Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Trabajo Final de Maestría en Tecnologías de la Información

Marco de referencia para la evaluación de calidad de herramientas utilizadas en la enseñanza de la programación basada en ISO 25000

Autor: María Cecilia Espíndola

Directora: Mgter. Cristina Greiner

Codirectora: Mgter. Gladys Dapozo

Año 2024

Dedicatoria

A Dios y a mi amada Mater por ser mi fortaleza.

A mis padres Jorge y Ani, por ser el ejemplo a seguir en cada paso que emprendo y así lograr mis metas.

A mi esposo Chumbi, por su apoyo incondicional, comprensión y tolerancia.

Resumen

La enseñanza de la programación en los niveles educativos preuniversitarios ha adquirido una relevancia particular en la actualidad. En Argentina, las políticas públicas están orientadas hacia la introducción de las Ciencias de la Computación en las escuelas, específicamente la enseñanza de la programación, considerada estratégica para el desarrollo nacional. En este contexto, se implementan diversas estrategias para que los estudiantes adquieran habilidades en programación desde una edad temprana, lo cual está asociado al desarrollo del pensamiento computacional. Como parte de las metodologías de enseñanza, los docentes deben decidir qué herramientas o entornos de programación utilizar. En este trabajo se presenta un marco de referencia para evaluar la calidad de las herramientas utilizadas en la enseñanza de la programación, con el fin de proporcionar a los docentes una guía para seleccionar las herramientas más adecuadas para sus objetivos de enseñanza. Para ello, se definió un modelo de calidad, seleccionando características y subcaracterísticas propuestas por el estándar ISO 25010. Se establecieron métricas basadas en el estándar ISO 25023, adaptadas para el propósito de este trabajo, y se llevó a cabo la evaluación siguiendo las etapas establecidas por el estándar ISO 25040. Los resultados obtenidos muestran que las herramientas evaluadas, en general, cumplen con los criterios de calidad definidos en el modelo, aunque varía su grado de cumplimiento en la evaluación de las distintas características. Como contribución de esta propuesta, se ha formalizado un modelo de calidad y su respectiva medición, basado en estándares internacionales, que permitirá su aplicación en la evaluación de la calidad de este tipo de software educativo, que es relativamente reciente en el campo de la educación preuniversitaria.

Palabras claves:

Calidad de producto software, evaluación de calidad de producto, herramientas de software para la enseñanza de la programación

Abstract

Teaching programming at pre-university educational levels has gained particular relevance today. In Argentina, public policies are aimed at introducing Computer Science in schools, specifically programming education, considered strategic for national development. In this context, various strategies are implemented for students to acquire programming skills from an early age, which is associated with the development of computational thinking. As part of teaching methodologies, teachers must decide which programming tools or environments to use. This paper presents a framework for evaluating the quality of tools used in programming education, in order to provide teachers with a guide to select the most suitable tools for their teaching objectives. To do this, a quality model was defined, selecting characteristics and subcharacteristics proposed by the ISO 25010 standard. Metrics based on the ISO 25023 standard were established, adapted for the purpose of this work, and the evaluation was carried out following the stages established by the ISO 25040 standard. The results obtained show that the evaluated tools, in general, meet the quality criteria defined in the model, although their degree of compliance varies in the evaluation of different characteristics. As a contribution of this proposal, a quality model and its respective measurement have been formalized, based on international standards, which will allow its application in the evaluation of the quality of this type of educational software, which is relatively recent in the field of pre-university education.

Keywords: *Software product quality, product quality evaluation, software tools for teaching programming*

Agradecimientos

Quisiera expresar mi más profundo agradecimiento a mis directoras de tesis Mgter. Cristina Greiner y Mgter. Gladys Dapozo, su guía amorosa y paciente ha sido invaluable durante este trayecto y a lo largo de mi formación académica.

Durante este proceso he atravesado por momentos de luces y otros de oscuridad, que los he podido sortear gracias a su guía y a la luz de sus conocimientos.

A mis colegas Ana Company y mi eterna compañera de estudio Verónica Casabonne les agradezco profundamente por su apoyo y aliento.

Por último, a mi casa de estudio, FaCENA, por ser mi refugio durante tantos años de formación y crecimiento tanto personal como profesional.

Índice de contenidos

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción

1. Introducción

La expansión de las TIC (Tecnologías de la Información y la Comunicación) y la interconexión mundial brindan oportunidades para incrementar el progreso, minimizar la brecha digital e impulsar las sociedades del conocimiento. El mundo enfrenta grandes transformaciones producidas por la cultura digital, en la cual tanto el pensamiento computacional como la robótica y la programación tienen un rol fundamental dado que generan nuevos modos de relaciones sociales, construcción de conocimiento y desarrollo de la ciencia. En este sentido, las escuelas deben brindar conocimiento que favorezca la inserción de los estudiantes en la cultura actual y en la sociedad del futuro, promoviendo la integración de saberes emergentes en los procesos de enseñanza y aprendizaje. Diversos países han incluido la programación en sus planes de estudios por su incidencia para el despliegue de habilidades, como el desarrollo del pensamiento lógico, la capacidad de abstracción, la resolución de problemas y el pensamiento creativo, entre otras. Pero, en los últimos años, estos saberes se han convertido en un objeto de estudio en sí mismos debido a su trascendencia y su creciente influencia en la vida cotidiana y en el mundo del trabajo [1].

Dada la importancia que tiene este conocimiento, la forma en que se enseñan los contenidos relacionados con las Ciencias de la Computación o Informática en las escuelas y universidades está siendo discutida estos últimos años, con el objetivo de lograr ciudadanos activos y críticos de este mundo tecnológicamente intensivo [2]. Se busca que los contenidos que se aborden en la formación preuniversitaria provean una visión acertada de las Ciencias de la Computación, fomentando el desarrollo de habilidades abstractas de pensamiento computacional en este proceso y no sólo presentando el uso de tecnologías concretas [3]. Además, se considera importante no solo formar sino también inspirar a los estudiantes, por lo cual el enfoque didáctico debería poner énfasis en elementos clave tales como la motivación, la resolución de problemas del mundo real, el trabajo en equipo y la participación activa de los estudiantes [4]. De aquí que la enseñanza de la programación, en los últimos años, ha incorporado entornos lúdicos y la programación en bloques, para hacer más accesible el aprendizaje desde edades tempranas.

También las universidades, en los últimos años han ido incorporando la programación por bloques como introducción de los conceptos básicos de programación en sus cursos introductorios [5], [6], [7].

Este Trabajo Final de Maestría se ha desarrollado en el marco del proyecto de investigación PI 21F016 "Modelos, metodologías y recursos para el desarrollo del pensamiento computacional", aprobado por SECYT-UNNE, cuyo objetivo general se orienta al estudio y aplicación experimental de estrategias educativas que incorporen métodos y herramientas innovadoras que contribuyan a promover el pensamiento computacional en los procesos formativos de profesionales STEM (Science, Technology, Engineering and Mathematic) en la universidad y contribuir en la formación de docentes de los niveles preuniversitarios para lograr la incorporación de contenidos relacionados con las Ciencias de la Computación en las escuelas. Para el logro de este objetivo, se proponen dos grandes líneas de acción, enfocadas en: La promoción del pensamiento computacional en el nivel universitario, y la formación de formadores de los niveles preuniversitarios.

1.1. Planteamiento del problema

Importancia educativa de la programación y la calidad del software

La tendencia mundial en educación es fomentar el pensamiento computacional y la enseñanza de la programación. Sin embargo, muchos docentes carecen de las habilidades necesarias y de los nuevos enfoques de la didáctica de la programación. Distintos reportes de países del primer mundo [2][4], describen las dificultades de la enseñanza de la computación, entre las que se mencionan:

- La enseñanza de la computación en las escuelas es poco satisfactoria, debido a las siguientes causas: no existen estándares para la enseñanza de la computación; los temas tratados casi siempre se limitan a la enseñanza del uso de utilitarios básicos; hay una preocupante falta de profesores capacitados para enseñar los temas propios de la disciplina; no hay esquemas de educación continua para mantener actualizados a los profesores; la deficiente infraestructura escolar es muchas veces una limitación (falta de laboratorios, de conectividad, de kits educativos, etc.).
- No se reconoce a las Ciencias de la Computación como una disciplina académica rigurosa cuya enseñanza es imprescindible para mejorar las perspectivas profesionales de todos los estudiantes.

Tal como expresa el autor de [8], la enseñanza de la programación se ha transformado en un tema popular en los últimos años, tanto que los gobiernos y empresas invierten en comprar o desarrollar hardware, software y materiales didácticos para la enseñanza de la programación,

sin tener en cuenta su didáctica. “Se dedica mucho esfuerzo en diseñar herramientas, pero mucho menos en evaluarlas de forma rigurosa”.

En este contexto, el presente Trabajo Final de Maestría (TFM) propone un marco de referencia para evaluar la calidad del software que se utiliza para la enseñanza de la programación. Esta propuesta se basa en los conceptos de calidad de productos software y los estándares para su evaluación, perteneciente al campo de la Ingeniería de Software. Estos conceptos y técnicas se aplicarán a los productos software utilizados en la enseñanza inicial de la programación, principalmente en los niveles educativos preuniversitarios, con el propósito de contribuir a la selección de las herramientas, por parte de los docentes, teniendo en cuenta los objetivos que se pretenden alcanzar en su práctica docente.

Al hablar de calidad se hace referencia a las expectativas que los usuarios tienen en relación con una necesidad específica y cómo estas expectativas se ven satisfechas. Estas necesidades suelen estar vinculadas a las características fundamentales de un producto o servicio que son consideradas importantes por los usuarios. Según Juran [9]**,** la calidad es la totalidad de funciones y características de un bien o servicio que atañen a su capacidad para satisfacer necesidades expresas o implícitas. Este autor señaló también que la calidad es “la adecuación para el uso satisfaciendo las necesidades del cliente”.

La creciente presencia del software en la vida cotidiana ha generado una mayor exigencia por altos estándares de calidad que satisfagan las necesidades de los usuarios. En este contexto, la calidad del software se convierte en un aspecto crítico que incide directamente en la experiencia de usuario.

Para evaluar la calidad del software, en este trabajo se tomaron como referencia las normas ISO/IEC 25000, también llamadas SQuare (Requisitos y Evaluación de Calidad de Productos de Software), una familia de normas que tiene por objetivo la creación de un marco de trabajo común para evaluar la calidad del producto software.

1.2. Objetivos del Trabajo Final de Maestría (TFM*)*

Objetivo General

Desarrollar un marco de referencia para evaluar las características de las herramientas o entornos utilizados en el proceso de enseñanza y aprendizaje de la programación inicial, de

acuerdo con las pautas establecidas por el estándar ISO 25000, que contribuya a orientar a los docentes en la selección de las herramientas más adecuadas para incorporar en sus propuestas docentes.

Objetivos específicos

- Profundizar los conceptos sobre la incorporación de las Ciencias de la Computación en las escuelas y sobre los conceptos de calidad de software, en particular, sobre la evaluación de calidad de productos software.
- Relevar y caracterizar las principales herramientas o entornos utilizados en la enseñanza inicial de la programación.
- Diseñar un marco de referencia que permita la evaluación de la calidad de las herramientas o entornos, de acuerdo con el estándar ISO 25000.
- Validar el marco de referencia con las herramientas software más utilizadas en la actualidad en la enseñanza de la programación en los niveles educativos primario y secundario.

1.3. Trabajos previos vinculados con el TFM

1. María C. Espíndola, Cristina L. Greiner, Gladys N. Dapozo. Evaluación de calidad de herramientas utilizadas en la enseñanza de la programación basada en ISO 25000. Libro de actas del XX Workshop de Investigadores en Ciencias de la Computación (WICC 2018). 978-987-3619-27-4. Editorial Eudene. Universidad Nacional del Nordeste. 26 y 27 de abril de 2018. Disponible en: h http://sedici.unlp.edu.ar/handle/10915/68691.
2. Gladys N. Dapozo, Raquel H. Petris, Cristina L. Greiner, Ana M. Company and María C. Espíndola. “Formación docente para incorporar la programación en las escuelas”. En: Marco Galindo, María Jesús; Bañeres Besora, David; Marco Simó, Josep Maria (eds.). Actas de las XXIV Jornadas sobre la Enseñanza Universitaria de la Informática, Barcelona, 4-6 de julio de 2018. Barcelona: Asociación de Enseñantes Universitarios de la Informática, 2018, pp. 31-38. Disponible en: http://rua.ua.es/dspace/handle/10045/125416
3. Gladys Dapozo, Cristina Greiner, Raquel Petris, María Viviana Godoy Guglielmone, María Cecilia Espíndola. “Enseñanza de programación en la universidad. Estrategia basada en programación por bloques”. En Innovation and Practice in Education 2019.

Código de ISBN/ISSN: 978-84-09-09792-0.2019. Editorial CIATA.org. Ciudad Real, España. Junio 2019. Disponible en:

https://www.researchgate.net/publication/334098095_Innovation_and_Practice_in_Education_2019

4. G. Dapozo, Y. Medina, R. Petris, S. Vallejos, M. C. Espíndola, I. Sambrana, M. Burghardt, F. Princich, A. M. "Oferta educativa en programación y robótica para docentes de los niveles preuniversitarios". Revista: Anales SAEI, Simposio Argentino de Educación en Informática 48 JAIIO. Vol. 1. 2019. Disponible en: http://170.210.201.137/pdfs/saei/SAEI-13.pdf
5. G. N. Dapozo, C. L. Greiner, R. H. Petris, Y. Medina, A. M. Company and M. C. Espíndola, "Motivación y logros en la formación de docentes para introducir la programación y la robótica en los niveles educativos no universitarios," 2020 IEEE Congreso Bienal de Argentina (ARGENCON), Resistencia, Argentina, 2020, pp. 1-8, doi: 10.1109/ARGENCON. Disponible en: https://ieeexplore.ieee.org/document/9505572

1.4. Organización del trabajo final

Para cumplir con los objetivos propuestos en el TFM, el contenido está organizado de la siguiente forma:

Capítulo 1- Introducción

En este capítulo se presenta el contexto del cual surge esta propuesta. Por una parte, se detalla el estado de situación de la enseñanza de la programación en el sistema educativo argentino, en particular referido a los niveles educativos preuniversitarios. Destaca luego la importancia de los entornos de programación o herramientas software utilizados en la enseñanza y se plantea la necesidad de evaluar su calidad como producto, en los aspectos técnicos y educativos, a fin de conformar un marco de referencia que le resulte útil a los docentes para facilitar la elección de las herramientas en sus propios procesos de enseñanza.

Capítulo 2- Marco teórico

En el marco teórico se abordan los conceptos sobre los temas relacionados con el trabajo propuesto. Se abordan conceptos de calidad de software, el estándar ISO 25000 y antecedentes del proceso de incorporación de las Ciencias de la Computación en el aula.

Capítulo 3 – Herramientas para la enseñanza de la programación inicial

Este capítulo describe las características de las herramientas software utilizadas para la enseñanza de la programación que fueron seleccionadas para realizar el proceso de evaluación de calidad.

Capítulo 4 – Determinación del Modelo de calidad

En este capítulo, se define el modelo de calidad para la evaluación de las herramientas software de enseñanza de la programación, seleccionando las características y subcaracterísticas que propone el estándar ISO 25010 que mejor se adecuen a los propósitos de la evaluación, así como las métricas seleccionadas del conjunto que ofrece el estándar ISO 25023, en algunos casos adaptadas para cumplir el propósito de esta evaluación.

Capítulo 5- Proceso de evaluación

En este capítulo se detalla el proceso de evaluación, basado en el estándar ISO 25040, implementado para evaluar las herramientas o entornos de programación, de acuerdo al modelo de calidad definido previamente.

Capítulo 6- Conclusiones y líneas de investigación futuras

Se comentan las principales conclusiones y se indican las posibles líneas futuras de investigación.

Anexos

En el apartado de Anexos se incluye material complementario como evidencia del trabajo realizado.

Anexo I: Se detalla el cálculo de las métricas que darán valor a la medición de las características y subcaracterísticas del modelo de calidad definido para la evaluación de las herramientas o entornos de programación utilizados para la enseñanza de la programación en los niveles educativos preuniversitarios.

Anexo II: Se detalla el Reporte de evaluación de calidad de las herramientas de programación que se realiza en la etapa final del proceso de evaluación, basado en la ISO 25040.

Capítulo 2

Marco Teórico

2 Marco Teórico

2.1 Ciencias de la Computación en el aula

En la sociedad contemporánea se observan dos tendencias significativas: la creciente influencia de la tecnología computacional en la vida cotidiana y la aparición de conceptos informáticos en el ámbito educativo. La adquisición de conocimientos en tecnología informática se ha vuelto esencial para el ejercicio pleno de la ciudadanía en el siglo XXI, sin embargo, esta es una deuda pendiente de los sistemas educativos formales [10].

Las Ciencias de la Computación (CC), considerada como la tecnología principal del siglo XXI, tienen un impacto profundo en los aspectos sociales, políticos y económicos. Este impacto se evidencia en la cuarta Revolución Industrial. A pesar de su relevancia, la comprensión de estos procesos y tecnologías es limitada para la mayoría de las personas. La democratización del acceso a este conocimiento y su integración en los sistemas educativos son aspectos clave. Estos pasos son fundamentales para garantizar una ciudadanía informada y capacitada en el uso de la tecnología [10].

La importancia de integrar las Ciencias de la Computación en la escuela

La integración de las CC en la educación es esencial debido a la relevancia de la tecnología digital en la vida moderna. La educación en CC no solo implica el uso de dispositivos tecnológicos, sino también su estudio para su apropiación, valoración, entendimiento, transformación.

La inclusión de CC en la educación obligatoria busca democratizar el acceso a los conocimientos computacionales, reducir la brecha digital y de género, y fomentar vocaciones en TIC. En este contexto, el papel de los sistemas educativos es crucial, ya que la escuela es el principal lugar donde los jóvenes pueden acceder a estos conocimientos y prácticas.

Las políticas públicas

El mapa de políticas educativas con uso intensivo de nuevas tecnologías ha cambiado significativamente en los últimos años. En Argentina, el Programa Conectar Igualdad y Primaria Digital, incorporaron al sistema educativo de gestión estatal una cantidad importante de equipamiento informático, destinado a alumnos y docentes. A su vez, se instaló un piso tecnológico y se proveyeron de servidores y routers a cada escuela [11].

En el año 2015 el Consejo Federal de Educación (CFE), conformado por el ministro de Educación de la Nación y los ministros de Educación de todas las provincias, a través de la Resolución Nº 263/15[1] declaró de importancia estratégica para el sistema educativo argentino la enseñanza y el aprendizaje de la programación durante la escolaridad obligatoria, para fortalecer el desarrollo económico-social de la Nación.

El Plan Estratégico Nacional 2016-2021 ARGENTINA Enseña y Aprende [12] considera que "resulta imperioso fortalecer la enseñanza y el aprendizaje de los conocimientos fundamentales de la matemática, la lengua, las ciencias sociales y naturales, la robótica y la tecnología, el arte, las lenguas extranjeras y la ciudadanía, junto con el desarrollo de capacidades y habilidades cognitivas, interpersonales e intrapersonales de manera transversal".

En el 2016 se creó el Plan Nacional Integral de Educación Digital (PLANIED) [13], que integra los programas Conectar Igualdad y Primaria Digital, con la misión de "integrar a la comunidad educativa en la cultura digital, promoviendo la innovación pedagógica y la calidad de los aprendizajes". Luego se sumó el plan Escuelas del Futuro [14], cuyo propósito es "construir un modelo pedagógico innovador, que permita a los alumnos disfrutar de la construcción de su aprendizaje, en un marco de creatividad, exploración y colaboración, en contacto con una variedad de soluciones tecnológicas". Este último ofrece juegos y desafíos que incluyen drones, robots, plataformas interactivas y laboratorios virtuales.

En abril de 2018 se creó el Plan Aprender Conectados[2], una política integral de innovación educativa que busca garantizar la alfabetización digital para el aprendizaje de competencias y saberes necesarios para la integración en la cultura digital y la sociedad del futuro.

En septiembre del 2018 el CFE, por medio de la Resolución 343/18, aprobó los Núcleos de Aprendizajes Prioritarios (NAP) de Educación Digital, Programación y Robótica, que representan un gran avance para la enseñanza de los contenidos relacionados con la Informática como disciplina.

Esta resolución establece la obligatoriedad de la educación digital, la programación y la robótica en todos los establecimientos del país. Con su implementación, los documentos

1 https://www.educ.ar/noticias/127730/el-cfe-declaro-de-importancia-estrategica-a-la-ensenanza-y-el-aprendizaje-de-la-programacion
2 https://www.argentina.gob.ar/educacion/aprender-conectados

curriculares de todas las jurisdicciones incluyen e implementan los NAP, adoptando diferentes estrategias y considerando las particularidades de sus contextos, necesidades, realidades y políticas educativas, en el lapso de dos años.

La iniciativa Program.AR

Esta iniciativa, impulsada por la Fundación Sadosky[3], tiene como objetivo llevar la enseñanza y el aprendizaje de las Ciencias de la Computación a la escuela argentina. Incluye múltiples aspectos relacionados con la difusión y popularización de la disciplina, la generación de contenidos escolares y la formación docente, entre otros.

Para la formación docente, Program.AR[4] ha elaborado un curso que incluye contenidos, herramientas y actividades especialmente diseñados para llevar la enseñanza de la programación a las escuelas. El enfoque pedagógico se basa en el "aprendizaje por indagación", una metodología de enseñanza-aprendizaje a través de la cual los estudiantes deben encontrar soluciones a un problema a partir de un proceso de investigación y reflexión sobre las actividades realizadas para construir la solución [15]. Cabe destacar que la UNNE, desde el año 2015, participa en esta iniciativa con un equipo docente a cargo de la capacitación en Programación y Didáctica, destinada a maestros y profesores de establecimientos primarios, secundarios y terciarios [16].

Para lograr los objetivos de las políticas públicas orientadas a incorporar la Ciencia de la Computación en las escuelas, la formación de los docentes es fundamental.

2.2 Calidad de software

La calidad de software se refiere a la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo plenamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente [17].

Por su parte, el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) [18], define calidad de software como el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario,

[3] http://www.fundacionsadosky.org.ar/

[4] http://program.ar/formacion-docente/

denotando que el énfasis radica en los requisitos específicos del sistema y en la búsqueda de la satisfacción del cliente. En la evaluación de la calidad software, existen varias dimensiones fundamentales a tener en cuenta: requisitos de calidad, modelos de calidad y características de calidad.

Los requisitos determinan necesidades del usuario a cubrir por el producto software, los modelos de calidad establecen estructuras de características y subcaracterísticas relacionadas para valorar el producto.

Para garantizar la calidad de software se debe implementar algún modelo o estándar de calidad que permita la gestión de atributos en el proceso de construcción de software, teniendo en cuenta que la concordancia de los requisitos y su construcción son la base de las medidas de calidad establecidas [19].

La norma ISO/IEC 25000 SQuaRE [20] define un modelo de calidad. Es la primera propuesta en incluir un modelo especifico de la calidad en uso de forma separada a la especificación habitual de la calidad, y además se trata de una de las normas más actuales y extendidas para este propósito.

Según esta norma, se entiende característica de calidad como una categoría de atributos de la calidad que interviene en la calidad del software.

Modelos de calidad de software

Los modelos de calidad son aquellos documentos que integran la mayor parte de las mejores prácticas, proponen temas de administración en los que cada organización debe hacer énfasis, integran diferentes prácticas dirigidas a los procesos clave y permiten medir los avances en calidad. En términos de calidad de software, el modelo debe enfocarse en el seguimiento y evaluación de cada etapa de construcción del producto software. En este ámbito, el modelo de calidad debe permitir evaluar el sistema, cualitativa o cuantitativamente, y de acuerdo con esta evaluación la organización podrá proponer e implementar estrategias que permitan la mejora del proceso dentro de las etapas de análisis, diseño, desarrollo y pruebas del software.

Los modelos de calidad de software se clasifican de acuerdo con el enfoque de evaluación, ya sea a nivel de proceso, producto o calidad en uso [19].

Calidad a nivel de proceso

La calidad de un sistema software debe ser programada desde el inicio del proyecto, y posteriormente, en cada etapa del proceso de desarrollo se debe llevar a cabo el control y seguimiento de los aspectos de calidad, para minimizar los riesgos y ofrecer soporte continuo. Se garantiza así un óptimo nivel de cumplimiento de los factores de calidad, teniendo en cuenta que si en alguna de las etapas se deja de lado la verificación de los factores y criterios es posible que se presente deficiencia en alguno de éstos y disminuirá el nivel de calidad no solo del proceso, sino también del producto en desarrollo.

Calidad a nivel de producto

La principal finalidad del modelo de calidad de producto es especificar y evaluar el cumplimiento de criterios del producto, para lo cual se aplican medidas internas y/o medidas externas.

Se define al producto software como la colección de componentes necesaria para garantizar el correcto funcionamiento y mantenimiento eficiente durante su ciclo de vida, incluyendo programas informáticos o código, documentación, datos necesarios y procedimientos [18].

En calidad del software se consideran tres perspectivas complementarias: interna, externa y en uso.

Calidad interna/externa

La calidad interna considera las propiedades estáticas del software, que dependen únicamente de cómo se ha especificado, diseñado e implementado. Ejemplos de las características de calidad interna son la complejidad del software, su modularidad o el tamaño de sus artefactos. La calidad externa tiene en cuenta el comportamiento del software, tanto en los entornos de prueba como de producción. Se miden factores como el tiempo de ejecución o la memoria consumida en un determinado hardware.

Tanto la calidad interna como externa son las clásicamente aplicadas y más estudiadas.

Calidad en uso

Se define como la visión de calidad de los usuarios en un contexto que integra el software, y es medida sobre los resultados de usar el software en ese contexto, antes que sobre las

propiedades del software en sí mismo [21]. Los enfoques interno y externo consideran la calidad del producto, en tanto que la calidad en uso evalúa la experiencia del usuario y el efecto de la interacción entre el usuario y la aplicación.

Los procesos y evaluaciones de calidad del software en la industria están principalmente centrados en características estáticas, de rendimiento, relativas a la calidad interna y externa, y no en la evaluación del software en uso. Estas perspectivas se miden a través de un conjunto de pruebas automatizadas y se encuentran estandarizadas e instaladas en el ciclo de vida del software.

Por su parte, la experiencia del usuario, la percepción que éste tiene del software o el uso que hace del mismo, en la industria generalmente se evalúa a través de la opinión explicita del usuario: encuestas de satisfacción, entrevistas u otros métodos que involucren preguntar directamente al usuario.

2.3 Estándares de calidad de producto software

En el presente TFM se analizará e implementará el modelo de calidad establecido por la Norma ISO/IEC 25000.

2.3.1 La familia ISO/IEC 25000

La norma ISO/IEC 25000 - SQuaRE, es una familia de normas que tiene por objetivo la creación de un marco de trabajo común para evaluar la calidad del producto software [22].

La familia ISO/IEC 25000 es el resultado de la evolución de otras normas, especialmente de la ISO/IEC 9126 (describe las particularidades de un modelo de calidad del producto software), e ISO/IEC 14598 (aborda el proceso de evaluación de productos software). Está compuesta por cinco divisiones, que se muestran en la Figura 1.

Figura 1. Divisiones de la Familia ISO/IEC 25000
Fuente: https://iso25000.com/index.php/normas-iso-25000

2.3.2 ISO/IEC 25010- Modelo de Calidad

El modelo de calidad representa la piedra angular en torno a la cual se establece el sistema para la evaluación de la calidad del producto. En este modelo se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado.

La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Son precisamente estos requisitos (funcionalidad, rendimiento, seguridad, mantenibilidad, etc.) los que se encuentran representados en el modelo de calidad, el cual categoriza la calidad del producto en características y subcaracterísticas.

El modelo de calidad del producto definido por la ISO/IEC 25010 se encuentra compuesto por las ocho características de calidad que se muestran en la Figura 2.

Las características definidas en ISO/IEC 25010 se describen de la siguiente manera:

Adecuación funcional: Hace referencia a la capacidad que tiene un producto software para proveer las funciones que satisfacen los requerimientos declarados e implícitos, cuando el software se utiliza bajo determinadas condiciones.

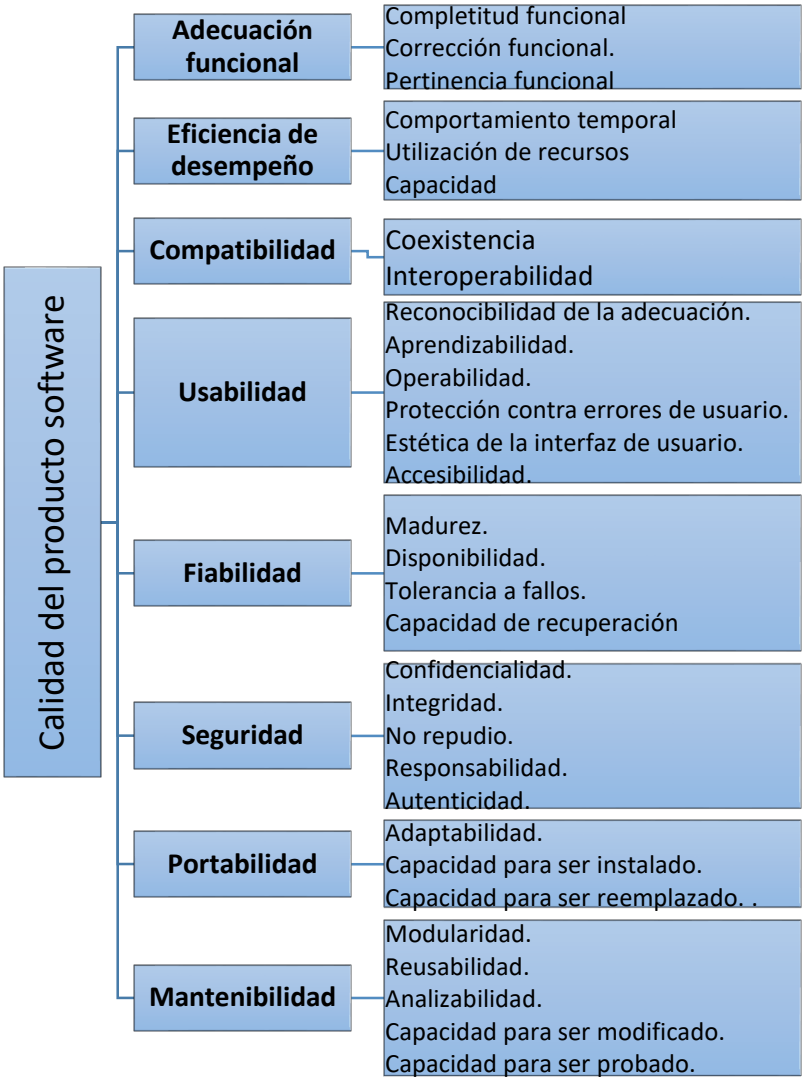

Figura 2. Modelo de calidad del producto software ISO/IEC 25010
Fuente: Elaboración propia adaptado de https://iso25000.com/index.php/normas-iso-25000

- **Eficiencia de desempeño**: Representa el desempeño del producto relativo a la cantidad de recursos utilizados bajo determinadas condiciones.
- **Compatibilidad:** Capacidad de dos o más sistemas o componentes de intercambiar información y llevar a cabo funciones específicas bajo el mismo entorno de hardware y/o software.
- **Usabilidad**: Capacidad del producto para ser aprendido, atractivo, usado y entendido por el usuario bajo determinadas condiciones.

- **Fiabilidad:** Representa el desempeño de un sistema o componente a la hora de realizar funciones específicas bajo determinadas condiciones y periodos de tiempo determinados. Fiabilidad: Calidad/Tiempo.
- **Seguridad:** Capacidad de proteger la información de manera tal que no pueda ser leída o modificada por cualquier persona o sistema no autorizados.
- **Mantenibilidad:** Representa el esfuerzo requerido para realizar modificaciones de forma efectiva y eficiente debido a necesidades.
- **Portabilidad**: Capacidad del producto software de ser transferido eficientemente de un entorno hardware o software a otro.

2.3.3 ISO/IEC 25040: Modelo de Referencia, Evaluación y Guía

Define un modelo de referencia para la evaluación, considerando las entradas, restricciones y recursos necesarios para obtener las salidas correspondientes [22].

El proceso para llevar a cabo la evaluación del producto de software consta de cinco etapas, que se muestran en la Figura 3.

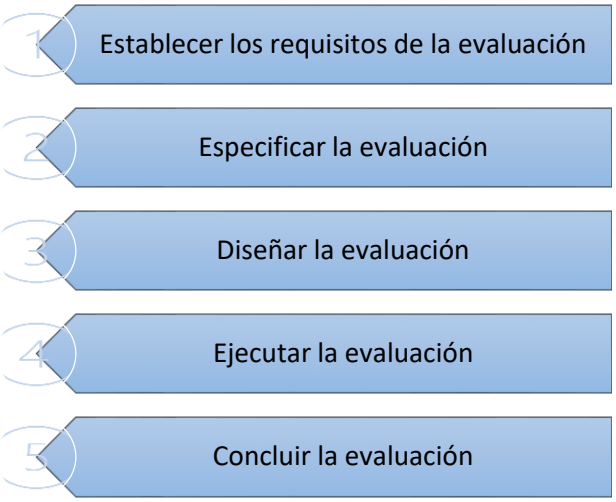

Figura 3. Modelo de referencia para la evaluación ISO/IEC 25040
Fuente: Elaboración propia adaptado de https://iso25000.com/index.php/normas-iso-25000/iso-25040

Etapa1 - Establecer los requisitos de la evaluación.

1.1: *Establecer el propósito de la evaluación:* Tarea destinada a indicar el propósito por el cual la organización quiere evaluar la calidad de un producto de software. 1.2: *Obtener los requisitos de calidad del producto*: Requiere identificar las partes interesadas en el producto (desarrolladores, posibles adquisidores, usuarios,

proveedores, entre otros). Además, se describen los requisitos de calidad del producto utilizando un modelo de calidad.

1.3: *Identificar las partes del producto que se deben evaluar:* Se deben identificar y documentar las partes del producto de software incluidas en la evaluación. Tener en cuenta que el tipo de producto a evaluar depende de la fase en el ciclo de vida en que se realiza la evaluación.

1.4: *Definir el rigor de la evaluación*: Se busca definir el rigor de la evaluación basándose en el propósito y en el uso previsto del producto software. Al hablar de riesgos se hace referencia a diferentes tipos, tales como riesgos para la seguridad, riesgos económicos o riesgos ambientales.

Etapa 2 - Especificar la evaluación. Dentro de esta actividad se especifican los módulos de evaluación (métricas, herramientas y técnicas), junto con los criterios de decisión a aplicar.

2.1: *Seleccionar los módulos de evaluación*: El evaluador es encargado de seleccionar las métricas de calidad, técnicas y herramientas que cubran todos los requisitos de la evaluación. Dichas métricas deben poder compararse con los criterios definidos para poder tomar decisiones. Para esta tarea, se puede tener en cuenta la norma ISO/IEC 25020.

2.2: *Definir los criterios de decisión para las métricas*: Se deben definir los criterios de decisión, los cuales son umbrales numéricos que se pueden relacionar con los requisitos de calidad y con los criterios de evaluación para decidir la calidad del producto.

2.3: *Definir los criterios de decisión de la evaluación*: Se deben definir los criterios para las diferentes características evaluadas. Estos resultados, en un mayor nivel de abstracción, permiten realizar la valoración de la calidad del producto en forma general.

Etapa 3 - Diseñar la evaluación. En esta actividad se define el plan con las tareas que se deben realizar en la evaluación.

3.1: *Planificar las actividades de la evaluación*: Requiere planificar las actividades de la evaluación, teniendo en cuenta la disponibilidad de los recursos humanos y materiales necesarios, el presupuesto, los métodos de evaluación y estándares adaptados, las herramientas de evaluación, entre otros.

Etapa 4: Ejecutar la evaluación. Actividad destinada a la ejecución de las actividades de la evaluación, obteniendo las métricas de calidad y aplicando los criterios de evaluación.

4.1: *Realizar las mediciones*: Realizar las mediciones sobre el producto de software para obtener los valores de las métricas seleccionadas e indicadas en el plan de evaluación. Todos los resultados deben ser registrados.

4.2: *Aplicar los criterios de decisión para las métricas*: Aplicar los criterios para las métricas sobre valores obtenidos en la medición de un producto.

4.3: *Aplicar los criterios de decisión de la evaluación*: Aplicar los criterios de decisión de la evaluación, generando como resultado el grado en el que el producto cumple con los requisitos de calidad establecidos.

Etapa 5: Concluir la evaluación. Se culmina la evaluación de la calidad del producto de software, realizando un informe de resultados que será entregado al cliente y, junto con este, se revisan los resultados obtenidos.

5.1: *Revisar los resultados de la evaluación*: El evaluador y el cliente (si existe) son los encargados de revisar los resultados obtenidos en la evaluación, con el objetivo de realizar una mejor interpretación y una mejor detección de errores.

5.2: *Crear el informe de evaluación*: Una vez analizados los resultados, se elabora un informe, indicando los requisitos, los resultados, las limitaciones y restricciones, el personal evaluador, entre otros.

5.3: *Revisar la calidad de la evaluación*: El evaluador se encarga de revisar los resultados de la evaluación y la validez del proceso, de los indicadores y de las métricas aplicadas. En base a esto se obtiene un feedback, el cual debe servir para mejorar el proceso de evaluación.

5.4: *Tratar los datos de la evaluación*: Al concluir la evaluación, el evaluador debe realizar el tratamiento de los datos según lo acordado con el cliente, ya sea devolviéndolos, modificándolos, guardándolos, etc.

2.3.4 ISO/IEC 25023: Medición de la calidad de los productos de software y sistemas.

Esta norma proporciona un conjunto de medidas de calidad para las características de los productos de sistema/software [23]. Estas medidas pueden ser utilizadas para especificar requisitos, así como para medir y evaluar la calidad del producto de sistema/software.

- **Alcance:** Esta norma define medidas de calidad para evaluar cuantitativamente la calidad del sistema y del producto de software en términos de características y subcaracterísticas definidas en ISO/IEC 25010. Está compuesta por:

- Un conjunto básico de medidas de calidad para cada característica y subcaracterística;
- Una explicación de cómo aplicar medidas de calidad de sistemas y productos de software.

Esta norma no asigna rangos de valores de las medidas a niveles nominales o grados de cumplimiento porque estos valores se definen en función de la naturaleza del sistema, producto o parte del producto, y dependiendo de factores como la categoría del software, nivel de integridad y necesidades de los usuarios. Algunos atributos podrían tener un rango deseable de valores, que no depende de las necesidades específicas del usuario sino de factores genéricos; por ejemplo, factores cognitivos humanos.

Las medidas de calidad propuestas están destinadas principalmente a ser utilizadas para garantizar la calidad y mejorar los productos de sistemas y software durante o después del proceso del ciclo de vida de desarrollo.

2.4 Conceptos de medición de la calidad del producto de software.

La calidad de un producto de sistema/software es el grado en que satisface las necesidades declaradas e implícitas de sus diversas partes interesadas y, por lo tanto, proporciona valor. Estas necesidades declaradas e implícitas están representadas en la serie de estándares SQuaRE mediante modelos de calidad que categorizan la calidad del producto del sistema/software en características, que en algunos casos se subdividen en subcaracterísticas. Las propiedades mensurables relacionadas con la calidad de un sistema/producto de software se denominan propiedades para cuantificar y pueden asociarse con medidas de calidad. Estas propiedades se miden aplicando un método de medición. Un método de medición es una secuencia lógica de operaciones utilizadas para cuantificar propiedades con respecto a una

escala específica. El resultado de aplicar un método de medición se denomina elemento de medida de calidad.

Las características y subcaracterísticas de calidad se pueden cuantificar aplicando funciones de medición. Una función de medición es un algoritmo utilizado para combinar elementos de medición de calidad. El resultado de aplicar una función de medición se llama medida de calidad. De esta manera, las medidas de calidad se convierten en cuantificaciones de las características y subcaracterísticas de calidad. Se puede utilizar más de una medida de calidad para medir una característica o subcaracterística de calidad.

2.4.1 Enfoque para la medición de la calidad

El enfoque para la medición de la calidad se basa en las necesidades de calidad del usuario, que incluyen requisitos para la calidad del sistema en uso en contextos específicos. Estas necesidades identificadas se toman en cuenta al especificar medidas de calidad externas e internas, utilizando características y subcaracterísticas de calidad del producto de software.

La calidad del producto de software se puede evaluar de tres formas:

- midiendo las propiedades internas, que normalmente son medidas estáticas de productos intermedios.
- midiendo las propiedades externas, que normalmente implican medir el comportamiento del código cuando se ejecuta.
- midiendo la calidad en las propiedades de uso, que se realizan cuando el producto está en uso real o simulado.

Esta Norma ofrece un conjunto sugerido de medidas de calidad para sistemas y software (tanto medidas internas como externas) que se deben utilizar en conjunto con el modelo de calidad ISO/IEC 25010.

Los usuarios de esta Norma tienen la libertad de modificar las medidas de calidad definidas, pueden definir y utilizar medidas de calidad que no estén identificadas o definidas en la Norma.

Cuando se emplea una medida de calidad nueva o modificada que no está identificada en esta Norma Internacional, el usuario debe detallar cómo dicha medida se relaciona con el modelo

de calidad ISO/IEC 25010, o con cualquier otro modelo de calidad alternativo que se esté utilizando.

La mayoría de las medidas de calidad emplean una función de medición que normaliza el valor resultante dentro de un rango de 0,0 a 1,0. Un valor más cercano a 1,0 se considera mejor. En los casos donde esto no se cumple, la interpretación se describe en una nota aclaratoria.

2.4.2 Medidas de calidad de los productos de sistemas y software

Las medidas de calidad se pueden utilizar con diferentes técnicas de evaluación que podrían elegirse según las características de calidad y los niveles de calificación de la evaluación, dependiendo de si se utilizan como medidas internas o externas [23].

Medidas de Adecuación Funcional

Se utilizan para evaluar el grado en que un producto o sistema proporciona funciones que satisfacen las necesidades declaradas e implícitas cuando se utiliza en condiciones específicas [23].

Se encuentra compuestas por las siguientes medidas:

- **Completitud funcional:** Se utilizan para evaluar el grado en que el conjunto de funcionalidades cubre todas las tareas especificadas y los objetivos del usuario.
- **Corrección funcional:** Se utilizan para evaluar el grado en que un producto o sistema proporciona los resultados correctos con el grado de precisión requerido.
- **Pertinencia funcional:** Se utilizan para evaluar el desempeño en relación con la cantidad de recursos utilizados en las condiciones establecidas. Los recursos pueden incluir otros productos de software, la configuración de software y hardware del sistema y materiales (por ejemplo, papel de impresión, medios de almacenamiento).

Medidas de usabilidad

Las medidas de usabilidad se utilizan para evaluar el grado en que un producto o sistema puede ser utilizado por usuarios específicos para lograr objetivos específicos con eficacia, eficiencia y satisfacción en un contexto de uso específico.

Muchas medidas de usabilidad externas son probadas por usuarios que intentan utilizar una función. Los resultados estarán influenciados por las capacidades de los usuarios y las características del sistema anfitrión. Esto no invalida las mediciones, ya que el software evaluado se ejecuta en condiciones explícitamente especificadas por una muestra de usuarios que son representativos de un grupo de usuarios identificado. Para productos de uso general, se podrían utilizar representantes de una variedad de grupos de usuarios. Para obtener resultados confiables, es necesaria una muestra de un grupo grande de usuarios comprometidos, aunque se puede obtener información útil de grupos más pequeños. Los usuarios realizan la prueba sin pistas ni asistencia externa.

Las medidas de usabilidad inevitablemente generarían resultados algo subjetivos. En caso de dificultades para medir con una escala de razón, se puede utilizar una escala ordinal como alternativa dependiendo de la situación, por ejemplo: 1,0 (excelente), 0,8 (bueno), 0,6 (promedio), 0,4 (pobre), y 0,2 (malo) [23].

Se encuentra compuestas por las siguientes medidas:

- **Idoneidad y Reconocibilidad:** Los usuarios deben poder seleccionar un sistema/producto de software que sea adecuado para el uso previsto. Las medidas de calidad para el reconocimiento de la idoneidad se utilizan para evaluar el grado en que los usuarios pueden reconocer si un producto o sistema es apropiado para sus necesidades. Se pueden utilizar medidas de idoneidad y reconocibilidad para evaluar si los nuevos usuarios pueden comprender si el producto o sistema de software es adecuado para sus fines o no.
- **Capacidad de aprendizaje**: se utilizan para evaluar el grado en que un producto o sistema puede ser utilizado por usuarios específicos para lograr objetivos específicos de aprendizaje, para utilizar el producto o sistema con eficacia, eficiencia, libre de riesgos y satisfacción en un contexto de uso específico.
- **Operabilidad**: se utilizan para evaluar el grado en que un producto o sistema tiene atributos que lo hacen fácil de operar y controlar.

  Se espera que las medidas de operabilidad se midan mediante pruebas operativas realizadas por representantes de los operadores o usuarios finales, o se pueden medir mediante análisis estáticos como la revisión de requisitos, especificaciones de diseño o

manuales de usuario. Si se utiliza una medida de calidad de operatividad interna o externa para evaluar si los usuarios pueden operar y controlar el software. Las medidas de operatividad se pueden clasificar según los siguientes principios de diálogo en ISO 9241-110: idoneidad del software para la tarea; carácter autodescriptivo del software; controlabilidad del software; conformidad del software con las expectativas del usuario; tolerancia a errores del software; idoneidad del software para la individualización.

- **Protección contra errores del usuario:** se utilizan para evaluar el grado en que el sistema protege a los usuarios contra cometer errores. Se espera que las medidas de protección contra errores del usuario se midan mediante pruebas operativas realizadas por representantes de los operadores o usuarios finales, o se pueden medir mediante análisis estáticos, como la revisión de requisitos, especificaciones de diseño o manuales de usuario.
- **Estética de la interfaz de usuario**: se utilizan para evaluar el grado en que la interfaz de usuario permite una interacción agradable y satisfactoria para el usuario.
- **Accesibilidad**: se utilizan para evaluar el grado en que un producto o sistema puede ser utilizado por personas con la más amplia gama de características y capacidades para lograr un objetivo específico en un contexto de uso específico.

Medidas de Portabilidad

Las medidas de portabilidad se utilizan para evaluar el grado de eficacia y eficiencia con el que un sistema, producto o componente puede transferirse de un hardware, software u otro entorno operativo o de uso a otro.

Se encuentra compuestas por las siguientes medidas:

- **Adaptabilidad**: se utilizan para evaluar el grado en que un producto o sistema puede adaptarse de manera efectiva y eficiente a hardware, software u otros entornos operativos o de uso diferentes o en evolución.
- **Instalabilidad**: se utilizan para evaluar el grado de efectividad y eficiencia con el que un producto o sistema puede instalarse y/o desinstalarse exitosamente en un entorno específico.

- **Reemplazabilidad:** se utilizan para evaluar el grado en que un producto puede reemplazar a otro producto de software específico para el mismo propósito en el mismo entorno.

2.5 Conclusiones del capítulo

Se presenta el estado de situación de las iniciativas orientadas a incluir las Ciencias de la Computación en las escuelas argentinas, para ilustrar el grado de avance que tienen estas iniciativas en la actualidad que fundamentan la necesidad de aportar mayor conocimiento sobre las herramientas o entornos de programación que se utilizan en la enseñanza de la programación. Como se puede observar, las iniciativas son bastante novedosas y aun se ensayan distintas propuestas que incluyen hardware y software diversos, pero existen pocos antecedentes de evaluación de la calidad de las herramientas o entornos de programación que se utilizan en la enseñanza en los niveles educativos preuniversitarios.

Se detallan también los conceptos sobre calidad de productos software, estándares de modelo de calidad, de proceso de evaluación y de medición, que serán utilizados en el marco de trabajo.

Capítulo 3

Herramientas para la Enseñanza de Programación Inicial

Este capítulo realiza una introducción acerca de las herramientas que se utilizan para la enseñanza de la programación inicial y luego se describen las características de las herramientas software seleccionadas para realizar el proceso de evaluación de calidad.

3. Herramientas para la Enseñanza de Programación Inicial

3.1. Criterios para elegir un entorno para enseñar a programar a principiantes

La enseñanza de programación en los niveles educativos preuniversitarios ha vuelto crucial para entender nuestra sociedad y cultura moderna. Los cursos de enseñanza de la programación a nivel inicial, tradicionalmente se enfocan en la programación desde una perspectiva profesional, lo que puede resultar demasiado complejo para estos niveles. Por lo tanto, han surgido nuevos enfoques que buscan simplificar la enseñanza de la programación.

Un aspecto clave de la enseñanza de la programación inicial, es la elección del lenguaje de programación y el entorno de trabajo, sin embargo, esta elección a menudo puede ser abrumadora debido a la variedad de opciones disponibles, lo que puede llevar a los docentes a tomar decisiones basadas en tendencias o desconocimiento en lugar de una consideración cuidadosa de las implicaciones para el curso que desea desarrollar.

Entra en juego la “paradoja del lenguaje”, el lenguaje y el entorno a utilizar no son importantes, pero sin embargo son importantes. Por un lado, no es importante qué lenguaje o entorno se elige, porque lo importante son los conceptos. Pero, por otra parte, dado que el lenguaje, experimentado a través del entorno, es la única forma de expresar adecuadamente esos conceptos, es importante ver de qué manera este entorno afecta la forma de transmitir los conceptos deseados [8].

En [24] el autor propone un análisis de los distintos aspectos que deberían considerarse en la elección de un entorno para enseñar a programas a principiantes. A continuación, se brinda una síntesis de estas ideas.

Para facilitar la elección de entornos y lenguajes de programación propone una clasificación basada en varios aspectos, como la expresividad del lenguaje, la interactividad formativa y la evaluación automática. Esta clasificación se organiza en diferentes dimensiones y facetas, proporcionando un marco útil para tomar una decisión informada sobre el entorno y el lenguaje a utilizar en la enseñanza de la programación. Las dimensiones son las siguientes:

Propósito y expresividad: lenguaje de propósitos general o específico

Esta dimensión incluye el propósito del lenguaje, los elementos fundamentales que ofrece, las consideraciones asociadas al paradigma de programación y la definición del lenguaje.

En cuanto al propósito, los lenguajes de propósitos generales, utilizados en la industria a menudo carecen de consideraciones didácticas y pueden ser demasiado complejos para la enseñanza inicial. Por otro lado, los lenguajes diseñados específicamente para aprender a programar, como Logo, Scratch, Gobstones, PilasBloques, han ganado relevancia en la última década.

Al evaluar un lenguaje de programación para la enseñanza, es importante considerar el conjunto de elementos primitivos que ofrece para expresar problemas. Los lenguajes diseñados para enseñar a programar suelen ofrecer un universo de discurso concreto y familiar para los estudiantes, logo con la tortuga y sus dibujos, los personajes y sus acciones en Scratch y Pilas Bloques, y el tablero y las bolitas en Gobstones.

Además, es importante considerar qué tan bien estos elementos son entendidos por los estudiantes, qué tan útiles son para expresar conceptos de programación y qué tan generalizables son para luego enseñar elementos más abstractos de los lenguajes de propósitos generales.

Otra faceta a considerar está relacionada con los paradigmas de programación en los que se basan los lenguajes. La mayoría de los lenguajes populares para los primeros cursos se basan en el paradigma imperativo, pero también existen otros paradigmas como el funcional o el de objetos. Por lo tanto, es importante considerar si la enseñanza de la programación se limitará a enseñar un subconjunto de la programación imperativa/estructurada, o si se buscará una formación más amplia y de largo plazo que permita luego el aprendizaje de otros paradigmas.

Formas de construcción: lenguaje textual, o entorno de bloques

En la programación tradicional, los programas se expresan y ensamblan mediante cadenas de texto. Sin embargo, en la programación basada en bloques, los programas se construyen mediante bloques, similares a las piezas de un rompecabezas. La programación basada en bloques simplifica la construcción de programas por la combinación que está dada por la

forma misma de los bloques. Esto elimina la necesidad de conocer las reglas de sintaxis para establecer qué combinaciones son válidas.

El primer entorno de bloques fue Logo, desde entonces, los entornos de enseñanza de programación han comenzado a incorporar bloques como forma de construir los programas. Entre los entornos actuales basados en bloques se encuentran Scratch, Pilas Bloques y Gobstones entre otros. Estos entornos basados en bloques se utilizan principalmente para principiantes de todas las edades, especialmente los niveles preuniversitarios.

El debate actual en la enseñanza de programación se centra en si es mejor usar entornos basados en bloques o en lenguajes de texto, y cuándo y cómo se debe hacer la transición entre ellos.

Los entornos basados en bloques tienen varias ventajas, su expresión visual proporciona ayuda concreta para entender cómo se pueden utilizar y ensamblar los bloques. Además, la categorización visual de los bloques en una "caja de herramientas" facilita la exploración y comprensión gradual de las diferentes categorías y bloques. Sin embargo, los bloques tienen limitaciones en su capacidad de expresión debido a su forma fija y la necesidad de un entorno digital para su uso.

Por otro lado, los lenguajes de texto permiten una mayor variedad de combinaciones de elementos del lenguaje y no requieren un entorno digital para su uso. Sin embargo, los errores de sintaxis, que son comunes en los principiantes, pueden ser desmotivantes.

Por lo tanto, si se espera aprender programación más allá de las ideas básicas, es necesario aprender a trabajar con lenguajes de texto. La cuestión es cuándo y cómo se debe hacer esta transición desde los bloques al texto.

Herramientas de soporte: entorno de mundo abierto, o controlado

Esta dimensión incluye aspectos como la posibilidad de crear cualquier programa sin restricciones, la disponibilidad de actividades controladas para guiar a los estudiantes, y las herramientas asociadas para trabajar con conceptos específicos y para colaborar con la evaluación del rendimiento.

Las facetas de esta dimensión incluyen la interactividad formativa, la evaluación automática de actividades, la transición de bloques a texto y el grado de control que el docente tiene sobre el entorno.

La interactividad formativa se refiere a cómo los lenguajes de programación se aprenden y evolucionan a través de la interacción. Los entornos pueden clasificarse en dos grupos: entornos de mundo abierto, donde los estudiantes o docentes diseñan los ejercicios de programación, y entornos de mundo controlado, donde los ejercicios de programación están predefinidos [25].

Los entornos de mundo abierto permiten a los usuarios diseñar y programar diferentes tipos de aplicaciones. Los entornos de mundo controlado tienen una secuencia predefinida de contenidos y conceptos, y suelen incluir herramientas de evaluación automática. Estos entornos pueden ser efectivos para cursos con muchos estudiantes y pueden almacenar las soluciones enviadas, la evaluación de las mismas y proporcionar retroalimentación formativa a cada estudiante [26].

Los entornos controlados de programación tienen la característica de contar con herramientas de evaluación automática, como es el caso de Pilas Bloques. Estas herramientas son posibles debido a que los ejercicios son concretos y con un problema definido.

Para los entornos abiertos, existen herramientas que pueden realizar evaluaciones automáticas de habilidades en programadores principiantes, independientes del ejercicio particular. Sin embargo, estas herramientas pueden terminar detectando errores que no impactan directamente en la incorporación de los aspectos conceptuales.

La tercera faceta en la dimensión de análisis de lenguajes de programación para la enseñanza se relaciona con la estrategia didáctica para facilitar la transición de bloques a texto. Los entornos de bloques son comúnmente utilizados para enseñar a programar a principiantes, permitiéndoles centrarse en aspectos conceptuales y evitando la complejidad sintáctica de los lenguajes de texto.

Existen dos enfoques principales para facilitar esta transición. Uno es el enfoque pedagógico, que deja la responsabilidad de la transición a cargo del docente. El otro enfoque es diseñar entornos de enseñanza de programación que faciliten la transición de un entorno basado en bloques a un lenguaje de texto. Los nuevos entornos diseñados para facilitar la transición de

bloques a texto se pueden clasificar en entornos de doble modalidad y entornos que traducen de bloque a texto [27].

Los entornos de doble modalidad permiten al estudiante elegir si desea programar en bloques o en texto y traducen automáticamente los cambios de un lenguaje a otro. Sin embargo, en entornos que traducen de bloques a texto, como Gobstones, no es posible traducir texto a bloques debido a que los bloques tienen un menor nivel de expresividad que los lenguajes textuales.

La última faceta se encuentra relacionada con el grado de control que un docente tiene sobre el entorno de aprendizaje y cómo esto afecta la enseñanza. Se consideran dos extremos: los entornos de mundo abierto, que ofrecen todas las herramientas disponibles sin restricciones, y los entornos de mundo cerrado, que ofrecen un conjunto limitado de actividades y herramientas predefinida. Ambos enfoques tienen sus desafíos, los entornos abiertos pueden ser abrumadores para los estudiantes principiantes, mientras que los entornos cerrados pueden limitar la flexibilidad didáctica del docente.

Al seleccionar un entorno para impartir un curso introductorio de programación, se debe considerar las dimensiones mencionadas y contrastarlas con los objetivos que se buscan alcanzar con los estudiantes durante la enseñanza inicial de programación.

3.2. Plataformas para la enseñanza de programación

Program.AR impulsó el desarrollo de plataformas para la enseñanza de programación en las aulas, debido a las limitaciones de las herramientas existentes, que no estaban en castellano, requerían equipamiento importante, conectividad permanente, y no respondían al enfoque educativo que esperaba darles.

Estas nuevas plataformas, están basadas en instrucciones representadas por bloques encastrables, similar a la plataforma Scratch desarrollada por el Massachusetts Institute of Technology. La programación por bloques permite aprender y enseñar a programar evitando los problemas de sintaxis. Además, estas plataformas son de acceso libre y gratuito, pueden descargarse y usarse sin Internet, y algunas incluso pueden usarse en celulares. Estas características hacen que estas plataformas sean accesibles y fáciles de usar, facilitando la enseñanza de la programación en las aulas de Argentina.

3.3. Descripción de las herramientas seleccionadas

La autora de este TFM, como miembro del equipo de la Fundación Sadosky en Corrientes y docente de una asignatura de enseñanza inicial de programación en la universidad, comprende la importancia de introducir la programación a los estudiantes desde temprana edad, y la relevancia de las herramientas utilizadas para este propósito.

Por la experiencia adquirida en ambos contextos, considera que herramientas como Pilas Bloques, Scratch y Gobstones contribuyen en este proceso, y reconoce su valor para la enseñanza de programación a los estudiantes. Opina que son una excelente vía para familiarizar a los estudiantes con el mundo de la programación y ayudarles a desarrollar habilidades para su futuro.

Estas herramientas son especialmente valiosas por varias razones:

- **Accesibilidad:** No requieren conocimientos previos de programación, lo que las hace ideales para estudiantes de todas las edades y niveles de experiencia.
- **Aprendizaje lúdico**: Incorporan elementos de juego, lo que ayuda a mantener a los estudiantes motivados y comprometidos con el aprendizaje.
- **Desarrollo de habilidades**: Ayudan a los estudiantes a desarrollar habilidades importantes como el pensamiento lógico, la resolución de problemas, la creatividad y el trabajo en equipo. Estas habilidades son valiosas no solo en el campo de la programación, sino también en muchas otras áreas.
- **Gratuitas y de código abierto**: Estas herramientas son gratuitas y de código abierto, lo que permite a todas las escuelas, independientemente de su presupuesto, contar con ellas y utilizarlas en sus planes de estudio.

Las herramientas seleccionadas son plataformas de programación visual que se utilizan para la enseñanza de la programación, especialmente para los principiantes.

Utilizan un enfoque de programación basado en bloques, lo que facilita el aprendizaje de la programación.

Los bloques gráficos funcionan como piezas de lego, que al juntarse permiten crear acciones y programar proyectos. Estos cuentan con un menú seleccionable donde se pueden visualizar los comandos y acciones disponibles. Se dividen en categorías, como acciones, procedimientos, apariencia, sonido, eventos, control, sensores, operadores entre otros.

Están diseñados con un enfoque educativo, lo que significa que se centran en enseñar conceptos fundamentales de programación de una manera accesible y divertida, ayudan a los estudiantes a desarrollar habilidades de pensamiento computacional, como la resolución de problemas, el razonamiento lógico y la creatividad.

Algunas, cuentan con una comunidad en línea donde los usuarios pueden compartir sus proyectos y colaborar con otros.

En general, proveen una versión web y una versión escritorio multiplataforma, permitiendo a los usuarios trabajar en sus proyectos incluso cuando no tienen acceso a la red.

3.1.1 Pilas Bloques:

Pilas Bloques es una aplicación para enseñar y aprender a programar por medio de bloques de forma simple y divertida. Posee desafíos con diversos niveles de dificultad para que niñas, niños y adolescentes puedan conocer el mundo de la programación. Está desarrollada en Argentina, desde la iniciativa Program.AR de la Fundación Sadosky[5]

La aplicación es software libre (puede adaptarse y modificarse de acuerdo a las necesidades del país o la región) y sus personajes reflejan la flora y la fauna argentina. Se encuentra disponible en: https://pilasbloques.program.ar/

Pilas Bloques es una herramienta que se integra con secuencias didácticas diseñadas para el aula. Estas secuencias están expresadas en los manuales y cuadernillos desarrollados por Program.AR para la enseñanza de las CC. Además, viene preinstalada en las Netbooks distribuidas por el Plan Conectar Igualdad. Estas netbooks utilizan el sistema operativo Huayra, desarrollado por el Estado argentino específicamente para su uso en el ámbito educativo.

Los desafíos de Pilas Bloques están organizados en secuencias de complejidad ascendente e invitan a los estudiantes a resolver un problema concreto en un entorno acotado, sin ofrecerles todos los conceptos ni los pasos para resolverlo, con el objetivo de promover “el aprendizaje por indagación” [28].

[5] https://pilasbloques.program.ar/acerca-de-pilas-bloques/

Los distintos ejercicios se encuentran categorizados por conceptos, por ejemplo: condicionales, repeticiones, procedimientos (Ver ejemplo en la Figura 4), cuenta con distintos personajes, y en cada ejercicio se debe resolver un problema planteado.

Ha tenido evolucionado constantemente desde su versión 1.1.2 de enero del 2017 a la versión actual Versión 2.5.0 de febrero de 2024.

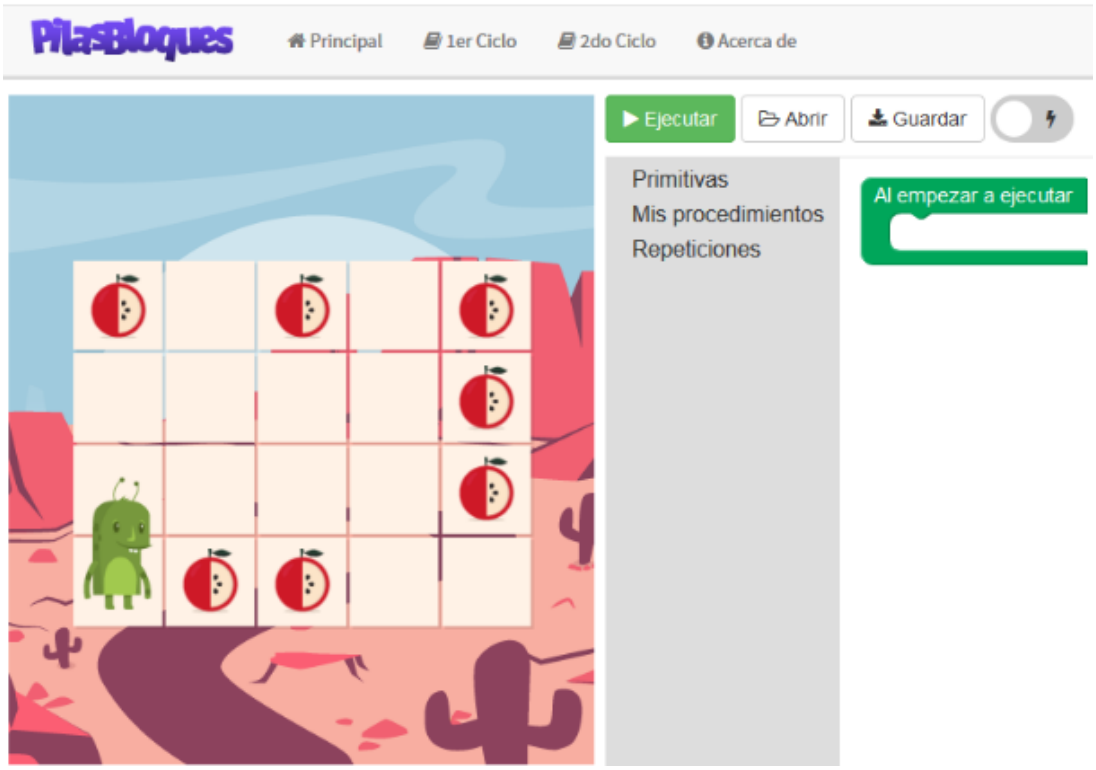

Figura 4. Desafío de PilasBloques
Fuente: https://pilasbloques.program.ar/online/#/desafio/4

3.1.2 Scratch:

Es un lenguaje de programación, desarrollado por el Lifelong Kindergarten group en el Media Lab del Instituto Tecnológico de Massachusetts, en el año 2003.

Se encuentra disponible en: https://scratch.mit.edu/

El fin didáctico de Scratch es enseñar a programar; y es posiblemente la herramienta de aprendizaje de programación más extendida en el mundo. Scratch ha sido desarrollado en torno a varios conceptos, como la facilidad de uso y el trabajo colaborativo, ya que todos los trabajos pueden ser alojados en la web de Scratch para la visualización de su código, lo cual permite su ampliación o modificación por cualquier otro usuario [29]. Esta herramienta

ahorra la escritura de código que puede inducir a errores, reemplazándola por el arrastre de bloques que se conectan entre sí como un puzle, lo que facilita su uso y comprensión. La Figura 5 muestra un ejemplo del proyecto de dinosaurios.

A lo largo del tiempo, este software ha ido evolucionando a través de las diferentes versiones. Es un proyecto de desarrollo cerrado, pero de código abierto, y la programación está dirigida por eventos.

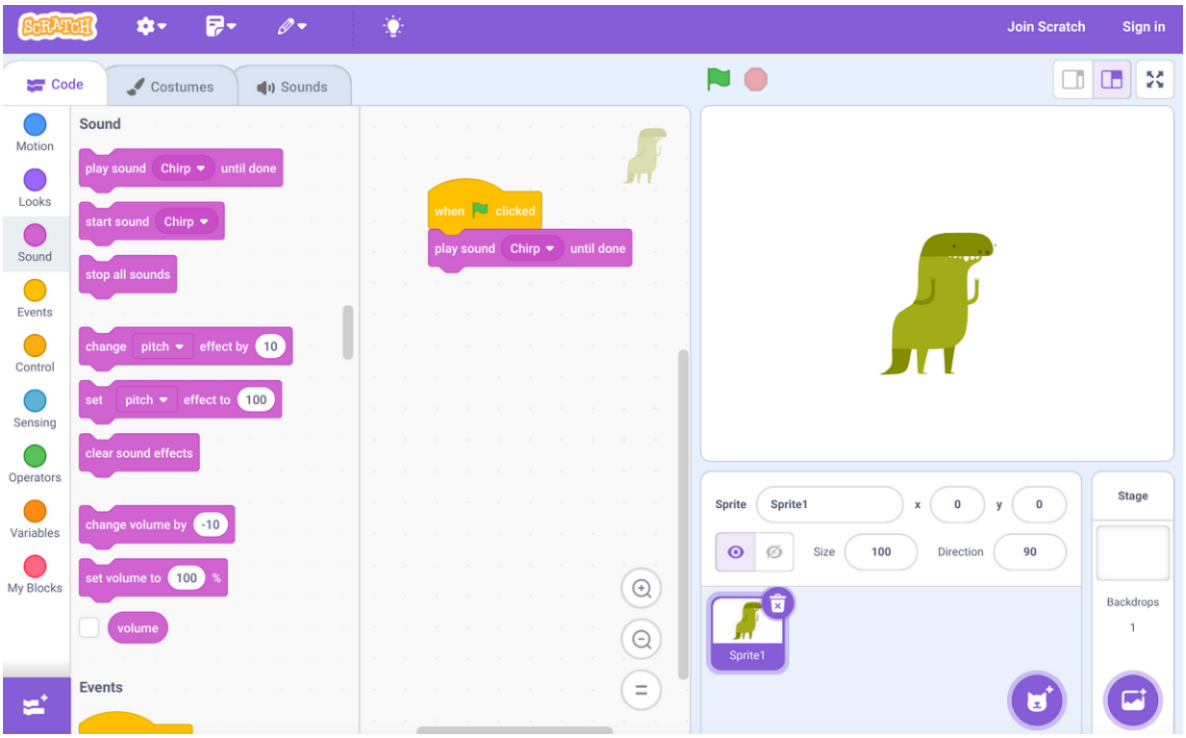

Figura 5. Entorno de Scratch
Fuente: https://scratch.mit.edu/projects/editor/?tutorial=getStarted

Diversas investigaciones dan cuenta de que la utilización de Scratch ha permitido incorporar el pensamiento computacional y favorecido la enseñanza de la programación, tal como se indica en [30], [31], [32].

3.1.3 Gobstones:

Gobstones [8] es un lenguaje de programación creado en la Universidad Nacional de Quilmes donde su principal referente es el Dr. Pablo E. Martínez López. Se diferencia de otros lenguajes dedicados a tal fin debido a varias características. La más importante es que busca orientar al programador a un pensamiento denotacional (el "qué" de los programas), en lugar del tradicional pensamiento operacional (el "cómo" o secuencia de instrucciones). Otra de

esas características es que el pasaje de este lenguaje a otros utilizados en la industria resulta más sencillo, pues posee conceptos fundamentales comunes a todos los lenguajes (pero con una separación mucho más clara que en ellos), y por la utilización de una sintaxis similar a la de éstos. La sintaxis del lenguaje fue diseñada para ser similar a lenguajes como C y Java ya que posee bloques de código encerrados por llaves, definiciones de procedimientos y funciones, parámetros por valor, variables exclusivamente locales [33]. La interfaz del entorno se visualiza en la Figura. 6.

Se encuentra disponible en: https://gobstones.github.io/gobstones-jr/.

Gobstones cuenta con una web dividida en tres secciones con diversos objetivos:

- **Gobstones Sr:** Pensado para acercar la programación en texto, permite a los estudiantes aprender conceptos más avanzados utilizando un lenguaje formal.
- **Gobstones Teacher**: Pensado para los docentes con el fin de generar guías de ejercicios
- **Gobstones Jr**: Pensado para estudiantes que quieren empezar a incursionar en el mundo de la programación, brindando la posibilidad de abrir proyectos y aprender programando en bloques.

En este TFM se seleccionó la versión Gobstones Jr, este lenguaje utiliza un enfoque pedagógico pensado para estudiantes que quieren empezar a incursionar en el mundo de la programación, brindando la posibilidad de abrir proyectos y aprender programando en bloques en el cual intenta que la tarea de escribir código sea tan simple como crear y mover bloques.

La eficacia de Gobstones radica en su capacidad para expresar de manera óptima una secuencia didáctica diseñada para enseñar programación a un nivel básico.

Gobstones se presenta como una herramienta didáctica innovadora, centrada en la experiencia práctica y visual para hacer que la programación sea accesible y comprensible.

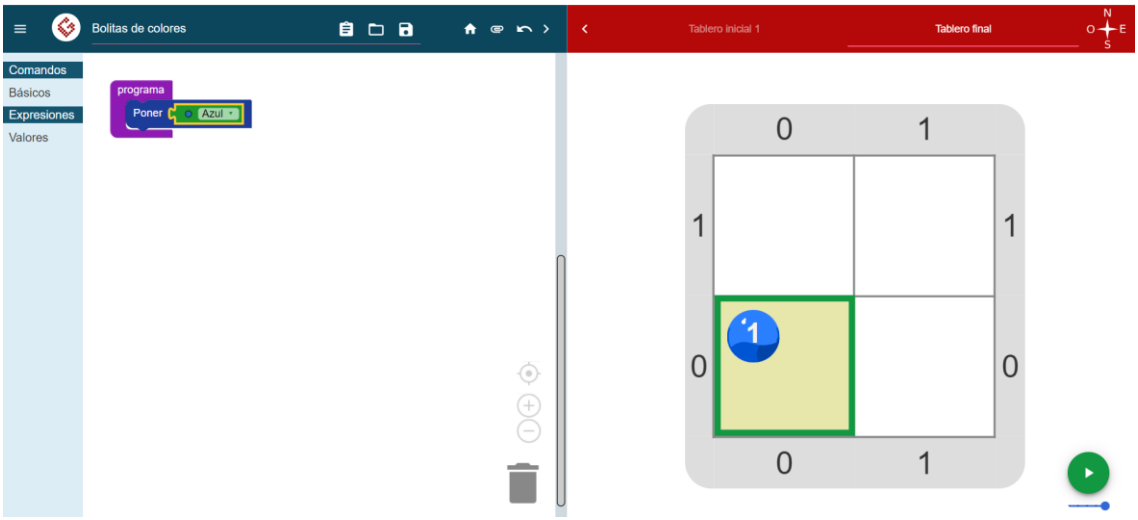

Figura 6. Entorno de Gobstones
Fuente: https://gobstones.github.io/gobstones-jr/?course=gobstones/curso-LPYSD2

3.4. Conclusiones sobre las herramientas seleccionadas

Del análisis de las características de las herramientas software descriptas previamente, desde el enfoque educativo, se destacan algunas características relevantes, coincidiendo con [29].

- Permiten a los alumnos concentrarse en la lógica de la programación abstrayéndose de la gramática del propio lenguaje.
- Permiten aplicar principios de programación sin tener que preocuparse por la sintaxis de un lenguaje de programación tradicional o tener que enfrentarse a la intimidación de un lenguaje de programación de uso profesional.
- Buscan que las instrucciones sean a nivel visual, mediante bloques ensamblables, que tienen diferentes instrucciones, de manera que programar se reduce a seleccionar y ensamblar ordenadamente las instrucciones para ejecutar.
- Son entornos lúdicos ya que incluyen personajes, sonido e interacciones que permiten generar programas construidos con elementos de programación básica estructurada
- Favorecen el aprendizaje de los alumnos dado que se fundamentan en el aprendizaje colaborativo, el pensamiento crítico, el desarrollo de las cualidades creativas, en visualizar, expresar, explorar y comprender.

- Poseen interfaces simples, con lo cual la interacción los hace accesibles, y está orientada a la rápida comprensión de los alumnos. Asimismo, los entornos son motivadores, orientados al aprendizaje a través del juego.
- Favorecen el pensamiento crítico en el contexto de la exploración y el descubrimiento y la independencia del nivel de destreza de los alumnos, tanto a nivel de comprensión lectora como en experiencia informática.
- Permiten a los alumnos formular problemas simples y construir estrategias para su resolución, incluyendo su descomposición en pequeñas partes, utilizando secuencias ordenadas de instrucciones, valiéndose de la creatividad y experimentando con el error como parte del proceso.

Desde la literatura existe consenso respecto de que estas herramientas favorecen el aprendizaje gracias a sus prestaciones que permiten abordar el pensamiento computacional y la programación desde un enfoque más amplio, interactivo y lúdico.

Sin embargo, conviene tener en cuenta que, por sus características, deben ser utilizados según las edades y la didáctica por aplicar según el contexto.

Por otra parte, dado que la programación visual con bloques no es necesario escribir las sentencias, solo se requiere mover los bloques. Entonces, tanto el número de categorías de bloques de programación como el número de bloques dentro de cada categoría son reducidos, de manera que solo se conservan los más básicos de la sintaxis de cualquier lenguaje de programación.

Por tanto, es importante tener en cuenta sus prestaciones para saber si se adaptan a las necesidades, expectativas y recursos. Esta valoración y percepción depende del docente y de la aplicación práctica que desee hacer, así como del contexto y los recursos tecnológicos con los que se cuente.

Capítulo 4

Determinación del modelo de calidad

4 Determinación del modelo de calidad

En este capítulo se lleva a cabo la definición de las características y subcaracterísticas del modelo de calidad para evaluar la calidad de las herramientas utilizadas en la enseñanza de la programación inicial, junto con una descripción detallada de las métricas que se utilizarán en el proceso de evaluación.

4.1 Modelo de la calidad ISO 25010

Las características vinculadas con el modelo de calidad están definidas en el estándar ISO/IEC25010:2005, compuesto por ocho características, que se muestra en la Figura 7.

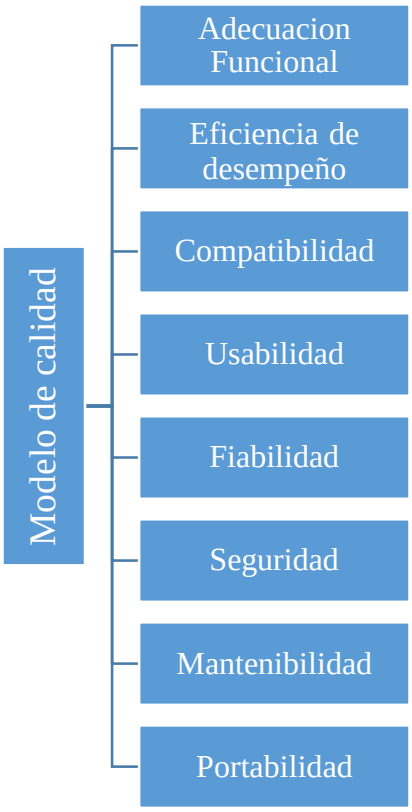

Figura 7. Modelo de calidad definido por ISO/IEC25010
Fuente: Elaboración propia

4.1.1 Definición de Categorías del nivel de importancia

El modelo de calidad ISO/IEC 25010 proporciona un marco integral para evaluar la calidad del software. Sin embargo, no todas las características de este modelo son igualmente relevantes para todas las aplicaciones software.

En la Tabla 1 se presenta una categorización de distintos niveles de importancia que se pueden asignar a las características del modelo, y su significado. El mismo criterio podrá ser aplicado a las subcaracterísticas.

Tabla 1. Categorías del Nivel de Importancia de las características

Nivel de Importancia	Símbolo	Significado
Alto	A	El grado de importancia de la característica es alto, por ende, se realiza la medición.
Medio	M	El grado de importancia de la característica no es tan relevante, pero puede o no ser medida, dependiendo del criterio del evaluador.
Bajo	B	El grado de importancia de la característica tiene escasa relevancia y no será medida.
No aplica	NA	El grado de importancia de la característica no es relevante o no se puede medir, debido a diferentes factores.

Una característica de categoría Alta permite determinar si satisface todas las necesidades del usuario, permite un trabajo eficiente, proporciona una experiencia de usuario satisfactoria, es fácil de usar y entender, y es flexible para adaptarse a una variedad de tareas y proyectos.

Una característica de categoría Media permite determinar parcialmente el cumplimiento de las necesidades del usuario. La herramienta puede ser funcional y útil, pero no proporciona una experiencia de usuario excepcional. Presenta una eficiencia moderada, permite al usuario realizar su trabajo, pero no de la manera más eficiente posible. Presenta una flexibilidad limitada, la herramienta puede ser capaz de adaptarse a una variedad de tareas y proyectos, pero no a todos.

Una característica puede ser considerada de categoría Baja porque no permite determinar significativamente el cumplimiento de las necesidades del usuario.

Una característica puede ser categorizada como “No aplica” porque puede no ser relevante para la tarea que el usuario necesita realizar, o para el contexto en que el software es utilizado.

Es importante tener en cuenta que la categorización de los distintos niveles de importancia puede ser subjetiva y depende de las necesidades y expectativas específicas del usuario, pero puede servir como guía.

4.2. Modelo de calidad propuesto

4.2.1. Determinación del nivel de importancia de las características

En el ámbito de la educación en Ciencias de la Computación, las herramientas para la enseñanza de programación inicial juegan un papel importante al proporcionar una plataforma accesible y atractiva para introducir a los estudiantes en los conceptos fundamentales de la programación. La calidad de estas herramientas puede tener un impacto significativo en la experiencia de aprendizaje del estudiante, por lo tanto, evaluar su calidad de manera efectiva contribuye a este proceso.

Utilizando la categorización de la Tabla 1, en función del propósito y el contexto de uso educativo del software a evaluar, se asignó a cada característica del modelo de calidad de la ISO 25010 el nivel de importancia que se muestra en la Tabla 2.

Tabla 2. Nivel de Importancia para el Modelo de Calidad propuesto por la ISO 25010

CARACTERICAS DEL MODELO DE CALIDAD	
Característica	**Nivel de importancia**
Adecuación Funcional	M
Eficiencia de desempeño	NA
Compatibilidad	NA
Usabilidad	A
Fiabilidad	NA
Seguridad	NA
Mantenibilidad	NA
Portabilidad	M

En el caso de la Adecuación funcional en el contexto de la programación inicial, los estudiantes están aprendiendo los fundamentos de la programación, y no necesariamente requieren todas las funciones avanzadas que podrían ofrecer algunas herramientas de programación. Por lo tanto, es suficiente con que la herramienta permita realizar las tareas

básicas de programación de manera eficiente y efectiva, por lo que se le asigna un nivel de importancia media.

La portabilidad es importante porque permite a los estudiantes utilizar la herramienta en diferentes sistemas operativos y dispositivos. Sin embargo, en la etapa inicial de aprendizaje de programación, los estudiantes pueden estar trabajando principalmente en un tipo específico de dispositivo o sistema operativo, por lo que la portabilidad puede no ser tan crítica, motivo por el cual se le asigna un nivel de importancia media.

La usabilidad, en el contexto de la programación inicial, es de suma importancia y se le asigna un alto nivel de importancia por las siguientes razones:

- **Facilita el aprendizaje**: Una herramienta de programación con alta usabilidad es intuitiva y fácil de usar, lo que permite a los estudiantes centrarse en aprender los conceptos de programación en lugar de luchar con la interfaz de la herramienta. Esto facilita el proceso de aprendizaje y ayuda a los estudiantes a progresar más rápidamente.
- **Reduce la frustración**: Las herramientas de programación con baja usabilidad pueden resultar frustrantes para los usuarios principiantes. La alta usabilidad ayuda a minimizar dicha frustración al hacer que la herramienta sea fácil de comprender y utilizar.
- **Promueve el compromiso**: Cuando una herramienta resulta sencilla de utilizar, es más probable que los estudiantes la empleen con frecuencia, continuando así con su práctica y aprendizaje. La facilidad de uso puede propiciar un mayor nivel de compromiso en el aprendizaje de la programación
- **Prepara para el futuro**: Al aprender a programar con una herramienta de alta usabilidad, los estudiantes pueden concentrarse en entender los conceptos de programación, lo que les será útil al avanzar a herramientas y lenguajes de programación más complejos en el futuro.

El modelo de calidad ISO/IEC 25010 incluye características como eficiencia de desempeño, compatibilidad, fiabilidad, seguridad y mantenibilidad. Sin embargo, no todas estas características son aplicables o relevantes al evaluar herramientas de programación visual como Pilas Bloques, Scratch y Gobstones, cuyo objetivo es facilitar el aprendizaje de los conceptos fundamentales de la programación de una manera visual y lúdica. Por lo tanto, las

características mencionadas, que son más relevantes para los productos de software de alto rendimiento, pueden no ser tan críticas en este contexto.

Eficiencia de desempeño: En la etapa inicial de aprendizaje de programación, los estudiantes seguramente no estarán trabajando en proyectos que requieran un alto rendimiento. Por lo tanto, esta característica no es considerada en este contexto.

Compatibilidad: Las herramientas de programación iniciales se diseñan con un enfoque en la enseñanza de los conceptos básicos de la programación. La compatibilidad con otros productos o entornos puede complicar este proceso de aprendizaje. A medida que los programadores ganan experiencia y sus necesidades cambian, seguramente utilizarán herramientas más avanzadas que ofrecen mayor compatibilidad y otras características.

Seguridad: Las herramientas de programación iniciales se utilizan en un entorno controlado, como un aula o un entorno de desarrollo personal, en estos casos, los riesgos de seguridad son mínimos. Implementar características de seguridad puede aumentar innecesariamente la complejidad de la herramienta. A medida que los programadores ganan experiencia y sus necesidades cambian, seguramente utilizarán herramientas más avanzadas que brindan mayor seguridad.

Mantenibilidad: Esta característica es más relevante para el desarrollo de software en otros contextos, y no implica una preocupación primordial para los usuarios de herramientas lúdicas, ya que no se espera que modifiquen o mantengan el software y por ello es considerada menos crítica en un entorno educativo inicial.

Fiabilidad: En la etapa inicial de aprendizaje de programación, los estudiantes trabajan en tareas y proyectos que son relativamente simples y no críticos. Las herramientas de programación inicial suelen utilizarse en un entorno educativo controlado, donde los problemas de fiabilidad pueden ser gestionados y resueltos de manera efectiva por los educadores o el personal de soporte técnico.

En función del propósito y el contexto de uso educativo de las herramientas de programación inicial, se considera más apropiado centrarse en características como la usabilidad, la facilidad de aprendizaje y la capacidad de fomentar el pensamiento computacional a la hora de evaluar estas herramientas, dejando de lado las características menos relevantes al

contexto, por lo tanto, no se consideran las demás características del modelo de calidad que propone la ISO 25010.

4.2.2. Descripción de las características seleccionadas

a) Adecuación Funcional

Se enfoca en garantizar que el software cumpla con las necesidades y requisitos de los usuarios en las actividades o secuencias didácticas propuestas. Es decir, el software en estudio debe proporcionar las funciones necesarias para satisfacer las necesidades de los usuarios. En el contexto de la enseñanza de la programación, esto podría incluir la capacidad de crear programas interactivos, animaciones, juegos, aplicaciones móviles, entre otros.

Se considera la siguiente subcaracterística:

- *Completitud Funcional*: Capacidad del sistema para proporcionar todas las funciones especificadas por el usuario.

b) Usabilidad

Se centra en la experiencia del usuario. Se le asigna un alto nivel de importancia, dado que considera la facilidad con la que el software puede ser aprendido, comprendido y utilizado. Asegura que la aplicación cumpla con las necesidades y expectativas del usuario, lo que a su vez mejora significativamente su experiencia de uso. De esta manera permite a los usuarios comenzar a programar rápidamente sin una curva de aprendizaje empinada. De esta forma puede aumentar la motivación y el compromiso de los usuarios, lo que es especialmente importante en contextos educativos.

Además, tiene en cuenta la accesibilidad para usuarios con diferentes habilidades y capacidades, lo que hace que sea inclusiva y accesible.

Contar con una alta usabilidad puede llevar a una mayor satisfacción del usuario, eficiencia, y una menor tasa de errores.

Se consideran las siguientes subcaracterísticas

- *Capacidad para reconocer su adecuación*. Capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades.
- *Capacidad de Aprendizaje***.** Capacidad del producto que permite al usuario aprender su aplicación.

- *Capacidad de ser usado*. Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
- *Protección contra errores de usuario*. Capacidad del producto para proteger a los usuarios de cometer errores.
- *Estética de la interfaz de usuario*. Capacidad de la interfaz de agradar y satisfacer la interacción con el usuario.
- *Accesibilidad.* Capacidad del producto de adaptarse a usuarios con determinadas características y capacidades.

c) Portabilidad

Esta característica se refiere a la capacidad de un software para funcionar correctamente en diferentes plataformas y entornos. Se le asigna un nivel de importancia media. La portabilidad asegura que las herramientas sean accesibles en una variedad de plataformas y dispositivos, permitiendo a más estudiantes acceder a ellas. Permite trabajar en proyectos desde cualquier dispositivo o sistema operativo, lo que proporciona una gran flexibilidad y facilita el aprendizaje. Asegura que el trabajo de los usuarios no quedará obsoleto si cambian de dispositivo o sistema operativo. Esto contribuye a la durabilidad y relevancia a largo plazo de su aprendizaje.

Se considera la siguiente subcaracterística:

- *Adaptabilidad.* Capacidad de adaptación de un producto a diferentes entornos de hardware, software, operacionales o de uso.

Por lo tanto, la selección de estas características para evaluar la calidad de las herramientas de programación inicial se justifica por su relevancia y su impacto en la eficacia de estas herramientas como recursos de aprendizaje.

En función de lo expresado, teniendo en cuenta la importancia asignada a las características seleccionadas, y la justificación de la menor relevancia de otras en este contexto, se propone un modelo de evaluación de calidad para llevar a cabo la evaluación de herramientas de programación inicial, que se muestra en la Figura 8.

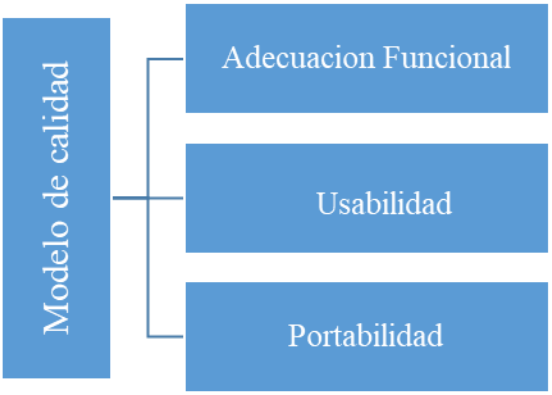

Figura 8. Modelo de calidad para la evaluación de herramientas de programación
Fuente: Elaboración propia

4.2.3. Descripción de las subcaracterísticas seleccionadas

Se describen las subcaracterísticas de las características seleccionadas para el modelo propuesto.

Adecuación Funcional

a) **Completitud funcional:** en el modelo de calidad ISO/IEC 25010 se refiere a la capacidad de un producto software para proporcionar funciones que satisfacen las necesidades y requisitos declarados e implícitos cuando se utiliza bajo condiciones específicas.

En el contexto de herramientas de programación inicial, la completitud funcional es un criterio de evaluación relevante porque estas herramientas están diseñadas para enseñar conceptos de programación a los principiantes.

Por lo tanto, es importante que proporcione un conjunto completo de funciones que permita a los usuarios aprender y experimentar con una amplia gama de conceptos de programación.

Debe ser capaz de dar soporte a los usuarios a medida que progresan desde los conceptos más básicos hasta los más avanzados. Esto requiere una completitud funcional que permita a los usuarios realizar tareas más complejas a medida que adquieren más habilidades.

Los usuarios pueden presentar diferentes necesidades y objetivos de aprendizaje, una herramienta con una alta completitud funcional puede adaptarse a una variedad de contextos de aprendizaje, proporcionando las funciones necesarias para satisfacer las necesidades de diferentes usuarios. Al proporcionar una amplia gama de funciones, estas herramientas pueden ayudar a preparar a los usuarios para el uso de herramientas de programación más

avanzadas. La familiaridad con un conjunto completo de funciones puede facilitar la transición a lenguajes de programación y entornos de desarrollo más complejos.

En el modelo propuesto en este TFM se definieron doce funcionalidades básicas, que son fundamentales en el contexto del análisis de las herramientas utilizadas para la enseñanza inicial de la programación, para realizar el cálculo de las métricas.

Las funcionalidades definidas son:

1. **Crear un programa (desafío):** Permite a los usuarios comenzar a desarrollar el código desde cero. Es el punto de partida para cualquier proyecto o tarea de programación.
2. **Abrir un programa**: La posibilidad de abrir programas existentes permite continuar trabajando en proyectos previos. Los usuarios pueden acceder al trabajo realizado previamente para realizar modificaciones o mejoras.
3. **Editar:** La capacidad de editar programas es crítica porque los usuarios deben poder realizar cambios en su código, corregir errores, agregar nuevas funcionalidades y optimizar su implementación.
4. **Guardar (determinado):** Guardar programas en ubicaciones específicas garantiza que el trabajo no se pierda. Los usuarios pueden retomar su progreso en cualquier momento sin preocuparse por perder cambios importantes.
5. **Guardar (seleccionado ubicación):** La opción de elegir la ubicación de guardado es útil para organizar proyectos y mantener una estructura de carpetas coherente. Los usuarios pueden almacenar sus programas en directorios relevantes.
6. **Descartar/Eliminar:** En algunas ocasiones los usuarios pueden querer descartar un programa incompleto o innecesario. Esta funcionalidad les permite eliminar archivos sin complicaciones.
7. **Escribir un enunciado**: La capacidad de redactar enunciados es importante para crear desafíos o ejercicios. Los usuarios pueden proporcionar instrucciones claras y concisas para otros programadores.
8. **Escribir título:** Asignar un título a un programa facilita su identificación y seguimiento. Los títulos ayudan a los usuarios a recordar el propósito o contenido de cada archivo.

9. **Ejecutar:** La funcionalidad de ejecución permite a los usuarios probar el código y verificar si produce los resultados esperados. Es esencial para depurar y validar soluciones.
10. **Ver desafío:** Si la herramienta está diseñada para desafíos de programación, ver el enunciado del desafío es crucial. Proporciona contexto y objetivos claros para los usuarios.
11. **Compartir desafío:** La posibilidad de compartir desafíos con otros programadores fomenta la colaboración y el aprendizaje conjunto. Los usuarios pueden discutir soluciones y comparar enfoques.
12. **Descargar actividades:** Descargar actividades o ejercicios es útil para el aprendizaje autodidacta y/o compartir con la comunidad. Los usuarios pueden practicar fuera de línea y revisar conceptos en su propio ritmo.

Estas funcionalidades son esenciales para una experiencia de programación completa y efectiva, ya que abarcan desde la creación hasta la colaboración y la gestión de proyectos.

Usabilidad

a) **Capacidad de reconocer su adecuación**: según el modelo de calidad ISO/IEC 25010, se refiere a la habilidad de un producto software para verificar que los requisitos se han cumplido de manera correcta. En el contexto de las herramientas de programación inicial, esta subcaracterística podría no ser un criterio de evaluación relevante. Esto se debe a que estas herramientas están diseñadas con el objetivo principal de enseñar conceptos de programación a principiantes, más que para producir software que cumpla con requisitos específicos.

Sin embargo, la idea de que el software debe cumplir con ciertos requisitos puede ayudar a fomentar buenas prácticas desde el principio. A medida que los usuarios progresan y comienzan a utilizar herramientas de programación más sofisticadas, ya estarán familiarizados con el concepto de validar la adecuación de los requisitos.

El proceso de verificar si un programa cumple con los requisitos especificados puede contribuir al desarrollo de habilidades de resolución de problemas y mejorar la comprensión de los conceptos de programación. Estas herramientas a menudo priorizan la simplicidad y la accesibilidad sobre la capacidad de reconocer su adecuación.

Están diseñadas para ser intuitivas y fáciles de entender, lo que puede implicar simplificaciones que reducen la capacidad de reconocer su adecuación en comparación con las herramientas de programación más avanzadas. Las tareas realizadas con estas herramientas suelen ser relativamente simples y no requieren una validación de requisitos sofisticada.

Para esta subcaracterística se identificaron y cuantificaron propiedades relevantes de un software considerando su adecuación para la enseñanza de la programación en los niveles iniciales de educación.

Para ellos se definieron las siguientes propiedades.

1. **Programar mediante arrastrar y soltar**: Esta característica permite a los principiantes entender los conceptos básicos de la programación sin tener que preocuparse por la sintaxis. Facilita la comprensión de la estructura del código y el flujo de control.
2. **Visualizar proyectos de otras personas**: permite a los estudiantes aprender de proyectos existentes, fomenta la comprensión de diferentes enfoques y soluciones. Además, pueden aprender buenas prácticas y técnicas de diseño al analizar proyectos de otros programadores, de esta manera, permite acelerar el proceso de aprendizaje y fomentar una comunidad de aprendizaje colaborativo.
3. **Introducir al paradigma de programación orientada a objetos (POO):** POO es un paradigma de programación que se utiliza en muchos lenguajes de programación actuales. Percibir como mínimo el concepto de objeto proporciona una base para la posterior comprensión de este paradigma.
4. **Introducir código mediante texto**: arrastrar y soltar es útil para los principiantes, es importante introducir también la codificación mediante texto para familiarizar a los usuarios con la sintaxis que se utiliza en la programación profesional. Esto les permite comprender la sintaxis y la lógica detrás de las instrucciones.
5. **Utilizar funciones con parámetros:** Las funciones con parámetros son fundamentales para entender la abstracción y cómo se puede utilizar para escribir código más eficiente y reutilizable.

6. **Similar a entornos de programación profesionales**: Familiarizar a los estudiantes con entornos similares a los que se utilizan en la programación profesional puede ayudar a que la transición a dichos entornos sea más fácil en el futuro.
7. **Conectar al exterior mediante sensores**: Esta característica permite a los estudiantes interactuar con el mundo real, lo que puede hacer que la programación sea más tangible e interesante. Pueden crear proyectos que utilicen sensores (como temperatura, luz o movimiento) para recopilar datos y tomar decisiones basadas en eventos externos.

b) **Capacidad de ser entendido**: según la norma ISO/IEC 25010 se refiere a qué tan fácil es para los usuarios entender la funcionalidad de la herramienta y cómo usarla. En el contexto de la programación inicial, esta subcaracterística es relevante porque estas herramientas están diseñadas para enseñar conceptos de programación a los principiantes. Por lo tanto, es crucial que los usuarios puedan entender rápidamente cómo usar la herramienta para facilitar su aprendizaje. Si los usuarios no pueden entender cómo usar una herramienta, es probable que encuentren dificultades para lograr sus objetivos. Por lo tanto, evaluar la capacidad de ser entendido puede ayudar a identificar posibles problemas de usabilidad. Si una herramienta es fácil de entender, los usuarios pueden comenzar a trabajar con ella más rápidamente y motiva a los usuarios a seguir aprendiendo. Esto puede mejorar la eficiencia general del proceso de aprendizaje.

c) **Operabilidad**: en el modelo de calidad ISO/IEC 25010 se refiere a qué tan bien la herramienta funciona en términos de su rendimiento y la ausencia de errores. En el contexto de herramientas de programación inicial esta subcaracterística es relevante porque una herramienta que funciona sin problemas y sin errores puede proporcionar una mejor experiencia de aprendizaje para los novatos.

   La falta de operatividad de una herramienta puede afectar la confianza del usuario en ella. Si un Software funciona bien y de manera confiable, los usuarios pueden tener más confianza en usarla para aprender y experimentar.

d) **Protección contra errores de usuario:** en la ISO/IEC 25010 se refiere a la capacidad del sistema para proteger a los usuarios de cometer errores. En el contexto de la programación inicial esta subcaracterística es relevante porque los novatos en programación están propensos a cometer errores mientras aprenden. Una herramienta que ofrece una buena protección contra errores de usuario puede ayudar a los principiantes a

aprender de manera más efectiva y segura, permitiéndoles experimentar y aprender de sus errores sin causar problemas graves. Si los usuarios saben que la herramienta los protegerá de cometer errores graves, pueden sentirse más cómodos experimentando con diferentes enfoques y soluciones. Esto puede fomentar un aprendizaje más profundo y efectivo.

Las herramientas que protegen a los usuarios de cometer errores suelen ser más fáciles de usar, esto puede mejorar la usabilidad general de la herramienta, lo que a su vez puede mejorar la experiencia de aprendizaje del usuario.

Al dar los primeros pasos en programación, es importante que los alumnos desarrollen buenos hábitos desde el principio, incluyendo la capacidad de evitar y manejar errores. Una herramienta que ofrece una buena protección contra errores de usuario puede ayudar a los alumnos a desarrollar estas habilidades importantes.

En este TFM, se seleccionaron para el análisis de las herramientas dos acciones esenciales: la estructura repetitiva y el uso de parámetros. El propósito es comprobar cómo las herramientas emiten mensajes de error para prevenir fallos del usuario. Ambas acciones son fundamentales y los estudiantes deben dominar al aprender a programar. Al enfocarse en ellas, se puede obtener una evaluación de cómo las herramientas iniciales de programación gestionan la protección de errores lo cual es vital para un aprendizaje efectivo.

e) **Estética de la interfaz de usuario:** se define en el modelo de calidad ISO/IEC 25010 como la apariencia y la disposición de la interfaz de usuario de una herramienta de software. En el contexto de herramientas de programación inicial, la estética es un criterio de evaluación relevante porque una interfaz de usuario atractiva y bien diseñada puede aumentar la motivación y el compromiso de los usuarios, especialmente para los que están dando los primeros pasos en programación. Esto puede hacer que la experiencia de aprendizaje sea más agradable y atractiva.

Muchos alumnos en programación se benefician del aprendizaje visual al utilizar bloques de colores y formas distintas para representar diferentes conceptos de programación. Esto puede ayudar a los usuarios a entender estos conceptos de manera más intuitiva.

f) **Accesibilidad**: en el modelo de calidad ISO/IEC 25010 se refiere a la capacidad de un producto software para ser usado por personas con la más amplia gama de capacidades y

situaciones. En el contexto de herramientas de programación inicial, la accesibilidad es un criterio de evaluación importante porque la programación es una habilidad valiosa que debe ser accesible para todos, independientemente de sus capacidades físicas o cognitivas. Evaluar la accesibilidad de estas herramientas asegura que están diseñadas para ser utilizadas por una amplia gama de usuarios, incluyendo personas con discapacidades. La accesibilidad puede influir en la usabilidad de una herramienta. Que sea accesible para una amplia gama de usuarios puede hacer que sea más fácil de usar, por lo tanto, puede proporcionar una mejor experiencia de aprendizaje.

Portabilidad

a) **Adaptabilidad:** se refiere a la capacidad de un producto software para ser utilizado eficazmente en diferentes contextos o condiciones. Al considerar herramientas de programación inicial la adaptabilidad es un criterio de evaluación relevante dado que son utilizadas por una amplia gama de usuarios, desde niños hasta adultos, con diferentes niveles de habilidad y experiencia en programación.

La adaptabilidad permite que estas herramientas puedan ser utilizadas para una variedad de tareas, desde la creación de juegos simples hasta la construcción de aplicaciones más complejas. Así también pueden ser utilizadas en una variedad de entornos, desde el aula hasta el hogar, y con diferentes sistemas operativos. La adaptabilidad permite que las herramientas funcionen eficazmente en todos estos entornos.

4.2.4. Determinación del nivel de importancia de las subcaracterísticas

Una vez establecidas las características a considerar se seleccionaron las subcaracterísticas que formarán parte del modelo de calidad y a continuación se le asignó un nivel de importancia, que se observa en la Tabla 3.

Tabla 3. Nivel de Importancia de las Subcaracterísticas

Característica	Subcarac-terística	Impor-tancia	Motivo de selección
Adecuación funcional	Completitud funcional	M	Se califica con valor **M** porque en esta etapa es más importante que la herramienta sea fácil de usar y comprensible que tener una amplia gama de funcionalidades. Contar con demasiadas funcionalidades puede complicar la herramienta y hacerla más difícil de aprender.

Usabilidad	Reconocibilidad de la adecuación	A	Se califica con valor **A** dado que la capacidad de reconocer si una herramienta de programación es adecuada para sus necesidades ahorra tiempo y esfuerzo que de otro modo se gastaría en probar y descartar herramientas inadecuadas.
	Aprendibilidad	A	Se califica con valor **A** porque la facilidad de aprendizaje de la herramienta permite concentrarse más rápidamente en el aprendizaje de la programación.
	Operabilidad	M	Se califica con valor **M** debido a que las herramientas de programación inicial son intuitivas y fáciles de usar, lo que facilita su operación y control.
	Protección contra errores del usuario	M	Se califica con valor **M** porque se considera menos relevante evaluar si el usuario puede evitar realizar operaciones incorrectas.
	Estética de la interfaz de usuario	M	Se califica con valor **M** porque es menos relevante evaluar si la interfaz del software agrada al usuario.
	Accesibilidad	A	Se califica con valor **A** porque es significativo que el software pueda ser utilizado por personas con discapacidad.
Portabilidad	Adaptabilidad	M	Se califica con valor **M** porque no se considera crítica durante las etapas iniciales del aprendizaje de la programación.

4.3. Definición de las métricas

En la familia de la ISO/IEC 25000, la ISO/IEC 25023 proporciona un conjunto de métricas para evaluar diversas características de calidad del software. No obstante, no todas las métricas son aplicables a todas las aplicaciones de software de manera uniforme. En el contexto de las herramientas de programación inicial, y en función del modelo de calidad propuesto para su evaluación, se han seleccionado métricas específicas que corresponden a las características definidas en el modelo.

La elección de las métricas se justifica por su relevancia y su impacto directo en la eficacia de estas herramientas como recursos de aprendizaje. Estas métricas garantizan que las herramientas sean efectivas para el propósito previsto, fáciles de usar y accesibles para una amplia gama de usuarios. De esta forma, se asegura que las herramientas de programación visual sean de alta calidad y cumplan con las expectativas de los usuarios. A continuación, se describen las métricas utilizadas para evaluar las características del modelo propuesto.

Adecuación Funcional: se puede observar en la Tabla 4.

Tabla 4. Métrica de Calidad para Adecuación Funcional

Subcaracterística	Métrica	Propósito de la métrica de calidad	Método de aplicación	Formula	Valor deseado	Tipo de medida
Completitud funcional	Completitud de la implementación funcional	¿Cuán completa es la implementación de acuerdo a las funcionalidades requeridas para el propósito de uso (enseñanza de programación)?	Contar el número de las funciones requeridas y el número de funciones que faltan o están incorrectas	X=1-(A/B) Donde B>0 A=número de funciones que están incorrectas o que no fueron implementadas. B=número de funciones básicas requeridas.	0<=X<= 1 Cuando más cercano a 1 mejor	X=contable A=contable B=contable

Usabilidad: se pueden observar en la Tabla 5.

Tabla 5. Métricas de Calidad para Usabilidad

Subcaracterística	Métrica	Propósito de la métrica de calidad	Método de aplicación	Formula	Valor deseado	Tipo de medida
Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	¿Qué cantidad de características adecuadas para la enseñanza de la programación presenta el software?	Contar la cantidad de características que posee según la lista de referencias.	X = A/B Donde B>0 A= Numero de características que posee. B= Cantidad de características definidas como referencia.	0<=X<=1 Cuando más cercano a 1 mejor	X=contable A=contable B=contable
Capacidad de aprendizaje	Integridad de la guía de usuarios	¿Qué cantidad de guías de apoyo al aprendizaje se ofrecen desde la herramienta?	Contar la cantidad de documentación que posee la herramienta para poder ejecutar cada una de las funcionalidades básicas consideradas.	X = A/B Donde B>0 A= Cantidad de documentación para realizar las funcionalidades básicas. B= Cantidad de funcionalidades básicas	0<=X<=1 Cuando más cercano a 1 mejor	X=contable A=contable B=contable
	Efectividad de la documentación del usuario o ayuda del sistema	¿Qué cantidad de funciones están descriptas correctamente en la documentación del usuario o ayuda en línea?	Contar el número de las funciones descriptas correctamente y contar el número total de funciones básicas	X=A/B Donde B>0 A= números funciones descripta correctamente B= Cantidad de funcionalidades básicas	0<=X<=1 Cuando más cercano a 1 es mejor	X=contable A=contable B=contable

Subcaracterística	Métrica	Propósito de la métrica de calidad	Método de aplicación	Formula	Valor deseado	Tipo de medida
Operabilidad	Claridad del mensaje	¿Qué cantidad de mensajes son auto explicativos?	Contar el número de mensajes implementados con explicaciones claras y el número total de mensajes implementados	X=A/B Donde B>0 A=número de mensajes implementados con explicaciones claras B= número total de mensajes implementados de las funciones básicas	0<=X<=1 Cuando más cercano a 1 es mejor	X=contable A=contable B=contable
Protección contra errores del usuario	Prevención del uso incorrecto	¿Cuántas funciones tienen la capacidad de evitar operaciones incorrectas?	Contar el número de funciones implementadas para evitar fallos de funcionamiento provocados por el uso incorrecto y el número total de operaciones iniciales incorrectas	X=A/B Donde B>0 A= número de acciones y entradas de usuario que están protegidas para no causar ningún mal funcionamiento del sistema B= número de acciones y entradas de usuario que podrían protegerse para evitar que causen mal funcionamiento del sistema.	0<=X<=1 Cuando más cercano a 1 es mejor	X=contable A=contable B=contable
Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	Que tan agradable es la combinación de color, texto y espacio en la interfaz de usuario.	Asignar un valor en la escala de likert	X=A/B Donde B>0 Donde A puede ser: 1- Desagradable 2- Poco agradable 3- Neutro 4- Agradable 5-Muy agradable	1<=X<=5 Cuando más cercano a 1 mejor	X=contable A=contable

Subcaracterística	Métrica	Propósito de la métrica de calidad	Método de aplicación	Formula	Valor deseado	Tipo de medida
	Legibilidad adecuada	Que tan adecuada es la legibilidad en función de la combinación de color, texto y fuente en la interfaz de usuario	Asignar un valor en la escala de likert	X=A/B Donde B>0 Donde A puede ser: 1- Nada adecuada 2- Poco adecuada 3- Neutro 4- Adecuada 5-Muy adecuada	1<=X<=5 Cuando más cercano a 1 mejor	X=contable A=contable
Accesibilidad	Accesibilidad para usuarios con discapacidad	¿Qué cantidad de funciones están disponibles para usuarios con discapacidades físicas?	Contar el número de funciones a las que pueden acceder personas con discapacidad y contar total de funciones básicas	X=A/B Donde B>0 A= número de funciones a las que pueden acceder personas con discapacidad B= número total de funciones básicas	0<=X<= 1 Cuando más cercano a 1 es mejor	X=contable A=contable B=contable
	Idiomas admitidos	¿Qué cantidad de idiomas admite el software?	Contar la cantidad de idiomas que admite	X = A/B Donde B>0 A= Número de idiomas requeridos que admite. B= Cantidad de idiomas necesarios para ser soportados.	0<=X<= 1 Cuando más cercano a 1 mejor	X=contable A=contable B=contable

Portabilidad: se puede observar en la Tabla 6.

Tabla 6. Métrica de Calidad para Portabilidad

Subcaracterística	Métrica	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida
Adaptabilidad	Adaptabilidad en entorno software	¿Es el software lo suficiente capaz de adaptarse al entorno sistema software?	Contar el número de sistemas operativos soportados y contar número total de sistemas operativos considerados	X = 1-(A/B) Donde B>0 A= Numero de sistemas operativos no soportados B= número total de sistemas operativos considerados	0<=X<= 1 Cuando más cercano a 1 mejor	X=contable A=contable B=contable

4.4. Determinación de la ponderación a cada característica

La ponderación que se otorga a las características dependerá del nivel de importancia asignado. Estas deben ser asignadas a las características que conforman el modelo de calidad y la sumatoria no debe superar el 100%. En la Tabla 7 se presenta la ponderación asignada a las características.

Tabla 7. Ponderación de las características

Característica	Nivel de Importancia	Ponderación	Motivo de ponderación
Adecuación Funcional	M	20%	Se pondera con valor 20% porque en las etapas iniciales del aprendizaje de la programación, los usuarios pueden no necesitar todas las funcionalidades que ofrecen las herramientas más avanzadas.
Usabilidad	A	50%	Se pondera con valor 50% porque es significativo evaluar qué tan entendible, agradable y fácil de usar es el software, dado que los principiantes necesitan herramientas que sean intuitivas y fáciles de usar para poder concentrarse en aprender los conceptos básicos de la programación.
Portabilidad	M	30%	Se pondera con valor 30% porque en las etapas iniciales del aprendizaje de la programación se trabaja en entornos más acotados y no es tan necesaria la adaptación a diferentes entornos.

4.5. Determinación de la ponderación a cada subcaracterística

Se otorga una ponderación a las subcaracterísticas que conforman el modelo de calidad, considerando su incidencia dentro de la característica y la sumatoria no debe superar el 100% como se muestra en la Tabla 8.

Tabla 8. Ponderacion de las Subcaracterísticas

Característica	Subcaracterística	Nivel de importancia	Ponderación	Motivo de Ponderación
Adecuación funcional	Completitud funcional	M	100%	Se pondera con 100% porque es necesario que la herramienta de programación proporcione las funciones básicas definidas en el modelo para garantizar que los programadores principiantes tengan acceso a todas las funcionalidades necesarias para aprender y desarrollar sus habilidades de codificación. Por otra parte, es la única subcaracterística considerada para esta característica
Usabilidad	Reconocibilidad de la adecuación	A	30%	Se pondera con 30% porque seleccionar una herramienta de programación adecuada facilita el aprendizaje y el desarrollo de habilidades de programación y es importante evitar pérdidas de tiempo y esfuerzo en herramientas que no cumplen con las necesidades del usuario
	Aprendibilidad	A	10%	Se pondera con 10% dado que las herramientas de programación inicial por defecto están diseñadas para ser intuitivas y fáciles de usar, facilitando el proceso de aprendizaje
	Operabilidad	M	20%	Se pondera 20% dado que es un gran aporte que las

Característica	Subcaracterística	Nivel de importancia	Ponderación	Motivo de Ponderación
				herramientas cuenten con recursos de aprendizaje como tutoriales y comunidad de soporte que faciliten el acceso a sus funcionalidades para la realización de las tareas deseadas
	Protección contra errores del usuario	M	10%	Se pondera 10% dado que estas herramientas, al tener predefinidos bloques, estructuras y primitivas tienen menos posibilidades de que se cometan errores
	Estética de la interfaz de usuario	M	20%	Se pondera 20% dado que las interfaces agradables y atractivas permiten una mejor experiencia de usuario lo cual facilita el proceso de aprendizaje y hace que la programación sea más accesible para los principiantes
	Accesibilidad	A	10%	Se pondera 10% porque si bien la accesibilidad para discapacitados es un plus, esta situación representa un porcentaje muy bajo
Portabilidad	Adaptabilidad	M	100%	Se pondera 100% dado que la adaptabilidad es la única subcaracterística que contribuye a la característica Portabilidad.

4.6. Nivel de puntuación final

Se establece una escala de medición conceptual que se utilizará para determinar el resultado final del grado de cumplimiento de las características de calidad del modelo definido. Para el valor de medición, se toma como referencia la escala de clasificación que establece la

Norma ISO 15504 para el valor de medición, adaptando el criterio de decisión y el nivel de puntuación al objetivo de determinar la calidad del producto, como se muestra en la Tabla 9.

Tabla 9 Escala de Medición para resultado final de calidad

Criterio de decisión de la evaluación	Nivel de puntuación	Valor de medición
Mala calidad	Inaceptable	0-15%
Calidad Regular	Mínimamente aceptable	>15-50%
Buena calidad	Aceptable	>50-85%
Alta calidad	Cumple con los requisitos	>85%

4.7. Matriz de calidad

Una vez seleccionadas las características, subcaracterísticas y atributos de calidad con su respectiva ponderación, se propone la matriz de calidad para determinar la condición final de la herramienta evaluada como se muestra en la Tabla 10.

La matriz de calidad consta de las siguientes partes:

Característica: nombre de la característica.

Subcaracterística: nombre de la subcaracterística.

Métrica: nombre de la métrica.

Fórmula: formula de la métrica.

Valor de A: valor de elemento que conforma la métrica

Valor de B: valor de elemento que conforma la métrica

Valor de la métrica: valor que se obtiene a partir de la aplicación de la métrica

Ponderación de la subcaracterística: porcentaje de incidencia dentro de la característica.

Resultado: es el valor del producto del resultado de la métrica por el porcentaje de importancia de la subcaracterística.

Ponderación de la característica: porcentaje de importancia asignado por parte del evaluador.

Calificación: es el valor del producto de los valores parciales de las subcaracterística por el porcentaje de importancia de la característica.

Calificación final: es la suma de los valores finales de las características según el nivel de importancia asignado por el evaluador.

Procedimiento para la aplicación de la Matriz de Calidad por el Evaluador:

1. **Especificación de las Características Seleccionadas** detalla las características que ha seleccionado del Modelo de Calidad.
2. **Especificación de las Subcaracterísticas Seleccionadas** detalla las subcaracterísticas seleccionadas del Modelo de Calidad.
3. **Especificación de las Métricas Seleccionadas**: precisa las métricas que se han seleccionado para la evaluación.

Aplicación de los Criterios de Decisión de las Métricas:

4. **Especificación de la Ponderación:** En función a la importancia asignada a las características y subcaracterísticas seleccionadas, el evaluador asigna el porcentaje de ponderación que determinó a cada una de ellas. El evaluador se asegura de que la suma total de estas no exceda el 100%.
5. **Cálculo de la Ponderación de las Subcaracterísticas:** Una vez ingresados los valores que conforman la métrica (A y B), obtenido el valor de la misma, se aplica la ponderación de las subcaracterísticas, lo que da como resultado un valor parcial.
6. **Cálculo de la Calificación a Nivel de Característica:** Al resultado obtenido en el paso anterior, se aplica la ponderación de la característica, lo que da como resultado la calificación a nivel de característica.
7. **Cálculo del Resultado Final:** Finalmente, se realiza la suma de las ponderaciones a nivel de característica, lo que da como resultado el valor final para cada herramienta de programación inicial.

Aplicación de los Criterios de Decisión de la Evaluación:

Determinación de Cumplimiento de Niveles de Calidad: Finalmente, el evaluador aplica los criterios de decisión de la evaluación. Este proceso determina si la herramienta cumple o no con los niveles de calidad establecidos.

Tabla 10. Matriz de Calidad

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B) A=número de funciones que están incorrectas o que no fueron implementadas. B=número de funciones básicas requeridas.								
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B A= Numero de características que posee. B= Cantidad de características definidas como referencia.								
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B Cantidad de documentación para realizar las funcionalidades básicas. B= Cantidad de funcionalidades básicas.								

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B A= números funciones descripta correctamente B= Cantidad de funcionalidades básicas.								
	Operabilidad	claridad del mensaje	X=A/B A=número de mensajes implementados con explicaciones claras B= número total de mensajes implementados.								

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B Donde B>0 A= número de acciones y entradas de usuario que están protegidas para no causar ningún mal funcionamiento del sistema B= número de acciones y entradas de usuario que podrían protegerse para evitar que causen mal funcionamiento del sistema.								

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
	Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	X=A/B X = A Donde A puede ser: 1- Desagradable 2- Poco agradable 3- Neutro 4- Agradable 5-Muy agradable B=5								
		Legibilidad adecuada	X=A/B X=A/B Donde B>0 X = A Donde A puede ser: 1- Nada adecuada 2- Poco adecuada 3- Neutro 4- Adecuada 5-Muy adecuada B=5								

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B A= número de funciones a las que pueden acceder personas con discapacidad B= número total de funciones básicas.								
		Idiomas admitidos	X = A/B A= Número de idiomas requeridos que admite. B= Cantidad de idiomas necesarios para ser soportados.								
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X = 1-(A/B) Donde B>0 A= Numero de sistemas operativos no soportados B= número total de sistemas operativos considerados.								

Capítulo 5

Proceso de *Evaluación*

En el Capítulo 4 se ha definido un modelo de calidad para evaluar herramientas software utilizadas en la enseñanza inicial de la programación.

Este modelo de calidad incluye las características y subcaracterísticas que se consideraron más apropiadas para este tipo de software utilizado en la enseñanza de la programación por parte de docentes de las escuelas primarias y secundarias, principalmente. Incluye también las métricas que generarán los valores para cuantificar el grado de cumplimiento de los criterios de calidad definidos.

Para validar el modelo se procederá a la evaluación de un conjunto de herramientas siguiendo los pasos definidos en la norma ISO 25040, perteneciente a la familia ISO/IEC 25000. Esta norma consta de 13 procesos que se desarrollan en 5 etapas, que se muestran en la Tabla 11.

Tabla 11. ISO/IEC 25040 – Proceso de evaluación

Etapas	Procesos
1. Establecer los requisitos de la evaluación	1.1. Establecer el propósito de la evaluación. 1.2. Obtener los requisitos de calidad del producto. 1.3. Identificar las partes del producto que se deben evaluar. 1.4. Definir el rigor de la evaluación.
2. Especificar la evaluación	2.1. Seleccionar los módulos de evaluación. 2.2. Definir los criterios de decisión para las métricas. 2.3. Definir los criterios de decisión de la evaluación.
3. Diseñar la evaluación	3.1. Planificar las actividades de la evaluación.
4. Ejecutar la evaluación	4.1. Realizar las mediciones. 4.2. Aplicar los criterios de decisión para las métricas. 4.3. Aplicar los criterios de decisión de la evaluación.
5. Finalizar la evaluación	5.1. Revisar los resultados de la evaluación. 5.2. Crear el informe de evaluación. 5.3. Revisar la calidad de la evaluación y obtener feedback. 5.4. Tratar los datos de la evaluación.

5. Proceso de evaluación

Para realizar la evaluación del grado de cumplimiento de los criterios de calidad definidos en el modelo de calidad se seguirán cada una de las etapas establecidas en el proceso de evaluación definido en la norma ISO 25040.

Para la evaluación se considerarán las herramientas PilasBloques, Scratch y Gobstones, cuya selección se justificó en el Capítulo 3.

Etapa 1: Establecer los requisitos de la evaluación

En esta primera etapa se debe:

1.1- **Establecer el propósito de la evaluación***: Tarea destinada a indicar el propósito por el cual la organización quiere evaluar la calidad de un producto de software.*

El propósito de la evaluación consiste en determinar en qué medida las herramientas software seleccionadas cumplen con las características de calidad definidas en el modelo de calidad, que se muestra en la Tabla 10.

1.2- **Obtener los requisitos de calidad del producto**: *Requiere identificar las partes interesadas en el producto.*

En este caso, se considera la visión de los usuarios del producto, y se seleccionan las características y subcaracterísticas del modelo de calidad, definido y justificado en el capítulo 4, que se resume en la Tabla 12.

Tabla 12. Modelo de calidad propuesto con sus características y subcaracterísticas

Modelo de calidad definido	
Característica	**Subcaracterística**
Adecuación funcional	Completitud funcional
Usabilidad	Reconocibilidad de la adecuación
	Capacidad de aprendizaje
	Operabilidad
	Protección contra errores del usuario
	Estética de la interfaz de usuario
	Accesibilidad
Portabilidad	Adaptabilidad

1.3- **Identificar las partes del producto que se deben evaluar:**

Para esta evaluación, se considera la versión web de cada una de las herramientas seleccionadas:

Pilas Bloques (https://pilasbloques.program.ar/online/#/),

Scratch (https://scratch.mit.edu/projects/979618619/editor),

Gobstones (https://gobstones.github.io/gobstones-jr/).

1.4- **Definir el rigor de la evaluación**:

Considerando el propósito y el uso de los productos software, en este proceso no se consideran los factores de riesgo que establece la norma, como ser, el riesgo para la seguridad, el riesgo económico o el riesgo ambiental.

Etapa 2: Especificar la evaluación

En esta segunda etapa, se especifican los **módulos de evaluación** (métricas, herramientas y técnicas) junto con los criterios de decisión a aplicar.

2.1. Seleccionar los módulos de evaluación

Para la evaluación se utilizarán las métricas del modelo de calidad para evaluar el grado de cumplimiento en las características y subcaracterísticas, que se muestran en la Tabla 13.

Tabla 13 Métricas del Modelo de calidad propuesto

Característica	Subcaracterísticas	Métricas	Fórmula
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B) Donde B>0 A=número de funciones que están incorrectas o que no fueron implementadas. B=número de funciones básicas requeridas.
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B Donde B>0 A= Numero de características que posee. B= Cantidad de características definidas como referencia.
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B Donde B>0 A= Cantidad de documentación para realizar las funcionalidades básicas. B= Cantidad de funcionalidades básicas.

Característica	Subcaracterísticas	Métricas	Fórmula
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B Donde B>0 A= números funciones descripta correctamente B= Cantidad de funcionalidades básicas
	Operatividad	claridad del mensaje	X=A/B Donde B>0 A=número de mensajes implementados con explicaciones claras B= número total de mensajes implementados en las funciones básicas
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B Donde B>0 A= número de acciones y entradas de usuario que están protegidas para no causar ningún mal funcionamiento del sistema B= número de acciones y entradas de usuario que podrían protegerse para evitar que causen mal funcionamiento del sistema.
	Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	X=A/B Donde B>0 Donde A puede ser: 1- Desagradable 2- Poco agradable 3- Neutro 4- Agradable 5-Muy agradable
		Legibilidad adecuada	X=A/B Donde B>0 Donde A puede ser: 1- Nada adecuada 2- Poco adecuada 3- Neutro 4- Adecuada 5-Muy adecuada

Característica	Subcaracterísticas	Métricas	Fórmula
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B Donde B>0 A= número de funciones a las que pueden acceder personas con discapacidad B= número total de funciones básicas
		Idiomas admitidos	X = A/B Donde B>0 A= Número de idiomas requeridos que admite. B= Cantidad de idiomas necesarios para ser soportados
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X = 1-(A/B) Donde B>0 A= Numero de sistemas operativos no soportados B= número total de sistemas operativos considerados

2.2. Definir los criterios de decisión para las métricas

Se considera el nivel de importancia de las características y subcaracterísticas definidas en el modelo propuesto, así como la ponderación en el conjunto.

2.3. Definir los criterios de decisión de la evaluación

Se considera el nivel de importancia de las características y subcaracterísticas definidas en el modelo propuesto, así como la ponderación en el conjunto.

Etapa 3: Diseñar la evaluación

Las actividades de la evaluación deben ser planificadas, teniendo en cuenta la disponibilidad de los recursos (humanos, materiales) que puedan ser necesarios. Se debe considerar el presupuesto, los métodos de evaluación, las herramientas de evaluación, entre otros.

Para esta evaluación se han diseñado actividades de programación que involucran la utilización de conceptos básicos de programación, similares a las que un docente podría proponer a sus estudiantes, que serán implementadas en las plataformas Pilas Bloques, Scratch y Gobstones.

En esta etapa, con el objetivo de probar el modelo de calidad definido (Tabla 9), la tesista, docente con experiencia y vasta trayectoria en la enseñanza de la programación inicial, oficiará de "Evaluador experto" y desarrollará las actividades diseñadas en cada uno de los programas y aplicará las métricas definidas en el modelo de calidad.

3.1. Planificar las actividades de la evaluación:

Las actividades están diseñadas para que el estudiante se enfoque en el problema y no en la sintaxis, de esta manera, se facilita que se apropie de los conceptos fundamentales de la programación, evitando el obstáculo frustrante de los errores de sintaxis. Además, la actividad debe prever la utilización de las funcionalidades básicas que se establecieron en el modelo de calidad.

3.1.1. **Actividad Pilas Bloques:**

Para el diseño de la actividad (desafío en PilasBloques), se definió el tamaño del escenario, los obstáculos, y el objetivo a alcanzar y el personaje a utilizar, como se puede observar en la Figura 9.

Nombre de la actividad: Recolectando basura.

Desafío: Ayuda a Capy y Guyrá que quieren cuidar su humedal. Para lograrlo, van a limpiar los desechos que dejaron tirados los turistas. Ayúdalos a recoger todas las latas que se encuentran tiradas.

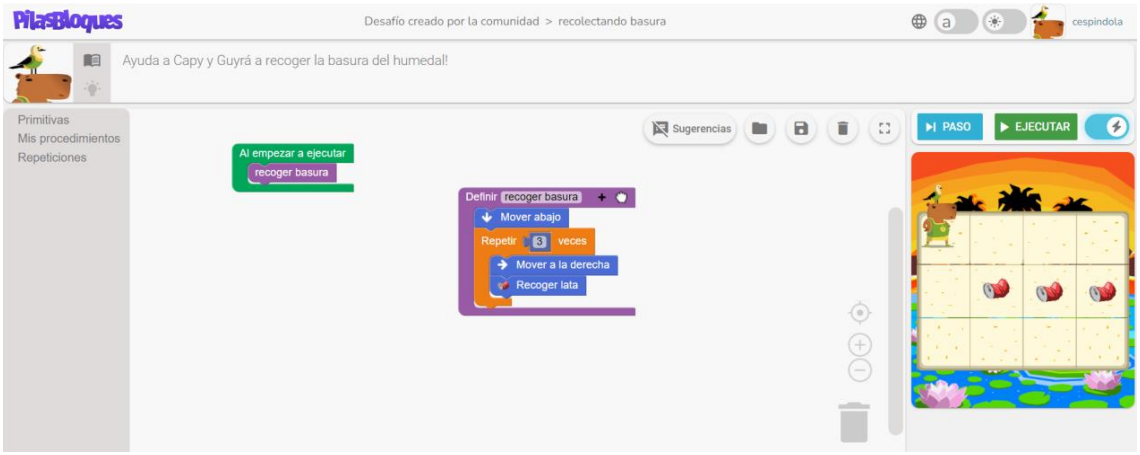

Figura.9. Entorno de Pilas Bloques
Desafío: Recolectando basura

3.1.2. **Actividad Scratch:**

Para el diseño de la actividad se eligió el fondo, los objetos y los comandos a utilizar, y los objetivos a alcanzar.

Nombre de la actividad: Instalando juegos:

Desafío: El autómata tiene que instalar un videojuego en las tres computadoras de la biblioteca. Para ello, debe encender cada computadora, ingresar la contraseña (ABC), cargar el juego y finalmente apagar la máquina, como se muestra en la Figura. 10.

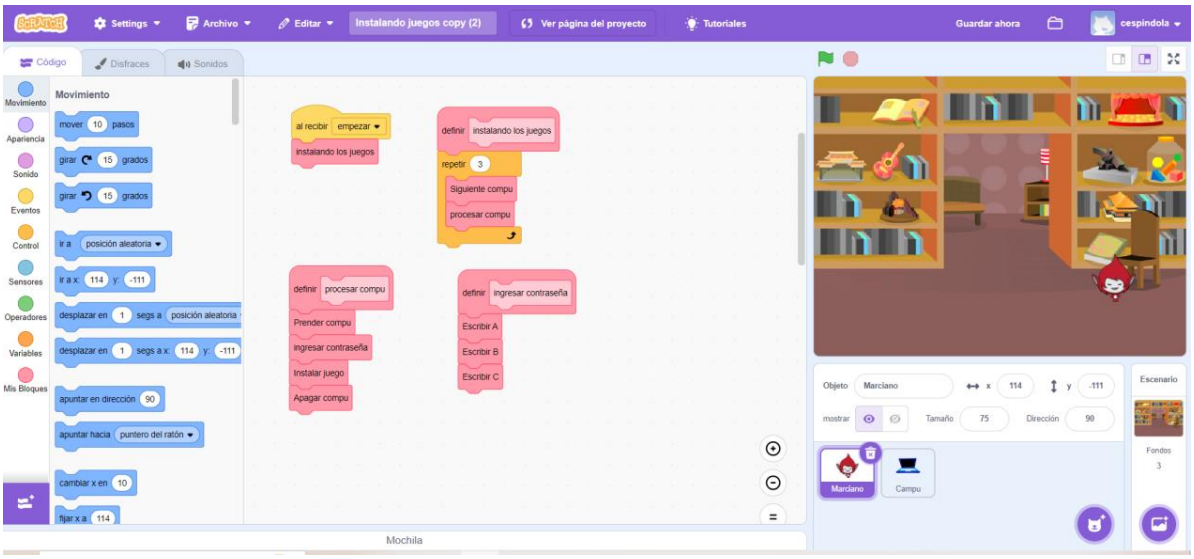

Figura.10. Entorno de Scratch
Actividad: Instalando juegos

3.1.3. **Actividad Gobstones:**

Para el diseño de la actividad se eligió el fondo, los objetos y los comandos a utilizar, y los objetivos a alcanzar.

Nombre de la actividad: Instalando bolitas.

Desafío: Esta actividad consiste en colocar en el tablero 3 bolitas rojas de manera consecutiva, para ellos debe utilizar un procedimiento **colocarbolitas**, que realice la colocación de las 3 bolitas rojas.

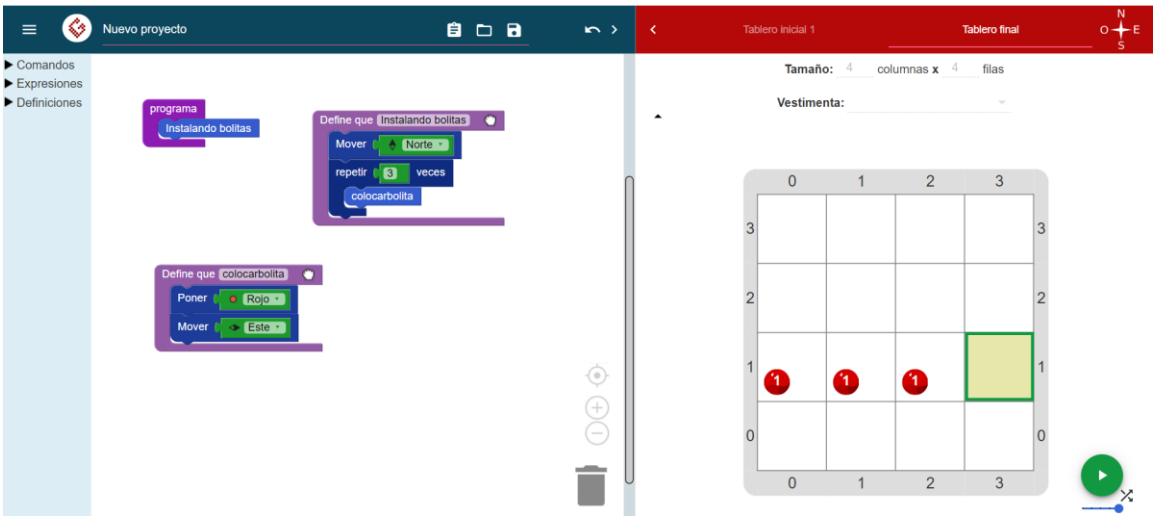

Figura. 11. Entorno de Gobstones
Actividad: Instalado bolitas

Etapa 4: Ejecutar la evaluación

4.1 Realizar las mediciones

Para realizar las mediciones se desarrollaron las actividades diseñadas en cada una de las herramientas. El evaluador experto, siguiendo la matriz de calidad donde están especificadas cada una de las métricas, fue otorgando valor a cada una de ellas, en una planilla de cálculo que determina el valor de la métrica.

En la Tabla 14 se muestran los valores de las métricas obtenidos para PilasBloques, en la Tabla 15 los valores obtenidos para Scratch y en la Tabla 16 los valores obtenidos para Gobstones.

4.1.1 **Cálculo de las métricas Pilas Bloques.** Se puede observar en la Tabla14.

Tabla 14. Valor de las métricas para la herramienta Pilas Bloques

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	1	12	0,92
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B	3	7	0,43
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B	10	12	0,83
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B	10	12	0,83
	Operabilidad	Claridad del mensaje	X=A/B	9	12	0,75
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1,00
	Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	X=A/B	5	5	1,00
		Legibilidad adecuada	X=A/B	4	5	0,80
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B	12	12	1,00
		Idiomas admitidos	X=A/B	3	3	1,00
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X = 1-(A/B)	0	3	1,00

4.1.2 **Cálculo de las métricas Scratch.** Se puede observar en la Tabla 15

Tabla 15. Valor de las métricas para la herramienta Scratch

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	3	12	0,75
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B	6	7	0,86
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B	6	12	0,50
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B	6	12	0,5
	Operatividad	Claridad del mensaje	X=A/B	6	12	0,5
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1
	Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	X=A/B	4	5	0,8
		Legibilidad adecuada	X=A/B	3	5	0,6
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B	8	12	0,67
		Idiomas admitidos	X = A/B	3	3	1
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X = 1-(A/B)	0	3	1

4.1.3 **Cálculo de las métricas Gobstones.** Se puede observar en la Tabla 16.

Tabla 16. Valor de las métricas para la herramienta Gobstones

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	5	12	0,58
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B	4	7	0,57
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B	3	12	0,25
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B	3	12	0,25
	Operatividad	Claridad del mensaje	X=A/B	2	12	0,17
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1
	Estética	Personalización de la apariencia de la interfaz del usuario	X=A/B	4	5	0,8
		Legibilidad adecuada	X=A/B	4	5	0,8
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B	0	12	0
		Idiomas admitidos	X = A/B	1	3	0,33
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X =1-(A/B)	0	3	1

4.2 Aplicar los criterios de decisión para las métricas

En este paso, se asigna el valor de la ponderación a cada subcaracterística, tal como se especifica en la Matriz de Calidad, y se calcula la calificación parcial y la calificación final para el producto específico. Las Tablas 17, 18 y 19, muestran los valores del cálculo para cada uno de los entornos de programación.

Tabla 17. Matriz de calidad herramienta Pilas Bloques

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	1	12	0,92	100	92%	20	18%	
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y	X = A/B	3	7	0,43	30	13%	50	6%	
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B	10	12	0,83	5	4%		2%	
		Efectividad de la documentación del	X=A/B	10	12	0,83	5	4%		2%	
	Operabilidad	claridad del mensaje	X=A/B	9	12	0,75	20	15%		8%	**85%**
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1,00	10	10%		5%	
	Estética de la interfaz de usuario	Personalización de la apariencia de la	X=A/B	5	5	1,00	10	10%		5%	
		Legibilidad	X=A/B	4	5	0,80	10	8%		4%	
	Accesibilidad	Accesibilidad para usuarios con	X=A/B	12	12	1,00	5	5%		3%	
		Idiomas admitidos	X = A/B	3	3	1,00	5	5%		3%	

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X = 1-(A/B)	0	3	1,00	100	100%	30	30%	

Tabla 18. Matriz de calidad herramienta Scratch

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	3	12	0,75	100	75%	20	15%	**82%**
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B.	6	7	0,86	30	26%	50	13%	
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B	6	12	0,50	10	5%		3%	
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B	6	12	0,5	10	5%		3%	
	Operatividad	Claridad del mensaje	X=A/B	6	12	0,5	10	5%		3%	
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1	10	10%		5%	
	Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	X=A/B	4	5	0,8	10	8%		4%	
		Legibilidad adecuada	X=A/B	3	5	0,6	10	6%		3%	

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B	8	12	0,67	5	3%		2%	
		Idiomas admitidos	X = A/B	3	3	1	5	5%		3%	
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X=1-(A/B)	0	3	1	100	100%	30	30%	

Tabla 19. Matriz de calidad herramienta Gobstones

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	5	12	0,58	100	58%	20	12%	
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B	4	7	0,57	30	17%	50	9%	
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B,	3	12	0,25	10	3%		1%	
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B	3	12	0,25	10	3%		1%	**67%**
	Operatividad	Claridad del mensaje	X=A/B	2	12	0,17	10	2%		1%	
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1	10	10%		5%	
	Estética de la interfaz de usuario	Personalización de la apariencia	X=A/B	4	5	0,8	10	8%		4%	

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
		de la interfaz del usuario									
		Legibilidad adecuada	X=A/B	4	5	0,8	10	8%		4%	
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B	0	12	0	5	0%		0%	
		Idiomas admitidos	X = A/B	1	3	0,33	5	2%		1%	
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X=1-(A/B)	0	3	1	100	100%	30	30%	

4.3 Aplicar los criterios de decisión de la evaluación

Aplicando los criterios de evaluación a las herramientas de programación inicial se obtienen los resultados que se observan en la Tabla 20.

Tabla 20. Criterios decisión de la evaluación

Herramienta de programación	Valor de la medición	Criterio de evaluación
Pilas Bloques	**85**	**Buena calidad**
Scratch	**82**	**Buena calidad**
Gobstones	**67**	**Buena calidad**

Etapa 5: Finalizar la evaluación

5.1. Revisar los resultados de la evaluación

Después de aplicar las métricas y criterios de calidad establecidos en la Matriz de calidad a cada producto software, se analizan los resultados.

5.1.1. Resultados de la evaluación de las subcaracterísticas

Tabla 21. Resultados de la evaluación de las subcaracterísticas

Característica	Subcaracterísticas	Ponderación	PilasBloques	Scratch	Gobstones
Adecuación Funcional	**Completitud Funcional**	100%	92%	75%	58%
Usabilidad	**Reconocibilidad de la adecuación**	30%	13%	26%	17%
	Capacidad de aprendizaje	10%	4%	5%	3%
			4%	5%	3%
	Operabilidad	20%	15%	5%	2%
	Protección contra errores del usuario	10%	10%	10%	10%
	Estética de la interfaz de usuario	20%	10%	8%	8%
			8%	6%	8%
	Accesibilidad	10%	5%	3%	0%
			5%	5%	2%
Portabilidad	**Adaptabilidad**	100%	100%	100%	100%

A continuación se realiza el análisis de los resultados de las subcaracteristicas de las 3 caracteristicas seleccionadas: 1) Adecuacion Funcional, 2) Usabilidad, 3) Portabilidad

1. Adecuación Funcional

a) **Completitud Funcional**: Al representar gráficamente los valores obtenidos por las herramientas, en la figura 12 se puede observar que Pilas Bloques presenta un alto grado de cumplimiento de esta subcaracterística, lo que indica que realiza satisfactoriamente en gran medida las funcionalidades básicas evaluadas. Le sigue Scratch, con un cumplimiento un poco menor, pero superando a Gobstones, el entorno que muestra la menor completitud funcional entre las herramientas evaluadas.

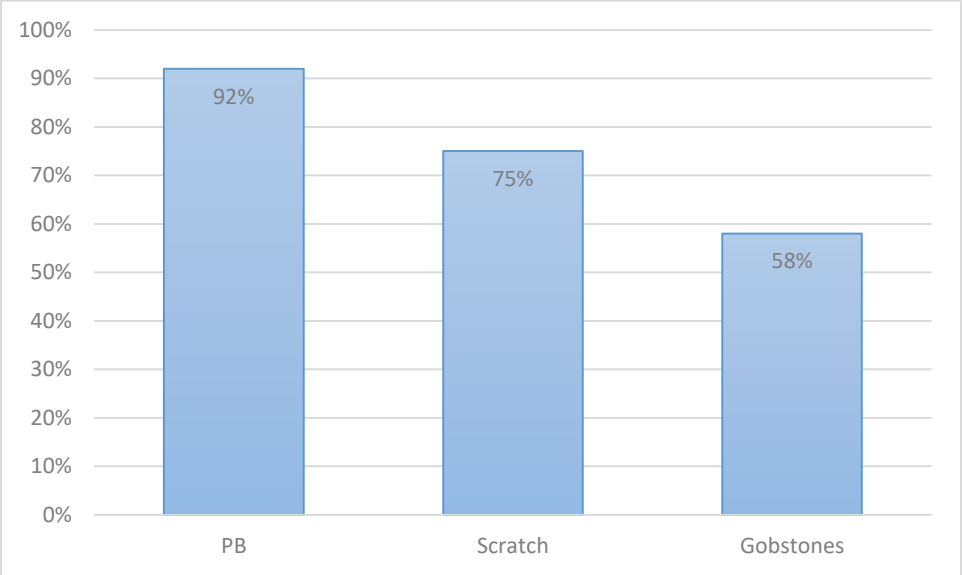

Figura 12. Completitud Funcional: grado de cumplimiento en cada herramienta

2. Usabilidad

a) **Reconocibilidad de la Adecuación**: Partiendo de tabla 21 y tomando como base la ponderación asignada a esta subcaracterística en el modelo propuesto (30%), al normalizar los valores y representarlos gráficamente se puede observar en la figura 13 el grado de cumplimiento de cada herramienta de esta subcaracterística.

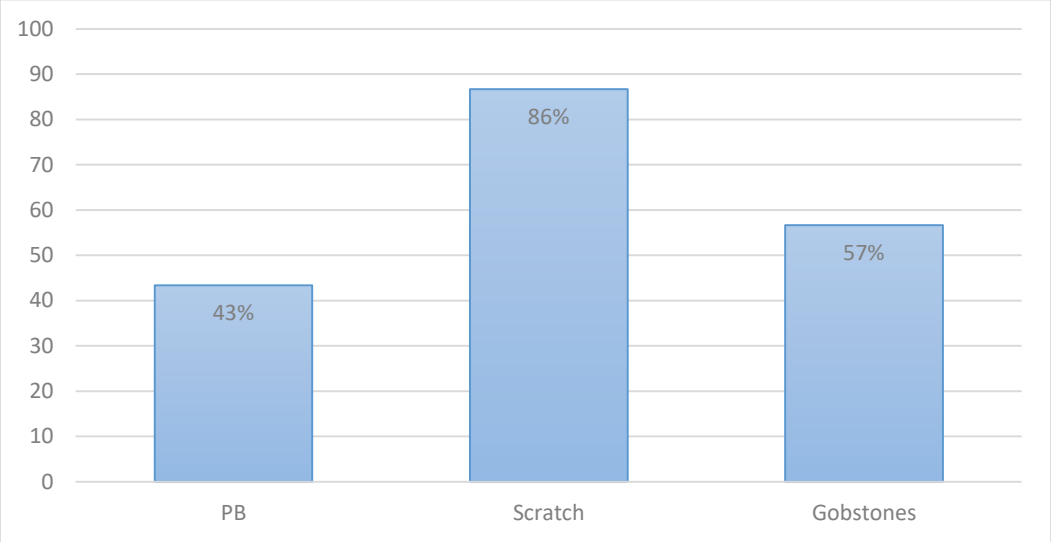

Figura 13 Reconocibilidad de la adecuación: grado de cumplimiento en cada herramienta

Los resultados reflejan que Scratch es el entorno que cumple en mayor medida la reconocibilidad de la adecuación. La alta puntuación indica que Scratch es efectivo en hacer que en su uso el entorno sea intuitivamente reconocible para los usuarios.

Gobstones presenta un cumplimiento considerablemente menor que Scratch, pero aún superior a Pilas Bloques. Con este valor, puede decirse que Gobstones proporciona propiedades que permiten a los usuarios reconocer y comprender sus características en gran medida.

Pilas Bloques presenta la puntuación más baja, lo cual significa que, aunque ofrece propiedades útiles para la programación visual, estas no son tan fácilmente reconocibles o comprensibles en comparación con Scratch y Gobstones.

b) **Capacidad de Aprendizaje**: A partir de los valores de la tabla 21 y tomando como base la ponderación asignada a esta subcaracterística en el modelo propuesto (10%), al normalizar los valores y representarlos gráficamente se observa en la figura 14 el grado de cumplimiento de cada herramienta.

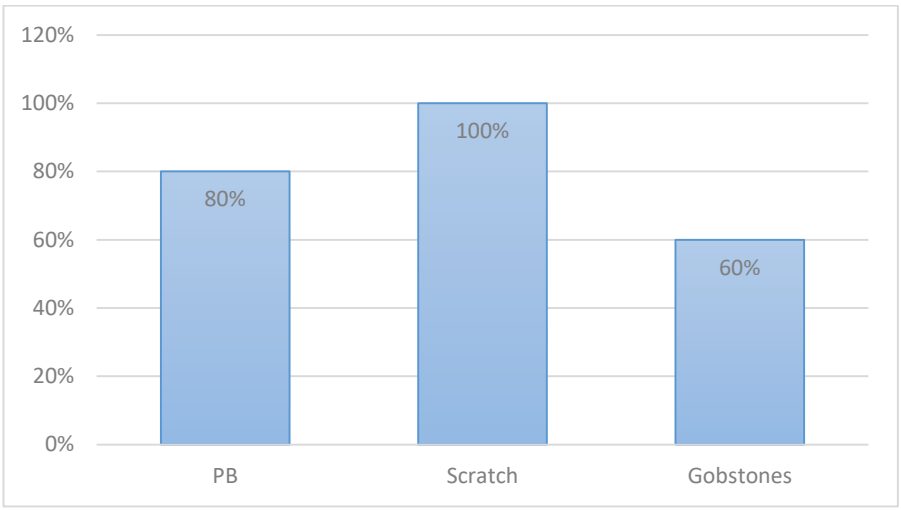

Figura 14. Capacidad de aprendizaje: grado de cumplimiento en cada herramienta

Los resultados muestran que Scratch es el entorno que mejor refleja la capacidad de aprendizaje, dado que los usuarios pueden fácilmente encontrar respuestas a sus preguntas y solucionar los problemas gracias a la documentación que brinda.

Pilas Bloques ofrece un cumplimiento menor, pero de igual manera indica una buena capacidad de aprendizaje. Gobstones, con la puntuación más baja, sugiere que los usuarios podrían encontrar que la información proporcionada es insuficiente o menos accesible.

c) **Operabilidad**: considerando los valores de la tabla 21 y la ponderación de esta subcaracterística en el modelo propuesto (20%), al normalizar los valores, la figura 15 ilustra el nivel de cumplimiento de cada herramienta.

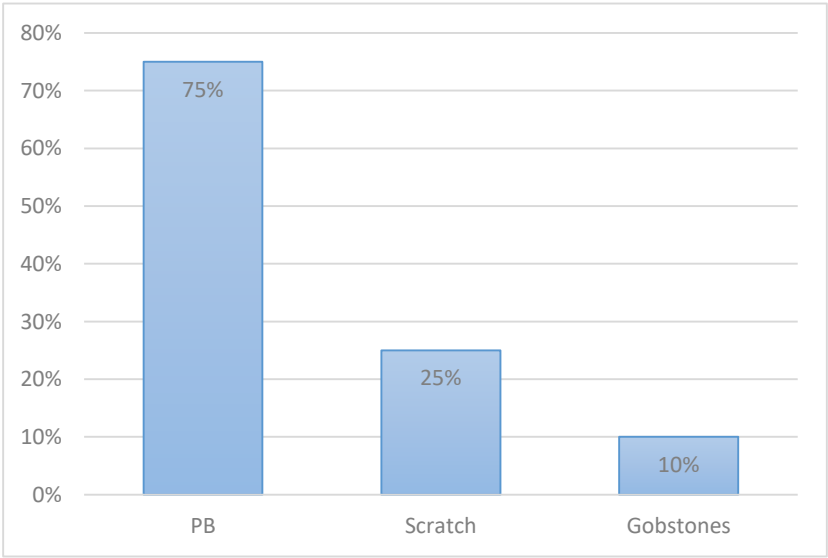

Figura 15. Operabilidad: grado de cumplimiento en cada herramienta

Pilas Bloques es el entorno que presenta la puntuación más alta en términos de operatividad, es decir, es la herramienta que brinda la mayor claridad en los mensajes.

Scratch presenta un cumplimiento mucho menor, lo que significa que su operabilidad podría generar dificultades en la interacción con la herramienta y el aprendizaje

Gobstones, con una puntuación muy baja, presenta serias deficiencias en cuanto a operatividad en comparación con Pilas Bloques y Scratch, lo que puede dificultar la experiencia del usuario y el aprendizaje.

d) **Protección Contra Errores del Usuario**: Considerando los valores de la tabla 21 y tomando como base la ponderación asignada a esta subcaracterística en el modelo propuesto (10%), al normalizar los valores y representarlos gráficamente se observa en la figura 16 el grado de cumplimiento de cada herramienta.

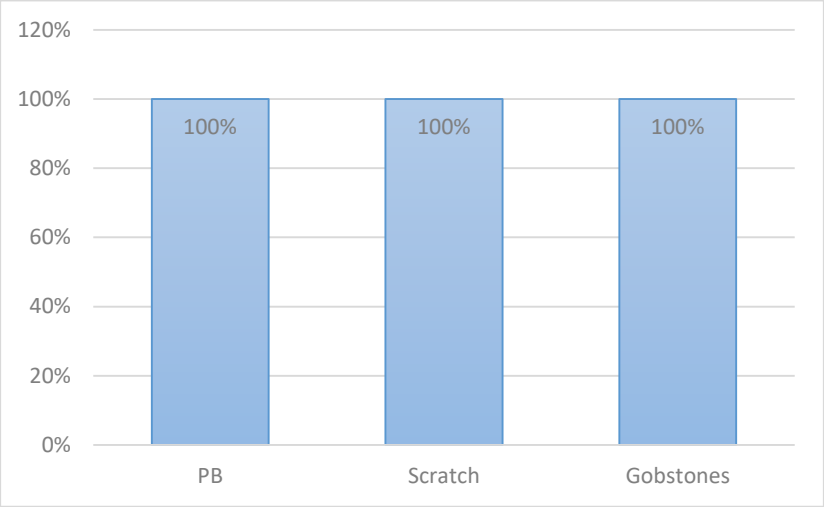

Figura 16. Protección contra errores del Usuario: grado de cumplimiento en cada herramienta

Pilas Bloques, Scratch y Gobstones cumplen con el 100% de las funcionalidades evaluadas. Esto indica que las tres herramientas presentan un nivel similar de robustez en términos de prevención y manejo de errores del usuario. Los desarrolladores de estas herramientas han implementado medidas efectivas para ayudar a los usuarios a evitar errores y utilizar las herramientas de manera eficiente, lo que contribuye a una experiencia de usuario más estable y fiable.

e) **Estética de la interfaz de usuario:** Con los valores de la tabla 21 y tomando como referencia la ponderación asignada a esta subcaracterística en el modelo propuesto (20%), al normalizar los valores y representarlos como muestra la figura 17, se observa el grado de cumplimiento de cada herramienta.

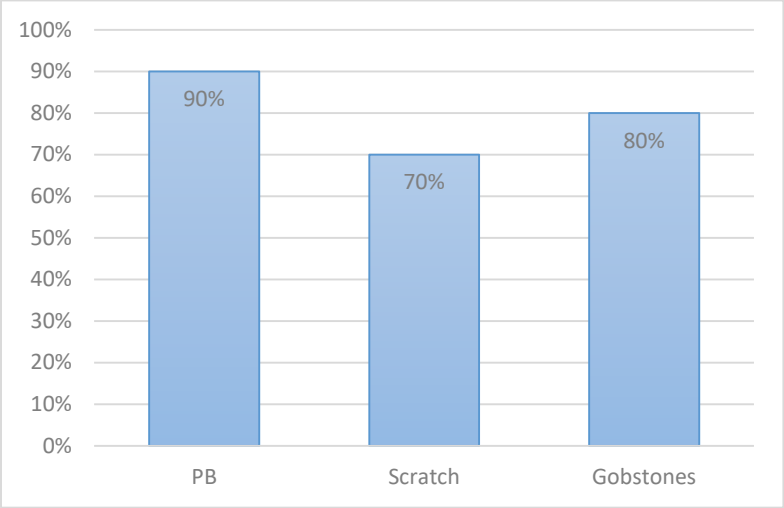

Figura 17. Estética de la interfaz de usuario: grado de cumplimiento en cada herramienta

Los resultados reflejan que Pilas Bloques es el entorno que cumple en mayor medida la estética de la interfaz de usuario, es decir, permite un alto grado de personalización en la apariencia de la interfaz y una alta legibilidad.

Gobstones cumple con esta subcaracterística, pero con algunas limitaciones en relación a Pilas Bloques. Scratch es el entorno con la puntuación más baja en relación a las dos herramientas, esto sugiere que tanto la personalización de la apariencia como la legibilidad podrían mejorarse para ofrecer una experiencia más agradable y accesible a los usuarios.

f) **Accesibilidad**: Tomando como referencia la tabla 21 y la ponderación asignada a esta subcaracterística en el modelo propuesto (10%), al normalizar los valores y representarlos como muestra la figura 18, se observa el grado de cumplimiento de cada herramienta

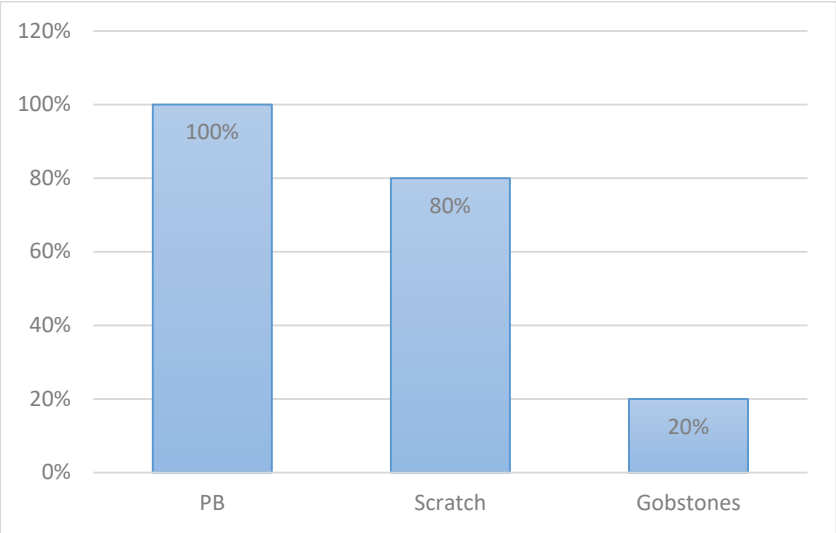

Figura 18. Accesibilidad: grado de cumplimiento en cada herramienta

Pilas Bloques es un entorno con excelente cumplimiento en accesibilidad. Esta herramienta es compatible con el software NVDA, proporciona una experiencia accesible para personas con discapacidades visuales, además, ofrece soporte para múltiples idiomas, lo que facilita el aprendizaje y el uso de la herramienta sin barreras lingüísticas, permitiendo que un mayor número de alumnos pueda beneficiarse de ella.

Scratch, con un cumplimiento menor, es compatible con el software NVDA, aunque algunos aspectos podrían no estar tan optimizados como en Pilas Bloques**.** También soporta una amplia gama de idiomas, facilitando su uso en diversos contextos lingüísticos.

Gobstones, con el cumplimiento más bajo, revela deficiencias en accesibilidad. Presenta limitaciones de compatibilidad con el software de asistencia como NVDA y ofrece un soporte muy limitado en cuanto a idiomas. Esto puede dificultar el uso de la herramienta.

3. Portabilidad

a) **Adaptabilidad**: Al representar gráficamente los valores obtenidos por las herramientas, en la figura 19 se puede observar que las 3 herramientas alcanzaron el mayor grado de cumplimiento. Esto indica que las plataformas se han desarrollado para ser accesibles en los principales sistemas operativos. En consecuencia, estas herramientas demuestran una alta portabilidad y una excelente capacidad de adaptación a diferentes entornos de software

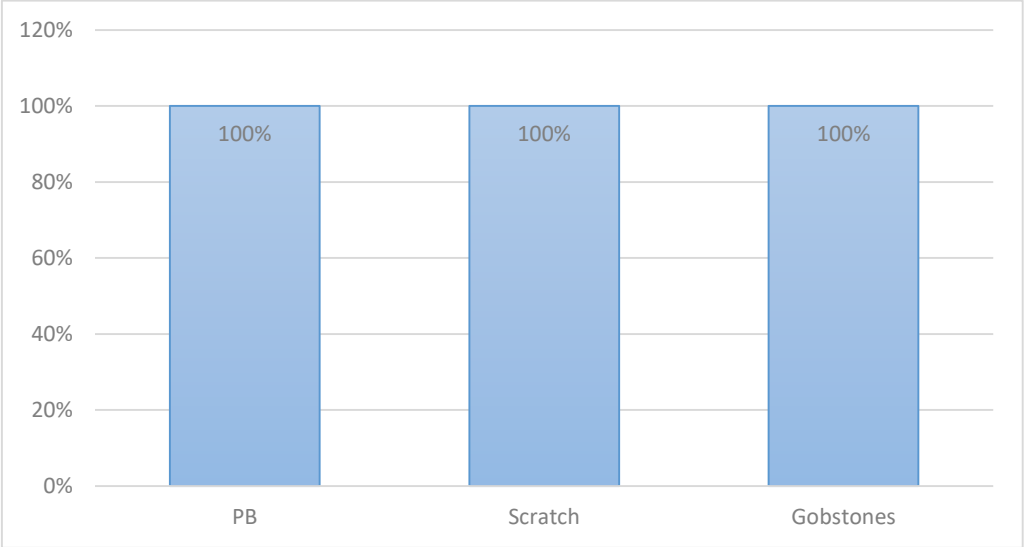

Figura 19. Adaptabilidad: grado de cumplimiento en cada herramienta

5.1.2. **Resultados de la evaluación de las características**

En la Tabla 22 se exponen los resultados de la evaluación de las características en cada uno de los productos software.

Tabla 22. Resultados de la evaluación de las Características

Característica	Ponderación	PB	Scratch	Gobstones
Adecuación funcional	20%	18%	15%	12%
Usabilidad	50%	37%	36,50%	25%
Portabilidad	30%	30%	30%	30%
Total	100%	85%	82%	67%

A continuación, se realiza una representación gráfica de los valores obtenidos y un análisis del grado de cumplimiento de la característica para cada herramienta.

Adecuación Funcional: tomando como base la ponderación (20%), se observa en la figura 20 el cumplimiento de esta característica.

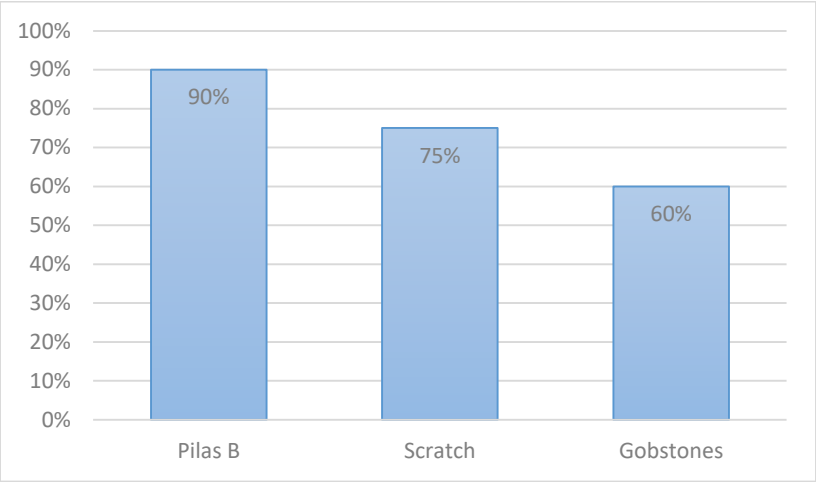

Figura 20. Adecuación Funcional: grado de cumplimiento en cada herramienta

Pilas Bloques presenta un alto grado de cumplimiento de esta característica, lo que indica una sólida completitud funcional. Esto significa que cumple en gran medida con los requisitos necesarios para la enseñanza de la programación, proporcionando una amplia gama de funcionalidades.

Scratch, con un cumplimento menor, sugiere un cumplimiento con algunas limitaciones de esta característica, pero en mejor situación que Gobstones, que con un 60% presenta el menor grado de cumplimiento de la adecuación funcional.

Usabilidad: tomando como base la ponderación 50% asignada al modelo, se observan gráficamente los resultados en la figura 21.

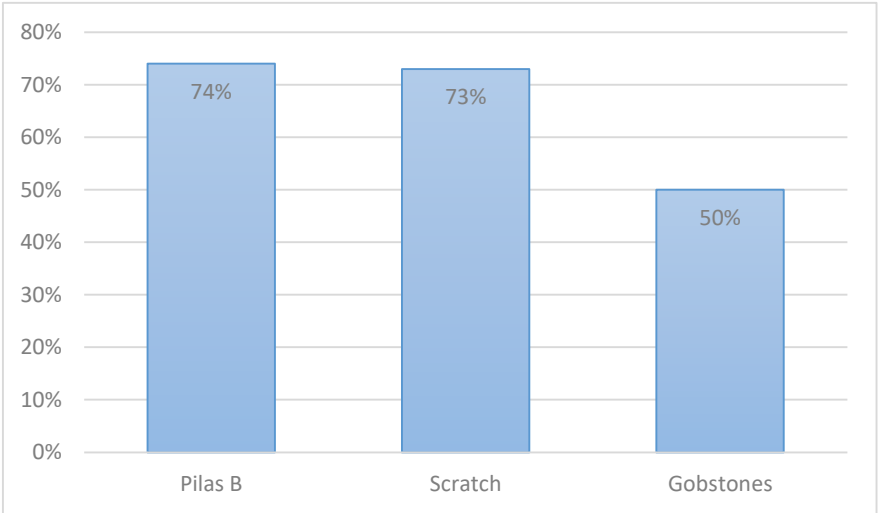

Figura 21. Usabilidad: grado de cumplimiento en cada herramienta

Pilas Bloques y Scratch muestran valores similares de cumplimiento de esta característica lo que indica que ambas herramientas ofrecen una buena experiencia de usuario. Gobstones con un cumplimiento bastante menor, sugiere que enfrenta mayores desafíos en términos de facilidad de uso, lo que podría afectar la efectividad y la satisfacción del usuario.

Portabilidad: tomando como base la ponderación 30% asignada al modelo, se observan gráficamente los resultados en la figura 22.

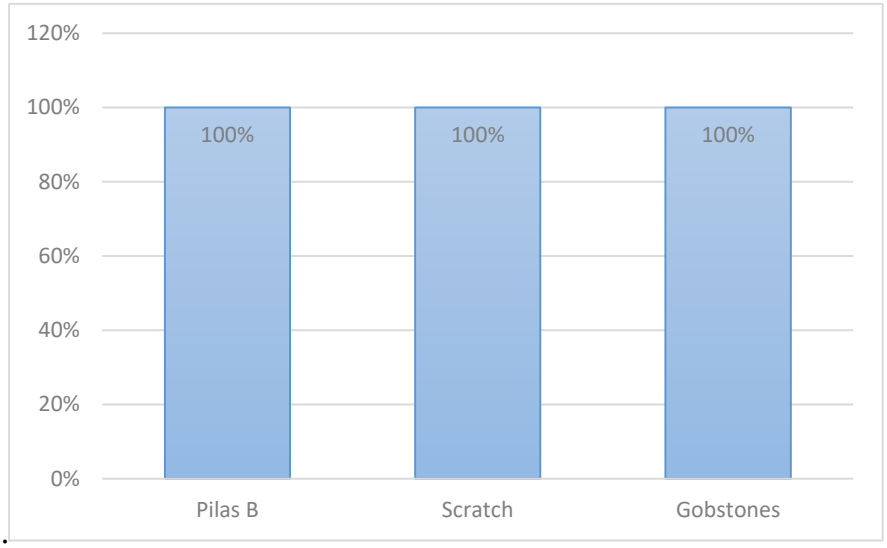

Figura 22. Portabilidad: grado de cumplimiento en cada herramienta

Pilas Bloques, Scratch y Gobstones alcanza un cumplimiento del 100%. Esto indica que las herramientas evaluadas son altamente portables y accesibles en una variedad de plataformas y dispositivos. Esta alta portabilidad asegura que los usuarios puedan trabajar en sus proyectos desde diferentes entornos sin preocuparse por problemas de compatibilidad, y que el aprendizaje realizado con estas herramientas se mantenga a lo largo del tiempo.

5.1.3. **Conclusión de la evaluación de las características**

Pilas Bloques: como se observa en figura 23 presenta un 90% de cumplimiento de la adecuación funcional**,** lo que indica una buena capacidad para satisfacer las necesidades con respecto a las funcionalidades básicas que se consideraron para la evaluación. En términos de usabilidad alcanza un 74% de cumplimiento, reflejando una buena experiencia de usuario, aunque con margen a mejoras en ciertos aspectos. Cumple en un 100% la característica de

portabilidad**,** destacándose por su capacidad para funcionar en diferentes plataformas y entornos.

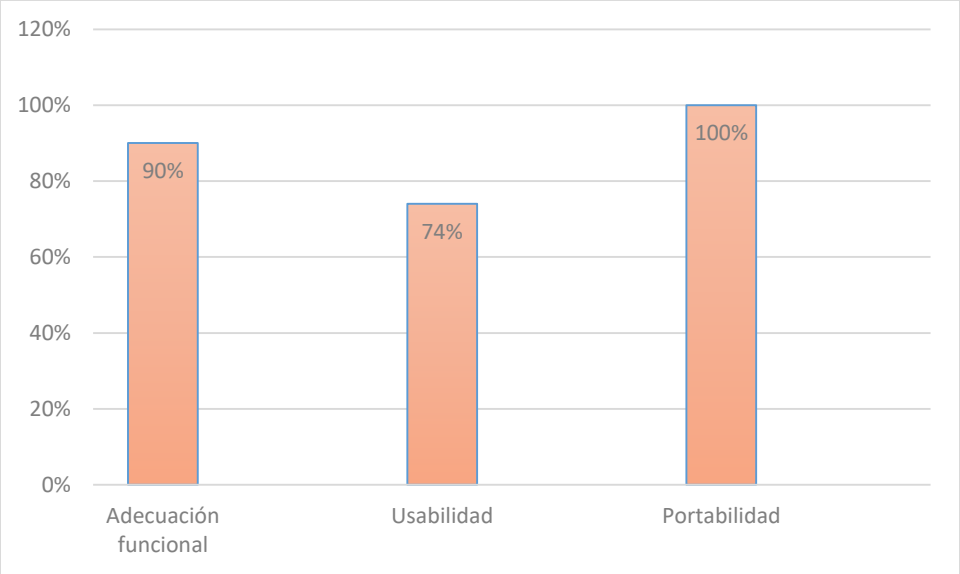

Figura 23 Resultados del modelo de Calidad para Pilas Bloques

Scratch: como se observa en la figura 24, presenta cumplimiento similar en cuanto adecuación funcional con un 75% y 73% en usabilidad, reflejando esto último una buena experiencia de usuario, aunque con margen a mejoras como Pilas Bloques. Presenta un 100% de cumplimiento en portabilidad, indicando que se adapta perfectamente a distintas plataformas.

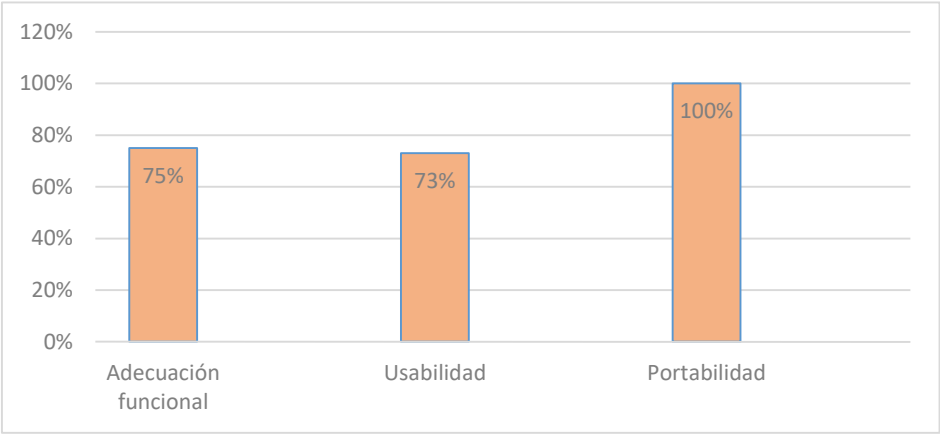

Figura 24. Resultados del modelo de Calidad para Scratch

Gobstones: como se observa en la figura 25, presenta un cumplimiento del 100% al igual que Pilas Bloques Scratch para la portabilidad.

En cuanto al cumplimiento de la adecuación funcional es más baja, que los anteriores entornos, lo que sugiere que la herramienta podría no satisfacer completamente todas las necesidades o expectativas funcionales. El cumplimiento de usabilidad es el más bajo de las 3, indicando que la herramienta podría ser más compleja de usar y aprender para los usuarios.

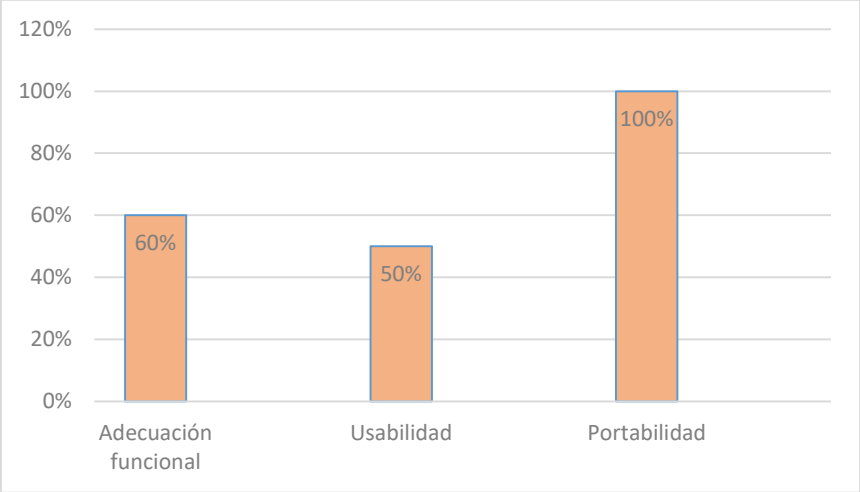

Figura 25. Resultados del modelo de Calidad para Gobstones

5.2. Informe de evaluación:

La norma ISO 25040 exige un informe final de la evaluación. En este caso, el informe está compuesto por los resultados de cada una de las etapas del proceso de evaluación que se ha descripto a lo largo de este capítulo. Para cumplimentar este ítem, el informe completo se encuentra en el Anexo II para no reiterar en este documento.

5.3. Acerca de la eficacia del Modelo de Calidad propuesto

La validación del modelo de calidad en la evaluación de herramientas software de enseñanza de la programación ha demostrado su eficacia dado que ha permitido un análisis comparativo sistemático de las características de estas herramientas, en particular en aspectos relevantes para la enseñanza, como son la adecuación funcional, la usabilidad y la portabilidad.

La información resultante de la evaluación permitirá guiar a los docentes en la selección de las herramientas más adecuadas para sus necesidades, reduciendo la incertidumbre en el proceso de elección mediante la aplicación de criterios claros y específicos.

El modelo es versátil y puede adecuarse para la evaluación de otras herramientas y otras características que resulten de interés, siguiendo el mismo esquema que se muestra en este trabajo.

Capítulo 6

Conclusiones y líneas futuras de investigación

Conclusiones y líneas futuras

La norma ISO/IEC 25000 es un estándar internacional que ofrece un marco para evaluar la calidad de productos software. En este trabajo se pudo comprobar su utilidad para evaluar la calidad de los entornos de programación, como PilasBloques, Scratch y Gobstones.

El estándar ISO/IEC 25010 proporciona un modelo de calidad genérico que facilita la clasificación de la calidad del producto software en términos de características y subcaracterísticas del software En el marco de este trabajo, las características y subcaracterísticas relevantes se seleccionaron específicamente para adaptarse al contexto de la evaluación de calidad del software que se utiliza para enseñanza de la programación para principiantes.

Por otro lado, la norma ISO/IEC 25040 describe un proceso de evaluación que proporciona un conjunto de pautas generales para realizar la evaluación de calidad del producto software. Además, la norma ISO/IEC 25023 ofrece un conjunto de métricas que son utilizadas para evaluar diferentes aspectos de la calidad del software. Estas métricas fueron adaptadas para permitir una evaluación detallada de su calidad, cuando se aplican a las herramientas de programación inicial.

En resumen, la aplicación de estos estándares permitió evaluar la calidad de las herramientas de programación inicial para asegurar que cumplen con un mínimo de requisitos de calidad preestablecidos.

Los resultados finales indican que Pilas Bloques, Scratch y Gobstones se clasifican como de Buena Calidad.

Pilas Bloques, en relación a los demás entornos, se destaca en el cumplimiento de varias subcaracterísticas de modelo propuesto. Esto permitirá al docente un conjunto de beneficios que contribuyen a un entorno

de aprendizaje más enriquecedor, optimizando la enseñanza de la programación y mejorando la experiencia.

Este modelo de calidad puede ayudar a los docentes a seleccionar las herramientas más adecuadas y garantizar que cumplan con los estándares de calidad necesarios. Además, proporciona una evaluación objetiva y detallada de la calidad de estas herramientas.

A futuro, con el conocimiento adquirido en este trabajo, se propone evaluar estas herramientas de programación desde la perspectiva de los usuarios, tanto docentes como

estudiantes, siguiendo las pautas de la ISO/IEC 25022 sobre Calidad en Uso. Esta evaluación podrá enriquecer la información sobre estas herramientas dado que surgirá de experiencias concretas de los docentes o estudiantes al usar estos entornos de programación. También podrá asociarse a esta evaluación, el nivel educativo, edad de los estudiantes, tipo de actividades, entre otras posibles.

Bibliografía

[1] F. Ripani, «Programación y robótica,» de *Programación y robótica: objetivos de aprendizaje para la educación obligatoria.*, Ed. Ministerio de Educación de la Nación Libro digital. ISBN 978-950-00-1200-3, 2017.

[2] F. Steve, «Real Sociedad de Londres.,» de *Shut down or restart? The way forward for computing in UK schools.*, Londres, UK, 2012.

[3] Fundación Sadosky, CC–2016 Una propuesta para refundar la enseñanza de la computación en las escuelas argentinas., Buenos Aires., 2013.

[4] (PCAST)., President's Council of Advisors in Science and Technology, «PCAST. Report to the President. Prepare and Inspire: K-12 Education in Science, Technology, Engineering, and Math (STEM) for America's Future.,» 2010.

[5] G. N. Dapozo, C. L. Greiner, R. H. Petris, M. V. Godoy and M. C. Espíndola, «"Enseñanza de programación en la universidad. Estrategia basada en programación por bloques".,» de *Libro "Innovation and Practice in Education". .I SBN 978-84-09- 09792-0. .*, Ciudad Real, España, CIATA.org Ciud, Junio 2019, pp. 89-99.

[6] H. C. Ahumada; D. Rivas;O. E. Quinteros; C.E. Gallardo; N. Contreras; M. Miranda; J. Favore; M. V. Póliche, «Programación por bloques en cátedras de ingeniería en informática.,» de *XXI Workshop de Investigadores en Ciencias de la Computación WICC 2019, Universidad Nacional de San Juan.*, San Juan, 2019.

[7] N. Rodriguez y G. Monjelat, «Repensando la programación como formación práctica en Ingeniería: Un estudio de caso en primer año.,» *chil. ing.*, vol. 26, nº 1, pp. 172-183., 2018.

[8] P. E. Martínez López, E. A. Bonelli., and F. A. Sawady, «El nombre verdadero de la programación. Una concepción de la enseñanza de la programación para la sociedad de la información.,» de *10mo Simposio de la Sociedad de la Información (SSI12), dentro de las 41ras Jornadas Argentinas de Informática (JAIIO '12), pp.1–23.*, 2012.

[9] J.M.Juran, «Made in USA: A Renaissance in Quality.,» *Harvard Business Review*, nº 71, pp. 42-50, 1993.

[10] K. Schwab, La cuarta revolución industrial., Barcelona. España: Debate edición, 2016.

[11] A. Rivas, «Un sistema educativo digital para la Argentina. Documento de trabajo N°165.,» CIPPEC, 2018.

[12] Ministerio de Educación y Deportes de la Nación, «Plan Argentina Enseña y Aprende. Anexo Resolución CFE Nº285/16. Plan Estratégico Nacional 2016-2021.1ra ed.,» 2016.

[13] Ministerio de Educación de la Nación, Competencias de Educación Digital. Colección Marcos Pedagógicos Aprender Conectados. 1ra ed., Libro digital. ISBN 978-950-00-1202-7, 2017.

[14] «Escuelas del Futuro. Ministerio de Educación de la Nación.,» 2017. [En línea]. Available: https://www.argentina.gob.ar/sites/default/files/dossier-23-59cbfd6633c30_0.pdf. [Último acceso: diciembre 2023].

[15] P.Martínez López, Las Bases Conceptuales de la Programación. Una nueva forma de aprender a programar., 1ra ed. ISBN: 978-987-33-4081-9, 2013.

[16] G. Dapozo, R. Petris, C. Greiner, A. Company, M.C. Espíndola, «Formación docente para incorporar la programación en las escuelas. Actas de las XXIV ISSN: 2531-06.,» de *Jornadas sobre Enseñanza Universitaria de la Informática (JENUI).*, Barcelona. España., 2018.

[17] R. S. Pressman, *Ingeniería de Software, un enfoque práctico.*, MCGRAW-HILL, 2010.7ma edición,, 2010.

[18] D, Galin, Software Quality: Concepts and Practice., IEEE COMPUTER SOCIETY, 2018.

[19] M. Callejas-Cuervo y A. C. Alarcón-Aldana, «Modelos de calidad del software, un estado del arte.,» *entramado*, Vols. %1 de %2vol. 13,, nº 1, pp. pp. 236–250,, jun. 2017..

[20] ISO25000:2005 Software Engineering, *Software productQuality Requirements and Evaluation (SQuaRE). Technical report, International Organization for Standardization.*.

[21] ISO 25010:2005, *Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. Technical report.*, International Organization for Standardization.

[22] J. Calabrese, R. Muñoz, A.Pasini, S. Esponda, M. Boracchia, P. Pesado, «Asistente para la evaluación de características de calidad de producto de software propuestas por ISO/IEC 25010 basado en métricas definidas usando el enfoque GQM.,» de *XXIII Congreso Argentino de Ciencias de la Computación.* , La Plata , 2017.

[23] ISO, *ISO/IEC 25023:2016. Systems and software engineering - Systems and software*..

[24] P. E. Martínez López, M J. Gómez, «¿Scratch, Python, o qué? Criterios para elegir un entorno para enseñar a programar a principiantes.,» de *JADiCC* , 2021.

[25] M. J. Gómez, *Aspectos de adquisición de lenguaje en la enseñanza de programación. Tesis de doctorado. FAMAF, UNC.*, 2020.

[26] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J.Bennedsen and J. Paterson, «Un estudio de la literatura sobre la enseñanza de la programación introductoria.,» *Boletín ACM SIGCSE*, vol. 39, nº 4, pp. 204-223., 2007.

[27] D. Weintrop y U. Wilensky, «Entre un bloque y una tipografía: Diseño y evaluación de entornos de programación híbridos.,» de *IDC 2017 - Actas de la Conferencia ACM 2017 sobre diseño de interacción y niños*, Stanford, Estados Unidos, 2017.

[28] C. G. Artiles Fontales, «Pilas Bloques. Aprende a programar jugando.,» *Observatorio de Tecnologia Educativa. Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado*, nº 18, 2019.

[29] A.Kuz, M. C. Ariste, «Análisis y revisión de softwares educativos para el aprendizaje de la programación en entornos lúdicos.,» *Tecné, Episteme y Didaxis: TED, Universidad Pedagógica Nacional; Facultad de Ciencia y Tecnología.*, nº 52, pp. 117-136, 2022.

[30] A.García Rodríguez, «Enseñanza de la programación a través de Scratch para el desarrollo del pensamiento computacional en educación básica secundaria.,» de *Revista Academia y Virtualidad.*, 2022.

[31] O.Angamarca, D. Andrade, «Enseñanza de programación a niños de edad escolar utilizando Scratch para mejora del razonamiento lógico.,» *Revista De Producción, Ciencias E Investigación*, vol. 6, nº 42, 2022.

[32] M.L. Cabra Páez, S.A. Ramírez Gamboa, «Desarrollo del pensamiento computacional y las competencias matemáticas en análisis y solución de problemas: una experiencia de aprendizaje con Scratch en la plataforma Moodle.,» *Revista Educación*, vol. 46, nº 1, 2022.

[33] P. Viale, C. Deco, «Introduciendo conocimientos sobre el pensamiento computacional en los primeros años de las carreras de ciencia, tecnología, ingeniería y matemáticas.,» *Energeia,*, vol. 17, nº 17, 2021.

Anexos

Anexo I: Cálculo de las métricas

Característica: Adecuación funcional

Métrica: Completitud de la implementación funcional

Se consideraron las doce funcionalidades básicas especificadas en el modelo de calidad propuesto.

Resultados: Como se puede observar en la Tabla1.

Tabla 1

N°	Funcionalidades	PB	Scratch	Gobstones
1	Crear un programa (desafío)	1	1	1
2	Abrir un programa	1	1	1
3	Editar ()	1	1	1
4	Guardar predeterminado	1	1	1
5	Guardar seleccionando ubicación	0	0	0
6	Descartar/eliminar	1	1	1
7	Escribir un enunciado	1	0	0
8	Escribir un título	1	0	0
9	Ejecutar	1	1	1
10	Ver el desafío	1	1	0
11	Compartir el desafío	1	1	0
12	Descargar la actividad	1	1	1
	Total	11	9	7

Característica: Usabilidad

Subcaracterística: Reconocibilidad de la adecuación

Se consideraron las propiedades especificadas en el modelo de calidad para esta subcaracterística.

Métrica: Medida de idoneidad y Reconocibilidad

Cálculos: Como se puede observar en la Tabla 2.

Tabla 2

N°	Características	PB	Scratch	Gobstones
1	Programar mediante arrastrar y soltar	1	1	1
2	Visualizar proyectos de otras personas	0	1	0
3	Introducir al paradigma de programación orientada a objetos	0	1	0
4	Introducir código mediante texto	0	0	1
5	Utilizar funciones con parámetros	1	1	1
6	Similar a entornos de programación profesionales	1	1	1
7	Conectar al exterior mediante sensores	0	1	0
	Total	3	6	4

Subcaracterística: capacidad de aprendizaje

Métrica: Integridad de la guía de usuarios
Para calcular esta métrica se consideraron las funcionalidades definidas en el modelo.

La integridad se relaciona con la capacidad de la herramienta para proporcionar guías completas y efectivas que ayuden a los usuarios a aprender y utilizar sus funcionalidades de manera exitosa.

En la Tabla 3 se puede observar sus cálculos.

Tabla 3

N°	Funcionalidades	PB	Scratch	Gobstones
1	Crear un programa (desafío)	1	1	0
2	Abrir un programa	0	1	0
3	Editar	1	0	0
4	Guardar predeterminado	1	1	1
5	Guardar seleccionando ubicación	0	0	0
6	Descartar/eliminar	1	0	1
7	Escribir un enunciado	1	0	0
8	Escribir un título	1	0	0
9	Ejecutar	1	1	1
10	Ver el desafío	1	0	0
11	Compartir el desafío	1	1	0
12	Descargar la actividad	1	1	1
	Total	10	6	3

Subcaracterística: capacidad de aprendizaje

Métrica: Efectividad de la documentación del usuario o ayuda del sistema
Para calcular esta métrica se consideraron las funcionalidades definidas en el modelo.

En la Tabla 4 se puede observar sus cálculos.

Tabla 4

N°	Funcionalidades	PB	Scratch	Gobstones
1	Crear un programa (desafío)	1	1	0
2	Abrir un programa	0	1	0
3	Editar	1	0	0
4	Guardar predeterminado	1	1	1
5	Guardar seleccionando ubicación	0	0	0
6	Descartar/eliminar	1	0	1
7	Escribir un enunciado	1	0	0
8	Escribir un título	1	0	0
9	Ejecutar	1	1	1

N°	Funcionalidades	PB	Scratch	Gobstones
10	Ver el desafío	1		0
11	Compartir el desafío	1	1	0
12	Descargar la actividad	1	1	1
	Total	10	6	3

Subcaracterística: Operatividad

Métrica: claridad del mensaje
Para calcular esta métrica se consideraron las funcionalidades definidas y los mensajes en las mismas.

En la Tabla 5 se puede observar sus resultados.

Tabla 5

N°	Funcionalidades	PB	Scratch	Gobstones
1	Crear un programa (desafío)	0	1	0
2	Abrir un programa	1	1	0
3	Editar	0	0	0
4	Guardar predeterminado	1	1	0
5	Guardar seleccionando ubicación	0	0	0
6	Descartar/eliminar	1	0	1
7	Escribir un enunciado	1	0	0
8	Escribir un título	1	0	0
9	Ejecutar	1	1	1
10	Ver el desafío	1	0	0
11	Compartir el desafío	1	1	0
12	Descargar la actividad	1	1	0
	Total	9	6	2

Subcaracterística: Protección contra errores del usuario

Métrica: Prevención del uso incorrecto
Para calcular esta métrica se consideraron las siguientes acciones: Estructuras Repetitivas y uso de parámetros en procedimientos.

En las herramientas de programación en bloque se denomina “agujero o hueco” al lugar donde se encuentra el número (parámetro) que se presenta en la estructura repetitiva, así como en los procedimientos.

En Pilas Bloques y en Gobstones, en la estructura repetitiva, se ingresaron datos incorrectos, (por ejemplo, un carácter) y se pudo observar que en el hueco toma un color rojo y al hacer clic afuera del mismo automáticamente se borra, como se observa en la Figura 1.

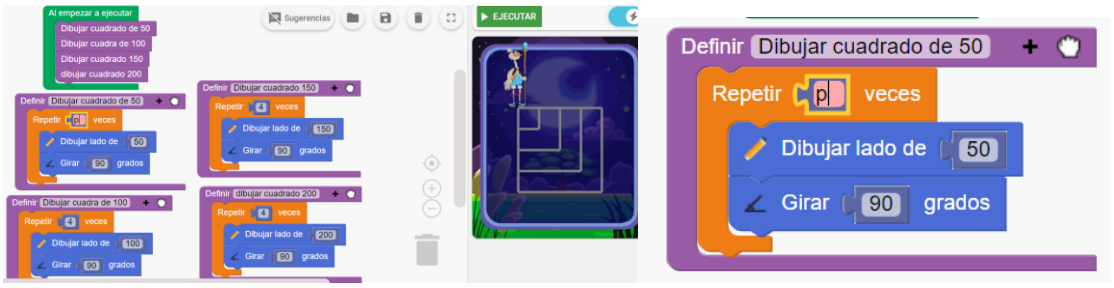

Figura 1

Scratch no permite el ingreso de un dato carácter o especial, como se observa en la Figura2.

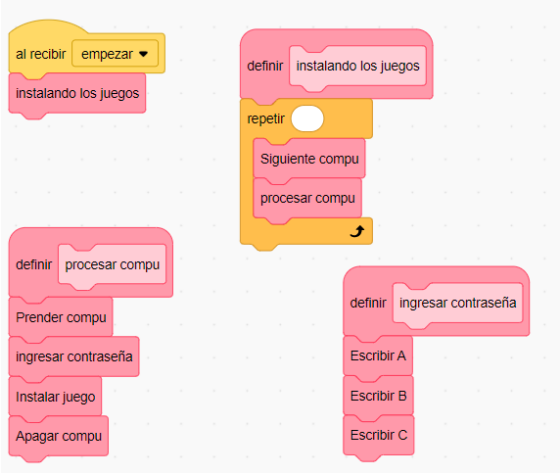

Figura 2.

Parámetros: Como se observa en la figura3, al igual que la estructura repetitiva, Pilas Bloques no permite el ingreso de dato de tipo carácter ni especial, como se observa en la Figura 4.

- **Pilas Bloques:**

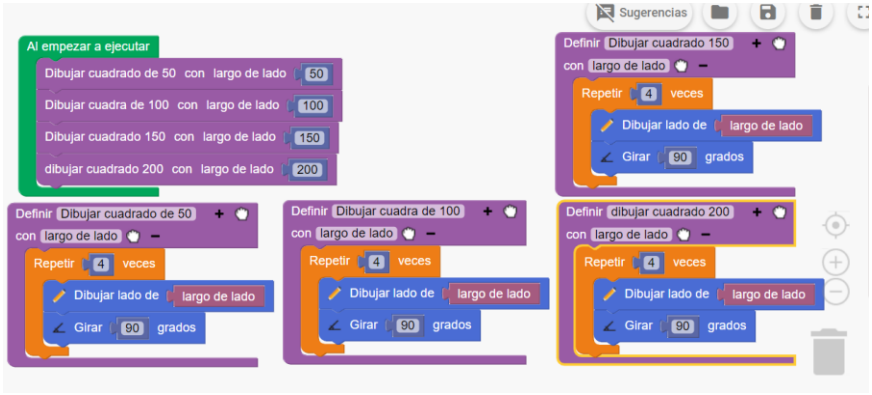

Figura 3

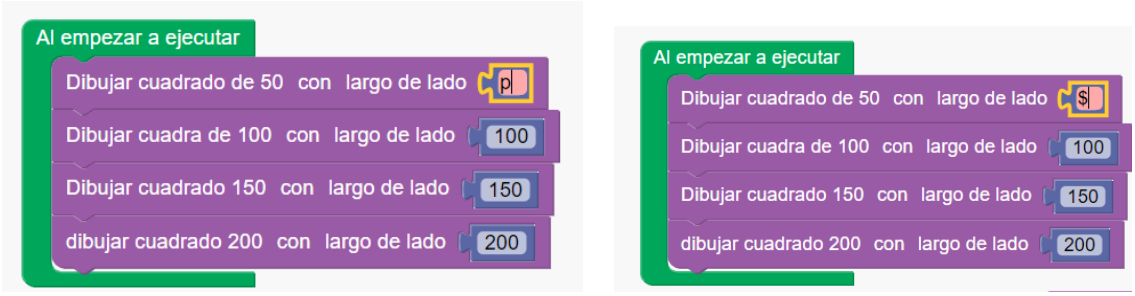

Figura 4

- **Gobstones:** No permite el ingreso de ningún dato ajeno a los parámetros definidos en los procedimientos, como se observa Figura 5.

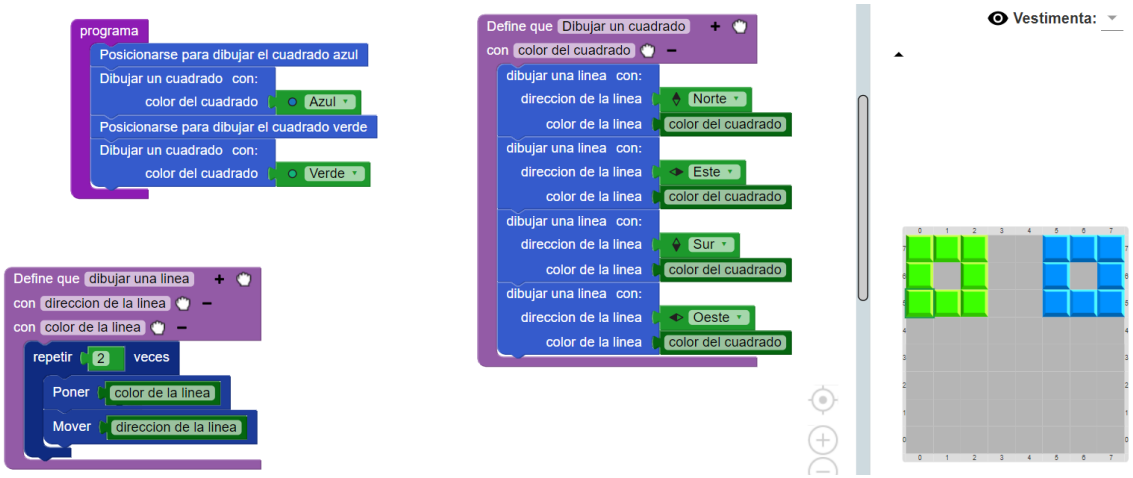

Figura 5

- **Scratch**: se comporta de igual manera que en la estructura repetitiva, como se observa en la figura 6.

Figura 6

Resultados: En la Tabla 6 se puede observar sus resultados.

Tabla 6

Acciones	PilasBloques	Scratch	Gobstones
Estructura repetitiva	1	1	1
uso de parámetros	1	1	1
Total	2	2	2

Subcaracterística: Estética de la interfaz de usuario

Métrica: Personalización de la apariencia de la interfaz del usuario
Se consideró la apreciación del evaluador, que oficia como evaluador experto.

En la Tabla 7 se puede observar sus resultados.

Tabla 7

N°	Que tan agradable es la combinación de color, texto y espacio en la interfaz de usuario	PB	Scratch	Gobstones
1	Desagradable			
2	Poco agradable			
3	Neutro			
4	Agradable		X	X
5	Muy agradable	X		

Métrica: Legibilidad adecuada
Se consideró la apreciación del evaluador, que oficia como evaluador experto.

En la Tabla 8 se puede observar sus resultados.

Tabla 8

N°	Que tan adecuada es la legibilidad en función de la combinación de color, texto y fuente en la interfaz de usuario	PB	Scratch	Gobstones
1	Desagradable			
2	Poco agradable			
3	Neutro		X	
4	Agradable	X		X
5	Muy agradable			

Subcaracterística: Accesibilidad

Métrica: Accesibilidad para usuarios con discapacidad

Para la medición de esta métrica se empleó el software NVDA (Non Visual Desktop Access) en las herramientas de programación inicial.

NVDA es un lector de pantalla gratuito y de código abierto, desarrollado por NV Access. Permite a las personas con discapacidades visuales o ceguera utilizar ordenadores de manera efectiva. Funciona mediante la lectura del texto en pantalla a través de una voz sintetizada. El usuario puede controlar lo que NVDA lee moviendo el cursor al área que contiene el texto relevante. Esto se puede hacer tanto con el ratón como con las teclas de flecha del teclado. Una de las ventajas de NVDA es su portabilidad, puede ejecutarse directamente desde una memoria USB, sin necesidad de instalación previa. Además, es multilingüe. Dispone de 11 idiomas, incluyendo el español.

Resultado: En la Tabla 9 se puede observar sus resultados.

Tabla 9

N°	Funcionalidades	PB	Scratch	Gobstones
1	Crear un programa (desafío)	1	1	0
2	Abrir un programa	1	1	0
3	Editar	1	1	0
4	Guardar predeterminado	1	1	0
5	Guardar seleccionando ubicación	0	0	0
6	Descartar/eliminar	1	0	0
7	Escribir un enunciado	1	0	0
8	Escribir un título	1	0	0
9	Ejecutar	1	1	0
10	Ver el desafío	1	1	0
11	Compartir el desafío	1	1	0
12	Descargar la actividad	1	1	0

N°	Funcionalidades	PB	Scratch	Gobstones
	Total	11	8	0

Métrica: Idiomas admitidos

Para la medición de esta métrica, se determinó que los idiomas requeridos son tres: español, inglés y portugués. Esta elección se justifica por el hecho de que Argentina es un país integrante del Mercosur, lo que hace que estos idiomas sean muy relevantes en la región.

En la Tabla 10 se puede observar sus resultados.

Tabla 10

N°	Idiomas admitidos	PB	Scratch	Gobstones
1	Español	1	1	1
2	Portugués	1	1	0
3	Ingles	1	1	0
	Total	3	3	1

Subcaracterística: Adaptabilidad

Métrica: Adaptabilidad en entorno software

Se considera el número de sistemas operativos no soportados.
En la Tabla 11 se puede observar sus resultados.

Tabla 11

N°	Sistema operativo	PB	Scratch	Gobstones
1	Windows	0	0	0
2	Linux	0	0	0
3	OS X	0	0	0
	Total	0	0	0

Anexo II: Informe de evaluación de calidad de herramientas de Programación Inicial

1. Información de Identificación

1.1. Identificación del Evaluador: Espíndola, María Cecilia

1.2. Informe de evaluación de calidad de herramientas de Programación Inicial

2. Cuerpo del informe

Se expresa en función de las etapas y procesos establecidos en la norma ISO 25040.

Etapa 1: Establecer los requisitos de la evaluación

1.1- **Establecer el propósito de la evaluación***:*

El propósito de la evaluación consiste en determinar en qué medida las herramientas software seleccionadas cumplen con las características de calidad definidas en el modelo de calidad, que se muestra en la Tabla 1.

1.2- **Obtener los requisitos de calidad del producto**:

Requiere identificar las partes interesadas en el producto. En este caso, se considera la visión de los usuarios del producto, y se seleccionan las características y subcaracterísticas del modelo de calidad propuesto, que se resume en la Tabla 1.

Tabla 1: Modelo de calidad propuesto, con sus características y subcaracterísticas

Modelo de calidad definido	
Característica	**Subcaracterística**
Adecuación funcional	Completitud funcional
Usabilidad	Reconocibilidad de la adecuación
	Capacidad de aprendizaje
	Operabilidad
	Protección contra errores del usuario
	Estética
	Accesibilidad
Portabilidad	Adaptabilidad

1.3- **Identificar las partes del producto que se deben evaluar**

Se considera la versión web de las herramientas seleccionadas:

Pilas Bloques (https://pilasbloques.program.ar/online/#/),

Scratch (https://scratch.mit.edu/projects/979618619/editor),

Gobstones (https://gobstones.github.io/gobstones-jr/).

1.4- **Definir el rigor de la evaluación**:

Considerando el propósito y el uso de los productos software, en este proceso no se consideran los factores de riesgo que establece la norma, como ser, el riesgo para la seguridad, el riesgo económico o el riesgo ambiental.

Etapa 2: Especificar la evaluación

En esta segunda etapa, se especifican los **módulos de evaluación** (métricas, herramientas y técnicas) junto con los criterios de decisión a aplicar.

2.1. Seleccionar los módulos de evaluación

Para la evaluación se utilizarán las métricas del modelo de calidad para evaluar el grado de cumplimiento en las características y subcaracterísticas, que se muestran en la Tabla 2.

Tabla 2 Métricas del Modelo de calidad generado

Característica	Subcaracterísticas	Métricas	Fórmula
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B) Donde B>0 A=número de funciones que están incorrectas o que no fueron implementadas. B=número de funciones básicas requeridas.
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B Donde B>0 A= Numero de características que posee. B= Cantidad de características definidas como referencia.

Característica	Subcaracterísticas	Métricas	Fórmula
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B Donde B>0 A= Cantidad de documentación para realizar las funcionalidades básicas. B= Cantidad de funcionalidades básicas.
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B Donde B>0 A= números funciones descripta correctamente B= Cantidad de funcionalidades básicas
	Operatividad	claridad del mensaje	X=A/B Donde B>0 A=número de mensajes implementados con explicaciones claras B= número total de mensajes implementados en las funciones básicas
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B Donde B>0 A= número de acciones y entradas de usuario que están protegidas para no causar ningún mal funcionamiento del sistema B= número de acciones y entradas de usuario que podrían protegerse para evitar que causen mal funcionamiento del sistema.
	Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	X=A/B Donde B>0 Donde A puede ser: 1- Desagradable 2- Poco agradable 3- Neutro 4- Agradable 5-Muy agradable

Característica	Subcaracterísticas	Métricas	Fórmula
		Legibilidad adecuada	X=A/B Donde B>0 Donde A puede ser: 1- Nada adecuada 2- Poco adecuada 3- Neutro 4- Adecuada 5-Muy adecuada
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B Donde B>0 A= número de funciones a las que pueden acceder personas con discapacidad B= número total de funciones básicas
		Idiomas admitidos	X = A/B Donde B>0 A= Número de idiomas requeridos que admite. B= Cantidad de idiomas necesarios para ser soportados
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X = 1-(A/B) Donde B>0 A= Numero de sistemas operativos no soportados B= número total de sistemas operativos considerados

2.2. Definir los criterios de decisión para las métricas

Se considera el nivel de importancia de las características y subcaracterísticas definidas en el modelo propuesto, así como la ponderación en el conjunto.

2.3. Definir los criterios de decisión de la evaluación

Se considera el nivel de importancia de las características y subcaracterísticas definidas en el modelo propuesto, así como la ponderación en el conjunto.

Etapa 3: Diseñar la evaluación.

Las actividades están diseñadas para que el estudiante se enfoque en el problema y no en la sintaxis, de esta manera se facilita que se apropie de los conceptos fundamentales de la programación, evitando el obstáculo frustrante de los errores de sintaxis. Además, cada

actividad prevee la utilización de las funcionalidades básicas que se establecieron en el modelo de calidad.

Actividad PilasBloques

Para el diseño de la actividad (desafío en PilasBloques), se definió el tamaño del escenario, los obstáculos, y el objetivo a alcanzar y el personaje a utilizar, como se puede observar en la Figura 1.

Nombre de la actividad: Recolectando basura.

Desafío: Ayuda a Capy y Guyrá que quieren cuidar su humedal. Para lograrlo, van a limpiar los desechos que dejaron tirados los turistas. Ayúdalos a recoger todas las latas que se encuentran tiradas.

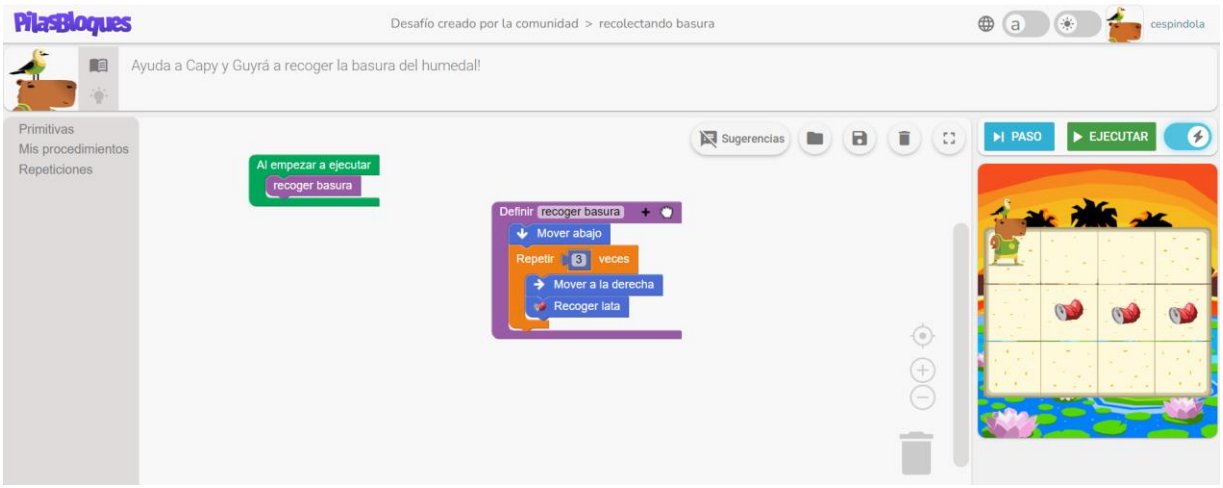

Figura 1: Entorno de Pilas Bloques
Desafío: Recolectando basura

Actividad Scratch:

Para el diseño de la actividad se eligió el fondo, los objetos y los comandos a utilizar, y los objetivos a alcanzar.

Nombre de la actividad: Instalando juegos:

Desafío: El autómata tiene que instalar un videojuego en las tres computadoras de la biblioteca. Para ello, debe encender cada computadora, ingresar la contraseña (ABC), cargar el juego y finalmente apagar la máquina, como se muestra en la Figura. 2.

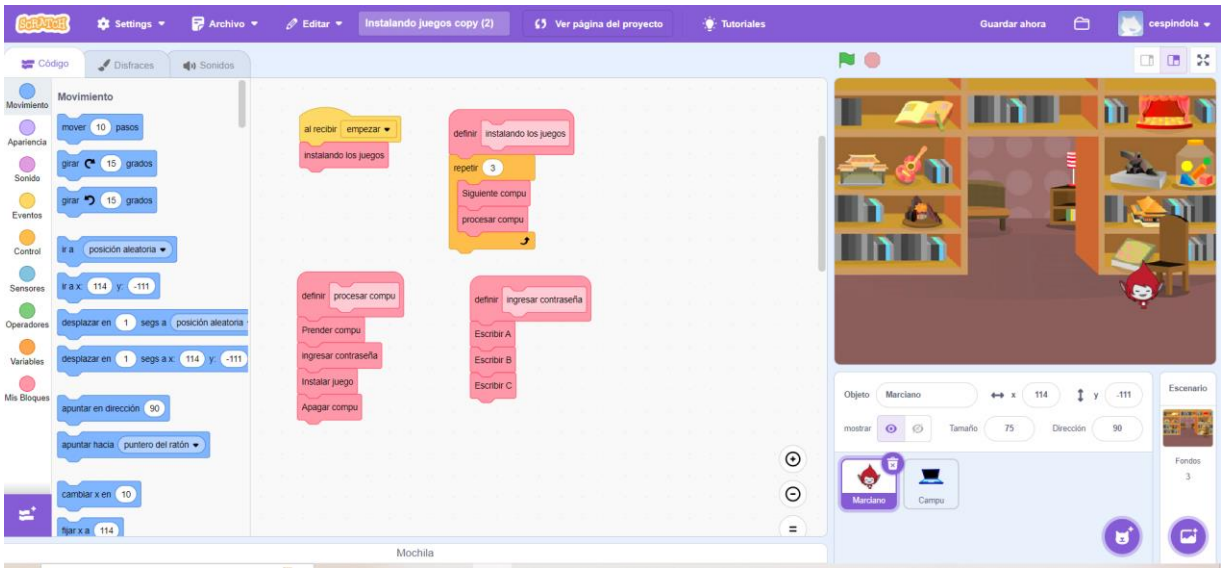

Figura 2: Entorno de Scratch
Actividad: Instalando juegos

Actividad Gobstones:

Para el diseño de la actividad se eligió el fondo, los objetos y los comandos a utilizar, y los objetivos a alcanzar.

Nombre de la actividad: Instalando bolitas.

Desafío: Esta actividad consiste en colocar en el tablero 3 bolitas rojas de manera consecutiva, para ellos debe utilizar un procedimiento **colocarbolitas**, que realice la colocación de las 3 bolitas rojas.

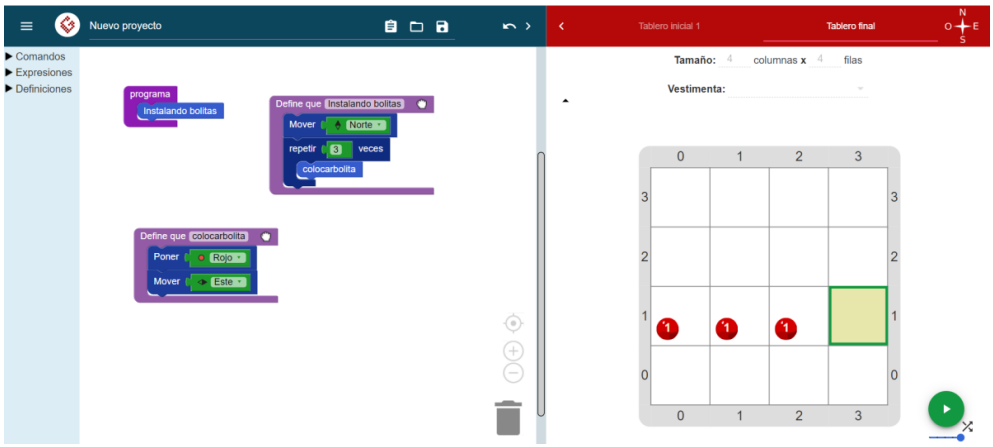

Figura 3: Entorno de Gobstones
Actividad: Instalando bolitas

Etapa 4: Ejecutar la evaluación

4.1 Realizar las mediciones

Para realizar las mediciones se desarrollaron las actividades diseñadas en cada una de las herramientas. El evaluador experto, siguiendo la matriz de calidad donde están especificadas cada una de las métricas, fue otorgando valor a cada una de ellas, en una planilla de cálculo que determina el valor de la métrica.

En la tabla 3 se muestran los valores de las métricas obtenidos para PilasBloques, en la tabla 4 los valores obtenidos para Scratch y en la tabla 5 los valores obtenidos para Gobstones.

4.1.1 Cálculo de las métricas Pilas Bloque

Se puede observar en la Tabla 3.

Tabla 3 Valor de las métricas para la herramienta Pilas Bloques

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	1	12	0,92
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B	3	7	0,43
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B	10	12	0,83
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B	10	12	0,83
	Operabilidad	Claridad del mensaje	X=A/B	9	12	0,75
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1,00
	Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	X=A/B	5	5	1,00
		Legibilidad adecuada	X=A/B	4	5	0,80
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B	12	12	1,00
		Idiomas admitidos	X=A/B	3	3	1,00
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X = 1-(A/B)	0	3	1,00

4.1.2 Cálculo de las métricas Scratch

Se puede observar en la Tabla 4.

Tabla 4 Valor de las métricas para la herramienta Scratch

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	3	12	0,75
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B	6	7	0,86
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B	6	12	0,50
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B	6	12	0,5
	Operatividad	Claridad del mensaje	X=A/B	6	12	0,5
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1
	Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	X=A/B	4	5	0,8
		Legibilidad adecuada	X=A/B	3	5	0,6
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B	8	12	0,67
		Idiomas admitidos	X = A/B	3	3	1
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X = 1-(A/B)	0	3	1

4.1.3 Cálculo de las métricas Gobstones

Se puede observar en la Tabla 5.

Tabla 5 Valor de las métricas para la herramienta Gobstones

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	5	12	0,58
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B	4	7	0,57
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B	3	12	0,25
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B	3	12	0,25
	Operatividad	Claridad del mensaje	X=A/B	2	12	0,17
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1
	Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	X=A/B	4	5	0,8
		Legibilidad adecuada	X=A/B	4	5	0,8
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B	0	12	0
		Idiomas admitidos	X = A/B	1	3	0,33

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X= 1-(A/B)	0	3	1

4.3 Aplicar los criterios de decisión para las métricas

- **Matriz de calidad de la herramienta Pilas Bloques**

  Se puede observar en la Tabla 6.

Tabla 6. Matriz de calidad Pilas Bloques

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	1	12	0,92	100	92%	20	18%	**85%**
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B	3	7	0,43	30	13%	50	6%	
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B	10	12	0,83	5	4%		2%	
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B	10	12	0,83	5	4%		2%	
	Operabilidad	claridad del mensaje	X=A/B	9	12	0,75	20	15%		8%	
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1,00	10	10%		5%	

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
	Estética	Personalización de la apariencia de la	X=A/B	5	5	1,00	10	10%		5%	
		Legibilidad adecuada	X=A/B	4	5	0,80	10	8%		4%	
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B	12	12	1,00	5	5%		3%	
		Idiomas admitidos	X = A/B	3	3	1,00	5	5%		3%	
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X = 1-(A/B)	0	3	1,00	100	100%	30	30%	

- **Matriz de calidad de la herramienta Scratch**

  Se puede observar en la Tabla 7.

Tabla 7 Matriz de calidad según ponderación de la herramienta Scratch

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	3	12	0,75	100	75%	20	15%	**82%**
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B.	6	7	0,86	30	26%	50	13%	

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B	6	12	0,50	10	5%		3%	
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B	6	12	0,5	10	5%		3%	
	Operatividad	Claridad del mensaje	X=A/B	6	12	0,5	10	5%		3%	
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1	10	10%		5%	
	Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	X=A/B	4	5	0,8	10	8%		4%	
		Legibilidad adecuada	X=A/B	3	5	0,6	10	6%		3%	
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B	8	12	0,67	5	3%		2%	
		Idiomas admitidos	X = A/B	3	3	1	5	5%		3%	
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X = 1-(A/B)	0	3	1	100	100%	30	30%	

- **Matriz de calidad de la herramienta Gobstones**

  Se puede observar en la Tabla 8.

Tabla 8. Matriz de calidad herramienta Pilas Bloques

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
Adecuación funcional	Completitud Funcional	Completitud de la implementación funcional	X=1-(A/B)	5	12	0,58	100	58%	20	12%	
Usabilidad	Reconocibilidad de la adecuación	Medida de idoneidad y reconocibilidad	X = A/B	4	7	0,57	30	17%	50	9%	
	Capacidad de aprendizaje	Integridad de la guía de usuarios	X = A/B,	3	12	0,25	10	3%		1%	
		Efectividad de la documentación del usuario o ayuda del sistema	X=A/B	3	12	0,25	10	3%		1%	**67%**
	Operatividad	Claridad del mensaje	X=A/B	2	12	0,17	10	2%		1%	
	Protección contra errores del usuario	Prevención del uso incorrecto	X=A/B	2	2	1	10	10%		5%	
	Estética de la interfaz de usuario	Personalización de la apariencia de la interfaz del usuario	X=A/B	4	5	0,8	10	8%		4%	

Característica	Subcaracterísticas	Métricas	Fórmula	A	B	Valor métrica	Pond. Subc	Resul	Pond. Carac	Calif.	Calif. final
		Legibilidad adecuada	X=A/B	4	5	0,8	10	8%		4%	
	Accesibilidad	Accesibilidad para usuarios con discapacidad	X=A/B	0	12	0	5	0%		0%	
		Idiomas admitidos	X = A/B	1	3	0,33	5	2%		1%	
Portabilidad	Adaptabilidad	Adaptabilidad en entorno software	X = 1-(A/B)	0	3	1	100	100%	30	30%	

4.4 Aplicar los criterios de decisión de la evaluación

Aplicando los criterios de evaluación a las herramientas de programación inicial se obtienen los siguientes resultados como se observa en la Tabla 9.

Tabla 9 Criterios decisión de la evaluación

Herramienta de programación	Valor de la medición	Criterio de evaluación
Pilas Bloques	**85**	**Buena calidad**
Scratch	**82**	**Buena calidad**
Gobstones	**67**	**Buena calidad**

Etapa 5: Finalizar la evaluación

4.1 Revisar los resultados de la evaluación

Después de aplicar las métricas y criterios de calidad establecidos en la Matriz de calidad a cada producto software, se analizan los resultados.

4.1.1 Resultados de la evaluación de las subcaracterísticas

Tabla 10 Resultados de la evaluación de las subcaracterísticas

Característica	Subcaracterísticas	Ponderación	PilasBloques	Scratch	Gobstones
Adecuación Funcional	Completitud Funcional	100%	92%	75%	58%
Usabilidad	Reconocibilidad de la adecuación	30%	13%	26%	17%
	Capacidad de aprendizaje	10%	4%	5%	3%
			4%	5%	3%
	Operabilidad	20%	15%	5%	2%
	Protección contra errores del usuario	10%	10%	10%	10%
	Estética de la interfaz de usuario	20%	10%	8%	8%
			8%	6%	8%
	Accesibilidad	10%	5%	3%	0%
			5%	5%	2%

Característica	Subcaracterísticas	Ponderación	PilasBloques	Scratch	Gobstones
Portabilidad	Adaptabilidad	100%	100%	100%	100%

A continuación se realiza el análisis de los resultados de las subcaracteristicas de las 3 caracteristicas seleccionadas: 1) Adecuacion Funcional, 2) Usabilidad, 3) Portabilidad

1. **Adecuacion Funcional**
   a) **Completitud Funcional**: Al representar gráficamente los valores obtenidos por las herramientas, en la figura 4 se puede observar que Pilas Bloques presenta un alto grado de cumplimiento de esta subcaracterística, lo que indica que realiza satisfactoriamente en gran medida las funcionalidades básicas evaluadas. Le sigue Scratch, con un cumplimiento un poco menor, pero superando a Gobstones, el entorno que muestra la menor completitud funcional entre las herramientas evaluadas.

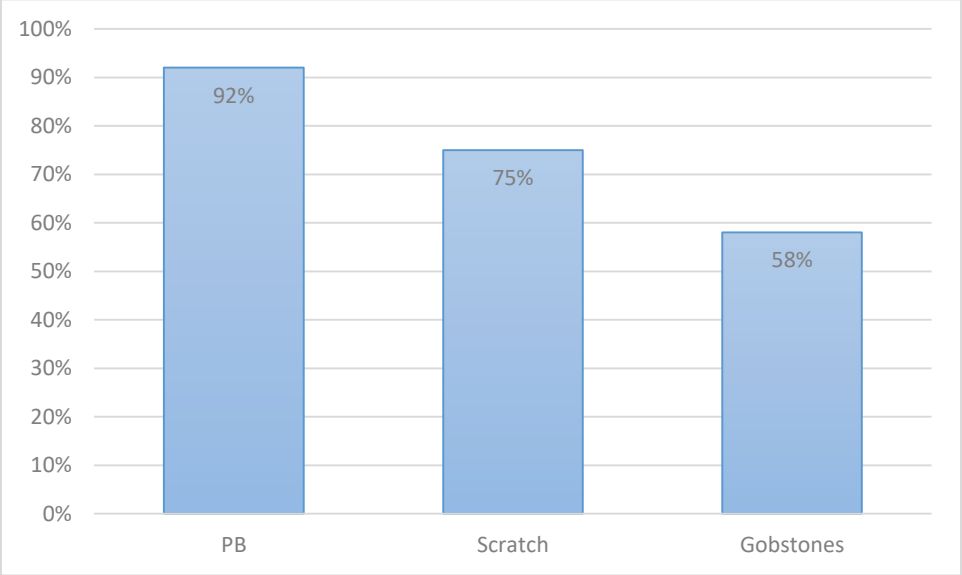

Figura 4. Completitud Funcional: grado de cumplimiento en cada herramienta

2. **Usabilidad**
   a) **Reconocibilidad de la Adecuación**: Partiendo de tabla 10 y tomando como base la ponderación asignada a esta subcaracterística en el modelo propuesto (30%), al normalizar los valores y representarlos gráficamente se puede observar en la figura 5 el grado de cumplimiento de cada herramienta de esta subcaracterística.

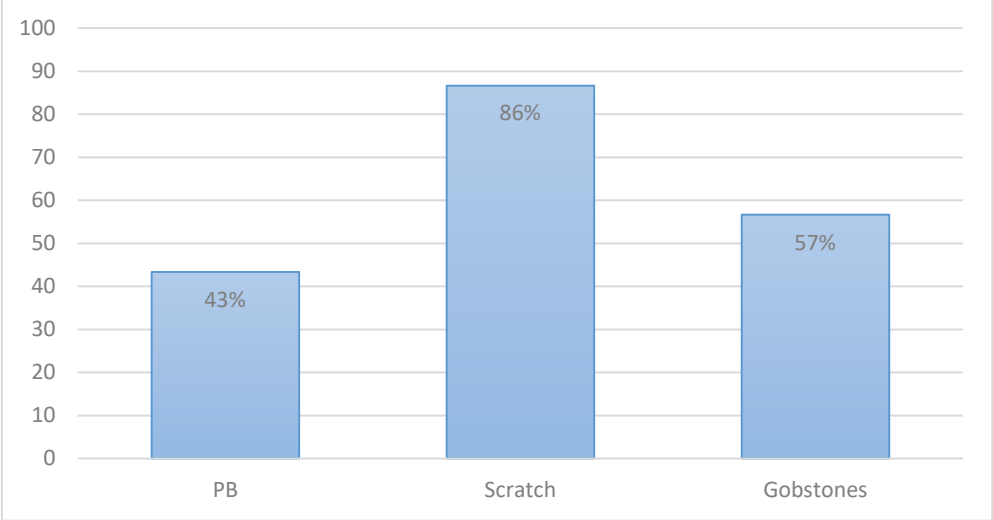

Figura 5. Reconocibilidad de la adecuación: grado de cumplimiento en cada herramienta

Los resultados reflejan que Scratch es el entorno que cumple en mayor medida la reconocibilidad de la adecuación. La alta puntuación indica que Scratch es efectivo en hacer que en su uso el entorno sea intuitivamente reconocible para los usuarios.

Gobstones presenta un cumplimiento considerablemente menor que Scratch, pero aún superior a Pilas Bloques. Con este valor, puede decirse que Gobstones proporciona propiedades que permiten a los usuarios reconocer y comprender sus características en gran medida.

Pilas Bloques presenta la puntuación más baja, lo cual significa que, aunque ofrece propiedades útiles para la programación visual, estas no son tan fácilmente reconocibles o comprensibles en comparación con Scratch y Gobstones

b) **Capacidad de Aprendizaje**: A partir de los valores de la tabla 10 y tomando como base la ponderación asignada a esta subcaracterística en el modelo propuesto (10%), al normalizar los valores y representarlos gráficamente se observa en la figura 6 el grado de cumplimiento de cada herramienta.

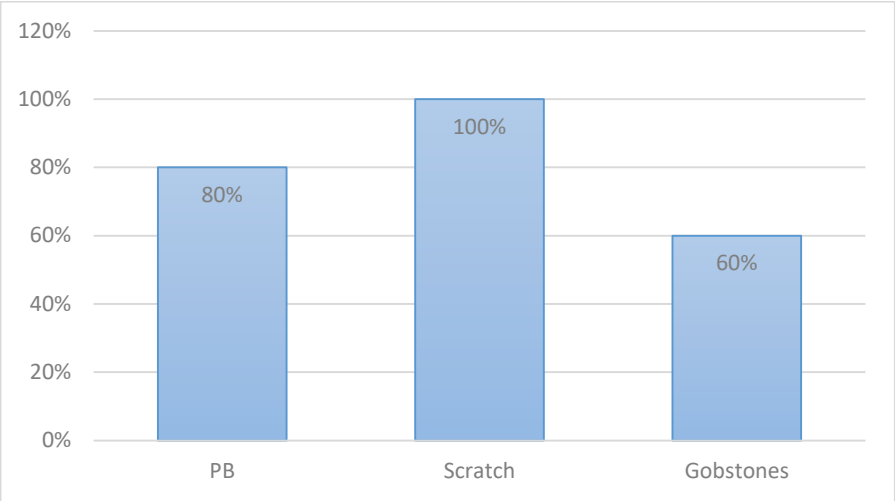

Figura 6 Capacidad de aprendizaje: grado de cumplimiento en cada herramienta

Los resultados muestran que Scratch es el entorno que mejor refleja la capacidad de aprendizaje, dado que los usuarios pueden fácilmente encontrar respuestas a sus preguntas y solucionar los problemas gracias a la documentación que brinda.

Pilas Bloques ofrece un cumplimiento menor, pero de igual manera indica una buena capacidad de aprendizaje.

Gobstones, con la puntuación más, baja sugiere que los usuarios podrían encontrar que la información proporcionada es insuficiente o menos accesible.

c) **Operabilidad:** Considerando los valores de la tabla 10 y la ponderación de esta subcaracterística en el modelo propuesto (20%), al normalizar los valores, la figura 7 ilustra el nivel de cumplimiento de cada herramienta.

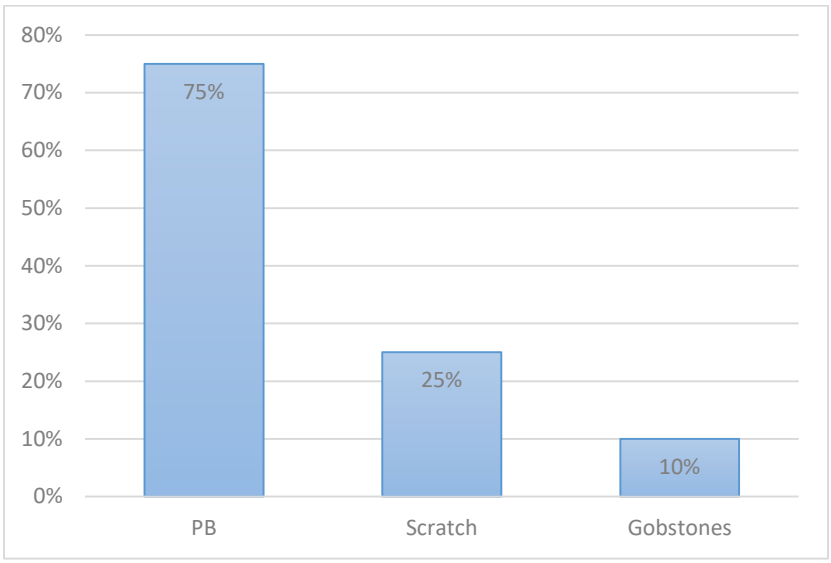

Figura 7 Operabilidad: grado de cumplimiento en cada herramienta

Pilas Bloques es el entorno que presenta la puntuación más alta en términos de operatividad, es decir, es la herramienta que brinda la mayor claridad en los mensajes.

Scratch presenta un cumplimiento mucho menor, lo que significa que su operabilidad podría generar dificultades en la interacción con la herramienta y el aprendizaje

Gobstones, con una puntuación muy baja, presenta serias deficiencias en cuanto a operatividad en comparación con Pilas Bloques y Scratch, lo que puede dificultar la experiencia del usuario y el aprendizaje.

b) **Protección Contra Errores del Usuario**: Considerando los valores de la tabla 10 y tomando como base la ponderación asignada a esta subcaracterística en el modelo propuesto (10%), al normalizar los valores y representarlos gráficamente se observa en la figura 8 el grado de cumplimiento de cada herramienta.

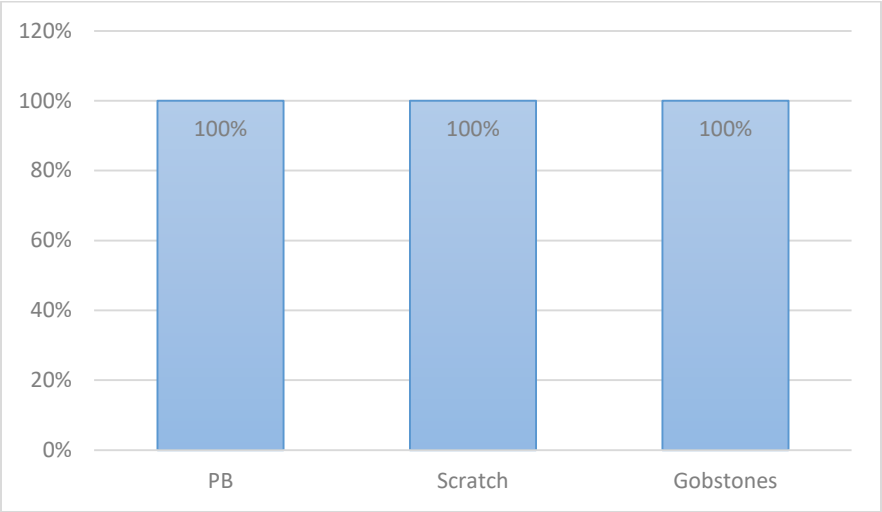

Figura 8 Protección contra errores del Usuario: grado de cumplimiento en cada herramienta

Pilas Bloques, Scratch y Gobstones cumplen con el 100% de las funcionalidades evaluadas. Esto indica que las tres herramientas presentan un nivel similar de robustez en términos de prevención y manejo de errores del usuario. Los desarrolladores de estas herramientas han implementado medidas efectivas para ayudar a los usuarios a evitar errores y utilizar las herramientas de manera eficiente, lo que contribuye a una experiencia de usuario más estable y fiable.

c) **Estética de la interfaz de usuario**: Con los valores de la tabla 10 y tomando como referencia la ponderación asignada a esta subcaracterística en el modelo propuesto (20%), al normalizar los valores y representarlos como muestra la figura 9, se observa el grado de cumplimiento de cada herramienta.

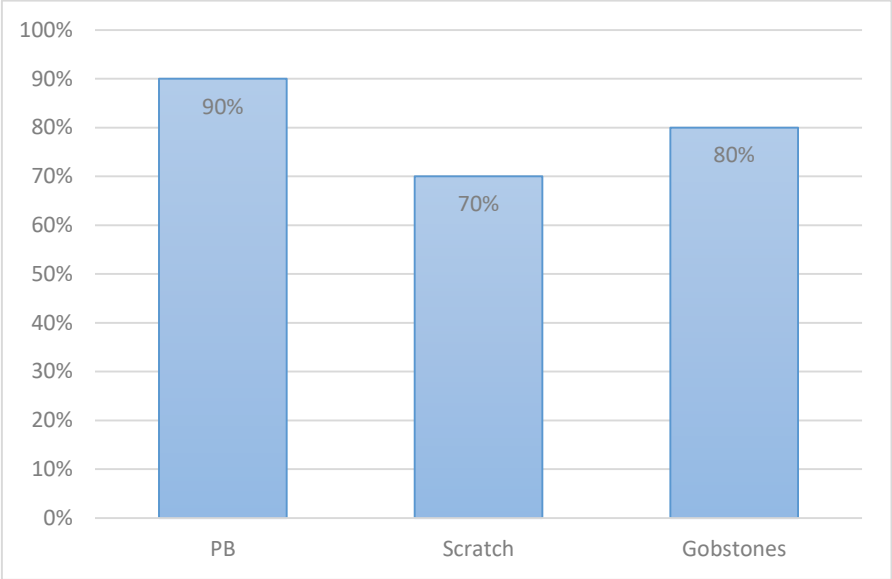

Figura 9 Estética de la interfaz de usuario: grado de cumplimiento en cada herramienta

Los resultados reflejan que Pilas Bloques es el entorno que cumple en mayor medida la estética de la interfaz de usuario, es decir, permite un alto grado de personalización en la apariencia de la interfaz y una alta legibilidad.

Gobstones cumple con esta subcaracterística, pero con algunas limitaciones en relación a Pilas Bloques. Scratch es el entorno con la puntuación más bajo en relación a las dos herramientas, esto sugiere que tanto la personalización de la apariencia como la legibilidad podrían mejorarse para ofrecer una experiencia más agradable y accesible a los usuarios.

d) **Accesibilidad**: Tomando como referencia la tabla 10 y la ponderación asignada a esta subcaracterística en el modelo propuesto (10%), al normalizar los valores y representarlos como muestra la figura 10, se observa el grado de cumplimiento de cada herramienta

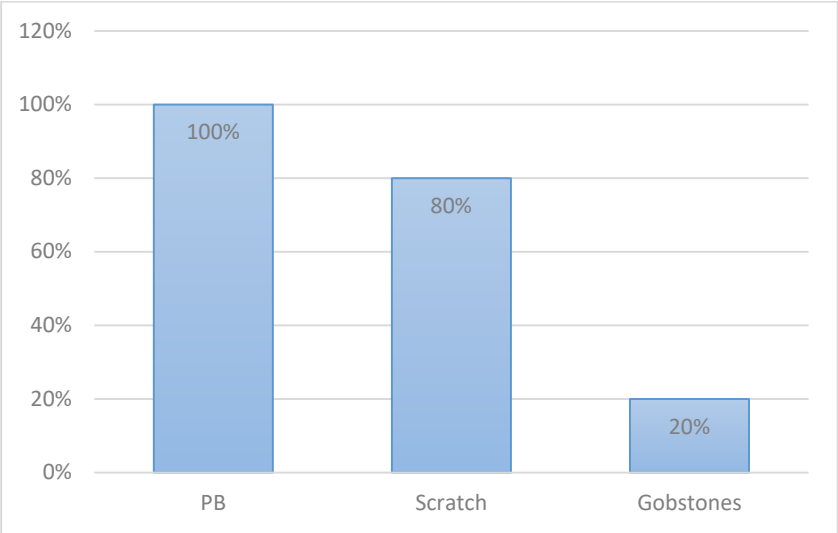

Figura 10 Accesibilidad: grado de cumplimiento en cada herramienta

Pilas Bloques es un entorno con excelente cumplimiento en accesibilidad. Esta herramienta es compatible con el software NVDA, proporciona una experiencia accesible para personas con discapacidades visuales, además, ofrece soporte para múltiples idiomas, lo que facilita el aprendizaje y el uso de la herramienta sin barreras lingüísticas, permitiendo que un mayor número de alumnos pueda beneficiarse de ella.

Scratch, con un cumplimiento menor, es compatible con el software NVDA, aunque algunos aspectos podrían no estar tan optimizados como en Pilas Bloques. También soporta una amplia gama de idiomas, facilitando su uso en diversos contextos lingüísticos.

Gobstones, con el cumplimiento más bajo, revela deficiencias en accesibilidad. Presenta limitaciones de compatibilidad con el software de asistencia como NVDA y ofrece un soporte muy limitado en cuanto a idiomas. Esto puede dificultar el uso de la herramienta.

3. **Portabilidad**

a) **Adaptabilidad:** Al representar gráficamente los valores obtenidos por las herramientas, en la figura 11 se puede observar que las 3 herramientas alcanzaron el mayor grado de cumplimiento. Esto indica que las plataformas se han desarrollado para ser accesibles en los principales sistemas operativos. En consecuencia, estas herramientas demuestran una alta portabilidad y una excelente capacidad de adaptación a diferentes entornos de software.

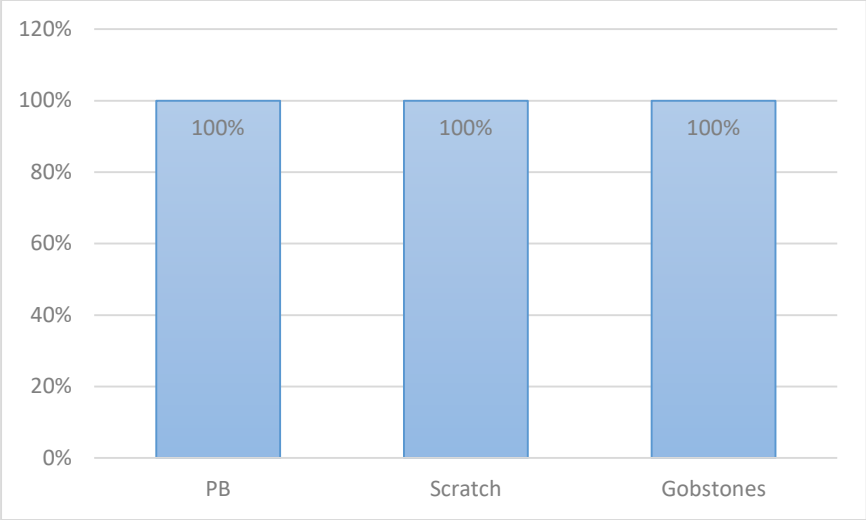

Figura 11 Adaptabilidad: grado de cumplimiento en cada herramienta

4.1.2 Resultados de la evaluación de las características

A continuación, se realiza una representación gráfica de los valores obtenidos y un análisis del grado de cumplimiento de la característica para cada herramienta.

Tabla 11 Resultados de la evaluación de las Características

Característica	Ponderación	PilasBloques	Scratch	Gobstones
Adecuación funcional	20%	18%	15%	12%
Usabilidad	50%	37%	36,50%	25%
Operabilidad	30%	30%	30%	30%
Total	100%	85%	82%	67%

Adecuación Funcional: tomando como base la ponderación (20%), se observa en la figura 12 el cumplimiento de esta característica.

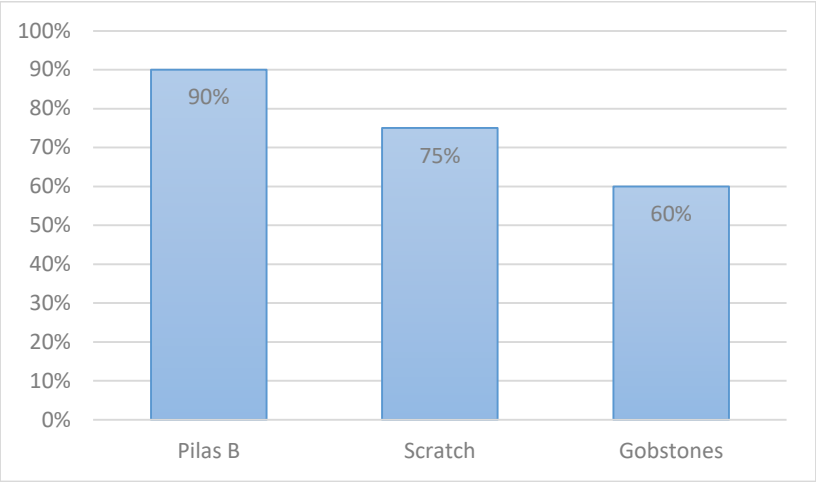

Figura12. Adecuación Funcional: grado de cumplimiento en cada herramienta

Se observa que Pilas Bloques tiene un alto grado de cumplimiento de esta característica, lo que indicando una sólida completitud funcional. Esto significa que cumple en gran medida con los requisitos necesarios para la enseñanza de la programación. proporcionando una amplia gama de funcionalidades.

Scratch con un cumplimento menor sugiere un cumplimiento con algunas limitaciones de esta característica, pero en mejor situación que Gobstones, que con un 60% presenta el menor grado de cumplimiento de la adecuación funcional.

Usabilidad: tomando como base la ponderación 50% asignada al modelo, se observan gráficamente los resultados en la figura 13.

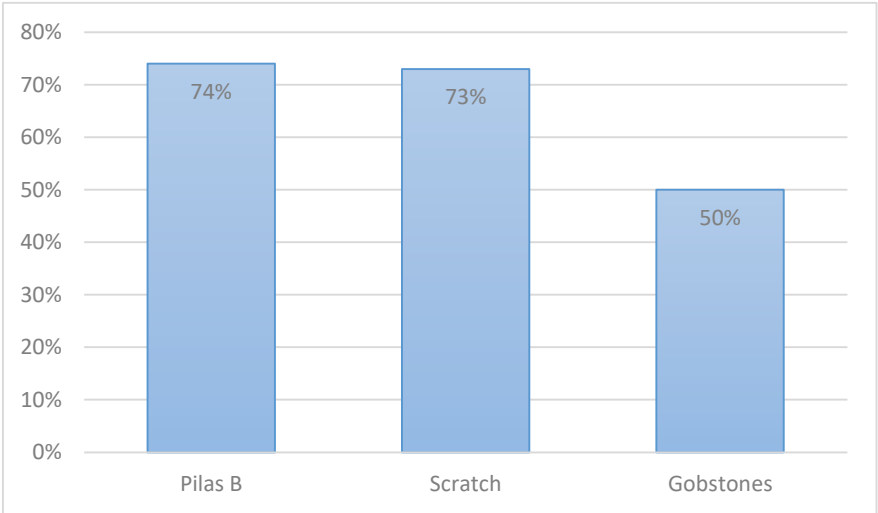

Figura13. Usabilidad: grado de cumplimiento en cada herramienta

Pilas Bloques y Scratch muestran valores similares de cumplimiento de esta característica lo que indica que ambas herramientas ofrecen una buena experiencia de usuario. Gobstones con un cumplimiento bastante menor, sugiere que enfrenta mayores desafíos en términos de facilidad de uso, lo que podría afectar la efectividad y la satisfacción del usuario.

Portabilidad: tomando como base la ponderación 30% asignada al modelo, se observan gráficamente los resultados en la figura 14.

Pilas Bloques, Scratch y Gobstones alcanza un cumplimiento del 100%. Esto indica que las herramientas evaluadas son altamente portables y accesibles en una variedad de plataformas y dispositivos. Esta alta portabilidad asegura que los usuarios puedan trabajar en sus proyectos desde diferentes entornos sin preocuparse por problemas de compatibilidad, y que el aprendizaje realizado con estas herramientas se mantenga a lo largo del tiempo.

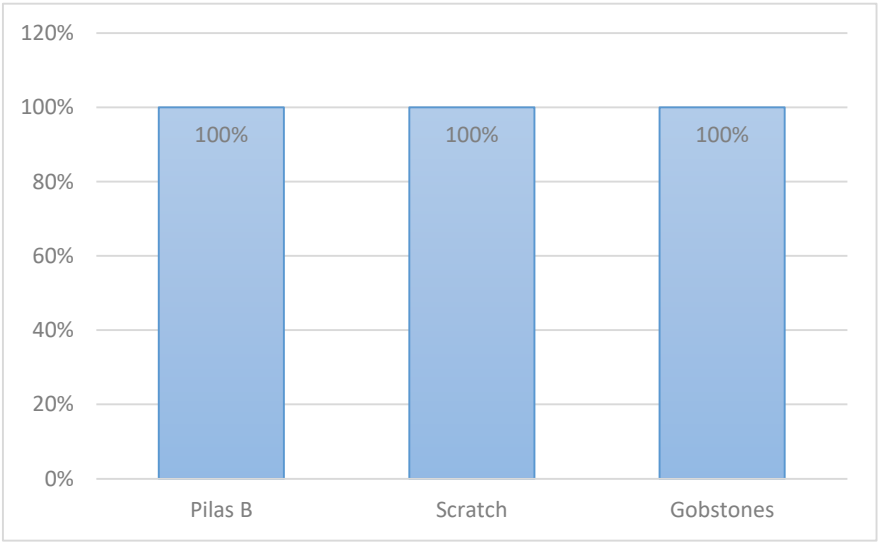

Figura 14 Portabilidad: grado de cumplimiento en cada herramienta

Conclusión de la evaluación de las características

Pilas Bloques: como se observa en figura 15 presenta un 90% de cumplimiento de la adecuación funcional**,** lo que indica una buena capacidad para satisfacer las necesidades con respecto a las funcionalidades básicas que se consideraron para la evaluación. En términos de usabilidad alcanza un 74% de cumplimiento, reflejando una buena experiencia de usuario, aunque con margen a mejoras en ciertos aspectos. Cumple en un 100% la característica de

portabilidad**,** destacándose por su capacidad para funcionar en diferentes plataformas y entornos.

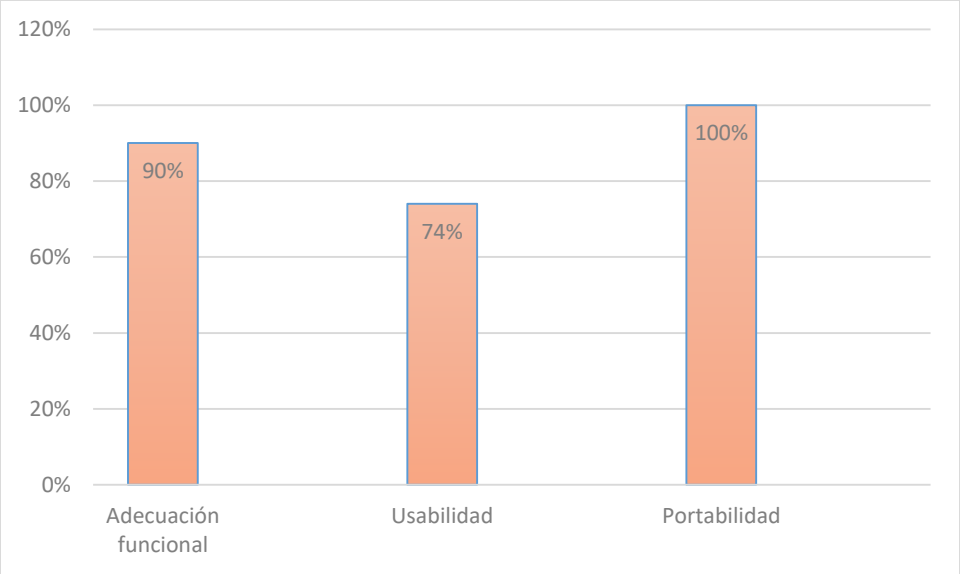

Figura 15 Resultados del modelo de Calidad para Pilas Bloques

Scratch: como se observa en la figura 16, presenta cumplimiento similar en cuanto adecuación funcional con un 75% y 73% en usabilidad, reflejando esto último una buena experiencia de usuario, aunque con margen a mejoras como Pilas Bloques. Presenta un 100% de cumplimiento en portabilidad, indicando que se adapta perfectamente a distintas plataformas.

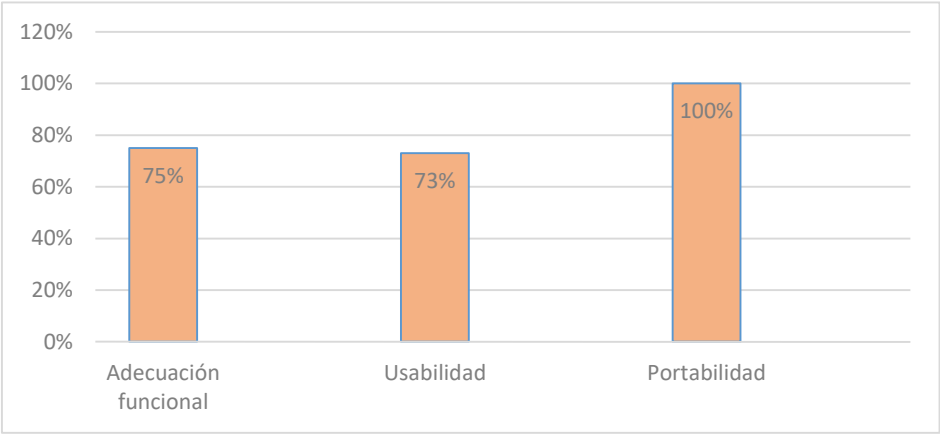

Figura 16 Resultados del modelo de Calidad para Scratch

Gobstones: como se observa en la figura 17, presenta un cumplimiento del 100% al igual que Pilas Bloques Scratch para la portabilidad.

En cuanto al cumplimiento de la adecuación funcional es más baja, que los anteriores entornos, lo que sugiere que la herramienta podría no satisfacer completamente todas las necesidades o expectativas funcionales. El cumplimiento de usabilidad es el más bajo de las 3, indicando que la herramienta podría ser más compleja de usar y aprender para los usuarios.

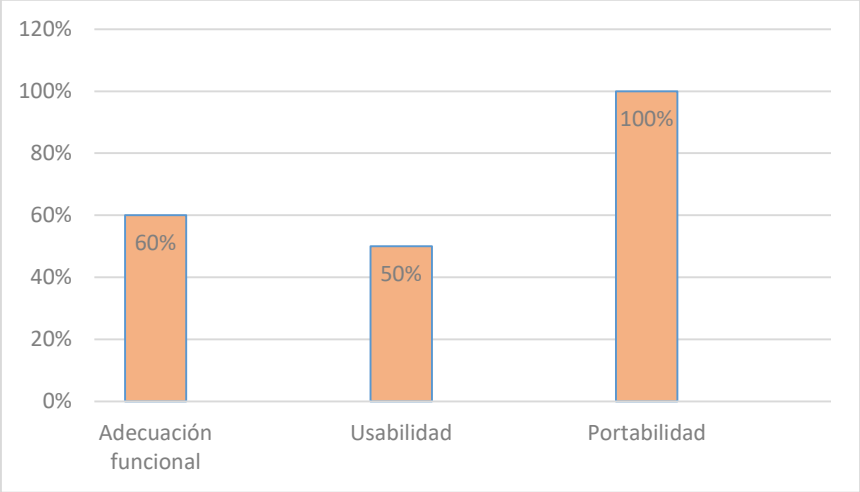

Figura17. Resultados del modelo de Calidad para Gobstones