ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

ANÁLISIS COMPARATIVO ENTRE CALIFICADORES AUTOMÁTICOS APLICADOS A PROGRAMACIÓN

ANÁLISIS COMPARATIVO ENTRE CALIFICADORES AUTOMÁTICOS BASADOS EN ANÁLISIS DE CÓDIGO, APLICADOS A PROGRAMACIÓN

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

JEFFERSON ALEXANDER GUALA FONSECA

jefferson.guala@epn.edu.ec

DIRECTOR: FRANKLIN LEONEL SÁNCHEZ CATOTA

franklin.sanchez@epn.edu.ec

DMQ, febrero 2023

CERTIFICACIONES

Yo, JEFFERSON ALEXANDER GUALA FONSECA declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

JEFFERSON ALEXANDER GUALA FONSECA

Certifico que el presente trabajo de integración curricular fue desarrollado por JEFFERSON ALEXANDER GUALA FONSECA, bajo mi supervisión.

FRANKLIN LEONEL SÁNCHEZ CATOTA

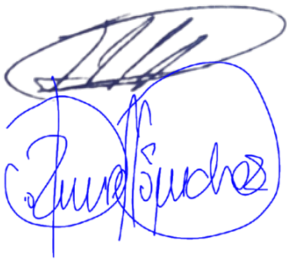
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

JEFFERSON ALEXANDER GUALA FONSECA

FRANKLIN LEONEL SÁNCHEZ CATOTA



DEDICATORIA

Para mis padres, abuelos, hermanos y tíos quienes siempre confiaron en mi a pesar de todo, y me han apoyado desde el principio hasta el final sin escusas.

AGRADECIMIENTO

Agradezco a cada uno de mis compañeros de clase, así como también a los profesores que me han brindado ayuda de una u otra forma para que pueda llegar al final. También agradezco a mi tutor Franklin Sánchez, quien siempre tuvo la disposición y atención para que pueda culminar con éxito el tema de investigación.

ÍNDICE DE CONTENIDO

RESUMEN

El proceso de calificar una tarea de programación en cursos muy grandes ha sido un reto desde hace muchos años atrás. La retroalimentación inmediata a los estudiantes acerca de sus trabajos de programación les ayuda a mejorar sus trabajos y a mejorar el aprendizaje. Sin embargo, en muchas ocasiones los profesores de las universidades desconocen sobre la existencia de herramientas que pueden ayudar a mejorar el proceso de calificación de tareas de programación y al mismo tiempo ayudar a mejorar el aprendizaje de los estudiantes de la clase.

Este documento muestra una revisión sistemática que nos permitirá buscar y analizar las herramientas existentes que ayuden a la calificación de tareas de programación y que dicha calificación se base en el análisis de código fuente.

El primer capítulo describe la fundamentación teórica en la cual se basa el trabajo, es decir muestra una revisión bibliográfica acerca de la revisión sistemática, se revisa lo que es una revisión sistemática, como se lo realiza, se habla acerca de los métodos de búsqueda, así como también de las bases de datos existentes para la búsqueda de documentos en el área de interés. Además, se presentan los objetivos y alcance deseados para el presente trabajo.

El segundo capítulo inicia con el concepto y pasos a seguir para la ejecución de la metodología escogida. Aquí se detallan los pasos seguidos para conseguir un análisis comparativo entre diferentes calificadores automáticos.

Finalmente, el capítulo 3 muestra los resultados obtenidos: un análisis comparativo con las características más relevantes y así como también las ventajas y desventajas de las herramientas dedicadas a la calificación basada en análisis de código fuente. También contiene las conclusiones y recomendaciones para el presente trabajo.

PALABRAS CLAVE: Herramientas de calificación automático, Tareas de programación, Código fuente.

ABSTRACT

The process of grading a programming assignment in very large courses has been a challenge for many years. Immediate feedback to students on their programming assignments helps them improve their work and enhance learning. However, college professors are often unaware of the existence of tools that can help improve the process of grading programming assignments and at the same time help improve student learning in the class.

This work shows a systematic review that will allow us to search for and analyze existing tools that help in the grading of programming assignments and that such grading is based on source code analysis.

The first chapter describes the theoretical foundation on which the work is based it shows a bibliographic review about the systematic review, it reviews what a systematic review is, how it is performed, it talks about the search methods, as well as the existing databases for the search of documents in the area of interest. In addition, the objectives and desired scope of the present work are presented.

The second chapter begins with the concept and steps to be followed for the execution of the chosen methodology. Here the steps followed to achieve a comparative analysis between different automatic raters are detailed.

Finally, chapter 3 shows the results obtained: a comparative analysis with the most relevant characteristics as well as the advantages and disadvantages of the tools dedicated to the qualification based on source code analysis. It also contains the conclusions and recommendations for the present work.

KEYWORDS: Automatic grading tools, Programming Assignments, Source Code.

1 INTRODUCCIÓN

Las herramientas de evaluación automática dedicadas al apoyo en el aprendizaje de programación a los estudiantes y dedicadas a facilitar el proceso de calificación para los profesores se han desarrollado ya desde hace muchos años atrás [1]. Estas herramientas realizan el proceso de calificar automáticamente las tareas de programación que los estudiantes entregan al profesor y que en cursos donde existen una gran cantidad de estudiantes realizar esta calificación de manera manual puede resultar una tarea muy demorada.

Estas herramientas de calificación automática ayudan tanto a los estudiantes como a los profesores; en el caso de los estudiantes, la evaluación les brinda una guía sobre su proceso de aprendizaje y para el profesor brinda un control sobre el proceso y desempeño del estudiante en el curso, sus funcionalidades incluyen la ejecución de los programas que los estudiantes entregan, realizando un análisis del código fuente contra el referencia dada por el profesor y en otros casos se realiza en base a pruebas unitarias del programa.

La calificación instantánea o inmediata en un curso de programación asegura que los estudiantes se preocupen mucho más por la asignatura y le dediquen más tiempo a practicar lo suficiente para así obtener mejores resultados en sus trabajos [2]. Además, este tipo de calificación implicaría que el profesor pueda aumentar el número de tareas sin que el esfuerzo de calificarlos aumente.

En la actualidad existen muchas herramientas de calificación dedicadas a los ejercicios de programación y muchos de ellos coinciden en características, además la gran mayoría están disponibles en la web, ya sea por interfaz web o disponible para su descarga, sin embargo, es poco conocido o rara vez se usan en los cursos de programación, ya sea por el tema de la aceptación de la herramienta por la universidad o por los estudiantes.

Se han visto otros estudios que han realizado una revisión sobre estas herramientas de calificación automática como en [1],[3],[4],[5]. Sin embargo, estos análisis muestran de manera general las herramientas y no discuten sobre las ventajas y desventajas, es decir no realizan una comparación entre ellas, así como también características importantes que ayuden tanto a los estudiantes como a los profesores para tomar una decisión sobre el uso de alguno de ellos.

Por tanto, en este documento se ha realizado una revisión sistemática (RSL) para recopilar, clasificar y analizar las herramientas dedicadas a la calificación para las tareas

de programación. Con esto se busca tener una visión general acerca del tema de interés para posteriormente realizar una evaluación y discusión de toda la información recopilada.

En el capítulo 2 se describe las preguntas de investigación y los pasos o procesos que se han usado para la selección de los documentos finales para el análisis. En el capítulo 3 se presenta una tabla con el número de herramientas encontradas, también una tabla comparativa entre las herramientas con las características más importantes que se pudieron identificar, así como también una tabla de ventajas y desventajas para cada una de estas herramientas.

1.1 Objetivo general

Realizar el estudio comparativo entre los sistemas de calificadores automáticos basados en análisis de código con el fin de proporcionar un documento de ayuda para considerar su uso.

1.2 Objetivos específicos

1. Realizar el estudio del arte de las herramientas de calificador automático basado en análisis de código que se han desarrollado en los últimos 10 años.
2. Identificar criterios para comparar los diferentes calificadores automáticos.
3. Determinar ventajas y desventajas de los diferentes calificadores automáticos.

1.3 Alcance

Se plantea como alcance del componente, partir de una revisión del estudio del estado del arte, para recopilar información acerca de los calificadores automáticos cuya calificación está basada en el análisis de código, este estudio nos permitirá encontrar soluciones vigentes en los últimos 10 años.

Dentro del grupo de soluciones encontradas, para el método de análisis de código se identificarán criterios que permitan compararlos.

Finalmente se realizará la comparación de manera que se puedan emitir recomendaciones para su aplicación en la enseñanza de programación en la educación superior.

1.4 Marco teórico

1.4.1. Estudio de metodología.

1.4.1.1. Revisión sistemática de la literatura

La revisión sistemática se refiere a una investigación científica o un estudio observacional en donde se hace combinaciones de estudios o métodos que permiten contestar o responder a una pregunta de investigación, es decir este debe seguir un protocolo predefinido de manera correcta y entendible para cuando alguien quiera reproducir el método.

En principio la revisión sistemática se utilizaba como un método preciso para el análisis de los resultados obtenidos por varios estudios primarios, con el fin de agrupar, resumir o acoplar las evidencias y disminuir todos los errores que aleatoriamente se encuentran en una investigación.

Visto de una manera mucho más formal según [6] ,las revisiones sistemáticas simplifican los resultados de investigación primarias ya que este hace uso de métodos que ponen una barrera al sesgo y el error aleatorio. Estos métodos son:

- Realizar la búsqueda sistemática y precisa de los artículos que son considerados como los más relevantes.
- Realizar la selección a través de criterios visibles y reproducibles de artículos que hemos decidido que serán incluidos a la revisión final.
- La agrupación de los datos obtenidos y la respectiva interpretación de los resultados finales.

Es importante no confundir una revisión sistemática con una revisión narrativa, si bien ambos forman parte de investigaciones o análisis que toman como base documentos electrónicos y referencias bibliográficas para llegar a obtener resultados que posiblemente otros autores ya lo han hecho, esto con el único objetivo de poder dar fundamento a un objetivo de análisis con respecto al tema de interés[7].

En una revisión narrativa se realiza una investigación amplia para poder describir y hablar sobre el estado de arte de un determinado tema, y su enfoque se basa en lo teórico sin la necesidad de detallar o informar acerca de las fuentes usadas para dicha investigación, ni su metodología o estrategia de búsqueda de información, ni los criterios usados para la selección final de los documentos [7]. Mientras que en una revisión sistemática, si bien también se realiza análisis de estudios que ya se han publicado, este

por el contrario obligatoriamente debe de contener una introducción, un desarrollo y con sus respectivas referencias, todo esto con el único objetivo de poder responder a una pregunta especifica de investigación la cual debe ser formulada con claridad y que además se usan métodos y criterios explícitos y sistemáticos para la recolección, selección y análisis de los documentos que serán incluidos en la revisión sistemática [6].

En la Tabla 1 se muestra estas diferencias de una forma mucho más clara:

Tabla 1 Diferencias entre revisión narrativa y revisión sistemática

	Revisión narrativa	**Revisión sistemática**
Revisión	General/ Amplio	Especifico
Fuentes/ Documentos	Es posible que no se especifique	Necesariamente se debe especificar las fuentes o referencias bibliográficas usadas
Búsqueda de documentos	Es posible que no se especifique	Se debe de especificar una estrategia de búsqueda
Selección de documentos	Es posible que no se especifique	Se deben de definir criterios explícitos y sistemáticos para la elección de documentos
Análisis de resultados	Variable	Se analiza de forma crítica y se busca responder a la pregunta de investigación planteada para la pregunta de investigación.

Además, dentro de la revisión sistemática encontramos dos formas las cuales son: Cuantitativa o metaanálisis y cualitativa u overview.

Según [8], un metaanálisis es una revisión sistemática cuantitativa que usa métodos estadísticos, así de esta manera le permite hacer un estudio comparativo de algunos similares que responden a preguntas sin respuesta, y por el contrario cuando no se hace uso o no se combina la estadística para el estudio se denomina una revisión sistemática cualitativa.

Muy a menudo se confunde una revisión sistemática con un metaanálisis, sin embargo, hay que tener en cuenta que no siempre es posible realizar un metaanálisis mientras que siempre va a ser posible realizar una revisión sistemática.

Según [8], un metaanálisis se lo puede definir como un método de análisis estadístico de una gama de resultados, análisis e investigaciones individuales con el único propósito de acoplar o incluir los nuevos hallazgos, o a su vez, son datos que fueron recolectados mediante una revisión sistemática haciendo uso de métodos o técnicas estadísticas [9], esto debido a que toda la información recolectada de un tema en específico se sintetiza en un valor numérico.

Entonces, una revisión sistemática es necesaria cuando tenemos una pregunta puntual por responder (pregunta de investigación) y varios artículos o estudios primarios con el fin de aclarar una incertidumbre o como ya se ha dicho antes, responder la o las preguntas de investigación, y no es necesaria cuando: se está también haciendo uso de un metaanálisis para aprobar un tema nuevo, es decir hacer que con varios estudios pequeños relacionados a este tema querer o pretender formar un estudio nuevo, algo significativo con algo pequeño o con artículos de mala calidad, también si el único objetivo que buscamos es añadir un artículo más a la lista de publicaciones sin antes haber tenido una necesidad y pregunta de investigación clara por responder.

1.4.1.1.1. Ventajas:

La ventaja más importante de la revisión sistemática es que permite acoplar o integrar los resultados obtenidos de otros estudios que poseen la misma pregunta de investigación principal a responderse, esto permite que se obtenga conceptos con una mayor precisión, dándonos la posibilidad de intuir patrones y tendencias en los resultados para futuros estudios, todo esto genera como resultado nuevos temas de estudio y preguntas de investigación, además el uso de la revisión sistemática es de vital importancia para solucionar la no concordancia entre estudios personales.

El metaanálisis nos permite obtener o crear información mucho más precisa y por ende de mucho más valor estadístico, con mejor puntaje de evidencia; ayuda a sesgar de mejor manera la selección de estudios similares con respecto al tema del cual se realiza dicha investigación.

1.4.1.1.2. Metodología:

Según [8] , para realizar una revisión sistemática se requiere de las siguientes etapas:

- **Definir una pregunta de investigación** Este es el primer paso para poder realizar una revisión sistemática, aquí debemos identificar y convertir el problema en una pregunta bien estructurada que se pueda responder, y para crear la pregunta podemos seguir un protocolo ya estructurado (población, intervención, comparación y desenlace o resultado), ya que una pregunta bien estructurada servirá como una justificación para poder desarrollar una revisión sistemática incluso esta pregunta servirá de guía para todo el proceso de revisión.

  Es importante notar que esta pregunta de investigación nace por la necesidad de investigación de un tema en un área en específico.

- **Identificación:** En esta etapa se decide sobre las estrategias de búsqueda de los documentos, como por ejemplo en relación del idioma de publicación, las fuentes en donde se encuentran, títulos y resúmenes con todas las combinaciones booleanas posibles, con el fin de que el investigador recopile una cantidad de artículos en relación con lo mencionado en la etapa de la definición de la pregunta.

  Es importante no tener errores en la búsqueda de documentos, ya que cometer errores en la obtención de datos puede dar un resultado erróneo e invalidar el análisis. El objetivo es obtener la mayor cantidad de archivos o estudios que nos sean posibles, de esta forma se obtendrá más información y evitaremos cometer errores u obtener información errónea.

  Es importante emplear las búsquedas en fuentes especificas del área de investigación, pero tampoco podemos excluir a las fuentes abiertas como Google Scholar. También es importante decidir el idioma de la búsqueda, y por razones prácticas se prefiere el idioma inglés.

- **Selección de artículos potenciales:** En esta etapa se aplican los criterios de inclusión y exclusión a todos los títulos y resúmenes que hemos obtenido, evaluar la similitud o concordancia en la selección de los documentos, esto con el fin de reducir al máximo los artículos obtenidos en la primera fase para quedarnos con aquellos que realmente aporten a la investigación.

  En los criterios de inclusión podemos tener los siguientes [8]:

  - Términos de búsqueda
  - Rango de tiempo y,
  - tiempo de publicación.

- **Elección:** En esta etapa se aplica la exclusión a los artículos que previamente ya hemos filtrado, esto con el fin de menorar los documentos que nos puedan causar sesgos en los resultados finales. También a esta etapa se la conoce como la evaluación de calidad de los métodos de estudio con los siguientes criterios: Prueba de diagnóstico y estudios de observación.
- **Inclusión:** En esta etapa se determina los artículos con los que se va a realizar la revisión y que sea posible llevarlo a cabo mediante una síntesis cualitativa o cuantitativa, si es posible se debe adjuntar un gráfico que muestre los documentos seleccionados con el total (en números) de cada fase, así como también el número de los documentos que se han excluido en la fase de elección.

En la Figura 1 se muestra un ejemplo de la forma o método de selección de artículos:

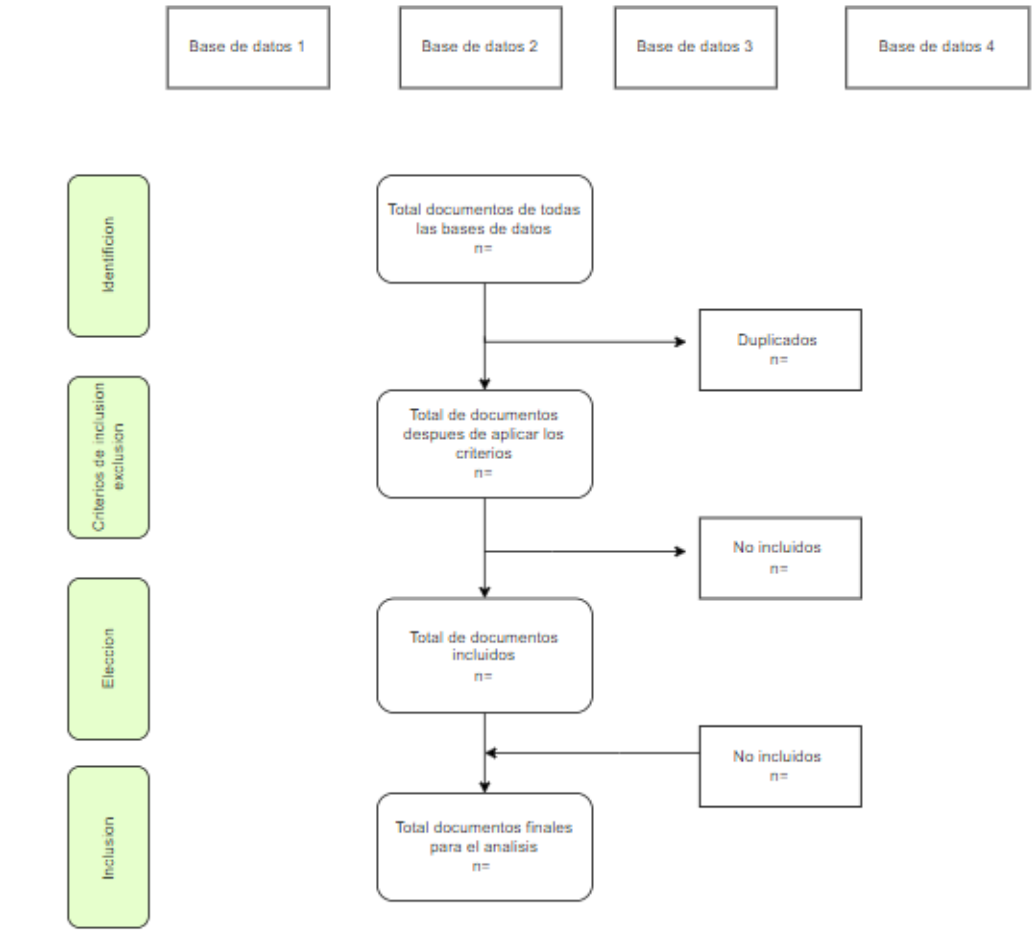


Figura 1 Método para la selección de artículos

- **Análisis de la información:** Esta etapa se la realiza al finalizar todas las anteriores, aquí se debe anotar las variables y datos que se manejaran en el estudio, así como también el formato de la recolección de datos [9]. Al plantear un análisis estadístico se debe de tener en cuenta los siguientes problemas:
    - Verificar los resultados finales y concluir si es posible o no integrar o combinar.
    - Verificar si se puede integrar en otros estudios.
    - Hacer un seguimiento de los sesgos.
    - Documentar y presentar los resultados obtenidos.

1.4.1.2. Métodos de búsqueda.

En la actualidad el internet se ha convertido en un instrumento muy valioso en donde podemos encontrar todo tipo de información relacionado a nuestras investigaciones, sin embargo, encontrar lo que estamos buscando suele ser una tarea complicada, la parte más difícil para poder empezar dicha investigación siempre va a ser el principio, es decir, plantearse las habilidades pertinentes para poder llevar a cabo una investigación bien realizada.[10]

Por tanto, es indispensable tener cierta habilidad o estrategias de búsqueda de datos específicos en el internet, recuperación de información o a su vez recuperación de datos. Estas habilidades se tratan de la utilización de sistemas en donde podemos realizar dicha recuperación, lenguaje y estructuras con semánticas acertadas.[11]

La estrategia más importante es la interrogación, esta trata de introducir en el buscador de internet una serie de palabras específicas, además es importante tener ya planteada la pregunta principal que se mencionó en el apartado del estudio de metodología, para realizar una búsqueda efectiva se suele hacer uso de los operadores de búsqueda, los más relevantes son:

1.4.1.2.1. Operadores lógicos.

Este tipo de operadores reemplazan a las preguntas que normalmente vemos en los buscadores como por ejemplo “incluir (AND)”,” incluir alguno (OR)” o “Excluir (NOT)” y se utiliza de la siguiente manera [12]:

- Operador AND: “+” o “&” estos son los operadores que se utilizan en el buscador, estos indican que nuestra búsqueda debe contener de manera estricta las palabras

que están luego de estos operadores como, por ejemplo: “Calificadores” & “programación”.

- Operador OR: Este operador indica o devuelve en el resultado de la búsqueda los archivos en donde contenga una de las dos palabras que están presentes con este operador como, por ejemplo: “Programación” OR “Código fuente”.
- Operador NOT:”-” o”!”, estos son los operadores que se suelen utilizar en los buscadores, este en cambio excluye todos los archivos que contengan la palabra especificada.

1.4.1.2.2. Operadores de Truncamiento.

Este tipo de operadores son de mucha utilidad cuando queremos buscar en un documento un término en específico, pero también términos que posean variaciones morfológicas, también podemos entenderlo como un operador para saber cómo se escribe una palabra o buscar todos los términos que tienen una misma raíz, los símbolos que se suelen usar son: ¿, $ y *, este operador se lo puede usar de las siguientes maneras [13]:

- Al final: Por ejemplo: Progra*. El resultado de la búsqueda seria palabras como programación, programar, etc.
- Al principio: Por ejemplo: *der. El resultado de la búsqueda seria palabras como acceder, comprender, etc.

Es importante que el truncamiento se lo realice con no menos de cuatro letras, de esta manera realizaremos una búsqueda más efectiva sin resultados no deseados o no relevantes.

