CENTRO DE CIECIAS BÁSICAS

DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN

TESIS

Uso del efecto de auto-explicación como apoyo en el aprendizaje de la programación, aplicando la teoría de carga cognitiva

QUE PRESENTA

IC. Carlos Sandoval Medina

PARA OBTENER EL GRADO DE MAESTRO EN INFORMÁTICA Y TECNOLOGÍAS COMPUTACIONALES

COMITÉ TUTORAL

Cotutor: Dr. Carlos Argelio Arévalo Mercado

Cotutora: Dra. Estela Lizbeth Muñoz Andrade

Asesor: MC. Julio Raymundo Dena Garza

Aguascalientes Ags., Marzo de 2022.

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

CARTA DE VOTO APROBATORIO
INDIVIDUAL

M. EN C. JORGE MARTÍN ALFÉREZ CHÁVEZ
DECANO (A) DEL CENTRO **DE CIENCIAS BÁSICAS**

P R E S E N T E

Por medio del presente como ***CO-TUTOR*** designado del estudiante ***CARLOS SANDOVAL MEDINA*** con ID ***285667*** quien realizó **la tesis** titulada: ***USO DEL EFECTO DE AUTO-EXPLICACIÓN COMO APOYO EN EL APRENDIZAJE DE LA PROGRAMACIÓN, APLICANDO LA TEORÍA DE CARGA COGNITIVA***, un trabajo propio, innovador, relevante e inédito y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia doy mi consentimiento de que la versión final del documento ha sido revisada y las correcciones se han incorporado apropiadamente, por lo que me permito emitir el **VOTO APROBATORIO**, para que ***él*** pueda proceder a imprimir***la*** así como continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a 7 de *Marzo* de *2022*.

Dr. Carlos Argelio Arévalo Mercado
Co-Tutor de *tesis*
Profesor Investigador, Departamento Sistemas de Información

c.c.p.- Interesado
c.c.p.- Secretaría Técnica del Programa de Posgrado

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

CARTA DE VOTO APROBATORIO
INDIVIDUAL

M. EN C. JORGE MARTÍN ALFÉREZ CHÁVEZ
DECANO (A) DEL CENTRO **DE CIENCIAS BÁSICAS**

_____
P R E S E N T E

Por medio del presente como ***CO-TUTORA*** designado del estudiante ***CARLOS SANDOVAL MEDINA*** con ID ***285667*** quien realizó **la tesis** titulada: ***USO DEL EFECTO DE AUTO-EXPLICACIÓN COMO APOYO EN EL APRENDIZAJE DE LA PROGRAMACIÓN, APLICANDO LA TEORÍA DE CARGA COGNITIVA***, un trabajo propio, innovador, relevante e inédito y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia doy mi consentimiento de que la versión final del documento ha sido revisada y las correcciones se han incorporado apropiadamente, por lo que me permito emitir el **VOTO APROBATORIO**, para que ***él*** pueda proceder a imprimir***la*** así como continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a 4 de *Marzo* de *2022*.

Dra. Estela Lizbeth Muñoz Andrade
Co-Tutora de *tesis*
Profesora Investigadora, Departamento de Sistemas Electrónicos

c.c.p.- Interesado
c.c.p.- Secretaría Técnica del Programa de Posgrado

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

CARTA DE VOTO APROBATORIO
INDIVIDUAL

M. EN C. JORGE MARTÍN ALFÉREZ CHÁVEZ
DECANO (A) DEL CENTRO **DE CIENCIAS BÁSICAS**

P R E S E N T E

Por medio del presente como ***ASESOR*** designado del estudiante ***CARLOS SANDOVAL MEDINA*** con ID ***285667*** quien realizó **la tesis** titulada: ***USO DEL EFECTO DE AUTO-EXPLICACIÓN COMO APOYO EN EL APRENDIZAJE DE LA PROGRAMACIÓN, APLICANDO LA TEORÍA DE CARGA COGNITIVA***, un trabajo propio, innovador, relevante e inédito y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia doy mi consentimiento de que la versión final del documento ha sido revisada y las correcciones se han incorporado apropiadamente, por lo que me permito emitir el **VOTO APROBATORIO**, para que ***él*** pueda proceder a imprimir***la*** así como continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a 4 de *Marzo* de *2022*.

Mtro. Julio Raymundo Dena Garza
Asesor de *tesis*
Profesor Investigador, Departamento de Sistemas de Información

c.c.p.- Interesado
c.c.p.- Secretaría Técnica del Programa de Posgrado

DICTAMEN DE LIBERACION ACADEMICA PARA INICIAR LOS TRAMITES DEL EXAMEN DE GRADO

Fecha de dictaminación dd/mm/aaaa: _____

NOMBRE: Carlos Sandoval Medina **ID** 285667

PROGRAMA: Maestría en Informática y Tencologías Computacionales **LGAC (del posgrado):** Gestión de sistemas y tecnologías de información para mejorar competitividad, innovación y cambio organizacional.

TIPO DE TRABAJO: (X) Tesis ( ) Trabajo Práctico

TITULO: Uso del efecto de auto-explicación como apoyo en el aprendizaje de la programación, aplicando la tería de carga cognitiva

IMPACTO SOCIAL (señalar el impacto logrado): Mediante esta tesis se contribuyó con material instruccional para el apoyo en el aprendizaje de la programación, y reducir asi el porcentaje de reprobación en las carreras afines a la programación, con resultados comprobados, dejando una base para futuras investigaciones

INDICAR SI NO N.A. (NO APLICA) SEGÚN CORRESPONDA:

	Elementos para la revisión académica del trabajo de tesis o trabajo práctico:
SI	El trabajo es congruente con las LGAC del programa de posgrado
SI	La problemática fue abordada desde un enfoque multidisciplinario
SI	Existe coherencia, continuidad y orden lógico del tema central con cada apartado
SI	Los resultados del trabajo dan respuesta a las preguntas de investigación o a la problemática que aborda
SI	Los resultados presentados en el trabajo son de gran relevancia científica, tecnologica o profesional según el área
SI	El trabajo demuestra más de una aportación original al conocimiento de su área
SI	Las aportaciones responden a los problemas prioritarios del país
SI	Generó transferecia del conocimiento o tecnológica
SI	Cumple con la ética para la investigación (reporte de la herramienta antiplagio)
	El egresado cumple con lo siguiente:
SI	Cumple con lo señalado por el Reglamento General de Docencia
SI	Cumple con los requisitos señalados en el plan de estudios (créditos curriculares, optativos, actividades complementarias,estancia, predoctoral, etc)
SI	Cuenta con los votos aprobatorios del comité tutoral, en caso de los posgrados profesionales si tiene solo tutorpodrá liberar solo el tutor
SI	Cuenta con la carta de satisfacción del Usuario
SI	Coincide con el título y objetivo registrado
SI	Tiene congruencia con cuerpos académicos
SI	Tiene el CVU del Conacyt actualizado
NA	Tiene el artículo aceptado o publicado y cumple con los requisitos institucionales (en caso que proceda)
	En caso de Tesis por artículos científicos publicados
NA	Aceptación o Publicación de los articulos según el nivel del programa
NA	El estudiante es el primer autor
NA	El autor de correspondencia es el Tutor del Núcleo Académico Básico
NA	En los artículos se ven reflejados los objetivos de la tesis, ya que son producto de este trabajo de investigación.
NA	Los artículos integran los capítulos de la tesis y se presentan en el idioma en que fueron publicados
NA	La aceptación o publicación de los artículos en revistas indexadas de alto impacto

Con base a estos criterios, se autoriza se continúen con los trámites de titulación y programación del examen de grado:

Sí X

No _____

FIRMAS

Elaboró:

* NOMBRE Y FIRMA DEL CONSEJERO SEGÚN LA LGAC DE ADSCRIPCION: Dr. Cesar Eduardo Velázquez Amador

NOMBRE Y FIRMA DEL SECRETARIO TÉCNICO: MITC. Jorge Eduardo Macías Luévano

*** En caso de conflicto de intereses, firmará un revisor miembro del NAB de la LGAC correspondiente distinto al tutor o miembro del comité tutoral, asignado por el Decano**

Revisó:

NOMBRE Y FIRMA DEL SECRETARIO DE INVESTIGACIÓN Y POSGRADO: Dra. Hydee Martínez Ruvalcaba

Autorizó:

NOMBRE Y FIRMA DEL DECANO: M.C. Jorge Martín Alférez Chávez

Nota: procede el trámite para el Depto. de Apoyo al Posgrado

En cumplimiento con el Art. 105C del Reglamento General de Docencia que a la letra señala entre las funciones del Consejo Académico: .... Cuidar la eficiencia terminal del programa de posgrado y el Art. 105F las funciones del Secretario Técnico, llevar el seguimiento de los alumnos.

AGRADECIMIENTOS

A CONACYT por el apoyo económico brindado a lo largo de estos dos años de posgrado.

A la Universidad Autónoma de Aguascalientes por brindarme la oportunidad de cursar un posgrado incorporado al programa nacional de posgrados de calidad, teniendo acceso a una educación de calidad.

A el Dr. Carlos Argelio Arévalo Mercado, la Dra. Estela Lizbeth Muñoz Andrade y el MC. Julio Raymundo Dena Garza, por el apoyo y tiempo otorgado a lo largo de estos dos años para la realización del presente trabajo de tesis.

A todos los profesores de cada una de las materias impartidas en el programa académico del posgrado.

A los alumnos de licenciatura que apoyaron con su participación en la aplicación del estudio de investigación.

DEDICATORIAS

A mi familia

Por apoyarme y creer siempre en mi para conseguir este importante paso, y por ser lo más importante que tengo en mi vida.

A mis padres

Por todo lo que me enseñaron y todo el apoyo que me dieron y aún me siguen dando.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

RESUMEN

A lo largo de la historia hasta la actualidad, el aprendizaje de la programación en las carreras afines a las ciencias de la computación ha representado un reto para los estudiantes, hasta el punto de convertirse en un problema académico – social, generando altos índices de reprobación y abandono en las carreras afines a esta materia.

En la búsqueda de alternativas que ayuden a los estudiantes en el proceso de aprendizaje de la programación, se desarrolla la presente investigación, la cual tiene como objetivo diseñar, desarrollar, probar y medir la efectividad de material instruccional tomando como base la teoría de carga cognitiva, utilizando específicamente ejercicios diseñados atendiendo al efecto de auto-explicación de dicha teoría, así como la identificación de temas estratégicos de programación.

La teoría de carga cognitiva es una teoría que alinea sus procesos con la arquitectura cognitiva humana y utiliza la información de la cognición humana para el desarrollo de material instruccional que facilite la transferencia de conocimientos a la memoria de largo, la cual es ilimitada, donde la información es organizada en esquemas los cuales pueden durar toda la vida y llegar a no consumir recursos cognitivos mediante la práctica y automatización (John Sweller et al., 2019).

La auto-explicación es una actividad cognitiva constructiva que los alumnos pueden realizar, a voluntad o en respuesta a una indicación, para comprender nueva información o aprender habilidades. Al auto explicarse, los estudiantes generan inferencias sobre relaciones

conceptuales que mejoran la comprensión, reforzando los conocimientos previos existentes y monitoreando el grado de comprensión de estos (John Sweller et al., 2019).

Atendiendo los fundamentos teóricos se diseñó y desarrolló el material instruccional, aplicándolo en un primer estudio mediante un cuasiexperimento a los alumnos de primer año de la carrera de Ingeniería en Sistemas Computacionales de la Universidad Autónoma de Aguascalientes, la medición de la efectividad se realizó mediante exámenes estandarizados y calibrados por el departamento de sistemas electrónicos utilizando un diseño PRE - POST, con base en los resultados se realiza la réplica del primer estudio refinando el diseño y aplicación del material instruccional, combinando el efecto de auto-explicación con el efecto de ejemplo resuelto mediante la adición de videos de ejemplos resueltos a los ejercicios diseñados en formato del efecto de auto-explicación, de igual forma la medición de la efectividad se realizó mediante exámenes estandarizados y calibrados por el departamento de sistemas electrónicos.

Los resultados obtenidos reflejan un impacto positivo de la utilización de material instruccional diseñado utilizando la teoría de carga cognitiva como base. Además, puede confirmarse que la combinación de efectos de la teoría de carga cognitiva como el ejemplo resuelto y problema por completar, con el efecto de auto-explicación producen resultados más efectivos.

ABSTRACT

Historically, the learning of programming in careers related to computer science has represented a difficulty for students, becoming an academic-social problem, generating high rates of failure and abandonment in careers related to this subject.

Looking for alternatives that help students in the learning process of programming, this research is developed, which aims to design, develop, test, and measure the effectiveness of instructional material based on the cognitive load theory, using specifically designed exercises considering the self-explanation effect, as well as the identification of strategic programming themes.

The cognitive load theory is a theory that aligns its processes with the human cognitive architecture and uses the information of human cognition for the development of instructional material that facilitates the transfer of knowledge to long memory, which is unlimited, where the information is organized in schemes which can last a lifetime and not consume cognitive resources through practice and automation (John Sweller et al., 2019).

Self-explanation is a constructive cognitive activity that students can perform, at will or in response to a prompt, to understand new information or learn skills. By self-explaining, students generate inferences about conceptual relationships that improve understanding, reinforcing existing previous knowledge and monitoring the degree of understanding of these (John Sweller et al., 2019).

Following the theoretical foundations, the instructional material was designed and developed, applying it in a first study through a quasi-experiment to the first-year students of the Computer Systems Engineering career of the Autonomous University of Aguascalientes, the effectiveness measurement was carried out through exams standardized and calibrated by the electronic systems department using a pre-post design, based on the results, the replica of the first study is carried out, refining the design and application of the instructional material, combining the self-explanation effect with the resolved example effect, adding videos of solved examples to the exercises designed in the format of the self-explanation effect, in the same way the effectiveness measurement was carried out through standardized and calibrated tests by the electronic systems department.

The results obtained reflect a positive impact of the use of instructional material designed using the cognitive load theory as a basis. Furthermore, it can be confirmed that the combination of cognitive load theory effects such as the solved example and problem to be completed, with the self-explanation effect produce more effective results.

CAPITULO I: INTRODUCCIÓN

Industria de software

La rápida evolución de la industria del software es evidente si se observa que a mediados de la década del 2000 ésta era predominantemente un negocio de productos, con un modelo tradicional de venta de aplicaciones “empaquetadas” (en 2005 representaba 49% de los ingresos totales). Para 2008, éstas habían caído al 15% (Yrjönkoski, Helander, & Jaakkola, 2016). A inicios del año 2020, el software es una parte orgánica de un grupo creciente de productos y nuevas formas de generar ingresos y generar ventajas estratégicas, lo que ha trasladado las fronteras de tal industria hacia un modelo principalmente de servicios, fuertemente basado en la nube, con una predominante utilización de dispositivos móviles, reemplazando en buena medida los canales tradicionales de ventas y distribución.

De los procesos que requiere la Ingeniería de software para su producción, es la programación la de mayor relevancia. Sin embargo, el aprendizaje de la programación ha sido ampliamente documentado como de gran dificultad para los principiantes de las carreras afines a las ciencias computacionales (Bain & Barnes, 2014; Hu, 2006; Jenkins, 2002; Pears et al., 2007) lo que suele provocar porcentajes de reprobación cercanos al 34%, incluso a nivel mundial (Borzovs, Niedrite, & Solodovnikova, 2015; C. Watson & Li, 2014).

Desarrollar software implica, necesariamente, por lo menos dominar un lenguaje de programación para una plataforma específica, ya sea para aplicaciones de escritorio, aplicaciones para sitios Web, dispositivos móviles o sistemas de información administrativos. Esta situación ha derivado en una alta demanda mundial y nacional de programadores (Alatorre, 2019; Heraysymchuk, 2019; Scorneica, 2019).

Ubicuidad de las Tecnologías de Información

De acuerdo con el INEGI, en la Encuesta Nacional sobre Disponibilidad y Uso de Tecnologías de la Información en los Hogares (ENDUTIH) 2018, se calculó que los usuarios de telefonía celular representan el 72.2% de la población de seis años o más y al menos 8 de cada 10 de estos usuarios cuentan con un teléfono inteligente. La encuesta también revela, que de los usuarios que se conectan a la internet mediante dispositivos móviles, el 96.9% lo hace para obtener información; 91.4%, entretenimiento; 90.0%, comunicación; 78.1%, acceso a contenidos audiovisuales, y sólo 76.6% para el acceso a redes sociales (INEGI et al., 2018).

Las computadoras, dispositivos móviles y nacientes tecnologías (IoT), se han vuelto omnipresentes en nuestra vida personal y laboral, y esto nos lleva a la necesidad de desarrollar el software adecuado que cumpla con los requerimientos de los dispositivos actuales, lo cual indica una creciente área de oportunidad, así como la demanda de profesionistas

en el área de programación (Kato & Shimakage, 2020). En 2011, Marc Andreessen señaló *"Software is eating the world"* (Andreessen, 2011).

Peng Wu (Fraser et al., 2015) señala que "*el futuro de la programación es impulsado por tres importantes cambios, ¿Qué puede ser programado? Hardware destino, ¿Cómo es programado? Modelo de negocio Y ¿Quién programa? Casi cualquiera puede ser programador y muchos de ellos no pertenecen a las ciencias de la computación".* Tomando como referencia esta cita, la importancia de la programación crece a medida que estos temas evolucionan.

Situación de reprobación en programación

Para una buena formación de profesionales en el área de las ciencias computacionales la enseñanza de la programación es uno de los factores o puntos más importantes dentro los conocimientos que debe poseer un profesional en el área de la computación. Esto no es una tarea fácil puesto que la programación requiere que el estudiante posea varias habilidades cognitivas mencionadas previamente, si estas habilidades no han sido desarrolladas previamente se constituyen como factores negativos para el aprendizaje. Esta afirmación se puede constatar con los altos índices de deserción y reprobación de cursos de programación, los cuales oscilan entre el 25 y 80 por ciento en Estados Unidos (Insuasti, 2016).

En 2019 un estudio realizado por (Simon et al., 2019), realizan una comparación de los índices de reprobación en las materias introductorias de programación con otras materias de ciencias básicas como matemáticas, física , química, ciencias de la computación, bilogía e ingeniería, tomando como muestra 17 universidades de diferentes países, de acuerdo a sus resultados señalan un porcentaje de aprobación de 75%, que en comparación con estudios de años anteriores (Bennedsen & Caspersen, 2007) reportan un porcentaje de aprobación del 67% y (Bennedsen & Caspersen, 2019) replica de su estudio en 2007 reportan un porcentaje de aprobación del 73%, donde las cifras se han mantenido estable con un ligero incremento en el transcurso de los últimos 15 años. En comparación con las materias de ciencias básicas en sus resultados presentan un 3% de diferencia en los porcentajes de aprobación, concluyen que los índices de reprobación en las materias introductorias de programación no son tan altos, pero igual aún queda camino para mejorar.

En la mayoría de los estudios respecto al ¿por qué? de los altos índices de fallo en la aprobación de cursos introductorios de programación hacen referencia a la motivación como uno de los factores más importantes, estos resultados se obtienen a base de experiencia y de manera cualitativa. (Watson & Li, 2014) Centran su investigación en resultados cuantitativos, realizaron una revisión y análisis de 161 cursos introductorios de programación de 15 países diferentes en un total de 51 instituciones. Los resultados generales obtenidos del estudio arrojaron una tasa de aprobación de 67.7 %, en la opinión de los autores

el resultado no les parece alarmante, pero están de acuerdo en que se debe mejorar. A continuación, se desglosan los resultados:

Tabla 1. Resultados tasa de aprobación (Watson & Li, 2014).

Pass Rate	Courses	Pass %
Overall CS1	161	67.7
Colleges	16	79.9
Universities	145	66.4
Small Class	10	80.1
Large Class	91	65.4

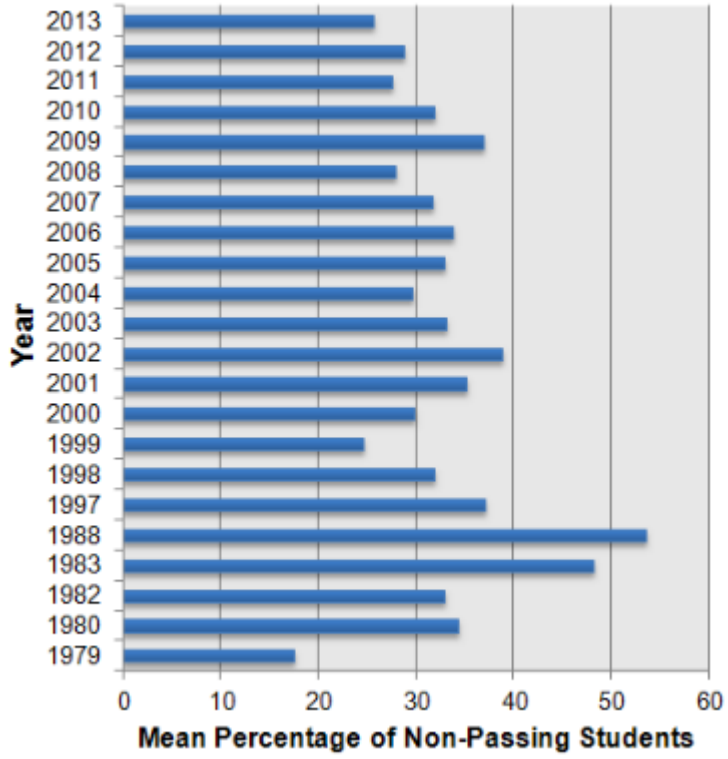


Ilustración 1. Reprobación de estudiantes por año en los cursos de programación (Watson & Li, 2014).

Borzovs et al., (2015) reportan que la deserción en las carreras de las ciencias de la computación arroja cada año un promedio de 30%

después del primer semestre y cerca del 50% sobre el primer año de estudio. En su estudio, analizan cuatro factores que consideran influyen en los altos índices de deserción, uno de los cuales es el curso introductorio de programación. También indican que el factor de mayor relevancia es el promedio general obtenido a nivel preparatoria. Sin embargo, encontraron que *los cursos básicos* en las carreras de las ciencias de la computación como lo son la introducción a la programación influyen en el abondo escolar de dichas carreras.

Bennedsen & Caspersen, (2007) reportaron que los índices de reprobación y abandono en los cursos introductorios de programación son en promedio de 33%. La investigación abarcó varias instituciones en diferentes partes del mundo (Estados Unidos, Australia, países de Europa y África, ninguno de América latina). En 2019, se realiza una réplica de la investigación donde se incluyeron en el estudio países de Asia y Sudamérica, los resultados obtenidos cambiaron un poco en relación al estudio presentado en 2007, en este caso reportan un promedio de 28% en el índice de reprobación y abandono (Bennedsen & Caspersen, 2019).

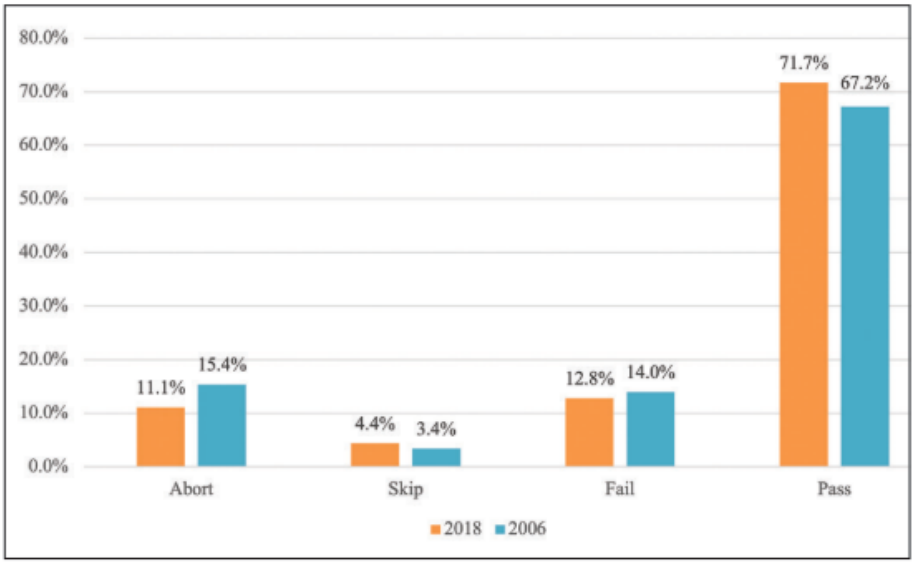


Ilustración 2. Comparación índice de reprobación y abandono en cursos introductorios de programación reportados en artículos del 2007 y 2019 (Bennedsen & Caspersen, 2019).

La mayoría de los estudios enlistados fueron realizados en su mayoría en países como Estados Unidos, países de Europa y algunos de Asía, solo uno de ellos menciona la inclusión de países de América latina. De tal suerte, Beltrán et al., (2015) presentan un análisis cuantitativo de los índices de reprobación de la materia Programación I impartida el primer semestre de las carreras de sistemas en la Universidad Central de Ecuador, tomando como periodo de tiempo para el análisis del año 2009 al 2015 y los resultados obtenidos muestran un 47% de reprobación.

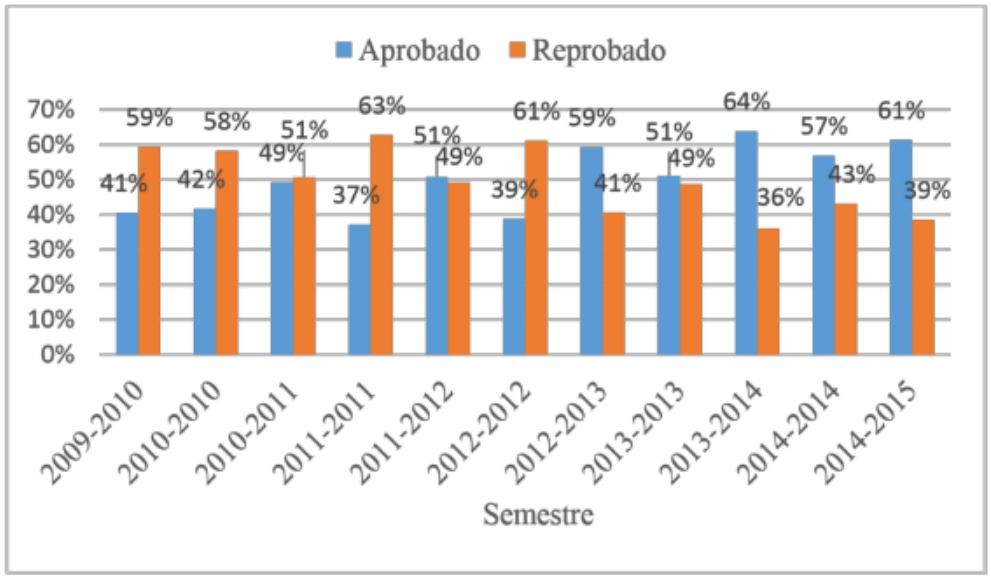


Ilustración 3. Porcentaje de aprobación y reprobación de la materia Programación I de la Universidad Central de Ecuador del 2009 al 2015 (Beltrán et al., 2015).

En México, particularmente en el Instituto Tecnológico de Mexicali, en donde se planteó el uso de herramientas de apoyo para recudir el índice de reprobación de la materia fundamentos de programación que se imparte en la carrera de Sistemas Computacionales, los registros de los índices de reprobación de dicha materia tomados para la investigación abarcan el periodo del año 2012 a 2015, el porcentaje de reprobación promedio de ese periodo fue 49.7%. El porcentaje más bajo de reprobación fue en el periodo 1 del 2014 con 27% y el más alto en el periodo 2 del 2013 con 66.3% (Juáres et al., 2016).

Tabla 2. Reprobación materia fundamentos de programación del Instituto Tecnológico de Mexicali, México (Juáres et al., 2016).

Estudiantes inscritos en la materia				*No acreditados en la Materia*	
Periodo	*Nombre Periodo*	*Nombre de la Materia*	*Estudiantes Inscritos*	*Cantidad*	*%*
20121	Enero Junio 2012	Fundamentos Programación	64	29	45.3%
20123	Agosto/Dic 2012	Fundamentos Programación	114	61	53.5%
20131	Enero Junio 2013	Fundamentos Programación	44	15	34.1%
20133	Agosto/Dic 2013	Fundamentos Programación	98	65	66.3%
20141	Enero Junio 2014	Fundamentos Programación	37	10	27%
20143	Agosto/Dic 2014	Fundamentos Programación	98	60	61.2%
20151	Enero Junio 2015	Fundamentos Programación	54	28	51.9%
			509	268	52.7%

En el caso concreto de la Universidad Autónoma de Aguascalientes, se tienen 3 carreras afines a las ciencias de la computación, Ingeniería en Computación Inteligente, Ingeniería en Sistemas Computacionales e Informática y Tecnologías Computacionales, el departamento de estadística de la Universidad Autónoma de Aguascalientes, reportan cada ciclo escolar las 5 materias con el más alto índice de reprobación por carrera, entre las cuales en cada ciclo escolar aparecen materias de programación en las carreras afines a las ciencias de la computación.

Tabla 3. Materias de programación reportadas en el reporte de materias con más alto índice de reprobación por carrera, tabla realizada a partir de la información del departamento de estadística de la Universidad Autónoma de Aguascalientes (2016 - 2019).

Carrera	**Materia**	**Porcentaje reprobación**	**Ciclo escolar**
Ingeniería en Computación Inteligente	Lenguajes de computación II	13.16 %	Enero – Junio 2016
Ingeniería en Sistemas Computacionales	Programación I	41.06 %	Enero – Junio 2016
Ingeniería en Sistemas Computacionales	Lógica de programación	33.54 %	Agosto – Diciembre 2016
Informática y Tecnologías Computacionales	Estructura de datos	18.75 %	Agosto – Diciembre 2016
Ingeniería en Sistemas Computacionales	Programación I	39.07 %	Enero – Junio 2017
Ingeniería en Sistemas Computacionales	Programación II	31.62 %	Agosto – Diciembre 2017
Ingeniería en Sistemas Computacionales	Lógica de programación	30.19 %	Agosto – Diciembre 2017
Informática y Tecnologías Computacionales	Programación estructurada	30.43 %	Enero – Junio 2017
Ingeniería en Sistemas Computacionales	Programación I	34.67 %	Enero – Junio 2018
Ingeniería en Sistemas Computacionales	Lógica de programación	28.48 %	Agosto – Diciembre 2018
Informática y Tecnologías Computacionales	Programación estructurada	28.26 %	Enero – Junio 2018

Informática y Tecnologías Computacionales	Algoritmos computacionales	45.65 %	Agosto – Diciembre 2018
Informática y Tecnologías Computacionales	Estructura de datos	23.81 %	Agosto – Diciembre 2018
Ingeniería en Computación Inteligente	Programación científica	10.26 %	Enero – Junio 2019
Ingeniería en Sistemas Computacionales	Programación I	27.15 %	Enero – Junio 2019
Informática y Tecnologías Computacionales	Programación estructurada	16.28 %	Enero – Junio 2019

Los estudios indican un alto índice de reprobación y abandono en las materias de cursos introductorios de programación tanto a nivel bachillerato como a nivel licenciatura, lo que nos obliga a buscar nuevas alternativas para elevar los índices de aprobación y de igual forma reducir la deserción en las carreras relacionadas con la programación. Pero no solo es el hecho de elevar el grado de aprobación, sino que los estudiantes generen competencias en el área de programación para cubrir los déficits actuales y futuros en el área.

CAPÍTULO II: ESTADO DEL ARTE

Aprendizaje de la programación

El aprender programación no solo implica aprender el lenguaje en sí, sino que es una competencia compleja la cual conlleva abstracción, refinamiento, modularidad, entre otros. El aprendizaje de la programación comprende aprender a analizar un problema y con base en ello formular posibles soluciones, seleccionar la más adecuada y probar/mejorar dicha solución. Lo cual se traduce en que el estudiante aprenda a resolver problemas y expresar la solución en una secuencia de instrucciones y estructuras de control básicas (lenguaje de programación). Compañ-Rosique et al., (2015) Afirman que "*Para cualquier persona diseñar la solución a un problema requiere de un esfuerzo importante de abstracción, aún más si tiene que expresarla en forma de algoritmo"* (p.12.) (Astudillo et al., 2018).

La naturaleza de los conceptos de programación sumado a las habilidades deseables que deben poseer los estudiantes para el aprendizaje de la programación y la forma en que los conocimientos se imparten al estudiante por medio de material instruccional, obligan a buscar alternativas para generar recursos didácticos que puedan ayudar en el proceso de enseñanza – aprendizaje de la programación.

Herramientas de apoyo para el aprendizaje de la programación

Existen muchos factores que influyen en el proceso de enseñanza-aprendizaje de la programación haciendo de este una tarea difícil, tanto en la parte de enseñanza para los profesores sobre la forma de como transmitir los conocimientos, así como en el aprendizaje de estos para el alumno. Por ello encontramos numerosas investigaciones en la búsqueda de alternativas para ayudar en el proceso de enseñanza-aprendizaje, las cuales se remontan prácticamente desde la creación o el inicio de la programación como tal, hasta la actualidad. A pesar ello, el proceso de enseñanza-aprendizaje de la programación en la actualidad aun representa un reto tanto para los profesores como para los alumnos.

En esta búsqueda de opciones que apoyen el proceso de enseñanza-aprendizaje de la programación en la parte correspondiente a los lenguajes y entornos de programación los investigadores han centrado esfuerzos en la creación de una gran variedad de ambos con la finalidad de que el aprendizaje de la programación sea más sencillo, como reportan (Kelleher & Pausch, 2005) en su investigación donde presentan una revisión de lenguajes y entornos de programación desarrollados con el fin de apoyar el aprendizaje de la programación, los autores proponen dos grupos para los lenguajes y entornos de programación, en el primero están los diseñados con el objetivo de ayudar a las personas a aprender a programar y en el segundo grupo se encuentran los que se diseñan basados en la idea de que la

importancia de la programación es poder ayudar a las personas a crear o construir cosas con base en sus necesidades (ver Anexo 1).

Desde los orígenes de la programación, debido a su naturaleza compleja, se han generado una gran cantidad de investigaciones enfocadas a apoyar la enseñanza – aprendizaje de la programación, (Becker & Quille, 2019) realizan una revisión de 50 años de trabajos realizados para la enseñanza de CS1 (cursos introductorios de programación) presentados en el simposio SIGCSE, obtienen 481 artículos y presentan una clasificación de los mismos de acuerdo a su enfoque en 8 categorías, 1) Primeros lenguajes y paradigmas, 2) CS1 Diseño, estructura y enfoque, 3) CS1 Contenido, 4) Herramientas, 5) Enfoque colaborativo, 6) Enseñanza, 7) Aprendizaje y evaluación y 8) Estudiantes, las cuales a su vez las dividen en subcategorías (ver Anexo 2).

Desde el enfoque educativo o de enseñanza se ha trabajado de igual forma en la búsqueda de alternativas (material instruccional, aplicaciones, objetos de aprendizaje, entre otros) que apoyen el proceso de enseñanza y faciliten el aprendizaje de los conceptos de programación. Estas opciones de apoyo son desarrolladas mediante la inclusión de nuevas tecnologías (realidad virtual, realidad aumentada, Moocs, entre otros), o basados en alguna teoría de aprendizaje como la teoría de la carga cognitiva.

Oberhauser & Lecon, (2017) enfocan su investigación en la utilización de la realidad virtual para la visualización, navegación y transmisión de información de estructuras de código de forma inmersiva e interactiva para apoyar a los procesos cognitivos, exploratorios, analíticos y descriptivos de código, mediante un prototipo llamado VR-based FlyThruCode (VR-FTC). En resumen, el objetivo de este prototipo es ayudar a los desarrolladores a tener una mejor visualización (de una forma inmersiva e interactiva) de estructuras de código beneficiando al mismo tiempo a los procesos de compresión.

Centeno Bragado, (2018) plantea la virtualización de un juego serio para la enseñanza de principios de programación que se basa en grupos de trabajo en donde se plantea primero un problema que requiere desarrollen un programa para su solución y escriban la respuesta en papel. La segunda parte es responder algunas preguntas de teoría sobre conceptos de programación con base en las respuestas se genera un código o clave que junto con la respuesta del primer problema forman la ubicación física en la escuela de donde tiene que ir por las instrucciones del siguiente nivel para poder continuar.

Kim et al., (2019) definen su proyecto como una “plataforma de aprendizaje de programación de realidad mixta”, basada en la combinación de realidad aumentada y realidad virtual (Mixed Reality) en donde el usuario debe escribir un programa que cree un camino para guiar a su avatar a una meta. Este programa se construye mediante

instrucciones o comandos dados en la interfaz de la plataforma y para ejecutar el código el usuario se involucra moviendo físicamente el avatar siguiendo las instrucciones que el mismo creó, al tiempo que el sistema sigue los movimientos en tiempo real generando retroalimentación visual de errores de código, ayudando a los usuarios a identificarlos más rápido y reducir la carga cognitiva.

En el tema del aprendizaje de la programación el constante desarrollo de ejercicios prácticos y una retroalimentación de manera oportuna por parte del profesor contribuyen en su proceso de aprendizaje, donde el proceso de retroalimentación se vuelve un tanto difícil con la cantidad de ejercicios multiplicado por la cantidad de alumnos, lo que ha dado paso a la creación de herramientas de calificación automática. (Guerrero et al., 2015) En su investigación presentan una revisión de herramientas de apoyo en el proceso de enseñanza-aprendizaje de la programación desde un enfoque extenso, clasificándolas en 4 grupos: calificación automática, multimedia, sistemas inteligentes de tutoría y de aprendizaje visual (ver Anexo 3).

Además de la inclusión de la tecnología ya sea mediante la utilización de esta o creación de nuevas herramientas para el apoyo del proceso de enseñanza - aprendizaje de la programación, se ha trabajado también dándole un enfoque “atractivo”, es decir que el estudiante se sienta motivado en utilizar estas herramientas y que esta motivación sirva como apoyo extra en las tareas del proceso de aprendizaje. El

mundo tecnológico actual en el que vivimos infiere en la cultura de los estudiantes, por lo cual son necesarias herramientas de aprendizaje que acoplen estas nuevas tecnologías en sus procesos de enseñanza - aprendizaje. Ejemplo de estos esfuerzos por desarrollar herramientas de este tipo son las aplicaciones de realidad virtual, realidad aumentada, realidad mixta y así como la gamificación.

Teorías de aprendizaje aplicadas al aprendizaje de la programación

Los modelos educativos han evolucionado, integrando al estudiante como participe de la generación de conocimiento y no solo ser el receptor del mismo, sino darle herramientas para que sea autodidacta (Alberto et al., 2019). Pero esto no es suficiente si el propio alumno no tiene el interés por aprender, los estudiantes deben ser atraídos y motivados en los procesos de enseñanza - aprendizaje, con lo cual se crea una relación de compromiso por parte del estudiante y la obtención de resultados se hace más satisfactorio para todos (profesores y estudiantes).

La evolución de los modelos educativos se ha dado desde la inclusión o modificación de los paradigmas o teorías de aprendizaje (conductismo, constructivismo, cognitivismo) y el surgimiento de nuevos paradigmas como el experimental y el conectivismo (Radianti et al., 2020), además de la inclusión de herramientas tecnológicas como complemento a estos modelos de aprendizaje.

De acuerdo con Shuell, (1986) en el conductismo el conocimiento es producto de la respuesta del estudiante ante estímulos de su entorno. El aprendizaje necesita repetición y la motivación es mayormente extrínseca.

El Constructivismo, Jean Piaget y Lev Vygotski principales exponentes, considera al aprendizaje como un activo donde los estudiantes generan (construyen) su representación o comprensión subjetiva de la realidad (Siemens, 2014). El constructivismo no hace distinción entre estudiantes con un conocimiento base y estudiantes novatos sin conocimiento previo de la disciplina, lo que hace que la construcción de conocimiento como sugiere el constructivismo en un entorno complejo como lo es el área de programación, genere ***una fuerte carga en la memoria de trabajo*** que es perjudicial para el aprendizaje afectando principalmente a estudiantes novatos, pero también a estudiantes con conocimiento previo de la disciplina (Kirschner et al., 2006).

El Conectivismo, asume que la gente procesa información a través de sus conexiones en el mundo digital y nunca para de aprender, buscando información fuera del área formal de la educación, compartiendo experiencias, aprendiendo a usar nuevas herramientas de tecnología, etc. (Siemens, 2014).

El Cognitivismo, expone que este tipo de aprendizaje se da con base en la adquisición de nuevo conocimiento mediante conocimiento adquirido

anteriormente donde los estudiantes crean o construyen a partir de estructuras ya existentes (Fosnot, 2013). La cognición humana o pensamientos existentes es el centro de estudio del cognitivismo, con base en ello el cognitivismo intenta mantener la atención del estudiante, haciendo sus actividades mentales menos complejas mediante la inclusión de material instruccional donde la información debe ser fácil de explicar y por lo tanto fácil de procesar por parte del estudiante, ejemplos de estos materiales instruccionales son las animaciones, explicaciones instructivas, ejemplos ilustrativos y demostraciones (Spyropoulou et al., 2013).

Ciencias cognitivas

El cognitivismo, también conocido como psicología cognitiva, es una corriente de teorías que se enfocan en los procesos mentales relacionados con el conocimiento y el aprendizaje, donde el estudiante cuenta con esquemas cognitivos ya existentes, que utiliza para asociarlos con nueva información, generando nuevos esquemas de información modificando los anteriores existentes (Cáceres & Munévar, 2017). En este contexto surgen conceptos como la metacognición, que se puede definir como el entender los procesos cognitivos (Argelio et al., 2018), es decir el conocimiento que tiene el estudiante sobre sus alcances y limitaciones cognitivas además de factores externos que puedan influir en el aprendizaje, así como el conocimiento de la disciplina a aprender (objetivos, complejidad y demás características que esta implique) (Bustingorry & Mora, 2008).

Las teorías cognitivas de aprendizaje señalan que la utilización o enfoque adecuados de los procesos cognitivos como lo son la memoria, pensamiento, abstracción, meditación y motivación optimizan el aprendizaje. Algunos ejemplos de estas teorías cognitivas son, la teoría de codificación dual, la cual afirma que la información que almacena una persona en la memoria de largo plazo es en forma dual, es decir en dos formas diferentes, visual y verbal, y esto hace que sea más fácil de recordar (Alomyan & Green, 2019). La teoría de esquemas, la cual afirma que la información puede ser almacenada en la memoria de largo plazo en conjuntos (esquemas) de conocimientos relacionados mediante sus principios de asimilación, organización y equilibrio (Zhiqing, 2015) y la Teoría de Carga Cognitiva.

Teoría de carga cognitiva

La teoría de carga cognitiva (J Sweller et al., 1998; John Sweller et al., 2019) es una teoría instruccional basada en principios de la biología evolutiva, que supone que el cerebro humano es un sistema de procesamiento de información natural. En su forma más simple, esta teoría identifica que el procesamiento de información en el cerebro humano pasa por dos subsistemas: la memoria de trabajo, que es muy limitada y la memoria de largo plazo que es ilimitada. Todo diseño instruccional tendría que ir orientado a disminuir la “carga cognitiva” impuesta a la memoria de trabajo. De tal suerte, a la fecha se han identificado 17 “efectos” que predicen resultados instruccionales para diferentes escenarios de aprendizaje (John Sweller et al., 2019). Estos

efectos suelen ser probados empíricamente, partiendo tanto de los principios de diseño de material instruccional indicados por el efecto, como de la medición (directa o indirecta) de la carga cognitiva impuesta al aprendiz. Si los resultados son positivos y el efecto instruccional implementado en el material facilita el aprendizaje, el efecto ha sido demostrado y puede ser utilizado para el diseño de futuro material instruccional (Fisch, 2017).

Del estudio de la teoría de carga cognitiva, la cual se basa en el proceso cognitivo humano, se han derivado varios efectos para el diseño instruccional los cuales han sido aplicados en numerosas investigaciones en diferentes áreas en la educación, en lo que respecta a la programación existen registros de la utilización de algunos de estos efectos para desarrollar recursos que apoyen el aprendizaje de la programación, lo cual abre un área de oportunidad para generar recursos que contribuyan al proceso de enseñanza - aprendizaje de la programación.

En contextos diferentes a la programación, Andersen et al., (2016) plantearon el uso de la Teoría de Carga Cognitiva basada en los principios de ejemplo resuelto y problema por completar, en una plataforma de simulación de realidad virtual de cirugía. Inicialmente en su hipótesis plantean que la carga cognitiva en los estudiantes de cirugía se puede reducir aplicando la teoria de carga cognitiva basada en principios (ejemplo resuelto y problema por completar) en el

ambiente de simulación de realidad virtual. Los resultados arrojaron que las instrucciones tradicionales en la plataforma de simulación de realidad virtual proveen menos carga cognitiva y mejor desempeño que la utilización de la teoria de carga cognitiva basada en los principicios mencionados.

Sands, (2019) utilizó principios basados en las ciencias cognitivas. Las técnicas planteadas fueron el ejemplo resuelto agregando preguntas para provocar la reflexion en el estudiante sobre problema, aplicando el “efecto de desvanecimiento”. Otra técnica utilizada fue la programación en parejas donde trabajan juntos para la resolución de problemas de forma alternada en las tareas. La tercer opción que se propuso fue el modelado con codificación en vivo, en donde el instructor presenta un problema y genera su modelo de solución en vivo frente a la clase el estudiante es testigo como se genera la solución desde el inicio hasta el final.

Shin, (2020) describió un experimento empirico enfocado al aprendizaje del lenguaje SQL, utilizando el concepto de instrucciones basadas en mapas como metodo instruccional, el cual a su vez esta basado en la teoría de la carga cognitiva, y lo comparó con el metodo de enseñanza convencional. La idea central del concepto de instrucciones basadas en mapas es que el estudiante describa como realizaría la sentencia de consulta (query) usando mapas conceptuales y luego proveer al estudiante el mapa conceptual generado por el instructor para que

verifique su solución. Según los resultados obtenidos el concepto de instrucciones basadas en mapas es superior al metodo convencional ya que reduce la carga cognitiva extraña y aumenta la carga cognitiva relacionada.

El efecto de auto-explicación

Derivado del estudio de la teoría de la carga cognitiva que comenzó en la década de 1980 como una teoría de diseño instruccional, como ya se ha mencionado, se han identificado 17 efectos, los cuales cuentan con resultados empíricos positivos. De éstos, el efecto del ejemplo resuelto y el problema por completar, son los prominentes y los más reportados. Sin embargo, el diseño de material basado en estos efectos también cuenta con limitaciones reportadas (Moreno, 2006, 2009). Por ejemplo, entre otros temas, se recomienda que deben tomarse en cuenta las características del estudiante y su motivación. Los autores de la Teoría de Carga Cognitiva señalan que la transferencia y el aprendizaje se optimizan al combinarse el efecto del ejemplo resuelto con otros efectos, tales como el de la auto-explicación. En este sentido, Redifer et al., (2016) definen el efecto de auto-explicación como el proceso de auto explicarse los pasos de resolución de un problema ya sea mientras se soluciona o se estudia un ejemplo resuelto, y además enfatizan que la auto-explicación es más efectiva si es estructurada. De tal suerte que los estudiantes que utilizan la auto-explicación pueden desarrollar conocimientos más completos, pero si se diseña la auto-explicación con resolución de problemas de forma simultánea puede aumentar la carga

cognitiva y dificultar el aprendizaje, sin embargo, si el estudiante cuenta con conocimientos previos solidos el efecto de auto-explicación produce mejores resultados.

CAPÍTULO III: ESTUDIOS RELACIONADOS

Se ha utilizado el efecto de auto-explicación en diversos estudios con resultados positivos en diferentes áreas temáticas, lo que abre una área de oportunidad para poder ser aplicado casi en cualquier área de estudio, tanto para el conocimiento conceptual como procedimental, donde hacer que los alumnos generen una explicación en lugar de presentarles es más eficaz, pero si después de que el alumno genera una explicación hace una segunda revisión en conjunto con la información nueva generada reforzará aún más el efecto (Bisra et al., 2018).

(Price et al., 2020) Diseñaron y desarrollaron un prototipo llamado “Instructor´s solution”, como apoyo para las tareas de programación en línea. El prototipo está basado en la comparación y explicación (auto-explicación), el cual presenta una solución para que el estudiante la compare con su solución, y además presenta una explicación de la solución proporcionada por el “Instructor´s solution” con espacios en blanco para que el estudiante utilice la auto-explicación para completarla. Se aplicó este prototipo en la universidad de Canadá en un curso introductorio de programación con los temas de ciclos y variables

a 109 estudiantes de manera aleatoria el “Instructor’s solution” en varias ocasiones a diferentes problemas, donde algunos recibían solo la parte de comparación, otros de auto explicación, otros ambos. Según los resultados obtenidos el estudiante puede aprender más con “explain prompts” pero esto se refleja en invertir más tiempo, lo que hace que muchos pierdan el interés.

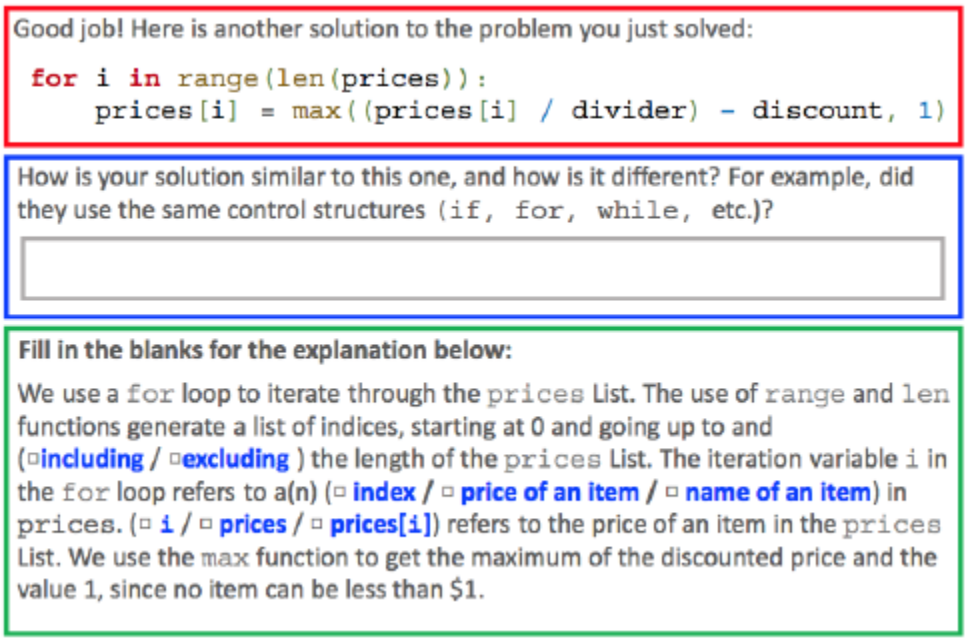


Ilustración 4. Interfaz Instructor’s solution, recuadro rojo solución Instructor´s solution, recuadro azul solución planteada por estudiante y recuadro verde explicación de la solución (explain prompts) (Price et al., 2020).

(Yen & Wang, 2017) Construyen un ambiente de programación, basando en la estrategia de auto-explicación y proporciona retroalimentación apropiada para programadores en C++. Metodología: Cuando el programador tiene errores de compilación, se le proporcionan

varios ejemplos y se le pide auto-explicación sobre la inferencia del error después de estudiar los ejemplos. Mediante un algoritmo desarrollado para este estudio la auto-explicación del programador es comparada con una base de datos, establecida y organizada en una ontología por programadores expertos y encuentra los posibles errores en la auto-explicación, basado en los resultados el ambiente logra retroalimentar el material de aprendizaje adecuado y ejemplos extendidos para que los programadores corrijan sus posibles conceptos erróneos.

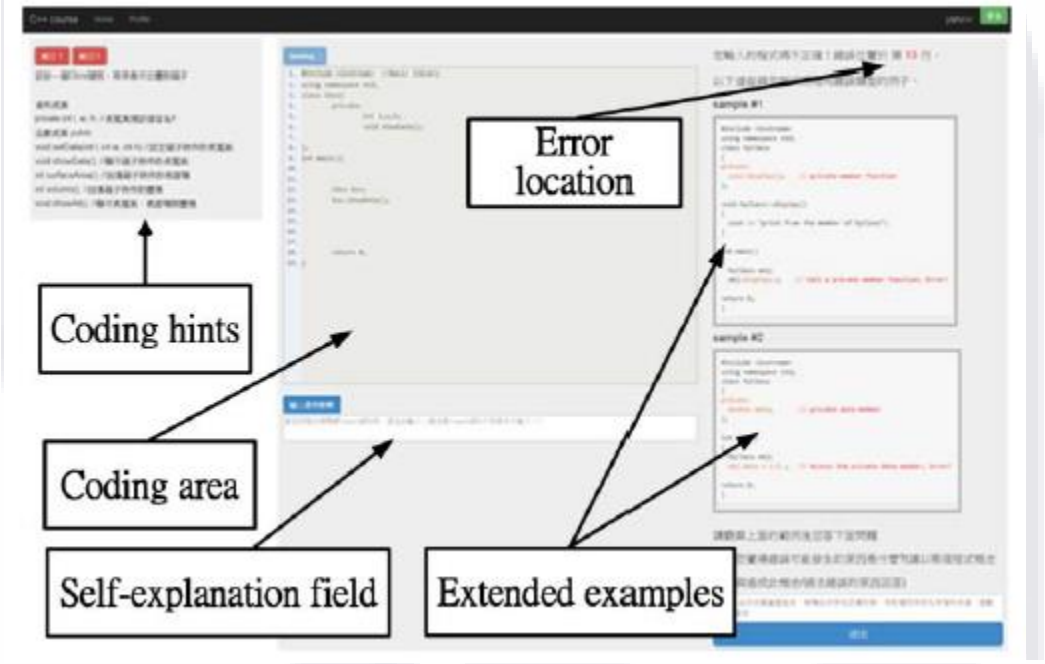


Ilustración 5.. Solicitud de auto explicación del error (Yen & Wang, 2017).

Ilustración 6. Visualización de material de aprendizaje para verificar conceptos erróneos (Yen & Wang, 2017).

(Aureliano et al., 2016) Presentan un método para el aprendizaje de la programación basado en el efecto de auto-explicación de la teoría de carga cognitiva, en el cual los estudiantes deben auto explicar pequeños bloques de código mientras están estudiando el proceso de construcción de programas por medio de videos utilizados como material instruccional, los resultados fueron comparados con otro grupo de estudiantes que solo visualizaron los videos sin el proceso de la auto-explicación. Los resultados obtenidos no indican una diferencia significativa en ambos grupos, atribuyen sus resultados al bajo nivel de dificultad del contenido seleccionado y las explicaciones presentadas en los videos.

(Garcia, 2017) Su estudio está basando en la inserción de comentarios de auto-explicación en el código de un conjunto de ejemplos resueltos dados. El experimento fue trabajado de dos formas diferentes, esto según las nuevas innovaciones referentes a como el usuario interactúa con la computación: *"the glass box" y "the black box".* El concepto de *black box* implica que el usuario utilice herramientas sin entender los mecanismos de estas, y el *glass box* el usuario puede ver y en ocasiones modificar estos mecanismos. Los resultados obtenidos fueron positivos en ambos casos, pero la mayoría de los comentarios de los estudiantes en el experimento de *black box* argumentan el deseo de tener mayor transparencia en los materiales de aprendizaje.

(Hiller et al., 2020) En su trabajo de investigación plantean que la utilización de la auto-explicación en los procesos de enseñanza – aprendizaje son importantes para que el estudiante genere relaciones entre los contenidos presentados y los ejemplos explicativos de estos contenidos, que ayuden en el aprendizaje. Enfocan su investigación en la forma de cómo aplicar la auto-explicación, esto es, ellos hacen referencia a dos formatos "closed-book" y "open-book". En el primer caso el estudiante no tiene acceso al material teórico solo se le presenta el ejemplo y se le pide auto-explicación mediante preguntas. Caso contrario la segunda opción donde se le presenta al estudiante el material teórico, ejemplo y se le pide la auto-explicación mediante preguntas. Los temas utilizados en el experimento fueron concernientes a la ionización. Los resultados que obtuvieron es que la forma de cómo

aplicar la auto-explicación influye y sugieren plantear la utilización de la auto-explicación con el formato de “open-book”.

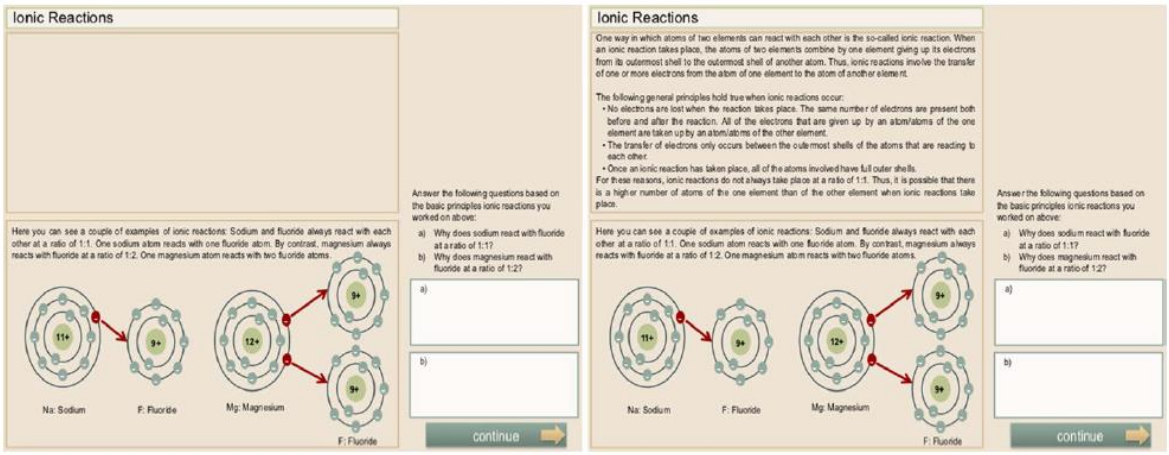


Ilustración 7. Ejemplo de auto-explicación con formato “closed-book” a la izquierda, y formato “open-book” a la derecha (Hiller et al., 2020).

(Heitzmann et al., 2018) *“Los profesores necesitan competencia para evaluar las situaciones en el aula y las necesidades de aprendizaje del alumno y para planificar, implementar y evaluar métodos pedagógicos”.* En recientes estudios diagnósticos de competencias a profesores muestran un déficit. En su investigación se desarrolló un experimento a futuros profesores mediante un entorno de aprendizaje asistido por computadora se implementaron ejemplos con errores añadiendo explicaciones con errores y retroalimentación adaptable con el objetivo facilitar el diagnostico de competencias de los futuros profesores. Los resultados obtenidos fueron negativos, plantean que cuando se utiliza la auto-explicación en contexto de ejemplos con errores afecta la eficacia de la auto-explicación.

(Bisra et al., 2018) Realizan un metaanálisis sobre una investigación realizada sobre los resultados de aprendizaje de participantes que utilizaron la auto-explicación mientras estudiaban o resolvían problemas. Los resultados obtenidos de esta investigación arrojan implicaciones importantes, la primera es, hacer que los alumnos generen auto-explicaciones suele ser más eficaz que presentarles una explicación. Segunda, los beneficios de la auto-explicación parecen ser aplicables a la mayoría de las áreas educativas. Tercera, el mayor beneficio de la auto-explicación puede surgir cuando el alumno hizo una explicación inicial y luego se les solicite la revisen cuando nueva información resalte lagunas o errores.

(Geela Venise Firmalo Fabic(&), Antonija Mitrovic, 2017) Mediante el uso de la herramienta Py Kinect, la cual es una herramienta móvil para el apoyo del aprendizaje del lenguaje Python, reportan los resultados del uso de esta herramienta enfocado a una de sus opciones “menu-based self-explanation prompts”. Usando problemas *“Parsons”,* son como problemas de rompecabezas donde hay que reordenar las líneas de código incluyen líneas de código incompletas, donde se les pidió a los estudiantes que explicaran los conceptos relacionados con las líneas de código incompletas, el estudio no da detalles de la metodología y el proceso empleados, los resultados que reportan afirman una gran diferencia en los resultados de evaluación aplicados a los estudiantes que utilizaron la herramienta comparados con estudiantes que no la utilizaron.

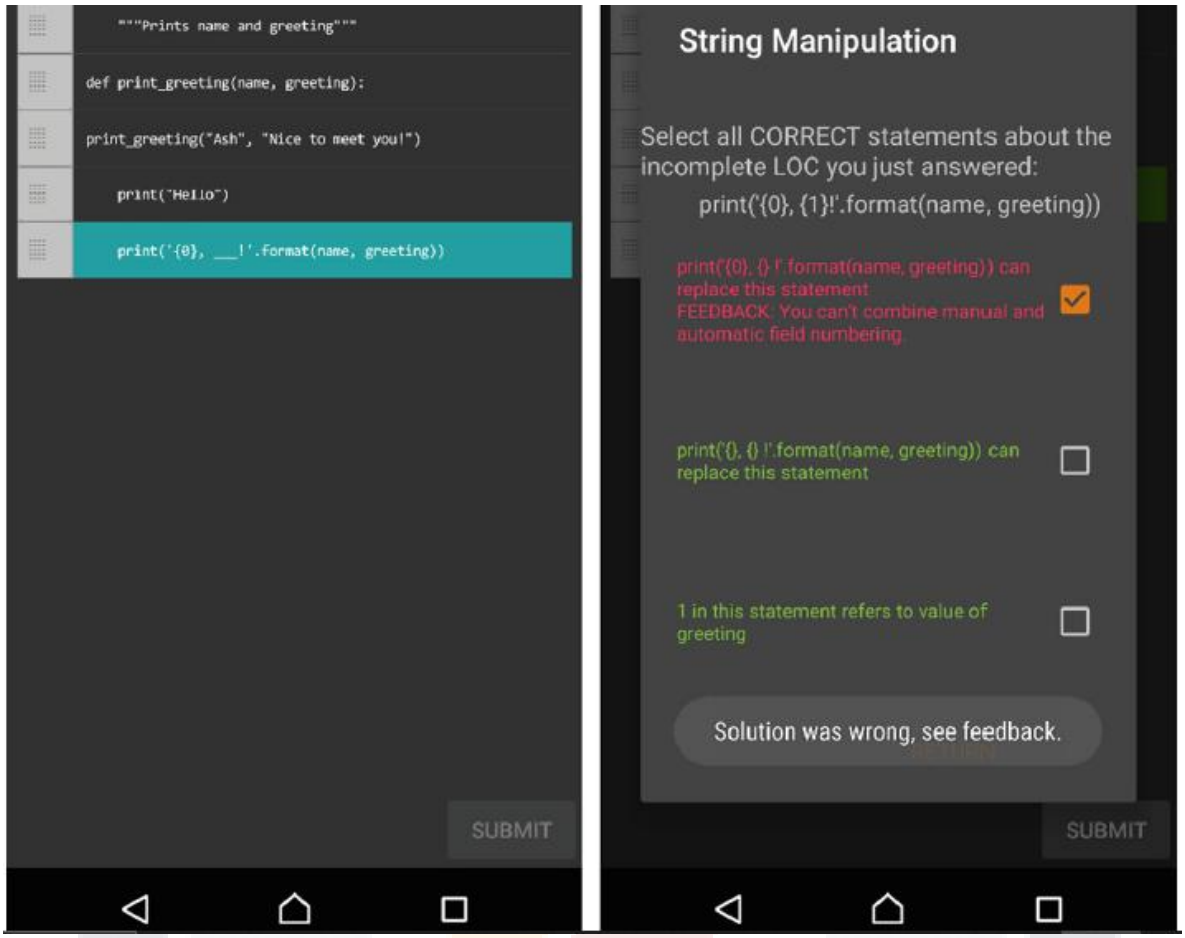


Ilustración 8. Problema "Parsons" con líneas incompletas de código a la izquierda, a la derecha SE prompt , en la herramienta Py Kinect (Geela Venise Firmalo Fabic(&), Antonija Mitrovic, 2017).

(Redifer et al., 2016) Examinan el impacto de la capacidad de la memoria de trabajo aplicando estrategia de auto-explicación en problemas de estadística. El experimento empleo 3 formas diferentes, el primer grupo se le presentaron problemas parcialmente resueltos pidiendo explicaran mientras completaban los problemas y después explicación a problemas completos. El segundo grupo, se les presento un problema resuelto luego se les proporciono un problema similar para resolver y explicar a la vez. El tercer grupo se les presentaron problemas a resolver sin auto – explicación. Y finalmente para la medición de la

capacidad de la memoria de trabajo utilizaron una tarea automatizada SymSpan (Symmetry Span). Según sus resultados opinan que la utilización de estrategias como la auto-explicación tendrán un mejor impacto en estudiantes con una capacidad de memoria de trabajo alta.

(Vihavainen et al., 2015) Utilizan el efecto de auto-explicación dividiendo el experimento en dos grupos, el primer grupo solo se aplica la auto-explicación, donde el estudiante tiene que explicar los detalles y causas de porque un bloque de código de un ejemplo ya programado no se obtienen los resultados esperados, el segundo grupo se le presenta el mismo ejemplo y de igual forma se le solicita al estudiante la auto-explicación pero con la diferencia que los estudiantes de este segundo grupo reciben preguntas de apoya adicionales. Según sus conclusiones la aplicación de la auto-explicación ya sea con o sin preguntas de ayuda, se traducen en beneficios, la más importante calificación más altas en los exámenes y mejores explicaciones en los mismos.

```
Scanner reader = new Scanner(System.in);

System.out.print("Visitor number: ");
int visitor = Integer.parseInt(reader.nextLine());

if (visitor >= 1000 && visitor % 25 == 0) {
    System.out.println("Receives a gift card!");
} else if (visitor % 2000 == 0) {
    System.out.println("Receives a large gift card!");
} else {
    System.out.println("Receives nothing.");
}
```

Your friend's program isn't working as he hopes. Explain to your friend the issues that are in his code and explain what causes them. Also, propose fixes for him.

[*Available only to the group with support questions*: In addition, what should the value of the variable *visitor* be for the current application to print "Receives a large gift card"? (a) 0, (b) 1000, (c) 1500, (d) 2000, (e) none of the given options.]

[text area for providing the explanation]

Ilustración 9. Ejemplo utilizado de auto – explicación con y sin preguntas de ayuda (Vihavainen et al., 2015).

CAPÍTULO IV: PROBLEMA DE INVESTIGACIÓN

La incorporación de dispositivos tecnológicos en la mayoría de nuestras actividades diarias, desde el uso de computadoras y otros dispositivos electrónicos en nuestros trabajos, así como dispositivos móviles para la comunicación y entretenimiento, hasta la inclusión de dispositivos inteligentes para el hogar, en los cuales una de las piezas más importantes para su funcionamiento es el software que se les implementa, y en el del desarrollo de este software uno de los procesos más relevantes es la programación. Con este rápido crecimiento de la industria de software surge la necesidad de contar con profesionales para cubrir la demanda actual y futura en materia de desarrollo de software, específicamente en los procesos de programación.

Sin embargo, la formación de profesionales en el área de programación a lo largo de la historia ha representado un reto, debido a que la mayoría de los estudiantes de las carreras afines a las ciencias computacionales presentan dificultades en el aprendizaje de la programación, lo que provoca índices de reprobación y abandono significativos. Lo anterior se ha traducido en una búsqueda constante de soluciones en materia educativa al problema o dificultad que representa el aprendizaje de la programación para la mayoría de los estudiantes.

El desarrollo de herramientas de apoyo y el diseño de material instruccional para el aprendizaje de la programación, usualmente no siguen una teoría de aprendizaje bien definida, ya que la mayoría de

estas se centra en la utilización de las nuevas tecnologías para su desarrollo. La falta de un enfoque hacia una teoría de aprendizaje en específico a la hora de desarrollar material instruccional, cualquier herramienta u objetos de apoyo en el proceso de enseñanza-aprendizaje puede hacer que se pierdan los objetivos y obtener resultados subóptimos.

Una de las teorías enfocadas y utilizadas en el diseño de material instruccional es la teoría de carga cognitiva, de la cual se derivan varios efectos instruccionales, los efectos de la teoría de carga cognitiva más conocidos, como lo son el efecto del ejemplo resuelto y el efecto del problema por completar, han sido estudiados y probados exhaustivamente obteniendo resultados positivos en su aplicación, pero se ha documentado que tienen limitaciones en su alcance de aplicación. De esta forma, el efecto de auto-explicación se ha indicado que es efectivo para reforzar estructuras de conocimiento existentes y su aplicación se recomienda para estudiantes que cuentan con un conocimiento base sobre los temas correspondientes.

Objetivo general

Diseñar, desarrollar, probar y medir la efectividad de material instruccional para el aprendizaje de la programación de estudiantes universitarios de ciencias computacionales, utilizando elementos de diseño indicados por el efecto de auto-explicación de la teoría de carga cognitiva.

Objetivos específicos

- Identificar los conceptos de programación a enseñar (objetivos de aprendizaje).
- Diseñar material didáctico para el aprendizaje de la programación basado en los lineamientos instruccionales recomendados por el efecto de auto-explicación de la Teoría de Carga Cognitiva.
- Identificar instrumentos de medición de carga cognitiva apropiados para el estudio del efecto de auto-explicación.
- Realizar estudios piloto para refinar el material instruccional diseñado y depurar el diseño experimental.
- Medir la efectividad del material instruccional con enfoque en los objetivos de aprendizaje.
- Medir la carga cognitiva de los estudiantes del grupo de control que se aplicó el material instruccional, así como de los estudiantes que se aplicó material instruccional tradicional.

Preguntas de investigación

¿Cuál es la efectividad del uso del material instruccional en términos de objetivos de aprendizaje en comparación con la metodología de enseñanza tradicional?

¿Qué ventajas u obstáculos prácticos representa el diseño y la producción de material instruccional basado en la teoría de carga cognitiva en general y el efecto de auto-explicación en particular?

¿Existen diferencias en la carga cognitiva de los estudiantes al aplicar modelos de aprendizaje tradicionales y aquellos basados en la teoría de carga cognitiva?

Justificación

La programación involucra un gran número de actividades cognitivas o de aprendizaje, como son el análisis de problema, planteamiento de posibles soluciones, diseño, codificación, pruebas y depuración hasta lograr la optimización de esta. Todo esto requiere pensamiento lógico algorítmico, así como el conocimiento de una gran cantidad de reglas de sintaxis, lo cual puede ocasionar frustración y falta de motivación en los estudiantes. Derivado de esto provoca que los estudiantes aun cuando terminan los cursos introductorios (conceptos básicos), sus habilidades en programación siguen siendo muy deficientes (Singh, 2017).

Existen varios factores que afectan de manera negativa el aprendizaje de la programación como lo son la complejidad de la sintaxis del lenguaje y sus conceptos, la carga cognitiva que implica el proceso de aprendizaje, el mal diseño del material instruccional y la falta de habilidades propias de los estudiantes para la resolución de problemas, todo esto se ve reflejado en altas tasas de deserción en el primer año de la carrera. La enseñanza – aprendizaje de conceptos de programación es un factor clave en la formación de profesionales de las ciencias de la computación, por tal motivo es necesario la búsqueda de

soluciones para disminuir la complejidad que este proceso de enseñanza – aprendizaje implica (Insuasti, 2016).

A pesar de que se han identificado muchos de los factores que afectan negativamente el aprendizaje de la programación y se han realizado varios estudios mediante la creación de herramientas de apoyo, material instruccional, enfoques educativos, todo esto para el apoyo del aprendizaje de la programación; la producción de material instruccional para abordar esta problemática no sido lo suficientemente amplia y efectiva, además de que existe una marcada tendencia en un enfoque instruccional tradicional donde el enfoque es a través de ejemplos y ejercicios subsecuentes (Insuasti, 2016).

De acuerdo con lo anterior, se pretende trabajar en el desarrollo de material instruccional para la enseñanza de conceptos de programación enfocada a estudiantes de carreras afines a las ciencias computacionales. Basando el desarrollo en una teoría de aprendizaje definida (teoría de la carga cognitiva) teniendo así una metodología de aprendizaje más concreta, que les permita a los estudiantes obtener las competencias necesarias de los conceptos de programación. El enfoque de la evaluación del material instruccional es a los objetivos de aprendizaje en comparación con los métodos tradicionales de enseñanza – aprendizaje de la programación.

Con el desarrollo este material instruccional se pretende contribuir con material pedagógico basado en la teoría de carga cognitiva mediante el efecto instruccional de auto-explicación con resultados verificados empíricamente, en busca de mejorar el problema académico y social de reprobación en las materias de programación de las carreras de sistemas. Además, proporcionar evidencia de la validez de los efectos de la teoría de carga cognitiva utilizados. Y finalmente, aportar pautas de diseño instruccional para la creación de material didáctico basado en el efecto de auto-explicación, que ayuden a los profesores de cursos de programación en su proceso de enseñanza.

CAPÍTULO V: MARCO TEÓRICO

Aspectos básicos importantes de la cognición humana son críticos para el diseño de material instruccional. En primer lugar, la teoría de carga cognitiva indica que la mayoría de los temas que se imparten en las instituciones educativas el ser humano no ha evolucionado para aprender de una forma natural. Segundo, estos temas requieren que el alumno adquiera dominio especifico en lugar de habilidades cognitivas específicas. Tercero, el conocimiento cognitivo genérico no requiere instrucción explicita porque lo adquirimos de forma natural, en cambio el dominio especifico de conceptos o habilidades del contenido de los programas educativos requieren instrucción explicita. La teoría de carga cognitiva es una de las pocas teorías que ha generado una gran cantidad de novedosos métodos instruccionales a partir del conocimiento de la cognición humana, ha sido citada entre 10,000 y 20,000 ocasiones (John Sweller, 2016).

Teoría de la carga cognitiva

La teoría de carga cognitiva es una teoría enfocada al diseño de material instruccional alineando sus procesos de diseño con la arquitectura cognitiva humana (Andrade-Lotero, 2012), donde la teoría de carga cognitiva utiliza la información de la cognición humana para el desarrollo de experimentos empíricos donde se han ido identificando efectos instruccionales para reducir la carga en la memoria de trabajo con el fin de facilitar la transferencia de conocimientos a la memoria de largo plazo (John Sweller, 2016), debido a que la capacidad de la memoria

trabajo es limitada, si los materiales instruccionales sobrecargan sus recursos, el aprendizaje será incierto (Andrade-Lotero, 2012).

Comprender la forma en como nuestro cerebro realiza el proceso cognitivo es uno de los principales retos para explicar y mejorar el aprendizaje, con base en ello la teoría de carga cognitiva indica que el procesamiento de la información en el cerebro pasa por tres estructuras, primero la memoria sensorial que son los sistemas visual y auditivo por los cuales recibimos la información mediante estímulos visuales o sonoros en donde su tiempo de duración es de 1 a 3 segundos, segundo la memoria de trabajo la cual permite procesar la información por periodos de 15 a 30 segundos, donde su capacidad de procesamiento es limitada en cuanto cantidad de información a procesar, pero esta limitante aplica solo a información nueva que no está vinculada con esquemas almacenados en la memoria de largo plazo, y la tercera es la memoria de largo plazo, la cual es ilimitada, donde la información que pasa a esta, **es organizada en esquemas**, los cuales pueden durar toda la vida y llegar a no consumir recursos cognitivos mediante la práctica y automatización (Andrade-Lotero, 2012).

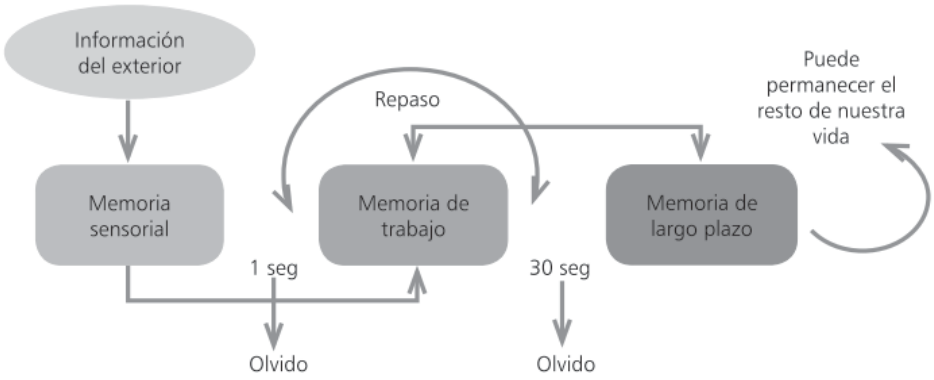


Ilustración 10. Estructura arquitectura cognitiva humana (Andrade-Lotero, 2012).

Esquemas

Un esquema es una agrupación o clasificación de diferentes elementos de información en una sola unidad, los cuales son almacenados en la memoria de largo plazo (J. Sweller et al., 2011), esta agrupación facilita a la memoria de trabajo su utilización cuando la información es requerida(Andrade-Lotero, 2012), pero aún con la agrupación de esquemas de información almacenados en la memoria de largo plazo, cuando estos esquemas son recientes o nuevos, la memoria de trabajo aún necesita un esfuerzo consciente y destinar recursos cognitivos para su utilización (J. Sweller et al., 2011).

La constante practica o utilización de un esquema de información nuevo, conforme avanza su utilización el esfuerzo consciente y los recursos cognitivos que la memoria de trabajo utiliza para su aplicación disminuyen, hasta llegar a un punto que esos esquemas se procesen de forma inconsciente y automática, mediante la automatización, es

aquí donde el estudiante mediante la extensa practica desarrolla el dominio especifico de este conocimiento, obteniendo habilidades y competencias (J. Sweller et al., 2011).

Tipos de carga cognitiva

La teoría de la carga cognitiva percibe tres tipos de carga que afectan los procesos de aprendizaje, en primer lugar, la carga cognitiva intrínseca la cual está asociada con la complejidad de la información o conocimientos por aprender, en segundo lugar, la carga cognitiva extraña la cual está relacionada con la forma como se presenta la información o conocimiento al estudiante dada por el método instruccional, la cual es modificable, y por último, la carga cognitiva relevante es la carga necesaria para aprender, esto es los recursos que dedica la memoria de trabajo en el proceso de aprendizaje (John Sweller et al., 2019).

Así pues, la carga cognitiva intrínseca hace referencia a la carga generada por naturaleza de la información a aprender (la dificultad de los conceptos o tareas por aprender) y los conocimientos base que el estudiante tiene sobre los temas por aprender (novatos, conocimientos básicos o expertos), basado en ello un concepto o tarea para un novato puede representar una alta complejidad, elevando su carga cognitiva intrínseca, mientras que para un experto no lo es, disminuyendo dicha carga (Andrade-Lotero, 2012). Derivado de lo anterior, la carga cognitiva intrínseca no puede ser modificada, puesto que es una carga esencial

y permanente en la naturaleza de la nueva información o tareas por aprender (John Sweller et al., 2019).

Referente a la carga cognitiva extraña, es la carga generada por el material instruccional en lo que respecta a la manera como se presenta la información al estudiante y la interacción de este con dicho material instruccional, el cual si contiene elementos que no aportan en la construcción del conocimiento hacen que se pierda el objetivo de aprendizaje sobrecargando la limitada memoria de trabajo con información poco o nada relevante perjudicando el aprendizaje (Andrade-Lotero, 2012). *"El material instruccional efectivo debe reducir estos elementos innecesarios"* (John Sweller et al., 2019).

En todo proceso de aprendizaje es necesario que el estudiante realice un esfuerzo mental o cognitivo, el cual tiene lugar en la memoria de trabajo, la cual como se vio anteriormente su capacidad es limitada, dicho esfuerzo cognitivo es la carga cognitiva relevante, basado en lo anterior es la carga cognitiva responsable de procesar la carga cognitiva intrínseca y la carga cognitiva extraña para colaborar al aprendizaje (John Sweller et al., 2019). De tal forma, a pesar de que la carga cognitiva relevante aporta a la suma total de la carga cognitiva, es necesaria para lograr la construcción de conocimiento (Andrade-Lotero, 2012).

Con base en lo anterior, la carga cognitiva generada en los procesos de aprendizaje, es impuesta en primer lugar por el material instruccional, en donde tenemos la carga cognitiva intrínseca y la carga cognitiva extraña, y en segundo lugar por los recursos cognitivos necesarios (carga cognitiva relevante) para que la memoria de trabajo atienda la carga cognitiva intrínseca y la carga cognitiva extraña (este tipo de carga quita recursos a la carga cognitiva relevante, dificultando el aprendizaje) (J. Sweller et al., 2011)(p. 264).

De acuerdo con la teoría de la carga cognitiva la forma en que la información es transmitida por medio del material instruccional y como este material instruccional interactúa con el proceso cognitivo del estudiante son parte fundamental para el aprendizaje, por ello, el enfoque de la teoría de la carga cognitiva es hacia el diseño y desarrollo de material instruccional que ayude a reducir la carga cognitiva extraña generada por el mismo material instruccional, ayudando así a la memoria de trabajo permitiendo enfocar más recursos a la carga cognitiva intrínseca favoreciendo el aprendizaje (Andrade-Lotero, 2012).

Efectos instruccionales

La teoría de la carga cognitiva es una de las pocas que han generado una gran cantidad de diseños instruccionales a partir de la arquitectura cognitiva humana, mediante los efectos instruccionales que se han ido identificando a partir de ella (John Sweller, 2016).

Tabla 4. Efectos de la teoría de carga cognitiva, elaborada a partir de (John Sweller et al., 2019) *y* (Andrade-Lotero, 2012)*.*

Solución libre (Goal-free)	Cuando se plantea el problema de tal forma que éste no tenga una única solución, liberando así la carga cognitiva.
Problema resuelto (Worked Example)	Provee al estudiante un ejemplo de un problema con una solución para ser estudiado.
Problema por completar (Completion Problem)	Provee al estudiante un ejemplo de un problema parcialmente resuelto que debe completar.
Atención divida (Split Attention)	Integrar múltiples fuentes de información, como imágenes y texto, en una sola para no dividir la atención y reducir la carga cognitiva.
Modalidad (Modality)	Utilización de los canales sensoriales (visión y audición) para integrar fuentes de información, reemplazando el texto por descripciones habladas. Se presupone que el procesamiento de audio se hace por un canal separado.
Auto-explicación (Self-Explanation)	Utilizar el problema resuelto o problema por completar, enriqueciéndolos, solicitando al alumno auto-explicación de la información (reforzamiento), para alumnos con conocimientos previos.
Redundancia (Redundancy)	Múltiples fuentes de información que tienen sentido en sí mismas, deben ser reemplazadas por una sola.
Interactividad (Interactivity)	La complejidad del material, que tiene muchos elementos interactuando entre sí, puede sobrecargar la memoria de trabajo dificultando el aprendizaje.
Variabilidad (Variability)	Reemplazar tareas con características similares, por tareas que difieran entre sí.

Imaginación (Imagination)	El alumno repase mentalmente los conceptos o procedimientos aprendidos sin acceso al material, para alumnos con conocimientos previos sólidos.
Elementos Aislados (Isolated Elements)	Presentar la información de manera secuencial en una forma aislada en lugar de una forma completamente interactiva (toda a la vez).
Reversión Experiencia (Expertise Reversal)	Los efectos de carga cognitiva que se presentan en alumnos novatos o con poca experiencia, no se encuentran o se invierten en alumnos con alta experiencia.
Desvanecimiento de Guía (Guidance-Fading)	Disminución del apoyo o guía conforme aumenta la experiencia del alumno.
Memoria Colectiva de Trabajo (Collective Working Memory)	Reemplazar tareas individuales por tareas colaborativas o en equipo, aumentando los recursos cognitivos.
Información Transitoria (Transient Information)	Los efectos de carga cognitiva que se encuentran para la información transitoria, generalmente no se encuentran para la información menos transitoria.
Movimiento Humano (Human Movement)	Reemplazar visualizaciones estáticas o poco realistas con visualizaciones que muestren movimientos humanos.
Autoadministración (Self-Management)	Los efectos de la carga cognitiva que se encuentra en materiales de instrucción mal diseñados se disminuyen cuando el alumno se le enseña explícitamente como reducir esa carga extraña.

Medición de la carga cognitiva

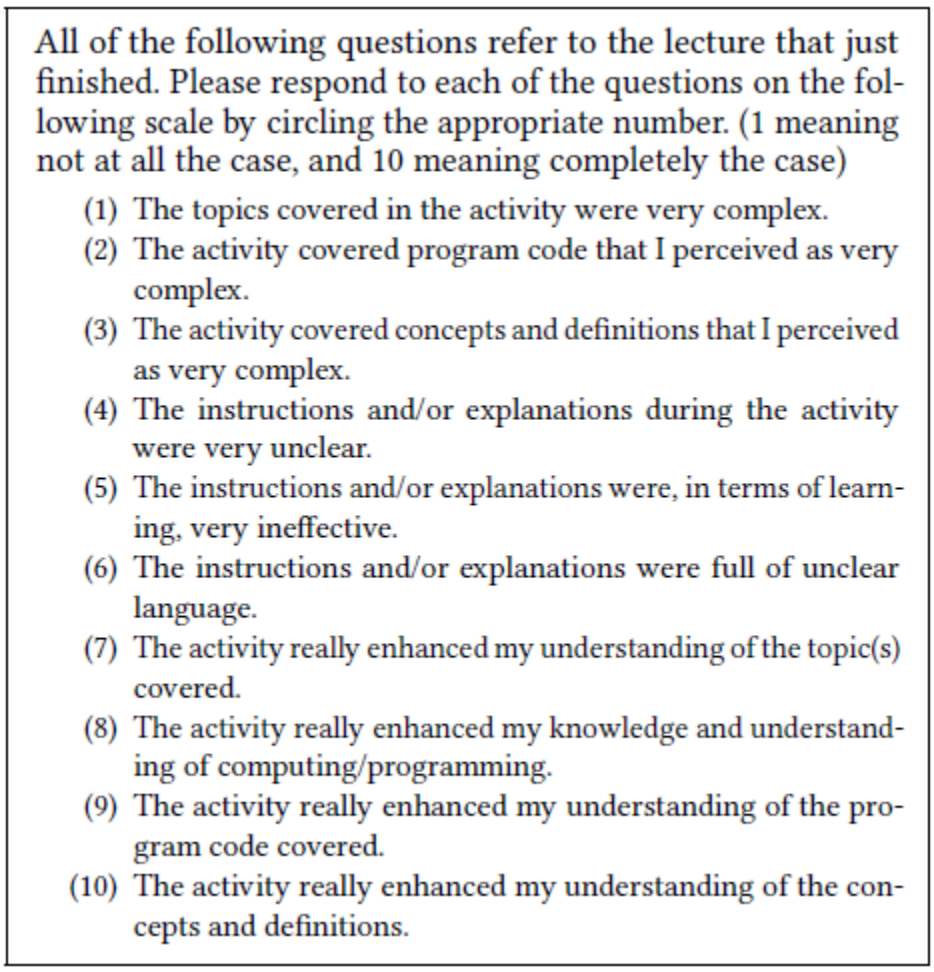
La carga cognitiva describe la cantidad de esfuerzo que realiza la memoria de trabajo cuando interactúa en algún ambiente de aprendizaje o en la resolución de problemas. Esta carga se puede dividir en: la impuesta por la tarea propia (intrínseca), la carga de entender y procesar el contenido (relacionada) y la forma en que la información es presentada (extraña) (Zagermann et al., 2018).

Existen medidas estandarizadas para medir la carga cognitiva como lo son los cuestionarios post-hoc como los NASA TLX, los cuales se basan en valoraciones subjetivas de los participantes, pero no evalúan los procesos inconscientes. Muchos estudios se han enfocado en la relación del movimiento de los ojos con el proceso cognitivo, los resultados revelan una relación del aumento de la carga cognitiva con un incremento en la dilatación de la pupila, mayor frecuencia de fijaciones, así como menor frecuencia de parpadeos (Zagermann et al., 2018).

Los métodos para la medición de la carga cognitiva comúnmente se agrupan en subjetivos y objetivos. Las medidas objetivas de la carga cognitiva no requieren que el estudiante reflexione y evalúe su esfuerzo mental o aprendizaje, muchos de estos métodos se basan en datos fisiológicos como el seguimiento ocular, pupilometría, frecuencia cardiaca y actividad neuronal, se pueden aplicar en el momento del aprendizaje, la desventaja que muchos son intrusivos y distraen la

atención. Los métodos subjetivos para la medición de la carga cognitiva solicitan al estudiante reporte el esfuerzo mental percibido y otros aspectos en el proceso de aprendizaje, algunos ejemplos de estos métodos son: Paas´s escala unidimensional de nueve puntos (unidimensional nine-point scale) para el esfuerzo mental de 1992, en 2013 Leppink desarrolla el cuestionario de autoevaluación de carga cognitiva (CL self-assessment questionnaire), entre otros (Zavgorodniaia et al., 2020).

En educación en el área de la programación se han realizado varios estudios para medir la carga cognitiva, se han utilizado tanto métodos objetivos (como el uso de electroencefalograma durante la compresión de programas, registro del seguimiento ocular y medición de la respuesta galvánica de la piel y el parpadeo de los ojos), como métodos subjetivos tales como instrumentos evaluadores de esfuerzo mental en forma de cuestionarios adaptados a la enseñanza de la programación. Por ejemplo, Zavgorodniaia et al., (2020) usaron el cuestionario previamente adaptado por (Morrison et al., 2014) de (Leppink et al., 2013), donde según los resultados obtenidos afirman que la utilización del cuestionario para la medición de carga cognitiva para programación es confiable.



All of the following questions refer to the lecture that just finished. Please respond to each of the questions on the following scale by circling the appropriate number. (1 meaning not at all the case, and 10 meaning completely the case)

(1) The topics covered in the activity were very complex.
(2) The activity covered program code that I perceived as very complex.
(3) The activity covered concepts and definitions that I perceived as very complex.
(4) The instructions and/or explanations during the activity were very unclear.
(5) The instructions and/or explanations were, in terms of learning, very ineffective.
(6) The instructions and/or explanations were full of unclear language.
(7) The activity really enhanced my understanding of the topic(s) covered.
(8) The activity really enhanced my knowledge and understanding of computing/programming.
(9) The activity really enhanced my understanding of the program code covered.
(10) The activity really enhanced my understanding of the concepts and definitions.

Ilustración 11. Cuestionario medición de carga cognitiva para programación (The CL self-assessment) tomado de (Zavgorodniaia et al., 2020).

Efecto de auto-explicación

La utilización de ejemplos resueltos a manera de estudio o repaso de procedimientos o conceptos presentados en el proceso de enseñanza - aprendizaje es comúnmente la forma más empleada para intentar construir conocimiento en los estudiantes, pero si los estudiantes no estudian exhaustivamente y comprenden a fondo estos ejemplos no se

obtendrá el aprendizaje esperado (John Sweller et al., 2019). El efecto de auto-explicación ayuda en el estudio y comprensión de estos ejemplos, lo que genera que el estudiante pueda crear y automatizar esquemas cognitivos mediante esta práctica (J. Sweller et al., 2011).

Es un proceso mental en el cual el estudiante al estar estudiando un ejemplo resuelto debe establecer la(s) relación(es) de este con el conocimiento previo adquirido mediante la auto-explicación, reforzando los conocimientos previos ya existentes apalancando la creación de esquemas cognitivos en la memoria de largo plazo, para ello, la aplicación del efecto de auto-explicación se recomienda para estudiantes con conocimientos previos, debido a que estudiantes sin el conocimiento previo adecuado pueden generar explicaciones erróneas y crear conceptos cognitivos incorrectos (J. Sweller et al., 2011).

El efecto de auto-explicación es un efecto derivado de la teoría de carga cognitiva con el fin de aplicarlo como apoyo en los procesos de enseñanza-aprendizaje para potenciar los objetivos de aprendizaje. El efecto de auto-explicación se puede definir como una actividad de construcción de aprendizaje activo donde se involucra al estudiante en los contenidos permitiendo monitorear el grado de comprensión de los mismos (Vihavainen et al., 2015).

El efecto de auto-explicación es una actividad cognitiva constructiva, para comprender nueva información o aprender habilidades, mediante

la generación de una auto-explicación de los conceptos o procedimientos previos adquiridos durante el proceso de enseñanza a manera de reforzamiento o reafirmación de los mismos, en ocasiones es necesario una explicación del instructor cuando el alumno no es capaz de generar una correcta o completa explicación (Bisra et al., 2018).

Limitaciones del efecto de auto-explicación

A pesar de que el uso del efecto de auto-explicación se ha probado como una técnica de aprendizaje poderosa, tiene algunas limitaciones que hay que atender a la hora de implementarla en el diseño de material instruccional (Rittle-Johnson & Loehr, 2017).

- La primer limitación es que promueve la atención a tipos particulares de información, reduciendo los detalles, lo cual sugiere que la auto-explicación no se debe aplicar a información detallada.
- Segunda, la auto-explicación de los propios métodos o soluciones puede reducir el aprendizaje cuando estos métodos o soluciones no son correctos.
- Tercera, la auto-explicación dirige la atención a una información en particular, lo que puede reducir la atención en hacia otra información importante.
- Y cuarta, la auto-explicación promueve el aprendizaje, pero en algunas situaciones, técnicas instruccionales alternativas puede ser igual o más efectivas, como por ejemplo una explicación del instructor.

CAPÍTULO VI: METODOLOGÍA

Basado en lo anterior, en donde se ha planteado el contexto de la teoría de la carga en la cual se han identificado 17 efectos instruccionales, los cuales se han utilizado para el diseño de material instruccional experimental, en este caso de estudio orientado al aprendizaje de la programación, obteniendo resultados positivos, partiendo de ello, se plantea el diseño de material instruccional para el aprendizaje de la programación basado en el efecto de auto-explicación de la teoría de carga cognitiva.

Diseño del material

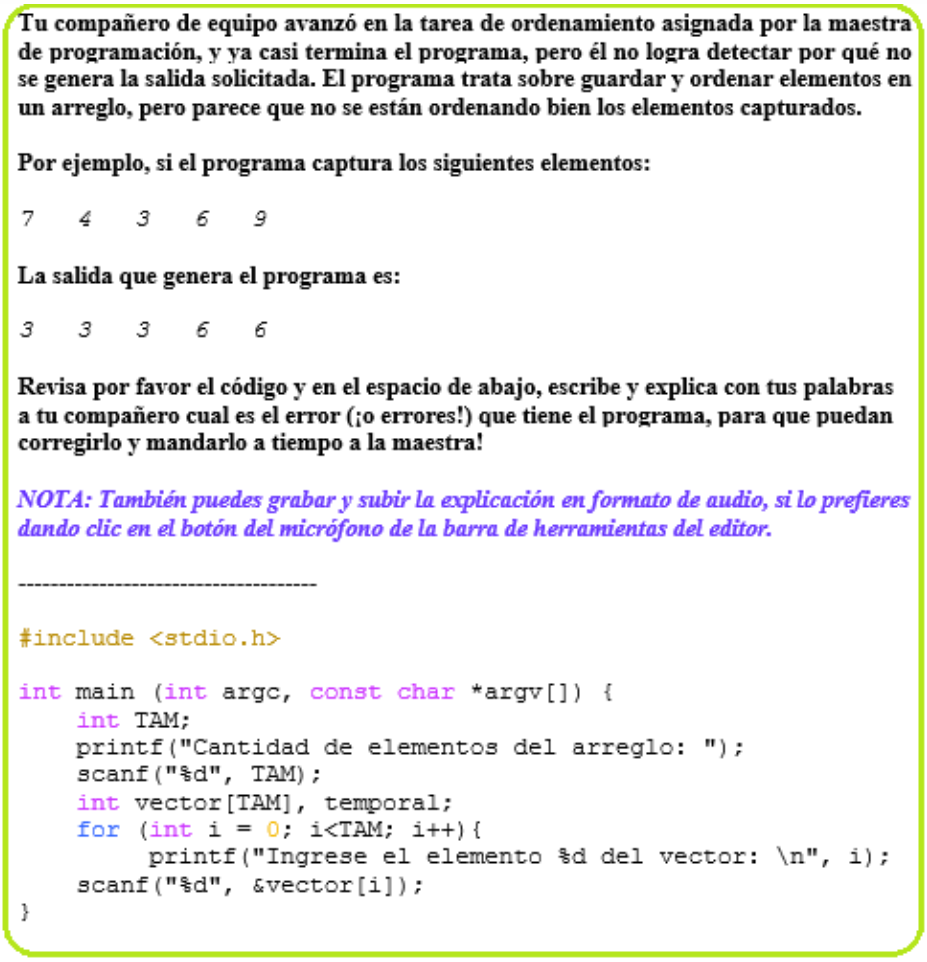
Revisando el programa de la materia “Programación I” del primer semestre de la carrera de Ing. En Sistemas Computacionales para la elección de los temas a incluir en el material instruccional, se opta por arreglos y vectores. De acuerdo con lo anterior, se diseñaron siete ejercicios similares (isomórficos) a los vistos en las clases, en formato de auto-explicación introduciendo errores en el código.

La estructura de diseño de los ejercicios está compuesta de varias partes:

- La *descripción del ejercicio*, en la cual se plantea una “situación real”, en donde el estudiante tiene que identificar el error o errores en el código de un programa (similar a los vistos en clase) “desarrollado por un compañero” el cual no arroja los resultados

deseados, explicar a su “compañero” el ¿por qué? de los errores y pueda corregir el programa para que arroje la salida correcta.

- La *salida del programa,* se muestra un ejemplo de lo que genera como resultado el programa del ejercicio en cuestión.
- *Código fuente,* se anexa el código del programa del ejercicio que el estudiante debe analizar y encontrar los errores.



Tu compañero de equipo avanzó en la tarea de ordenamiento asignada por la maestra de programación, y ya casi termina el programa, pero él no logra detectar por qué no se genera la salida solicitada. El programa trata sobre guardar y ordenar elementos en un arreglo, pero parece que no se están ordenando bien los elementos capturados.

Por ejemplo, si el programa captura los siguientes elementos:

7 4 3 6 9

La salida que genera el programa es:

3 3 3 6 6

Revisa por favor el código y en el espacio de abajo, escribe y explica con tus palabras a tu compañero cual es el error (¡o errores!) que tiene el programa, para que puedan corregirlo y mandarlo a tiempo a la maestra!

NOTA: También puedes grabar y subir la explicación en formato de audio, si lo prefieres dando clic en el botón del micrófono de la barra de herramientas del editor.

```
#include <stdio.h>

int main (int argc, const char *argv[]) {
    int TAM;
    printf("Cantidad de elementos del arreglo: ");
    scanf("%d", TAM);
    int vector[TAM], temporal;
    for (int i = 0; i<TAM; i++){
         printf("Ingrese el elemento %d del vector: \n", i);
    scanf("%d", &vector[i]);
}
```

```
printf("El contenido del arreglo es: \n");

for (int i = 0; i<TAM; i++){
      printf("%d \t", vector[i]);
    }

for (int i = 0; i< TAM; i++){
     for (int j = 0; j< TAM; j++){
          if (vector[j] > vector[j+1]){
              vector[j] = vector[j+1];
              temporal = vector[j];
              vector[j+1] = temporal;
          }
     }
}

printf("\n El contenido ordenado del arreglo es: \n");

for (int i = 0; i<TAM; i++){
     printf("%d \t", vector[i]);
}

return 0;

}
```

Ilustración 12. Ejemplo estructura de ejercicios en formato auto-explicación con errores de código.

Para medir la carga cognitiva se diseñó un cuestionario tomando como referencia el cuestionario adaptado por (Morrison et al., 2014) para la medición de la carga cognitiva en cursos introductorios de las ciencias de la computación, en este caso enfocado a la programación, tanto para el grupo de control como para el grupo experimental (ver Anexo 4). Se utilizó la aplicación de Forms de Office para la aplicación en línea de dicho cuestionario.

Cuestionario CL (Ejercicios auto-explicación)

Instrucciones: Las siguientes preguntas corresponden a la actividad con los ejercicios de auto-explicación que se realizó hace unos días, por favor responde cada pregunta seleccionando en una escala del 0 - 10 el número apropiado desde tu sentir de acuerdo a cada pregunta.

* Obligatorio

1. ID: *

Escribe tu respuesta

2. ¿Los temas cubiertos en la actividad fueron muy complejos?

0	1	2	3	4	5	6	7	8	9	10

Nada complejos — Muy complejos

3. ¿La actividad contenía código que percibí como muy complejo?

0	1	2	3	4	5	6	7	8	9	10

Nada complejo — Muy complejo

4. ¿La actividad contenía conceptos y definiciones que percibí muy complejos?

0	1	2	3	4	5	6	7	8	9	10

Nada complejos — Muy complejos

Ilustración 13. Ejemplo aplicación cuestionario para la medición de la carga cognitiva en Microsoft Forms.

Posterior a este primer estudio piloto, se realizó un segundo estudio en Abril 2021, afinando detalles encontrados en la aplicación del experimento aplicado en Noviembre 2020.

En este caso el diseño del material se desarrolló con base a la misma estructura de los ejercicios en formato de auto-explicación con errores en código, cuatro ejercicios en total, pero en esta ocasión basados en los temas de cadenas, y además se generó un segundo material para un tercer grupo experimental, donde a los ejercicios en formato de auto-explicación se añadieron videos explicativos (ejemplos resueltos) de ejemplos isomórficos a los ejercicios planteados, generando así material basado en el efecto de auto-explicación apoyado por el efecto del ejemplo resuelto utilizando un formato audiovisual.

Diseño experimental

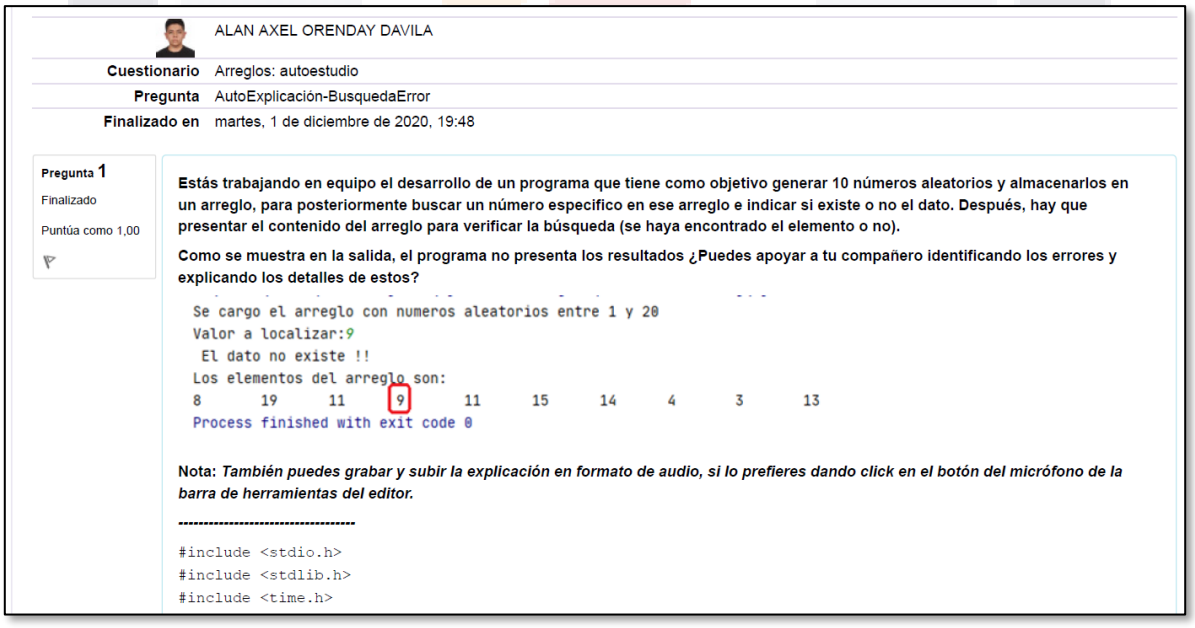
Estudio piloto

La aplicación de los ejercicios se llevó a cabo mediante un estudio cuasi experimental, tomando dos grupos con características similares de promedio y conocimientos previos, del primer semestre de la carrera de Ingeniería en Sistemas Computacionales, con el mismo instructor y por ende mismo material didáctico, teniendo así un grupo de control y un grupo experimental, y así medir la efectividad de los ejercicios aplicados comparando resultados de ambos grupos.

La asignación de los grupos para el estudio quedó de la siguiente manera, se designó como grupo experimental 1° B del turno vespertino con un total de 54 alumnos inscritos al inicio del curso, de los cuales

solo 39 realizaron los ejercicios planteados en el experimento, pero de estos se realizó un filtro descartando 4, quedando una población de 35, debido a que no presentaron exámenes parciales, los cuales eran necesarios para poder comparar los resultados y hacer el análisis pre y post experimento. Para el caso del grupo de control, se asignó a 1° A del turno matutino en el cual se tenía una población de 49 estudiantes, de los cuales 3 no presentaron el segundo parcial, reduciendo entonces cantidad de estudiantes para el estudio de este grupo a 46.

Para el análisis de los datos se tomó como variable dependiente las calificaciones obtenidas en los reactivos correspondientes a los temas de arreglos y vectores de los exámenes parciales 2º y 3º. En el caso de la variable independiente correspondió a los resultados obtenidos en los siete ejercicios en formato de auto-explicación planteados en el experimento.



ALAN AXEL ORENDAY DAVILA

Cuestionario Arreglos: autoestudio
Pregunta AutoExplicación-BusquedaError
Finalizado en martes, 1 de diciembre de 2020, 19:48

Pregunta 1
Finalizado
Puntúa como 1,00

Estás trabajando en equipo el desarrollo de un programa que tiene como objetivo generar 10 números aleatorios y almacenarlos en un arreglo, para posteriormente buscar un número especifico en ese arreglo e indicar si existe o no el dato. Después, hay que presentar el contenido del arreglo para verificar la búsqueda (se haya encontrado el elemento o no).

Como se muestra en la salida, el programa no presenta los resultados ¿Puedes apoyar a tu compañero identificando los errores y explicando los detalles de estos?

```
Se cargo el arreglo con numeros aleatorios entre 1 y 20
Valor a localizar:9
 El dato no existe !!
Los elementos del arreglo son:
8       19      11      9       11      15      14      4       3       13
Process finished with exit code 0
```

Nota: ***También puedes grabar y subir la explicación en formato de audio, si lo prefieres dando click en el botón del micrófono de la barra de herramientas del editor.***

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

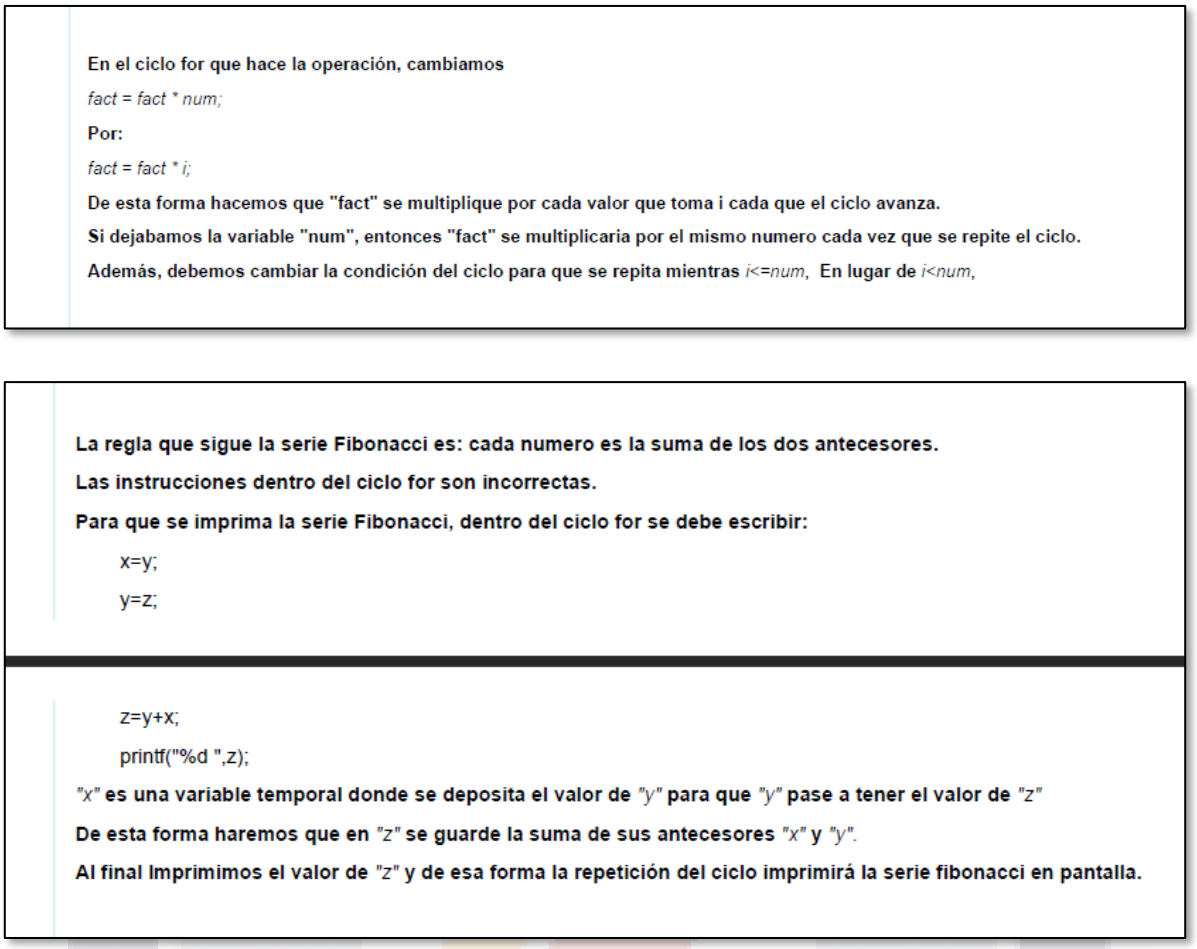
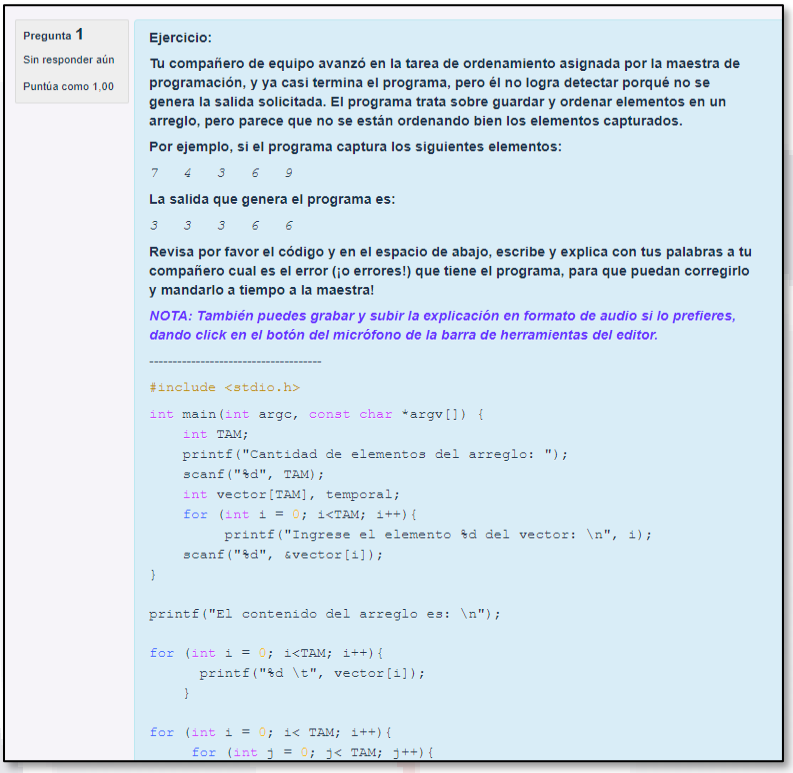


Ilustración 14. Ejemplo respuesta a uno de los ejercicios planteados en el experimento.

El periodo de aplicación de los ejercicios del primer estudio tuvo una duración de 2 semanas en el mes de Noviembre de 2020, durante este tiempo los estudiantes pudieron realizar los ejercicios propuestos en la plataforma de aula virtual. Al término de este periodo inmediatamente se dio acceso al formulario en línea diseñado para la medición de la carga cognitiva, en el caso del grupo de control que no realizaron los ejercicios de igual forma se les aplicó el cuestionario haciendo enfoque a los ejercicios realizados en clase con respecto a los temas de vectores y arreglos.

Se utilizó la plataforma de aula virtual de la Universidad Autónoma de Aguascalientes para la aplicación de los ejercicios. La implementación de los ejercicios en la plataforma permitió a los estudiantes que sus respuestas (explicaciones) pudieran ser escritas o en formato de audio, véase la ilustración 15.



Pregunta 1

Sin responder aún

Puntúa como 1,00

Ejercicio:

Tu compañero de equipo avanzó en la tarea de ordenamiento asignada por la maestra de programación, y ya casi termina el programa, pero él no logra detectar porqué no se genera la salida solicitada. El programa trata sobre guardar y ordenar elementos en un arreglo, pero parece que no se están ordenando bien los elementos capturados.

Por ejemplo, si el programa captura los siguientes elementos:

7 4 3 6 9

La salida que genera el programa es:

3 3 3 6 6

Revisa por favor el código y en el espacio de abajo, escribe y explica con tus palabras a tu compañero cual es el error (¡o errores!) que tiene el programa, para que puedan corregirlo y mandarlo a tiempo a la maestra!

NOTA: También puedes grabar y subir la explicación en formato de audio si lo prefieres, dando click en el botón del micrófono de la barra de herramientas del editor.

```
#include <stdio.h>
int main(int argc, const char *argv[]) {
    int TAM;
    printf("Cantidad de elementos del arreglo: ");
    scanf("%d", TAM);
    int vector[TAM], temporal;
    for (int i = 0; i<TAM; i++){
         printf("Ingrese el elemento %d del vector: \n", i);
    scanf("%d", &vector[i]);
}

printf("El contenido del arreglo es: \n");

for (int i = 0; i<TAM; i++){
      printf("%d \t", vector[i]);
    }

for (int i = 0; i< TAM; i++){
     for (int j = 0; j< TAM; j++){
```

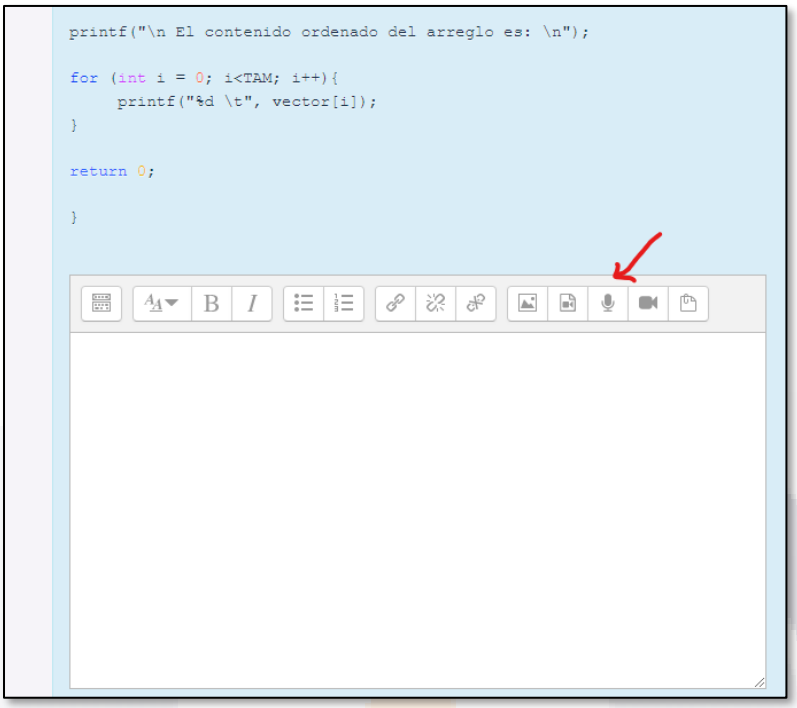


Ilustración 15. Ejemplo implementación ejercicios en plataforma de aula virtual.

El método de evaluación utilizado fueron exámenes estandarizados y calibrados por la academia de programación del Departamento de Sistemas Eléctricos, aplicados tanto al grupo de control como al grupo experimental, tomando para los datos correspondientes solo a los temas concernientes de arreglos y vectores del segundo (pre) y tercer parcial (post).

Segundo estudio

De igual forma se utilizó un estudio cuasi experimental, tomando en para este segundo estudio tres grupos con características similares de promedio y conocimientos previos, del segundo semestre de la carrera

de Ingeniería en Sistemas Computacionales, con el mismo material didáctico, teniendo así un grupo de control y dos grupos experimentales, y así medir la efectividad de los ejercicios aplicados comparando resultados de los tres grupos.

La asignación de los grupos para el estudio quedó de la siguiente manera, se designó como grupo de control 2°B con un total de 46 alumnos inscritos al inicio del curso, pero de estos se realizó un filtro descartando 1, quedando una población de 45, debido a que no presentaron exámenes parciales, los cuales eran necesarios para poder comparar los resultados y hacer el análisis pre y post experimento. Para el caso del grupo experimental 1, se asignó a 2°A en el cual se tenía una población de 51 estudiantes, de los cuales se descartó 1 puesto que no presentó uno de los exámenes parciales, reduciendo entonces cantidad de estudiantes para el estudio de este grupo a 50. Y, por último, para el grupo experimental 2 se asignó a 2°C en el cual se tenía una población de 50, de los cuales se descartó 1 puesto que no presentó uno de los exámenes parciales, reduciendo entonces cantidad de estudiantes para el estudio de este grupo a 49.

Para el análisis de los datos se tomó como variable dependiente las calificaciones obtenidas en los reactivos correspondientes a los temas de cadenas de los exámenes parciales 2º y 3º. En el caso de la variable independiente correspondió a los resultados obtenidos en los cuatro ejercicios en formato de auto-explicación planteados en el experimento.

El periodo de aplicación de los ejercicios del segundo estudio tuvo una duración de 1 semana en el mes de Abril de 2021, durante este tiempo los estudiantes pudieron realizar los ejercicios propuestos en la

plataforma de aula virtual. Al término de este periodo inmediatamente se dio acceso al formulario en línea diseñado para la medición de la carga cognitiva, en el caso del grupo de control que no realizaron los ejercicios de igual forma se les aplicó el cuestionario haciendo enfoque a los ejercicios realizados en clase con respecto a los temas de cadenas.

Se utilizó la plataforma de aula virtual de la Universidad Autónoma de Aguascalientes para la aplicación de los ejercicios. La implementación de los ejercicios en la plataforma permitió a los estudiantes que sus respuestas (explicaciones) pudieran ser escritas o en formato de audio, para el caso del segundo grupo experimental además de los ejercicios en formato de auto-explicación se les dio acceso a videos en formato de ejemplo resuelto.

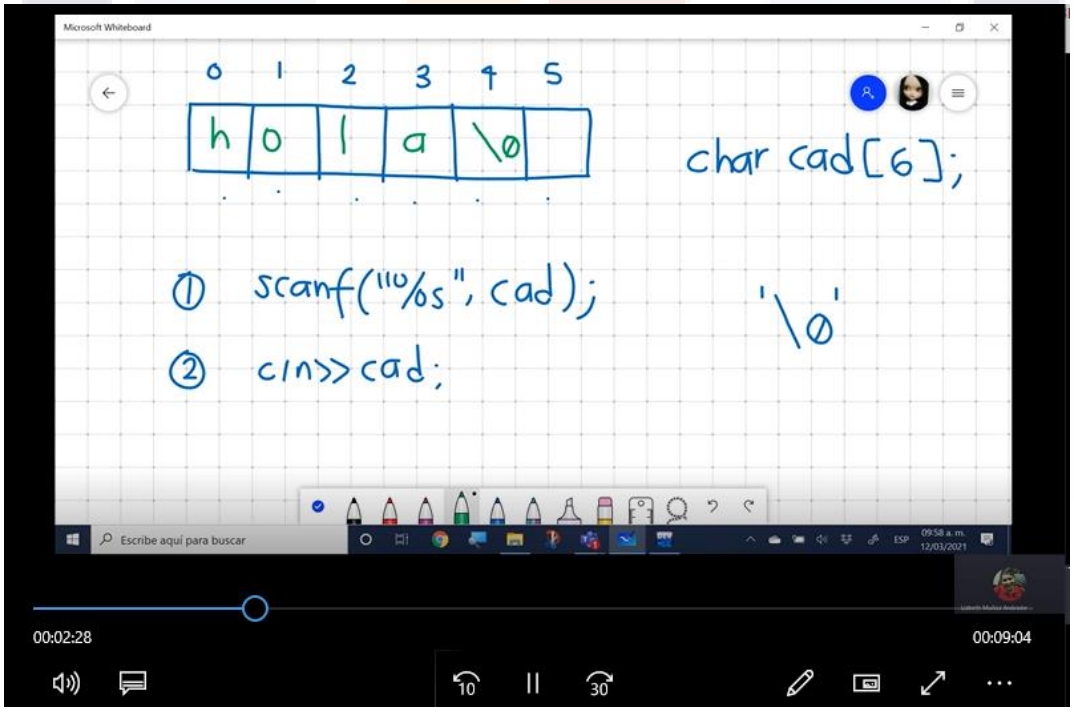


Ilustración 16. Captura de pantalla de uno de los videos en formato de ejemplo resuelto.

El método de evaluación utilizado fueron exámenes estandarizados y calibrados por la academia de programación del Departamento de Sistemas Eléctricos, aplicados tanto al grupo de control como al grupo experimental, tomando para los datos correspondientes solo a los temas concernientes de cadenas del segundo (pre) y tercer parcial (post).

CAPÍTULO VII: RESULTADOS

Estudio piloto

Los resultados recolectados correspondientes al estudio piloto realizado en Noviembre de 2020 arrojaron un pequeño incremento en el promedio de las calificaciones en la medición post con un promedio de 7.9 en comparación con la medición pre con 7.3, obteniendo 0.6 puntos de incremento en el promedio de calificaciones del grupo experimental.

Tabla 5. Estadística descriptiva grupo experimental.

Estadísticos

		PreExp	PostExp
N	Válidos	35	35
	Perdidos	12	12
Media		7.3314	7.9371
Desv. típ.		1.85088	1.20174
Varianza		3.426	1.444

Para confirmar si existe una diferencia estadística significativa se realiza la prueba t con las calificaciones PRE y POST, el resultado de la prueba t arroja un valor p de 0.025 lo que indica que, si existe una diferencia

estadística significativa entre las calificaciones PRE y POST, lo que significa que se obtuvo una ganancia en el aprendizaje, sugiriendo un efecto positivo debido al tratamiento por ejercicios de auto-explicación.

Tabla 6. Prueba t de muestras relacionadas PRE-POST grupo experimental.

Prueba de muestras relacionadas

		Diferencias relacionadas					t	gl	Sig. (bilateral)
		Media	Desviación típ.	Error típ. de la media	95% Intervalo de confianza para la diferencia				
					Inferior	Superior			
Par 1	PreExp - PostExp	-.60571	1.52256	.25736	-1.12873	-.08270	-2.354	34	.025

Observando gráficamente estos resultados se tienen los histogramas de frecuencias para las calificaciones PRE y POST, donde se puede observar un agrupamiento mayor de los datos entre las calificaciones 7 y 10 en las calificaciones POST respecto a las calificaciones PRE donde se observan más datos por debajo del 7.

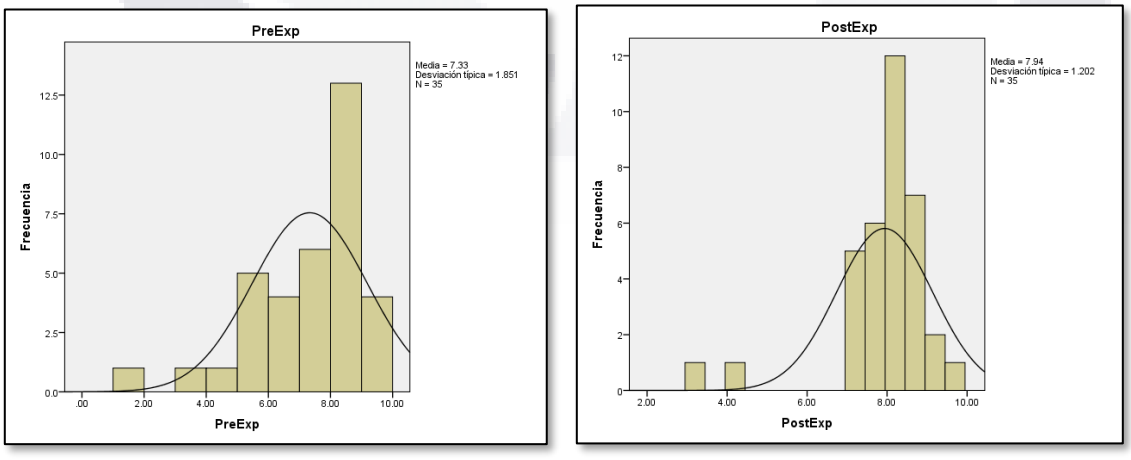


Ilustración 17. Comparación de histogramas de frecuencias PRE-POST experimental.

El grupo de control no realizaron los ejercicios de auto-explicación planteados, estudiaron de manera tradicional. Los resultados estadísticos muestran un promedio de 8.01 en las calificaciones PRE y 8.12 en las calificaciones POST. Donde tenemos una diferencia de 0.11 puntos en el promedio.

Tabla 7. Estadística descriptiva para grupo de control.

Estadísticos

		PreCtrl	PostCtrl
N	Válidos	49	49
	Perdidos	0	0
Media		8.0122	8.1284
Desv. típ.		2.15074	2.47100
Varianza		4.626	6.106

Para confirmar si existe una diferencia estadística significativa se realiza la prueba t con las calificaciones PRE y POST, el resultado de la prueba t arroja un valor p de 0.514 lo que indica que, no existe una diferencia estadística significativa entre las calificaciones PRE y POST, sugiriendo que el grupo mantuvo su promedio en ambos exámenes. Esto también se puede observar gráficamente en la distribución de los datos de ambas calificaciones en los histogramas de frecuencias a continuación.

Tabla 8. Resultados de prueba t, muestras relacionadas PRE-POST grupo de control.

Prueba de muestras relacionadas

		Diferencias relacionadas					t	gl	Sig. (bilateral)
		Media	Desviación típ.	Error típ. de la media	95% Intervalo de confianza para la diferencia				
					Inferior	Superior			
Par 1	PreCtrl - PostCtrl	-.11612	1.23556	.17651	-.47102	.23877	-.658	48	.514

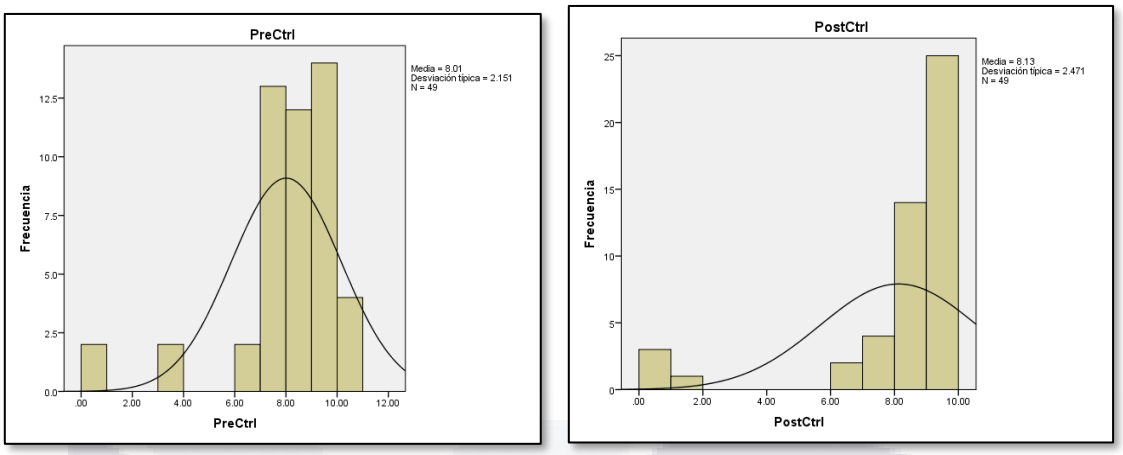


Ilustración 18. Comparación de histogramas de frecuencias PRE-POST grupo de control.

Resultados medición carga cognitiva estudio piloto

Como se mencionó anteriormente se utilizó el cuestionario NASA-TLX adaptado para programación para medir la carga cognitiva, este cuestionario consta de 10 preguntas con respuestas en una escala de 0-10. Las preguntas 1, 2 y 3 miden la carga cognitiva intrínseca (complejidad de lo que se quiere aprender), las preguntas 4, 5 y 6 corresponden a la carga cognitiva extrínseca (como se enseña), y las preguntas 7, 8, 9 y 10 la carga cognitiva pertinente (se refiere a la carga impuesta a la memoria de trabajo por el propio proceso de aprendizaje).

Los resultados arrojan promedios similares en lo que respecta a la carga cognitiva extrínseca y la pertinente, pero se puede apreciar una diferencia en la carga cognitiva intrínseca, según estos resultados el grupo experimental percibió los temas a aprender menos complejos que el grupo de control. Sin embargo, esto nos arroja pocos resultados,

debido a que en el grupo de control solo 14 contestaron el cuestionario de una población de 49.

Tabla 9. Resultados medición carga cognitiva estudio piloto.

Ejercicios tradicionales (14)				**Ejercicios auto-explicación (37)**			
	Promedio	Desviación estándar			Promedio	Desviación estándar	
1	7.14	2.07	IL = 6.79 σ = 1.85	1	5.25	2.59	IL = 4.96 σ = 2.42
2	6.93	1.90		2	5.41	2.23	
3	6.29	1.59		3	4.22	2.32	
4	1.64	1.65	EL = 1.74 σ = 1.59	4	2.08	2.44	EL = 2.22 σ = 2.23
5	1.64	1.95		5	2.11	2.07	
6	1.93	1.33		6	2.46	2.22	
7	7.43	2.14	GL = 7.70 σ = 1.84	7	7.08	2.50	GL = 7.34 σ = 2.18
8	7.79	1.81		8	7.30	2.23	
9	7.57	1.55		9	7.57	1.94	
10	8.00	1.96		10	7.43	2.05	

Segundo Estudio

De igual forma que en el estudio piloto, se midió el aprendizaje de los temas a través de las calificaciones obtenidas en los reactivos correspondientes a los temas propuestos de los exámenes parciales.

Los resultados estadísticos descriptivos del grupo experimental 1 (auto-explicación) de las calificaciones PRE y POST tratamiento, arrojan un promedio de 7.75 en las calificaciones antes del experimento, y un promedio de 9.45 posterior a este, lo que indica un aumento de 1.7 puntos en el promedio de calificaciones.

Tabla 10. Estadística descriptiva para el grupo experimental 1 (auto-explicación).

Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desv. típ.
AutoexpPre	50	3.60	10.00	7.7560	1.91878
AutoexpPost	50	8.00	10.00	9.4560	.74042
N válido (según lista)	50				

Para el caso del grupo de control se observa un promedio de 8.09 en las calificaciones PRE y 8.73 en las calificaciones POST, donde indica que se tiene una diferencia de 0.63 puntos mayor en la calificación POST.

Tabla 11.Estadística descriptiva para el grupo de control.

Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desv. típ.
ControlPre	46	.80	10.00	8.0957	2.55221
ControlPost	46	3.80	10.00	8.7326	1.84259
N válido (según lista)	46				

Para el caso del grupo experimental 2 (auto-explicación + videos de ejemplos resueltos) se obtuvo un promedio de 5.96 PRE y 8.39 POST, lo que refleja un aumento de 2.4 puntos en el promedio de las calificaciones.

Tabla 12.. Estadística descriptiva para el grupo experimental 2 (auto-explicación + videos ejemplo resuelto).

Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desv. típ.
EjemAutoexpPre	49	2.00	10.00	5.9694	2.15525
EjemAutoexpPost	49	2.00	10.00	8.3959	1.75570
N válido (según lista)	49				

Para confirmar si existe una diferencia estadística significativa se realiza la prueba t para los tres grupos con las calificaciones PRE y POST. El grupo experimental 1 (auto-explicación) la prueba t arroja un valor p de 0.000, lo que indica que, si existe una diferencia estadística significativa entre las calificaciones PRES y POST, lo que significa que se obtuvo una ganancia en el aprendizaje. El grupo experimental 2 (auto-explicación + videos ejemplo resuelto), la prueba t arroja un valor p de 0.000, lo que indica que, si existe una diferencia estadística significativa entre las calificaciones PRES y POST, lo que significa que también se obtuvo una ganancia en el aprendizaje.

En el caso del grupo de control la prueba t arroja un valor p de 0.017 lo que indica existe diferencia estadística significativa entre las calificaciones PRE y POST, lo que indica que el grupo de control también obtuvo una ganancia en el aprendizaje.

Los resultados anteriores se pueden observar en la gráfica de la ilustración.

Tabla 13. Resultados de prueba t, muestras relacionadas PRE-POST de los tres grupos.

Prueba de muestras relacionadas

		Diferencias relacionadas					t	gl	Sig. (bilateral)
		Media	Desviación típ.	Error típ. de la media	95% Intervalo de confianza para la diferencia Inferior	Superior			
Par 1	AutoexpPre - AutoexpPost	-1.70000	2.06990	.29273	-2.28826	-1.11174	-5.807	49	.000
Par 2	ControlPre - ControlPost	-.55111	1.48609	.22153	-.99758	-.10464	-2.488	44	.017
Par 3	EjemAutoexpPre - EjemAutoexpPost	-2.42653	2.52906	.36129	-3.15296	-1.70010	-6.716	48	.000

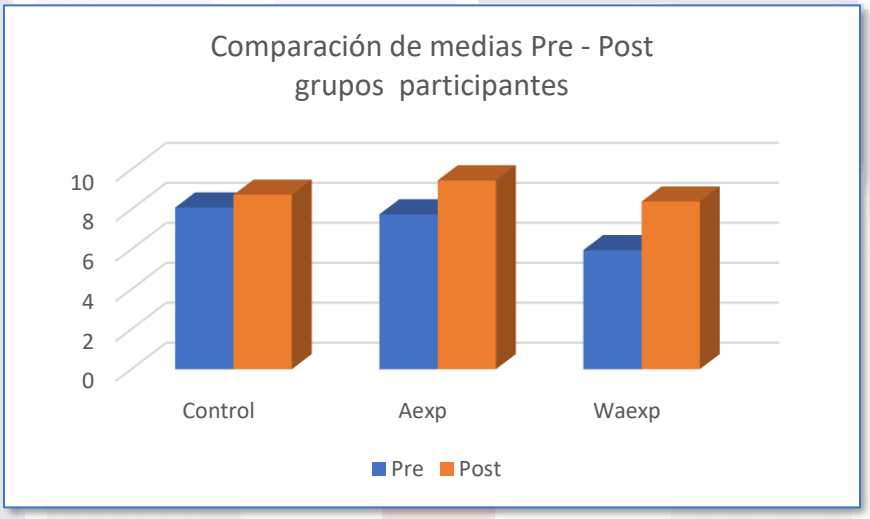


Ilustración 19. Comparación de medias PRE-POST grupos participantes.

Se realizó el Anova de un factor para los tres grupos, tanto para las calificaciones PRE como POST. El análisis Anova revela que los grupos son diferentes en cuanto al promedio de calificaciones tanto en el PRE como en el POST, con un valor p de 0.000 para las calificaciones PRE y un valor p de 0.002 para las calificaciones POST.

Tabla 14. Resultados Anova de un factor para los tres grupos en las calificaciones PRE.

ANOVA de un factor

CalificPre

	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Inter-grupos	126.042	2	63.021	12.849	.000
Intra-grupos	696.486	142	4.905		
Total	822.528	144			

Tabla 15.Resultados Anova de un factor para los tres grupos en las calificaciones POST.

ANOVA de un factor

CalifPost

	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Inter-grupos	29.051	2	14.525	6.296	.002
Intra-grupos	327.603	142	2.307		
Total	356.654	144			

Derivado del análisis Anova se realiza un análisis post-hoc Tukey y Gabriel, para identificar cuál de los grupos es el diferente en cuanto al promedio de calificaciones tanto en las PRE como en las POST.

Los resultados de la prueba post-hoc Tukey correspondiente a las calificaciones PRE, revelan que no hay diferencias entre el grupo de auto-explicación y el grupo de control en la medición pre con un valor p de 0.734. El grupo “más diferente” a los otros dos, es el grupo de Ejemplo resuelto y auto-explicación con un valor p de 0.000 para los

otros dos grupos. Los resultados de la prueba Gabriel básicamente, confirman los resultados de la prueba de Tukey, salvo que el valor p entre auto-explicación y Control es aún mayor, siendo de 0.836.

Tabla 16. Resultados prueba post-hoc Tukey calificaciones PRE.

Comparaciones múltiples

Variable dependiente: CalificPre
HSD de Tukey

(I) Grupo	(J) Grupo	Diferencia de medias (I-J)	Error típico	Sig.	Intervalo de confianza al 95%	
					Límite inferior	Límite superior
Autoexplicación	Control	-.33965	.45246	.734	-1.4113	.7320
	EjemploYautoexplicación	1.78661*	.44519	.000	.7322	2.8411
Control	Autoexplicación	.33965	.45246	.734	-.7320	1.4113
	EjemploYautoexplicación	2.12626*	.45467	.000	1.0494	3.2032
EjemploYautoexplicación	Autoexplicación	-1.78661*	.44519	.000	-2.8411	-.7322
	Control	-2.12626*	.45467	.000	-3.2032	-1.0494

*. La diferencia de medias es significativa al nivel 0.05.

Tabla 17. Resultados prueba post-hoc Gabriel calificaciones PRE.

Comparaciones múltiples

Variable dependiente: CalificPre
Gabriel

(I) Grupo	(J) Grupo	Diferencia de medias (I-J)	Error típico	Sig.	Intervalo de confianza al 95%	
					Límite inferior	Límite superior
Autoexplicación	Control	-.33965	.45246	.836	-1.4322	.7529
	EjemploYautoexplicación	1.78661*	.44519	.000	.7114	2.8618
Control	Autoexplicación	.33965	.45246	.836	-.7529	1.4322
	EjemploYautoexplicación	2.12626*	.45467	.000	1.0283	3.2243
EjemploYautoexplicación	Autoexplicación	-1.78661*	.44519	.000	-2.8618	-.7114
	Control	-2.12626*	.45467	.000	-3.2243	-1.0283

*. La diferencia de medias es significativa al nivel 0.05.

La prueba post-hoc Tukey para las calificaciones POST indican que, no hay diferencias entre el grupo auto-explicación y el de control (p=0.055), (***pero este valor está en el límite de significancia que es 0.05***), y tampoco entre el grupo de control y el de Ejemplo resuelto + auto-explicación (p=0.528). Los resultados indican que si hay diferencia significativa entre los dos grupos experimentales de auto-explicación y el de ejemplo resuelto + auto-explicación (p=0.002). Los resultados de

la prueba post-hoc Gabriel confirman los resultados de la prueba de Tukey, pero la significancia entre el grupo de control y el de auto-explicación está más lejos del límite de 0.05, siendo de 0.062.

Tabla 18.Resultados prueba post-hoc Tukey calificaciones POST.

Comparaciones múltiples

Variable dependiente: CalifPost
HSD de Tukey

(I) Grupo	(J) Grupo	Diferencia de medias (I-J)	Error típico	Sig.	Intervalo de confianza al 95%	
					Límite inferior	Límite superior
Autoexplicacion	Control	.72339	.31031	.055	-.0116	1.4584
	EjemploYautoexplicación	1.06008*	.30533	.002	.3369	1.7833
Control	Autoexplicacion	-.72339	.31031	.055	-1.4584	.0116
	EjemploYautoexplicación	.33669	.31183	.528	-.4019	1.0753
EjemploYautoexplicación	Autoexplicacion	-1.06008*	.30533	.002	-1.7833	-.3369
	Control	-.33669	.31183	.528	-1.0753	.4019

*. La diferencia de medias es significativa al nivel 0.05.

Tabla 19.Resultados prueba post-hoc Gabriel calificaciones POST.

Comparaciones múltiples

Variable dependiente: CalifPost
Gabriel

(I) Grupo	(J) Grupo	Diferencia de medias (I-J)	Error típico	Sig.	Intervalo de confianza al 95%	
					Límite inferior	Límite superior
Autoexplicacion	Control	.72339	.31031	.062	-.0259	1.4727
	EjemploYautoexplicación	1.06008*	.30533	.002	.3227	1.7975
Control	Autoexplicacion	-.72339	.31031	.062	-1.4727	.0259
	EjemploYautoexplicación	.33669	.31183	.628	-.4163	1.0897
EjemploYautoexplicación	Autoexplicacion	-1.06008*	.30533	.002	-1.7975	-.3227
	Control	-.33669	.31183	.628	-1.0897	.4163

*. La diferencia de medias es significativa al nivel 0.05.

Resultados medición de carga cognitiva segundo estudio

Los resultados obtenidos de la medición de carga cognitiva a los tres grupos indican que el grupo con resultados más altos en la carga cognitiva es el grupo experimental 2 auto-explicación + ejemplo resuelto videos. De igual forma los resultados no arrojan mucha información, al igual que en el estudio piloto la participación fue poca.

Tabla 20.Resultados medición carga cognitiva segundo estudio.

Grupo de control: Ejercicios tradicionales (23)				**Grupo experimental 1: Ejercicios auto-explicación (25)**				**Grupo experimental 2: Auto-explicación + ejemplo resuelto (30)**			
	Promedio	Desviación estándar			Promedio	Desviación estándar			Promedio	Desviación estándar	
1	3.48	3.23	IL = 3.06	1	4.56	2.63	IL = 4.13	1	6.00	2.52	IL= 6.60
2	3.30	3.48		2	4.12	2.74		2	8.47	1.70	
3	2.39	3.04	σ = 3.25	3	3.72	2.72	σ = 2.70	3	5.34	2.39	σ = 2.20
4	0.83	1.75	EL = 1.01	4	1.76	2.22	EL = 1.74	4	5.07	2.50	EL = 3.14
5	0.91	1.44		5	1.76	2.03		5	1.73	2.45	
6	1.30	2.05	σ = 1.75	6	1.71	1.15	σ = 1.80	6	2.63	2.81	σ = 2.59
7	8.00	1.93	GL = 8.09	7	7.52	1.15	GL = 7.83	7	2.17	2.49	GL = 6.74
8	7.78	2.26		8	7.80	1.66		8	8.27	1.28	
9	8.39	1.62		9	8.00	1.73		9	8.17	1.56	
10	8.17	1.87	σ = 1.92	10	8.00	1.55	σ = 1.52	10	8.37	1.69	σ = 1.76

CONCLUSIONES

A partir de los resultados obtenidos, donde se tiene en primer lugar el estudio piloto un valor p de 0.025 como resultado en la prueba t de las calificaciones PRE y POST correspondientes al grupo experimental, indicando una diferencia estadísticamente significativa entre ambas calificaciones, lo que afirma que el material instruccional proporcionado al grupo experimental tuvo un efecto positivo reflejándose en una ganancia positiva en el promedio de calificaciones posterior a la aplicación del mismo. En el caso del segundo estudio aplicado, se tuvieron dos grupos experimentales, en donde los resultados de la prueba t para las calificaciones PRE y POST para ambos grupos experimentales arrojaron un valor p de 0.000, indicando una diferencia estadísticamente significativa entre ambas calificaciones para ambos grupos, reflejando un efecto positivo traducido en una ganancia de aprendizaje que se muestra en el promedio de calificaciones posterior a la aplicación del estudio. Reafirmando lo anterior, en el caso del estudio piloto el aumento en el promedio de calificaciones posterior a la aplicación del tratamiento fue de 0.6 puntos, para el segundo estudio el aumento en el promedio de calificaciones para el grupo experimental 1 (auto-explicación) fue de 1.7 puntos y para el grupo experimental 2 (auto-explicación+ videos ejemplo resuelto) fue de 2.4 puntos.

Con base en lo anterior, se puede afirmar que la utilización de material instruccional diseñado utilizando la teoría de carga cognitiva como base, para este estudio específicamente el efecto de auto-explicación tiene un impacto positivo en el aprendizaje de los alumnos el cual se refleja en los resultados obtenidos en los promedios de las calificaciones.

Además, se reafirma que la combinación de efectos de la teoría de carga cognitiva como el ejemplo resuelto y problema por completar, con el efecto de auto-explicación producen resultados más efectivos.

El objetivo general de la tesis fue "*diseñar, desarrollar, probar y medir la efectividad de material instruccional para el aprendizaje de la programación de estudiantes universitarios de ciencias computacionales, utilizando elementos de diseño indicados por el efecto de auto-explicación de la teoría de carga cognitiva*", el cual se cumplió obteniendo resultados positivos.

Los objetivos específicos fueron cumplidos, como se detalla a continuación:

1. Identificar los conceptos de programación a enseñar (objetivos de aprendizaje).
    - Se identificaron los conceptos de programación a enseñar (objetivos de aprendizaje), para el estudio piloto arreglos, ciclos y matrices, para el segundo estudio cadenas.

2. Diseñar material didáctico para el aprendizaje de la programación basado en los lineamientos instruccionales recomendados por el efecto de auto-explicación de la Teoría de Carga Cognitiva.
    - Se diseño el material didáctico de los temas seleccionados siguiendo los lineamientos instruccionales del efecto de auto-explicación de la Teoría de Carga Cognitiva, para el caso del segundo estudio se agregaron videos de ejemplo resuelto a los ejercicios diseñados.

3. Identificar instrumentos de medición de carga cognitiva apropiados para el estudio del efecto de auto-explicación.
    - Se identificaron los diferentes tipos de instrumentos para medición de carga cognitiva como lo son subjetivos y objetivos, donde se optó por un método subjetivo (cuestionario de la NASA TLX) adaptado para programación.

4. Realizar estudios piloto para refinar el material instruccional diseñado y depurar el diseño experimental.
    - Se aplicó un primer estudio (piloto), con 7 ejercicios en formato de auto-explicación de los temas de arreglos, matrices y ciclos, y se utilizó el cuestionario para la medición de carga cognitiva. Se identificaron detalles a corregir para el segundo estudio.

5. Medir la efectividad del material instruccional con enfoque en los objetivos de aprendizaje.
    - La medición de la efectividad se realizó utilizando exámenes estandarizados y calibrados por el departamento de sistemas, mediante la comparación de las medias de los exámenes pre y post al tratamiento.

6. Medir la carga cognitiva de los estudiantes del grupo de control que se aplicó el material instruccional, así como de los estudiantes que se aplicó material instruccional tradicional.

- Para la medición de carga cognitiva se aplicó el instrumento de medición de carga cognitiva cuestionario de la NASA TLX adaptado para programación, posterior a la aplicación del tratamiento.

Respondiendo a las preguntas de investigación planteadas, se concluye que la utilización de material instruccional diseñado bajo los lineamientos de la teoría de carga cognitiva tiene una efectividad mejor ya que la ganancia en términos de aprendizaje (reflejado en los promedios de calificaciones) fue mejor que la metodología de enseñanza tradicional, y si se combina el efecto de ejemplo resuelto o problema por completar con el efecto de auto-explicación la efectividad reflejada es aún mayor.

Por último, en lo que respecta a los resultados de la medición de carga cognitiva, no se puede afirmar que exista una diferencia de carga cognitiva entre los estudiantes de metodología tradicional y estudiantes que utilizaron el material instruccional diseñado, puesto que la participación de los estudiantes en el proceso de medición de carga cognitiva fue muy pobre.

Limitaciones del estudio

En temas de objetivos de aprendizaje la utilización de la teoría de carga cognitiva para el diseño y aplicación de material instruccional proporcionan a los estudiantes alternativas para potenciar su aprendizaje con resultados positivos. Pero en el diseño y aplicación de

la presente investigación se identificaron limitaciones, las cuales se detallan a continuación.

En el contexto de la modalidad de educación a distancia producto de la pandemia de COVID-19, al no tener un control en la aplicación del tratamiento experimental, los participantes pudieron haber utilizado otras fuentes o herramientas de estudio adicionales al material instruccional proporcionado. Además, un componente del efecto de auto-explicación es la acción de “solicitar respuestas” (prompts) durante la explicación de un problema, esto no fue posible implementarlo y se sustituyó por explicaciones textuales o en audio, pero de forma asíncrona.

La aplicación del presente estudio se realizó bajo un diseño cuasiexperimental, no aleatorizado, por lo que los resultados obtenidos no son aún generalizables.

En la medición de carga cognitiva presentó limitaciones esto debido también a la modalidad de educación en línea, puesto que, al no tener un control en la aplicación de la herramienta para la medición, la participación fue muy pobre y fuera de tiempo, requiere ajustarse probablemente con instrumentos más precisos, bajo un proceso controlado y con mayor inmediatez al tratamiento.

REFERENCIAS

Alberto, G., Samperio, T., Árcega, A. F., De Jesús Gutiérrez Sánchez, M., & Navarrete, A. S. (2019). La Gamificación En Los Ambientes De Realidad Virtual Móvil the Gamification in the Environments of Mobile Virtual Reality. *Tecnológico Nacional de México En Celaya Pistas Educativas*, *41*(133), 671–699. http://itcelaya.edu.mx/ojs/index.php/pistas

Alomyan, H., & Green, D. (2019). Learning theories: Implications for online learning design. *ACM International Conference Proceeding Series*, 126–130. https://doi.org/10.1145/3355966.3358412

Andersen, S. A. W., Mikkelsen, P. T., Konge, L., Cayé-Thomasen, P., & Sørensen, M. S. (2016). The effect of implementing cognitive load theory-based design principles in virtual reality simulation training of surgical skills: a randomized controlled trial. *Advances in Simulation*, *1*(1), 1–8. https://doi.org/10.1186/s41077-016-0022-1

Andrade-Lotero, L. A. (2012). Cognitive load theory, design and multimedia learning: A state of the art. *Magis: Revista Internacional de Investigación En Educación*, *5*(10), 75–92.

Andreessen, B. M. (2011). Why Software Is Eating The World. *The Wall Street Journal*, 1–9.

Argelio, C., Mercado, A., Lizbeth, E., Andrade, M., Manuel, J., & Reynoso, G. (2018). El efecto de la autoeficacia y el trabajo colaborativo en estudiantes novatos de programación The effect of self-efficacy and peer collaboration in novice programming students. *INVESTIGACIÓN Y CIENCIA DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES*.

Astudillo, G. J., Bast, S. G., Willging, P., Segovia, D., Castro, L., & Distel, J. M. (2018). *Estrategias innovadoras en los procesos de enseñanza y de aprendizajes deinformática*. 420–424. http://sedici.unlp.edu.ar/handle/10915/67495

Aureliano, V. C. O., Tedesco, P. C. D. A. R., & Caspersen, M. E. (2016). Learning programming through stepwise self-explanations. *Iberian Conference on Information Systems and Technologies, CISTI*, *2016-July*. https://doi.org/10.1109/CISTI.2016.7521457

Becker, B. A., & Quille, K. (2019). *50 Years of CS1 at SIGCSE*. *1*, 338–344. https://doi.org/10.1145/3287324.3287432

Beltrán, J., Sánchez, H., & Rico, M. (2015). Análisis cuantitativo y cualitativo del aprendizaje de Programación I en la Universidad Central del Ecuador. *Revista Tecnológica - ESPOL*, *28*(5), 194–210. http://www.rte.espol.edu.ec/index.php/tecnologica/article/view/434/301

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin*, *39*(2), 32–36. https://doi.org/10.1145/1272848.1272879

Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in introductory programming - 12 years later. *ACM Inroads*, *10*(2), 30–35. https://doi.org/10.1145/3324888

Bisra, K., Liu, Q., Nesbit, J. C., Salimi, F., & Winne, P. H. (2018). Inducing Self-Explanation: a Meta-Analysis. *Educational Psychology Review*, *30*(3), 703–725. https://doi.org/10.1007/s10648-018-9434-x

Borzovs, J., Niedrite, L., & Solodovnikova, D. (2015). Factors affecting attrition among first year computer science students: The case of university of Latvia. *Vide. Tehnologija. Resursi - Environment, Technology, Resources*, *3*, 36–42. https://doi.org/10.17770/etr2015vol3.174

Bustingorry, S. O., & Mora, S. J. (2008). Metacognicion: Un camino para aprender a aprender. *Estudios Pedagogicos*, *34*(1), 187–197. https://doi.org/10.4067/S0718-07052008000100011

Cáceres, Z., & Munévar, O. (2017). Evolucion De Las Teorias Cognitivas Y Sus Aportes a La Educación. *Actividad Física Y Desarrollo Humano*, *7*(2). https://doi.org/10.24054/16927427.v2.n2.2016.2408

Centeno Bragado, L. A. (2018). *Virtualización de un juego serio para el aprendizaje de la programación. Virtualization of a serious game for learning programming.* Universidad de Valladolid.

Compañ-Rosique, P., Satorre-Cuerda, R., Llorens-Largo, F., & Molina-Carmona, R. (2015). Enseñando a programar: un camino directo

para desarrollar el pensamiento computacional. *Revista de Educación a Distancia (RED)*, *46*. https://doi.org/10.6018/red/46/11

Fisch, S. M. (2017). Bridging Theory and Practice: Applying Cognitive and Educational Theory to the Design of Educational Media. *Cognitive Development in Digital Contexts*, 217–234. https://doi.org/10.1016/B978-0-12-809481-5.00011-0

Fosnot, C. T. (2013). *Constructivism: Theory, perspectives, and practice*. Teachers College Press.

Fraser, S., Bak, L., DeLine, R., Feamster, N., Kuper, L., Lopes, C., & Wu, P. (2015). The future of programming languages and programmers (Panel). *SPLASH Companion 2015 - Companion Proceedings of the 2015 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity*, 63–66. https://doi.org/10.1145/2814189.2818719

Garcia, R. E. (2017). *Writing In-Code Comments to Self-Explain in*. *17*(4), 1–21.

Geela Venise Firmalo Fabic(&), Antonija Mitrovic, and K. N. (2017). Investigating the Effectiveness of Menu-Based Self-explanation Prompts in a Mobile Python Tutor. *Aied 2017*, *1*(August 2018), 498–501. https://doi.org/10.1007/978-3-319-61425-0_49

Guerrero, M., Guamán, D. S., & Caiza, J. C. (2015). Revisión de Herramientas de Apoyo en el Proceso de Enseñanza- Aprendizaje de Programación. *Revista Politécnica Nacional (Ecuador)*, *35*(1), 84–90.

Heitzmann, N., Fischer, F., & Fischer, M. R. (2018). Worked examples with errors: when self-explanation prompts hinder learning of teachers diagnostic competences on problem-based learning. *Instructional Science*, *46*(2), 245–271. https://doi.org/10.1007/s11251-017-9432-2

Hiller, S., Rumann, S., Berthold, K., & Roelle, J. (2020). Example-based learning: should learners receive closed-book or open-book self-explanation prompts? *Instructional Science*, *0123456789*. https://doi.org/10.1007/s11251-020-09523-4

Insuasti, J. (2016). Problemas de enseñanza y aprendizaje de los fundamentos de programación * Problems of teaching and learning

the basics of programming Problemas de ensino e aprendizagem dos fundamentos de programação. *Revista Educación y Desarrollo Social*, *10*(2), 12011–15318. https://doi.org/10.18359/reds.1701

Juáres, J., López, M., & Villareal, Y. (2016). Estrategias para Reducir el Indice de Reprobación en Fundamentos de Programación de Sistemas Computacionales del I . T . Mexicali. *Revista de Gestión Empresarial y Sustentabilidad*, *2*, 25–41. http://rges.umich.mx/index.php/rges/article/view/10/9

Kato, J., & Shimakage, K. (2020). Rethinking Programming "Environment": Technical and Social Environment Design toward Convivial Computing. *ACM International Conference Proceeding Series*, 149–157. https://doi.org/10.1145/3397537.3397544

Kelleher, C., & Pausch, R. (2005). Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Comput. Surv.*, *37*(2), 83–137. https://doi.org/10.1145/1089733.1089734

Kim, J., Agarwal, S., Marotta, K., Li, S., Leo, J., & Chau, D. H. (2019). Mixed reality for learning programming. *Proceedings of the 18th ACM International Conference on Interaction Design and Children, IDC 2019*, *2*, 574–579. https://doi.org/10.1145/3311927.3325335

Kirschner, P. A., Sweller, J., & 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. In *Educational Psychologist*.

Leppink, J., Paas, F., Van der Vleuten, C. P. M., Van Gog, T., & Van Merriënboer, J. J. G. (2013). Development of an instrument for measuring different types of cognitive load. *Behavior Research Methods*, *45*(4), 1058–1072. https://doi.org/10.3758/s13428-013-0334-1

Moreno, R. (2006). When worked examples don't work: Is cognitive load theory at an Impasse? In *Learning and Instruction* (Vol. 16, Issue 2 SPEC. ISS., pp. 170–181). https://doi.org/10.1016/j.learninstruc.2006.02.006

Moreno, R. (2009). Cognitive load theory: more food for thought. *Instructional Science*. https://doi.org/10.1007/s11251-009-9122-9

Morrison, B. B., Dorn, B., & Guzdial, M. (2014). *Measuring cognitive load in introductory CS*. 131–138. https://doi.org/10.1145/2632320.2632348

Oberhauser, R., & Lecon, C. (2017). Virtual reality flythrough of program code structures. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3110292.3110303

Price, T. W., Williams, J. J., Solyst, J., & Marwan, S. (2020). *Engaging Students with Instructor Solutions in Online Programming Homework*. 1–7. https://doi.org/10.1145/3313831.3376857

Radianti, J., Majchrzak, T. A., Fromm, J., & Wohlgenannt, I. (2020). A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda. *Computers and Education*, *147*(November 2019), 103778. https://doi.org/10.1016/j.compedu.2019.103778

Redifer, J. L., Therriault, D. J., Lee, C. S., & Schroeder, A. N. (2016). Working Memory Capacity and Self-Explanation Strategy Use Provide Additive Problem-Solving Benefits. *Applied Cognitive Psychology*, *30*(3), 420–429. https://doi.org/10.1002/acp.3219

Rittle-Johnson, B., & Loehr, A. M. (2017). Eliciting explanations: Constraints on when self-explanation aids learning. *Psychonomic Bulletin and Review*, *24*(5), 1501–1510. https://doi.org/10.3758/s13423-016-1079-5

Sands, P. (2019). Addressing cognitive load in the computer science classroom. *ACM Inroads*, *10*(1), 44–51. https://doi.org/10.1145/3210577

Shin, S. S. (2020). Structured Query Language Learning: Concept Map-Based Instruction Based on Cognitive Load Theory. *IEEE Access*, *8*, 100095–100110. https://doi.org/10.1109/ACCESS.2020.2997934

Shuell, T. J. (1986). Cognitive Conceptions of Learning. *Review of Educational Research*, *56*(4), 411–436. https://doi.org/10.3102/00346543056004411

Siemens, G. (2014). Connectivism: A learning theory for the digital age (2004). *Online: Http://Www. Elearnspace. Org/Articles/Connectivism. Htm (Accessed October 2012)*.

Simon, Luxton-Reilly, A., Ajanovski, V. V., Fouh, E., Gonsalvez, C., Leinonen, J., Parkinson, J., Poole, M., & Thota, N. (2019). Pass rates in introductory programming and in other STEM disciplines. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 53–71. https://doi.org/10.1145/3344429.3372502

Singh, G. (2017). Using virtual reality for scaffolding computer programming learning. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST*, *Part F1319*. https://doi.org/10.1145/3139131.3141225

Spyropoulou, M., Simaki, V., Koumpouri, A., Ntourou, D., Malagkoniari, D., & Sorra, M. (2013). Evaluating the correspondence of educational software to learning theories. *ACM International Conference Proceeding Series*, 250–257. https://doi.org/10.1145/2491845.2491882

Sweller, J., Ayres, P., & S., K. (2011). Cognitive Load Theory. In *Springer Science+Business Media, LLC 2013*. http://www.springer.com/series/8640

Sweller, J, van Merrienboer, J., & Paas, F. (1998). Cognitive Architecture and Instructional Design. *Educational Psychology Review*, *10*(3), 251–295.

Sweller, John. (2016). *Cognitive Load Theory and Computer Science Education*. 1–1. https://doi.org/10.1145/2839509.2844549

Sweller, John, van Merriënboer, J. J. G., & Paas, F. (2019). Cognitive Architecture and Instructional Design: 20 Years Later. *Educational Psychology Review*, *31*(2), 261–292. https://doi.org/10.1007/s10648-019-09465-5

Vihavainen, A., Miller, C. S., & Settle, A. (2015). Benefits of self-explanation in introductory programming. *SIGCSE 2015 - Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, *68*, 284–289. https://doi.org/10.1145/2676723.2677260

Watson, C., & Li, F. W. B. (2014). Failure rates in introductory programming revisited. *ITICSE 2014 - Proceedings of the 2014 Innovation and Technology in Computer Science Education Conference*, 39–44. https://doi.org/10.1145/2591708.2591749

Yen, C. W., & Wang, T. I. (2017). Using Self-Explanation and Ontology for Providing Proper Feedbacks in a Programming Environment. *Proceedings - 2017 6th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2017*, 585–590. https://doi.org/10.1109/IIAI-AAI.2017.136

Zagermann, J., Pfeil, U., & Reiterer, H. (2018). Studying eye movements as a basis for measuring cognitive load. *Conference on Human Factors in Computing Systems - Proceedings*, *2018-April*, 1–6. https://doi.org/10.1145/3170427.3188628

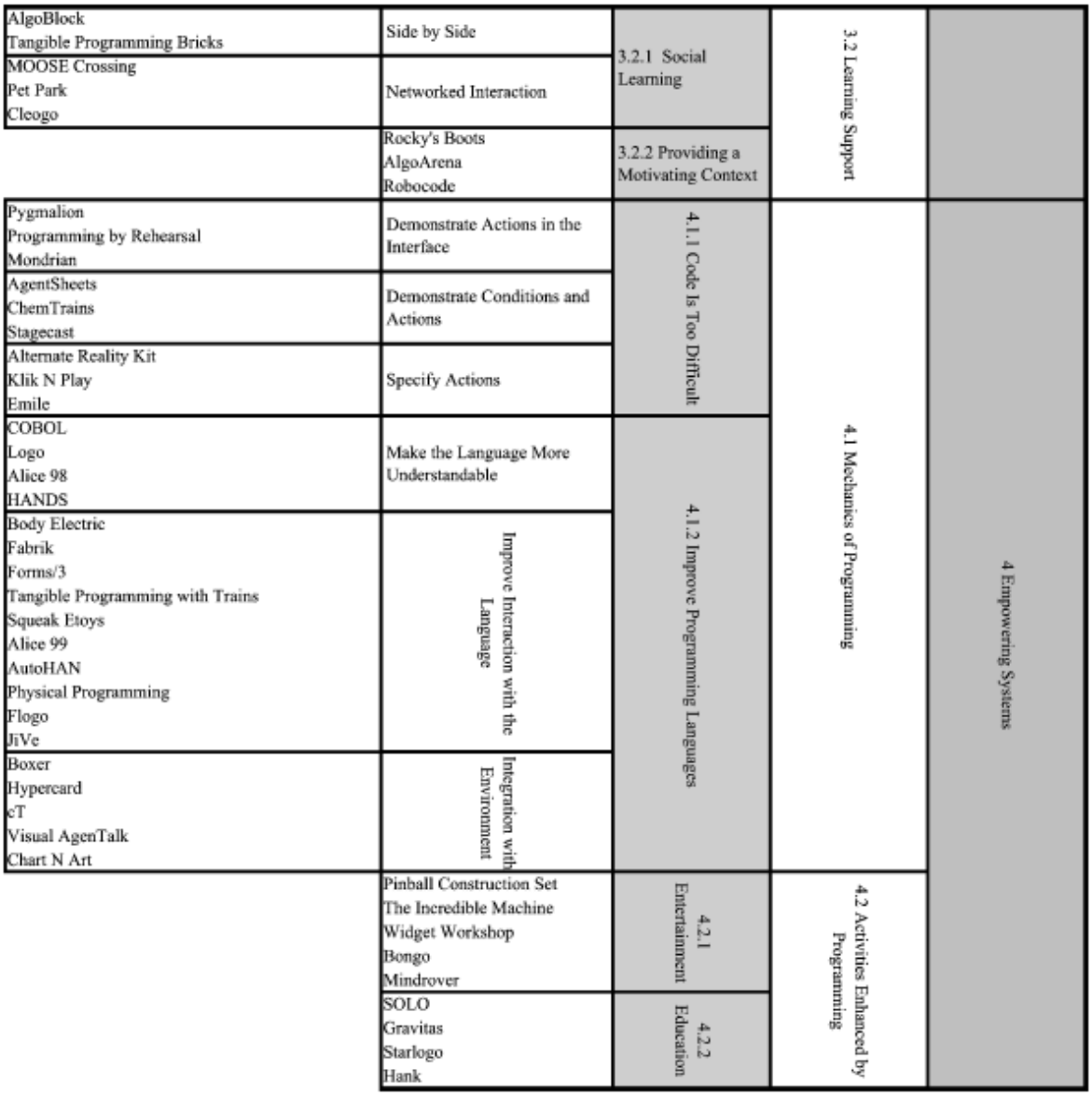
Zavgorodniaia, A., Duran, R., Hellas, A., Seppala, O., & Sorva, J. (2020). *Measuring the Cognitive Load of Learning to Program: A Replication Study*. 3–9. https://doi.org/10.1145/3416465.3416468

Zhiqing, Z. (2015). Assimilation, Accommodation, and Equilibration: A Schema-Based Perspective on Translation as Process and as Product. *International Forum of Teaching and Studies*, *11*(12), 84–89. https://pdfs.semanticscholar.org/41a2/bee7642597871e79b84fba3e7e8d25de1851.pdf

ANEXOS

Anexo 1. Clasificación de los lenguajes y entornos de programación según la taxonomía de (Kelleher & Pausch, 2005).

BASIC SP/k Turing Blue JJ GRAIL	1. Simplify the Language	Simplify Entering Code	3.1.1 Expressing Programs	3.1 Mechanics of Programming	3 Teaching Systems
Cornell Program Synthesizer GNOME MacGnome	2. Prevent Syntax Errors				
TORTIS-Slot Machine Pict Play Show and Tell My Make Believe Castle Thinkin' Things Collection 3: Half Time LogoBlocks Pet Park Blocks Drape Electronic Blocks Alice 2 Magic Forest	1. Construct Programs Using Graphical or Physical Objects	Find Alternatives to Typing Programs			
TORTIS - Button Box Roamer LegoSheets Curlybot	2. Create Programs Using Interface Actions				
Leogo	3. Provide Multiple Methods for Creating Programs				
	Pascal Smalltalk Playground Kara	New Programming Models	3.1.2 Structuring Programs		
	Liveworld Blue Environment/BlueJ Karel++ Karel J Robot J Karel	Making New Models Accessible			
	Atari 2600 Basic	Tracking Program Execution	3.1.3 Understanding Program Execution		
	Karel Josef Turingal	Make Programming Concrete: Actors in Microworlds			
	ToonTalk Prototype 2	Models of Program Execution			



Anexo 2. Clasificación “50 años de enseñanza CS1” (Becker & Quille, 2019).

Category	Sub - Category	Number of papers
First languages & Paradigms	General languages & Paradigms	4
	Specific paradigms	19
	Specific text-based languages	15

	Other	7
CS1 Design, structure & approach	General design, structure, approach	8
	How CS1 relates to CS0 or CS2	10
	Dealing with ACM model curricula	6
	Physical settings: Online, remote or MOOC delivery	15
	Physical settings: Labs	13
	Physical settings: Other	2
	Other	6
CS1 Content	Upper-level topics in CS1	9
	Software engineering approaches	11
	Other	18
Tools	Editors, API´s, etc.	9
	Libraries, etc.	6
	Visualization	7
	Debugging & testing	9
	Other	7
Collaborative approaches	Pair programming	14
	Peer instruction	6
	Other	16
Teaching	General teaching	3
	Model problems & exercises	24
	Specific topics (arrays, recursion, etc.)	6
	Games	9
	Hardware (robots, etc.)	8
	Aids, examples & tricks	12
	Flipped approaches	5
	Video	6
	Other	32
	General learning	1
	Conceptual or cognitive issues	9
	Learning styles	5

c

Learning & assessment	Reading, writing, tracing & debugging	13
	Errors	11
	Other learning	2
	General assessment	5
	Automatic tutoring & assessment systems	11
	Authentic assessment	6
	Exams	7
	Other assessment	11
Students	Non-majors	13
	Retention	11
	Gender, diversity, inclusion & accessibility	12
	Prior knowledge	5
	Concept inventories, geek genes, misconceptions	6
	Predicting & measuring success	17
	Programming process data	8
	Other	6

Anexo 3. Clasificación herramientas de apoyo en el proceso de enseñanza-aprendizaje de la programación desde un enfoque extenso en 4 grupos: calificación automática, multimedia, sistemas inteligentes de tutoría y de aprendizaje visual (Guerrero et al., 2015).

1. Herramientas de calificación automática:

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

2. Sistemas inteligentes de tutoría:

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

3. Herramientas de aprendizaje visual:

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

4. Herramientas multimedia utilizadas o desarrolladas en diferentes investigaciones:

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

Referencias proyectos citados en el punto 4:

Herramienta	Proyecto de investigación
Screencast	B. San Miguel, S. Aguirre, J. del Alamo and M. Cortés, "A proposal for enhancing the motivation in students of computer programming," *ICERI2012 Proceedings,* pp. 1157-1164, 2012.
Diario	S. Alhazbi, "Using e-journaling to improve self-regulated learning in introductory computer programming course," in *Global Engineering Education Conference (EDUCON), 2014 IEEE,* 2014, pp. 352-356.
Wiki	R. A. Lotufo, R. C. Machado, A. Körbes and R. G. Ramos, "Adessowiki on-line collaborative scientific programming platform," in *Proceedings of the 5th*

	International Symposium on Wikis and Open Collaboration, 2009, pp. 10.
Wiki	Universidad de Guadalajara, "Programa de apoyo a estudiantes en materias de programación LTI 2012-B,"
Imágenes	K. Olmos, C. Morales, T. Rojas and L. Fernández, "Objetos de aprendizaje enfocados a la resolución de problemas para facilitar la enseñanza de la programación,".
Cuestionarios	V. J. Carrasco, J. H Guevara J, “Implementación de un conjunto de objetos de aprendizaje y una aplicación web de cuestionarios dinámicos integrados en una herramienta de soporte para el autoaprendizaje de los conceptos de programación básica dirigido a estudiantes de la Facultad de Ingeniería en Electricidad y Computación”, Proyecto Fin de Carrera, ESPOL, Guayaquil, 2009. Disponible en: http://www.cib.espol.edu.ec/Digipath/D_Tesis_PDF/D-39238.pdf.
Juegos, textos, animaciones e imágenes	J. A. Villalobos, N. A. Calderon and C. H. Jiménez, "Developing programming skills by using interactive learning objects," *ACM SIGCSE Bulletin,* vol. 41, pp. 151-155, 2009.
Video, simulaciones y animaciones	J. H. Sharp and L. A. Schultz, "An exploratory study of the use of video as an instructional tool in an introductory C# programming course," *Information Systems Education Journal,* vol. 11, pp. 33, 2013.
Video	

	J. B. Fenwick Jr, B. L. Kurtz, P. Meznar, R. Phillips and A. Weidner, "Developing a highly interactive e-book for CS instruction," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education,* 2013, pp. 135-140.
e-book	S. Naz, S. H. Shirazi, T. Iqbal, D. Irfan, M. Junaid and Y. Naseer, "Learning Programming through Multimedia and Dry-run," 2014.
Imágenes	S. Alhazbi, "Using e-journaling to improve self-regulated learning in introductory computer programming course," in *Global Engineering Education Conference (EDUCON), 2014 IEEE,* 2014, pp. 352-356.

Anexo 4. Cuestionario diseñado para la medición de la carga cognitiva, tomando como referencia el cuestionario adaptado por (Morrison et al., 2014) en cursos introductorios de las ciencias de la computación.

1. ¿Los temas cubiertos en la actividad fueron muy complejos?
2. ¿La actividad contenía código que percibí como muy complejo?
3. ¿La actividad contenía conceptos y definiciones que percibí muy complejos?
4. ¿Las instrucciones y/o explicaciones durante la actividad fueron muy poco claras?
5. ¿Las instrucciones o explicaciones fueron, en términos de aprendizaje, muy poco efectivos?
6. ¿Las instrucciones y/o explicaciones contenían mucho lenguaje poco claro?
7. ¿La actividad realmente mejoró mi comprensión de los temas cubiertos?

8. ¿La actividad realmente mejoró mi conocimiento y compresión de la programación en general?
9. ¿La actividad realmente mejoró mi comprensión del código mostrado en los ejercicios?
10. ¿La actividad realmente mejoró mi comprensión de los conceptos y definiciones estudiados?