Ensayo de investigación

# Experiencias en la aplicación de una estrategia de aprendizaje basada en un juez en línea de programación y los principios de la ciencia cognitiva

Recibido: 29-05-2020 Aceptado: 26-11-2020 (Artículo Arbitrado)


## Resumen

Un juez en línea de programación, es una plataforma que ofrece un compendio de ejercicios de programación. La herramienta realiza la evaluación automática de las propuestas de solución enviadas y presenta retroalimentación inmediata sobre el resultado, así como de las causas subyacentes, además incluye la funcionalidad para dar seguimiento al progreso de los usuarios, los cuales, típicamente son estudiantes. En este trabajo se presentan las experiencias desde la perspectiva de los estudiantes, en la aplicación de una estrategia de aprendizaje que integra el uso de un juez en línea de programación en combinación con los principios de la ciencia cognitiva, con el objetivo de establecer si la estrategia contribuye a mejorar sus habilidades prácticas de programación, así como para determinar acciones de mejora. El trabajo se realizó en el contexto de un curso de programación estructurada, ofertado en una institución de nivel superior en México.

## Abstract

An online programming judge is a platform that offers a set of programming exercises. This tool automatically evaluates the solutions submitted and provides immediate feedback on the results as well as the underlying causes, while also including the functionality to track the progress of users, which are normally students. In this work, the experiences from the students' perspectives are presented on the application of a learning strategy that integrates the use of an online programming judge in combination with the principles of cognitive science. One of the main aims was to determine whether the strategy contributed to improving practical programming skills, as well as identifying improvement actions. The work was carried out in the context of a structured programming course, offered at a university in Mexico.

## Résumé

Un juge de programmation en ligne est une plateforme qui propose un recueil d'exercices de programmation. L'outil effectue l'évaluation automatique des propositions de solutions soumises et présente un retour immédiat sur le résultat, ainsi que les causes sous-jacentes, il comprend également la fonctionnalité de suivi de la progression des utilisateurs, qui sont généralement des étudiants. Dans ce travail, les expériences sont présentées du point de vue des étudiants, dans l'application d'une stratégie d'apprentissage qui intègre l'utilisation d'un juge de programmation en ligne en combinaison avec les principes des sciences cognitives, dans le but d'établir si le La stratégie aide à améliorer vos compétences pratiques en programmation, ainsi qu'à déterminer les actions d'amélioration. Le travail a été réalisé dans le cadre d'un cours de programmation structuré, offert dans un établissement de niveau supérieur au Mexique.



Hugo Enrique Martínez Cortés[1*]
Myriam Karenina Reyes Sánchez[2]



**Palabras clave:** Programación, estrategia de aprendizaje, juez en línea, ciencia cognitiva.
**Keywords:** Programming, learning strategy, online judge, cognitive science.
**Mots-clés:** Programmation, stratégie d'apprentissage, juge en ligne, sciences cognitives.


## Introducción

Actualmente la economía global se sustenta primordialmente en el manejo eficiente de la información y el conocimiento, esto deriva en un incremento en el uso y desarrollo de las Tecnologías de Información y Comunicación (TIC) en el sector empresarial, con el objetivo de mejorar la competitividad y productividad, así como lograr una reducción de costos. Como parte importante de las TIC, la industria de desarrollo de software a nivel mundial también experimenta este mismo fenómeno de crecimiento. Guel y Araiza (2015) afirman que "*el desarrollo de software constituye un sector económico importante a nivel mundial y se encuentra en el centro de todas las grandes transformaciones; sobre todo si se*


[1]Instituto de Computación
[2]Laboratorios de Posgrado
Universidad Tecnológica de la Mixteca

Correspondencia:
[*]hugoe@mixteco.utm.mx

*considera que los grandes problemas del momento son la economía digital, la evolución de las empresas y la administración del conocimiento, entre otras*". De acuerdo al U.S. Department of Labor, Bureau of Labor Statistics (2020), se proyecta un crecimiento del 21% en la tasa de empleo para desarrolladores de software, en el periodo de 2018 a 2028, una tasa mayor al promedio de crecimiento calculado para todas las ocupaciones incluidas en el reporte. En el caso de México, de acuerdo a AMITI (2013), existe un rezago en el aprovechamiento de las TIC, las mayores debilidades del país son la infraestructura, la banda ancha móvil y el costo de la telefonía celular. Sin embargo, en el mismo informe se describe que el subsector de desarrollo de software, experimentó uno de los mayores crecimientos en los últimos años, y se espera que este subsector tendrá una mayor importancia en la industria TIC en el periodo 2011-2025. A pesar de este escenario favorable para la industria del desarrollo de software en México, de acuerdo a IMCO (2014) el factor de capital humano se encuentra en una situación precaria, esto se debe a una baja calidad de la educación y de la capacitación en el país. En este sentido, se expresa que los ingenieros en sistemas poseen habilidades técnicas, pero carecen de una adecuada formación en negocios, ante lo cual, recomienda alinear los programas educativos con las necesidades de los emprendedores, así como flexibilizar carreras de ingeniería para promover más habilidades de negocios. Al respecto, Mejía (2017) expresa que, debido a la falta de competencias de los egresados, solo un pequeño porcentaje logra integrarse con éxito en la industria, por lo tanto, resulta imprescindible incrementar el acervo del capital humano en ciencia y tecnología, tanto en número como en la calidad de la formación.

## El problema de la educación

Típicamente, la industria de desarrollo de software recluta profesionistas egresados de programas educativos a nivel Licenciatura, en las áreas de Ciencias en Computación e Ingeniería de Software. En este contexto, Watson y Li (2014), establecen que estas carreras enfrentan problemas importantes, como la dificultad para incorporar nuevos alumnos, así como una alta tasa de deserción o abandono de los estudiantes activos. Además, expresan que una de las causas principales de esta situación, es la dificultad que se atribuye al curso de programación básica, que se incluye normalmente en los primeros semestres de los planes de estudio. El temario para este curso, incluye el concepto de algoritmo y los conocimientos para construir o codificar un programa de cómputo en un lenguaje de programación. En el trabajo de Watson y Li (2014), se realizó una revisión sistemática de la literatura sobre el tópico de cursos de programación básica, y calcularon una tasa de aprobación del 67.7% para este curso a nivel mundial, lo que genera una tasa de reprobación del 32.3%. La tasa de aprobación se define como el porcentaje de alumnos que cumplen los criterios mínimos para acreditar el curso, con relación al total de alumnos que finalizan el curso. En un reciente estudio, Bennedsen y Caspersen (2019), determinaron una tasa de reprobación similar, del 28% para el curso de programación básica, a nivel mundial. En el caso de México, a pesar de que no existe a la fecha un estudio a nivel nacional, algunas universidades del país reportan que la tasa de reprobación del curso de programación básica, es de las más altas, o incluso la más alta, de sus programas educativos (Amado Moreno et al., 2014; Juárez, López y Villareal, 2016; Rangel, García y Habib, 2012; Rodríguez Pérez, 2017).

Se han realizado trabajos de investigación con el propósito de explicar las causas que originan estos altos índices de reprobación y abandono en los cursos de programación. Gomes, Brito y Henriques (2016) argumentan que esta situación se debe al nivel de conocimientos previo que poseen los estudiantes, la propia naturaleza compleja que tiene la programación, los métodos de estudio y aprendizaje, así como las estrategias pedagógicas que se usan típicamente en dichos cursos. Bodganovich y Treskat (2016) se identifican las principales causas para estos problemas, como son: prácticas de enseñanza tradicionales que no son efectivas con los estudiantes "digitales" (los cuales presentan deficiencia de atención, falta de concentración y poca motivación para aprender a programar), el perfil de ingreso solicitado a los estudiantes en su incorporación al programa educativo y finalmente el poco tiempo disponible que tienen los estudiantes para realizar las asignaciones del curso.

De acuerdo a Paiva, Leal y Queirós (2016) los educadores tienen el desafío de conciliar múltiples elementos educativos con el propósito de compro-

meter y motivar a los estudiantes en el aprendizaje de la programación. En primer término, los profesores deben crear contenidos didácticos atractivos para explicar los conceptos de programación, así como adaptar este contenido a las necesidades y preferencias de cada estudiante. Seleccionar ejercicios que cubran todo el programa de estudios y asegurarse de que puedan ser calificados objetivamente, con la cantidad correcta de retroalimentación, y en un periodo que permita a los estudiantes mantenerse enfocados. Se debe también motivar el trabajar en equipo, para fomentar la interacción y el aprendizaje con otros estudiantes en ambientes de competencia o colaboración. Además, es necesario elegir las herramientas correctas para la enseñanza.

Los métodos tradicionales de enseñanza, caracterizados por la transmisión de conocimientos desde los educadores hacia estudiantes pasivos, ya no son suficientes para enfrentar el gran desafío que implica la educación de la programación en cursos de nivel superior. Por esta razón, y como una alternativa a este enfoque, han surgido propuestas dirigidas a la aplicación de métodos de aprendizaje activo, en general estos métodos requieren que los estudiantes tomen el control de su aprendizaje y de esta manera se involucren activamente en el proceso educativo (Bonwell & Eison 1991). En el enfoque de aprendizaje activo los estudiantes deben construir sus propios conocimientos y habilidades, el énfasis del aprendizaje es constructivo, acumulativo, orientado a objetivos, auto-regulado y diferente para cada individuo (Boekaerts y Cascallar, 2006; Locke y Latham, 1990). Algunos métodos notables de aprendizaje activo son: el aprendizaje basado en problemas (Lykke et al., 2014), el aprendizaje basado en proyectos (Jazayeri, 2015), el aprendizaje basado en equipos, el aprendizaje colaborativo (Silva y Madeira, 2010) y el aprendizaje cooperativo (Ismail, Ngah y Umar, 2010).

Existe otro enfoque pedagógico fundamentado en los principios de la ciencia cognitiva, la cual, estudia los complejos mecanismos de la mente humana. Bajo este enfoque didáctico, se establece que la formación basada en el constructivismo, en donde existe una mínima orientación en el proceso educativo por parte de los profesores, es significativamente menos efectiva y eficiente, comparada con la formación en donde el profesor brinda orientación diseñada específicamente para dar soporte al proceso cognitivo necesario para el aprendizaje (Kirschner, Sweller y Clark, 2006). Con el objetivo de lograr un aprendizaje efectivo, se argumenta que el procedimiento educativo se debe apoyar en la constitución de la arquitectura cognitiva humana, como son las características de la memoria de trabajo, la memoria de largo plazo, así como las complejas relaciones que se presentan entre estas. Adicionalmente, Butler Marsh, Slavinsky y Baraniuk (2014) expresan que, en ciertos casos, los métodos de aprendizaje activo, presentan diversas barreras para su implantación en un entorno educativo, debido a que, pueden requerir una revisión completa del plan educativo y pedagogía; implicar tiempo, dinero y experiencia; así como la aplicación de un esfuerzo considerable por parte de los profesores.

