UNIVERSIDAD DE COSTA RICA

SISTEMA DE ESTUDIOS DE POSGRADO

SOFTWARE PARA LA EVALUACIÓN DE DESTREZAS METACOGNITIVAS EN EL APRENDIZAJE DE LA PROGRAMACIÓN

Tesis sometida a la consideración de la Comisión del

Programa de Doctorado en Educación

para optar al grado y título de Doctorado Académico en Educación

RICARDO GANG VINCENZI

Ciudad Universitaria Rodrigo Facio, Costa Rica

2023

Dedicatorias

A mi esposa Andrea, por su gran apoyo y por su amor

y a mis hijas Ari y Tati por haber hecho mejor al mundo con su llegada.

Agradecimientos

A Dios sobre todas las cosas, por su perfecto diseño y perfecta configuración de personas y circunstancias, para que la esperanza prevalezca siempre y la oportunidad mayor siempre sea la siguiente. A mi esposa, hijas, hermana y demás familia. A mi mamá que formó mi ser de la manera más dulce, con su ejemplo y a mi papá por ser mi mejor amigo.

¡Qué grandes maestros!

A don Mauricio, por su paciencia, guía y acertados consejos.

A mis asesores, porque no escatimaron en apoyo y voluntad de colaboración.

Esta tesis fue aceptada por la Comisión del Programa de Doctorado en Educación de la Universidad de Costa Rica, como requisito parcial para optar al grado y título de Doctorado Académico en Educación.

Dra. Flor Jiménez Segura
Decana del Sistema de Estudios de Posgrado

Dr. Mauricio Molina Delgado
Director de Tesis

Dra. Jaqueline García Fallas
Asesora

Dr. Carlos Vargas Castillo
Asesor

Dra. Annia Espeleta Sibaja
Representante Programa de Doctorado en Educación

Ricardo Gang Vincenzi
Sustentante

Resumen

Al momento de programar las acciones de los estudiantes, estas están guiadas principalmente, por su valoración del avance hacia la solución. En esta investigación educativa se propone una manera de explorar la relación entre destrezas metacognitivas y acciones al aprender a programar. Se capturan las acciones del principiante al momento mismo de la programación para su posterior análisis, en aras de realizar una medición de destrezas metacognitivas de regulación, organización y planificación de las subtareas que se realizan en la solución programada de un problema. Se estudia la relación entre las acciones y destrezas metacognitivas con análisis de grupos de estudiantes y la caracterización de estos por destrezas. Se propone la utilización de un sistema experto difuso, cuyas reglas de producción representen las relaciones entre acciones y destrezas metacognitivas, que puede refinarse con aplicaciones adicionales del software, en más contextos y con más estudiantes.

Contenidos

Resumen.....4
Lista de Tablas.....7
Lista de Figuras.....8
Capítulo 1 - Introducción.....1
1.1 Tema.....1
1.2 Problema.....5
1.3 Justificación.....5
1.4 Contexto.....7
1.5 Objetivos.....8
1.5.1 Objetivo General.....8
1.5.2 Objetivos Específicos.....9
1.6 Supuestos.....9
Capítulo 2 – Marco Teórico.....10
2.1 Metacognición.....10
2.1.1 Facetas.....13
2.1.1.1 (MK) Conocimiento metacognitivo.....14
2.1.1.2 (ME) Experiencias metacognitivas.....15
2.1.1.3 (MS) Destrezas metacognitivas.....16
2.2 Evaluación de la metacognición.....17
2.2.1 Medidas de auto reporte.....18
2.2.2 Instrumentos de prueba.....20
2.2.3 Métodos alternativos.....21
2.3 Programación de computadoras.....23
2.3.1 Aprendizaje de programación.....25
2.3.2 Metacognición y programación.....31
Capítulo 3 – Metodología.....35
3.1 Estudio 1.....36
3.1.1 Estudio 1 - Descripción.....36
3.1.2 Estudio 1 - Participantes.....36
3.1.3 Estudio 1 - Instrumentos.....37
3.1.4 Estudio 1 - Procedimiento y estrategia de análisis.....38
3.2 Estudio 2.....40
3.2.1 Estudio 2 - Descripción.....40
3.2.2 Estudio 2 - Participantes.....41

3.2.3 Estudio 2 - Instrumentos....41
3.2.4 Estudio 2 - Procedimiento y estrategia de análisis....41
3.3 Diseño y desarrollo del software....43
3.4 Estudio 3....44
3.4.1 Estudio 3 - Descripción....44
3.4.2 Estudio 3 - Participantes....45
3.4.3 Estudio 3 - Instrumentos....45
3.4.4 Estudio 3 - Procedimiento y estrategia de análisis....45
Capítulo 4 - Resultados....48
4.1 Resultados de Estudio 1....48
4.1.1 Codificación por temas emergentes....50
4.1.2 Codificación por destrezas metacognitivas....57
4.1.3 Sistema experto difuso....61
4.2 Resultados de Estudio 2....67
4.3 Resultados de Desarrollo de Software....70
4.3.1 Diseño....70
4.3.2 Módulo de Captura....74
4.3.2.1 Funcionamiento de la aplicación web....75
4.3.3 Módulo de Análisis....81
4.4 Resultados de Estudio 3....84
4.5 Indicadores de destrezas....88
4.5.1 Análisis de grupos....92
4.5.2 Mapa Auto Organizado....101
4.5.3 Acciones y destrezas....106
Capítulo 5 - Conclusiones y Recomendaciones....108
5.1 Discusión Estudio 1....109
5.2 Discusión Estudio 2....111
5.3 Discusión Estudio 3....111
5.4 Implicaciones....112
5.5 Limitaciones....113
5.6 Recomendaciones de investigación futura....114
5.7 Hallazgos....116
Referencias....118
Apéndices....147

Lista de Tablas

Tabla 3.1	37
Tabla 4.1	51
Tabla 4.2	52
Tabla 4.3	54
Tabla 4.4	55
Tabla 4.5	57
Tabla 4.6	59
Tabla 4.7	60
Tabla 4.8	61
Tabla 4.9	64
Tabla 4.10	65
Tabla 4.11	68
Tabla 4.11	69
Tabla 4.12	71
Tabla 4.13	74
Tabla 4.14	105

Lista de Figuras

Figura 3.1	35
Figura 3.2	39
Figura 3.2	43
Figura 4.1	49
Figura 4.2	66
Figura 4.3	72
Figura 4.4	76
Figura 4.5	77
Figura 4.6	77
Figura 4.7	78
Figura 4.8	80
Figura 4.9	82
Figura 4.10	83
Figura 4.11	86
Figura 4.12	87
Figura 4.13	90
Figura 4.14	91
Figura 4.15	94
Figura 4.16	95
Figura 4.17	97
Figura 4.18	98
Figura 4.19	100
Figura 4.20	102

Capítulo 1 - Introducción

Proporcionar fundamento teórico relacionado al aspecto metacognitivo que apoya el aprendizaje de la programación permite al docente orientar mejor sus esfuerzos, por ejemplo conociendo si el refuerzo metacognitivo debe ser adaptado al contexto de programación (Loksa et al., 2022; Prather et al., 2020). Es decir, se puede obtener un beneficio, si se conoce mejor la relación entre la teoría metacognitiva y de aprendizaje autorregulado y el aprendizaje de la programación que posibilita (Loksa et al., 2022).

Para desarrollar esto, se combinan diversos referentes de la teoría educativa con conceptos de programación desde la perspectiva de metacognición. Se buscó establecer relación entre las destrezas metacognitivas y el aprendizaje en programación. El presente estudio consta de capítulos que describen la base teórica, el diseño metodológico, el software desarrollado, el análisis que el software permitió realizar exitosamente y los resultados de ese análisis.

1.1 Tema

La investigación con respecto a la enseñanza de la programación suele desarrollarse evaluando alguna técnica didáctica y midiendo su éxito en un ámbito definido. No existe teoría formal, según las definen Goetz y Lecompte (1998), que orienten la enseñanza de la programación.

Cuantiosa literatura gira en torno a teorías sustantivas, según esta categorización, describiendo situaciones en ambientes muy bien definidos. Por ejemplo, Arshad (2009) reporta la técnica de *think-alouds* como medida didáctica. Esta técnica es utilizada para estudiar el proceso mental del individuo al realizar cualquier tarea. En este caso un experto en programación resuelve un problema frente a un público de estudiantes que es

incentivado a interactuar con el experto, el experto indica verbalmente sus procesos mentales (esta técnica es utilizada también para valorar metacognición).

En su artículo obtuvo resultados favorables en cuanto a la capacidad de resolución de problemas con un estilo adecuado. Esto le permite al estudiante producir códigos que favorezcan la reutilización finalmente recortando los tiempos de programación de nuevas soluciones.

Con esta técnica es posible identificar múltiples investigaciones educativas en programación que se basan en el constructivismo, explícita o implícitamente, como marco dentro del cual, proponen una técnica didáctica o estrategia metodológica. Múltiples estudios, con resultados positivos, negativos e inconclusos, constituyen en su fondo teórico, constructivismo educativo. Así, el constructivismo es la primera gran categoría identificada como de abordaje prioritario o al menos preferido en el planteamiento de una investigación educativa en programación.

Múltiples teorías han influenciado el trabajo realizado hasta el momento en la investigación educativa en programación. El construccionismo, el aprendizaje activo, constructivismo social e interaccionismo simbólico son ejemplos de esto. El construccionismo, por ejemplo, ha sido la base de múltiples lenguajes de programación en los que se proponen los principios de esta teoría como ejes centrales, con el fin específico de favorecer la enseñanza. LOGO (Papert, 1999) es un lenguaje de programación que ilustra este principio y fue propuesto por Papert (en algún momento colaborador de Piaget) como lenguaje que facilita el aprendizaje permitiéndole al estudiante construir un objeto manipulable y visible.

Desde una perspectiva de psicología cognitiva se demostró que, al aprender a programar en el lenguaje de programación LISP (Corbett & Anderson, 1992) un sistema (software) de tutoría inteligente (ITS, por sus siglas en Inglés) que permitía a estudiantes responder cada vez más rápidamente a los ejercicios propuestos en el sistema. La teoría de ACT (*adaptive control of thought*) (Anderson, 1976), llamada también arquitectura

cognitiva, ampliada a ACT* (Anderson, 1983) y luego a ACT-R (Anderson, 1993; Newell, 1990) constituye un modelo de los procesos cognitivos que ocurren al aprender.

Se enfoca en los componentes estructurales sobre los comportamientos observables. Esto incluye la representación del conocimiento y los elementos del proceso que influyen en la adquisición y almacenamiento de la información nueva adquirida. Esta teoría evoluciona del estudio de la memoria asociativa (HAM) del trabajo de Anderson con Bower (1973). ACT-R estudia la estructura de la mente humana pero también las aplicaciones del modelo en la inteligencia artificial; distingue entre tres tipos de memoria: declarativa, de trabajo actual y procedimental.

ACT* ya entendía el conocimiento como inicialmente declarativo que luego de obtenido podía pasar a ser procedimental con inferencias basadas en el conocimiento ya adquirido. Con FMRI confirman observaciones sobre la estructura del cerebro, por ejemplo, mapeando regiones del cerebro con actividad aumentada visible (Qin et al., 2007).

Otros investigadores, como Hull y Du Boulay (2009) se han enfocado en proponer soluciones, basándose en metacognición y motivación. Estos realizaron un trabajo en el que proponen un sistema computacional para combinar tutorías de programación con guías metacognitivas y motivacionales. Explorando la relación entre motivación y metacognición y muestran cómo pueden combinarse para producir un efecto mayor en los estudiantes al aprender a programar.

Con una perspectiva ligeramente distinta, Salomon y Perkins (1987) habían estudiado la relación entre el aprendizaje de programación y las capacidades cognitivas. Específicamente, trataron el cómo de la transferencia de habilidades adquiridas aprendiendo a programar hacia habilidades cognitivas en general. Señalaron el potencial del uso de la programación para desarrollar capacidades cognitivas.

Carbone et al., (2002) estudiaron el tema del desarrollo de capacidades cognitivas, específicamente en estudiantes de primer año de programación. En este estudio sugieren una serie de principios para diseñar actividades didácticas para enseñar programación que promuevan el desarrollo de capacidades metacognitivas.

El interés por relacionar la metacognición en esta población tiene que ver con lo novedoso del dominio de computación para el estudiante. La situación en la que se encuentra a menudo requiere de un gran esfuerzo mental que involucra conocer un lenguaje nuevo, una metodología (o conjunto de metodologías) de resolución de problemas, una estrategia de modelaje, capacidad de abstracción y planificación basándose en el modelo.

Estas habilidades requieren de un esfuerzo intencionado por parte del estudiante que es mejor guiado con mayor conocimiento acerca de su propio aprendizaje. Además, debe involucrarse en su propio aprendizaje y medir su avance con respecto al resto del grupo para regular sus acciones. A lo largo del documento se usa el término "metacognición" de manera general aunque en la investigación se enfocó específicamente en las destrezas metacognitivas (que son parte de la metacognición) por encima de la experiencia metacognitivas y el conocimiento metacognitivo. Todo esto se detalla más adelante.

Kramarski y Mizrachi (2004) estudiaron la manera en la que una guía metacognitiva puede mejorar el rendimiento en resolución de problemas matemáticos incluyendo el uso de razonamiento matemático y la comprensión del problema a resolver. Es posible que el aprendizaje en programación se vea beneficiado usando técnicas que además de enseñar el dominio, desarrollen las habilidades metacognitivas.

A pesar de estas investigaciones que relacionan la metacognición con la educación en programación, el presente trabajo investigativo puede desarrollarse más. La metacognición en general y las habilidades metacognitivas en particular, pueden proporcionar una perspectiva nueva para llenar vacíos teóricos y actuales en el campo. Eso sí, en la ausencia de trabajo investigativo en este sentido se dificulta evaluar fácilmente la influencia de estas categorías en el éxito de nuevas propuestas educativas en programación.

Para esta investigación se implementó un diseño metodológico que se detalla más adelante. Éste incluye la recolección de conocimiento adquirido mediante la experiencia, con entrevistas a expertos, la observación de estudiantes programando y el registro de

acciones de estudiantes programando, para relacionar éstas con las destrezas metacognitivas.

1.2 Problema

Una pregunta inicial y general, sobre la cual podría plantearse una investigación es, entonces: ¿cómo mejorar el proceso de enseñanza/aprendizaje de la programación de computadoras? Esta interrogante resume un problema que podría tener varios aspectos a considerar y su solución podría tener implicaciones favorables en el desempeño de los estudiantes de computación a lo largo de la carrera.

Aún así se trata de un problema muy general y su manejo puede resultar amplio y de complejidad prohibitiva. Un problema relacionado, pero más manejable, podría plantearse sobre el tema de la metacognición en estudiantes principiantes de programación y cómo enfocar esfuerzos docentes en torno a estas variables.

Más específicamente, y planteado como problema: existe poca información acerca de cómo las destrezas metacognitivas son utilizadas por estudiantes principiantes de programación. Este problema se debe en parte a la dificultad de evaluar la metacognición en estudiantes que interactúan con una computadora, cuando muchas veces, incluso, esa interacción es la que posibilita el aprendizaje.

1.3 Justificación

La programación es fundamental en las ciencias de la computación. Como menciona Reynolds: "Es también el quehacer de la programación que yace en el centro intelectual de las ciencias de computación (tan central para este tema como la mecánica clásica para la física), y ha dado paso al concepto de pensamiento computacional" (2008). Esto se debe a que por medio de esta actividad es que se logran utilizar los recursos computacionales de manera que se conviertan en una herramienta útil, pues sin la

programación el hardware sería un simple circuito que, aunque sea digital secuencial, sería incapaz de ser configurado o utilizado para realizar diversas tareas.

La importancia de mejorar la calidad de la enseñanza y el aprendizaje de la programación viene de su papel central para la carrera de computación. Así, los estudiantes serían los primeros beneficiados por el estudio de este problema. Si la investigación propuesta sirve de base para la evaluación y la creación de instrumentos y técnicas didácticas apropiadas, los docentes se benefician también. Esta investigación tiene también efectos positivos sobre la organización del currículo considerando el contenido de los cursos. La investigación posibilita llegar a la base de la actualización y posible reacomodamiento de los contenidos de los cursos actuales de programación.

La importancia de este problema tiene que ver con la relevancia de la computación misma. La computación está presente en casi cualquier ámbito de la vida cotidiana. Esto es igualmente cierto para informáticos y científicos de la computación como para cualquier persona ajena a esta área de profesionalización. Autores como Reynolds, consideran que, a mejor capacidad de programación mayor es la calidad del trabajo de un científico de la computación, es decir: "la habilidad de construir software confiable y eficiente es la marca central de un profesional talentoso de computación, y la rareza presente de esta habilidad es la mayor causa de la triste prevalencia de sistemas computacionales plagados de errores." (2008).

Desde la perspectiva más optimista, mejorando los sistemas computacionales que nos rodean se podría notar cambios favorables en muchos aspectos de la vida cotidiana de la humanidad. Desde una perspectiva más individualista, la importancia de este problema está en el efecto positivo sobre el desempeño de los estudiantes de computación y las implicaciones que esto tendría para su vida profesional y académica.

1.4 Contexto

La ubicación de los cursos de programación al inicio del plan de estudios de computación es por ser éste un tema fundamental para los demás cursos. Esto define también el tipo de población con la que se está tratando. Los cursos de programación se imparten a estudiantes principiantes de la carrera de computación principalmente, pero también a estudiantes principiantes de otras carreras.

Históricamente, según consulta a las bases de datos de la Escuela de Ciencias de Computación e Informática (ECCI), los cursos de Programación I y II tienen una promoción de 49%. Esta población, cercana a la mitad de los estudiantes, que sí logra avanzar lo ha hecho con notas cercanas a la nota mínima para aprobar un curso (nota 70). Esto origina toda clase de problemas en cursos subsiguientes pues los estudiantes no cuentan con buenas bases en programación. Por lo que, se hace necesario, también, contemplar la posibilidad de que la población meta de esta investigación incluya una cierta cantidad de repitentes del curso por lo que debe decidirse si incluirlos o no en la investigación.

Las tasas de aprobación de los cursos introductorios de programación han sido tratados por múltiples autores (Bennedsen y Caspersen, 2007, 2019; Mehmood et al., 2020; Watson y Li, 2014). En general, hay consenso de la dificultad del curso pero no de cifras alarmantes ni de fracasos ni de deserción. Watson y Li llegaron a una cifra de 67% tasa de éxito en el curso introductorio de computación igual a la cifra primero reportada por Bennedsen y Caspersen, los primeros con un estudio de 161 cursos y los últimos con datos de 63 instituciones . Doce años más tarde estos últimos actualizaron su cifra a un 72% de aprobación del curso. Por lo tanto, en todos los casos los autores señalaron cursos, como los de álgebra, con menores tasas de aprobación.

Una revisión sistemática (Mehmood et al., 2020) identificó, utilizando 60 fuentes de información relacionadas a cursos introductorios de programación, tres grandes categorías de clasificación: artículos sobre currículum, enseñanza aprendizaje y evaluación. Esta revisión sirve para establecer un marco para la investigación relacionada a la educación en

computación. La selección de artículos fue sistemática y redujo, según criterios claramente enunciados, la selección de artículos de miles de artículos a los 60 mencionados. La exclusión de artículos se realizó mediante un panel de aprobación que analizaba cada exclusión.

Usualmente, los programas de computación incluyen más de un curso dedicado a la enseñanza de la programación. En el primer curso de programación se enseñan conceptos fundamentales que en los cursos siguientes son ampliados y utilizados para aplicaciones más complejas.

Existe un ente llamado Association for Computing Machinery (ACM) y otro llamado Institute of Electric and Electronics Engineers (IEEE) que lideran un esfuerzo por mantener un currículo de computación que reúna las tendencias de Universidades del mundo y que contemple actualizaciones necesarias por razones tecnológicas y de mercado laboral. El currículo más reciente para ellos fue presentado en el 2020 (ACM/IEEE, 2020). En este documento se incluye un listado de los conceptos y habilidades considerados dentro de lo que describen como fundamentos de programación. Esta lista y la siguiente (ACM., 2013) se utilizaron en esta investigación.

Se indagó en la Universidad de Costa Rica, por lo que la población de estudiantes fue, en su mayoría, ciudadanos costarricenses. Por otro lado, se aplicó el software dos veces con estudiantes del curso introductorio en Java llamado Programación 1 de la Escuela de Ciencias de la Computación e Informática.

1.5 Objetivos

1.5.1 Objetivo General

Construir una solución de software, basada en destrezas metacognitivas, que registre comportamientos indicativos de las destrezas en estudiantes principiantes de programación, para fundamentar el fortalecimiento de la enseñanza de la programación.

1.5.2 Objetivos Específicos

1. Identificar las interacciones con la computadora del estudiante principiante en programación que puedan dar información acerca de sus destrezas metacognitivas.
2. Diseñar y desarrollar software para el registro de interacciones del estudiante principiante en programación con el computador que brinden información sobre sus destrezas metacognitivas.
3. Identificar los efectos en el código indicativos de relaciones entre destrezas metacognitivas y la programación mediante el análisis de la información recolectada por el software.
4. Analizar e interpretar las relaciones encontradas entre efectos en el código y destrezas metacognitivas para mejorar la comprensión de la manera en la que los estudiantes principiantes de programación utilizan sus habilidades metacognitivas para aprender a programar.

1.6 Supuestos

Esta investigación supone que los comportamientos del programador en la interacción de la computadora se pueden registrar de manera útil, para medir las destrezas metacognitivas. Esto porque no necesariamente existe un mapeo de uno a uno entre destrezas y acciones y tampoco se puede decir que toda acción corresponde a una destreza metacognitiva. Por lo que, este supuesto implica la posibilidad de obtener conclusiones útiles e informativas aún con estas condiciones.

Capítulo 2 – Marco Teórico

2.1 Metacognición

El proceso de monitoreo de la memoria y su relación con el "saber que se sabe" o sentimiento de saber, fue estudiado por Hart (1965, 1966, 1967) evaluando la precisión con que un individuo podía estimar correctamente cuánto retiene en memoria. Se ha confirmado que, para casos de retención de datos es posible tener una buena valoración de cuánto se conoce (Gruneberg y Monks, 1974). Análogamente, se ha estudiado el fenómeno de punta de la lengua, que se da al producirse cierta ansiedad o inquietud al conocer que se tiene recolección parcial de un dato pero también saber que se conoce el dato completo pero hay dificultad para recuperarlo (Freedman y Landauer, 1966; Roger Brown y David McNeill, 1966) . Es decir, conocimiento sobre el conocimiento. También se ha estudiado la relación y diferencia de este fenómeno con el sentimiento de saber (S. R. Brown, 2000; Schwartz, 2006).

Giambattista Vico argumentó, por otro lado, que la realidad es producto de la creación del individuo, es decir, su conocimiento es una construcción activa según sus experiencias (1710). La teoría del constructivismo es básicamente sobre el aprendizaje y según múltiples autores es posible describir cómo el conocimiento acerca de las cosas exteriores se construye dentro del individuo que las aprende de manera activa y no simplemente como receptor pasivo de información (Bruning et al., 2004; Confrey, 1990; Fox, 2001).

Schraw y Moshman (1995) propusieron que la interacción social, junto con el aprendizaje cultural y la construcción individual son indispensables para la creación de teorías metacognitivas, que son teorías acerca de su propia cognición. Previamente, para Vygotsky la regulación surge a partir de la interacción con el mentor o experto que primeramente cumple un papel de fijar metas y planificar el aprendizaje pero que, posteriormente, el control y la regulación, son asumidas por el que aprende (1978, 1981).

El tratamiento que hace Vygotsky de la regulación del aprendiz, lo convierte en un precursor del estudio de la metacognición (Bråten, 1991). En el constructivismo dialéctico (versus exógeno o endógeno) el conocimiento sucede a partir de las interacciones que ocurren entre el individuo y la realidad externa (Bruning et al., 2004). Es decir, para crear conocimiento, el individuo pasa por un proceso de regulación donde internamente confronta lo que conoce ya, con información externa del momento y con información adicional sobre las interacciones que suceden en el momento del aprendizaje.

John Locke habló de ideas complejas, que definió como ideas compuestas de otras ideas, algunas de las cuales son de reflexión, que definió como el notar las operaciones propias de la mente por lo que surgen ideas acerca de estas operaciones (Locke, 1924). Jean Piaget (1977) estudió cómo se desarrolla la reflexión (*reflecting abstraction*), que definió como la construcción de conceptos de más alto orden basándose en las experiencias previas. La introspección es otro concepto, que fue tratado de previo a metacognición, que se refiere al proceso de reflexión interno sobre el pensamiento, fue utilizado en contextos muy limitados (James, 1890; Wundt, 1912).

Flavell (1976), definió metacognición como el conocimiento de uno acerca de la propia cognición o cosas relacionadas. Si se entiende que la cognición involucra percepción, memoria, representación del conocimiento, emociones, razonamiento y solución de problema entonces la metacognición es cognición sobre la cognición propia. En neurociencias cognitivas la metacognición en relación con tomas de decisiones se relacionan con actividad en la corteza pre frontal que ejerce control mediante la recepción e interpretación de estímulos sensoriales (Fleming et al., 2012).

Se puede hablar de dos modelos de metacognición, el de Flavell (1979) en el que se tienen cuatro partes o componentes: el conocimiento metacognitivo, las experiencias metacognitivas, las metas y las estrategias. En este modelo el conocimiento es lo concerniente a las tareas, metas y situación del individuo y las experiencias. El componente de metas incluye la planificación necesaria para la obtención de las mismas y las estrategias son para el monitoreo y regulación de la cognición. Esto se relaciona con lo

que el autor trató primero como metamemoria (J. Flavell, 1971), que describió como la memoria con la función de monitoreo y regulación de operaciones de búsqueda y recuperación de lo que se conoce.

El modelo de Brown (1987), relacionado, no independiente al modelo de Flavell, está compuesto por dos formas o dimensiones: conocimiento metacognitivo y regulación metacognitiva. Brown señala que ambas formas están fuertemente relacionadas por lo que separarlas para entenderlas por aparte lleva a una sobresimplificación pero que sí son distinguibles. El primero es relativamente estable, es decir, no cambia hasta que cambien el conocimiento de un dominio y además es comunicable por el sujeto, aun cuando lo que se cree que se sabe puede estar equivocado.

Por otro lado, según la misma autora, la regulación incluye las estrategias de monitoreo, planificación, fijación de metas y evaluación durante el aprendizaje. Además, esta regulación no siempre es comunicable, pero es posible saber hacer algo pero no saber que se logró hacerlo y qué acciones se tomaron.

La metacognición es multidimensional pues el conocimiento incluye conocimiento procedimental (sobre cómo hacer cosas), declarativo (sobre uno mismo y qué factores pueden mejorar el rendimiento y condicional (sobre cuándo usar conocimiento procedimental o declarativo) y la regulación incluye planificación, monitoreo y evaluación (Schraw, 1998).

Se han hecho investigaciones para evaluar la utilidad de promover la metacognición, típicamente mediante algún esfuerzo del docente, por ejemplo, proponiendo activamente actividades que obliguen a realizar más pensamiento metacognitivo (Claudine Kirsch, 2012; Downing et al., 2009; Sara Cotterall y Garold Murray, 2009). Analizando las prácticas de estudiantes previo a una intervención y posteriormente para evaluar la mejoría, en algunos casos se comprueban mejoras en metacognición, pero no necesariamente en el rendimiento académico (Annelies Raes et al., 2016; David R. Thompson, 2012; De Backer et al., 2015; Fu-Yun Yu, 2005; Khosa y Volet, 2013; Kramarski, 2008; Michelle Bauml y Sarah Quebec-Fuentes, 2016).

Se ha demostrado que es posible mejorar la capacidad metacognitiva promoviendo mediante alguna herramienta o apoyo metacognitivo, (Danielle S. McNamara y Amy M. Shapiro, 2005; Gulbinskienė et al., 2017; Kapa, 2001; Manoli Pifarré y Ruth Cobos, 2010). En algunos casos, se ha demostrado también una mejora en el rendimiento académico o comprensión de la materia vista, por ejemplo brindando apoyo metacognitivo por correo electrónico, entre docentes y estudiantes (Kramarski y Ritkof, 2002) o, similarmente, mediante otro medio electrónico, el de foros (Kramarski y Mizrachi, 2004). Según D'Avanzo (2003) cuando se apoya la metacognición se desarrolla también el aprendizaje.

2.1.1 Facetas

Se han hecho esfuerzos por organizar el estudio de la metacognición identificando facetas tales como: conocimiento metacognitivo (MK), destrezas metacognitivas (MS) y las experiencias metacognitivas (ME) (Efklides, 2001, 2008; Efklides y Tsiora, 2002). Estos autores señalan que la relación de la metacognición con tantos procesos diversos de la regulación del comportamiento, como las interacciones sociales, la resolución de problemas y el funcionar cotidiano, se debe a la naturaleza multifacética y multinivel de la metacognición. Además de las facetas mencionadas (MK, MS y ME) indican que pueden suceder a un nivel consciente o inconsciente. Por ejemplo, el monitoreo en la realización de una tarea puede ser inconsciente incluso cuando genera pequeños ajustes de control al detectar inconsistencias entre el resultado esperado y el obtenido. Sin embargo, cuando la corrección efectuada inconscientemente no es suficiente se produce un efecto afectivo en la forma de sentimientos metacognitivos (Efklides, 2008). El modelo teórico descrito por Efklides (2007) se basa en el modelo de regulación y control metacognitivo (Nelson y Narens, 1990) en el que las facetas forman parte del Meta Nivel que controla las acciones del Nivel de Objetos y a la vez monitorea este nivel, en un ciclo.

Cada una de estas facetas se puede analizar en distintos niveles pudiendo incluso ubicarse dentro del esquema previamente descrito de la memoria declarativa y no declarativa. Estudiando la relación entre metacognición y aspectos afectivos (Efklides,

2005) por lo que se han observado los juicios acerca de la dificultad de una tarea o los sentimientos acerca de la confianza generada resolviendo un problema y la sobre confianza en el rendimiento (Sidi et al., 2017).

2.1.1.1 (MK) Conocimiento metacognitivo

Se trata del conocimiento acerca de las tareas posibles (incluyendo metas), estrategias y sus usos y conocimiento acerca de uno mismo o los demás (J. H. Flavell, 1979). Incluye conocimiento declarativo (acerca de cosas), pero también, como ya se mencionó, procedimental (acerca de cómo hacer cosas) y condicional (acerca de la toma de decisiones) (G. Schraw, 1998; G. Schraw y Dennison, 1994). Se ha demostrado relación entre la Teoría de Mente (TOM) y el conocimiento metacognitivo (Lockl y Schneider, 2007), en un estudio longitudinal concluyeron que el desarrollo temprano de la TOM es precursor de metamemoria, donde se almacena el conocimiento metacognitivo acerca del uso de la memoria, que es conocimiento acerca de memoria y habilidades relacionadas al monitoreo y a la autorregulación de la memoria. El conocimiento metacognitivo ha sido ligado al uso del lenguaje por las capacidades que provee al individuo de representación interna, reflexión y posteriores inferencias de la relación entre lo interno y lo externo (Efklides, 2008; Lockl y Schneider, 2007). Por ejemplo, estudiando el desarrollo del conocimiento metacognitivo, se ha ligado el conocimiento metacognitivo declarativo con la capacidad de verbalización (Schneider, 2008; Schneider et al., 2022).

Se ha determinado que el conocimiento metacognitivo es la base para organizar el aprendizaje (Ismael, 2016), incluyendo estrategias de aprendizaje (Cordula Artelt y Wolfgang Schneider, 2014) y se ha mostrado que se correlaciona con mejores rendimientos y mejor motivación en el aprendizaje autorregulado (Efklides, 2014; Efklides y Metallidou, 2020; Gregory Schraw et al., 2011; Hubertina Thillmann et al., 2012; Tiago Roberto Kautzmann y Augustin, 2019; Yi Tian et al., 2018).

Se han estudiado las funciones que permite el conocimiento metacognitivo, como la capacidad de llevar a cabo una conversación pues sirve para unir temas y guiar los

mensajes que se desean comunicar (Bruning et al., 2004). Esto pues permite enlazar las ideas, logrando coherencia. También basándose en el conocimiento metacognitivo, se puede producir la autoexplicación, que es la explicación a uno mismo de lo que está aprendiendo (Aleven y Koedinger, 2002). Estudiando la utilidad de la autoexplicación, estos autores concluyeron que estudiantes que recibieron tutoría en auto explicación lograron un aprendizaje más profundo comparado con estudiantes que solamente resolvieron problemas como práctica.

2.1.1.2 (ME) Experiencias metacognitivas

Flavell (1979) describió las experiencias metacognitivas como instancias de conocimientos metacognitivos que pasan a la consciencia, incluyendo información acerca de tareas, metas y estrategias . Efklides (2001, 2005)las describió como de lo que se es consciente y lo que se siente al realizar una tarea y procesar la información relacionada a esa tarea. Lo que caracteriza, principalmente, las experiencias metacognitivas es que involucran juicios y evaluaciones acerca de la tarea o el proceso, ya sea su desarrollo o sus resultados (Efklides, 2008). Es aquí donde se reúnen experiencias como el "sentimiento de saber" y el fenómeno relacionado de "punta de lengua", también el sentimiento de confianza y el sentimiento de familiaridad. (2008). Incluye también la autorregulación de esfuerzo y juicios sobre el esfuerzo, aprendizaje, tiempo requerido y demás sentimientos relacionados a la dificultad y confianza con la que se realizan las tareas (Efklides y Tsiora, 2002).

Estudiando las experiencias metacognitivas en la resolución de problemas se ha mostrado que los reportes de confianza, satisfacción y estimación de correctitud (juicios de experiencias metacognitivas) son más elevados en casos de éxito realizando una tarea y los reportes de sentimiento de dificultad, tiempo requerido y necesidad de pensamiento (también juicios de experiencias metacognitivas) son más elevados en casos de fracaso, lo que sugiere que los individuos reflejan en su reporte de experiencias metacognitivas la evaluación de su rendimiento final o eficacia (Akama y Yamauchi, 2004).

2.1.1.3 (MS) Destrezas metacognitivas

Cuando se recurre intencionalmente a una estrategia o procedimiento para controlar la cognición se está desarrollando una destreza metacognitiva (Efklides, 2008). Las destrezas metacognitivas son los procesos necesarios para utilizar información metacognitiva ya aprendida y el conocimiento sobre estos procesos (M. Veenman et al., 2006). Incluye la organización, planificación, monitoreo y evaluación para llevar a cabo el control cognitivo (M. Veenman y Elshout, 1999). Bannert y Mengelkamp (2007) entienden las habilidades metacognitivas como aquellas necesarias para llevar el aprendizaje a mejorar mediante la autorregulación e incluyen habilidades como control, monitoreo, planificación cuando se aprende y cuando se resuelve un problema.

El uso de estrategias de control de la cognición propias del ciclo regulatorio cognitivo se combina con MK además de información específica a la tarea que se esté desarrollando, ejecutándose MS cuando se tienen ME de que algo fluye correcta o incorrectamente (Efklides, 2008). Según esta autora, en este sentido ME es la entrada que impulsa el control mediante MS, basándose en MK. Así, las tres facetas cumplen funciones distintas en la autorregulación, MK y ME brindan insumos que disparan MS para el control de la cognición (2008).

El monitoreo permite identificar potenciales errores más temprano lo que lleva a tomar mejores decisiones y se puede mejorar con actividades como múltiples revisiones de redacción de un documento (Cho et al., 2010; Cho y Cho, 2007; Yeung y Summerfield, 2012). Además el monitoreo de errores, contrario al monitoreo de desempeño, puede estar relacionado con la ansiedad (Moser et al., 2013). La calidad del monitoreo parece ser sensible a la dificultad de la tarea que se realiza y que puede mejorar con la práctica (Kessel et al., 2014).

Se ha enlazado el desarrollo de destrezas metacognitivas con mejor rendimiento académico (Adiansyah et al., 2021; Albab et al., 2020; Donna Bryce et al., 2015; Ido Roll et al., 2011; Michael Hogan et al., 2014). Algunos estudios detectaron una correlación

positiva entre estrategias metacognitivas (que combinan múltiples destrezas incluyendo regulación) y el rendimiento académico (Vrugt y Oort, 2008), pero concuerdan en que el componente metacognitivo no es el único que explica el éxito académico. Incluso la correlación depende de factores como el nivel académico general y la materia a cubrir (Adiansyah et al., 2021)

2.2 Evaluación de la metacognición

Se ha tratado la medición de habilidades metacognitivas, por ejemplo en relación con la comprensión de lectura y la manera de medir progreso metacognitivo (Block, 2005). Se han identificado dos categorías en las que se han concentrado investigaciones similares con una visión positivista descontextualizante y otra naturalista contextualizante (Anderson et al., 2008). En la primera algunos aspectos y variables propias del contexto como el contenido, la motivación de los estudiantes, el tiempo y la complejidad de las tareas no son tomados en cuenta a favor de buscar soluciones sencillas a problemas complejos mediante un modelo artificial (Nelson y Narens, 1990). En el paradigma naturalista contextualizante esas variables son tan importantes como cualquier otra cosa y la explicación debe considerar estos aspectos (D. Anderson y Nashon, 2007; G. P. Thomas y McRobbie, 2001).

Las técnicas para valorar la capacidad metacognitiva de un estudiante pueden ser en línea y fuera de línea, los primeros siendo los que se ejecutan en el momento de la actividad metacognitiva y los otros no (M. Veenman, 2005; M. Veenman et al., 2006). Veenman (2005) señala que el uso de métodos en línea es más apropiado para valorar el uso de habilidades metacognitivas pues el uso de métodos fuera de línea, que requieren retrospección y memoria, fallan, prediciendo resultados de aprendizaje al ser menos precisos y subjetivos. Los métodos que sugiere Veenman son los que pueden ser directamente observables en el momento de la actividad metacognitiva; estos incluyen la observación de movimientos del ojo, valoraciones concurrentes mediante bitácoras digitales y técnicas de verbalización por parte de los sujetos.

Se han estudiado las maneras en las que mejor se puede evaluar la metacognición en alguna de sus facetas (Bannert y Mengelkamp, 2007; Block, 2005; Hashmi et al., 2019; Marion Händel et al., 2013; Saraç y Karakelle, 2012; M. Veenman, 2005; Yeh y Lo, 2005). Las mediciones de habilidades no parecen estar ligadas a un dominio o contexto sino que son propias del individuo (M. V. J. Veenman et al., 1997). Se han estudiado diversas técnicas para determinar sus virtudes y dificultades (Ericsson y Simon, 1998; Montague y Applegate, 2010; Sulcer et al., 2008). Por último, se han utilizado medidas de autorreporte, instrumentos de prueba y otros.

2.2.1 Medidas de auto reporte

Se han usado cuestionarios para evaluar metacognición, como medida de autorreporte, los cuales pueden ser generales o específicos a tareas o cuestionarios en los que las preguntas se relacionan más directamente a la tarea específica y no a lo que el individuo generalmente hace mostrando que el segundo tipo de cuestionario está más cercanamente relacionado a lo que realmente realizó (comparando con think aloud) (Schellings et al., 2013).

Las habilidades metacognitivas utilizadas para controlar la cognición no necesariamente son siempre evidentes, el sujeto no siempre es consciente de la naturaleza metacognitiva de sus acciones y esto limita la precisión en la medición por autorreporte (Reder y Schunn, 1996).

A lo largo del tiempo ha habido diferencias en cuanto a los instrumentos utilizados para el autorreporte, incluyendo cuestionarios y entrevistas, además de diferencias en la visión de metacognición utilizada (con dos factores (conocimiento y regulación de la cognición) o tres factores (agregando *responsiveness* o sensibilidad metacognitiva) (Craig et al., 2020). Estos autores, estudiando 22 artículos publicados entre los años 1993 y 2018 comprobaron el uso de cuestionarios de autorreporte incluyendo: *Metacognitive Awareness Inventory* (MAI) (G. Schraw y Dennison, 1994), *Motivated Strategies for Learning Questionnaire* (MSLQ) (Pintrich y Groot, 1990), *Metacognitive Skills Scales* (MSS)

(Altindağ y Senemoglu, 2013), *Awareness Independent Learning Inventory* (AILI) (Meijer et al., 2013) y *Metacognitive Self-Assesment Scale* (MSAS) (Pedone et al., 2017).

El MAI es un instrumento para la medición de metacognición mediante 52 preguntas (G. Schraw y Dennison, 1994). La intención del instrumento es, como herramienta de autorreporte, hacer mediciones con respecto a 8 componentes de la metacognición, agrupados en dos categorías principales de conocimiento de la cognición y regulación de la cognición.

Los componentes de conocimiento acerca de la cognición incluye conocimiento procedimental, condicional y declarativo. Esto abarca el conocimiento acerca de uno mismo y las estrategias que puede utilizar, cómo utilizarlas y en qué situaciones utilizarlas. Los componentes de regulación incluyen planificación, estrategias de manejo de información, evaluación del avance, estrategias de depuración y evaluación posterior acerca del rendimiento. Similarmente, pero con menor relación a metacognición y más a estrategias de aprendizaje en general, algunos de los ítems del instrumento conocido como MSLQ (Pintrich y Groot, 1990) se relacionan con la metacognición. También se han desarrollado otras técnicas enfocadas en el momento de resolución de un problema (Evita Kusumawati et al., 2021; Frenken, 2020).

Se han desarrollado otros instrumentos de autorreporte, por ejemplo, un cuestionario con el que los estudiantes indican, con escalas de Likert, su nivel percibido en distintas preguntas (G. Thomas et al., 2008), el cual fue desarrollado de manera teórica, consta de 72 elementos que posteriormente redujeron a 30 aplicando un análisis estadístico entre resultados de la aplicación del instrumento a 465 estudiantes en Hong Kong.

En este estudio, concluyeron que su instrumento es efectivo en la clasificación metacognitiva de individuos según sus habilidades. Con otro instrumento original, en línea, se ha medido el conocimiento metacognitivo (Yeh y Lo, 2005) para medir el nivel de conocimiento metacognitivo en estudiantes que utilizaban un sistema en línea (digital) en el aprendizaje. Los estudiantes eran adultos que debían navegar un contenido teórico organizado en una de tres posibles formas (jerárquicamente, linealmente y en forma de

red). Luego de medir el nivel metacognitivo con el instrumento MAI se dividió el grupo de 146 estudiantes universitarios en tres (según estructura del contenido) y se midió su comportamiento frente a la estructura tomando en cuenta su navegación hacia elementos de ayuda o entre las páginas visitadas y no visitadas. Por lo que, aunque reconocen limitaciones de este sistema encontraron que sí parece ser una herramienta valiosa para medir el nivel de conocimiento metacognitivo en general.

2.2.2 Instrumentos de prueba

El instrumento VisA (Jacobse y Harskamp, 2012) evalúa la precisión con la que estudiantes estiman la confianza para tener éxito previo a una tarea y luego al finalizarla. Esta es la estructura típica de los instrumentos de prueba para evaluar la metacognición. Por ejemplo, el Metacognitive Knowledge in Mathematics Questionnaire (MKMQ) pero diseñado para ser utilizado en concordancia con una prueba de matemática y valorar así las respuestas mediante una prueba (Efklides y Vlachopoulos, 2012). El instrumento de Knowledge Monitoring Ability (KMA) también compara los juicios acerca del rendimiento obtenido con los resultados obtenidos (Tobias y Everson, 2000).

Se ha utilizado la teoría de detección de señales (SDT por sus siglas en inglés) para medir la eficiencia con la que sujetos miden su rendimiento real realizando tareas (Maniscalco y Lau, 2012; Fleming, 2016; Fleming y Daw, 2016), es decir, la precisión con que sujetos estiman su correctitud en pruebas y la confianza con la que se toman decisiones. Este enfoque además, permite separar la medición propia de la metacognición del efecto de sesgo en las respuestas de un cuestionario de autorreporte (Maniscalco y Lau, 2012).

Ante esto, parece haber una tendencia hacia valoraciones de metacognición basadas en tareas específicas, y entonces sensibles a un dominio en particular, alejándose de evaluaciones de dominio general (Ozturk, 2017).

El mismo autor destacó que la mayoría de las investigaciones que revisó evaluaba al menos la regulación metacognitiva y en no todas evaluaban el conocimiento

metacognitivo. Otro tipo de instrumento de prueba es el que presenta al estudiante la tarea de identificar errores, bajo el supuesto de que los errores interrumpen la comprensión y los que más monitorean su comprensión los notan mejor (Baker y Cerro, 2000).

2.2.3 Métodos alternativos

Se han comparado múltiples métodos o tratado el tema de evaluación metacognitiva teóricamente (Anderson et al., 2008; Block, 2005; Schmitz et al., 2004; G. J. Schraw y Impara, J.C., 2000; Staats et al., 2003; G. Thomas et al., 2008; M. Veenman, 2005; M. Veenman et al., 2006; Wernke et al., 2011). Existen investigaciones que definen instrumentos para la valoración acerca del uso de un sujeto de habilidades metacognitivas. Por ejemplo, Choi (2006) quien define un instrumento en el que se basa en una escala de Likert para que el sujeto pueda describir su experiencia metacognitiva. Este instrumento se utiliza en el aula para ayudar a enseñar a los estudiantes a desarrollar su propia metacognición.

La técnica *think-aloud* (verbalización concurrente), consiste en que el sujeto expone su proceso mental mientras realiza una tarea, esta técnica ha sido utilizada pues permite no interrumpir la actividad en sí sino brindar un vistazo al proceso mental (Bannert y Mengelkamp, 2007; Davey, 1983; Ericsson, 1987; Ericsson y Simon, 1993, 1980, 1998; Montague y Applegate, 2010). Inicialmente se desarrolló como parte del análisis de protocolo utilizado por psicólogos para estudiar los procesos cognitivos de los sujetos. Similarmente, se ha utilizado esta técnica para observar la capacidad de resolución de problemas en un marco cognitivo-metacognitivo.

Ericsson y Simon (1993) describen verbalización en general y el análisis de protocolo necesario para extraer datos que se puedan analizar, aportan un fundamento teórico a la fuente de información de verbalización en general y think aloud en particular que describen como una forma de verbalización concurrente a la tarea que se realiza (1993). Por otro lado, sitúan su explicación en el modelo cognitivo del funcionamiento de la memoria (memoria de registro sensorial, memoria de corto plazo y memoria de largo

plazo). Según estos autores, la verbalización puede suceder de tres maneras distintas. En un primer nivel de verbalización, el sujeto explica oralmente sin mayor tiempo de procesamiento adicional y en forma de narración lo que ejecuta por lo que su proceso cognitivo y actividad no sufren interrupciones. En un segundo nivel de verbalización el sujeto trae, posiblemente de memoria de largo plazo a memoria de corto plazo conocimientos para su verbalización. Esto requiere procesamiento adicional, que es visible en pausas al hablar, pero no se interrumpe el proceso cognitivo que realiza el sujeto. Un tercer nivel de verbalización requiere de explicaciones más profundas sobre los procesos mentales por lo que el uso de la memoria de largo plazo es más intenso y va acompañado de procesos cognitivos más complejos (introspección). Este tipo de verbalización sí requiere interrumpir y posiblemente variar el proceso cognitivo que desarrolla el sujeto.

El segundo nivel de verbalización de Ericsson y Simon (1993) es propiamente el conocido como *think-alouds.* En este nivel de verbalización el sujeto explica lo que piensa, con uso ocasional de su memoria de largo plazo pero sin interrumpir considerablemente su actividad cognitiva actual. La verbalización de este nivel requiere mayor procesamiento pero sí refleja fielmente el proceso cognitivo que se está ejecutando sin mayor interferencia.

Una actividad que ha sido evaluada metacognitivamente ha sido el comportamiento de búsqueda de información o ayuda en el momento en la que se requiere, que es un aspecto específico del aprendizaje auto regulado. Además, se ha hecho con un método de medición en un ambiente digital en el que se da una tutoría a estudiantes. Mediante el registro automático de las acciones del usuario del sistema se ha estudiado el comportamiento que exhiben, específicamente, en la búsqueda de ayuda (Aleven et al., 2010). En comparación con otros métodos resultó que la medición de esta estrategia específica de la autorregulación es similar. Estudiaron también, cómo ofrecer, en el sistema, la tutoría digital de búsqueda de información. Por otro lado, se comprobó que los estudiantes mostraron mejoría en cuanto a destrezas metacognitivas como la búsqueda de

información pero también que la mejora en el conocimiento del dominio no sufrió cambio significativo.

La evaluación de la metacognición también se puede clasificar por métodos en línea y fuera de línea (esto se refiere a si el método mide en el momento de la actividad o no). Se han comparado los resultados obtenidos de mediciones en línea y fuera de línea (Saraç y Karakelle, 2012), en una población de 47 estudiantes de quinto grado de escuela midieron metacognición utilizando cuatro métodos distintos. Dos de los métodos utilizados se consideran como en línea y dos fuera de línea. Los dos métodos en línea fueron *think-alouds* y una escala de comprensión de lectura, y los dos métodos fuera de línea fueron un cuestionario posterior y una escala de apreciación del docente. Luego de un análisis comparativo entre métodos en línea y fuera de línea determinaron que la medición de la metacognición puede realizarse con ambos métodos pues parecen estar midiendo estructuras internas independientes, sugieren, además, que los métodos en línea, como *think-alouds*, de medición están más ligados a aspectos internos no tan evidentes mientras que los métodos fuera de línea, como los cuestionarios, pueden estar más relacionados a los aspectos más evidentes y visibles de la metacognición (Saraç y Karakelle, 2012).

Además concluyeron que una medición que mezcla métodos y que sean de tipo distinto arrojaría los mejores resultados, por ejemplo, la utilización de un método en línea con un método fuera de línea como un cuestionario posterior.

2.3 Programación de computadoras

Los artículos empíricos presentados en el Simposio Técnico acerca de Educación en Ciencias de la Computación (SIGCSE), relacionados al primer curso de programación (CS1), en sus primeros 50 años pueden clasificarse, según su enfoque principal, en enseñanza, estudiantes, primeros lenguajes y paradigmas de programación, aprendizaje y evaluación, diseño y contenido del primer curso, herramientas y enfoques colaborativos (Becker y Quille, 2019). Señalan algunas tendencias de investigación, por ejemplo, que las

investigaciones predominantes son las enfocadas en estudiantes, su aprendizaje y, en tercer lugar, su enseñanza. Una observación similar hacen Luxton-Reilly et al, que además señalan que en la evaluación incluyen investigaciones sobre la posible retroalimentación que se le brinda al estudiante (2018). Los mismos autores señalan que las tendencias han ido en aumento para el caso de desarrollo de herramientas, aunque se ha reportado más el éxito de la herramienta más que las implicaciones en el rendimiento o las causas del éxito, y la estabilidad en otras áreas más teóricas acerca del aprendizaje.

Además, señalan que algunas investigaciones se han concentrado, por ejemplo, en estilos de aprendizaje aunque esto ya se ha desacreditado. Otros aportes investigativos que aportan claridad al tema de la educación para programar recalcan las diferencias entre expertos y no expertos, tanto en conocimiento como en cantidad de lenguajes que terminan conociendo (Robins et al., 2003).

Para estudiar el aprendizaje de la programación Blikstein (2011) desarrolló un ambiente de programación para registrar acciones del usuario a nivel de teclazos y movimientos de mouse y guardar bitácoras de las acciones con el objetivo de permitir la exploración manual del código y estudiar el posible significado de los cambios en el código respecto a la actividad principal del estudiante.

Los cambios de códigos los midieron respecto a la cantidad de caracteres presentes y las acciones dependiendo de si estaba editando código actual, agregando código nuevo o haciendo una operación de movimiento de código mediante copiar y pegar en el mismo programa. No se hizo análisis sintáctico de los cambios realizados en el código. El objetivo de este enfoque fue identificar los momentos principales de creación de un programa. Se analizó el código de un ejercicio en 9 estudiantes. En cuatro de ellos, por ejemplo, se identificó un momento inicial de sentar las bases del programa, seguido de varios momentos de navegación en el código, ráfagas de producción en los que la cantidad de caracteres aumenta considerablemente y una etapa final en la que el estudiante se dedica a embellecer y acomodar su código.

Algunas revisiones de literatura sirven de guía para comprender cómo se ha estudiado el aprendizaje de programación de computadoras (Carbone y Kaasbøll, 1998; Costelloe, 2004; Pears et al., 2007; Robins et al., 2003; Simon et al., 2008). Además, se ha estudiado la metacognición y la cognición en relación con la programación (Aharoni, 2000; Ginat, 2001; Prather et al., 2020).

2.3.1 Aprendizaje de programación

Registrando comportamientos de programación, en una aplicación para realizar ejercicios programados, se detectaron estudiantes en riesgo de fracasar en un curso de programación (Estey y Coady, 2016). Se observaron comportamientos de compilación y consulta de ayudas, que denominaron como de alto nivel. Es decir, el sistema registraba las veces que el usuario compilaba el código y las veces que hacía uso de una facilidad de la aplicación que brinda información adicional. Encontraron que bajas tasas de compilación y alto uso de ayudas son comportamientos asociados con estudiantes en riesgo de reprobar. Además demostraron que intentar más ejercicios y compilar más veces se relaciona con el éxito en el curso. En una siguiente investigación (Estey et al., 2017) estudiaron los cambios de comportamiento a lo largo de un semestre. Encontraron que algunos comportamientos de riesgo se dan durante el semestre aún en estudiantes que terminan exitosamente y que si se toma en cuenta la trayectoria de comportamientos en el semestre se fortalece la capacidad de predicción de su modelo.

Gross y Powers (2005), concentrándose en el uso de ambientes gráficos, llevaron a cabo un estudio sobre el estado de investigaciones alrededor de la educación en programación. Los autores clasificaron los documentos estudiados según el enfoque metodológico en una de tres categorías: empíricos, analíticos y anecdóticos. Construyeron una manera de describirlos según algunos aspectos como la generalización posible y el planteamiento del problema que se puede aplicar para tener una idea de la calidad de las evaluaciones de los ambientes.

Existe una revisión de literatura llevada a cabo por Sheard et al (Sheard et al., 2009) sobre reportes de investigaciones empíricas realizadas entre los años de 2005 y 2008 y tomando como fuentes los documentos entregados a un conjunto de conferencias centradas sobre el tema de educación en computación. Ilustrando qué se ha hecho en investigaciones de campo y ante esto observaron que la gran mayoría de las investigaciones realizadas fueron de orden cuantitativo, trataban sobre un solo curso, y estudiaban la habilidad o entendimiento alcanzado por los estudiantes.

Luego de etiquetar el uso de métodos cualitativos como informal o anecdótico y de métodos cuantitativos como interesados únicamente en el qué más que en el por qué, estos autores aclaran como su revisión se concentró en la búsqueda de rigor en los procesos investigativos analizados. En ese mismo estudio (2009) se encontró que la mayoría de estudios cualitativos correspondían a estudios sobre teoría o modelos de aprendizaje y los cuantitativos corresponden más a métodos y herramientas de evaluación.

Pears et al (2007) organizaron la literatura relacionada a la educación en programación en cuatro categorías con respecto a su temática principal: currículo, pedagogía, lenguaje a utilizar y herramientas. Encontraron más fuentes en la categoría de herramientas de apoyo, incluyendo software y robots, para enseñar a programar. También argumentan que no se ha podido llegar a la práctica docente en el aula por falta de organización y por enfocarse únicamente en un aspecto reducido como una herramienta o un lenguaje pues, según ellos, se necesita más investigación central al proceso de aprendizaje de la programación.

En computación una gran cantidad de investigadores se basan en información anecdótica sobre las experiencias de los docentes y se ha propuesto que la validez y generalización pueden beneficiarse con la utilización de múltiples fuentes, como cuestionarios, entrevistas, observaciones en la misma situación o con la repetición a lo largo de múltiples ciclos para apoyar innovaciones en el aula (Carbone y Kaasbøll, 1998). Algunos autores se concentran en la diferencia entre programadores expertos y principiantes y entre principiantes eficientes e ineficientes (Robins et al., 2003). En este

último señalan algunas implicaciones prácticas para el docente basándose en la literatura encontrada y en los comportamientos típicos de principiantes eficientes e ineficientes, señalando incluso que las estrategias preexistentes de los novatos es lo que principalmente diferencia entre los aprendices efectivos y los inefectivos (2003).

Basándose en el triángulo didáctico (Kansanen, 1999) Berglund y Lister argumentan que muchos estudios dejan de lado las relaciones entre docente, estudiante y contenido, enfocándose más en algunas de éstas relaciones o solamente en los ejes del triángulo (2010). Sugieren una línea investigativa que explore y desarrolle las maneras en las que los docentes puedan conocer mejor a los estudiantes, por ejemplo. Bennedsen y Caspersen (2008) encontraron que no necesariamente hay una correlación entre la capacidad de abstracción de un estudiante y su capacidad de programar aunque, en conflicto a esto, estudiando a estudiantes más avanzados de programación, se ha mostrado también que los estudiantes más exitosos tenían mayor probabilidad de tener buena habilidad para abstraer conceptos (Westbrook, 2005).

Algunos estudios tratan la enseñanza de la programación en general desde una perspectiva más general, por ejemplo atendiendo causas del éxito o fracaso en general (Cook, 2008; Hagan y Markham, 2000; Kurtz, 1980; Lister et al., 2009; Schulte y Bennedsen, 2006; Shackelford y Badre, 1993; Sheard et al., 2009; Stein, 1998; Wolfe, 2004).

Hagan y Markham (2000) señalan que los estudiantes con un conocimiento previo de algún lenguaje de programación tienden a tener mejor rendimiento. La capacidad de rastrear el código también ha sido relacionada con el mayor éxito programando (Lister et al., 2009).

Se han comparado los modelos de enfocarse en la resolución de problemas o en aspectos de sintáctica y semántica, no encontraron distinción pero sí que es más influyente la retroalimentación constructiva en lugar de estar basada en los errores (Shackelford y Badre, 1993). Malik introduce una estrategia llamada ADRI que, junto a un editor, promueve la combinación de ambos modelos (Malik, 2016; Malik y Coldwell-Neilson,

2017). Enfocándose en la opinión de los docentes acerca de lo que consideran más importante y contrastando con lo que realmente enseñan se ha encontrado que la docencia ha girado principalmente en aspectos concretos de programación en lugar de aspectos más abstractos (Schulte y Bennedsen, 2006).

Otros estudios tratan un enfoque más específico, por ejemplo, estudiando patrones de diseño en el aprendizaje (Porter, 2006; Porter y Calder, 2004) o algún aspecto específico de la programación (Christensen y Caspersen, 2002), promoviendo comunidades en línea (Henriksen et al., 2010), métodos de evaluación no tradicionales como mapas conceptuales (Keppens y Hay, 2008), programación en pares (Nagappan et al., 2003) o algún aspecto novedoso como programación gráfica (Calloni et al., 1997; Guibert et al., 2005; Westphal et al., 2003; Wolz, 2001; Wolz et al., 2009).

Algunos estudios sobre el rendimiento en cursos introductorios de programación han dado cierta luz al tema sobre qué se requiere para que estudiantes mejoren calificaciones. Se han identificado algunos factores determinantes como llevar un curso previo de computación (Bergin y Reilly, 2005; Morrison y Newman, 2001; Quille y Bergin, 2018) y la cantidad de lenguajes de programación conocidos (Hagan y Markham, 2000). Kurtz identificó la relación entre niveles de desarrollo intelectual con el buen rendimiento en exámenes aunque no tanto en programas (Kurtz, 1980). También otros como la nota promedio en cursos del área (Corman, 1986), estilos de aprendizaje preferidos (L. Thomas et al., 2002), o alguna combinación de factores como formación matemática con expectativa de nota (Rountree et al., 2004).

Con respecto a otros factores se ha encontrado cierto conflicto, pues factores como experiencia previa en programación o conocimiento de otros lenguajes no se asocia tanto a rendimiento (Byrne y Lyons, 2001), pues para estos autores la habilidad matemática y científica fueron predictores del éxito. Esto contradice el estudio de Hagan y Markam (2000).

Se ha estudiado también el efecto LEM, *Learning Edge Momentum* (Ahadi y Lister, 2013; Robins, 2010), para explicar los resultados en rendimiento en el curso introductorio a

programación no separando poblaciones entre aquellos con experiencia previa y sin experiencia previa, o expertos y no expertos sino más bien explicando el fenómeno como el resultado de tener éxito luego de una secuencia de pequeños éxitos entre temas muy conceptualmente relacionados que se han ido adquiriendo progresivamente.

Blackbox es una iniciativa de los desarrolladores de BlueJ (un software para desarrollo de código) (N. C. C. Brown et al., 2014, 2018) que almacena las acciones de programadores en su editor de texto. Mediante la participación miles de usuarios han podido recoger información de programación con más de 30 millones de sesiones observadas. La base de datos de Blackbox no está libremente disponible sino que es controlada bajo invitación a participar en investigación. Algunas investigaciones posibles tienen que ver con el comportamiento de depuración de los programadores o de cambios típicos en el código. Los usuarios típicamente son principiantes puesto que el ambiente BlueJ es diseñado para el uso de aprendices de programación. BlueJ presenta guías visuales y herramientas para instanciación de clases que típicamente no están disponibles en herramientas de uso general, no dirigidas a principiantes.

Los usos típicos de Blackbox reportados por Brown et al. (N. C. C. Brown et al., 2018) se basan en una teoría previa, intentan construir teoría a partir de los datos o hacen un estudio exploratorio de la base de datos sin proponer ningún modelo explicativo. Reconocen que la base de datos es muy grande y compleja al punto que es difícil realizar indagaciones a la base de datos completa.

Jadud (2006) utiliza datos obtenidos de BlueJ para explorar el comportamiento de compilación de los programadores y desarrolla lo que llama coeficiente de error (EQ) para obtener información del comportamiento de compilación de los programadores. Jadud y Dorn (2015) replican los resultados de Jadud pero utilizando los datos de Blackbox.

Además hacen un estudio de otras 18 publicaciones que usan también este juego de datos, y mencionan ventajas y dificultades. También, indican por ejemplo, que el tamaño de los datos es una limitante fuerte, y que hay problemas de investigación que se pueden tratar gracias a los datos recolectados.

Los investigadores Allan y Kolesar (1996) encontraron que desarrollar las habilidades de los estudiantes para resolver problemas influye más en su rendimiento que en duplicar los cursos de programación. En un estudio en el que se diseñó e impartió un curso de desarrollo de habilidades para resolver problemas se logró demostrar que el rendimiento académico de los estudiantes que llevaron este curso fue mayor que el rendimiento de estudiantes que llevaron un curso previo de programación.

Un enfoque de desarrollo de una manera de predecir el rendimiento fue desarrollado por Watson et al. (2013) para asignar un puntaje al tratamiento de los estudiantes al código en el momento de presentar un error. En un estudio en el que utilizaron el código fuente en intervalos de tiempo, llegaron a concluir que el comportamiento de los estudiantes ante los errores era una buena manera de predecir el rendimiento académico del curso. Midieron el rendimiento académico como una nota asignada a todas las asignaciones del curso, no solo al examen final. Además, a diferencia de esta investigación registraron el código fuente únicamente en los momentos de compilación y no hicieron observaciones a nivel de acciones.

En Nigeria se estudió el fracaso de los estudiantes de los cursos de programación (Dasuki y Quaye, 2016) identificando factores no necesariamente aplicables en otros contextos como la falta de motivación y expectativa a futuro, ansiedad, influencias de los pares y hasta mala preparación o conducta de los instructores. En el Reino Unido, (Murphy et al., 2017) se compararon los primeros cursos de programación, 80 en total, a nivel universitario encontrando que una gran mayoría utilizaban Java a pesar de que reportaban mayor facilidad de uso e instrucción del lenguaje Python. Este último y C# también son reportados como los más utilizados. El paradigma más ampliamente reportado fue el de Orientación a Objetos seguido de Procedimental. Eclipse fue la herramienta utilizada en el curso más ampliamente mencionada. En Australia (Mason et al., 2012) se desarrolló un estudio de 44 cursos en 28 universidades para determinar datos muy similares respecto a los lenguajes utilizados, herramientas y paradigmas.

2.3.2 Metacognición y programación

Analizando los modelos mentales que estudiantes de programación utilizan para comprender el problema que están programando Ma et. al (Ma et al., 2007, 2009) confirman que existe una gran dificultad para visualizar los conceptos fundamentales de programación. Proponen una estrategia de enseñanza basada en conflictos cognitivos que les permita mejorar su monitoreo y reconocer más pronto errores en su comprensión.

Un estudio que se enfoca en la evaluación de la metacognición y en los posibles beneficios de la misma para la programación es realizado por Bergin et al. (2005). Los autores llegaron a la conclusión que un estudiante con buenas estrategias metacognitivas y de manejo de recursos tiende a tener mejor rendimiento aprendiendo a programar. Para esto utilizaron un instrumento que mide las estrategias cognitivas, metacognitivas y de manejo de recursos. Al aplicar el instrumento a los estudiantes obtuvieron que un modelo de regresión basado en estas estrategias fue suficiente para explicar el 45% de la varianza en la habilidad de programación.

Butler y Morgan (2007) estudiaron la percepción de dificultad de cientos de estudiantes sobre conceptos de programación como algoritmos, orientación a objetos y estructuras de datos. Lo realizaron con un instrumento en el que los estudiantes asignaron una dificultad en una escala de 1 al 7 a los conceptos de programación seleccionados por ellos y obtuvieron como resultado que la orientación a objetos es lo más difícil según la percepción de los estudiantes.

Existen teorías que buscan modelar el aprendizaje humano modelando sistemas procesadores de información automáticos que construyen un modelo de procesamiento de información para intentar explicar cómo el cerebro aprende. Estas teorías pueden adoptarse para estudiar la manera de aprender computación, como sugiere Bower (2004). Estas teorías incluyen de Código Dual, de Niveles de Procesamiento, Procesamiento Paralelo Distribuido y Procesadores de Capacidad Limitada.

Con 20 estudiantes sin experiencia previa en programación, Fay y Mayer (1994) lograron demostrar que el desarrollar habilidades de planificación y estrategia antes de

enseñar sintaxis y semántica de un lenguaje, lo cual condujo a resultados favorables. Específicamente se enseñaron los conceptos de modularización para permitir la reutilización posterior. Estas habilidades son de naturaleza metacognitiva que debe desarrollarse con una mayor población para verificarse.

En un estudio sistemático de literatura (Prather et al., 2020) los autores recogen qué teorías han sido utilizadas específicamente en la educación de la programación relacionadas con la metacognición y la autorregulación. Además enumeran los instrumentos utilizados y las actividades desarrolladas para observar las evidencias de metacognición.

Finalmente, llegan a varias conclusiones que incluyen el uso poco preciso de los términos y la inclusión en estudios de múltiples aspectos por lo que sugieren definir claramente qué aspecto de la metacognición se va a explorar, como el de planificación. Además sugieren tener particular rigor con el momento de la observación pues señalan que mediciones posteriores a una actividad no brindan la misma información que observaciones durante una actividad. Es decir, no siempre lo que el participante reporta posteriormente es lo que realmente realizó. Entre los instrumentos reportados encontraron la dificultad de contar con instrumentos específicos pues cada aplicación de instrumentos parece ofrecer propias definiciones de su aplicación. Como el instrumento más común encontraron los cuestionarios, especialmente el Motivated Strategies for Learning Questionnaire (MSLQ) que es de autorreporte.

Ginat (2001) explica de manera anecdótica, mediante una investigación en la que se entrenó a 23 estudiantes avanzados a participar en una competencia de programación, cómo el desarrollar habilidades de auto-explicación estimula el control que mantiene el estudiante sobre el proceso de resolución programada de un problema. El trabajo de Ginat luego (2002) identifica habilidades metacognitivas importantes para la resolución de problemas y estrategias de resolución de problemas aunque no de manera empírica.

Aharoni (2000) señala que la investigación sobre procesos cognitivos en la educación de computación está poco desarrollada. Este investigador utilizó entrevistas

semiestructuradas para obtener información de estudiantes principiantes de un curso de estructuras de datos (relacionado a programación). Según la información obtenida de los sujetos, se construyó un modelo utilizado por científicos cognitivos para explicar cómo las personas construyen y manipulan conceptos.

Porter y Calder (2004) asociaron técnicas para la resolución programada de problemas con las categorías de dominio cognitivo de Bloom (1956). Esto lo hicieron para probar un modelo de enseñanza de resolución de problemas basados en la búsqueda de un patrón de programación apropiado para un problema dado. Sus pruebas no arrojaron los resultados que esperaban. El principal problema que describen es que los estudiantes no reportaron la información según las categorías identificadas. No es claro si esto fue así porque ya habían superado su nivel cognitivo inicial por lo que sus descripciones ya no calzaban o porque el modelo era demasiado simplista para intentar explicar el proceso de resolución de problemas mediante programación.

Algunos autores se basan en la perspectiva metacognitiva para ver el problema de aprender o enseñar a programar (Bergin et al., 2005; Boyer et al., 2008; Butler y Morgan, 2007; Hull y Du Boulay, 2009; Kapa, 2001).

Bergin et al (2005) midieron la motivación y el uso de estrategias de aprendizaje (cognitivas, metacognitivas y de utilización de recursos) en estudiantes graduados de colegio para relacionar esa información con su rendimiento en programación. Encontraron que a mayor nivel de motivación se obtenía mejor rendimiento y mayor uso de estrategias metacognitivas (y cognitivas).

Además encontraron que a mayor rendimiento mayor uso de estrategias metacognitivas como planificación, monitoreo y regulación. Utilizaron un instrumento para medir los tres aspectos en que se interesaron y su análisis fue estadístico. Existe un riesgo a la validez del uso de cuestionarios como este señalado por múltiples autores (Bergin et al., 2005; Leppänen et al., 2016) y tiene que ver con la naturaleza posterior y de autorreporte del cuestionario.

En un estudio sobre autorregulación y aprender a programar (Leppänen et al., 2016) se concentró en las pausas de los programadores durante su actividad. Encontraron que pausas frecuentes, de tan solo unos segundos resultaron correlacionarse con los resultados positivos en el puntaje obtenido pero pausas de minutos correlacionaron negativamente. Sin embargo, no se pudo determinar la relación entre el estilo de pausas de los sujetos con la medición metacognitiva utilizada.

Butler y Morgan (2007) estudiaron las dificultades en general que enfrentan los principiantes en programación con un cuestionario. Pero su enfoque principal se basó en el conocimiento metacognitivo de los estudiantes preguntándoles sobre su dominio conceptual y capacidad de implementar conceptos que definieron como de bajo nivel y de alto nivel.

Además, se ha mostrado que la aplicación la técnica de think-aloud, mencionada previamente, produjo resultados favorables en cuanto a la capacidad de resolución de problemas de los estudiantes, su capacidad de programar con un estilo adecuado y su capacidad de modularizar el problema favoreciendo la reutilización de código y recortando así los tiempos de programación de nuevas soluciones (Arshad, 2009).

Capítulo 3 – Metodología

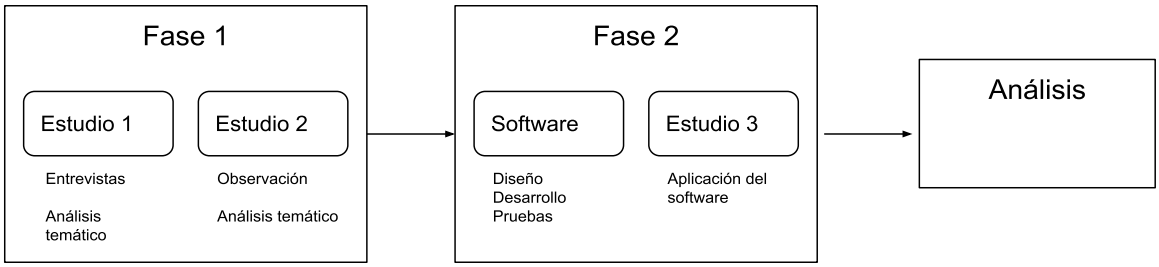
Se desarrollaron tres estudios, secuencialmente, que reunieron las tareas para: identificar acciones indicativas de habilidades metacognitivas, identificar otros requerimientos para diseñar el software de medición (Estudio 1 y Estudio 2), creación del instrumento (desarrollo de software) y su aplicación (Estudio 3). Se puede considerar que los estudios 1 y 2 constituyeron una primera fase y el estudio 3, una segunda fase. Las dos fases se articulan por medio de un punto de integración, que fue la creación del software, según los resultados de los estudios 1 y 2 para la recolección de información del estudio 3.

Se siguió un diseño secuencial exploratorio, un método mixto, en los estudios 1 y 2 se recogió información para análisis cualitativo, mediante entrevistas y observación, con el fin de desarrollar un instrumento (software), para recolectar información de estudiantes en el mismo momento que programan (medición en línea).

Este software se aplicó en el Estudio 3. El diseño metodológico se muestra en la Figura 3.1. A partir de lo anterior, es primordial recalcar que, al ser un diseño secuencial, cada fase informa y fundamenta la posterior. Esto es, por ejemplo que, los resultados de la primera fase se utilizaron para desarrollar la segunda fase.

Figura 3.1

Diseño metodológico multiestudio y las tareas principales



3.1 Estudio 1

3.1.1 Estudio 1 - Descripción

Basado en las destrezas metacognitivas de planificación, organización y regulación, el propósito de este estudio fue identificar acciones observables en estudiantes que aprenden a programar indicativas de esas destrezas metacognitivas y la forma de hacerlo. El resultado de este estudio se usó para el diseño de software y se trató de un estudio cualitativo exploratorio, de tipo genérico o básico (Merriam y Tisdell, 2016).

En este estudio se recogió información de participantes expertos en programación y se realizó un análisis temático, este concluyó en la clasificación de la información, una codificación temática, mediante el uso de etiquetas, en la forma de temas y códigos. Como producto directo del análisis temático se obtuvo el resultado esperado de este estudio, un listado de acciones a observar en el estudiante, junto a opiniones expertas sobre cómo se pueden asociar esas acciones a las destrezas metacognitivas de planificación, organización y regulación. También, se obtuvieron también requerimientos del software, detalles de funcionalidades que debe tener.

3.1.2 Estudio 1 - Participantes

En este estudio participaron expertos en programación. La selección de expertos respondió al propósito de obtener información específica en el ámbito de programación buscando experiencia y conocimiento adquirido con el tratamiento directo en la educación de la programación.

Los expertos seleccionados para las entrevistas incluyeron un docente/investigador de amplia experiencia (experto principal) y docentes experimentados de programación, asistentes de curso, investigadores y estudiantes avanzados. Según el tipo de participante se trataron distintos temas. Con todos los participantes se realizó una introducción para contextualizar la investigación y tratar las destrezas metacognitivas de planificación, organización y regulación como temas de fondo principal. En la Tabla 3.1 se presentan los

tipos de expertos participantes y otros temas tratados en las entrevistas. En este proceso se buscó y se concluyó cuando se alcanzó saturación de los datos, es decir, un punto donde las entrevistas ya no aportaron datos nuevos sobre acciones ni maneras de interpretarlas.

Tabla 3.1

Descripción de los principales temas tratados en entrevistas con expertos

Tipo de Experto	Principales temas tratados	Cantidad de Entrevistas
Experto principal	Acciones indicativas de destrezas metacognitivas, operación general del software, interpretación de acciones registradas como señal de destrezas metacognitivas (reglas de producción).	15
Docentes de Programación	Acciones del programador, efectos en el código, habilidades metacognitivas requeridas, reacción a los errores (en relación a la regulación) y avance en aprendizaje.	2
Investigadores	Ejercicios que requieran mayor involucramiento metacognitivo	2
Asistentes de Curso de Programación	Principales errores y la destreza metacognitiva de regulación	2
Estudiantes avanzados	Principales errores y la destreza metacognitiva de regulación	2

3.1.3 Estudio 1 - Instrumentos

En este estudio el principal instrumento fue la entrevista semi estructurada con una guía de preguntas predefinida. Estas entrevistas se desarrollaron con carácter exploratorio

y como mecanismo de manejo de los datos recolectados se utilizaron apuntes que posteriormente, estuvieron sujetos a análisis temático.

Cada entrevista inició con una introducción para contextualizar la investigación y abordar los temas de destrezas metacognitivas de planificación, organización y regulación como temas de fondo.

Estas destrezas se definieron para guiar la conservación en ese sentido. Las entrevistas siguieron con preguntas generales, que luego se desarrollaron a profundidad para luego cerrar con preguntas más puntuales que recogieran algún dato adicional.

3.1.4 Estudio 1 - Procedimiento y estrategia de análisis

En el estudio 1 se realizaron 24 entrevistas semi estructuradas a distintos expertos. Se eligió este tipo de entrevista por la naturaleza exploratoria de este estudio y para permitir a los entrevistados entrar en profundidad y explicar sus respuestas (Bryman, 2012; Robson y McCartan, 2017).

Cada entrevista inició con una introducción que cubrió los objetivos de la investigación y la descripción de las destrezas metacognitivas mencionadas. Para cada acción identificada se desarrolló el tema de su posible interpretación en relación con una de las destrezas metacognitivas.

Las entrevistas se realizaron de manera formal, con una guía de preguntas y en espacio privado. Para analizarlas, se construyó una lista de apuntes, obtenidos de las respuestas de los entrevistados. Se utilizaron apuntes de las entrevistas debido a la no anuencia de algunos de los participantes a ser grabados. A estos apuntes se les realizó un análisis temático, que implica el manejo lógico, rastreable y documentado, para sistematizar el manejo de la información obtenida y aumentar así la confiabilidad (Tobin y Begley, 2004).

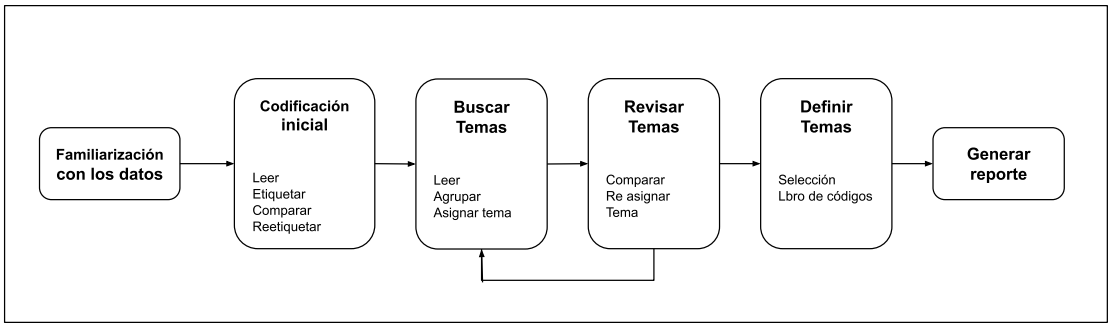
Las tareas realizadas para este estudio fueron: redacción de preguntas guía para las entrevistas (ver apéndice A), selección de participantes, realización de las entrevistas, toma

de apuntes y, para análisis temático, la organización de datos, selección de códigos y temas y generación de reportes de distribución de los datos por códigos y temas (codificación o clasificación). El tratamiento a la teoría es más inductivo en el sentido de que se empezó con los datos en busca de significados con cierta flexibilidad de interpretación.

El tratamiento posterior a la captación de datos fue mediante análisis temático, como se mencionó. Este es un proceso iterativo que incluye la familiarización con los datos, la codificación en sí, la generación y revisión de temas, y la construcción en paralelo del libro de códigos que describe y distingue entre códigos y temas (Alhojailan, 2012; Boyatzis, 1998; Braun y Clarke, 2006, 2019). Este proceso se ilustra en la Figura 3.2. La codificación se realizó tanto con temas predefinidos como con temas emergentes. Esto debido a que se buscó determinar la opinión de expertos acerca de acciones indicativas a destrezas metacognitivas, lo que permitió hacer una codificación basada en los temas de planificación, organización y regulación; y una codificación emergente según, las acciones mismas, que sirvió, en segundo plano, de base propiamente para el diseño del software.

Figura 3.2

Proceso de análisis temático



El resultado principal de este estudio es una clasificación de opiniones de los expertos sobre las destrezas metacognitivas y la construcción de reglas de producción para representar este conocimiento experto de relación entre destrezas metacognitivas y acciones de programadores. También se obtuvo otra clasificación que fundamenta el diseño del software necesario para el Estudio 3 y esta incluye la organización de acciones por efecto sintáctico y semántico y también por requerimientos de software (de captura o de análisis, ver capítulo de resultados).

3.2 Estudio 2

3.2.1 Estudio 2 - Descripción

En el estudio 2 se realizó la observación de estudiantes programando un ejercicio sencillo en lenguaje Java. El propósito fue la identificación de acciones observables en programadores, y la clasificación de las mismas como indicativas de alguna de las destrezas metacognitivas elegidas para la investigación. Específicamente se definió la meta de etiquetar las acciones, de los que aprenden a programar, según una destreza metacognitiva. Adicionalmente, se corroboraron las acciones identificadas en el Estudio 1 y se buscó alguna acción no identificada aún. Por último, se puede mencionar, como propósito adicional, el recoger información de insumo para el diseño del software para el siguiente estudio (Estudio 3).

Los datos necesarios para este estudio provienen del registro de la interacción misma del estudiante programador con la máquina. En el proceso se observaron las acciones de estudiantes de programación, en un ambiente físico de laboratorio de computación, resolviendo un ejercicio sencillo de programación. Las acciones tomadas por los estudiantes fueron registradas, automáticamente en detalle. Además, el análisis temático se realizó en la forma de apuntes y fragmentos de código.

3.2.2 Estudio 2 - Participantes

Para este estudio se realizó observación de estudiantes de computación, en un primer curso de programación, en la creación y modificación de un ejercicio sencillo de programación. Se observaron a 10 estudiantes que se ofrecieron voluntariamente a participar. Los estudiantes estaban en su primer curso de programación, en la UCR, con edades entre 19 y 20 años.

3.2.3 Estudio 2 - Instrumentos

En este estudio el principal instrumento es el software de registro de acciones en bitácoras y el software para la visualización de bitácoras. Otros instrumentos secundarios son las computadoras utilizadas y el software de visualización de pantallas remotas.

3.2.4 Estudio 2 - Procedimiento y estrategia de análisis

El estudio 2 recolectó datos de manera automática mediante un software básico (capturador de acciones) que se desarrolló de manera inicial con esta única intención para utilizarse como instrumento de observación. Este software permitió llevar una bitácora de las acciones del estudiante en el computador mientras resolvía un problema de programación. Sus acciones del teclado para ingreso de símbolos al código y del control de cursor de la computadora (uso de flechas) son ejemplos de acciones que se registraron. Adicionalmente, la bitácora de estas acciones se usó como fuente de información que se estudió para encontrar acciones compuestas de múltiples acciones. Esto es, por ejemplo, múltiples teclazos que se unen para formar una única acción sintáctica o semántica como crear una variable o modificar una condición de un ciclo (crear la variable x ingresando “int x;” es una acción compuesta por múltiples acciones como ingresar una “i”, luego una “n”, luego una “t”, etc.).

Los participantes fueron observados en un espacio designado como laboratorio de computación, en el que se usaron 10 computadoras para que los participantes programaran

y una con acceso directo y simultáneo a las otras pantallas. Para la observación, luego de una breve introducción a la investigación y obtener su consentimiento informado a participar, se les informó del hecho de que serían observados física y digitalmente además de que sus acciones de teclado quedarían registradas. Luego se les solicitó a los participantes resolver un ejercicio independientemente, en sus computadoras. La duración de la sesión de observación fue, en todos los casos, menor a 45 minutos.

Los datos obtenidos de la observación incluyeron, los códigos finales, las acciones registradas en bitácoras y apuntes tomados en el momento de la programación y posteriormente, con la lectura de las bitácoras. La lectura de las bitácoras constituyó la segunda tarea de familiarización con los datos, la primera fue en la observación en vivo. Esto sirvió de base para la codificación y generación de temas, que a su vez sirvió de base para la interpretación de resultados.

Con el análisis temático se generaron dos codificaciones, una con códigos predefinidos y otra de codificación abierta. La primera tuvo el propósito específico de clasificar las acciones, según destrezas metacognitivas, siguiendo la teoría para el tipo de tareas realizadas a las que corresponden las acciones. Para esta codificación se definió de previo el libro de códigos con los criterios de asignación de código para los temas de planificación, organización y regulación. La segunda codificación fue abierta y el libro de códigos, junto con los códigos y los criterios de aplicación de los códigos, surgieron de la clasificación de los apuntes.

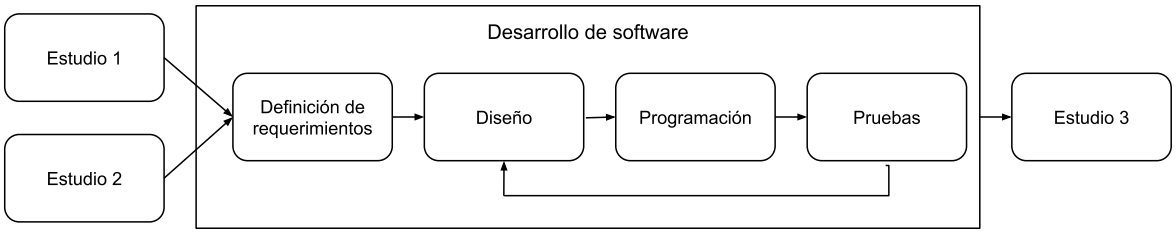
En ambos casos, el análisis temático, luego de la observación se basó en apuntes de observaciones, en lugar de en las acciones a bajo nivel registradas en bitácoras, por el hecho de que las acciones a ese nivel están más alejadas del propósito del estudiante. Por lo que, se consideró más conveniente analizar las acciones de un nivel más elevado (compuestas por múltiples acciones pequeñas de bajo nivel), registradas por medio de apuntes, que produzcan un efecto real sintáctico o semántico en el código. Todos estos apuntes se complementaron con fragmentos de código para ilustrar las acciones en el libro de códigos del Estudio 2.

3.3 Diseño y desarrollo del software

Posterior a la primera fase, Estudios 1 y 2, se siguió un proceso de creación de un instrumento nuevo, consistente en un proceso de desarrollo de software a partir de las restricciones y funcionalidades identificadas en la interpretación de resultados de los estudios. Es decir, se diseñó e implementó un software que cumpliera con la recolección de información cuantitativa de estudiantes que programan. Además, se siguió un proceso de producción de un instrumento de recolección de datos susceptibles de cuantificación, a partir de los resultados de la fase anterior. Este proceso se da según este diseño metodológico pues posibilita el Estudio 3, en el que se aplica el instrumento y se analizan los resultados. En la Figura 3.2 se muestra este diseño.

Figura 3.2

Proceso de desarrollo del software de la investigación



El desarrollo del software empezó con la conversión del resultado de la fase anterior en funcionalidades deseadas del software para realizar el diseño y desarrollo del software para el registro de las acciones y el análisis de los resultados obtenidos de la aplicación del software nuevo. Algunas otras tareas incluyeron el diseño de interfaz,

diseño y desarrollo de software para el análisis, la selección de plataforma de desarrollo, programación de prototipo, definición de condiciones de prueba y de aplicación del software. También, se diseñó la manera de registrar las acciones y la manera de reportar los resultados del registro para cada sujeto que se estudie (cada ejecución del programa). El proceso de desarrollo contempla eventualidades del proceso como modificaciones necesarias al diseño que surgen en la implementación.

La implementación del software comprende tareas específicas propias de cualquier sistema computacional. Éstas incluyen la elección de estructuras de datos, de funcionalidades específicas según el usuario (administrativo o no), el interfaz y las tecnologías específicas para el registro de interacciones. El desarrollo en sí es un ciclo incremental que comprende las fases de diseño, implementación, prueba y corrección. Cada iteración del ciclo de desarrollo del software produce como resultado un software cada vez más funcional y robusto.

3.4 Estudio 3

3.4.1 Estudio 3 - Descripción

En este estudio se recolectaron datos de programadores, con el software desarrollado y en el momento de programación, para su posterior análisis. El propósito de este estudio fue aplicar el software de captura y análisis de acciones para identificar los efectos en el código indicativos de destrezas metacognitivas para su análisis e interpretación.

Además, se buscó conectar los resultados de los estudios previos, que identificaron posibles relaciones entre acciones y destrezas metacognitivas, y permitieron desarrollar el software específicamente para registrar esas acciones con los resultados de este estudio que registró las acciones reales de programadores, y sus efectos en el código, y resultados de la aplicación de un instrumento de autoreporte de destrezas metacognitivas conocido como MAI (G. Schraw y Dennison, 1994).

3.4.2 Estudio 3 - Participantes

Esta fase incluyó la aplicación del software completo para observación en estudiantes de computación en un primer curso de programación en la Universidad de Costa Rica. Se aplicó el software en 29 estudiantes del segundo semestre del año 2018 y en 20 estudiantes del primer semestre del año 2023. Los estudiantes se ofrecieron voluntariamente a participar y tenían entre 19 y 21 años.

3.4.3 Estudio 3 - Instrumentos

En el tercer estudio se utilizó el cuestionario de autoreporte de destrezas metacognitivas MAI. El software de análisis R (R Core Team, 2021) fue utilizado para análisis de grupos y generación de gráficos. También se utilizaron software de desarrollo y despliegue para la depuración de datos y análisis de grupos. En este estudio se utilizó el software desarrollado para tal efecto, y sirvió para el registro de las acciones de los estudiantes participantes y también para el análisis de los datos recolectados.

3.4.4 Estudio 3 - Procedimiento y estrategia de análisis

En este estudio se aplicó el software en un ambiente de laboratorio de computación a estudiantes programando. Los datos se recolectaron automáticamente con el software, desde el módulo de captura, según los cambios en el código que realizaban los estudiantes.

A diferencia del estudio 2, en el que se registraron acciones de teclado sin interpretación del tipo de acción. En este estudio el software posibilita, y el análisis interpreta, las acciones por su significado sintáctico y semántico en el código. Esto es, no solo se registraron teclazos, sino que se registró cuando los teclazos corresponden a un efecto en el código como la creación de una variable o el cambio de una condición en una instrucción condicional. Esto implica que las observaciones fueron más cercanas a la

intención del programador. Es decir, se registraron efectos en el código, que producen cambios sintácticos y semánticos en el programa.

Al final de la programación de los ejercicios, se les solicitó a los estudiantes completar una escala de medición de metacognición, MAI (G. Schraw y Dennison, 1994), en la que pudieron reportar su opinión acerca de sus propias características típicas a la hora de enfrentarse ante un problema en un contexto de aprendizaje. Los resultados fueron registrados en una bitácora aparte.

La cantidad y la estructura de los datos que se registraron es grande y compleja. Debido a esto se utilizó un sistema experto difuso, un aprendizaje automático y otras técnicas de identificación de grupos para aproximar la evaluación de destrezas metacognitivas mediante la caracterización de los estudiantes y grupos. Con los datos se realizaron distintos tipos de análisis. Luego de familiarización con los datos, se aplicó el sistema experto y luego se realizó análisis de grupos (clustering) de estudiantes que agrupan según patrones identificados en las acciones registradas, con la intención de caracterizar cada grupo según las destrezas metacognitivas asociadas. Es en este punto donde se combinan los resultados de los tres estudios, relaciones señaladas por expertos, relaciones señaladas por teoría y observación y destrezas metacognitivas auto reportadas.

Para el análisis de grupos se aplicaron distintas técnicas como el análisis de componentes principales (PCA), agrupación jerárquica y mapas auto organizados (SOM). Esta última técnica posibilita una inspección visual de los grupos de estudiantes. Para complementar esta inspección visual se combinaron los resultados de los tres estudios mediante la aplicación de las reglas de producción, con un sistema experto que asigna nivel de destrezas metacognitivas según los resultados del primer estudio, con información del segundo estudio y la información recogida del MAI del tercer estudio.

La interpretación completa de los datos se da cuando se relacionan las acciones observadas con las destrezas metacognitivas que pueden estar indicando. Analizando esas relaciones, logrando, por ejemplo, encontrar que el valor arrojado por el sistema experto se para regulación se correlaciona con el valor de regulación según el autorreporte MAI se

obtiene mayor certeza de que las acciones observadas en efecto indican regulación, en este ejemplo, a partir de lo que señalaron los expertos.

Capítulo 4 - Resultados

4.1 Resultados de Estudio 1

En este estudio se realizó un análisis temático de las opiniones de expertos, registradas como apuntes de entrevistas, de acciones de los estudiantes que aprenden a programar de acuerdo con destrezas metacognitivas. Adicionalmente, y relacionado a este punto, se formularon reglas de producción para representar este conocimiento experto de relación entre destrezas metacognitivas y acciones de programadores. También se obtuvo, como resultado, una parte de la clasificación que fundamenta el diseño del software necesario para el Estudio 3, esta incluye la organización de acciones por efecto sintáctico y semántico y también por requerimientos de software (de captura o de análisis, ver capítulo de resultados). A continuación se detallan estos resultados.

En este estudio se realizó un análisis temático siguiendo los seis pasos típicamente asociados a este método (Braun y Clarke, 2006, 2019) que incluyen la familiarización de los datos, la codificación, la asignación de temas a los códigos y la generación del resultado final. La familiarización con los datos arrojó múltiples ideas sobre codificaciones que, en este primer paso significó utilizar al máximo la flexibilidad ofrecida por la herramienta seleccionada. Para esto, en una hoja de cálculo se incluyeron las anotaciones y se etiquetaron preliminarmente para leer y estudiar los datos.

Se realizó un análisis temático de dos maneras, primeramente por temas predefinidos según la destreza metacognitiva que se le puede asociar a un apunte, adicionalmente se realizó una clasificación con temas emergentes de los apuntes según su contenido o significado.

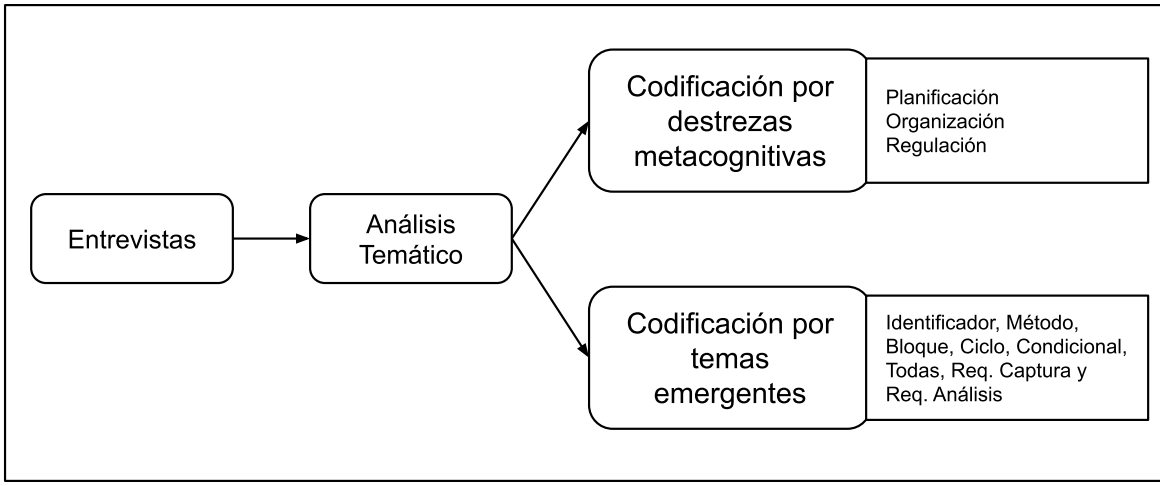
Los códigos y temas finalmente seleccionados se mencionan más adelante. Naturalmente, surgieron múltiples temas emergentes y posibles incluyendo por tecnología posible de detección de la acción y por efecto sintáctico o semántico de la acción en el código. Este paso es iterativo, en cada pasada se hizo una nueva asignación de códigos y

temas a los apuntes incluyendo una clasificación en dos niveles (nivel 1 y nivel 2) o se le modificó el código a algo. Los niveles corresponden a la agrupación de códigos bajo códigos más generales y finalmente agrupación bajo temas aún más generales. De esta manera cada tema agrupa apuntes etiquetados con un código de nivel 2 y cada código de nivel 2 agrupa apuntes etiquetados con un código de nivel 1. La definición de temas (grupos de códigos) es el último paso en la clasificación y previo a un proceso iterativo de refinamiento de estos temas (pasos tres y cuatro).

En el quinto paso se seleccionó una organización temática final, ilustrada en la Figura 4.1, que incluye una codificación por destrezas metacognitivas y otra por el efecto sintáctico y/o semántico que la acción produce en el código. La selección de esta organización temática, como resultado del proceso iterativo de codificación y revisión de códigos y temas permitió, por un lado recoger el conocimiento de expertos con respecto a la relación entre algunas acciones y destrezas metacognitivas, y por otro lado la generación más directa de requerimientos de software y la búsqueda de soluciones para implementar estos requerimientos. Esto último se detalla más adelante como parte de la descripción del desarrollo del software.

Figura 4.1

Codificaciones resultantes del análisis temático del estudio 1



Como sexto paso del análisis temático, se definen los detalles de los temas incluyendo los códigos que agrupan y los criterios de aplicación de temas y códigos. La codificación se resume en un libro de códigos. El libro de códigos tiene la lista completa de temas y códigos utilizados finalmente, junto con una descripción. Se definieron criterios de aplicación de estos temas a los apuntes, que también se incluyen en el libro de códigos. Para la codificación con temas emergentes finalmente se definieron estos temas como resultado del análisis temático: Identificador, Método, Bloque e Instrucción, Ciclo, Condicional, Requerimiento de Captura y Requerimiento de Análisis. Estos nombres corresponden, principalmente, a la categoría sintáctica en la que se puede agrupar una acción y, en el caso de los últimos dos, a funcionalidades deseadas o requerimientos que los expertos identificaron como necesarios o deseables.

Es necesario indicar que el análisis temático, aunque lleva a un proceso sistemático resulta siempre de la opinión subjetiva del investigador por lo que al replicarse podría arrojar resultados distintos. Sin embargo, en este caso, lo que podría variar son los criterios de aplicación de los temas de destrezas metacognitivas o, en la codificación por temas emergentes, la selección de temas del análisis ya que la asignación de temas, según categoría sintáctica o semántica, es directa y de acuerdo al efecto en el código que produce.

4.1.1 Codificación por temas emergentes

Como parte del análisis temático de los datos de entrevistas está la codificación de los datos por temas emergentes. En esta codificación se realizó un proceso iterativo de agrupación de apuntes según el contenido de estos y todas las etiquetas, tanto los códigos como los temas, surgieron a partir de esto.

El libro de códigos incluye, para esta codificación, los temas emergentes junto a su descripción y criterio para aplicación. A diferencia de la codificación por temas emergentes, el criterio para la aplicación se va construyendo conforme se avanza en el

proceso y no de forma previa. Así, en este caso los resultados son tanto la codificación como el contenido del libro de códigos.

La descripción de los temas, en el libro de códigos, sirve para comprender a qué se refiere el código, o tema, y el criterio de asignación sirve para especificar cómo se asignó cada código o tema, lo que facilita la replicación del estudio a partir de este punto. El libro de códigos de este estudio completo se incluye como Apéndice B. Para ilustrar el contenido del libro de códigos, en la Tabla 4.1 se incluyen, como ejemplo, algunos de los códigos y temas utilizados, con una descripción y el criterio de aplicación.

Tabla 4.1

Ejemplos de criterios de aplicación en apuntes de entrevistas

TEMA	Criterio de aplicación
Identificador	Se indica algo respecto al uso de datos, su espacio, su variable asociada y el nombre de la variable o valor
Método	Se indica la creación, modificación, uso o eliminación de métodos, llamados así, explícitamente o no (subprocesos reutilizables).
Otras	Se indican acciones no directamente relacionada con una categoría sintáctica o semántica específica sino con acciones más generales como la compilación o ejecución.

En la codificación basada en temas emergentes, se identificaron algunos temas asociados con una categoría sintáctica o semántica de la acción sobre el código fuente que se está programando. Dos temas emergieron del análisis para agrupar códigos asignados a apuntes relacionados directamente a lo que los expertos consideraron requerimientos del software de captura o análisis. Los temas relacionados a requerimientos se describen más adelante.

De los temas definidos según una categoría sintáctica o semántica, la categoría de "Identificador", por ejemplo, incluye las acciones que fueron codificadas como acciones de

variable (agregar una variable o cambiarle su nombre o su valor). Es decir, las acciones relacionadas a las variables y sus nombres fueron agrupadas bajo este tema.

Similarmente, los otros temas incluyen acciones agrupadas según ese tema o categoría. Bajo el tema de "Otras", donde se agrupan acciones que no necesariamente corresponden a otras categorías sintácticas o semánticas. En la Tabla 4.2 se muestran estos temas, con una breve descripción del tema y la proporción.

Tabla 4.2

Temas emergentes que agrupan acciones codificadas

Tema	Descripción
Identificador	Reúne acciones sobre variables.
Método	Reúne acciones sobre los encabezados o el contenido de los métodos.
Bloque e instrucción	Reúne acciones sobre bloques que pueden estar dentro de métodos en un primer nivel o en un nivel más profundo e instrucciones no necesariamente incluídas en otras categorías, como una instrucción de control.
Ciclo	Reúne acciones sobre los ciclos utilizados en el código
Condicional	Reúne acciones sobre estructuras condicionales IF y SWITCH
Otras	Reúne acciones más generales, que pueden realizarse aún cuando se esté en cualquier parte del código, por ejemplo, la compilación o la ejecución del código.

Previo a llegar a estos temas, y a partir de la codificación de los apuntes, surgió la necesidad de organizar las etiquetas o códigos en múltiples niveles. Se consideraron necesarios dos niveles para agrupar los apuntes, y luego también agrupar esos primeros grupos de apuntes. Es decir, en un primer nivel (Nivel 1) los códigos agrupan apuntes, y en un segundo nivel (Nivel 2) los códigos agrupan códigos de primer nivel. Esto hace que los

temas (etiquetas asignadas a códigos) en realidad estén más directamente asociados a la codificación de Nivel 2. La Tabla 4.3 muestra la organización en temas y códigos que se definió para los grupos de códigos de ambos niveles.

Tabla 4.3

Resumen de análisis temático, codificación por temas emergentes

TEMA	Codificación Nivel 2	Codificación Nivel 1	Cantidad
IDENTIFICADOR	Agregar variable	Agregar muchas variables	**3**
		Agregar variable	**5**
	Modificar nombre de variable	Modificar nombre de variable	**5**
		Muchos cambios de nombre de variables	2
		Pocos cambios de nombre de variable	1
	Mover variable	Mover variables	2
	Usar variable	Usar pocas variables	2
		Usar variable	2
		IDENTIFICADOR Total	**22**
MÉTODO	Agregar método	Agregar método	8
	Llamar método	Llamar método	2
		Muchos llamados a método tarde	3
	Mover método	Mover método	4
		MÉTODO Total	**17**
CICLO	Crear ciclo	Crear ciclo	**10**
		Crear ciclos temprano	**4**
	Modificar condiciones de ciclo	Modificar condiciones de ciclo	**2**
		Modificar condiciones de ciclo tarde	**1**
		Modificar condiciones de ciclo temprano	**1**
	Modificar contenido de ciclo	Modificar contenido de ciclo	**2**
		CICLO Total	**20**
BLOQUE E INSTRUCCIÓN	Agregar instrucción	Agregar instrucción	4
	Crear bloque	Crear bloque	3
	Modificar instrucción	Eliminar instrucción	3
		Modificar instrucción	1
		BLOQUE E INSTRUCCIÓN Total	**11**
CONDICIONAL	Crear condicional	Crear condicional	5
		Crear condicional if	2
	Modificar caso de switch	Modificar caso de switch	2
	Modificar condición de if	Modificar condición de if	5
	Modificar contenido de if	Modificar contenido de if	4
		CONDICIONAL Total	**17**
OTRAS ACCIONES	Compilar	Compilar	8
	Consultar información adicional	Consultar información adicional	4
	Ejecutar	Ejecutar	2
	Navegar	Navegar en el código	5
		TODAS Total	**18**
		TOTAL DE APUNTES	**99**

Aunque el contenido temático es lo que guía el análisis y son pocos datos, se puede observar como la distribución de apuntes a temas es pareja, siendo el tema de "Identificador" el que más agrupa acciones. Las cuatro acciones de compilar, consultar, ejecutar y navegar en el código agrupan similar cantidad de observaciones que los otros temas. Esto sugiere la pertinencia de estas cuatro acciones, esto porque, a pesar de ser solo cuatro acciones recibieron similar cantidad de menciones que otros grupos de acciones.

Para la codificación por temas emergentes, que tiene seis temas, se muestran en la Tabla 4.4 solo algunos ejemplos de apuntes que dieron origen a los temas. El detalle completo incluye los apuntes y está en el Apéndice C.

Tabla 4.4

Ejemplos de datos codificados por temas emergentes de entrevistas

Tema y Descripción	Datos	Código Nivel 1	Código Nivel 2
Tema IDENTIFICADOR Agrupa acciones de creación, uso y modificación de variables.	"Algunos agregan código indiscriminadamente" "Unos cuantos definen varias variables al inicio sin saber para qué serán." "Lo típico es definir variables conforme se van requiriendo"	Agregar muchas variables	Agregar variable
	"Si no hay claridad cambian mucho los nombres de variables" "Usualmente los nombres de variables se mantienen hasta el final, aunque sean malos nombres"	Muchos cambios de nombre de variables	Modificar nombre de variable
Tema MÉTODOS Agrupa acciones de creación, modificación y llamado de métodos.	"Usar métodos es señal de mayor comprensión del problema" "Una solución con métodos está más organizada" "Si temprano se definen métodos hay planificación"	Agregar método	Agregar método
	"Cuando se definen y llaman más métodos el quiere decir que el programa está mejor estructurado" "Si se planifica bien se puede dividir el problema en métodos para después solo llamarlos"	Muchos llamados a método tarde	Llamar método
	"Progresar en el código es cuando ya tiene pensado qué va a hacer." "Cuando agrega una línea de código se va implementando lo que pensó"	Agregar instrucción	Agregar instrucción
	"Cuando se cambian los términos de un ciclo es clara evidencia que se está intentando corregir algo" "Cambiar cualquier parte de un ciclo muestra compresión de que hay un problema y se está haciendo algo para remediarlo"	Modificar condiciones de ciclo tarde	Modificar condiciones de ciclo

Tabla 4.4 (cont.)

Ejemplos de datos codificados por temas emergentes de entrevistas

Tema: CONDICIONALES Agrupa acciones de creación, modificación de condicionales. Incluye las acciones de creación y modificación de casos de condicionales "*switch*".	"Los condicionales se usan para estructurar el código" "Uno piensa en términos de condicionales, para ir acomodando ideas"	If	Crear condicional if
	"Los cambios en el código se realizan cuando se sabe que hay un problema"	Modificar condición de if	Modificar condición de if
Tema: TODAS Agrupa acciones que pueden más generales no necesariamente relacionadas a una categoría sintáctica o semántica.	"Compilar y ejecutar son tareas básicas que deben poder realizarse y registrarse" "Compilar es un intentar" "Algunos programan y programan y no compilan hasta el final, y vienen los problemas"	Compilar	Compilar
	"Compilar y ejecutar son tareas básicas que deben poder realizarse y registrarse"	Ejecutar	Ejecutar

Esta codificación también se complementa con otros dos temas asociados a los apuntes de las entrevistas acerca de requerimientos del software. Estas se muestran aquí aparte pues no están directamente relacionadas a las acciones observables en estudiantes sino más bien al ambiente del software de observación. Como se ve a continuación, éstas se clasificaron como requerimientos de captura o análisis. De este análisis surgió la necesidad de modularizar el código según funcionalidades y se determinó que algunos de estos requerimientos son específicamente para el módulo de captura y otros para el módulo de análisis del software desarrollado. Es decir, este es un resultado aparte del análisis temático que se tomó también como insumo al diseño del software, necesario para el estudio 3.

Los resultados relacionados a requerimientos se presentan, aparte, en la Tabla 4.5. Esta incluye los temas de Análisis y Captura que agrupan los códigos de nivel 2, que a su

vez agrupan los códigos de nivel 1. Además, se muestra también la cantidad de apuntes clasificados a cada código de Nivel 1 y el porcentaje correspondiente a los apuntes específicamente acerca de requerimientos.

Tabla 4.5

Resúmen de análisis temático, codificación por temas emergentes, para requerimientos del software

TEMA	Codificación Nivel 2	Codificación Nivel 1	Cantidad	Porcentaje
Análisis	Análisis	análisis	2	3.77%
		comparación entre estudiantes	5	9.43%
	Bitácora	bitácora	4	7.55%
		Requerimientos de Análisis Total	**11**	**20.75%**
Captura	Ambiente	compilar/ejecutar	3	5.66%
		facilidad de uso	8	15.09%
		funcionalidades extras	2	3.77%
		ide	7	13.21%
		identificar elementos del código	1	1.89%
	Ambiente/Visual	ide	5	9.43%
	Programación	compilar/ejecutar	6	11.32%
		identificar elementos del código	2	3.77%
		metainformación	1	1.89%
	Programación automática	funcionalidades extras	7	13.21%
		Requerimientos de Captura Total	**42**	**79.25%**
		Total de Apuntes	**53**	**100%**

4.1.2 Codificación por destrezas metacognitivas

Para el análisis temático por temas predefinidos según destrezas metacognitivas se asignaron los temas de planificación, organización y regulación a los apuntes de las entrevistas. Estas destrezas metacognitivas fueron seleccionadas previamente y discutidas con los entrevistados como paso introductorio de las entrevistas. Aunque sí fue posible agrupar algunos de los apuntes tomados de entrevistas como más relacionadas a alguna de

estas destrezas, no todos los apuntes tomados de las entrevistas pueden ser etiquetados de esta forma, pues no todos se relacionan directamente con una destreza metacognitiva.

Para el caso de esta codificación tanto los temas como los criterios para su aplicación fueron definidos previamente al análisis. Eso sí, señalar una acción como relacionada a una destreza, en este estudio es una manera de representación del conocimiento del experto, en su opinión. Por ejemplo, si el experto señaló que, según su experiencia, conoce que una acción es indicativa de un momento en que el estudiante ha planificado entonces el apunte correspondiente fue etiquetado como de planificación. Esta relación recalca, entonces, la experiencia y conocimiento del experto, y debe entenderse como tal.

Más adelante, se analizaron estas relaciones junto a los resultados del estudio 2, se detectaron relaciones similares basadas en observación, y luego se combinaron también con resultados del estudio 3 que recogió información de las acciones tomadas y de las destrezas metacognitivas con un autorreporte. En la Tabla 4.6 se muestran estos temas junto con la descripción de estos y los criterios para la aplicación de estos temas a apuntes.

Tabla 4.6

Temas por destreza metacognitiva y criterios para su aplicación en apuntes de entrevistas

TEMA (destreza metacognitiva)	Descripción	Criterio para su aplicación
Planificación	Agrupa apuntes que relacionan acciones con planificación	Asocia acción con metas, subtareas, plan, estrategia, pasos o algún conocimiento previo usado en una acción futura
Organización	Agrupa apuntes que relacionan acciones con organización	Asocia acción a orden, modularización, estructura, acomodo o descomposición
Regulación (monitoreo/evaluación)	Agrupa apuntes que relacionan acciones con regulación	Asocia acción a cambios, adaptación, reacción, evaluación del proceso o del programa

Aplicando estos criterios a los apuntes de las entrevistas se asociaron destrezas metacognitivas, en la opinión de expertos, a acciones que los programadores toman al programar. En la Tabla 4.7 se muestran las acciones que los expertos señalan como indicativas de alguna de las destrezas metacognitivas de organización, planificación o regulación. Algunos de los apuntes no se asocian con destrezas metacognitivas (indicados como “Otros” en la tabla).

Tabla 4.7

Resúmen de análisis temático por tema predefinido según destreza metacognitiva

DESTREZA	Codificación Nivel 2 EFECTO DIRECTO EN EL CÓDIGO	Codificación Nivel 1 ACCIÓN DEL PROGRAMADOR	Cantidad
ORGANIZACIÓN	AGREGAR	agregar método	6
		agregar variable	2
		crear bloque	2
		crear ciclos/ crear ciclos temprano	4
		crear condicional	4
		llamar método	3
	MODIFICAR	indentación	1
		mover método	1
		pocos cambios de nombre de variable	2
	NINGUNO	compilar/ejecutar	1
		ORGANIZACIÓN Total	**26**
PLANIFICACIÓN	AGREGAR	agregar método temprano	2
		agregar variable/agregar variable temprano	1
		crear bloque	1
		crear ciclos temprano	1
		crear condicional	1
		llamar método	1
	MODIFICAR	modificar condiciones de ciclo	2
	NINGUNO	navegar en el código	3
		PLANIFICACIÓN Total	**12**
REGULACIÓN	ELIMINAR	eliminar instrucción	1
	MODIFICAR	mover variable	1
		modificar caso de switch	3
		modificar condición/contenido de if	5
		modificar condiciones de ciclo	3
		modificar contenido de if	1
		modificar instrucción	1
	NINGUNO	compilar/ejecutar	5
		navegar en el código	1
		REGULACIÓN Total	**21**
		OTROS	45
		TOTAL	**104**

En la codificación puede suceder un sobrelapado (Gibbs, 2007). Esto sucedió en la codificación según destrezas metacognitivas. Es decir, hay algunos apuntes que indican algo acerca de más de una destreza metacognitiva por lo que el apunte puede ser etiquetado

o agrupado bajo más de un tema. Por ejemplo, un apunte de una entrevista es: “Si se planifica bien se puede dividir el problema en métodos para después solo llamarlos”. En este caso aplica tanto el criterio de aplicación del tema planificación como el de organización. Este sobrelapado es comprensible debido al hecho de que como destreza metacognitiva, la organización sucede en apoyo a la planificación. Para ilustrar el contenido del reporte del análisis temático, en este caso para la codificación por temas predefinidos por destreza metacognitiva, en la Tabla 4.8 se muestran algunos ejemplos de datos junto a la destreza metacognitiva asociada. El detalle completo del análisis temático aparece como Apéndice C.

Tabla 4.8

Ejemplos de apuntes de entrevista según destrezas metacognitivas

Destreza metacognitiva	Ejemplos
Planificación	“Si temprano se definen métodos hay planificación”
	“Si se planifica bien se puede dividir el problema en métodos para después solo llamarlos”
	"es curioso cuando se llama un método que todavía no se ha definido, quiere decir que ya hay noción de lo que se ocupa más adelante"
Organización (parte de planificación)	"es más ordenado usar una variable para cada cosa distinta"
	“Cuando se definen y llaman más métodos quiere decir que el programa está mejor estructurado”
	“Una solución con métodos está más organizada”
Regulación (monitoreo/evaluación)	“Cuando se cambian los términos de un ciclo es clara evidencia que se está intentando corregir algo”
	“Hay que estar probando lo que se está haciendo”
	"se compila para probar"

4.1.3 Sistema experto difuso

Otro resultado del Estudio 1 se obtuvo del análisis temático por destrezas metacognitivas como temas predefinidos. A partir de esta codificación fue posible crear un conjunto de reglas para un sistema experto difuso (Zadeh, 1965, 1999, 2008). El propósito

de este sistema es la representación del conocimiento de los expertos mediante un proceso formal en reglas de producción. De esta manera, se puede aplicar para evaluar el conjunto de acciones tomadas por el estudiante como indicativas de una u otra destreza metacognitiva, según opinión de expertos.

Un sistema experto difuso tiene variables difusas y reglas. Las reglas de producción sirven para representar el conocimiento de expertos y obtener así una evaluación, o diagnóstico, automáticamente aplicando un motor de inferencia con las reglas a datos nuevos (Buckley et al., 1986; Siler y Buckley, 2005). Las reglas se aplican según valores de variables difusas y se generan conclusiones difusas.

En este caso se propone el uso del sistema experto para obtener una evaluación de las destrezas metacognitivas a partir de las acciones tomadas por los estudiantes, o el efecto en el código. Así, las variables del sistema experto son los conteos de acciones tomadas y las destrezas metacognitivas. Al tratarse del conocimiento expresado en lenguaje natural con cierto nivel de graduación con respecto a los conocimientos expresados acerca de destrezas metacognitivas, la lógica bivalente no es suficiente, la lógica difusa captura mejor el conocimiento comprendido (Kandel, 1991; Zadeh, 1999).

El conocimiento de los expertos que se representa es con respecto a la relación entre destrezas metacognitivas y las acciones que pueden indicarlas. Esto permite la evaluación de los estudiantes en un contexto definido y con respecto al ejercicio resuelto ya que se basa en las acciones tomadas para resolver ese problema.

Se puede aplicar considerar esta evaluación en comparación con otros estudiantes del mismo grupo, al que se les solicitó resolver el mismo ejercicio. Es decir, si algún estudiante destaca respecto al resto en cuanto a una acción el sistema experto lo calificará mejor y si tiene una cantidad de acciones de un tipo promedio igual será la calificación del sistema experto.

El sistema experto difuso, al representar el conocimiento de los expertos, ofrece la posibilidad de evaluación de los estudiantes según sus acciones. Así, puede compararse el

resultado de evaluación con el sistema experto y con el resultado de la aplicación del autorreporte MAI. Más detalles de esto en las siguientes secciones.

El proceso de definición de las reglas de un sistema experto (difuso o no) es conocido como la representación del conocimiento y hay un campo de la Inteligencia Artificial conocido como Ingeniería de Conocimiento que se dedica a este tipo de tareas (McShane et al., 2022; Salem y Nikitaeva, 2019).

La representación del conocimiento en reglas difusas es un caso especial para cuando las variables lo ameriten por su naturaleza graduable. La lógica difusa que fundamenta los sistemas expertos difusos incluye un proceso de manipulación de las variables para que las inferencias obtenidas tomen en cuenta este aspecto y sean congruentes con los valores que ingresan al sistema. La inferencia de un sistema experto difuso pasa por etapas de fusificación, aplicación de reglas, agregación de los resultados y defusificación (Kandel, 1991; Negnevitsky, 2005; Tavana y Hajipour, 2019).

El proceso de desarrollo de un sistema experto difuso, aparte de la selección de motor de inferencia, en este caso R (R Core Team, 2021), incluye la definición de variables difusas, conjuntos difusos y las reglas de inferencia. Los conjuntos difusos son conjuntos con las fronteras difusas, donde los elementos tienen un grado de pertenencia en lugar de una condición determinista de pertenencia o no. En el estudio se definió una variable difusa para cada acción registrada y para cada destreza metacognitiva.

Para todas las variables se eligió manejar tres niveles de valor (bajo, medio y alto) por ser una representación sencilla de las posibilidades enunciadas por los expertos que no señalaron mayor nivel de detalle en la distinción de los valores de las variables. Así, por ejemplo, para el conteo de la cantidad de veces que el programador agregó una variable temprano se establecen los tres niveles de bajo, medio y alto. Estos valores se calculan luego de la normalización de los datos de todos los participantes. De esta manera el sistema es capaz de manejar la incertidumbre de qué significa exactamente un bajo nivel (o medio, o alto) de una variable. Bajo, medio o alto se refieren a valores en comparación al resto. Esto hace que un programador que se destaque en una destreza con niveles altos en

todas las variables relacionadas, obtenga en el sistema un valor alto de pertenencia al conjunto correspondiente a esa destreza.

La tabla 4.9 muestra las acciones relacionadas, según la opinión de expertos, a las destrezas metacognitivas de planificación, regulación y organización. Para cada acción, se definió una regla en el sistema experto que hace aumentar o bajar el nivel de pertenencia al conjunto de esa destreza metacognitiva según el conteo de acciones de cada programador. Así, las reglas toman la forma "SI TOMA X ACCIÓN ENTONCES INDICA Y DESTREZA". Al tratarse de lógica difusa, no basta con que se cumpla una acción para obtener el nivel más alto sino que se toma en consideración todos los niveles de todas las variables.

Tabla 4.9

Reglas de producción del sistema experto difuso basado en destrezas metacognitivas, según opinión de expertos

Reglas son de la forma: **SI TOMA ACCIÓN → INDICA DESTREZA**	
ACCIÓN	DESTREZA
Agregar variables / métodos temprano, Desarrollar ciclos y métodos Usar variables y métodos	PLANIFICACIÓN
Cambios de nombre de variable Agregar y usar método mover método agregar variable temprano	ORGANIZACIÓN
Ver código (ejecutar/navegar) Leer instrucciones modificar / reorganizar código en métodos	REGULACIÓN

Nota: Se muestran las acciones relacionadas a destrezas, de cada una se hace una correspondencia a una regla de la forma **SI TOMA ACCIÓN → INDICA DESTREZA**. Por ejemplo, según la primera acción asociada a planificación si el programador llama un método aún no creado se indica planificación.

La Tabla 4.10 (a) muestra la definición de una de las variables para la opinión del experto que se resume así: "agregar muchas variables temprano puede ser que está

dividiendo para atacar". En esta frase se ve muy claro el aspecto difuso de la opinión pues gradúa o califica la opinión al decir "muchas" y al decir "puede ser".

Adicionalmente, la planificación es implícita (entendida como definición de una estrategia) pues no la menciona por nombre. La Tabla 4.10 (b) muestra una regla del sistema que se construye a partir del conocimiento representado en el apunte de la entrevista tomando en cuenta las graduaciones o calificaciones que hace y por eso como premisa establece un nivel alto de variables definidas temprano y como conclusión de la regla establece solo un nivel medio (no hay clara indicación de planificación, de hecho la mención es implícita). Adicionalmente, se muestran reglas derivadas de la anterior, para representar las otras situaciones asociadas. En estas otras reglas se asigna un nivel bajo de planificación, no porque esto es lo que el experto señala sino porque estas reglas en particular no servirán para aportar valor al diagnóstico final de Planificación pues no hay información, en esta misma regla, para hacer lo contrario. En este ejemplo solo se muestra la representación del conocimiento para una acción, en el sistema experto difuso completo se toman en cuenta todas las acciones ya mencionadas para cada destreza metacognitiva por lo que el diagnóstico final (resultado del sistema experto) se obtiene luego de aplicar todas las posibles reglas según las acciones registradas para cada estudiante.

Tabla 4.10

Una variable y tres reglas del sistema experto en R, para la acción de agregar variables temprano.

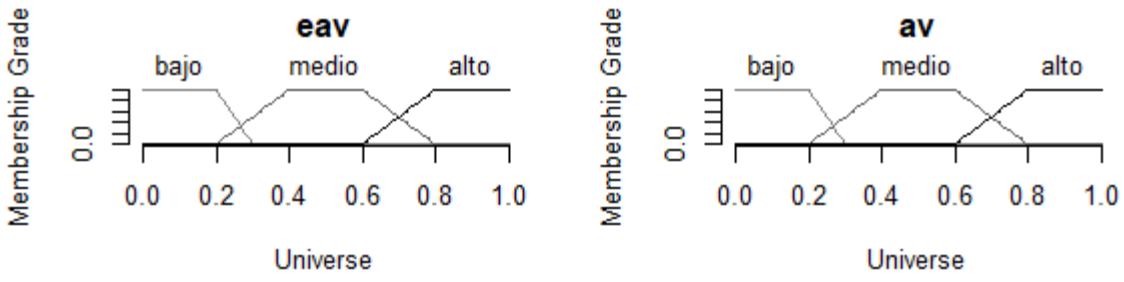
(a) Variables difusas	Agregar Variables Temprano Valores posibles: Bajo, Medio y Alto Planificación Valores posibles: Bajo, Medio y Alto
(b) Regla	Si AgregarVariablesTemprano es Alto entonces Planificación es Medio Si AgregarVariablesTemprano es Bajo entonces Planificación es Bajo Si AgregarVariablesTemprano es Medio entonces Planificación es Bajo

La Figura 4.2 muestra el gráfico de dos variables difusas según se utilizan en el sistema experto, con tres niveles de bajo, medio y alto, y con trapezoides definiendo las fronteras entre los niveles. Esto es parte de la manera en la que el sistema maneja la incertidumbre entre valores similares. Se aprecia, además, que los valores posibles de entrada están en un rango entre 0 y 1. Esto se hace porque la intención no es determinar un valor a partir del cual una variable indica una destreza sino más bien asignar mayor membresía a la variable difusa según mayor cantidad de acciones de un tipo se haya mostrado en relación con otros estudiantes que realizaron el mismo ejercicio.

A partir de lo anterior, es pertinente señalar esto para tomarlo en cuenta en el análisis e interpretación de los resultados. En resumen, el valor asignado por el sistema experto es relativo a otros estudiantes que realizaron el mismo ejercicio y no un nivel global de una destreza. El código del sistema experto difuso se encuentra en el Apéndice J.

Figura 4.2

Variables difusas, correspondientes a acciones registradas, del sistema experto que representa conocimiento de expertos



Una vez recogidos los datos de una aplicación del software se aplicó el sistema experto difuso para obtener un grado de pertenencia al conjunto difuso correspondiente a las destrezas metacognitivas mencionadas para cada ejercicio. Esto se hace, luego de

normalizar los datos de todos los estudiantes al que se le aplicó el software obteniendo así un nivel de uso entre 0 y 1 para cada acción. El sistema toma en cuenta todos estos valores y evalúa, según el conocimiento de expertos codificado en las reglas, el nivel de destreza metacognitiva para planificación, organización y regulación. Estos niveles luego se compararon con los niveles determinados a partir de los resultados de los siguientes dos estudios.

4.2 Resultados de Estudio 2

En el estudio 2 se levantó igualmente una lista de apuntes, pero los apuntes fueron provenientes de la observación. Igual que en el estudio anterior un análisis temático se realizó de dos maneras, con temas predefinidos y con temas emergentes. Utilizando como base las tres destrezas metacognitivas señaladas en la investigación, se clasificaron apuntes según la destreza observada.

Una gran diferencia con respecto a la anterior, es que esta clasificación combina las observaciones con una base en teoría relacionada (Bergin et al., 2005; Havenga, 2015; Havenga et al., 2012; Westhuizen y Chetty, 2013). Esto se hace por medio de los criterios de aplicación del tema, que para este estudio son distintos. Por ejemplo, algunas acciones indican planificación del aprendizaje al evidenciar que el estudiante está incurriendo en decisiones para retrasar o no algunas modificaciones al código. Otras acciones pueden agruparse según el proceso organizativo que debe estar realizando el estudiante en el código y en su proceso mental. También hay acciones que indican cómo el programador está interactuando con el código para evaluar progreso, con indicativas de la regulación del aprendizaje.

Algunas acciones pueden ser indicativas tanto de la habilidad de organización como de la habilidad de planificación, por ejemplo. Para ilustrar, la inclusión de una sección de código en un bucle puede indicar una organización de ideas del programador

pero también su habilidad para planificar sus tareas. Este sobrelapado surge naturalmente de la complejidad de la programación.

El resultado de la codificación de apuntes de observación según destrezas metacognitivas permite tener una segunda aproximación al análisis de relaciones entre destrezas metacognitivas y acciones del programador. En este caso la clasificación, basada en teoría y de acciones observadas reales constituye un insumo adicional para identificar y analizar los efectos en el código indicativos de destrezas metacognitivas. Esta codificación se muestra en la Tabla 4.11.

Tabla 4.11

Temas por destreza metacognitiva y criterios para su aplicación en observación

Destreza metacognitiva	Criterio para su aplicación	Autores
Planificación	Fija metas, requiere conocimiento previo, indica lectura/análisis de tareas, indica alguna decisión sobre pasos a seguir	(Bergin et al., 2005; G. Schraw y Moshman, 1995)
Organización (parte de planificación)	Descompone en partes, determina necesidades, apoya el plan	(Sternberg y Sternberg, 2012, Van der Stel y Veenman, 2014)
Regulación (monitoreo/evaluación)	Sirve para evaluar avance, sirve para evaluar completitud de la solución	(Young, 2008)

Enlazando con los resultados del Estudio 1, en el análisis temático del Estudio 2, para la codificación con temas emergentes, se definió una clasificación basada en efectos sintácticos y semánticos en el código. Entre los motivos para seleccionar esa segunda clasificación están la relación con requerimientos de software que potencia esta clasificación como punto de integración con la fase siguiente y la proximidad del efecto sintáctico y semántico al proceso cognitivo y metacognitivo que desarrolla el estudiante. El solo hecho de agrupar teclazos en un acción que representa un efecto sobre el código, como crear una variable o modificar un método, hace que esta clasificación esté más cercana a la intención del programador que un registro de simples teclazos sin interpretación sintáctica o semántica.

Siguiendo igualmente los seis pasos de análisis temático ya mencionados para el estudio anterior, en el el Estudio 2 se encontró una segunda clasificación que surgió naturalmente por la tecnología necesaria para observar las acciones. Por ejemplo, se puede observar al programador por sus acciones en el teclado, en el mouse o incluso por la atención que concentra en distintas partes del código (según donde concentra su vista) y también se consideró más apropiado una clasificación centrada en la intención del estudiante según el efecto que produce en el código pues está más cercano a sus procesos mentales.

Tabla 4.11

Acciones observadas asociadas a destrezas según criterios de aplicación de estudio 2

Destreza metacognitiva	Acciones observadas
Planificación	Agregar y utilizar métodos y variables. Agregar y eliminar instrucciones
Organización (parte de planificación)	Agregar y modificar métodos, modificar variables o instrucciones
Regulación (monitoreo/evaluación)	Navegar en el código, compilar o ejecutar y hacer algunos cambios

Esto implica que de este estudio se pudieron descartar algunas acciones involuntarias de movimientos de mouse o teclado y otras como “hacer click a instrucción” y “seleccionar código” se descartaron por dificultad de determinación del significado de la acción en el proceso de solución. Otras incluyen acciones como navegar un solo espacio, modificaciones cortas y rápidas (como cuando hay error de teclazos) y acciones sobre las instrucciones de uso de bibliotecas externas del código (*import*). Se pudo observar que estas acciones no necesariamente significaban algo pues ocurrían muy comúnmente, asociado a un error secundario (como los teclazos involuntarios). En el caso de los import, se descartó al ser muy poco informativo pues algunos estudiantes utilizan, según se observó, la inclusión generalizada de bibliotecas necesarias (import con un “*”) que es ambigüo, en el Estudio 1 esta acción también fue considerada ambigua en la opinión de los expertos.

4.3 Resultados de Desarrollo de Software

Entre fases se llevó a cabo un proceso de desarrollo de software, como punto de integración entre los primeros dos estudios y el tercero, incluyendo el diseño y la implementación de software educativo de evaluación de destrezas metacognitivas. Se realizó un proceso de desarrollo iterativo incremental (Basili y Turner, 1985; Larman y Basili, 2003), comúnmente utilizado en programación, en el que se realizan pasos que se aplican secuencialmente en un ciclo de desarrollo. Esto permite programar, en cada iteración, funcionalidades al sistema, que se van agregando como requerimientos cumplidos en el software. Este proceso iterativo es conducente a un software, siempre funcional, cada vez más coherente con los requerimientos obtenidos de los estudios 1 y 2.

En esta sección se muestran los resultados de este proceso, incluyendo la definición de requerimientos para el software, el diseño y código final del software en sí. Se reportan los resultados de los pasos de diseño e implementación, luego de varias repeticiones.

4.3.1 Diseño

La información recopilada de los primeros dos estudios se utilizó como base para el diseño y desarrollo del software así como las posibilidades existentes en cuanto a tecnologías utilizadas. Este último aspecto es secundario pero sirvió para definir detalles técnicos en el desarrollo de software. Se diseñó el software de tipo heterogéneo en cuanto a plataformas utilizadas con interfaz gráfico sencillo y funcional que cumpliera la función de observación de estudiantes en pluralidad de ambientes físicos y análisis con requisitos de computación básicos.

Los requerimientos definidos para el desarrollo de software son el insumo principal para el diseño del software. La definición de requerimientos del sistema se basa en los resultados de los primeros dos estudios en cuanto a acciones a observar y demás

requerimientos de captura y análisis. La tabla 4.12 muestra el resultado de la definición de requerimientos.

Tabla 4.12

Requerimientos de software según insumos de los primeros dos estudios

Propósito	Evaluar destrezas metacognitivas mediante el registro de acciones de programadores
Población meta	Estudiantes de programación
Uso	Mediante un ambiente típico de programación se registran acciones de estudiantes en la solución programada de un problema. Este registro se utiliza para obtener una evaluación de destrezas metacognitivas en comparación a otros usuarios del sistema.
Requerimientos	● Registrar las acciones identificadas en los estudios 1 y 2 como indicativas de destrezas metacognitivas ● Presentar un ambiente típico de programación, fácil de usar, intuitivo, con números de línea, coloreo de elementos de programación y capacidad de compilación y ejecución ● Reportar problemas de compilación (con la información típica asociada) y resultados de ejecución de manera típica ● No incluir funcionalidades de programación automática como autocompletar y refactorización ● Permitir ingreso de apuntes ● Registrar acciones, de manera invisible, en una bitácora que incluya el estado actual del código y demás detalles para análisis ● Permitir reunir datos de múltiples programadores ● Permitir identificar grupos/categorías de programadores ● Evaluar el grado de pertenencia de un programador a un grupo/categoría ● Permitir la caracterización de los grupos por inspección visual ● Permitir el análisis de las relaciones entre efectos en el código y destrezas metacognitivas

Los requerimientos son las funcionalidades necesarias para implementar en la programación del software. Incluyen desde el interfaz de usuario, los componentes del interfaz, la captura de acciones y la recolección de datos para su posterior análisis. Un ejemplo es la necesidad de presentar al usuario, el estudiante programador, un interfaz sencillo y familiar a los ambientes de programación que conoce y controlar así la posible afectación de curva de aprendizaje del ambiente. Algunos requerimientos provienen del

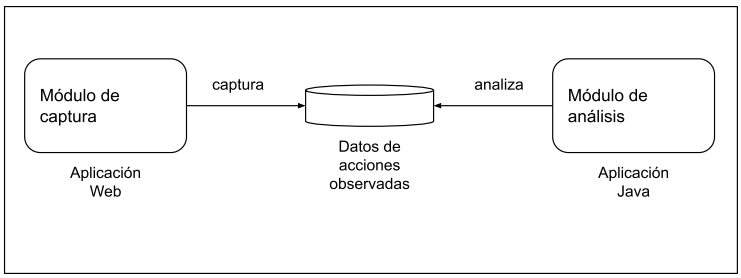
estudio 1, otros del estudio 2, o de ambos, y otros vienen del contexto y objetivos de la investigación.

El resultado del diseño es de dos componentes principales: un módulo de captura de acciones y otro de análisis. Esta estructura se muestra en la Figura 4.3. A nivel técnico cabe la descripción del sistema como de aplicación web para la captura de datos y de plataforma universal (según máquina virtual) en Java para el módulo de análisis.

El módulo de capturas cumple los requerimientos de captura que son las acciones seleccionadas a observar del estudiante sobre el código. Esto incluye acciones en el teclado pero también en conjuntos de teclas presionadas que indican una acción sintáctica sobre el código que se puede interpretar, también, semánticamente. Estos datos capturados se almacenan en una base de datos para su posterior análisis con el módulo de análisis.

Figura 4.3

Estructura del software



Los requerimientos reúnen una cantidad de implicaciones para el código que deben producirse para concretarse en una funcionalidad. Las funcionalidades son las observables directa o indirectamente ya una vez que se esté ejecutando el software. Algunas funcionalidades son menores y más bien tienen que ver con la usabilidad o facilidad de uso del software y así no todas son mencionadas aquí.

De los requerimientos, el diseño consiste en la toma de decisiones acerca de cómo convertirlos en funcionalidades concretas del software. Por ejemplo, la necesidad de presentar al programador un ambiente IDE (*Interactive Development Environment*) típico,

fácil de usar e intuitivo, hizo que se incluyera como funcionalidad un espacio para la programación que ocupe la mayor parte de la pantalla, con facilidades de edición de texto propias de la programación como los números de línea y la coloración de componentes léxicos del código. A la vez, la misma razón hizo que se excluyeran otros componentes como la sugerencia de ediciones al código y la información adicional de métodos y clases que suele aparecer en ambientes más complejos.

El diseño define las posibles interacciones del usuario con el software por lo que lleva a decisiones sobre los componentes de interfaz. En el módulo de análisis, por ejemplo, se incorporan botones para la compilación del código y para la ejecución, por tratarse de funcionalidades primordiales para la programación. Otros botones para facilidades más avanzadas como la de depuración o realización de funcionalidades más complejas, como las de refactorización no se permiten en el interfaz por potenciales problemas de interpretación. Por ejemplo, uno de los requerimientos definidos establece que la generación de instrucciones y las funciones avanzadas en el código, en general, no se incluyeron como funcionalidades por tratarse de acciones en el código que pueden, más bien, invisibilizar el proceso mental del aprendiz (según opinión de expertos del estudio 1).

Otro aspecto del diseño fue determinar las maneras principales en las que se puede capturar el tipo de acción del programador sobre el programa que escribe. Algunas de estas acciones requieren de la consideración de la estructura actual del código así como la estructura en un momento inmediato anterior a la acción del programador. Se determinó, en el diseño, que la relación entre el estado del programa previo a la acción y posterior a la acción es clave en múltiples acciones observadas como en la modificación de código dentro de una estructura condicional, por ejemplo.

Por lo que, observando un momento anterior y comparándolo con el actual se puede determinar si se trata de una modificación de código o la generación de código nuevo. Los tipos de fuentes de información se muestran en la Tabla 4.13.

Tabla 4.13

Tipos de fuentes de información para determinar las acciones del aprendiz de programación

Tipo de fuente de información	Descripción
Árbol de análisis sintáctico	Se refiere a la estructura sintáctica del código en un momento dado. Por ejemplo, la estructura considera el grado de anidamiento entre estructuras de control y/o condicionales. Así, un método que tiene dos estructuras de ciclo al mismo nivel tiene menor altura de uno que tenga un ciclo dentro de otro ciclo.
Código fuente	Algunas acciones se pueden observar directamente con el código fuente. Por ejemplo, la modificación de una instrucción de una expresión condicional se puede distinguir de la inclusión de una instrucción nueva comparando el tamaño de ambos códigos.
Mouse/uso de botón	Las acciones de compilación, ejecución y lectura del ejercicio se detectan mediante la interacción del mouse o del teclado con el código.
Teclas de flechas	La navegación en el código se puede determinar con el uso de flechas de manera seguida.

4.3.2 Módulo de Captura

La facilidad de uso, definida como requerimiento en los estudios previos, influyó también en la selección de plataforma de implementación del módulo de captura. El software es capaz de capturar las acciones desde un navegador, al ser un sistema Web, sin necesidad de instalación en las máquinas sea en un laboratorio o en la computadora que esté utilizando el estudiante. Esto aumenta la portabilidad del sistema, hace que pueda correrse en la mayor cantidad posible de escenarios con adaptaciones mínimas. Esto permitirá, a futuro, seguir alimentando el conocimiento posible acerca de los resultados conforme se aplique más veces la herramienta. Cada vez que se registren acciones de usuarios la capacidad de análisis aumenta correspondientemente. Esto es una característica

intrínseca del aprendizaje automático propuesto para el análisis de resultados (mapa auto organizado).

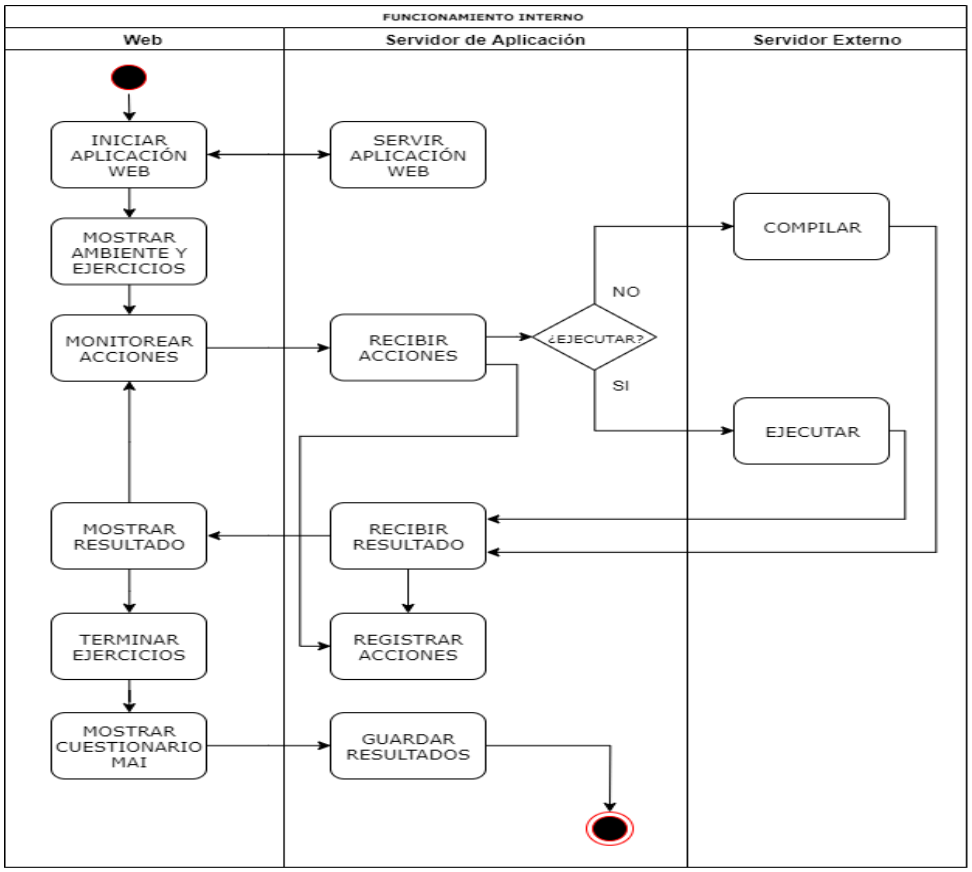
La interacción con el servidor de aplicaciones se logra con una combinación de HTML, Javascript y PHP. Los tres son lenguajes de programación que determinan la manera en la que se implementa la aplicación web. La aplicación web se ejecuta en el navegador del usuario pero se sirve, es decir, se provee, de código almacenado en un servidor de aplicaciones web. Como base de datos se seleccionó SQLite (Hipp, 2020) por ser de software libre que permite el almacenamiento de la información desde el ambiente web, en el servidor y su posterior lectura sin modificaciones de formato en otro ambiente, como para el análisis de los datos.

4.3.2.1 Funcionamiento de la aplicación web

La aplicación web tiene dos componentes principales que es el software que se ejecuta propiamente en el navegador del usuario y el código que se ejecuta en el servidor. El código del navegador inicialmente está en un servidor web y luego pasa al navegador gracias al servidor que provee el código. Las acciones se registran primeramente en el navegador para ser enviadas al servidor de aplicaciones web. Esta interacción y la interacción del servidor de aplicaciones con el servidor externo de compilación se muestra en la figura 4.4.

Figura 4.4

Funcionamiento de la aplicación web



En el módulo de captura la interacción comienza con una pantalla inicial. Esta pantalla incluye una solicitud al usuario para indicar su nivel de dominio en Java. Esto como información preliminar opcionalmente utilizable en el análisis. Se indica también, en esta pantalla, claramente la naturaleza investigativa del ambiente de desarrollo. En todo momento se aclara que la participación es opcional y se procede con la interacción únicamente si el usuario decide continuar. En esta pantalla se incluye también el nombre de CODELEARNER. Este nombre se utiliza en el desarrollo de software. La pantalla principal se muestra en la Figura 4.5.

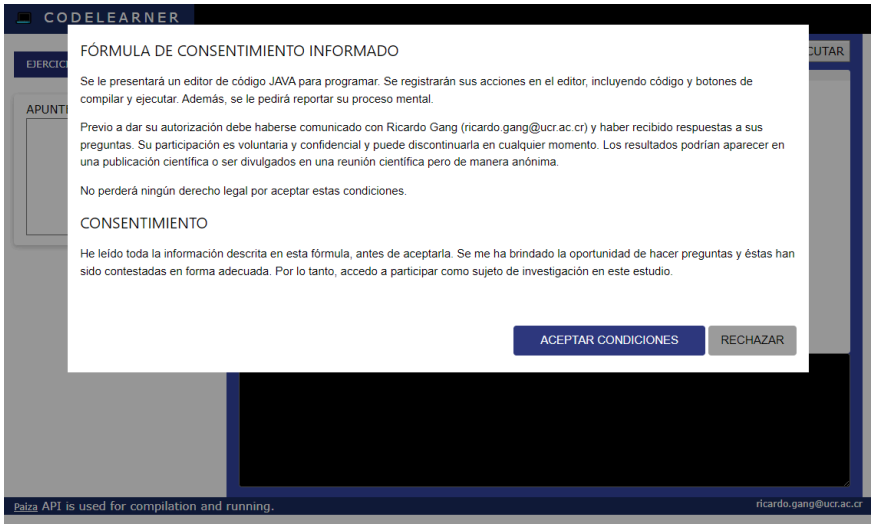
Figura 4.5

Pantalla inicial del módulo de captura

En la Figura 4.6 se muestra el anuncio de solicitud de aceptación de consentimiento informado. Se le informa al estudiante, en esta pantalla, acerca de la participación en la investigación y se le aclara que su participación es voluntaria y puede ser interrumpida en cualquier momento. En caso de rechazar el consentimiento la aplicación redirecciona a la pantalla anterior y en caso de aceptación del consentimiento la aplicación continúa.

Figura 4.6

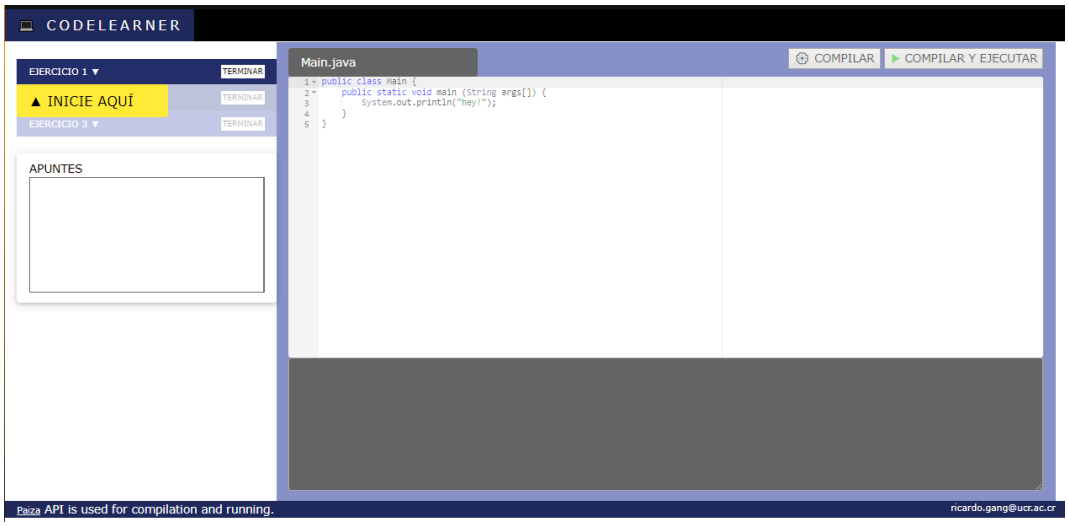
Solicitud de consentimiento informado del módulo de captura



Según requerimientos, y para reducir la complejidad asociada al aprendizaje de un ambiente nuevo se incluyeron guías visuales para ayudar a la comprensión del uso de la aplicación. Por ejemplo, en la Figura 4.7 se muestra un rótulo que resalta en color amarillo el inicio de la actividad con una flecha hacia el primer ejercicio. Los ejercicios se muestran por unos segundos y luego se ocultan. Esto cumple dos propósitos, primeramente el de ordenar el interfaz y reducir a solo componentes estrictamente necesarios para la programación. Adicionalmente, cumple el propósito de permitir registrar las veces que el usuario muestre el ejercicio como indicación de las veces que requiere información para continuar con la programación.

Figura 4.7

Pantalla de inicio del sistema de captura de acciones del programador



En el ambiente de programación se muestra un rótulo similar cuando se le pide al usuario realizar unos apuntes. Esta es una funcionalidad de la aplicación web que le recuerda, mediante un rótulo momentáneo, al usuario registrar su pensamiento sobre las acciones que está realizando en un momento dado. Este recordatorio se muestra de manera

aleatoria, no en el mismo momento y no siempre se muestra. El recuadro para realizar los apuntes está en el interfaz, a mano izquierda y se muestra en la Figura 4.8 (a).

Entre los requisitos se tiene también que el ambiente debe tener los componentes mínimos para la interacción con el código como los números de línea y el coloreo de elementos léxicos y también de funcionalidades de compilación y ejecución. Estas características se muestran en la Figura 4.8 (b). Además, se logra apreciar como el ambiente no tiene mucho más que eso en cuanto a la manipulación de código.

Se excluyen del interfaz, según los requerimientos, otras funcionalidades disponibles en otros ambientes que pueden distraer del ejercicio como funcionalidades para manipular archivos, proyectos, esquemas de código o facilidades de depuración. Esto tiene como propósito enfocar las acciones del estudiante de manera que el registro de las acciones esté más ligado a su proceso de mediación con el código fuente que programa.

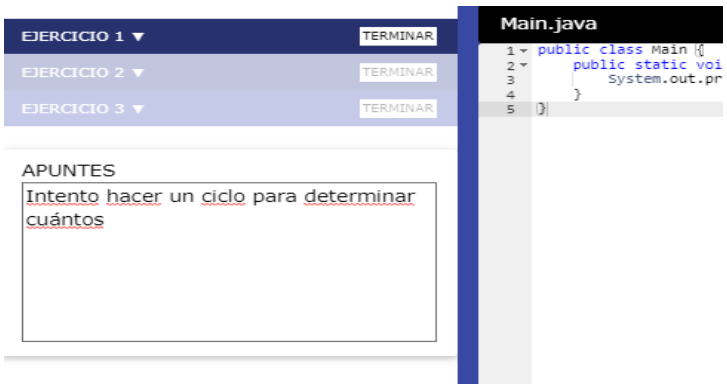
La consola de resultados, que se muestra en la Figura 4.8 (c), es donde se muestran los resultados de compilación y los resultados de ejecución. Los resultados de compilación pueden ser un anuncio de compilación exitosa o una lista de errores encontrados en el código. Los errores de compilación que se muestran en el software de esta investigación son los errores propios del compilador de Java. De esta manera los errores se muestran con el mismo formato ya conocido por los estudiantes. Al ser la lista de errores del compilador normal se incluye también información adicional a cada error útil para la depuración de errores. El proceso de depuración incluye la localización del error y la corrección. Al respecto el software tiene dos funcionalidades relacionadas. Como la información del compilador acerca de los errores incluye la línea de error y la señalización de la columna en la línea donde se encuentra el error entonces el software permite seleccionar un error y esto hace que se ubique el cursor en el código en la línea del error.

La otra funcionalidad adicional en el ambiente con respecto a la señalización de errores es el de registrar que se presentó el error para conocer el momento en el que el error fue detectado por el usuario presionando el botón de compilación y el momento en que el error fue corregido, aún cuando no presiona el botón posteriormente. Esto permite medir la

regulación del usuario ante el conocimiento de que su código presenta un error contabilizando las veces que intentó corregir y las modificaciones realizadas para corregir el error.

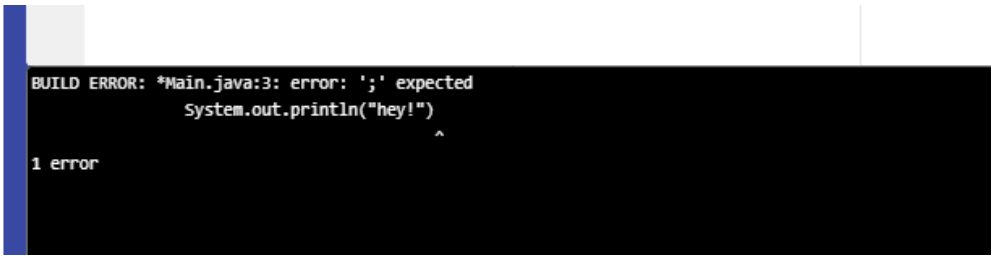
Figura 4.8

Aspectos importantes del interfaz del módulo de captura



a. El recuadro de apuntes permite hacer anotaciones

b. Pocos botones y autoexplicativos



c. Consola que muestra resultados de ejecución y problemas encontrados en la compilación

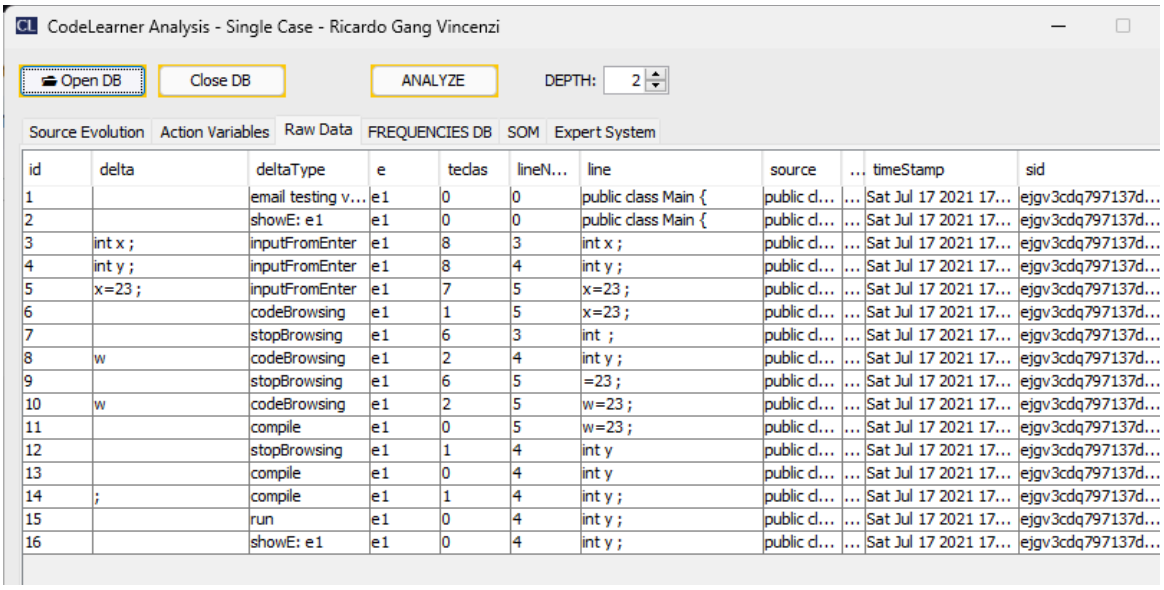
4.3.3 Módulo de Análisis

El módulo de análisis es el componente que hace posible la interpretación de los datos recogidos de la aplicación del módulo de captura. El módulo de análisis cumplió también un proceso de desarrollo de software similar al módulo de captura de datos. La diferencia yace en el uso de las bases de datos, la producción de insumos a las partes del módulo y la aplicación de los algoritmos de análisis.

Una vez registradas las acciones capturadas se almacenan en una base de datos del servidor de datos. Para acciones que cambian el código fuente, el módulo de análisis permite comparar cada cambio producido con el código fuente desarrollado hasta el momento, para identificar ese cambio. De esta forma, la observación es, entonces, también a nivel semántico. Es decir, se realizan observaciones a nivel semántico ya que el sistema es capaz de distinguir una acción que no produce ningún cambio semántico con respecto a las que sí lo hacen.

El módulo de análisis, que se muestra en la Figura 4.9, recoge la información almacenada en base de datos por el módulo de captura. En el módulo de análisis se permite la navegación visual de la información básica de dos maneras principales. En la primera, la información de las acciones de un usuario se puede estudiar viendo directamente los datos de toda la base de datos generada para un usuario. Es decir, se pueden inspeccionar visualmente los datos de todas las acciones. La otra manera es con un novedoso explorador de la evolución del software.

Figura 4.9
Interfaz del módulo de análisis



id	delta	deltaType	e	teclas	lineN...	line	source	...	timeStamp	sid
1		email testing v...	e1	0	0	public class Main {	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
2		showE: e1	e1	0	0	public class Main {	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
3	int x ;	inputFromEnter	e1	8	3	int x ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
4	int y ;	inputFromEnter	e1	8	4	int y ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
5	x=23 ;	inputFromEnter	e1	7	5	x=23 ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
6		codeBrowsing	e1	1	5	x=23 ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
7		stopBrowsing	e1	6	3	int ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
8	w	codeBrowsing	e1	2	4	int y ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
9		stopBrowsing	e1	6	5	=23 ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
10	w	codeBrowsing	e1	2	5	w=23 ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
11		compile	e1	0	5	w=23 ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
12		stopBrowsing	e1	1	4	int y	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
13		compile	e1	0	4	int y	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
14	;	compile	e1	1	4	int y ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
15		run	e1	0	4	int y ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...
16		showE: e1	e1	0	4	int y ;	public cl...	...	Sat Jul 17 2021 17...	ejgv3cdq797137d...

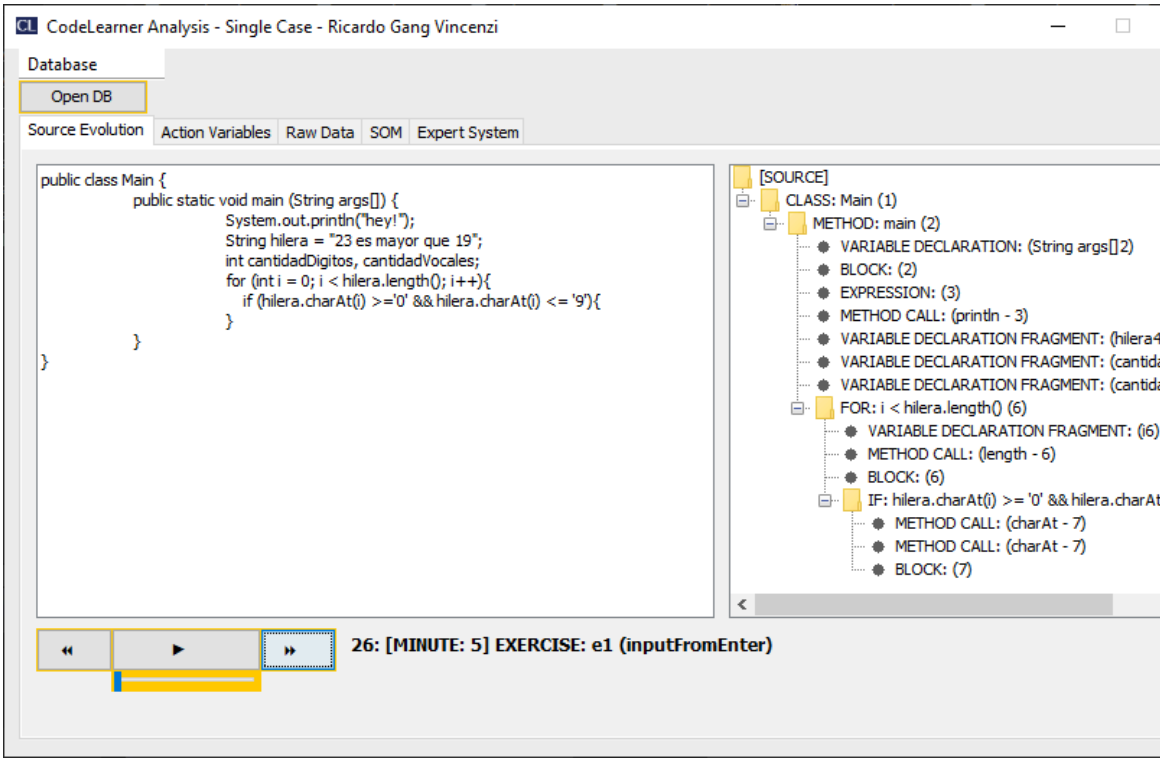
El explorador de la evolución del software, presente en el módulo de análisis, constituye en sí un resultado que surgió de la intención de la investigación de capturar acciones a nivel sintáctico y semántico, además tiene un gran potencial de uso para docentes en el aula o personas investigadoras.

Por otro lado, permite al investigador analizar los cambios en el código según vaya evolucionando hacia el producto final y permite retroceder y avanzar manualmente. Es una herramienta completa en sí misma, capaz de permitir navegar por los cambios registrados en el código y estudiar así el proceso de desarrollo mediante observación. Además tiene un visualizador de estructura de código que permite observar también la profundidad en el código de los cambios que se van realizando. El explorador se muestra en la Figura 4.10.

.

Figura 4.10

Explorador de evolución del código fuente



Toda acción que se considere pertinente queda registrada para su posterior análisis. El análisis propuesto de acciones registradas comprende la recopilación de los datos de múltiples estudiantes en una forma manipulable y comprensible. Como se ha mencionado, la cantidad y la complejidad de la estructura de los datos que se registraron sugieren que técnicas como el análisis de grupos y el aprendizaje automático son las adecuadas para aproximar la evaluación de destrezas metacognitivas mediante la caracterización de los grupos.

Una vez registradas las acciones, se contabilizan las frecuencias correspondientes a las interpretaciones de las acciones. Es decir, cada acción es interpretada como el tipo de efecto producido en el código o de acción mostrada por el usuario. Esta interpretación se realiza en el módulo de análisis y se lleva una cuenta de acciones tomadas. Estos datos son el insumo del análisis. Para cada usuario se puede hacer una evaluación de sus destrezas metacognitivas, indicadas por sus acciones con las reglas de producción construidas a partir de los resultados de los primeros dos estudios. Esta evaluación puede ser comparada con los resultados del instrumento de autorreporte MAI.

Aunque es valioso estudiar los datos de un solo estudiante y así evaluar sus destrezas metacognitivas, este valor aumenta cuando se comparan con los datos ya obtenidos para otros estudiantes. Esto porque identificando grupos de estudiantes con patrones similares en cuanto a sus acciones, permite aproximar la caracterización de las destrezas de esos grupos. Mientras más estudiantes utilicen el software más mejora la precisión de la caracterización de cada grupo identificado. Así, cada aplicación posterior del software va mejorando la posible interpretación de resultados.

El paso inicial del software de análisis es el cargar los datos correspondientes a las acciones de los usuarios del software de captura. Estas acciones están registradas en un archivo de base de datos que debe buscarse con las funcionalidades que aporta el software de análisis. Una vez cargado se puede analizar acción por acción o analizar todas las acciones a la vez. La primera funcionalidad permite verificar la correcta interpretación de las acciones y la segunda, permite realizar el análisis más expedito de múltiples archivos. Mediante la funcionalidad de poder analizar acción por acción, aportada por el navegador de código fuente, es que se permite desarrollar pruebas de las variables a observar. Esta verificación, como se mencionó, aporta confiabilidad a las observaciones.

4.4 Resultados de Estudio 3

Para el tercer estudio se aplicó el software desarrollado a estudiantes de programación, de manera presencial y virtual. Aplicar el software consiste solicitar al usuario ingresar a una dirección web donde corre el módulo de captura, que como ya se

mencionó, tiene forma de editor de código y solicitarle resolver un problema escribiendo un programa. En la primera oportunidad, que fue presencial, se aplicó a 29 estudiantes. De manera virtual se aplicó el software a 20 estudiantes para un total de 49 participaciones. A todos se les solicitó completar el cuestionario MAI al finalizar su programación, en el mismo ambiente. Sin embargo, esta información no fue completada por 9 y 3 estudiantes, respectivamente, por lo que de esos se tiene información de sus acciones, pero no de sus resultados de MAI.

Todos los estudiantes participaron de manera voluntaria y anónima. Se registró la confirmación de participación y consentimiento informado, opcionalmente se solicitó correo electrónico para contacto en caso de detectar algún problema, luego de confirmar el registro de información esos correos fueron eliminados y se agrupó la información en una sola base de datos para la participación presencial y otra para la participación virtual. En la participación presencial se solicitó resolver tres ejercicios, pero en la participación virtual solamente se solicitó resolver un ejercicio.

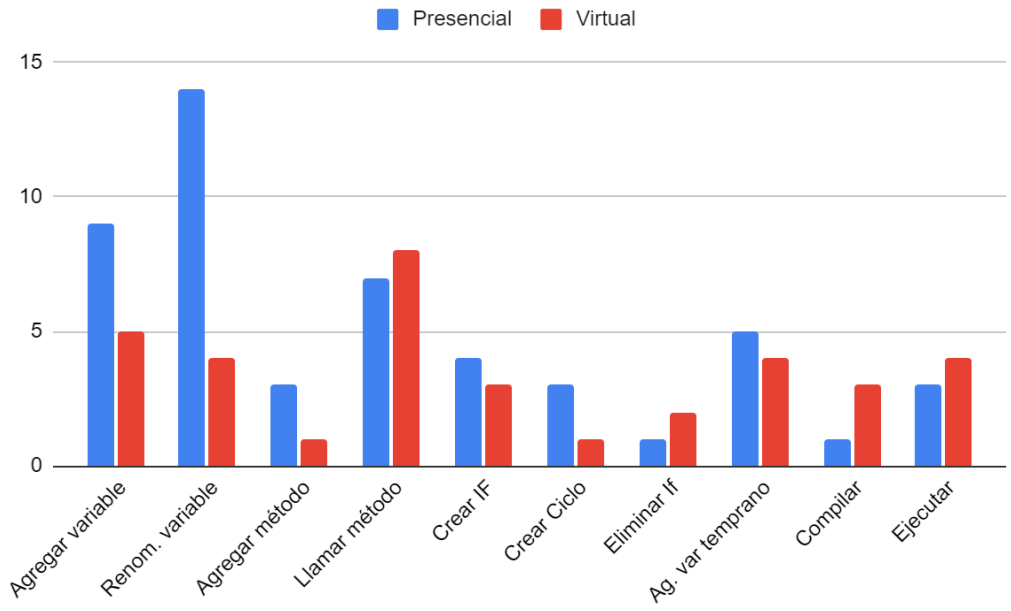
Los estudiantes que participaron presencialmente duraron en promedio 15 minutos y ejecutaron un promedio de 82 acciones en el ejercicio. Los que participaron virtualmente duraron en promedio 20 minutos y ejecutaron 157 acciones con el software. Esto se debió a que de la participación presencial a la virtual se cambió el primer ejercicio por uno ligeramente más complejo, precisamente para intentar generar más acciones. En la aplicación presencial, en promedio, cada acción registrada fue luego de 7 teclazos y los cambios de texto fueron de 7 caracteres. En la aplicación virtual, las acciones se registraron, en promedio cada 5 teclazos y los cambios en el código fueron de 8 caracteres. Teclazos se refiere a cuántas teclas fueron presionadas para producir el cambio y caracteres se refiere a cuántos caracteres están en el cambio efectuado. Estos números no necesariamente coinciden porque al escribir algunos teclazos son de corrección (como espacios, flechas y retrocesos).

Luego del registro de acciones de estudiantes se puede realizar el análisis de los datos recolectados, por ejemplo, para interpretarlas a nivel sintáctico y semántico. Este módulo construye un listado de frecuencias de acciones.

En la Figura 4.11 se muestran los promedios de algunas acciones registradas más comunes. Se puede apreciar que la estructura de los programas fue similar tomando en cuenta sólo estas variables, excepto que el ejercicio en presencial requirió quizás un ciclo más, y más variables. Además, como en presencial se definieron más variables, quizás eso influyó también en que se renombraron más variables.

Figura 4.11

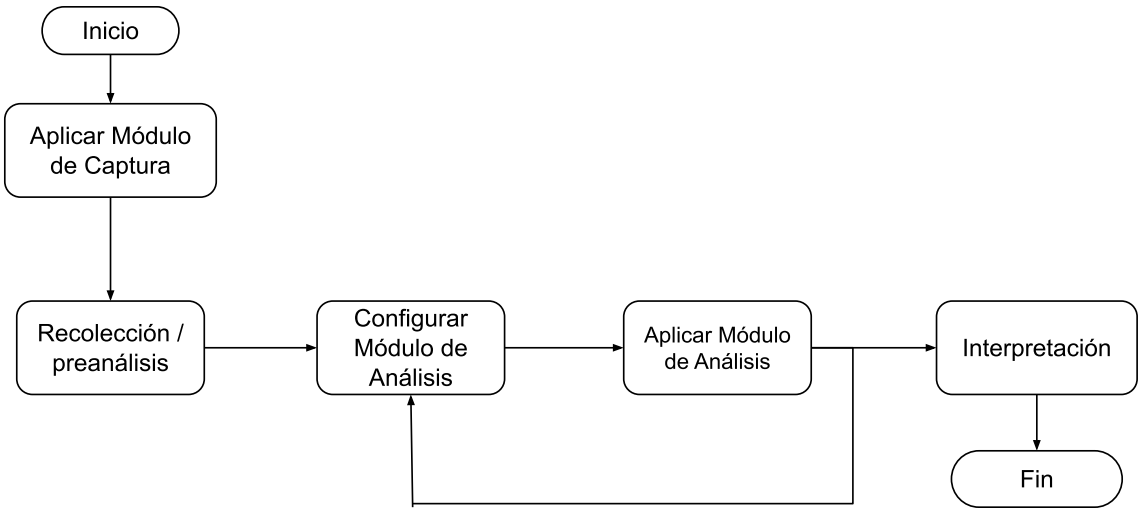
Promedio de algunas de las acciones registradas más comunes



El proceso de análisis de los datos registrados por el software empieza con la aplicación del software, luego la recolección de las bases de datos y su pre análisis, configuración y aplicación de software de análisis, análisis de grupos o *clusters*, y producción de resultados de análisis automático y análisis no automático para finalizar con la interpretación. Las fases de configuración, aplicación y producción de resultados se desarrollaron iterativamente como se describe en la Figura 4.12, al igual que todas las fases mencionadas.

Figura 4.12

Proceso de análisis de datos



Como lo que se estudia son los efectos en el código, la ubicación del estudio es precisamente en el centro de la acción de aprendizaje. Pero al haber muchos cambios en el código, el análisis se vuelve complejo y multidimensional. Se pueden registrar gran cantidad de hechos asociados desde la cantidad de teclas presionadas hasta el nivel sintáctico en el que se produjo el cambio en el código.

Además, se trabajó con las frecuencias de los efectos generados, es decir, para cada programa se cuentan los efectos de cada tipo y eso representa a cada individuo. Por ejemplo, para cada estudiante en total, cuántas variables creó, cuántos ciclos y cuántas veces le modificó el nombre a una variable, entre otras.

El preanálisis es una inspección de los datos manualmente que permite identificar algunas filas en las que por alguna razón hay datos faltantes o incluso columnas en las que no se registraron datos. De haberse encontrado alguno, se hubieran eliminado los datos que no pueden aportar información. La inspección incluyó reconocimiento de las características de los datos. Por ejemplo, qué tan densa es la tabla y cuánta variabilidad hay en los datos recolectados.

Para comprender mejor los datos se puede recurrir a inspección visual y a estudios de variabilidad. Aún recurriendo a las frecuencias de acciones, para altas dimensiones, gran cantidad de variables, la visualización de los datos no es práctica por lo que se puede recurrir a una estrategia de reducción de dimensionalidad y facilitar así la inspección visual. Para esto se utilizó el análisis de grupos (*clusters*), el proceso se conoce como agrupación (*clustering*). También se realizaron análisis indicativos de las variables más influyentes en la variabilidad de los datos, como el análisis de componentes principales (PCA).

4.5 Indicadores de destrezas

Se obtuvieron, a partir de los datos de la aplicación del software, tres indicadores por destreza. Una primera la produce la aplicación del sistema experto difuso. Esto es, basado en el conocimiento representado en el sistema experto difuso se asigna un valor entre 0 y 1 que puede servir para ordenar a los estudiantes según este valor y según la destreza (un valor para planificación, otro para organización y otro para regulación). Igualmente según las acciones etiquetadas por destreza en el estudio 2 y según teoría y observación. Este indicador consiste simplemente en la cantidad de las acciones señaladas

en el estudio 2 cómo relacionadas a cada destreza y normalizando resultados para obtener una evaluación entre 0 y 1. Finalmente, el otro indicador se obtiene con una evaluación por destreza según el resultado del autorreporte MAI para las preguntas asociadas a las mismas destrezas metacognitivas (G. Schraw y Dennison, 1994).

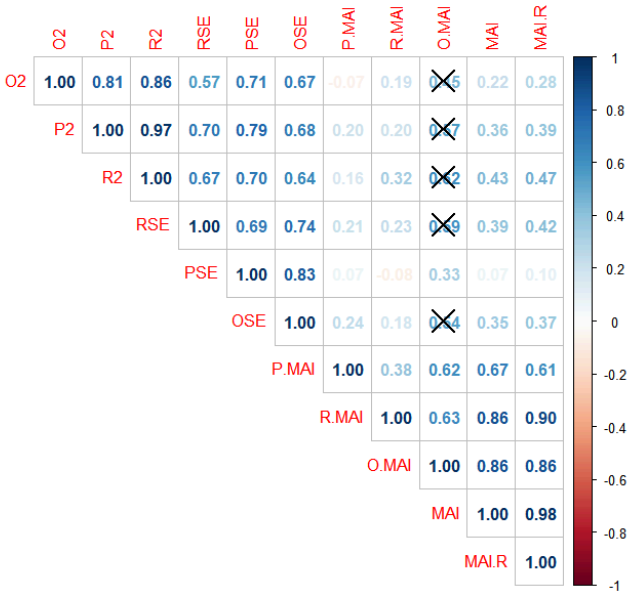
El sistema experto difuso se aplicó para determinar un grado de pertenencia, que se puede entender como evaluación, para las variables difusas de planificación, organización y regulación según fue definido el sistema experto. Esto se hizo para los datos de ambas aplicaciones. En ambos casos, se normalizaron los datos previo a la aplicación del sistema experto por lo que la lógica difusa del sistema es aplicada a estos valores normalizados, igualmente las reglas fueron definidas para funcionar con valores de entrada entre 0 y 1 como ya se mencionó previamente.

Se correlacionan las opiniones de los expertos con respecto a las acciones indicativas de destrezas metacognitivas con las acciones que se registraron y que según la teoría se asocian a destrezas metacognitivas y con las evaluaciones metacognitivas obtenidas mediante el autorreporte MAI.

Para esto, se obtuvieron matrices de correlación de las mediciones obtenidas por el software, según las acciones registradas según la asociación del Estudio 1 aplicando el sistema experto difuso (PSE, OSE y RSE), con las mediciones indicadas por la asociación del Estudio 2, según acciones observadas y teoría (P2, O2 y R2) y el nivel metacognitivo según el reporte y la asociación definida por Schraw y Dennison (G. Schraw y Dennison, 1994).

Figura 4.13

Correlación entre evaluaciones de destrezas metacognitivas para la aplicación presencial.



Nota: Para un valor p=0.1, se muestra la matriz de correlaciones de la primera aplicación. Se incluyen normalizados los datos del MAI y del MAI para las dimensiones de planificación, organización y regulación (P.MAI, O.MAI y R.MAI), y según el sistema experto para las mismas dimensiones de planificación (PSE), organización (OSE) y regulación (RSE) así como el indicador según acciones asociadas a destrezas metacognitivas del estudio 2 (P2, O2 y R2).

De esta tabla de correlaciones se logra apreciar que la tercera evaluación dada por el sistema experto difuso OSE y RSE se correlaciona positivamente con las destrezas de organización y regulación con las evaluaciones del MAI específicas para estas dimensiones denotadas aquí como O.MAI y R.MAI, aunque solo con regulación de manera significativa con un valor p=0.1.

Además, en este caso se obtuvieron valores de correlación positiva de 0.39 con respecto a la evaluación del MAI y de 0.42 con la evaluación de toda la dimensión de regulación del MAI (MAI.R). Con la nota de solo las preguntas de manejos de

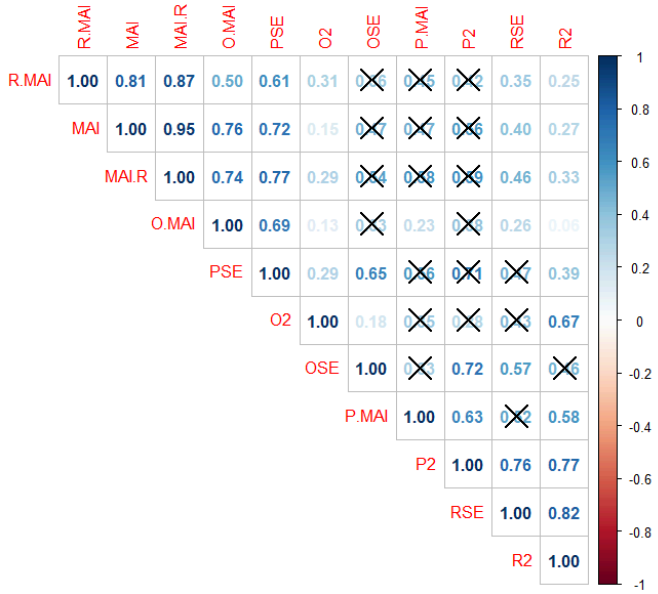
información y estrategias de depuración (IMS y DS del MAI), denotada aquí como R.MAI. Adicionalmente se correlaciona con un valor bajo de 0.23.

En el caso de planificación, la evaluación del sistema experto se correlaciona positivamente pero con valores muy bajos, principalmente con las preguntas específicas de organización (O.MAI), con un valor de 0.33.

Para la segunda aplicación se puede apreciar la tabla de correlaciones en la Figura 4.14.

Figura 4.14

Correlación entre evaluaciones de destrezas metacognitivas para la aplicación virtual.



Nota: Para un valor p=0.1, se muestra la matriz de correlaciones de la segunda aplicación entre las evaluaciones según el MAI (P.MAI, O.MAI y R.MAI), según el sistema experto que representa conocimiento de expertos tomado del estudio 1 (PSE, OSE y RSE) y según acciones asociadas a destrezas metacognitivas del estudio 2 (P2N, O2N y R2N).

Para la segunda aplicación se nota una correlación positiva para la evaluación según el sistema experto y la destreza de planificación (PSE) con las variables de MAI (0.72), MAI.R (0.77) y con la evaluación del MAI solo en O.MAI (0.69), solo en R.MAI (0.61).

Con P.MAI la correlación es positiva (0.55) pero no alcanza el umbral de p=0.1. La correlación con el MAI y con MAI.R es fuerte y significativa

Se aprecia también la correlación positiva para RSE con O.MAI, R.MAI, MAI y MAI.R. En cuanto a OSE las correlaciones son positivas pero no significativas con el umbral de p=0.1.

Otras observaciones pertinentes que se derivan directamente de estas tablas de correlación son, por ejemplo, la relación fuerte y esperada entre las evaluaciones de distintos componentes del MAI tanto entre sí como con la evaluación global del MAI.

Además, la correlación positiva entre el valor de los indicadores de planificación, organización y regulación según el sistema experto y los valores de las tres destrezas según estudio 2 (R2, P2, y O2) para ambas aplicaciones aunque más fuertes para la primera aplicación.

4.5.1 Análisis de grupos

El análisis de grupos, además de servir para la inspección visual mediante reducción de dimensionalidad, permite también la caracterización de grupos, y así de estudiantes según el grupo al que pertenece (Rapkin y Luke, 1993). La caracterización o creación de perfiles a traves de análisis de grupos es útil por su versatilidad pues es aplicable aún sin conocer las acciones que se tomarán (Antonenko et al., 2012). Esto también permite relacionar los resultados de los estudios al realizar la agrupación según las acciones registradas (definidas en estudios 1 y 2 como relacionadas a destrezas metacognitivas) y la caracterización según las reglas difusas definidas para el sistema experto.

La base para la identificación de grupos o conjuntos en los que se pueden agrupar a los estudiantes son los registros de acciones de múltiples estudiantes. Con los grupos identificados, el análisis posible es entonces el de la descripción de cada conjunto según las habilidades metacognitivas prevalentes en cada categoría. La posibilidad brindada por el módulo de análisis es la de estudiar habilidades metacognitivas en grupos de estudiantes

que actuaron similarmente. Por ejemplo, estudiar si prevalecen en ellos habilidades metacognitivas se tiene entonces una manera de categorizar al grupo de estudiantes y así también a un estudiante según sus acciones.

Para verificar la cantidad de grupos ideal para un juego de datos se pueden utilizar el método del codo y el método de silueta (Everitt et al., 2010), y la meta es buscar en los gráficos generados por estos método el punto a partir del cual se empieza a estabilizar la línea y escoger el menor número de grupos que sea útil (con muchos grupos puede haber sobre acoplamiento a los datos y con pocos grupos puede haber sub acoplamiento) (2010).

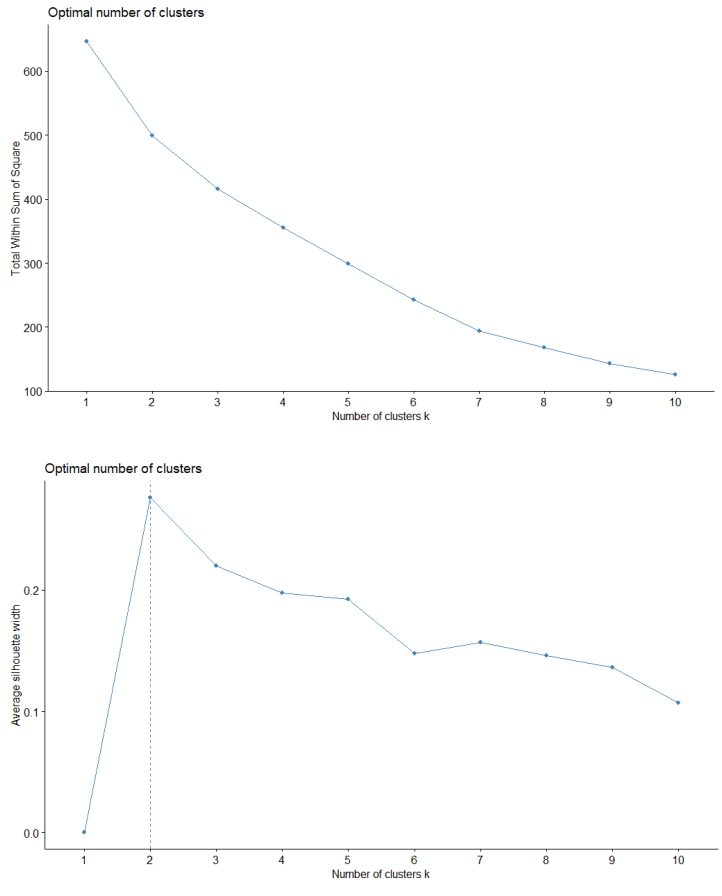
El método del codo sirve para comparar qué tan compactos están los grupos según distintas agrupaciones y de esa manera, se busca grupos más compactos. El método de silueta se basa en la comparación del promedio de cada punto a los otros puntos en el mismo grupo con el promedio de cada punto con los puntos del grupo más cercano.

Se busca maximizar esa distancia puesto que una mejor agrupación es dada cuando los puntos de un grupo están muy cercanos pero también lejanos de los otros grupos. Para utilizar estos métodos se pueden graficar los valores del coeficiente para distintos números de grupos y ver el punto a partir del cual, empieza a estabilizarse el gráfico.

La figura 4.15(a) muestra el gráfico de silueta para la primera aplicación. En la Figura 4.15 (b) se muestra el gráfico de silueta para la misma aplicación. En ambos casos se muestra que a partir de 2 grupos sucede el mayor cambio en tendencia que luego se estabiliza aunque es más evidente en el gráfico de silueta (b). El siguiente cambio mayor sucede a partir de 5 en el gráfico de silueta lo que indica que un rango de entre 2 y 5 es ideal para la agrupación. El número de 4 sugerido por la agrupación jerárquica calza en este rango.

Figura 4.15

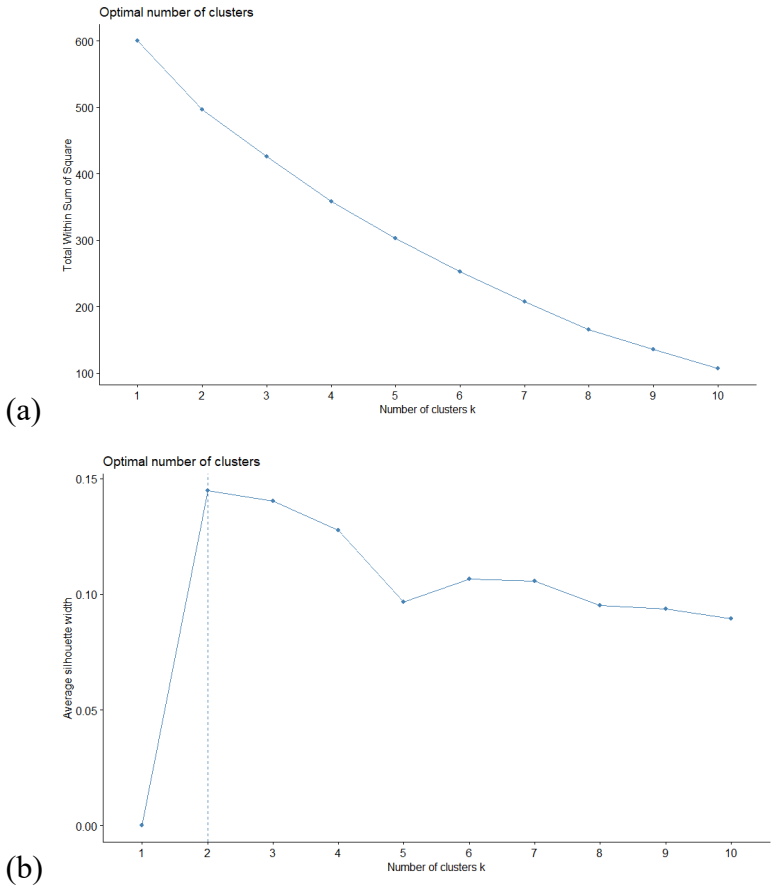
Gráfico de codo y de silueta de la primera aplicación para tres y cuatro grupos.



La figura 4.16(a) muestra el gráfico de codo para la segunda aplicación. En la Figura 4.16 (b) se muestra el gráfico de silueta para la misma aplicación. Similarmente, aunque no tan evidente el mayor cambio sucede a partir del 2, donde esta observación respalda lo mencionado acerca de un grupo para caracterizar y el resto considerarlo otro grupo. El siguiente mayor cambio sucede a partir de 5 en el gráfico de silueta, por lo que el número ideal de grupos está entre 2 y 5, lo que coincide con la observación de la agrupación jerárquica de que con cuatro grupos se logra al menos uno con suficientes individuos para caracterizar el grupo y un número similar de individuos fuera del grupo.

Figura 4.16

Gráfico de codo (a) y de silueta (b) la segunda aplicación.



Es posible que un algoritmo de identificación de grupos encuentre una manera de clasificar individuos aun cuando no exista por lo que es imperativo combinar distintos algoritmos y hacer un análisis más robusto (Antonenko et al., 2012). Una manera de clasificar los algoritmos de agrupación es desde la perspectiva de la estrategia del algoritmo. Esta clasificación incluye, pero no se limita a, agrupaciones por densidad, jerarquía, particiones y modelos (Greenlaw y Kantabutra, 2013; Lior, 2008).

Es necesario recalcar que es posible medir la diferencia entre grupos para llegar a una mejor agrupación (donde los grupos estén más lejanos entre sí y/o más compactos a lo

interno), esto se conoce como disimilitud entre grupos (Everitt et al., 2010). El método de creación de clusters jerárquicamente conocido como "Ward" produce grupos compactos (muy similares a lo interno) y distantes (muy disímiles entre grupos) (Frades y Matthiesen, 2009). Funciona uniendo subgrupos minimizando la varianza interna. En comparación, el método conocido como completo tiende a producir grupos compactos pero quizás más cercanos (2009).

La agrupación por densidad se basa en la densidad de datos alrededor de puntos en particular, DBSCAN es un ejemplo (Khan et al., 2014; Schubert et al., 2017). La agrupación de jerarquía puede ser divisiva en el sentido de que parte de todos los datos en un mismo grupo y trata de dividirlos según alguna métrica y es contraria a la agrupación jerárquica aglomerativa que intente unir datos según alguna métrica (Everitt et al., 2010). En cualquier caso, el resultado de una agrupación iterativa así es un gráfico conocido como "Dendrograma" que indica la agrupación realizada.

La agrupación por jerarquía puede utilizarse para conocer la cantidad de grupos en cada aplicación. Los dendrogramas del ejercicio 1 de la primera aplicación se muestran en la Figura 4.17. La Figura 4.17 (a) sugiere cuatro o cinco agrupaciones dependiendo del nivel de profundidad al que se quieran hacer agrupaciones. En los extremos se pueden hacer agrupaciones de un solo individuo por grupo pero el objetivo es más bien buscar grupos homogéneos entre sí suficientemente distintos entre grupos para separarlos (Rapkin y Luke, 1993).

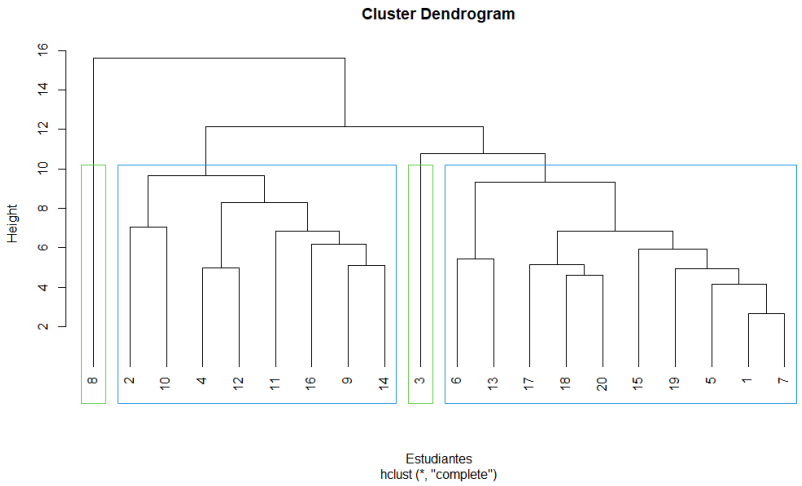
Sin embargo, una de estas agrupaciones sólo incluye una observación por lo que más bien se puede tomar la decisión de seleccionar tres o cuatro como el número de agrupaciones puesto que ese dato posiblemente es un dato no agrupable. La decisión entre tres o cuatro, en este caso, se basaría en cuánto balance se desea ya que, según se aprecia en la figura, eso significa partir el grupo más numeroso en dos.

Se observa que, según el método conocido como completo para la primera aplicación, el estudio de clustering jerárquico sugiere cuatro agrupaciones (cuatro son recalcadas intencionalmente). Sin embargo, solamente dos de estas agrupaciones reúnen a

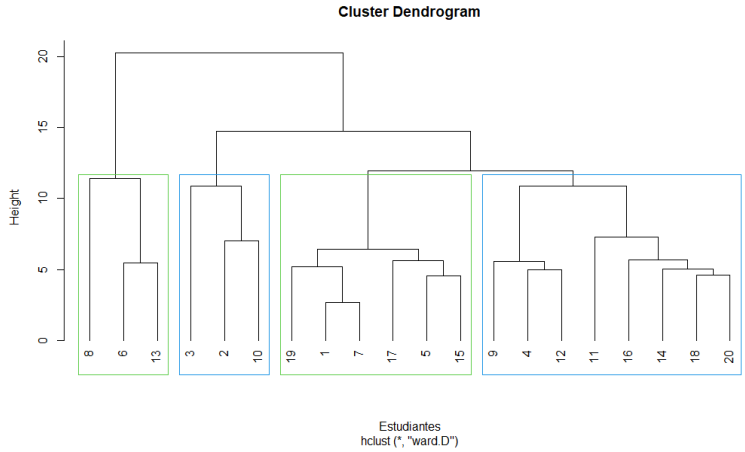
varios estudiantes por lo que, buscando balance entre grupos y mayor cubrimiento del total de estudiantes solo dos grupos son principales para analizar. En la Figura 4.17 (b) se muestra la agrupación realizada por el método Ward que reduce varianza a lo interno y maximiza distancia entre grupos.

Figura 4.17

a. Dendrograma de la primera aplicación con el método conocido como completo.



b. Dendrograma de la primera aplicación, con el método de enlace ward.D.

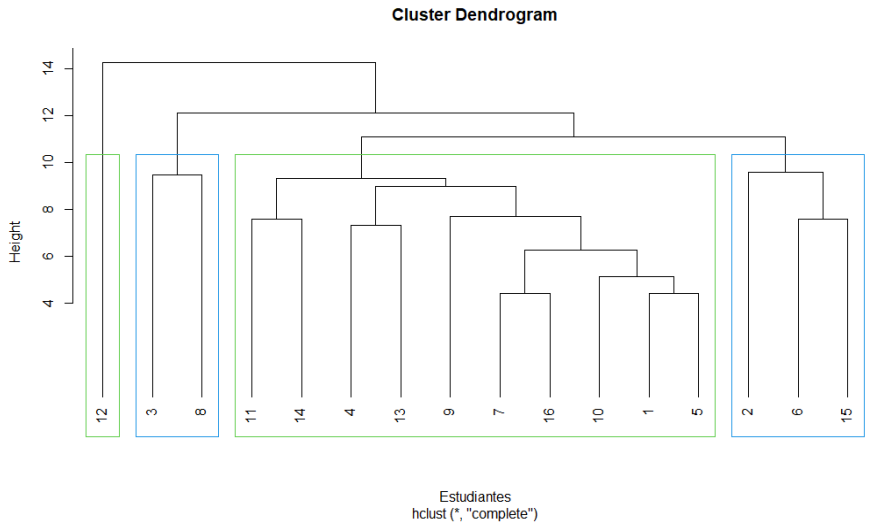


En esta agrupación se tiene la ventaja de que la distancia entre grupos ha sido ya maximizada y, también por algoritmo pues se utilizó el método Ward, la similitud entre los individuos de un mismo grupo ha sido maximizada. Dos de los grupos identificados son

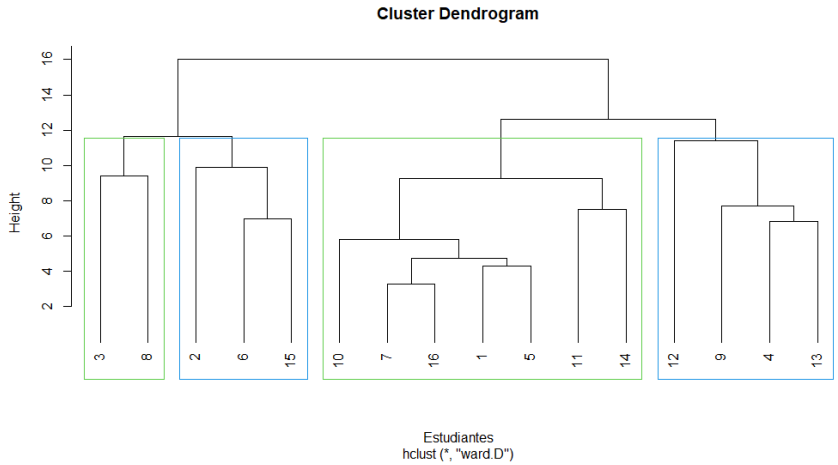
pequeños, de solo tres estudiantes por lo que la caracterización del grupo es más débil. Los otros dos grupos tienen a los individuos 1,5,7,15,17 y 19 y el otro 4,9,11,12,14,16,18 y 20.

Figura 4.18

a. Dendrograma de la segunda aplicación con el método conocido como completo.



b. Dendrograma de la segunda aplicación



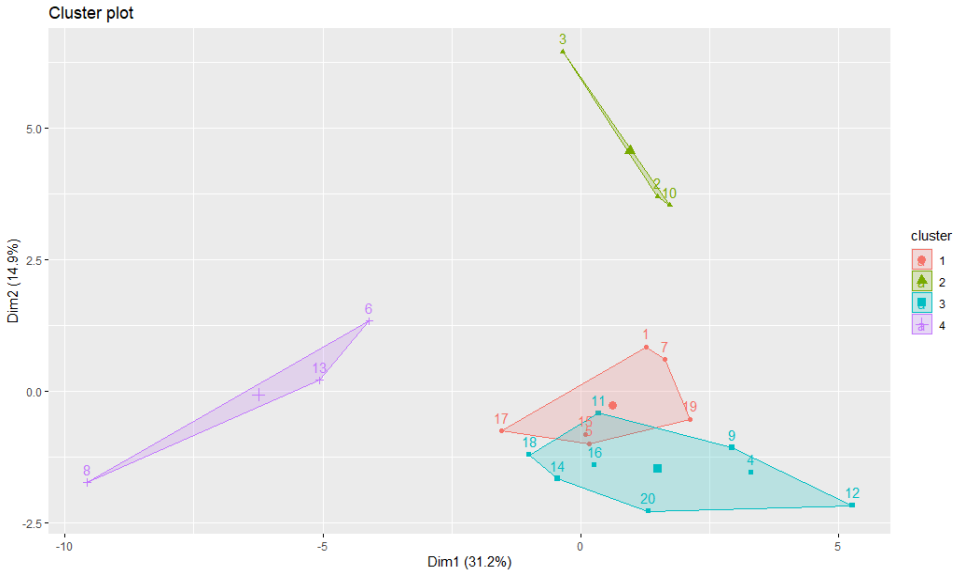
En la Figura 4.18 se muestran los dendrogramas de la segunda aplicación, primeramente el producido con el método completo en la Figura 4.18 (a) que muestra un solo grupo grande con otros tres grupos de 3 o menos individuos. En la Figura 4.18 (b) se

muestra la agrupación producida por el método Ward que encuentra los mismos grupos. Haciendo una agrupación, suponiendo que hay 5 grupos solo partiría el grupo grande separando 3 sujetos (según la ramificación) por lo que tampoco se lograría tener dos grupos balanceados grandes. En el caso de la segunda aplicación conviene la agrupación de 3 y 8 en un primer grupo, 2, 6 y 15 en otro, 1, 5, 7, 10, 11, 14 y 16 en un tercer grupo y 4, 9, 12 y 13 en un cuarto grupo.

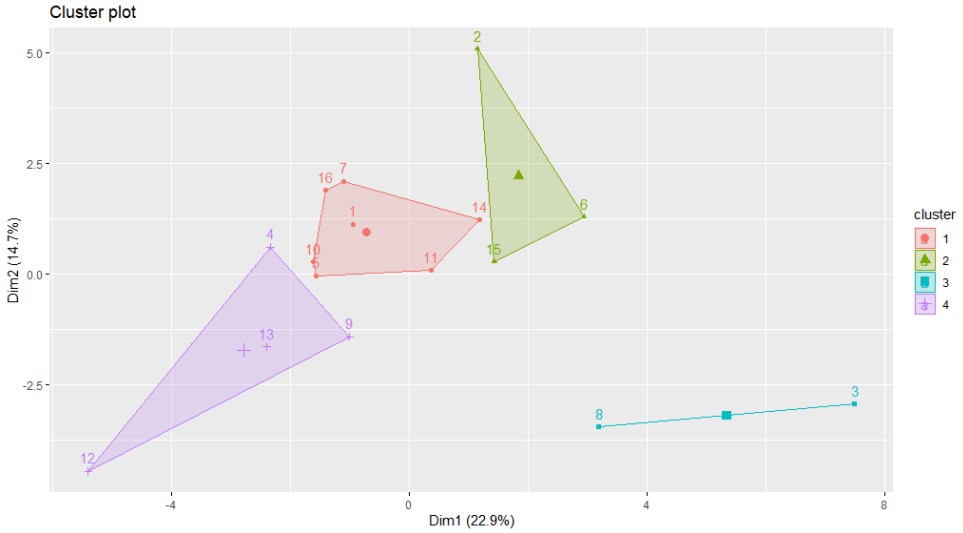
Estas agrupaciones se pueden visualizar utilizando los componentes principales para la ubicación de los puntos en un gráfico. En la Figura 4.19 (a) se muestran los grupos sugeridos por la agrupación con el método Ward para la primera aplicación y en la Figura 4.19 (b) se muestra la agrupación sugerida por el mismo método para la segunda aplicación. Para la primera aplicación se muestra un traslape entre grupos que se da por la imposibilidad de determinar una completa separación de grupos de ciertos individuos. Es decir, algunos individuos podrían estar en ambos grupos. En el contexto de la investigación esto es esperable pues más bien no se esperaría que las destrezas mostradas por estudiantes sean totalmente disímiles sino que compartan fortalezas y debilidades. Para el caso de la segunda aplicación se puede apreciar que los grupos están más separados e incluso el grupo grande se nota más compacto que los otros tres que están más dispersos. Esto respalda la observación de que conviene únicamente caracterizar este grupo.

Figura 4.19

a. Visualización de grupos utilizando agrupación jerárquica con el método Ward para la primera aplicación.



b. Visualización de grupos utilizando agrupación jerárquica con el método Ward para la segunda aplicación.



4.5.2 Mapa Auto Organizado

Adicional a las técnicas ya mencionadas para el análisis de grupos, en la investigación se utilizó la forma de agrupación de estudiantes determinada por un algoritmo de búsqueda de clases, o grupos (*clusters*), que implementó aprendizaje automático no supervisado. En particular, se utilizó la técnica de identificación de grupos conocida como "mapa auto organizado" (SOM por sus siglas en inglés, *Self Organizing Map*) (Asan y Ercan, 2012; Kohonen, 1990, 2013; Ponmalai y Kamath, 2019; Yin, 2008).

Con este algoritmo se identifican grupos en los datos de alta dimensionalidad, que en este caso corresponden a los datos de las acciones observadas en múltiples estudiantes, según un proceso de identificación y agrupación de datos similares (según las distancias entre individuos) en una red de neuronas artificiales.

La dimensión de la red es libre pero ciertas recomendaciones son útiles, como que se use una matriz hexagonal para la visualización, que sea bidimensional y que está bien que algunas neuronas queden vacías luego de la agrupación (Kohonen, 2013).

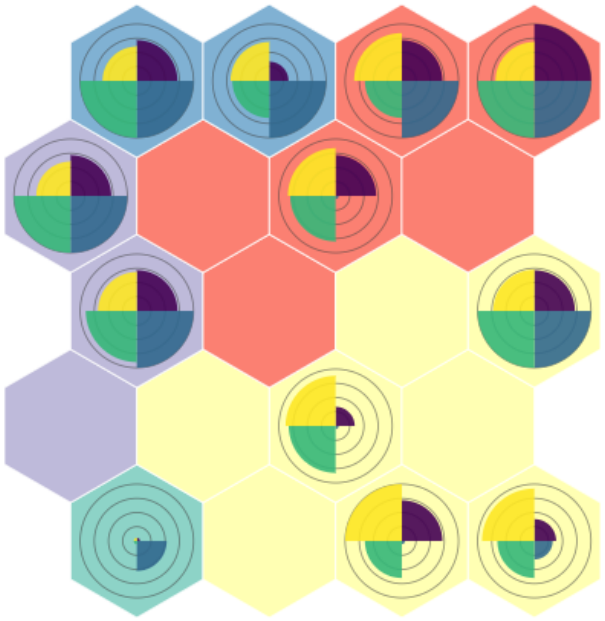
En la Figura 4.20, se muestra el resultado de haber entrenado un mapa autoorganizado, sin supervisión, en los datos de la primera y segunda aplicación. El algoritmo acomoda en una matriz predeterminada, con celdas de forma hexagonal, los individuos según el parecido que tengan en un proceso iterativo de entrenamiento basado en vecindarios grandes que van decreciendo con el tiempo.

Al final del entrenamiento, cuando ya no se realizan más movimientos de individuos es posible visualizar los individuos que pertenecen a cada grupo. Algunas veces los grupos están visiblemente separados como en este caso, otras veces es más difusa la separación. El fondo de los hexágonos en estos mapas está con color verde, amarillo, rojo y lila marcando grandes regiones del mapa indicadas por la agrupación. Esta es una manera de facilitar la visualización. Cada hexágono representa a una neurona artificial a la que se le asocian datos para agrupar los parecidos y alejar los distintos.

Figura 4.20

a. Mapa autoorganizado entrenado con los datos de la primera aplicación.

b. Mapa autoorganizado entrenado con los datos de la segunda aplicación.



Para la primera aplicación, el mapa auto organizado de la Figura 4.20 (a), agrupó en las neuronas de fondo amarillo a los datos de los estudiantes 2, 10 y 3 confirmando la agrupación por jerarquía. Estos coinciden en buenos niveles de regulación, y medianos niveles de planificación y organización, según el sistema experto, así como en mediano nivel de destrezas según el MAI. En la esquina del mismo lado pero superior, el mapa auto organizado confirmó también la agrupación de los datos de los estudiantes 6, 8 y 13 en neuronas de fondo rojo. Como se puede apreciar en la figura, éstas se relacionan por sus altos niveles de planificación, regulación y organización según el sistema experto y también por su alto nivel de destrezas metacognitivas según el MAI para la dimensión de regulación.

El siguiente grupo, está en las neuronas de fondo verde, y está integrado por los individuos 1, 7 y 19 al extremo izquierdo, adyacente el 5 y luego el 15 y finalmente en el mismo grupo el 17. Esta agrupación coincide con la agrupación realizada previamente. El mapa agrupa (en neuronas de fondo lila) los datos de los estudiantes 4 y 12 al extremo izquierdo junto con los datos de los estudiantes 9, 11, 14, 16, 18 y 20. Esta agrupación es distinta únicamente por el 18 que en este caso lo incluye y confirma la agrupación de los demás.

Como ya se hizo para los primeros dos grupos, se puede caracterizar cada grupo visualizando en esta figura los valores asignados por el sistema experto a estos grupos y la relación que pueden guardar con la evaluación de las destrezas metacognitivas según el MAI para la dimensión de regulación. Así, el grupo de fondo verde presenta mediano nivel de planificación, y bajos niveles de organización y regulación según la evaluación del sistema experto y muy variado nivel de destrezas según el MAI. El grupo de fondo lila presenta de mediano a alto nivel de destrezas según el MAI, excepto por el estudiante 20 y todos comparten en común un mediano nivel de planificación según el sistema experto excepto por el 4 y 12.

Para el caso de la segunda aplicación, el mapa que se muestra en la Figura 20 (b), también arrojó un resultado muy similar a la agrupación jerárquica con una leve diferencia.

Los grupos 3,8 y 2, 6 y 15 sí quedaron asociados a una sola región (de fondo azul y rojo respectivamente). Los datos de los estudiantes 2, 6 y 15 quedaron muestran el mayor nivel de planificación, 2 y 15 un alto nivel de regulación y los tres un mediano a alto nivel de organización según el sistema experto. Además, los tres muestran un nivel de regulación mediano a alto, según el MAI.

Asociados a la misma región, de fondo amarillo, los datos de 1, 5, 7, 10 y 16 están cercanos pero no adyacentes a los datos de los estudiantes 11 y 14. Se puede apreciar que esto seguramente se debe al alto nivel de estos últimos de regulación. Esta agrupación confirma, además de las anteriores, la agrupación según jerarquía. Al igual que en la agrupación por jerarquía el mapa agrupó los datos de 4, 9 y 13, con la diferencia de que el 12 que se había previamente agrupado. El mapa permite visualizar que esto seguramente se debe a sus bajos niveles de organización, planificación según el sistema experto y de regulación según el MAI.

En la segunda aplicación, el mapa facilita por medio de sus celdas vacías la separación de grupos que facilita su comprensión. Esto se debe a la baja densidad del mapa pues en este caso hay más neuronas que datos, ambos mapas podrían aumentar su valor analítico aumentando la cantidad de datos que agrupan . muestra que un balance entre la evaluación de regulación y organización según el sistema experto se asocia con buenos niveles de regulación según el MAI. La tabla 4.14 muestra la caracterización de grupos, realizada por inspección visual, según la evaluación determinada por el sistema experto difuso, para ambas aplicaciones y según la agrupación realizada por el algoritmo de mapas autoorganizados. Al ser las agrupaciones tan similares (difieren en tan solo un individuo por aplicación), la caracterización de los grupos será correspondientemente también similar.

Tabla 4.14

Agrupaciones para las dos aplicaciones según el mapa autoorganizado y caracterización de grupos según destrezas observadas por el sistema experto.

Aplicación	Individuos	Destrezas apreciables visualmente
Primera	Grupo 1: 1, 5, 7, 15, 17, 19 Grupo 2: 2, 3, 10 Grupo 3: 4, 9, 12, 14, 16, 18, 20 Grupo 4: 6, 8, 13	Grupo 1: Bajo nivel de organización, mediano a bajo nivel de planificación y bajo nivel de regulación según la evaluación del sistema experto. La mitad de este grupo muestra mediano nivel de regulación y el resto bajo nivel, según el MAI. Grupo 2: Se aprecia alto nivel de regulación, mediano nivel de planificación, mediano a alto nivel de organización según el sistema experto y mediano nivel de regulación según MAI. Grupo 3: Comparten un mediano nivel de destrezas según el MAI y un mediano nivel de planificación según el sistema experto y un mediano a alto nivel de destrezas según el MAI. Grupo 4: Se aprecia altos niveles de planificación, regulación, organización y correspondientemente de regulación según MAI para la regulación.
Segunda	Grupo 1: 1, 5, 7, 10, 11, 14, 16 Grupo 2: 2, 6, 15 Grupo 3: 3, 8 Grupo 4: 4, 9, 13	Grupo 1: presenta el mayor nivel de regulación según el MAI pero menores niveles de regulación, planificación y organización según el sistema experto por lo que debe considerarse la posibilidad de un error de reporte en el MAI. Grupo 2: Muestra mayores niveles de planificación y medianos niveles de organización y regulación según el sistema experto además de un buen nivel de regulación según el MAI. Grupo 3: muestra bajo nivel de planificación y organización según el sistema experto, correspondientemente muestra bajo nivel de regulación según el MAI. Grupo 4: Muestra altos niveles de regulación según experto pero muy variados niveles de las demás variables.

Una fortaleza de este tipo de análisis está en la evaluación de nuevos estudiantes que llegan a utilizar el sistema. Cuando esto sucede se puede aplicar el software y obtener una lectura de sus acciones registradas, con esto ya se puede hacer una evaluación rápida de la ubicación del estudiante con respecto a los otros grupos. Es este tipo de evaluación visual rápida la que puede permitir a un docente en el aula ubicar el comportamiento de un estudiante y valorar posibles apoyos metacognitivos en mínimo tiempo.

El mapa autoorganizado puede ser una herramienta útil que además tiene la posibilidad de aumentar su precisión si a estudiantes nuevos también se les aplica el cuestionario MAI, u otras técnicas de evaluación de metacognición. Esto le permite seguir entrenando el sistema para mejorar la evaluación proporcionada. Con más datos la delineación de grupos puede mejorar y así la asignación de estudiantes nuevos a grupos.

4.5.3 Acciones y destrezas

Según los estudios 1 y 2, algunas acciones se relacionan más directamente con algunas destrezas metacognitivas. Por ejemplo, en la Figura 4.21 se muestran las acciones más correlacionadas con destrezas.

Planificación	Usar y agregar variables, agregar instrucciones, agregar métodos y llamar métodos
Regulación	Ejecutar, compilar, reorganizar código, eliminar condicionales y usar variables
Organización	Agregar métodos y variables, usar variables, modificar nombres de variables, llamar métodos y ejecutar

Planificación

El sistema de inferencia difusa permitió reunir el conocimiento de los expertos con la interpretación de los datos recogidos por el software de captura de acciones. Al aplicar las reglas difusas al resumen de frecuencias de las acciones se obtiene una aproximación a las habilidades mostradas en las acciones y este hecho se verifica en las agrupaciones al presentarse mayorías de un color en algunos grupos. De no haber correspondencia entre los resultados de las técnicas se esperaría mayor dispersión en la coloración de todos los grupos. Esto se detalla a continuación.

Los datos recogidos por la herramienta de captura de acciones de los usuarios presentan patrones consistentes con lo que se esperaría ver como reflejo del registro de información representativa de las habilidades presentes en los estudiantes. Es decir, al evidenciarse la posibilidad de identificar grupos en la información, y verificable con la inferencia difusa basada en información de expertos, se concluye que la herramienta registra efectivamente información indicativa de las habilidades metacognitivas. Esto se muestra en la Figura 4.20 (a) para el caso de la primera aplicación de la herramienta. En la Figura 4.20 (b) se muestra la misma información pero en tres dimensiones que permite visualizar mejor la separación entre grupos. Es importante recalcar que la coloración de los datos en los gráficos no es obtenida del algoritmo, es obtenida independientemente del mismo mediante la aplicación del sistema de inferencia difusa descrito anteriormente. Este hecho permite corroborar otro gran resultado de la investigación, la aplicabilidad de la técnica del sistema de inferencia difusa al análisis de los resultados.

La identificación de grupos se basó en algoritmos de agrupación jerárquica, aprendizaje automatizado no supervisado (mapas autoorganizados). Los algoritmos jerárquicos producen como resultado dendrogramas basados en la distancia entre valores de los arreglos correspondientes a las frecuencias de acciones registradas entre múltiples estudiantes. Estos gráficos constituyen una primera agrupación posible siempre y cuando, como en el caso de esta investigación pueden producir grupos balanceados.

Capítulo 5 - Conclusiones y Recomendaciones

Las destrezas metacognitivas determinan las acciones tomadas en regulación de la tarea que se realiza, según el conocimiento metacognitivo y las experiencias metacognitivas (Efklides, 2008; M. Veenman et al., 2006). Observando y registrando las acciones tomadas en aprendices de programación existe potencial para evaluar el uso de destrezas metacognitivas. El desarrollo de software, que registra acciones indicativas de destrezas metacognitivas, fortalece la enseñanza de la programación, pues brinda la oportunidad de estudiar el uso de estas destrezas en relación con tareas específicas del proceso que lleva el estudiante para programar su solución a un problema.

En esta investigación se realizaron distintas tareas conducentes al cumplimiento de los objetivos propuestos. La metodología propuesta para tal fin incluyó la realización de tres estudios con insumos y resultados esperados muy claros que se articularon para cumplir esos objetivos.

Primeramente se realizó la identificación de interacciones posibles con la computadora que un estudiante de programación principiante puede realizar sobre el código y que pudiera dar información acerca de sus destrezas metacognitivas (Objetivo Específico 1).

Esto se cumplió en parte con el Estudio 1 que incluyó entrevistas a expertos que señalaron los momentos o situaciones en las que la regulación metacognitiva podría vislumbrarse según los requerimientos de la solución de un problema y las acciones posibles de los programadores. La lista de acciones a observar se complementa con el Estudio 2 que incluyó la observación de estudiantes programando.

El diseño y desarrollo del software para registro de acciones indicativas de destrezas metacognitivas (Objetivo Específico 2) se cumplió con el insumo de los estudios

1 y 2 que además de identificar acciones a observar también sirvieron para detallar y enumerar requerimientos del software que sirvieron de base para el diseño.

En el Estudio 3 se lograron identificar acciones específicas a observar que presumiblemente indican destrezas metacognitivas (Objetivo Específico 3). Esto es, se encontró cuáles efectos en el código podrían relacionarse más con destrezas metacognitivas mediante la comparación de las acciones tomadas. Luego esta clasificación se validó con la evaluación metacognitiva realizada con un instrumento externo (MAI). De esta manera, se interpretaron las relaciones encontradas entre acciones tomadas que afectan el código y las destrezas metacognitivas (Objetivo Específico 4).

La alta dimensionalidad de los datos hace que su análisis requiera de técnicas de exploración de datos que pueden incluir aprendizaje de máquinas o análisis estadístico. En esta investigación se siguió un enfoque de aprendizaje de máquinas que cumplió un proceso que se resume así, inicialmente se estudió la topología de los datos en crudo, sin agrupaciones ni decisiones sobre variables ni reducción de dimensionalidad.

Luego se estableció la necesidad de identificar técnicas de análisis. Seguidamente se aplicaron estas técnicas, conducentes a la reducción de la dimensionalidad. Esta reducción de dimensionalidad constituye una clasificación de los datos que permite el análisis y la caracterización de los grupos encontrados.

En resúmen, se construyó un software que registra acciones tomadas por estudiantes de programación, que además sirvió para identificar cuáles de estas acciones producen efectos en el código sugerían que se llevó a cabo autorregulación indicativa de destrezas metacognitivas. Esto sirve de base para fortalecer la enseñanza de la programación.

5.1 Discusión Estudio 1

El estudio de las destrezas metacognitivas debe considerar el dominio de conocimiento en el que se evalúa. Esto por la relación estrecha entre el dominio y el conocimiento metacognitivo (Brown, 1987). Esto implica que quiénes mejor conocen las

acciones de los estudiantes en una tarea en particular, como programación, algunas de las cuáles puedan indicar destrezas metacognitivas, deben tener también un profundo conocimiento del dominio. La identificación de estas acciones a observar para el dominio de programación, entonces, debe empezar con insumos de la experiencia de expertos como los docentes de programación, asistentes de curso, y estudiantes (novatos y avanzados).

En el dominio de programación, las acciones que los estudiantes pueden realizar en la resolución de un problema con código comprenden un conjunto finito, relativamente pequeño (menos de 50). De éstas, no todas indicarán necesariamente siempre algún proceso mental, pueden ser producto de algún error de teclado o distracción, de modo que no podrían ser individualmente analizadas a profundidad. Sin embargo, al identificar qué acciones observar y luego al observarlas en varios estudiantes que resuelven un problema se pueden empezar a vislumbrar patrones en los datos que señalen algo significativo. El primer paso es entonces esencial, el determinar qué observar.

En la primera fase de la investigación (Estudio 1 y 2) se obtuvieron datos iniciales de dos fuentes principales, entrevistas de expertos y observación de estudiantes programadores. Para el primer estudio se definieron los expertos a entrevistar, las preguntas guía a realizar y se realizaron las entrevistas. La información obtenida de las entrevistas se estudió mediante un análisis temático que produjo una clasificación de acciones de usuarios y requerimientos del sistema para la captura y el análisis. Las acciones de usuarios a registrar se convirtieron también en requerimientos.

Tanto en la definición de preguntas guía para la entrevista, como en las entrevistas en sí se veló por cumplir el objetivo de identificar las acciones a observar. En cada entrevista se realizó una introducción para contextualizar la entrevista en los objetivos de la investigación y respecto a las destrezas metacognitivas de planificación, organización y regulación como temas principales.

Adicionalmente, la forma en la que se llevó a cabo el análisis temático también consideró este objetivo. Es decir, en el análisis temático se buscó cumplir el objetivo de dos maneras concretamente, realizando un análisis con categorías predefinidas por

destrezas metacognitivas y realizando un segundo análisis sin categorías predefinidas de carácter más exploratorio y receptivo del aporte de los expertos.

5.2 Discusión Estudio 2

Para el Estudio 2 ya se inició con una lista de acciones a observar por lo que parte del trabajo de este estudio fue corroborar la completitud de la lista. Adicionalmente se buscó, basándose en teoría, etiquetar mediante análisis temático las acciones tomadas por los estudiantes observados. En un segundo plano, la observación permitió identificar requerimientos de diseño del software que se usó en el estudio 3.

Se buscaron coincidencias entre las acciones mencionadas por expertos y las acciones observadas en estudiantes. Posteriormente se logra una triangulación de las acciones identificadas con la aplicación del software en el estudio 3 al encontrar como estudiantes con acciones similares terminan clasificados en grupos similares según un algoritmo de agrupación.

En esta fase se definieron requerimientos de acciones a capturar, requerimientos acerca de la captura en general y requerimientos acerca del proceso de análisis de los datos registrados. Cada uno de los requerimientos terminó siendo una funcionalidad del software resultante. Hacer que el software haga un registro y análisis fiel a las acciones requiere de un diseño que incorpore esas acciones y demás requerimientos de diseño y análisis encontrados en esos estudios. Por lo que, este estudio sentó la base del diseño del software.

5.3 Discusión Estudio 3

En la segunda fase, para el Estudio 3, se llevó a cabo la creación de un instrumento nuevo mediante un proceso de desarrollo de software que incluye desde el diseño hasta la implementación. Esto en un proceso iterativo descrito en el capítulo previo como un resultado de los estudios 1 y 2.

El módulo de análisis es la herramienta principal para analizar los datos registrados por el módulo de captura. Permite la clasificación de las acciones según el efecto que

produce en el código o según la intención del usuario (para el caso de compilación o ejecución, por ejemplo). La clasificación de las acciones según el efecto en el código lo hace según la estructura sintáctica del código por lo que identifica qué modificación en el código realmente se realiza. Este módulo posibilita observaciones y conclusiones no disponibles de otra manera. Esto porque de manera automatizada ofrece una mirada profunda (cercana a la intención de modificación de código) en el proceso de aprendizaje que está desarrollando el estudiante mientras manipula el código. Es decir, el módulo de análisis identifica acciones básicas como compilación o ejecución así como la identificación de acciones más complejas como la modificación en estructuras de control que guían la solución de un problema. En cada paso tomado por los estudiantes en el desarrollo de su código clasifica la acción principal de ese paso.

5.4 Implicaciones

El haber identificado acciones que indican destrezas metacognitivas permite a futuro estudiar a fondo esa relación. El haber realizado la observación de acciones con software ilustra que con técnicas similares es posible obtener información de destrezas metacognitivas sin interrupción del proceso mental.

Es decir, el software representa una intervención mínima, que se puede aplicar a estudiantes donde el usuario puede concentrarse en su esfuerzo cognitivo y metacognitivo para resolver un problema aun cuando, en línea, se está evaluando este proceso.

Se pueden señalar múltiples implicaciones investigaciones futuras como la posibilidad de usar el módulo de análisis de cambios en la evaluación de ejercicios de programación. Otra posible aplicación de esta herramienta de análisis de cambios en la educación está en la posibilidad de estudiar la programación a partir de las estructuras de código que se generan y las maneras en las que se altera esta estructura. Por ejemplo, en un curso de estructuras de datos podría estudiarse la manera en la que se utilizan las posibilidades que aporta una determinada estructura y si se está utilizando apropiadamente,

si se está sub utilizando o si se está sobre utilizando aunque cada uso esté correcto. Esto se puede hacer inspeccionando visualmente los ámbitos en los que se está usando la estructura pero también definiendo estructuras sintácticas nuevas a identificar en el módulo de análisis para su clasificación.

Puede utilizarse el software para un estudio a fondo de lo que un estudiante realiza. Así, por ejemplo, pueden apreciarse los intentos fallidos de producir un bloque o las razones por las que intenta una u otra cosa involucrando al estudiante en el análisis posterior de su solución. De esta manera se puede enriquecer la interpretación de las acciones.

La profundidad en el código es otro aspecto que se puede reconfigurar en la herramienta de visualizador de árbol sintáctico. Si se realizan supuestos sobre la profundidad del código, acerca de, por ejemplo, la complejidad de una solución o el nivel de organización entonces esta herramienta permite visualizar la cantidad de veces que se alcanza un nivel de profundidad o el viaje entre niveles de profundidad. Un estudio similar se puede realizar para obtener información sobre las veces y las razones en las que se cambia de ámbito en el código fuente.

5.5 Limitaciones

La principal limitación del estudio radicó en el número reducido de casos finalmente considerados. Esto se dio por los datos que debieron descartarse por participación incompleta. Esta limitación es mitigada por el hecho de que finalmente queda la posibilidad de aplicaciones mayores por la escalabilidad propia del software resultante. Futuros estudios podrán utilizar el software para replicar, refutar o complementar los resultados obtenidos acá. Además, se demostró la utilidad del software en la captura de acciones a partir de la cual fuera posible la distinción y agrupación de estudiantes para estudiar su uso de destrezas metacognitivas.

Otra limitación yace en la elección de comparar resultados obtenidos con una herramienta de autorreporte (MAI) que no está definida específicamente al dominio de programación y que conlleva los riesgos propios de una herramienta de este tipo.

Por un lado una herramienta de evaluación contextualizada a programación pueda producir resultados más fieles a la realidad de cada sujeto. Por otro lado, las falencias de herramientas de autorreporte podrían evitarse con otras formas de evaluación.

Por último, futuros estudios podrán evaluar la utilidad del software de otra manera, por ejemplo, evaluando la efectividad de tratamientos aplicados a los grupos identificados. Sin embargo, las relaciones encontradas son teóricamente esperadas por lo que no parece haber conflicto encontrado en este estudio.

5.6 Recomendaciones de investigación futura

Basándose en los resultados y productos concretos de esta investigación, se pueden desarrollar investigaciones nuevas en múltiples sentidos. Por ejemplo, según las conclusiones pueden desarrollarse estudios de replicación y verificación de los resultados en los que se puedan quizás refinar y/o aumentar la precisión de las observaciones. Esto puede ser por mayor precisión en el análisis o mayor precisión en la captura de acciones. Quizás se puedan definir nuevas acciones a observar, como la creación de clases o distintos módulos de organización de código según el lenguaje que se escoja.

Hay opciones, posibles ahora por la existencia del software, para desarrollar la idea de implementar un sistema experto que sirva para tomar una determinación sobre la ubicación del caso actual y utilizar eso de alguna manera para servir de apoyo metacognitivo. Por ejemplo, si el sistema experto se utiliza en línea se podrían realizar algunas acciones como recordar más realizar un autorreporte o incluso implementar alguna solicitud de cambio de estrategia o revisión de algún fragmento de código según sea el caso.

En intentos de replicación se pueden recomendar acciones en dos sentidos, escalación hacia arriba o hacia abajo. Escalando hacia arriba, aplicando el software a más

estudiantes se pueden obtener múltiples ventajas como la posibilidad de tener más confianza de generalizar los resultados. Además de algunas implicaciones esperables de la escalación hacia arriba, como el manejo más complejo de los datos obtenidos, se puede esperar también que el efecto de los casos extremos se reduzca y así se tenga un mejor panorama general.

Las múltiples oportunidades de investigaciones futuras se dan también por la posibilidad de aplicar las mismas ideas en otros dominios. En la misma computación pueden desarrollarse aplicaciones similares pero no en la introducción de la programación sino en la programación de algo más concreto en un nivel superior de dominio. Por ejemplo, en el desarrollo de aplicaciones paralelas donde pueden observarse, además de las acciones en el código, las acciones que tengan que ver directamente con la producción y uso del paralelismo. Lo mismo se puede decir de aplicaciones de bases de datos donde puede verificarse, por ejemplo, el uso de patrones concretos determinados como seguros o de buenas prácticas. También en redes donde pueden estudiarse acciones conducentes a mejorar la seguridad de un sistema o en inteligencia artificial donde podría estudiarse la manera y uso de configuraciones de los algoritmos de aprendizaje o monitorear el uso de tipos de aprendizaje (supervisado, no supervisado).

En otra línea, también dentro de computación, puede complementarse el software de captura con el de tutoría y explorar el campo de tutores automáticos donde el software no solo captura acciones sino que también brinda información y ayuda, en aplicaciones similares a las descritas ya en este documento. Un tutor automático puede registrar acciones en el código pero también registrar consultas de las ayudas o por otro lado ofrecer ayudas según las acciones registradas. En este mismo ámbito puede también entonces considerarse entonces, las aplicaciones de tutores automáticos pero no necesariamente de computación.

Si se modifica el software de análisis, y se aplican las mismas ideas de registro de acciones a nivel léxico y sintáctico puede entonces también crearse una aplicación e investigación asociada al estudio de acciones y comportamientos en un documento en

general, no solo de código de un programa. Por ejemplo, puede desarrollarse una aplicación que observe acciones sintácticas en una redacción en un documento. Podrían, entonces, registrarse acciones de alto nivel de abstracción que, conjuntamente, lleven más significado sobre el documento, por ejemplo de agregar verbos, usar sujetos ya mencionados, introducir un nuevo tema en un documento o desarrollar un tema ya introducido. Incluso podrían agregarse capacidades para observar y registrar el uso de cierto tipo de lenguaje o cierto nivel de uso de adjetivos o hasta dialectos o dominio de conocimiento utilizados.

Pueden desarrollarse investigaciones similares en ámbitos técnicos, donde las acciones observadas y registradas tienen que ver con cambios en configuraciones o instrucciones giradas a una maquinaria o un robot industrial o en campos artísticos donde se registran cambios a la letra de una canción o a las notas de una partitura.

Observando cambios en imagen pueden registrarse cambios en cuanto a figuras, colores, uso de espacio, simetría del dibujo y otros parámetros. Esto mismo para dibujos en físico, con componentes de captura automática o en ambientes igualmente digitales observando cambios en la representación digital de la imagen creada por un artista digital.

En todas estas aplicaciones muchas de las mismas técnicas de análisis de datos se podrían aplicar sin mayor problema haciendo modificaciones a las configuraciones de análisis pero siguiendo un proceso muy similar. En múltiples de estas aplicaciones podrían definirse, similarmente, comportamientos y explorarse los patrones que conllevan información. El eje de habilidades metacognitivas puede cambiarse para explorar relación con otras temáticas.

5.7 Hallazgos

Se realizó una investigación coherente pero también pertinente de la realidad actual cuyo aporte teórico novedoso es de valor aplicable a problemas reales. Esto incluye tanto definición de acciones a observar, representando el conocimiento de expertos, el diseño y la

creación de la herramienta de registro de acciones y el análisis de información recolectada como la metodología utilizada para el análisis de los datos.

El principal resultado obtenido es tener la posibilidad de evaluación rápida, para aplicar en un contexto de aula, mediante el software y técnicas de análisis descritas. Esto permite a docentes hacer un diagnóstico de su grupo y enfocar su esfuerzo, según las limitaciones que identifique. Esta solución representa conocimientos de expertos, es configurable y escalable, de rápida aplicación y produce resultados visibles que incluso pueden analizarse paso por paso con estudiantes.

Finalmente, la investigación también construye la base de futuras investigaciones en aplicaciones más numerosas que sirvan de insumo adicional al sistema experto para mejorar su diagnóstico y aplicabilidad a diversidad de contextos y situaciones. Esto incluye investigaciones futuras en la educación de programación pero también en otros ámbitos de estudio en la computación y casi cualquier área donde estudiantes aprenden en interacción con la computadora de manera textual y que produzcan resultados sujetos a análisis sintáctico y semántico.

El aporte teórico es también resultado de la investigación e incluye la identificación de vacíos y posibilidades de investigación, según estudio de estado del arte. La dificultad inherente de medición de las destrezas metacognitivas en general y del programador, en particular, se trata mediante observación pasiva, no intrusiva, que reduce la imposición al proceso mental natural que se llevan en la resolución de problemas y así, entonces, se obtiene una lectura más pura que aporta valiosa información. La investigación también aporta el conocimiento experto del dominio en la medición de destrezas metacognitivas, y por ende también, ofrece una propuesta metodológica para realizar esto desde la inclusión del conocimiento hasta la representación del mismo en un sistema experto difuso.

Referencias

ACM. (2013). *Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science*. http://dx.doi.org/10.1145/2534860

ACM/IEEE. (2020). *Computing curricula 2020: Paradigms for Global Computing Education*. (December).

Adiansyah, Corebima, A. D., Zubaidah, S., y Rohman, F. (2021). The correlation between metacognitive skills and scientific attitudes towards the retention of male and female students in South Sulawesi, Indonesia. *International Journal of Evaluation and Research in Education*, *10*(4), 1272–1281.

Ahadi, A., y Lister, R. (2013). *Geek genes, prior knowledge, stumbling points and learning edge momentum: Parts of the one elephant?* 123–128.

Aharoni, D. (2000). Cogito, ergo sum! Cognitive processes of students dealing with data structures. *ACM Sigcse Bulletin*, 26–30. https://doi.org/10.1145/331795.331804

Akama, K., y Yamauchi, H. (2004). Task performance and metacognitive experiences in problem-solving. *Psychological reports*, *94*(2), 715–722.

Albab, U., Budiyono, B., y Indriati, D. (2020). Metacognition skills and higher order thinking skills (HOTS) in mathematics. *Journal of Physics: Conference Series*. https://doi.org/10.1088/1742-6596/1613/1/012017

Aleven, V., y Koedinger, K. R. (2002). An effective metacognitive strategy: Learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive Science*, *26*(2).

Aleven, V., Roll, I., McLaren, B. M., y Koedinger, K. R. (2010). Automated, Unobtrusive.

Action-by-Action Assessment of Self-Regulation During Learning With an Intelligent Tutoring System. Educational Psychologist, *45*(4).

Alhojailan, M. I. (2012). Thematic analysis: A critical review of its process and evaluation. *West East Journal of Social Sciences*, 39–47.

Allan, V. H., y Kolesar, M. V. (1996). Teaching Computer Science: A Problem Solving Approach that Works. *Proceedings of the Annual National Educational Computing Conference (Vol*, *13*, 2–9.

Altindağ, M., y Senemoglu, N. (2013). METACOGNITIVE SKILLS SCALE. *Hacettepe Üniversitesi Eğitim Fakültesi Dergisi [Hacettepe University Journal of Education]*, *28*, 15–26.

Anderson, D., y Nashon, S. (2007). Predators of knowledge construction: Interpreting students' metacognition in an amusement park physics program. *Science Education*, *91*(2).

Anderson, J. R. (1976). *Language, memory, and thought*. L. Erlbaum Associates.

Anderson, J. R. (1983). *The Architecture of Cognition*. Harvard University Press.

Anderson, J. R. (1993). *Rules of the Mind*. Erlbaum.

Anderson, J. R., y Bower, G. H. (1973). *Human Associative Memory*. Winston and Sons.

Anderson, Nashon, S. M., y Thomas, G. P. (2008). Evolution of Research Methods for Probing and Understanding Metacognition. *Research in Science Education*, *39*(2).

Annelies Raes, Tammy Schellens, Bram De Wever, y Dries F. Benoit. (2016). Promoting metacognitive regulation through collaborative problem solving on the web. *Computers in Human Behavior*, *58*, 325–342.

Antonenko, P. D., Toy, S., y Niederhauser, D. S. (2012). Using Cluster Analysis for Data Mining in Educational Technology Research. *Educational Technology Research and Development*, *60*(3), 383–398.

Arshad, N. (2009). Teaching Programming and Problem Solving to CS2 Students using Think-Alouds. *ACM SIGCSE Bulletin*, *41*(1), 372–376.

Asan, U., y Ercan, S. (2012). *An Introduction to Self-Organizing Maps* (pp. 299–319). https://doi.org/10.2991/978-94-91216-77-0_14

Baker, L., y Cerro, L. (2000). *3. Assessing Metacognition in Children and Adults*.

Bannert, M., y Mengelkamp, C. (2007). Assessment of metacognitive skills by means of instruction to think aloud and reflect when prompted. *Does the verbalisation method affect learning? Metacognition and Learning*, *3*(1).

Basili, V., y Turner, A. J. (1985). Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software Engineering*, *6*, 703–712.

Becker, B. A., y Quille, K. (2019). 50 years of CS1 at SIGCSE: A review of the evolution of introductory programming education research. *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 338–344. https://doi.org/10.1145/3287324.3287432

Bennedsen, J., y Caspersen, M. E. (2007). Failure Rates in Introductory Programming. *SIGCSE Bull.*, *39*(2), 32–36. https://doi.org/10.1145/1272848.1272879

Bennedsen, J., y Caspersen, M. E. (2008). *Abstraction ability as an indicator of success for learning computing science*. Fourth International Workshop on Computing Education Research, 15–25.

Bennedsen, J., y Caspersen, M. E. (2019). Failure Rates in Introductory Programming: 12 Years Later. *ACM Inroads*, *10*(2), 30–36. https://doi.org/10.1145/3324888

Bergin, S., y Reilly, R. G. (2005). *Programming: Factors that influence success*. *37*(1), 411–415.

Bergin, S., Reilly, R., y Traynor, D. (2005). *Examining the role of self-regulated learning on introductory programming performance*. First International Workshop on Computing Education Research.

Berglund, A., y Lister, R. (2010). Introductory programming and the didactic triangle. *Proceedings of the Twelfth Australasian Conference on Computing Education -*, *103*.

Blikstein, P. (2011). Using Learning Analytics to Assess Students' Behavior in Open-Ended Programming Tasks. *Proceedings of the 1st International Conference on Learning Analytics and Knowledge*, 110–116. https://doi.org/10.1145/2090116.2090132

Block, C. C. (2005). *What Are Metacognitive Assessments? Israel Susan E*.

Bloom, B. S. (1956). *Taxonomy of Educational Objectives*. Handbook I: The Cognitive Domain.

Bower, M. (2004). *Teaching and Learning Computing*. Citeseer. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.127.3459&rep=rep1&type=pdf

Boyatzis, R. E. (1998). *Transforming qualitative information: Thematic analysis and code development.* (pp. xvi, 184). Sage Publications, Inc.

Boyer, N. R., Langevin, S., y Gaspar, A. (2008). Self direction & constructivism in programming education. *Proceedings of the 9th ACM SIGITE conference on Information technology education - SIGITE '08*, *89.*

Bråten, I. (1991). Vygotsky as Precursor to Metacognitive Theory: II. Vygotsky as Metacognitivist. *Scandinavian Journal of Educational Research*, *35*(4), 305–320. https://doi.org/10.1080/0031383910350406

Braun, V., y Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Braun, V., y Clarke, V. (2019). Reflecting on reflexive thematic analysis. *Qualitative Research in Sport, Exercise and Health*, *11*(4), 589–597. https://doi.org/10.1080/2159676X.2019.1628806

Brown, A. L. (1987). Metacognition, Executive Control, Self-Regulation, and Other More Mysterious Mechanisms. En *Metacognition, Motivation, and Understanding* (pp. 65–116). L. Erlbaum Associates.

Brown, N. C. C., Altadmri, A., