; Beck *et al.*, 2019; Akram *et al.*, 2020; Dominguez *et al.*, 2010).

A partir de este análisis sobre las plataformas utilizadas para cursos de programación, se obtiene una perspectiva específica que permitió identificar que plataformas han implementaron realimentación automática. De las herramientas que se encuentran en la revisión de literatura, se citan 3 de las herramientas que se desplegaron con el fin de verificar el funcionamiento de la evaluación sumativa y/o formativa en sus entornos.

Herramientas

UNCode

Se han implementado diferentes herramientas y complementos desarrolladas con el fin de generar de manera automatizada tanto evaluación sumativa y formativa, como en el caso de la herramienta UNCode que provee una evaluación sumativa y formativa automatizada a los estudiantes (Ramirez-Echeverry *et al.*, 2018).

En el caso de UNCode, es una plataforma en línea para la práctica y evaluación automática de tareas de programación de computadoras, utilizada en la Universidad Nacional de Colombia, campus Bogotá. Esta herramienta toma como base a INGInious.

INGInious (Université catholique de Louvain, s. f.), es un sistema de calificación automatizada de código, que es usada como apoyo a entornos educativos. La herramienta recibe una tarea con un conjunto de casos de prueba relacionados para validar la misma. Para cada tarea, es posible realizar un número infinito de envíos, hasta que la evaluación sea correcta.

La tarea es ejecutada, y un script configurado al momento de crear la tarea, es el que se encarga de verificar y probar los entregables, determinando si la ejecución y la salida esperada es exitosa o fracaso.

Por otro lado, el archivo de la tarea es ejecutado dentro de un contenedor, con el fin de encarcelar completamente la ejecución del script, haciendo verificaciones del envío, entregando otra evaluación con cuatro estados posibles como éxito, bloqueo, tiempo de espera o error.

Se han realizado varias implementaciones de INGInious, entre ellas la realizada en M-IDEA (Universidad del Valle, s. f.; Hidalgo *et al.*, 2021), que ha desplegado la herramienta para trabajarla desde un sitio web para varios curso de la Universidad del Valle. Se ha configurado todos los complementos por defecto con los que cuenta la herramienta, adicionando complementos nuevos para la evaluación sumativa de varios lenguajes de programación en un curso, así como los módulos de estadísticas de todos los estudiantes, todos los envíos realizados tanto correctos como incorrectos se incluyen en las estadísticas con sus respectivos reportes de metadatos, además de contar con soporte para LMS como Moodle o OpenEdX. Con respecto a la realimentación automatizada no se ha desarrollado un módulo específico, se ha trabajado manualmente por parte de docente mediante la revisión de la evaluación sumativa que si se genera de manera automática.

En el caso de UNCode (Universidad Nacional de Colombia, 2018/2022) se ha desarrollado un conjunto de complementos para incluir otros lenguajes de programación y proporcionar realimentación formativa a los estudiantes. En general, los estudiantes pueden recibir nueve estados posibles de evaluación, aceptado, respuesta incorrecta, error de presentación, error de límite de salida, error de tiempo de ejecución, error de tiempo de ejecución de calificación, límite de tiempo, límite de memoria y error interno.

Se han desarrollado un conjunto de complementos que proporcionan realimentación para aclarar a los estudiantes sobre los temas y de cómo se realizan las tareas, mediante comentarios que permitan comprender que pasos debe seguir para facilitar la realización de esta. La herramienta cuenta con una clasificación de complementos como herramientas de aprendizaje de los estudiantes, herramientas de monitorio y/o seguimiento, y herramientas de realimentación (Ramirez-Echeverry *et al.*, 2018).

Dentro de nuestros intereses, en el momento del despliegue, se revisaron dos módulos que permiten configurar la realimentación por cada tarea de programación. El primer complemento revisado se denomina “Grader”, en el que normalmente se configuran los tiempos para cada caso de prueba, el número máximo de líneas de código, entre otros, campos que evita que el código llene la memoria del entorno del evaluador. En UNCode se añade la sección de casos de prueba, que permite agregar peso a cada caso y generar una calificación final, además de añadir una realimentación personalizada por cada caso de prueba. Si no se realiza correctamente la tarea y no pasa el respectivo caso de prueba configurado, puede decirle al estudiante en que está fallando, esta configuración la realiza el docente. Los casos de prueba fueron cargados con anterioridad en el módulo de archivos de tarea. Este complemento se encuentra activo desde el despliegue de la herramienta (Universidad Nacional de Colombia, 2018/2022).

El segundo complemento corresponde a “Hints”, este no se encuentra activo. El complemento debe añadirse al archivo “configuration.yaml”, como plugin_module para añadir el nombre del plugin a activar. Este complemento se denomina “task_hints” y genera pistas sobre las tareas. Los docentes crean sugerencias para la tarea, que pueden ser desbloqueadas para ayudar al estudiante a resolverla. En la configuración de las sugerencias se puede incluir una penalización que se aplica a la calificación, si los estudiantes solicitan las pistas.

No se utiliza ningún tipo de técnica de inteligencia artificial para la generación de realimentación automatizada, en las herramientas revisadas hasta el momento, la realimentación se realiza manual o la configura el docente desde su perfil para que se genere de forma automática.

NBGrader - WebCAT

Es una herramienta que facilita la creación y calificación de tareas de programación en Jupyter notebook, que permite configurar la calificación rápida de tareas.

Posteriormente a la calificación automática o manual de la tarea, se pueden crear comentarios para los estudiantes, estos pueden ser comentarios tipo realimentación y comentarios de versión. La generación de comentarios crea un archivo en html que es creado por el docente y que al momento del intercambio con nbgrader se realiza una copia de estos archivos (Jupyter Development Team, s. f.).

Los comentarios configurados por el docente pueden presentarse en el administrador de tareas para generar comentarios para todas las presentaciones calificadas o también se puede configurar para un estudiante individual. El docente es quien configura la realimentación con información detallada sobre los errores.

Nbgrader permite creación de tareas definiendo celdas calificadas automática y manualmente en el cuaderno de Jupyter (Manzoor *et al.*, 2020; Jupyter *et al.*, 2019). Para realizar la configuración automática de la evaluación se debe proporcionar el código de la solución y las pruebas unitarias para que las celdas califiquen automáticamente, no se generan automáticamente comentarios, por esta razón se ha buscado la manera de implementar Nbgrader con otros sistemas de calificación automática como Web-CAT que si cuenta con evaluación formativa.

Web-CAT (Edwards, 2014) cuenta con una API que admite los envíos de las tareas de programación, pero para que sea posible su funcionamiento se requiere de tres parámetros con información del envío; identificador del curso en Web-CAT, nombre para la tarea y nombre de la institución educativa que imparte el curso. La API incorpora un botón en el frontend del cuaderno de Jupyter para el envío de la tarea. Posteriormente, Web-CAT entrega la respuesta con una dirección URL en la que se pueden ver la evaluación sumativa y formativa.

Ask-Elle

Ask-Elle (Jeuring *et al.*, 2012) es un tutor inteligente que apoya el desarrollo paso a paso de los programas funcionales simples en lenguaje de programación Haskell, generando comentarios sobre si están o no en el camino correcto, también es posible solicitar pistas cuando lo requiera el estudiante. También es posible que los estudiantes compartan nuevas declaraciones y alternativas en el desarrollo de la tarea.

La clasificación de las tareas se realiza en base al esqueleto del programa u otra información que sugiere la estrategia de solución, pero se permiten variaciones en la implementación con estrategias de solución alternativa. Con respecto a la realimentación generada en las tareas, son calculadas automáticamente a partir de las soluciones y propiedades anotadas y especificadas por el docente para una tarea de programación (Universiteit Utrecht & Open Universiteit, s. f.-b).

Otra forma, de dar realimentación es a través de la generación de pistas mediante la programación de una estrategia, en la que se plasma un procedimiento de cómo resolver un ejercicio con los pasos básicos que debe seguir y combinar para llegar a una solución.

La arquitectura de Ask-Elle es basada en la web, para configurar la aplicación son necesarias las dependencias de Haskell, compilador para Haskell y el paquete ideas desarrollado por los autores (Universiteit Utrecht & Open Universiteit, s. f.-a). Para ello, utilizan un razonador de dominio, el cuál comienza leyendo las soluciones del modelo y la configuración establecida para un ejercicio, posteriormente, la presentación entregada por el estudiante se convierte en una estrategia de programación para el seguimiento de modelos en el que se incluyen las pruebas basadas en propiedades y los scripts que generan los comentarios por cada tarea.

Posteriormente, se complica el programa del estudiante, con el fin de informar errores o advertencias dirigidas a los estudiantes. Para establecer los comentarios a entregar al estudiante, se genera un árbol de sintaxis abstracta (AST) que compara la solución con el programa modelo definido por el estudiante, verificando las propiedades

definidas por el docente, con el fin de buscar contraejemplos o comentarios asociados que permitan al estudiante resolver sus inquietudes (Gerdes *et al.*, 2017).

Resultados

Como resultado de las implementaciones se tiene la comparación del despliegue y exploración de las herramientas anteriormente descritas. Para la comparación de características en la realimentación se usaron las métricas establecidas en el trabajo (Keuning *et al.*, 2018), para las características de la herramienta se usaron las citadas en el trabajo (Ramirez-Echeverry *et al.*, 2018) y adicional a estas, se tienen en cuenta las características de despliegue o puesta en marcha, establecidas en este trabajo. Se han comparado UNCode, Nbgrader con Web-CAT y Ask-Elle. La Tabla 1 presenta esta comparación.

Para la realimentación formativa se tuvieron en cuenta las métricas establecidas por (Narciss, 2008), que especifica tres métricas básicas con las que debe contar la realimentación: conocimiento sobre el desempeño alcanzado para un conjunto de tareas; conocimiento sobre el resultado de una tarea, si pasa todas las pruebas, si es igual al programa modelo o cumple con todas las restricciones y el conocimiento sobre los resultados correctos de una tarea, mediante una descripción o sugerencia de la solución correcta. Igualmente para determinar si las herramientas estudiadas cuentan con las métricas anteriormente mencionadas, se citan las estrategias de realimentación para tareas de programación (Keuning *et al.*, 2018).

La herramienta UNCode cuenta con una evaluación formativa configurable por parte del docente en dos complementos correspondientes a los casos de prueba y las pistas. Por cada tarea se definen los casos de prueba y pistas que quiera el docente configurar, así mismo, en la configuración inicial de los casos de prueba se definen las entradas y salidas por tarea para posteriormente configurar la realimentación por cada uno. El tipo de realimentación aplicado en los casos de prueba es de tipo de conocimiento sobre errores con pruebas de falla y errores de compilación, puesto que para verificar si la tarea es correcta, se evalúa la sintaxis y la salida esperada.

Las pistas son opcionales y por lo tanto pueden configurarse si se requieren con penalidad o sin penalidad. Este tipo de realimentación puede configurarse de manera libre, puesto que puede ser acerca del conocimiento sobre conceptos, como las temáticas vistas en clases a través de ejemplos o los pasos sobre cómo proceder para resolver o mejorar una tarea.

Tabla 1. Comparativo UNCode, Nbgrader (Web-CAT) y Ask-Elle.

	Variables	UNCode	Nbgrader (Web-CAT)	Ask-Elle
Características de Realimentación Formativa	Pruebas de entrada y salida.	X	X	X
	Configuración de realimentación.	X	X	X
	Evaluación formativa de casos de prueba.	X	X	X
	Evaluación formativa mediante pistas.	X		X
	Evaluación formativa mediante ejemplos.	X		X
Características de Despliegue	Dependencias actualizadas.	X	X	X
	Compatibilidad con diferentes S.O.		X	
	Fácil activación de complementos.	X		
	Uso de contenedores.	X	X	
Características de la herramienta	Diferentes lenguajes de programación.	X		
	Herramientas basada en la Web.	X	X	X
	Administración de usuarios, cursos y tareas.	X	X	X

Con respecto al despliegue, UNCode puede usarse como evaluador externo para sistemas de gestión de aprendizaje (LMS); funciona con las dependencias actualizadas, aunque en lo que respecta al sistema operativo se recomienda usar Centos, en nuestro caso fue implementado en Centos 6. No es posible implementar en Windows, solo es posible en Linux (kernel 3.10+). El backend tiene la responsabilidad de crear, administrar y eliminar contenedores para aislar de manera segura la ejecución y calificación del código fuente. En cuanto a los complementos, se pueden añadir de forma simple con el llamado de un método correspondiente al nombre dado al complemento, esto con el fin de extender las características existentes en la herramienta.

Entre las características principales de la herramienta, UNCode puede implementarse en un servidor para la administración de usuarios, cursos y tareas, los usuarios pueden ser administradores de la plataforma, docentes y estudiantes de los cursos. Los cursos se configuran con diferentes lenguajes de programación, incluso se ha expandido a la evaluación automática para cursos de inteligencia artificial con Jupyter notebook y leguajes de descripción de hardware, usado para cursos de electrónica digital.

La herramienta Nbgrader con Web-CAT, permite brindar realimentación inmediata a los estudiantes con sugerencias de cómo resolver una tarea o que pasos se deben seguir para resolverla, además de realimentación referente a los errores de compilación que puedan presentarse. La tarea resuelta en Jupyter es enviada a Web-CAT y tardará un tiempo en ejecutarse para comprobar el código.

Una vez hecho esto, devuelve un enlace que es abierto en una ventana emergente de Jupyter. Esto muestra algunos detalles básicos de la tarea, como el nombre de la tarea, el nombre del estudiante, la fecha de envío y el puntaje total para la tarea. Luego, la puntuación se desglosa un poco más en el cuadro Resumen de puntuación. Esto muestra en qué fase se perdieron los puntos. También habrá una lista de los archivos del envío. Al hacer clic, aparece un visor de código con más detalles sobre cómo Web-CAT calificó la tarea.

En cuanto al despliegue, la última versión de Web-CAT para Jupyter notebook fue actualizada hace tres años. En cuanto a la activación de complementos, no se encuentra una opción que permita incluir otros tipos de realimentación. Jupyter notebook solo permite trabajar con leguaje de programación Python, y es posible trabajarlo tanto de manera local con acceso a internet o en un servidor. Esta herramienta se incluye en Jupyter notebook como complemento. En el caso de otras plataformas, se puede incluir como una aplicación LTI (Learning Tool Interoperability) externa.

Finalmente, Ask-Elle, permite generar realimentación sobre la explicación de la tarea mediante información complementaria para procesar la tarea, asi como pistas que se entregan para continuar con el desarrollo de la tarea y las pistas con ejemplos que permiten medir los efectos del aprendizaje y eficacia en los estudiantes. La herramienta cuenta con facilidad para añadir tareas de programación y programar la realimentación. Los comentarios y sugerencias proporcionados por Ask-Elle se genera automáticamente a partir de las soluciones anotadas y definidas por el docente.

La implementación se realizó en Linux Debian con la instalación de las dependencias e Haskell, GHC y el paquete ideas. Ask-Elle es un entorno de aprendizaje que puede usarse externamente a la aplicación web disponible en (Universiteit Utrecht & Open Universiteit, s. f.-a). En Windows se presentaron inconsistencias en la instalación de dependencias, por la versión y tipo de Windows. Sólo se encuentra habilitado para el lenguaje de programación Ask-Elle.

Como se puede observar, los tipos de realimentación mayormente implementados en las herramientas son sobre el conocimiento de errores y el conocimiento de cómo proceder para resolverlos en una tarea de programación.

En la realimentación sobre conocimiento de errores, los resultados sugieren que este tipo de realimentación es positiva (Funabiki *et al.*, 2016),

indicando que el sistema contribuye al aprendizaje de los estudiantes y evaluación del material académico que los métodos usados tradicionalmente.

Mientras que en otro estudio, Kim *et al*. (2016), se revela que el 78% de las sugerencias son útiles para localizar y comprender errores, es decir, entender los problemas fácilmente; un estudio de usuario aplicado en el aula muestra que este tipo de herramientas mejora sustancialmente la productividad de los estudiantes.

En la realimentación sobre cómo proceder, se presentan experiencias que se llevaron a cabo para recopilar evidencia empírica sobre los efectos beneficiosos de las pistas (Jeuring *et al*., 2012; Ardimento *et al*., 2020). Los resultados obtenidos muestran que el sistema de pistas apoya el proceso de aprendizaje, aumentando su eficacia y evidenciando mejores resultados, en lo que respecta a la corrección del código, puesto que los estudiantes que utilizaron la herramienta obtuvieron calificaciones más altas que las que utilizan la enseñanza tradicional.

En la Figura 1 se puede observar cómo cada una las herramientas estudiadas cuentan con un número de estrategias de realimentación implementadas, se contempla que la herramienta UNCode es una de las que más estrategias ha empleado en la generación de realimentación en tareas de programación con sus módulos de creación de pistas y asignación de realimentación a cada caso de prueba.

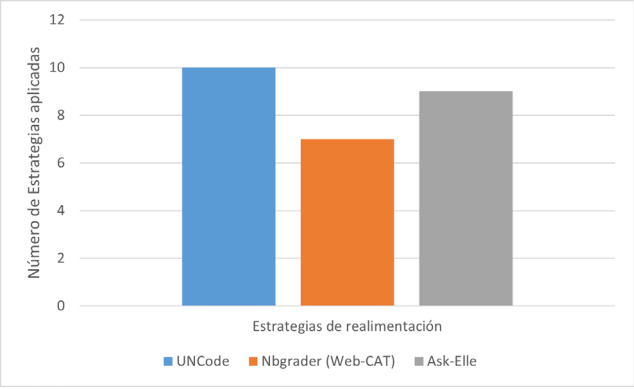

Fig. 1. Estrategias de realimentación empleadas en las herramientas analizadas.

Conclusiones

Se ha realizado un comparativo que permite identificar las variables evaluadas en cada herramienta desplegada, a través de la comprobación de los complementos o módulos de realimentación automatizada en tareas de programación, con el fin de determinar cuál es la herramienta que puede tomarse como base en nuestro estudio. Además, de la identificación de los tipos de realimentación abordados por cada una de las herramientas descritas anteriormente.

Entre los aportes más significativos se encuentran la incorporación de las estrategias de realimentación presentadas por Narciss, puesto que las mismas han sido implementadas y adaptadas en otros estudios sobre realimentación en tareas de programación. Cabe resaltar que estas estrategias permiten identificar y clasificar los tipos de realimentación usados en las herramientas como apoyo a los procesos de aprendizaje, mediante la realimentación formativa como eje fundamental.

Se observa que, en las herramientas desplegadas, los tipos de realimentación más usados son las sugerencias relacionadas con errores, es decir, la identificación de un error específico y la generación de comentarios que permitan guiar al estudiante ante la corrección de una tarea de programación. Los tipos de realimentación que no se han implementado en las herramientas son sobre problemas de rendimiento, comportamiento y problemas de estilo de un programa.

En este orden de ideas la herramienta que cuenta o permite la configuración de varias estrategias de realimentación es UNCode, puesto que cuenta con los módulos de realimentación para cada caso de prueba y las pistas que puede incluir comentarios sobre las restricciones y temáticas que aborda la tarea, asi como los errores de compilación y comentarios sobre cómo proceder para resolver o mejorar una tarea de programación. En trabajos futuros, se pretende incluir un algoritmo de aprendizaje de máquina para el entrenamiento y predicción de la

realimentación de acuerdo con la clasificación de los errores frecuentes cometidos en una tarea de programación.

Referencias

Akram, B., Azizolsoltani, H., Min, W., Wiebe, E., Navied, A., Mott, B. W., Boyer, K., Lester, J. C. (2020). A Data-Driven Approach to Automatically Assessing Concept-Level CS Competencies Based on Student Programs. En *CSEDM@EDM*

Angelone, A. M., Vittorini, P. (2020). The Automated Grading of R Code Snippets: Preliminary Results in a Course of Health Informatics. En *9th International Conference* (pp. 19-27). Springer. https://doi.org/10.1007/978-3-030-23990-9_3

Annamaa, A., Suviste, R., Vene, V. (2017). Comparing different styles of automated feedback for programming exercises. En *Proceedings of the 17th Koli Calling International Conference on Computing Education Research* (pp. 183-184). https://doi.org/10.1145/3141880.3141909

Ardimento, P., Bernardi, M. L., Cimitile, M. (2020). Software Analytics to Support Students in Object-Oriented Programming Tasks: An Empirical Study. *IEEE Access*, *8*, 132171-132187. https://doi.org/10.1109/ACCESS.2020.3010172

Beck, P., Mohammadi-Aragh, M. J., Archibald, C. (2019). An Initial Exploration of Machine Learning Techniques to Classify Source Code Comments in Real-time. En *ASEE Annual Conference & Exposition*

Dominguez, A. K., Yacef, K., Curran, J. (2010). Data mining to generate individualised feedback. En *Proceedings of the 10th international conference on Intelligent Tutoring Systems* (pp. 303-305). https://doi.org/10.1007/978-3-642-13437-1_52

Edwards, S. H. (2014). Work-in-progress: Program grading and feedback generation with Web-CAT. En *Proceedings of the first ACM conference on Learning @ scale conference* (pp. 215-216). https://doi.org/10.1145/2556325.2567888

English, J., English, T. (2019). Combining Summative and Formative Evaluation Using Automated Assessment. *Issues in Informing Science and Information Technology*, *16*, 143-151. https://doi.org/10.28945/4293

Funabiki, N., Mohri, T., Yamaguchi, S. (2016). Toward personalized learning in JPLAS: Generating and scoring functions for debugging questions. En *IEEE 5th Global Conference on Consumer Electronics* (pp. 1-4). https://doi.org/10.1109/GCCE.2016.7800392

Gerdes, A., Heeren, B., Jeuring, J., van Binsbergen, L. T. (2017). Ask-Elle: An Adaptable Programming Tutor for Haskell Giving Automated Feedback. *International Journal of Artificial Intelligence in Education*, *27*(1), 65-100. https://doi.org/10.1007/s40593-015-0080-x

Hidalgo, C., G, V. A. B., Calle, F. R., Osorio, F. A. G. (2021). Estrategia de enseñanza basada en la colaboración y la evaluación automática de código fuente en un curso de programación CS1. *Investigación e Innovación en Ingenierías*, *9*(1), 50-60

Jeuring, J., Gerdes, A., Heeren, B. (2012). Ask-Elle: A Haskell Tutor. En *21st Century Learning for 21st Century Skills* (pp. 453-458). Springer. https://doi.org/10.1007/978-3-642-33263-0_42

Jupyter Development Team. (s. f.). *nbgrader—Nbgrader 0.7.1 documentation*. https://nbgrader.readthedocs.io/en/stable/

Jupyter, P., Blank, D., Bourgin, D., Brown, A., Bussonnier, M., Frederic, J., Granger, B., Griffiths, T., Hamrick, J., Kelley, K., Pacer, M., Page, L., Perez, F., Ragan-Kelley, B., Suchow, J., Willing, C. (2019). nbgrader: A Tool for Creating and Grading Assignments in the Jupyter Notebook. *Journal of Open Source Education*, *2*, e32. https://doi.org/10.21105/jose.00032

Keuning, H., Jeuring, J., Heeren, B. (2016). Towards a Systematic Review of Automated Feedback Generation for Programming Exercises. En *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 41-46). https://doi.org/10.1145/2899415.2899422

Keuning, H., Jeuring, J., Heeren, B. (2018). A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. *ACM Transactions on Computing Education*, *19*(1), 1-43. https://doi.org/10.1145/3231711

Kim, D., Kwon, Y., Liu, P., Kim, I. L., Perry, D. M., Zhang, X., Rodriguez-Rivera, G. (2016). Apex: Automatic programming assignment error explanation. En *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications* (pp. 311-327). https://doi.org/10.1145/2983990.2984031

Manzoor, H., Naik, A., Shaffer, C. A., North, C., Edwards, S. H. (2020). Auto-Grading Jupyter Notebooks. En *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 1139-1144). https://doi.org/10.1145/3328778.3366947

Marin, V. J., Pereira, T., Sridharan, S., Rivero, C. R. (2017). Automated Personalized Feedback in Introductory Java Programming MOOCs. En *IEEE 33rd International Conference on Data Engineering* (pp. 1259-1270). https://doi.org/10.1109/ICDE.2017.169

McBroom, J., Yacef, K., Koprinska, I. (2020). Scalability in Online Computer Programming Education: Automated Techniques for Feedback, Evaluation and Equity. En *Proceedings of the 13th International Conference on Educational Data Mining*. https://educationaldatamining.org/files/conferences/EDM2020/papers/paper_252.pdf

Narciss, S. (2008). *Feedback Strategies for Interactive Learning Tasks* (pp. 125-144)

Ramirez-Echeverry, J. J., Restrepo-Calle, F., González, F. (2018). *Uncode: Interactive System for Learning and Automatic Evaluation of Computer Programming Skills*. https://doi.org/10.21125/edulearn.2018.1632

Razeeth, M., Kariapper, R. K. A. R., Pirapuraj, P., Nafrees, A., Rishan, U. M., Ali, S. (2019). E-learning at home vs traditional learning among higher education students: A survey based analysis. https://www.semanticscholar.org/paper/E-learning-at-home-vs-traditional-learning-among-a-Razeeth-Kariapper/546b671af8f0542edb932a7819b62e80a0c3010b

Shermis, M. D. (2014). State-of-the-art automated essay scoring: Competition, results, and future directions from a United States demonstration. *Assessing Writing*, *20*, 53-76. https://doi.org/10.1016/j.asw.2013.04.001

Srikant, S., Aggarwal, V. (2014). A system to grade computer programming skills using machine learning. En *Proceedings of the 20th International Conference on Knowledge Discovery and Data Mining* (pp. 1887-1896). https://doi.org/10.1145/2623330.2623377

Universidad del Valle. (s. f.). *Course list - INGInious M-iDEA*. http://ingin.ddns.net/courselist

Universidad Nacional de Colombia. (2022). *UNCode* [Python]. UNCode. https://github.com/JuezUN/INGInious

Université catholique de Louvain. (s. f.). *What is INGInious? —INGInious 0.7 documentation*. https://docs.inginious.org/en/v0.7/what_is_inginious.html

Universiteit Utrecht, Open Universiteit. (s. f.-a). *AskElle*. https://ideas.science.uu.nl/AskElle/

Universiteit Utrecht, Open Universiteit. (s. f.-b). *Ideas tutorial*. https://ideas.science.uu.nl/tutorial/

Vittorini, P., Menini, S., Tonelli, S. (2021). An AI-Based System for Formative and Summative Assessment in Data Science Courses. *International Journal of Artificial Intelligence in Education*, *31*(2), 159-185. https://doi.org/10.1007/s40593-020-00230-2

Yong Castillo, E., Bedoya Ortiz, D. H. (2022). *De la educación tradicional a la educación mediada por TIC*. https://pdf4pro.com/amp/view/de-la-educaci-243-n-tradicional-a-la-educaci-243-n-mediada-por-tic-733bf7.html