Adicionalmente a las estrategias pedagógicas, se han propuesto diversas herramientas y ambientes de aprendizaje para apoyar a los profesores y estudiantes en cursos de programación. Rongas, Kaarna, y Kalviainen (2004) establecieron una clasificación de estas herramientas, en las siguientes cuatro categorías:

1. Interfaces integradas de desarrollo.
2. Herramientas de visualización.
3. Ambientes de aprendizaje virtual.
4. Sistemas para el envío, administración y evaluación de ejercicios de programación, también conocidos como jueces en línea de programación (POJs por sus siglas en inglés).

Actualmente las herramientas POJ han adquirido mayor importancia a nivel mundial, y se ha popularizado su uso en el proceso enseñanza/aprendizaje de los cursos de programación debido al apoyo proporcionado tanto a los docentes como estudiantes. En el presente trabajo se presentan las experiencias de los estudiantes, en la aplicación de un juez en línea para la resolución de ejercicios de programación, bajo una estrategia de aprendizaje que incluye una intervención fundamentada en principios de la ciencia cognitiva, con el objetivo de establecer acciones de mejora y confirmar si la estrategia contribuye a consolidar las habilidades prácticas de programación en los estudiantes.

## Trabajos relacionados

En el trabajo de Luca Bez, E. Ferreira y A. Tonin (2013), se presenta un POJ denominado URI Online *Judge*

*Academic*, como una plataforma para el aprendizaje en los tópicos de algoritmos y programación. La herramienta permite a un profesor asignar un conjunto de ejercicios a los estudiantes, evaluar automáticamente sus soluciones, así como llevar un seguimiento de su avance, en un ambiente en línea, organizado e intuitivo. Por su parte Wang et al., (2015), presentan una herramienta denominada OJPOT, que combina un juez en línea, con la enseñanza orientada por práctica. Reportan que la incorporación de OJPOT en un curso de programación básica, permite que los estudiantes mejoren sus habilidades prácticas, en comparación con el enfoque de la enseñanza tradicional. En el trabajo de Restrepo-Calle et al., (2018) se propone UNCode, que constituye un ambiente educativo basado en web, para el aprendizaje de las habilidades de programación y la evaluación automática de ejercicios en un contexto académico. Esta herramienta integra un POJ, en combinación con elementos de ambientes de aprendizaje, para presentar retroalimentación formativa a los estudiantes. Afirman que UNCode contribuye a enfrentar las dificultades asociadas a la enseñanza de la programación y a la evaluación manual de los ejercicios. Combéfis y de Moffarts (2019) proponen la plataforma Pythia, para dar soporte al proceso de enseñanza/aprendizaje de la programación. Esta herramienta evalúa automáticamente diferentes tipos de ejercicios de programación, utilizando diversos criterios (funcionalidad, calidad de código, desempeño de ejecución, consumo de memoria, etc.). Aunado a la anterior, la plataforma puede generar retroalimentación "inteligente" para apoyar al aprendizaje, ya que presenta al estudiante consejos e información de sus fallas.

## Juez en línea de programación

Los jueces en línea de programación, son herramientas de aprendizaje electrónico que ofrecen un conjunto de problemas o ejercicios de programación en línea, con el propósito de que puedan ser resueltos por los usuarios, los cuales, típicamente son estudiantes universitarios que cursan programas relacionados con la disciplina de Ciencias en Computación. Estas herramientas realizan un proceso automático de compilación y evaluación de las soluciones enviadas por los estudiantes (Yera et al., 2017). En el presente trabajo se hace uso de OmegaUp, que ha sido desde 2011, la plataforma oficial de la Olimpiada Mexicana de Informática (OMI), en dicho evento se celebra un concurso de programación con participantes de instituciones educativas de nivel medio superior de todo el país. La plataforma permite a los estudiantes entrenar sus habilidades, por medio de la resolución de ejercicios de programación; además ofrece la funcionalidad para crear y calendarizar concursos de programación para los propios estudiantes; en la misma herramienta, los profesores tienen la posibilidad de crear nuevos ejercicios de programación y finalmente, ofrece la opción de dar seguimiento al progreso de los estudiantes a través de tareas y exámenes, que se evalúan automáticamente. Es importante destacar, que la integración de esta plataforma en el curso, se realiza bajo el soporte de una estrategia de aprendizaje, que incluye una intervención basada en principios de la ciencia cognitiva. En general, la interacción del estudiante con el POJ se realiza de la siguiente manera:

1. Selección del ejercicio. El alumno selecciona un problema o ejercicio, del conjunto disponible. A partir de este momento, el alumno inicia el proceso para darle solución.
2. Envío de una solución. Una vez que el alumno ha construido un programa en lenguaje de programación específico, como propuesta de solución al ejercicio seleccionado, envía el código fuente al POJ (ver la figura 1).
3. Evaluación automática. El POJ evalúa la solución enviada por el alumno, usando casos de prueba (entradas/salidas) predefinidos. La herramienta verifica si la salida del envío, coincide con la salida esperada o correcta. Si este es el caso, la solución se considera correcta y el ejercicio ha sido resuelto (ver la figura 2). En otro caso, el juez envía retroalimentación al alumno de forma inmediata, sobre la causa de la respuesta incorrecta. En seguida, el alumno realiza las modificaciones necesarias a su código y posteriormente realiza un nuevo envío, o tentativamente, puede decidir resolver un ejercicio diferente.

## Principios de la ciencia cognitiva

Se ha demostrado que la aplicación de los principios de la ciencia cognitiva como estrategia pedagógica, puede contribuir de manera significativa al mecanismo de aprendizaje de largo plazo, así como al grado de comprensión de la información; de esta manera, los estudiantes pueden lograr un aprendizaje eficaz

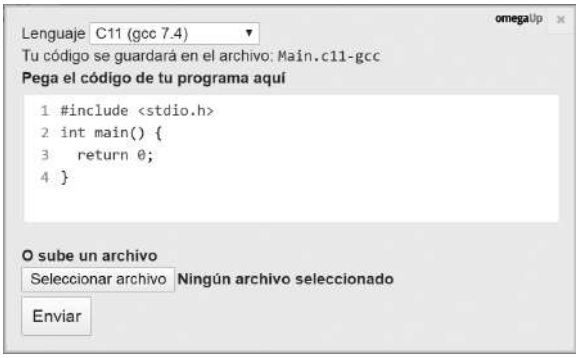


**Figura 1**. Pantalla de OmegaUp para el envío del código con la propuesta de solución para un ejercicio.

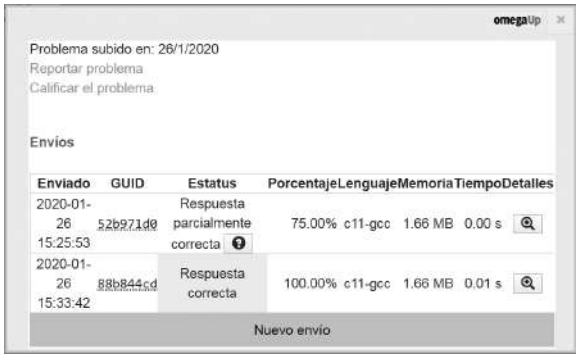


**Figura 2**. Pantalla de OmegaUp que presenta la retroalimentación de la evaluación para un código enviado.

y eficiente, al aplicar métodos que potencian y favorecen los mecanismos cognitivos necesarios para el aprendizaje de alto orden (Bjork y Yan, 2014). Al respecto, Butler et al. (2014), argumentan que al realizar una intervención en el proceso de aprendizaje de estudiantes de nivel universitario, aplicando de manera combinada los principios cognitivos de: práctica de recuperación repetida, espacio y retroalimentación; se logra una aportación significativa en el aprendizaje de material complejo de ingeniería. A continuación, se describen estos principios:

1. Práctica de recuperación repetida. La recuperación repetida de la información desde la memoria, puede mejorar el grado de entendimiento de dicha información, así como también, fortalece su consolidación en la propia memoria de largo plazo. La práctica de la recuperación promueve la adquisición de conocimiento y su aplicación en diferentes contextos (Roediger y Butler, 2011).

2. Espacio. Al espaciar o distribuir las actividades de estudio de la información en el tiempo, se produce una mayor retención de largo plazo, en comparación con la práctica de estudiar de forma repetida o consecutiva durante un solo periodo (Cepeda et al., 2006).

3. Retroalimentación. Se conceptualiza como la información que proporciona un agente (p. ej., el profesor), relacionado con el desempeño o entendimiento de una persona. La retroalimentación proporciona a los aprendices, la información que les permite corregir errores y mejorar su entendimiento (Hattie y Timperley, 2007).

# Desarrollo

## Planeación de la estrategia de enseñanza

La estrategia de enseñanza de este trabajo se planea en el contexto de un curso de programación estructurada. La asignatura de programación se imparte en el primer semestre y tiene una duración de 17 semanas, dividas en cuatro periodos, los tres primeros, con una duración de cinco semanas, y el cuarto, con una duración de dos semanas. La asignatura presenta un programa de estudios que incluye las unidades temáticas que se muestran en la tabla 1, desglosados de acuerdo al periodo en el que se imparten.

El curso se dicta de manera tradicional, es decir, los estudiantes toman una clase diaria en una sala de cómputo, bajo la tutela de un profesor especialista en el área, que presenta los temas del curso con el apoyo de una computadora y un proyector, de esta manera guía a los estudiantes en la elaboración de prácticas de programación durante la clase. Adicionalmente, el profesor otorga documentación en línea del curso a los estudiantes y solicita ejercicios de programación como tarea durante cada periodo. Al finalizar el curso, solicita un proyecto final que cubre todos los temas del programa de estudios. La evaluación del curso se lleva a cabo por medio de tres exámenes parciales y un examen final. Los tres primeros, se aplican al finalizar cada uno de los periodos iniciales y el examen final se aplica al concluir el último periodo.

La estrategia de enseñanza que se propone en este trabajo, tiene como propósito afectar en un nivel mínimo las actividades de enseñanza/aprendizaje

**Tabla 1**. Unidades temáticas del programa de estudios para el curso de programación

| Unidad temática | Periodo en el que se imparte |
|---|---|
| Introducción al lenguaje de programación C | Primero |
| Estructuras de control (secuencial, selectivas, iterativas) | Primero |
| Programación modular | Segundo |
| Arreglos | Segundo |
| Apuntadores | Tercero |
| Tipos de datos definidos por el programador | Tercero |
| Archivos | Cuarto |

de este curso tradicional y en consecuencia, reducir el costo de la implementación necesaria para la intervención. Por esta razón, la intervención se enfoca de manera directa en los ejercicios de programación, que constituyen las asignaciones principales que realiza un profesor a sus estudiantes, en el curso de programación. En dichas asignaciones, típicamente se solicita al estudiante, codificar un programa de cómputo en un lenguaje de programación (lenguaje C), para dar solución a los requerimientos planteados por cada ejercicio, bajo restricciones y condiciones bien establecidas. En un curso con un enfoque de enseñanza tradicional, el profesor invierte un esfuerzo considerable en la revisión de las soluciones propuestas por los alumnos, este trabajo aumenta en proporción directa con la dificultad y cantidad de los ejercicios, por lo que el conjunto de asignaciones para la práctica de los estudiantes en un curso clásico, puede presentar limitantes en términos de cantidad y complejidad. Para hacer frente a esta situación, el uso de la plataforma OmegaUp, ofrece la posibilidad de realizar una evaluación automática de los ejercicios, e incluye un conjunto numeroso de ejercicios prediseñados, de diversa complejidad, que pueden ser utilizados durante la ejecución del curso. En la tabla 2, se hace una descripción de la implementación de los principios cognitivos, para la estrategia de intervención propuesta en el presente trabajo, enfocada en los ejercicios de programación y utilizando la plataforma OmegaUp.

## Participantes

La estrategia de intervención se realizó con 71 estudiantes de un curso de programación estructurada en lenguaje C, impartido en el primer semestre de un programa educativo en Ingeniería en Computación. Los estudiantes se organizan en dos grupos, el primero integrado por 38 de ellos, y el segundo por 33. Es importante recalcar, que el 100% de los estudiantes tomaron el curso por primera vez.

## Materiales

Con la finalidad de disponer de un catálogo de ejercicios para cubrir los siete tópicos del programa de estudios (ver la tabla 3), se elaboraron 123 ejercicios de programación, 60 de complejidad baja, enfocados al repaso del tema de la clase anterior, 62 de complejidad media, para las tareas semanales y 1 ejercicio de complejidad alta para la asignación final del curso. Es importante mencionar que la última semana de cada periodo del curso, corresponde al espacio en el cual, no se resuelven ejercicios, ni tampoco se presentan nuevos temas.

La creación de los ejercicios se realizó de acuerdo a las especificaciones de la plataforma OmegaUp. Esta herramienta solicita que la redacción del problema se escriba en formato markdown y ofrece la opción de aplicar los elementos estructurales del lenguaje LaTeX, que se encuentra soportado completamente en la plataforma.

Aunado a la redacción del ejercicio, se requiere la elaboración de casos de prueba, que constituyen archivos de texto planos, en donde se registran las entradas y salidas esperadas para una solución correcta del ejercicio. Cada caso de prueba, permite a OmegaUp evaluar la solución propuesta bajo las condiciones específicas, definidas en el caso. El procedimiento general es el siguiente: OmegaUp compila y ejecuta el programa enviado, a continuación, alimenta la entrada del caso de prueba al programa y finalmente, confirma si la salida producida por éste, es igual a la salida correcta definida en el caso. De

**Tabla 2**. Implementación de los principios cognitivos en la estrategia de intervención

| Principio cognitivo | Descripción |
|---|---|
| Práctica de recuperación repetida | Al iniciar la clase se resuelve un ejercicio de programación de complejidad baja, sobre la temática de la clase anterior. El problema se resuelve utilizando la plataforma OmegaUp.<br>Al final de la semana, en OmegaUp se asigna a los alumnos un conjunto de problemas como tarea (mayor al tamaño del conjunto para un curso tradicional), de complejidad media, enfocados fundamentalmente en los temas vistos en esa semana, aunque, los ejercicios frecuentemente acumulan todos los temas vistos hasta ese momento.<br>Al final del curso, en OmegaUp se asigna a los alumnos un problema de complejidad alta, que cubre los temas principales del temario. |
| Espacio | En la última semana de cada periodo del curso, no se realizan asignaciones de ejercicios a los alumnos. Es decir, se deja un espacio de una semana sin prácticas de programación.<br>Durante todo el curso, al finalizar cada semana de clase, los estudiantes practican con los ejercicios asignados de manera semanal. El periodo para la entrega se especifica y valida en la plataforma OmegaUp. |
| Retroalimentación | La herramienta OmegaUp ofrece retroalimentación inmediata a las propuestas de solución de los ejercicios enviadas por los alumnos, e informa si la propuesta es correcta o no, en cuyo caso indica las causas del error de manera simplificada. La plataforma también indica el porcentaje de casos de prueba resueltos de manera correcta para cada ejercicio.<br>Durante cada clase, cuando los estudiantes realizan la práctica para resolver un ejercicio de programación sobre el tema del día anterior, el profesor otorga retroalimentación al grupo sobre la solución correcta del ejercicio en cuestión. |

esta manera, con el objetivo de verificar de manera exhaustiva si la solución propuesta es correcta bajo diversas condiciones, es necesario elaborar múltiples casos para cada ejercicio. El paquete con la descripción y los casos de prueba para cada ejercicio recién elaborado, se carga en la plataforma, y a partir de este momento forma parte del catálogo disponible en OmegaUp.

## Procedimiento

El curso se llevó a cabo en el año 2019 y para su realización, únicamente se necesitó instruir a los estudiantes en la primera clase sobre el funcionamiento de OmegaUp en los siguientes tópicos: procedimiento de registro, envío y evaluación de sus propuestas de solución, los mecanismos para dar seguimiento al curso, así como, la revisión de sus estadísticas. Dentro de la plataforma se creó un curso, módulo que permite dar seguimiento al desempeño de los estudiantes y realizar las asignaciones de ejercicios. La configuración de cada asignación, incluye la selección del conjunto de problemas, la fecha y hora para el inicio de su visualización por parte de los estudiantes, así como la fecha y hora límite para el envío de las propuestas de solución. OmegaUp valida el periodo de vigencia de cada asignación.

La primera actividad de la estrategia de intervención, que corresponde a la práctica de recuperación repetida, inició a partir del tercer día de clase. Durante cada práctica, se otorgó un periodo de 10 minutos para que los alumnos resolvieran de manera individual un ejercicio de complejidad baja, con la temática principal de la clase anterior, es importante destacar nuevamente, que la plataforma otorgó retroalimentación inmediata al alumno. Adicionalmente, al finalizar el periodo de práctica, el profesor realizó la retroalimentación de manera grupal, indicando la solución correcta del problema y haciendo hincapié en los tópicos importantes que el estudiante debía recordar.

Al finalizar la primera semana del curso, se realizó la asignación de la tarea semanal, compuesta por el conjunto de problemas de complejidad media, sobre los temas discutidos en dicho periodo. Durante el periodo de vigencia de la tarea, los estudiantes realizaron la codificación de sus programas para dar solución a los ejercicios y contaban con la posibilidad de organizar su trabajo, enviando sus propuestas en cualquier instante del día, ya que la plataforma se encuentra disponible en línea. OmegaUp evaluó los programas enviados y ofreció retroalimentación inmediata a los alumnos, en consecuencia, no tenían que esperar un tiempo considerable para obtener retroalimentación por parte del profesor, como sucede en un curso tradicional. Este procedimiento permitió, ofrecer a los estudiantes un catálogo más numeroso de ejercicios, en comparación al enfoque tradicional, de tal manera que se les brindó la posibilidad de realizar una mayor práctica de sus habilidades de programación. Incluso los estudiantes tuvieron la opción de resolver ejercicios no asignados oficialmente en el curso, pero que se encontraban disponibles en la plataforma. Al finalizar la primera semana del último periodo del curso (con duración de dos semanas), se realizó la asignación final de un ejercicio de complejidad alta a los estudiantes, el cual, cubría los tópicos principales de

**Tabla 3**. Cantidad y complejidad de ejercicios de programación para el curso de programación, por semana y unidad temática

| Unidad temática | Número de semana del curso | Cantidad de ejercicios de complejidad baja para el repaso de la clase anterior | Cantidad y complejidad de ejercicios para las tareas |
|---|---|---|---|
| Introducción al lenguaje de programación C | 1 | 3 | 5 de complejidad media |
| Estructuras de control (secuencial, selectivas, iterativas) | 2 | 5 | 5 de complejidad media |
| Estructuras de control (secuencial, selectivas, iterativas) | 3 | 5 | 5 de complejidad media |
| Estructuras de control (secuencial, selectivas, iterativas) | 4 | 5 | 5 de complejidad media |
| Programación modular | 6 | 4 | 5 de complejidad media |
| Programación modular | 7 | 5 | 6 de complejidad media |
| Arreglos | 8 | 5 | 5 de complejidad media |
| Arreglos | 9 | 5 | 6 de complejidad media |
| Apuntadores | 11 | 4 | 5 de complejidad media |
| Apuntadores | 12 | 5 | 5 de complejidad media |
| Tipos de datos definidos por el programador | 13 | 5 | 5 de complejidad media |
| Tipos de datos definidos por el programador | 14 | 5 | 5 de complejidad media |
| Archivos | 16 | 4 | 1 de complejidad alta |

todo el programa educativo. Los estudiantes resolvieron esta asignación final en equipos de dos personas. Durante la ejecución del curso, el profesor realizó por medio de la plataforma, un seguimiento personalizado del progreso de sus estudiantes, consultando para cada uno, la cantidad de problemas resueltos completamente, resueltos parcialmente (con los casos no exitosos), o bien, los no resueltos.

## Resultados

Al finalizar la ejecución del curso, se realizó una entrevista estructurada a todos los estudiantes que participaron en el curso, con el objetivo de conocer su perspectiva y experiencias en el uso del juez en línea OmegaUp, en el contexto de la estrategia de intervención basada en los principios de ciencia cognitiva. A continuación, se presentan los resultados obtenidos en base a las respuestas y organizados por categorías:

### Uso de la plataforma OmegaUp para las actividades de repaso diarias

- Los estudiantes argumentaron que la resolución diaria de ejercicios en la plataforma, les permitió recordar y consolidar los temas vistos previamente.
- El nivel de complejidad de estos ejercicios, fue considerado apropiado para el periodo de duración de la actividad.
- Consideran que la retroalimentación de la plataforma no es suficiente, ya que no les presenta información más detallada del origen de sus errores, y esto no les permite avanzar en la resolución de algunos ejercicios.
- Argumentan que la retroalimentación por parte del profesor al concluir la práctica es adecuada y les permitió aclarar sus dudas, ya que con la explicación del profesor y con la confirmación de la plataforma pudieron detectar sus errores y entender la solución correcta al ejercicio.

### Uso de la plataforma OmegaUp para las asignaciones semanales

- Los estudiantes consideran que el uso de la plataforma en línea para las actividades semanales, les permitió una práctica continua a lo largo del curso, y esto les ayudo a mejorar sus habilidades de programación.
- Consideran que la cantidad de ejercicios fue numerosa, pero el periodo de tiempo para enviar sus soluciones era adecuado. Destacan que tenían que organizar su tiempo para codificar sus programas ya que la plataforma no aceptaba envíos después de la fecha límite de entrega.
- El nivel de complejidad (en este caso medio) lo consideran apropiado, sin embargo, refieren que una cantidad mínima de ejercicios presentaban una redacción confusa, por lo cual no podían avanzar en su propuesta de solución, hasta obtener retroalimentación por parte del profesor.
- En otros casos aislados, el conjunto de casos de prueba para el ejercicio no era correcto, y aunque la solución de los estudiantes era válida, el juez lo evaluaba de manera negativa, situación que les generó frustración.
- Señalan que se presentaron situaciones de plagio de los programas por parte de sus compañeros.
- Nuevamente destacan que la retroalimentación de la plataforma no es suficiente, ya que no les presenta información más detallada del origen de sus errores.

### Uso de la plataforma OmegaUp para la asignación final

- Los estudiantes consideran que el uso de la plataforma en línea para el ejercicio final, les permitió reforzar y practicar los temas principales de todo el curso.
- El nivel de complejidad del ejercicio final lo consideraron apropiado, y era factible resolverlo en el tiempo indicado para la asignación.

Finalmente, con el objetivo de determinar el desempeño de los estudiantes, se calculó la tasa de aprobación del curso, con un valor del 69.2%, que es ligeramente mayor al promedio de la tasa de aprobación del 67.7%, calculada a nivel mundial en el trabajo de Watson y Li (2014). Del conjunto de estudiantes que aprobaron el curso, estos resolvieron el 81% del total de ejercicios de programación semanales asignados a través de la plataforma y el 92% del ejercicio final de complejidad alta, de tal manera, que estos alumnos ejecutaron las prácticas de programación de manera consistente, de acuerdo a la planeación de la estrategia de intervención.

## Conclusiones

A partir de los resultados obtenidos en las entrevistas, se puede concluir que desde la perspectiva de los estudiantes el uso del juez en línea OmegaUp en el contexto de la estrategia de aprendizaje basada en los principios cognitivos, contribuyó a consolidar sus habilidades prácticas de programación, argumentan que esto se debe a que resolvieron ejercicios de programación de manera continua e intensiva durante todo el curso, por medio, de las prácticas de repaso diarias, las

asignaciones semanales y la asignación final. La plataforma les ofreció un compendio en línea, de numerosos ejercicios de complejidad baja, media y alta. Sin embargo, se destacan algunas situaciones a tomar en cuenta, para mejorar la implantación de la estrategia. En primer término, se expresa que, aunque la retroalimentación de la plataforma es inmediata, no ofrece el nivel de detalle suficiente para orientar al estudiante en la detección del origen de sus errores, funcionalidad que es necesaria en el POJ, sobre todo en esta etapa del aprendizaje. Por lo que es imprescindible establecer la cantidad apropiada de retroalimentación, así como el momento preciso en que se debe ofrecer. Otro aspecto a destacar es la presencia de plagio en las soluciones enviadas por parte de los estudiantes, en este aspecto, resulta importante que la plataforma utilizada incluya la funcionalidad para la detección automática de código fuente plagiado, de tal manera que los programas con código similar no sean aceptados como válidos. En relación a los ejercicios de programación, es importante realizar un proceso de validación y verificación en la creación de cada nuevo ejercicio, de tal manera que la redacción de los problemas sea clara y precisa, y los casos de prueba presenten una definición correcta, para evitar situaciones de frustración en los estudiantes, que afecte su motivación y disposición para realizar prácticas de programación.

Como trabajo a futuro, se planea integrar el uso de una plataforma POJ bajo la estrategia de aprendizaje basada en principios cognitivos, en el contexto de un marco de referencia pedagógico, que ofrezca un contexto estandarizado y organizado, con un conjunto de definiciones, métodos, herramientas y técnicas, que permita alcanzar el objetivo de mejorar el proceso de enseñanza/aprendizaje para un curso de programación estructurada a nivel universitario y con ello contribuir a reducir la tasa de reprobación y deserción.

## Bibliografía


Amado Moreno, M. G., García Velázquez, Á., Brito Páez, R. A., Sánchez Luján, B. I., y Sagaste Bernal, C. A. (2014). Causa de reprobación en ingeniería desde la perspectiva del académico y administradores. *Ciencia y Tecnología*, 1(14), 233-249.

AMITI (2013), Mapa de ruta 2025, para transformar a México a través de la adopción de TIC. México, AMITI, Select, IMCO.

Bennedsen, J., y Caspersen, M. E. (2019). Failure rates in introductory programming. *ACM Inroads,* 10(2), 30–36.

Bjork. R. A. y Yan, V. X. (2014). The Increasing importance of learning how to learn. In M. McDaniel, R. Frey, S. Fitzpatrick, & H.L. Roediger (Eds), *Integrating cognitive science with innovative teaching in STEM disciplines*, (pp.15-36).

Boekaerts, M., y Cascallar, E. (2006). How far we have moved toward the integration of theory and practice in self-regulation. *Educational Psychology Review*, 18, 199–210.

Bonwell, C., y Eison, J. (1991). Active learning: Creating excitement in the classroom (ASHE-ERIC Higher Education Report No. 1). George Washington University, Washington, DC.

Butler, A. C., Marsh, E. J., Slavinsky, J. P., y Baraniuk, R. G. (2014). Integrating Cognitive Science and Technology Improves Learning in a STEM Classroom. *Educational Psychology Review*, 26(2), 331–340.

Cepeda, N. J., Pashler, H., Vul, E., Wixted, J. T., & Rohrer, D. (2006). Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin*, 132(3), 354–380.

Combéfis, S., y de Moffarts, G. (2019). *Automated Generation of Computer Graded Unit Testing-Based Programming Assessments for Education.* In 6th International Conference on Computer Science, Engineering and Information Technology (CSEIT-2019).

Gomes, A., Brito, F. y Henriques, P. (2016). Types of assessing student-programming knowledge. *Frontiers in Education* 2016, 1-8.

Guel, C. J., y Araiza, M. J. (2015). A Reflection on the Determinants of Development of Software Industry in Mexico. *Daena: International Journal of Good Conscience*, 10(3), 71-79.

Hattie, J., y Timperley, H. (2007). The Power of Feedback. *Review of Educational Research*, 77(1), 81–112.

Ismail, M. N., Ngah, N. A., y Umar, I. (2010). The Effects of Mind Mapping with Cooperative Learning on Programming Performance, Problem Solving Skill and Metacognitive Knowledge among Computer Science Students. *Journal of Educational Computing Research*, 42(1), 35-61.

Jazayeri, M. (2015). *Combining Mastery Learning with Project-Based Learning in a First Programming Course: An Experience Report*. In Proceedings of the 37th IEEE/ACM International Conference on Software Engineering, IEEE, 2, 315-318.

Juárez, J. L., López, M. C. y Villareal, J. (2016). Estrategias para Reducir el Índice de Reprobación en Fundamentos de Programación de Sistemas Computacionales del I.T. Mexicali. *Revista de Gestión Empresarial y Sustentabilidad*, 2(1), 25-41.

Kirschner, P. A., Sweller, J., y Clark, R. E. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist*, 41(2), 75–86.

Locke, E. A., y Latham, G. P. (1990). A Theory of Goal Setting and Task Performance. Englewood Cliffs, NJ: Prentice Hall.

Luca Bez, J., E. Ferreira, C., y A. Tonin, N. (2013). *URI Online Judge Academic: A Tool for Professors*. In Proceedings of the 2013 International Conference on Advanced ICT.

Lykke, M., Coto, M., Mora, S., Vandel, N., y Jantzen, C. (2014). *Motivating programming students by problem based learning and LEGO robots*. In IEEE Global Engineering Education Conference, 544–555.

Mejía, A. O. (2017). *Personal calificado en la industria de las TICs en México: el caso de las MYPYMES*. En XVII Congreso Latino-Iberoamericano de Gestión Tecnológica-ALTEC 2017.

Paiva, J., Leal, J. P., y Queiros, R. (2016). *Enki: A Pedagogical Services Aggregator for Learning Programming Languages.* In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, 332-337.

Rangel, O., García, L. y Habib, L. (2012). Eficiencia terminal de la materia de programación estructurada. *Revista Congreso Universidad*, 1(3), 1-9.

Restrepo-Calle, F., Ramírez-Echeverry, J. J., y Gonzalez, F. A. (2018). UNCode: *Interactive system for learning and automatic evaluation of computer programming skills*. In Proceeding of EDULEARN18.

Roediger, H. L., y Butler, A. C. (2011). The critical role of retrieval practice in long-term retention. *Trends in Cognitive Sciences*, 15(1), 20–27.

Rodríguez Pérez, I. (2017). La calidad de la educación superior y la reestructuración del programa de tutoría / The quality of higher education and the restructuring of the tutoring program. RIDE *Revista Iberoamericana para la Investigación y el Desarrollo Educativo*, 8(15), 135–154.

Rongas, T., Kaarna, A., y Kalviainen, H. (2004). *Classification of computerized learning tools for introductory programming courses: learning approach*. In Proceedings of the IEEE International Conference on Advanced Learning Technologies, 2004.

Silva, B., & Madeira, R. N. (2010). *A study and a proposal of a collaborative and competitive learning methodology*. In Education Engineering (EDUCON), IEEE.

U.S. Department of Labor, Bureau of Labor Statistics. (2020). Computer and Information Technology Occupations. Occupational outlook handbook, 2020. Recuperado de: https://www.bls.gov/ooh/computer-and-information-technology/home.htm?view_full.

Watson, C. & Li, F.W.B. (2014). *Failure Rates in Introductory Programming Revisited*. In Proceedings of the 19th annual Conference on Innovation and Technology in Computer Science Education – ITiCSE'14, New York, NY, USA ACM, 39–44.

Wang, G. P., Chen, S. Y., Yang, X., & Feng, R. (2015). OJPOT: online judge & practice oriented teaching idea in programming courses. *European Journal of Engineering Education*, 41(3), 304–319.

Yera, R., Rodríguez, R. M., Castro, J., & Martínez, L. (2017). A Recommender System for Supporting Students in Programming Online Judges. *Smart Innovation, Systems and Technologies*, 215–224.