1.4.1.2.3. Operadores de proximidad.

Este tipo de operadores son muy útiles al momento de querer buscar términos de acuerdo con su posición u orden en el documento, es decir permiten especificar una relación de proximidad entre las palabras requeridas, estos son[13]:

- Operador NEAR: “[]”, este es el operador que se suelen usar en el buscador, además este operador se lo usa entre dos palabras y lo que se verá son documentos que contengan las dos palabras, a veces se lo confunde con el “AND” lógico , ya que tanto “NEAR” como “AND” hacen intersecciones con dos palabras, sin embargo NEAR exige que la búsqueda de las palabras se realice con una

separación de 25 a 100 caracteres entre ellas, por ejemplo: Calificadores NEAR Programación.

- Operador ADJ: Este operador se lo usa entre dos palabras y nos devuelve solo los archivos que contengan en un campo solo los dos términos y además que estén juntos en dicho archivo sin importar el orden como, por ejemplo: “Calificadores” ADJ” automáticos”.
- Operador FAR: Este operador es usado entre dos palabras y nos devuelve los documentos que contengan en un campo solo los dos términos especificados, ya que este al igual que el “AND” lógico realiza una intersección, y además este operador exige que la búsqueda de las palabras se las realice entre 25 palabras o mucho más, se podría decir que es el operador contrario a “NEAR” como, por ejemplo: Calificadores FAR Programación.
- Operador SAME: Este operador al igual que los otros se lo usa entre dos palabras y devuelve en la búsqueda documentos en los que las palabras se encuentran localizadas dentro de la misma página es decir pueden estar en cualquier lugar no necesariamente dentro de una misma oración como, por ejemplo:” Calificadores” SAME “Código fuente”.
- Operador BEFORE: Este operador también es similar al operador “AND” lógico, es decir realiza una intersección entre las dos palabras especificadas y devuelve en la búsqueda documentos que contengan las dos palabras respetando el orden como, por ejemplo: Calificadores BEFORE Automáticos, en este ejemplo se hallara las dos palabras que estén o no cercanas, aunque respetando que la palabra “Calificadores” este en primer lugar y posteriormente la palabra “Automáticos”.
- Operador “”: Este operador nos ayuda a especificar palabras o cadenas de palabras precisas entre las comillas, de esta forma podemos identificar las palabras o frases dentro de un documento como, por ejemplo: “Código fuente”.

1.4.1.2.4. Símbolos.

Los más importantes para poder realizar una búsqueda efectiva se listan a continuación:

- Asterisco: “*”, este símbolo es usado para buscar términos exactos como, por ejemplo: Calificador*, el resultado será exactamente la palabra “Calificador” no Calificadores, etc.
- Paréntesis: “()”, este nos permite realizar combinaciones de operadores como si se tratara de una formula como, por ejemplo: (“Calificadores” OR “programación”) –

Lenguajes, en este ejemplo obtendremos búsquedas que contengan calificadores o programación, pero excluyendo la palabra Lenguajes.

- Arroba: “@”, este nos permite buscar las etiquetas que están asociadas a Twitter como, por ejemplo: @Cisco.
- “~”: Este símbolo se lo usa antes de la palabra que vamos a buscar y el resultado será la palabra y también los sinónimos de dicha palabra, y según [12] a veces incluye plural y singular [12].

1.4.1.2.5. Búsqueda por campos.

- Allintitle: Este nos permite buscar los archivos que tienen como titulo la frase que se indica con este operador, además permite ser usado con otros operadores o símbolos, como, por ejemplo: intittle: (automatic grader).
- Inbody: Este nos permite buscar en archivos las palabras que contengan lo indicado en el operador, es importante que dicha palabra este en el cuerpo del archivo o documento, puede parecer que este operador lo vemos por defecto en las búsquedas que realizamos normalmente, sin embargo, usarlo nos ayuda a reducir el número de resultados.
- Allinanchor: Este nos permite buscar en archivos que contengan las palabras especificadas en un Hipervínculo a otros archivos como, por ejemplo: allinanchor: automatic grader.
- Filetype: Este nos permite obtener archivos del tipo que le especificamos como, por ejemplo: .pdf, .doc, .ps, .xls, .ppt, etc.
- Related: Este nos permite encontrar sitios similares al ingresado junto con este operador como, por ejemplo: related: cisco.com y nos devolverá resultados como Microsoft.com o Zyxel networks.com.
- Domain: Este nos permite especificar el dominio que utiliza la página. Resulta útil para especificar la procedencia de los documentos como, por ejemplo: domain: .ec, este procede de Ecuador.
- Image: Este operador nos permite realizar búsquedas de imágenes relacionadas con la palabra especificada después del operador como, por ejemplo: image: programación.
- Url: Este operador nos permite realiza una búsqueda, pero solo en las URL de los archivos como, por ejemplo: url: programming.

1.4.1.2.6. Búsqueda por idioma.

Es ya muy conocido que por ejemplo en Google las búsquedas se realizan por defecto en 45 idiomas, pero si queremos realizar un filtro adicional, relacionado a los resultados de acuerdo con el idioma se lo puede realizar desde la página de búsqueda avanzada.

1.4.1.2.7. Búsqueda en localizaciones específicas.

- Loc: Este nos permite realizar búsquedas específicas de acuerdo con una localización ingresada como, por ejemplo: Universidades loc:Ecuador.

1.4.1.2.8. Combinando los operadores de búsqueda.

Todos estos operadores y símbolos que se han mencionado en los apartados anteriores se los pueden combinar con la única finalidad de hacer una búsqueda más avanzada y por consiguiente obtener un menor número de resultados en nuestras búsquedas.

Para esto solo se necesita combinar lo antes mencionado y como es de suponer se tiene muchas combinaciones posibles como, por ejemplo:

- Intitle:2022" herramientas de programación", con esta combinación se nos mostrara resultados donde el titulo contenga el año 2022 y se hable acerca de "herramientas de programación".

1.4.2. Bases de datos.

El internet además de constituir un canal de comunicación mundial es un repositorio que acumula gran cantidad de datos que no se puede calcular fácilmente, por tanto, el organizar toda esta diversidad de información ha sido un reto desde hace mucho tiempo atrás, y actualmente se lo ha logrado gracias al uso de las diferentes herramientas tecnológicas como lo son los buscadores y directorios [14].

Es importante saber cuál de los espacios web son los más adecuados para realizar la búsqueda efectiva de nuestra investigación, y cuáles de los buscadores nos permitirán encontrar información científica actual y relevante de una manera mucho más sencilla; estos buscadores son los académicos ya que podemos realizar las búsquedas de archivos que están en la web y específicamente podemos encontrar material que es académico es decir tesis, revistas, artículos etc.

Algunos de estos y los más relevantes que nos servirán para nuestra investigación se presentan a continuación [15] ,con un breve resumen a cerca de cada uno para poder tener la información científica:

- ACM: Association for Computing Machinery, este es un repositorio específicamente relacionado a informática y tecnologías de la información y aquí podemos encontrar textos completos de las ultimas publicaciones de ACM, así como de las organizaciones que son miembros de ACM, además de revistas, libros, SIG (Grupos de interés especial) y resúmenes que son específicamente de informática.
- IEEE Xplore: Institute of Electrical and Electronics Engineers, este es un repositorio de investigación académica referente a informática e ingeniería eléctrica y electrónica en donde podemos encontrar cientos de revistas, miles de actas de conferencias, miles de normas técnicas, miles de libros y documentos completos de los que son los más mencionadas o citadas en el mundo entero.
- Scopus: Este es un repositorio en donde podemos encontrar resúmenes y artículos de revistas científicas, este repositorio es parte de Elsevier, que es una compañía que realiza análisis de información de instituciones y profesionales. En esta base de datos podemos encontrar artículos variados para el área de la salud, tecnología, ciencias sociales y artes.
- Google Academic: En esta base de datos muy general en donde podemos encontrar de igual manera artículos, tesis, libros etc. Aquí encontramos todo tipo de trabajos de investigación académica.

Los buscadores mencionados anteriormente según [15], son los más relevantes en donde se podrá encontrar información precisa y relevante para nuestro caso de estudio. Así mismo al momento de elegir un repositorio de datos se debe realizar una evaluación sobre la información encontrada y ver si es adecuada y suficiente para nuestra investigación.

2 METODOLOGÍA

2.1. Estrategia metodológica

Como se mencionó en el capítulo anterior, el uso de la revisión sistemática constituye una herramienta bastante fuerte al querer realizar una investigación sobre otras investigaciones que ya fueron publicadas con anterioridad, por lo que para este análisis se hará uso precisamente de la revisión sistemática como la estrategia de la metodología a usar.

Se realizará la revisión sistemática de los artículos publicados en los últimos 10 años y se tendrá un enfoque cualitativo, de esta manera se podrá mostrar los estudios realizados sobre los calificadores automáticos basados en análisis de código y aplicados a programación más recientes.

El objetivo principal será identificar sistemáticamente y evaluar muchos artículos de la misma naturaleza y con el mismo objetivo. Para conseguir lo anteriormente descrito, según [8], se indica que una revisión sistemática se la debe de llevar a cabo como un estudio de investigación cualquiera, es decir con una parte conceptual, una de metodología y una de publicación de resultados, por lo que necesariamente una revisión sistemática debe cumplir con:

- Plantear una pregunta de investigación.
- Una parte de identificación, en donde se definirá el método de la búsqueda de los artículos.
- Criterios de inclusión y exclusión.
- Elección.
- Inclusión, es decir se define los documentos definitivos para la investigación.
- Resultados.

2.2. Definición de la pregunta de investigación

Como ya se ha dicho, el primer paso es la correcta definición de dicha pregunta. Esta pregunta debe ser estructurada mediante un proceso sistemático y explícito con el fin de cumplir con el objetivo ya mencionado.

En base a lo dicho, se tendrá como referencia las siguientes preguntas de investigación:

¿Qué herramientas de calificación automática aplicadas a programación y basadas en análisis de código, diseñados tanto para profesores como para estudiantes se han desarrollado en los últimos 10 años?

¿Cuáles son las características más importantes para estas herramientas de calificación automática?

Como se ve, las preguntas de investigación cumplen con el siguiente protocolo que describe en [6], el cual es:

Población y lugar de estudio: calificadores automáticos para los profesores o estudiantes en las universidades.

Que se va a estudiar: Herramientas de calificación automática aplicados a programación que son basados en análisis de código con sus características.

Periodo: En los últimos 10 años.

2.3. Identificación

Una vez que se ha planteado la pregunta de investigación de interés se procede a realizar la localización, y selección de los artículos que consideremos los más importantes y acertados para nuestro análisis, es decir en esta fase lo que se pretende es determinar un método o estrategia de búsqueda de los artículos publicados en las bases de datos de interés.

Por lo tanto, es importante conocer o tener habilidades de búsqueda para poder aprovechar de buena manera los buscadores mencionados en el capítulo anterior, así se ha propuesto una estrategia que consiste en formar una cadena de búsqueda con operadores lógicos, de truncamiento, de proximidad, símbolos, palabras sinónimas, palabras en inglés, así como también con operadores de campo, siendo este último el más usado, ya que permite tener una búsqueda más eficiente.

En principio se podría pensar en tener varias cadenas de búsqueda sin embargo, tener varias cadenas de búsqueda resulta tedioso y poco eficiente ya que las búsquedas se realizarán en distintas bases de datos y en muchas ocasiones muestra resultados iguales con cadenas de búsqueda distintas y el propósito es encontrar todos los artículos posibles que consideremos como los más relevantes; por lo que este proceso debe ser realizado de manera ordenada y precisa, para reducir el trabajo de selección de artículos.

Así con la finalidad de tener una sola cadena de búsqueda que sea capaz de encontrar todos los artículos posibles en una sola investigación en bases de datos científicas, fue necesario realizar una breve revisión de las palabras clave usadas en los artículos encontrados después de una búsqueda general, así como también fue de gran importancia buscar los **Thesaurus** de las bases de datos de interés, los Thesaurus son una lista de palabras o términos que representan lo mismo o tienen relaciones

semánticas o genéricas [16], de esta forma se facilita la construcción de una sola cadena de búsqueda con palabras distintas pero sinónimas entre sí; así se ha podido recopilar una lista de términos que son las palabras clave para buscar dentro de un documento, estos términos se listan en la Tabla 2.

Tabla 2 Términos para la búsqueda

Lista de palabras
environment
tool
mechanism
system
automatic grading
automatic assessment
automatically grading
Automated feedback
Assessment Tools
student's programs
programming assignments
source code analyzer

Por lo tanto, con la ayuda de los operadores "OR" y "AND" se ha formado una cadena de búsqueda que nos servirá para todas las bases de datos de interés, esta cadena de búsqueda se muestra a continuación:

(environment OR tool OR mechanism OR system) AND (automatic grading OR automatic assessment OR automatically grading OR Automated feedback OR Assessment Tools) AND (student's programs OR programming assignments) AND (source code analyzer)

Una vez que se ha definido la cadena de búsqueda con las palabras clave, se inició con la identificación de todos los artículos posibles para después realizar la búsqueda. Esto se realizó en las siguientes bases de datos: ACM Digital Library, IEEE Xplore y Scopus, que tras una breve investigación previa, se identificaron como las más apropiadas para el área de estudio, la cual es de ingeniería. Tras la búsqueda se obtuvo una lista de 392 documentos, entre artículos de revista, de congresos, conferencias, informes técnicos, etc. En la Figura 2, se puede ver el número de artículos encontrados en cada base de datos con la cadena de búsqueda propuesta.

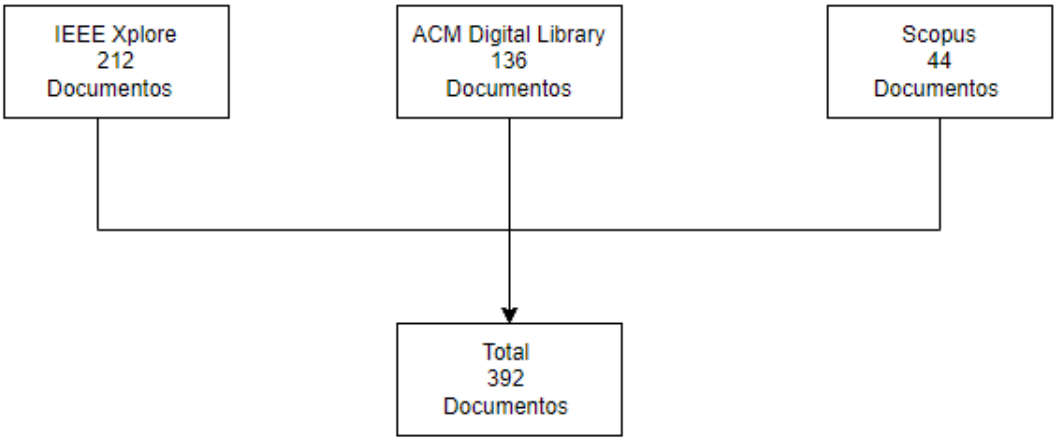


Figura 2 Artículos o documentos hallados

A la lista original se añadieron 5 artículos fuera del rango de año de publicación que se ha planteado, ya que estos artículos los consideramos de una gran importancia para nuestro estudio, en el Anexo 1 podemos ver esta lista en un archivo Excel el cual consta de 392 documentos con el título, autor, año de publicación, resumen y enlace de acceso.

Cabe mencionar que también se ha usado la base de datos de Google Academic, dado que al principio se realizó una búsqueda general en esta base de datos para tener una idea o empezar a familiarizarnos con los términos usados en cada artículo de interés, además es importante mencionar que esta base de datos no se incluye en la Figura 2, ya que aquí no se aplicó la cadena de búsqueda.

En la Figura 3, se muestra un fragmento de este archivo Excel (Base de datos con los documentos recopilados) el cual se ha de utilizar para la extracción de documentos finales.

Document Title	Authors	Publication	Abstract	PDF Link	Document Identifier
2TSW: Automated Assessment of Computer Programming Assignments, in a	G. Polito; M. Temperini; A. S	2019	Automated assessment and feedback of computer programming tasks can be a significant asset in computer science education. Web based systems providing such capabilities are designed to apply techniques ranging from static analysis of program correctness, to testing-based evaluation, and often can have application in frameworks supporting	https://ieeexplore.ieee.org/sta	IEEE Conferences
Automated Programming Assignment Marking Tool	H. Vimalaraj; T. B. K. P. Ther	2022	Due to the enrolment of a very high number of students to programming modules, marking of programming modules is becoming a very tedious and time-consuming process. Programming assignments mainly test for the student's ability to think logically	https://ieeexplore.ieee.org/sta	IEEE Conferences
SALP: A Scalable Autograder System for Learning Programming - A Work in Pr	D. Calderón; E. Petersen; O.	2020	Programming courses can be hard for students, but also for teachers, because of the huge amount of time that takes to manually grade each student's assignment and the different kind of valid solutions.	https://ieeexplore.ieee.org/sta	IEEE Conferences
Enhancing an automated system for assessment of student programs using th	X. T. Yu; C. M. Tang; C. K. Po	2017	It is now common to use automated systems for assessing students' computer programming exercises. Many existing systems determine the correctness of a program by matching its output strings with the ones pre-defined by the instructor. As a result, even when a	https://ieeexplore.ieee.org/sta	IEEE Conferences

Figura 3 Fragmento de la base de extracción de datos

2.4. Criterios de inclusión y exclusión

En esta fase se busca reducir los documentos encontrados en la fase de identificación (búsqueda), con el objetivo de quedarse con aquellos que realmente aporten a la investigación, para esto se debe establecer perspectivas de inclusión conceptuales como, por ejemplo, población, tipo de estudio, idioma de publicación, etc. o perspectivas de inclusión propias del método de búsqueda como, por ejemplo, que aparezcan en el título, en el cuerpo del artículo, rango de tiempo y tipo de publicación.
Por lo tanto, a continuación, se lista los criterios de inclusión para afrontar la pregunta de investigación:

- Se incluyen artículos que en su título se refieren a la calificación automática de tareas de programación basada en el análisis de código.
- Se incluyen artículos que en el resumen hablan de una o más herramientas de calificación automática.
- Se incluyen artículos que describen a una herramienta de calificación y que presentan características importantes acerca de ella.
- Se incluyen artículos que describen la solución que usan las herramientas en el caso de la calificación basado en análisis de código.
- También se considera importante incluir artículos que hablan de herramientas que realizan la calificación basada en análisis de código y basado en pruebas unitarias, ya que se considera como una herramienta novedosa.

Así como se definieron criterios de inclusión también se definieron criterios de exclusión con la finalidad de no tener demasiada información que no sea de mucho aporte al momento de realizar el análisis. A continuación, se lista los criterios de exclusión que se plantearon:

- Se excluyen los documentos que hablan de la calificación automática pero no de una herramienta como tal.
- Se excluyen artículos o documentos que hablan de un método distinto de calificación al análisis de código fuente.
- Se excluyen artículos o documentos que fueron publicados muchos años atrás al de la referencia y que no hablan de una herramienta como tal.
- Se excluyen artículos o documentos que hablan de una herramienta, pero no describen características importantes, sino solo una encuesta realizada a estudiantes de una universidad.

La aplicación de estos criterios tanto de inclusión como de exclusión se puede evidenciar en el Anexo 1, en este se puede ver que se han marcado con un visto para cada documento o artículo que cumple con los criterios de inclusión (descritos en columnas

frente a cada título del documento) de esta forma se fue descartando y aceptando documentos para luego poder obtenerlos. También en la Figura 4, podemos ver un fragmento del archivo de Excel en el que se evidencia la aplicación de estos criterios antes mencionados.

	criterios de inclusión														
	En el titulo							En el resumen							
Document Title	Automatic gr	Automatic asse	Automated f	Assessment Tc	source code ana	Programming assig	Palabras clave	que hablen de una herramienta de calific							
2TSW: Automated Assessment of Computer Programming Assignments, in a Gamified Web Based System						✓		✓							
Automated process for assessment of learners programming assignments						✓	✓								
Experience Paper: Search-Based Testing in Automated Driving Control Applications															
A Systematic Literature Review of Assessment Tools for Programming Assignments				✓		✓		✓							
Invited Talk 2: Techniques for Automating Assessment of Parallel Programming Assignments															

Figura 4 Formato de aplicación de los criterios de inclusión

2.5. Elección.

En esta fase el objetivo es identificar y seleccionar aquellos artículos que cumplieron con los criterios de inclusión, en otras palabras, se hace un proceso de exclusión, además se incluye un análisis de la validez de los resultados.

La selección de los artículos se lo realizo en dos fases. En la fase 1, se realizó una recopilación de los títulos, autores, año de publicación, resumen y el enlace de acceso en un archivo Excel (Base de datos con los documentos recopilados) para decidir si se incluía o no dichos artículos, como resultado de esta elección se obtuvo un total de 50 artículos y se excluyó el resto, en la Figura 5, se puede ver un fragmento del archivo de Excel, este archivo de extracción de datos completo se puede encontrar y revisar en el Anexo 1.

Document Title				
	Authors	Publication	Abstract	PDF Link
2TSW: Automated Assessment of Computer Programming Assignments, in a Gamified Web Based System	G. Polito; M. Temperini; A	2019	Automated assessment and feedback of computer programming tasks can be a significant asset in computer science education. Web based systems providing such capabilities are designed to apply techniques ranging from static analysis of program correctness, to testing-based evaluation, and often can have application in frameworks supporting	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8937377
Automated Programming Assignment Marking Tool	H. Vimalaraj; T. B. K. P. T	2022	Due to the enrolment of a very high number of students to programming modules, marking of programming modules is becoming a very tedious and time-consuming process. Programming assignments mainly test for the student's ability to	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9824339
SALP: A Scalable Autograder System for Learning Programming - A Work in Progress	D. Calderón; E. Petersen	2020	assignment and the different kind of valid solutions.	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9280574
Enhancing an automated system for assessment of student programs using the token pattern approach	Y. T. Yu; C. M. Tang; C. K	2017	It is now common to use automated systems for assessing students' computer programming exercises. Many existing systems determine the correctness of a program by matching its output strings with the ones pre-defined by the instructor. As	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8252370
Towards Using Continuous Integration Tools to Teach Programming Courses	E. Kral; P. Capek	2015	This work proposes a study focused on Continuous Integration (CI) tools in teaching programming	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7424225
LUD: An Automatic Scoring and Feedback System for Programming Assignments	M. G. Hahn; S. M. B. Nav	2022	The increase in usage of online learning systems, has caused a renewed interest in using computers to provide more support to the student's learning experiences, allow instructors to focus on activities that require human intervention, and enable courses	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9853763
Dante - Automated Assessments Tool for Students' Programming Assignments	P. Duch; T. Jaworski	2018	This paper presents Danet - an Automated	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8431146
			The widely popular approach for automatic grading in	

Figura 5 Lista de documentos después de aplicar criterios de inclusión a los títulos

En la fase 2, se leyó detenidamente los resúmenes de los artículos en búsqueda de los criterios de inclusión para decidir si se selecciona o no estos documentos. Los artículos que se incluyeron en la selección final son los que se consideraron los más relevantes, es decir que cumplen con responder las preguntas de investigación planteadas para esta investigación. Se seleccionaron 30 artículos de las 3 fuentes ya mencionadas, el documento completo con esta lista se lo puede evidenciar en el anexo 2. También en la Figura 6, se ve un fragmento de esta lista final.

Titulo	AUTORS	Date Added	Publication	ABSTRACT	DOI	URL
Automatic Grading of Programming Assignments: An Approac	X. Liu; S. Wa	15 Aug 2019	2019	consuming and error-prone if done manually. Existing tools generate feedback with failing test cases. However, this method is inefficient and the	10.1109/ICSI	https://ieeexplore.ieee.org/stamp/s
Automated Assessment Tools for grading of programming As	S. Nayak; R.	31 Mar 2022	2022	Programming assignments play a key role in computer science educational curriculum. Universities use substantial techniques to enhance the practical programming skills of students. This	10.1109/ICC	https://ieeexplore.ieee.org/stamp/s
Review of Programming Assignments Automated Assessment	M. Tarek; A.	1 Jun 2022	2022	recent years. Practising programming boosts problem-solving and logic skills. Diverse degrees, including Computer science and Engineering,	10.1109/MIL	https://ieeexplore.ieee.org/stamp/s
End-to-End Automation of Feedback on Student Assembly Pro	Z. Liu; T. Liu;	20 Jan 2022	2021	rapid feedback to students as they learn to write programs in assembly language (LC-3, a RISC-like educational instruction set architecture). At the	10.1109/ASE	https://ieeexplore.ieee.org/stamp/s
A Systematic Literature Review of Assessment Tools for Progr	D. M. Souza;	23 May 2016	2016	The benefits of using assessment tools for programming assignments have been widely discussed in computing education. However, as both researchers and instructors are unaware of the	10.1109/CSE	https://ieeexplore.ieee.org/stamp/s

Figura 6 Fragmento de la lista final de los documentos

2.6. Inclusión.

En esta fase el objetivo es determinar los artículos con los que se realizará el análisis, es decir un resumen de los artículos recuperados en cada fase de la revisión sistemática, incluso los que se seleccionan de manera definitiva, esto se puede ver en la Figura 7.

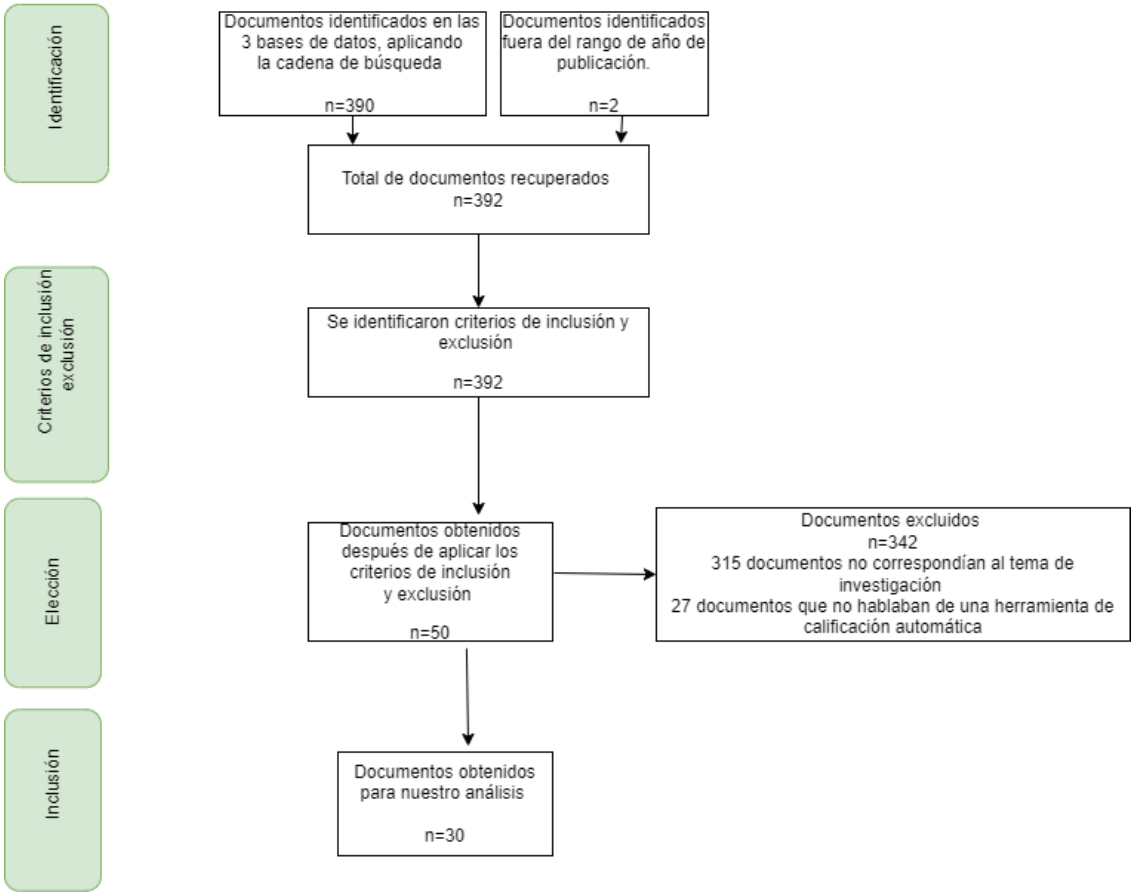


Figura 7 Resumen de artículos recuperados en cada fase de la RS

Como se describe en la Figura 7, después de aplicar todos los criterios de inclusión y exclusión al final se ha obtenido un total de 30 documentos los cuales servirán para poder responder a las preguntas de investigación y poder llegar al objetivo general el cual es realizar el análisis comparativo de las herramientas de calificación automática.

Una vez que se seleccionaron los documentos se empezó a buscar y tratar de obtener los 30 documentos por todos los medios posibles, la página web de Sci-hub fue de gran ayuda ya que pudimos obtener casi todos los documentos de forma gratuita, para los demás fue necesario buscar ayuda externa.

Para ir descartando los documentos obtenidos y los que no, se aumentó una columna al final en el archivo Excel en el cual lo llamamos estado, de si se encuentra disponible o no, es decir de si se pudo obtener estos documentos completos, además marcamos con colores para poder diferenciarlos, de esta manera se pudo tener el control y el orden para obtener todos los documentos de forma eficiente, al final se obtuvieron 20 documentos de los 30, en la Figura 8 se puede ver un fragmento de lo mencionado, así mismo el archivo completo de Excel se puede ver en el anexo 2.

Titulo	AUTORS	Date Added	Publication Y	ABSTRACT	DOI	URL	Estado
Automatic Grading of Programming Assignments: An A	X. Liu; S. Wa	15 Aug 2019	2019	Programming assignment grading can be time-consuming and error-prone if done manually. Existing tools generate feedback with failing test cases. However, this method is inefficient and the	10.1109/ICSE	https://ieeexplore.ieee.org/stamp/stam	SI
Automated Assessment Tools for grading of programn	S. Nayak; R. A	31 Mar 2022	2022	Programming assignments play a key role in computer science educational curriculum. Universities use substantial techniques to enhance the practical programming skills of students. This	10.1109/ICC	https://ieeexplore.ieee.org/stamp/stam	NO
Review of Programming Assignments Automated Ass	M. Tarek; A.	1 Jun 2022	2022	recent years. Practising programming boosts problem-solving and logic skills. Diverse degrees, including Computer science and Engineering,	10.1109/MIU	https://ieeexplore.ieee.org/stamp/stam	NO
End-to-End Automation of Feedback on Student Asse	Z. Liu; T. Liu;	20 Jan 2022	2021	rapid feedback to students as they learn to write programs in assembly language (LC-3, a RISC-like educational instruction set architecture). At the	10.1109/ASE	https://ieeexplore.ieee.org/stamp/stam	NO
A Systematic Literature Review of Assessment Tools f	D. M. Souza;	23 May 2016	2016	The benefits of using assessment tools for programming assignments have been widely discussed in computing education. However, as both researchers and instructors are unaware of	10.1109/CSE	https://ieeexplore.ieee.org/stamp/stam	SI

Figura 8 fragmento del método de obtención de documentos finales

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

En esta fase se responderán a las preguntas de investigación previamente planteadas, en cada caso se realiza un comentario de los resultados así mismo se menciona la utilidad para el docente y estudiante.

Se presenta una tabla con las herramientas identificadas y así mismo una tabla comparativa con la mayoría de las características que fue posible identificar, es importante mencionar que se ha centrado en las características de las herramientas, mas no en las herramientas como tal.

A. ¿Qué herramientas de calificación automática aplicados a programación y que son basados en análisis de código se han desarrollado en los últimos 10 años?

Se ha conseguido identificar 16 herramientas de calificación automática para tareas de programación a través de la revisión sistemática, estas herramientas se listan en la Tabla 3.

Tabla 3 Herramientas de calificación automática

ID	Herramienta	Referencia
H1	**AutoGrader**	[17]
H2	**Autolep**	[18],[1]
H3	**Automark**	[19],[1]
H4	**BOSS**	[20],[1],[21]
H5	**GAME**	[22],[1],[21]
H6	**JARPEB**	[23],[1]
H7	**MARMOSET**	[24],[1]
H8	**ONLINE ASSESSMENT MANAGEMENT SYSTEM**	[25],[1]
H9	**WEB-CAT**	[26],[5],[27],[21]
H10	**Checkstyle**	[4],[28]
H11	**eGrader**	[27]
H12	**CAP**	[29],
H13	**Cppcheck**	[4],[30]
H14	**PMD**	[4],[31]

H15	**OOPGRADER**	[32]
H16	**Style++**	[33]

Las herramientas mencionadas en la Tabla 3 corresponden a herramientas de evaluación automática, aclaramos esto ya que hemos podido identificar herramientas que son semiautomáticas, es decir herramientas que realizan la calificación con la ayuda manual del profesor, esta investigación se deja como un trabajo futuro que se podría incluir a este análisis.

B. ¿Cuáles son las características más importantes para estas herramientas de calificación automática?

Las características más mencionadas que se identificaron se listan a continuación:

- C1: Tecnologías usadas
- C2: Lenguaje de implementación.
- C3: ¿Resistente a diferentes nombres para una sola variable?
- C4: Tiempo de la calificación
- C5: Enfoque
- C6: Retroalimentación inmediata
- C7: Tipo de realimentación
- C8: Almacena historial
- C9: Informe estadístico
- C10: Modo de trabajo
- C11: Lenguajes de programación compatibles
- C12: Condiciones en el Código para la calificación
- C13: Idioma de configuración
- C14: Almacenamiento
- C15: Cantidad de archivos aceptados
- C16: Sugerencia de posibles soluciones
- C17: Técnica de análisis de código

En la Tabla 4 se lista las herramientas con las características mencionadas, además desde ya se aclara que el objetivo de esta tabla es realizar una comparación con dichas herramientas para que esta sirva de guía o ayuda tanto a los profesores como a los estudiantes al momento de querer elegir una herramienta para su implementación en la clase de programación.

Tabla 4 Herramientas y sus características

Herramienta	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**C13**	**C14**	**C15**	**C16**	**C17**
H1	White-box fuzzer Pin	Python	Si	4 a 7 segundos	Estudiante	Si	Incorrecto, correcto.	No especifica	SI	Interfaz de usuario Web (WUI) Interfaz gráfica de usuario (GUI)	Java	Tipo de letra No subprocesos múltiples	Ingles	No especifica	>10000	No	Técnica de análisis semántico con un código de referencia.
H2	No especifica	No especifica	No especifica	No especifica	Estudiante	Si	Mensaje de alerta de un error	No especifica	No	Interfaz gráfica de usuario (GUI)	C++	No especifica	Ingles	Local	No especifica	No	Técnica de análisis estático y testeo dinámico.
H3	No especifica	No especifica	No especifica	No especifica	Estudiante	Si		Si	No	Interfaz de línea de comandos (CLI) Interfaz gráfica de usuario (GUI)	Fortrant77	No especifica	Ingles	Local	No especifica	No	Comprueba que el estilo del código sea diferente, al cargado como referencia.
H4	Java Sherlock(para detector plagio)	Java	No especifica	No especifica	Estudiante Profesor	Si	Mensaje de alerta de un error	No especifica	Si	Interfaz gráfica de	C, C++, java	No especifica	Ingles	Base de datos de la institución.	>5000	SI	Comparación caracter a caracter.

										usuario (GUI) Interfaz de usuario Web (WUI)							
H5	No especifica	No especifica	No especifica	No especifica	Profesor	Si	No especifica	No especifica	Si	Interfaz gráfica de usuario (GUI)	C, C++, java	Se requiere que el evaluador complete un esquema de programación.	Ingles	Local (en un archivo de texto)	>10000	No	Análisis en términos del número de comentarios en el código del estudiante en comparación con el número de líneas del código fuente. Análisis de las prácticas de sangría y su consistencia del código fuente. Estilo de programación. Palabra clave a la salida.

H6	Java's dynamic loading	Java	No especifica	4 segundos	Estudiante	Si	Correcto, incorrecto.	No especifica	Si	Interfaz gráfica de usuario (GUI) Interfaz de usuario Web (WUI)	Java	Se debe de seguir las convenciones de nomenclatura de Java	No especifica	Local	>100	No	No especifica
H7	Apache Tomcat MySQL FindBugs(Herramienta de análisis estático) Clover	No especifica	No especifica	No especifica	Estudiante	Si	No especifica	Si	No	Interfaz de usuario Web (WUI) Se integran en los IDE	C, C++, java, Ruby, Caml	No especifica	No especifica	No especifica	>10000	Si	No especifica
H8	No especifica	No especifica	No especifica	No especifica	Profesor	No	No especifica	No especifica	No	Interfaz de usuario Web (WUI)	Java	No especifica	No especifica	No especifica	No especifica	No	No especifica
H9	Junit Apple's WebObjects framework	Java	No especifica	No especifica	Estudiante, profesor	Si	Salida de casos de prueba.	Si	Si	Interfaz de usuario Web (WUI)	Java, C++, C, Pascal, otros.	No especifica	No especifica	No especifica	Envíos ilimitados	No	Pruebas de análisis estático sobre las soluciones

										Entorno de desarrollo integrado (IDE)							de los estudiantes se realizan usando Checkstyle y PMD.
H10	No especifica	No especifica	Se ajusta a un conjunto de reglas de estilo (El usuario lo puede configurar).	3 a 4 seg	Estudiante, profesor	No	Mensaje de error explicativo	No especifica	No	Se integran en los IDE Se puede usar como una aplicación de línea de comandos	Java	No especifica	No especifica	No especifica	Cientos	Si	No especifica
H11	unit Software Engineering Metrics (SEM)	No especifica	No especifica	No especifica	Estudiante, profesor.	Si	Reporte de los errores cometidos, con una calificación.	Si	Si	Interfaz gráfica de usuario (GUI)	Java	No especifica	No especifica	No especifica	Cientos	No	La lista de soluciones, carpetas y patrones de identificación se cargan en forma de tabla en la macro principal de eGrader.

H12	Applet Java	No especifica	No especifica	No especifica	Estudiante, profesor	No	Lista de errores del programa evaluado. Código de solución y estadísticas sobre los trabajos presentados	No	Si	Interfaz de usuario Web (WUI) Interfaz gráfica de usuario (GUI)	Java, C, C++	No especifica	No especifica	Almacena en MySql (Intranet de la universidad)	Cientos	No	No especifica
H13	SCA GRADER	No especifica	No especifica	4 minutos para 2000 archivos	Estudiante, Profesor	Si	No especifica	No	No	Interfaz de usuario Web (WUI)	C, C++	No especifica	Ingles	SCA Grader DB	>2000	N0	No especifica
H14	Java JRE Open JDK	No especifica	No especifica	0.8 Segundos	Estudiante, Profesor	No	No especifica	Si	Si	Se puede usar como una aplicación de línea de comandos	Java, XML, Apex, otros.	Al menos Java 8 Archivador zip	Ingles	No especifica	No especifica	No	Encuentra fallas de programación comunes, como variables no usadas, catch vacíos
H15	No especifica	DCSL		180 programas en 12 Seg	Estudiante, Profesor	Si	Informe detallado con una calificación.	Si	Si	Se integran en los IDE	Java, C#, DCSL	Se debe cumplir con las reglas de diseño de DCSL.	Inglés	Local	>100	No	No especifica

H16	No especifica	No especifica	No especifica	No especifica	Estudiante, Profesor	Si	Informe detallado con una calificación.	Si	Si	Se puede usar como una aplicación de línea de comandos	C++	No especifica	Ingles	No especifica	Cientos	No	No especifica

Se han identificado un total de 17 características que se ha considerado como las más importantes para que un profesor pueda tomar la decisión de si optar por una de ellas, así mismo con estas se pretende realizar un análisis comparativo con el fin de identificar ventajas y desventajas y así cumplir con el objetivo previamente planteado, dichas características se describen a continuación:

[C1] Tecnologías usadas: En este análisis se pudo identificar que muchas herramientas usaban tecnologías adicionales para complementar su funcionalidad, como por ejemplo de las herramientas que pudimos identificar dichas tecnologías tenemos AutoGrader (H1), esta herramienta usa White-box fuzzer[34] y Pin[35] las cuales ayudan a buscar rutas semánticamente diferentes a las del estudiante y a la de referencia dada por el profesor, el White-box fuzzer genera entradas de pruebas del programa para luego probar el programa con valores incorrectos y de esta forma verifica si la solución mostrada por el estudiante es igual a la del profesor, y en el caso de Pin sin profundizar mucho es una herramienta que sirve para detectar errores.
En el caso de BOSS (H4), es una herramienta que usa las tecnologías de JAVA y Sherlock, esta última es una herramienta que permite detectar plagio de los trabajos entregados por los estudiantes, esta función podría ser muy útil a considerar especialmente para profesores que dictan la materia en más de un curso.

En el caso de MARMOSET (H7), hace uso de las herramientas:

- Apache Tomcat: Es un servidor Web
- MySql: Base de datos.
- FindBug: Esta es una herramienta de análisis estático para encontrar errores en java.
- Clover: Esta es una herramienta que permite recopilar datos de cobertura de un código.

  En el caso de Web-Cat (H9), hace uso de las herramientas:

- Junit: Esta herramienta es útil para realizar pruebas unitarias de aplicaciones en java.
- Apple's WebObjects framework: Servidor para aplicaciones.
- Checkstyle y PMD que son otras herramientas de análisis estático del estilo de la programación.

Las herramientas mencionadas son las que se han identificado que usan otras herramientas novedosas para lograr una calificación más precisa y complementar sus funcionalidades.

[C2] Lenguaje de implementación: Se refiere al lenguaje en el que fueron programadas estas herramientas, se identificó que casi todos fueron desarrollados en Java salvo la herramienta OOPGRADER (H15) el cual ha sido desarrollado en DCSL[36] el cual es un lenguaje de dominio (DSL) que se basa en anotaciones.

Esta característica será importante analizar en caso de que se quiera realizar un aporte a la herramienta, entonces los programadores podrán ver si el lenguaje de implementación es se lo permite.

[C3] Resistente a diferentes nombres para una variable: Esta característica será útil para que los estudiantes no se preocupen por los nombres de las variables a usar, es decir ellos son libres de usar el nombre que creyesen correcto, sin embargo en la documentación encontrada no se encuentra mucha información acerca de esta característica, salvo para la Herramienta Autograder (H1) y Checkstyle (H10); Autograder (H1) menciona que puede calificar sin importar los nombres usados en las variables, mientras que Checkstyle (H10) requiere que se configure un conjunto de reglas de estilo.

[C4] Tiempo de calificación: Se pudo identificar que para una calificación de aproximadamente cientos de archivos los programas tardan entre 0.8 a 4 segundos, esto se puede ver en la Tabla 4 para las herramientas que se han encontrado el dato. Esta característica es de mucha utilidad especialmente para cursos muy grandes en donde se requiera optimizar el tiempo de respuesta a los trabajos de los estudiantes.

Una herramienta que permita calificar los trabajos de los estudiantes en un tiempo casi instantáneo mejorará el desempeño de los estudiantes en sus posteriores entregas ya que sabrá al instante las fallas de su código y podrá rectificarlo.

[C5] Enfoque de la herramienta: Es una característica importante en este análisis y se detectaron tres enfoques los cuales son:

Orientado al profesor: Se refiere a las herramientas de calificación automática en las que el profesor inicia el proceso de calificación, una vez que el o los estudiantes terminen el proceso de entrega el profesor inicia la calificación automática y obtiene los resultados de las entregas para su publicación, estas herramientas son: GAME (H5) y ONLINE ASSESSMENT MANAGEMENT SYSTEM (H8) [1].

Orientado al estudiante: Se refiere a las herramientas de calificación automática en las que el estudiante puede tener la opción de calificar su trabajo, es posible que el profesor defina los parámetros de la tarea a ser calificada, al finalizar la calificación el estudiante puede ver

el resultado al instante, así como también el profesor, estas herramientas son: MARMOSET (H7), JARPEB (H6), Automark (H3), Autolep (H2) y AutoGrader (H1) [1].

Orientado al profesor y el estudiante: Se refiere a las herramientas de calificación automática que hace uso de las estrategias centradas al profesor y de las estrategias centradas al estudiante, estas herramientas son: Style++ (H16), OOPGRADER (H15), PMD (H14), Cppcheck (H13), CAP (H12), eGrader (H11), Checkstyle (H10), WEB-CAT (H9) y BOSS (H4) [1].

Como se ve, son varias las herramientas que están centradas tanto para profesores como para estudiantes, para ser específicos un total de 9 herramientas, esto podría deberse a que estas herramientas tienen mejor tiempo de calificación, y ayuda al aprendizaje del estudiante y acorta tiempos de calificación al profesor.

Por tanto, si el objetivo del profesor es asegurar una respuesta inmediata de calificación a los estudiantes deberían inclinarse a las herramientas orientadas a los estudiantes y si el objetivo es tener un control de los envíos de tareas y de sus calificaciones, debería inclinarse a herramientas orientadas a los profesores, de todas formas, también podría optar por las herramientas que se enfocan tanto para estudiantes como para profesores, ya que estas combinan las ventajas de las otras herramientas.

[C6] Retroalimentación inmediata: Una retroalimentación inmediata es ideal para que el estudiante sepa de los errores cometidos o a su vez si su solución es correcta o incorrecta (Depende de la herramienta) tan pronto cuando entregue su tarea, en la Tabla 4 se puede notar que casi todas las herramientas brindan una retroalimentación inmediata incluso en las que son orientadas a los profesores, así mismo no se pudo encontrar este dato de algunas herramientas sin embargo, no hay duda que estas herramientas también dan una retroalimentación inmediata ya que esta es la principal característica que debe de tener una herramienta de calificación automática según .

[C7] Tipo de retroalimentación: Se refiere a la información que da la herramienta al momento de calificar los programas de los estudiantes, de los que se han podido identificar se clasificaron y se puede ver en la Tabla 5:

Tabla 5 Retroalimentación dada por las herramientas

Tipo de retroalimentación	**Herramientas**
Incorrecto, Correcto	H1, H6
Mensaje de alerta de un error.	H2, H4

Salida de casos de prueba.	H9
Mensaje de error explicativo.	H10
Reporte de los errores cometidos, con una calificación.	H11, H15, H16
Lista de errores del programa evaluado. Código de solución y estadísticas sobre los trabajos presentados	H12

Como se puede ver la mayor parte de las herramientas dan un reporte de errores cometidos con una calificación, este tipo de retroalimentación resulta la más apropiada para los estudiantes, así podrán saber no solo si han fallado o no, si no cuál es su calificación y de esta forma se motivará a mejorar sus trabajos para la próxima entrega.

Así mismo se tiene una serie de opciones por las que un profesor puede orientarse de acuerdo con la dinámica del curso, esto se puede ver en la Tabla 4.

[C8] Almacenamiento de historial: Esta es una característica importante ya que permite a los estudiantes ver su historial de trabajo y así se evita que se cometan errores del pasado, así como también evita que se pierdan trabajos y en el caso de los profesores tienen la información de todos los estudiantes y puede verificar algún tipo de plagio, las herramientas que se han identificado con esta característica son: Automark (H3), Marmoset (H7), WEB-CAT (H9) y eGrader (H11).

[C9] Informe estadístico: Un informe estadístico sirve especialmente a los profesores para que vean el resultado de aprendizaje de los estudiantes y puedan ver las falencias en el curso y puedan tomar decisiones correctas para el curso, las herramientas que se han podido identificar son: AutoGrader (H1), BOSS (H4), GAME (H5), WEB-CAT (H9), eGrader (H11) y OOPGRADER (H15).

[C10] Modo de trabajo: Según el análisis realizado se logró identificar 3 tipos de modos lo cuales son:

- Interfaz de usuario web (WUI): La herramienta está disponible a través de una página web desde cualquier navegador.
- Interfaz gráfica de usuario (GUI): La herramienta está disponible a través de una aplicación de escritorio y el almacenamiento se lo hace localmente.

- Interfaz de línea de comando (CLI): Por lo general las funciones están disponibles en un paquete de programas o scripts que se ejecutan por consola y los resultados de la calificación también se muestran por consola.

En la Tabla 4 se distingue el modo de trabajo para cada herramienta.

[C11] Lenguaje de programación compatible: Según el análisis realizado se nota que los lenguajes de programación más aceptados para la calificación son: Java, C++, C, esto se puede ver en la Tabla 4, aunque evidentemente se nota que aún existe cierta restricción de lenguajes de programación aceptados por las herramientas para su calificación, las herramientas cumplirían perfectamente para un curso introductorio de programación ya que el lenguaje Java, C y C++, C# son los que se dictan en estos cursos.

Además, se pudo encontrar herramientas que aceptaban otros lenguajes no tan comunes como: Fortran77, Ruby, Caml, Pascal, DCSL, así los profesores pueden elegir la herramienta de acuerdo con el lenguaje de programación que se utilizara en el curso. Además, se pone como prioridad investigar a futuro herramientas que no tengan esta restricción de lenguajes de programación.

[C12] Condiciones en el código para calificar: Algunas herramientas como en el caso de GAME (H5) requiere que el profesor llene un esquema de programación, este se almacena en un archivo de texto básico en el cual se detalla el lenguaje de programación admitido, numero de espacios en blanco, palabra clave (El método de calificación es buscar una palabra clave a la salida del programa del estudiante), etc, se puede ver un ejemplo en la Figura 9 [37].

```
@ This is a schema for marking IP1 lab assessment for Test1
@ Values must be comer delimited
@ Language type supported at present C, Cpp, and java
#language_type
java
#path
\projects\marker\testing\p11
#marking_Criteria
keyword
#coutput_mark
5
#compiles_mark
5
@ Structure mark for now just looks at the number of function and
magic numbers
#programming_style_mark
3
@ Indentation mark is made up of two parts
@ 1) the mark alocated for the source code indentation
@ 2) correct indentation size (number of white spaces)
#indentation_mark
3, 4
@ Commenting mark is made up of two parts
@ 1) the mark allocated for source code commenting
@ 2) the percentage level the is considered good commenting
@ This is calculated by (number of comments)/(lines of code)
#commenting_mark
4, 30
#input
stdin
inputTest1.txt
#output
stdout
results.txt
#coutput
value
81
#instructions
```

Figura 9 Esquema de calificación para GAME

Así como esta herramienta se pudo identificar algunas otras que tienen estas condiciones, esto se lo puede ver en la Tabla 4, sin tanto detalle.

[C13] Idioma de configuración: La mayoría de las herramientas vienen con el idioma inglés por defecto, por lo que es necesario que el profesor tome en cuenta esta característica al momento de elegir una.

[C14] Almacenamiento: Muchas herramientas no especifican esta característica, sin embargo, es evidente intuir que las herramientas que tienen una interfaz web los archivos se almacenan en un servidor web o una base de datos de la institución y las que tienen una interfaz de usuario y las que se integran mediante paquetes el almacenamiento se lo hace localmente, esto lo podemos verificar en la Tabla 4.

Además, se notó que estas herramientas no necesariamente requieren de una computadora de altas características para el caso de que se requiera instalar en el PC, así también en el caso de herramientas WEB se puede acceder a través de cualquier navegador que actualmente usamos.

[C15] Cantidad de archivos aceptados: Se ha identificado que todas las herramientas han sido usadas en cursos de programación grandes, de cientos de estudiantes por lo que estas herramientas pueden calificar simultáneamente cientos de trabajos al mismo tiempo y en cursos diferentes, esto no sería un problema para los profesores al momento de elegir una herramienta.

[C16] Sugerencia de posibles soluciones: Si bien la tarea de un calificador automático es comprobar que el programa es correcto o incorrecto y en algunos casos con una puntuación sobre 100, algunas herramientas han dado una funcionalidad más y esta es la de dar una posible solución al error del código, esto beneficia especialmente a los estudiantes ya que además de saber que el código esta errado tienen información importante sobre la solución y claro que esto ayuda a que el aprendizaje del estudiante sea más didáctico.

Las herramientas que brindan esta funcionalidad son: BOSS(H4), MARMOSET (H7), CHECKSTYLE (H10).

[C17] Técnica de análisis de código: Todas las herramientas poseen una técnica de análisis de código, unas más novedosas que otras, en la Tabla 4 podemos ver de alguno de ellos, de los cuales se han encontrado la información, como por ejemplo WEB-CAT (H9), realiza pruebas de análisis estático (Una comprobación de caracteres a caracteres) sobre las soluciones de los estudiantes y se ayuda de otras herramientas de calificación automática como Checkstyle (H10) y PMD (H14).

GAME (H5) realiza un análisis en términos del número de comentarios en el código, comparándolo con el del código fuente, verifica las prácticas de la sangría, estilo de programación y la palabra clave a la salida como habíamos mencionado anteriormente para una herramienta.

En cambio, Automark (H3) basa su técnica en verificar que el código entregado por el estudiante sea diferente al planteado como solución por parte del profesor.

El análisis de código varia en cada herramienta, sin embargo, todos llegan al mismo objetivo el cual es calificar de manera automática los trabajos de los estudiantes, teniendo en cuenta un código fuente o de referencia.

Adicionalmente a las características mencionadas en el análisis realizado, se pudo identificar herramientas que poseían una calificación hibrida es decir su calificación lo realizaba analizando el código fuente y realizando casos de prueba, estas herramientas son Autolep (H2), WEB-CAT (H9).

Esta forma de calificar resulta ser más eficiente ya que primero realiza una ejecución con casos de pruebas y verifica que el programa funcione correctamente y luego con el análisis de código se verifica que la sintaxis y la semántica del código sea como la del código de referencia o la solución dada por el profesor.

Si bien se ha hablado de la gran ayuda que resulta usar estas herramientas y de algunas funcionalidades novedosas que nos brindan, también se considera importante analizar el grado de complejidad que estas herramientas representan para los estudiantes, ya que algunas herramientas son fáciles de usar y otras requieren de más atención en su configuración.

Es importante hablar de cuán difícil seria para un estudiante acoplarse a la metodología del uso de una de estas herramientas, no se pudo encontrar esta información para todas las herramientas salvo de Jarpeb (H6).

Según [38] menciona que se ha probado la herramienta Jarpeb (H6) en un curso de 300 estudiantes de los cuales más del 80% mencionan que al usar esta herramienta al principio se les dificulta, sin embargo, luego de familiarizarse la herramienta resulta ser fácil de usar y la califican como una herramienta muy justa para calificar sus tareas de programación y al mismo tiempo un tanto divertida e interesante.

Por tanto, en base a la información de esta herramienta podemos intuir que la curva de aprendizaje es positiva es decir mientras más se use esta herramienta los estudiantes mejoran sus habilidades de aprendizaje y sus calificaciones son un tanto mejores.

Cada herramienta tiene diferentes características y se ha comparado en la Tabla 4, también cada herramienta sobresale una de la otra ya sea por una funcionalidad novedosa o por su manera de calificar.

En la Tabla 6, se listan las herramientas con su ventajas y desventajas, uno de los objetivos de este análisis es analizar las ventajas y desventajas de cada una de las herramientas, esto ayudara al lector a identificar la mejor herramienta de acorde a sus necesidades o las necesidades del curso.

Tabla 6 Ventajas y desventajas de las herramientas

Herramienta	**Ventajas**	**Desventajas**
H1	Para el caso de la interfaz gráfica de usuario no requiere de una computadora con características especiales.	Solo acepta el lenguaje de programación de JAVA.

H2	Devuelve un informe detallado de los errores sintácticos y semánticos.	Solo acepta lenguaje de programación de C++.
H3	Su enfoque es basado en el estudiante por lo que ayuda a la mejora continua del estudiante.	Solo acepta lenguaje de programación Fortran77.
H4	Contiene herramientas para detectar plagio. Producto de código abierto.	La navegación puede ser un poco confusa.
H5	Fácil de usar. Ayuda al aprendizaje del estudiante.	Para poder calificar se requiere que el evaluador llene un esquema de evaluación.
H6	Contiene herramientas para detectar plagio. También se pude usar sin internet.	La calificación dada solo es correcto e incorrecto.
H7	Recopila datos para mostrar el desarrollo de un estudiante.	Requiere del apoyo de otras herramientas para poder llevar a cabo una calificación.
H8	Permite detectar plagio.	Poca información técnica acerca de la herramienta. Solo acepta el lenguaje de programación Java.
H9	Se puede unir a eclipse usando un plugin. Hace uso de otras herramientas como	Requiere de una previa familiarización con la herramienta.

	Checkstyle y PMD para calificar.	
H10	Permite encontrar problemas de diseño de clases. Problemas de diseño de métodos. Problemas de formato.	Solo acepta lenguaje de programación Java.
H11	Envía un reporte con los errores cometidos con una nota de 0 a 100.	No tiene interfaz web Solo acepta el lenguaje de programación JAVA
H12	CAP ayuda a la corrección del programa de un estudiante. En el informe final CAP muestra la solución verdadera propuesta por el docente.	Sobrecarga de servidor. Una vez que el estudiante ve la solución verdadera del programa la nota no se modificará.
H13	Es de código abierto. Posee Checkstyle con funcionalidades y soporte premium.	No detecta errores de sintaxis.
H14	Es más preciso en el análisis del código, ya que verifica las variables no usadas y bloques catchs vacíos.	No posee interfaz de usuario, solo se lo puede usar mediante líneas de comandos.
H15	Entrega una calificación detallada de cada parte del código. Permite detectar el plagio.	Poco flexible para admitir soluciones diferentes. No es muy específico acerca de los errores.

H16	Aporta con contenido para mejorar el aprendizaje de los estudiantes. Su arquitectura es flexible, se puede añadir más funciones de calificación.	Solo acepta el lenguaje de programación C++.

Tanto en la Tabla 4, y en la Tabla 6 se muestran las características y las ventajas y desventajas respectivamente para cada una de las herramientas, por lo que los profesores podrán ver esta información y podrán tomar una mejor decisión sobre la herramienta compatible para su curso de programación; como ya se ha mencionado, existen herramientas que presentan otro método de calificación por lo que se incentiva a los investigadores analizar este tipo de herramientas.

3.2 Conclusiones

Se ha realizado el estudio y posteriormente se han mostrado los resultados obtenidos después de realizar una revisión sistemática para hallar características, ventajas y desventajas de las diferentes herramientas de calificación automática la calificación de las tareas de programación lo hacen basado en un código fuente. Este análisis permitió ver y entender el estado del arte para las herramientas que actualmente están siendo usadas, así como también ha permitido ver diferentes soluciones disponibles para estas herramientas y futuras investigaciones acerca del modo de calificación. Una vez identificadas las herramientas se pudo realizar un análisis comparativo con un total de 17 características que se pudieron identificar, tal como se ve en la Tabla 4.

Los resultados mostrados después de aplicar la revisión sistemática sugieren que existen muchas herramientas que efectivamente usan un análisis de código fuente para su calificación y muchas de ellas no solo califican las tareas de programación de los estudiantes, sino que también ayudan y guían al estudiante en sus tareas, también el profesor aparte de reducir esfuerzos por calificar tiene la información del progreso de los estudiantes en cada entrega de tareas. Cada herramienta posee ventajas y desventajas, por tanto, el elegir una herramienta de calificación automática para un curso de programación resulta ser una tarea muy importante.

En este aspecto, los resultados mostrados en este documento pueden usarse como guía a los profesores de programación de las instituciones de educación superior para elegir que herramienta es la mejor opción para las necesidades del curso. Por ejemplo, se ha podido identificar herramientas que poseen funcionalidades extras como el de poder detectar plagio en las entregas de los estudiantes como es el caso de BOSS (H4), JARPEB (h6), online assessment management system (h8), OPPGRADER (h15). Estas herramientas además de calificar las tareas de programación en base a un código fuente, también permiten la detección de plagio y de esta manera permite al profesor tener aún más control sobre los trabajos de los estudiantes.

También, se pudieron ver herramientas que usan dos métodos de calificación, es decir además de hacer un análisis de código fuente realizan pruebas unitarias del código, por lo que se destaca la necesidad de una investigación más profunda acerca de estas herramientas.

3.3 Recomendaciones

A partir del análisis realizado es posible realizar algunas recomendaciones sobre como los nuevos investigadores podrían ampliar el análisis a nuevas herramientas con otros nuevos y novedosos métodos de análisis.

Primero se recomienda ampliar la investigación hacia las nuevas herramientas que usan novedosos métodos de calificación, así como también aquellas que usan varios métodos para la calificación.

Ampliar la investigación incluyendo ejemplos de uso para cada herramienta, de esta forma se tendría aún más información para que el profesor pueda decidir por optar o no la herramienta de calificación automática.

Impulsar o promover a los estudiantes a realizar más revisiones sistemáticas sobre estas herramientas, así se obtendrá más información sobre estas o nuevas herramientas aún más novedosas.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] D. M. Souza, K. R. Felizardo, and E. F. Barbosa, “A systematic literature review of assessment tools for programming assignments,” in *Proceedings - 2016 IEEE 29th Conference on Software Engineering Education and Training, CSEEandT 2016*, May 2016, pp. 147–156. doi: 10.1109/CSEET.2016.48.

[2] K. M. Ala-Mutka, “A survey of automated assessment approaches for programming assignments,” *Computer Science Education*, vol. 15, no. 2, pp. 83–102, 2005, doi: 10.1080/08993400500150747.

[3] Petri Ihantola, Tuukka Ahoniemi, and Ville Karavirta, *Review of Recent Systems for Automatic Assessment ofProgramming Assignments*. ACM, 2010.

[4] Susilo Veri Yulianto and Inggriani Liem, *Automatic Grader for Programming Assignment Using Source Code Analyzer*. 2014.

[5] Guerrero M., GuamánD. S., and Caiza J. C., “Revisión de Herramientas de Apoyo en el Proceso de Enseñanza Aprendizaje de Programación,” 2015.

[6] I. Ferreira González, G. Urrútia, and P. Alonso-Coello, “Systematic reviews and meta-analysis: Scientific rationale and interpretation,” *Rev Esp Cardiol*, vol. 64, no. 8, pp. 688–696, 2011, doi: 10.1016/j.recesp.2011.03.029.

[7] Enferm Paul, “RevisonSistematica-RevisionNarrativa,” 2007.

[8] Jaiberth Antonio Cardona AriasLuis Felipe Higuita GutiérrezLeonardo Alberto Ríos OsorioRevisionessistemáticasde, “Revisiones sistemáticas de la literatura científica,” 2016.

[9] Zulma Ortiz, “Curso sobre Revisiones Sistemáticas y Meta-análisis-2005,” 2005. [Online]. Available: www.epidemiologia.anm.edu.ar

[10] Marta Fuentes Agustin, “PONENCIA VIRTUAL EDUCA 2001,” 2001.

[11] M. de La Villa, S. García Pérez, and M. J. Maña, “¿De verdad sabes lo que quieres buscar? Expansión guiada visualmente de la cadena de búsqueda usando ontologías y grafos de conceptos *,” 2011.

[12] J. A. Olivas, “MERINET: Rigurous Methods for the Future Internet View project Graph-based text mining View project,” 2014. [Online]. Available: https://www.researchgate.net/publication/228480257

[13] J. Gómez, “Técnicas para el proceso de búsqueda, acceso y selección de información digital: los operadores,” Aug. 2017.

[14] Quiroz Hugo, Pérez David, and Pérez Maria, “Problemática en el uso de buscadores académicos para la consulta y elaboración de trabajos.,” pp. 1–8, 2018.

[15] Marquina Julián, “16 buscadores académicos que harán que te olvides de Google,” 2016. https://www.julianmarquina.es/16-buscadores-academicos-que-haran-que-te-olvides-de-google/ (accessed Jul. 04, 2022).

[16] Y. Jing and W. B. Croft, “An Association Thesaurus for Information Retrieval.”

[17] X. Liu, S. Wang, P. Wang, and D. Wu, “Automatic grading of programming assignments: An approach based on formal semantics,” in *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training, ICSE-SEET 2019*, May 2019, pp. 126–137. doi: 10.1109/ICSE-SEET.2019.00022.

[18] T. Wang, X. Su, P. Ma, Y. Wang, and K. Wang, “Ability-training-oriented automated assessment in introductory programming course,” *Comput Educ*, vol. 56, no. 1, pp. 220–226, Jan. 2011, doi: 10.1016/j.compedu.2010.08.003.

[19] W. H. Fleming, A. Redish, and W. F. Smyth, “Comparison of manual and automated marking of student programs,” 1988.

[20] M. Joy, N. Griffiths, and R. Boyatt, “The BOSS Online Submission and Assessment System.”

[21] S. Nayak, R. Agarwal, and S. K. Khatri, “Automated Assessment Tools for grading of programming Assignments: A review,” in *2022 International Conference on Computer Communication and Informatics, ICCCI 2022*, 2022. doi: 10.1109/ICCCI54379.2022.9740769.

[22] M. Blumenstein, S. Green, S. Fogelman, A. Nguyen, and V. Muthukkumarasamy, “Performance analysis of GAME: A generic automated marking environment,” *Comput Educ*, vol. 50, no. 4, pp. 1203–1216, May 2008, doi: 10.1016/j.compedu.2006.11.006.

[23] D. Spinellis, P. Zaharias, and A. Vrechopoulos, “Coping with plagiarism and grading load: Randomized programming assignments and reflective grading,” *Computer Applications in Engineering Education*, vol. 15, no. 2, pp. 113–123, Aug. 2007, doi: 10.1002/cae.20096.

[24] J. Spacco *et al.*, “Experiences with Marmoset: Designing and Using an Advanced Submission and Testing System for Programming Courses.”

[25] S.-C. Ng, A. Kwok-Fai Lui, and L.-S. Wong, “Tree-Based Comparison for Plagiarism Detection and Automatic Marking of Programming Assignments,” 2012.

[26] S. H. Edwards, “Teaching Software Testing: Automatic Grading Meets Test-first Coding.” [Online]. Available: http://pmd.sourceforge.net/

[27] M. Tarek, A. Ashraf, M. Heidar, and E. Eliwa, “Review of Programming Assignments Automated Assessment Systems,” in *MIUCC 2022 - 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference*, 2022, pp. 230–237. doi: 10.1109/MIUCC55081.2022.9781736.

[28] CheckStyle, “CheckStyle,” 2022. https://checkstyle.sourceforge.io/ (accessed Jan. 15, 2023).

[29] Óscar Sapena, Mabel Galiano, Marisa Llorens, and Natividad Prieto, *Aprender, enseñar y evaluar con CAP, un Corrector Automático de tareas de Programación*. Universitat Jaume I, 2013. doi: 10.6035/e-tiit.2013.13.

[30] Cppcheck., “Cppcheck.,” 2022. https://cppcheck.sourceforge.io/ (accessed Jan. 15, 2023).

[31] PMD, “PMD,” 2022, Accessed: Jan. 15, 2023. [Online]. Available: http://pmd.sourceforge.net/ snapshot/

[32] D. M. Le, “Model-based automatic grading of object-oriented programming assignments,” *Computer Applications in Engineering Education*, vol. 30, no. 2, pp. 435–457, Mar. 2022, doi: 10.1002/cae.22464.

[33] K. Ala-Mutka, T. U. Solita, and H.-M. Järvinen, “Supporting Students in C++ Programming Courses with Automatic Program Style Assessment Supporting Students on C++ Programming Courses 246,” 2004.

[34] G. F. Vidal Oriola and S. Tamarit Muñoz, “Generación de trazas para la ejecución concólica de programas Erlang Trabajo fin de Máster Tutor,” 2014.

[35] C.-K. Luk Robert Cohn Robert Muth Harish Patil Artur Klauser Geoff Lowney Steven Wallace Vijay Janapa Reddi and K. Hazelwood, “Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation.”

[36] D. M. Le, “Model-based automatic grading of object-oriented programming assignments,” *Computer Applications in Engineering Education*, vol. 30, no. 2, pp. 435–457, Mar. 2022, doi: 10.1002/cae.22464.

[37] M. Blumenstein, S. Green, S. Fogelman, A. Nguyen, and V. Muthukkumarasamy, “Performance analysis of GAME: A generic automated marking environment,” *Comput Educ*, vol. 50, no. 4, pp. 1203–1216, May 2008, doi: 10.1016/j.compedu.2006.11.006.

[38] T. Wang, X. Su, P. Ma, Y. Wang, and K. Wang, “Ability-training-oriented automated assessment in introductory programming course,” *Comput Educ*, vol. 56, no. 1, pp. 220–226, Jan. 2011, doi: 10.1016/j.compedu.2010.08.003.

5 ANEXOS

ANEXO I

https://tinyurl.com/27s2k559

ANEXO II

https://tinyurl.com/yejr4typ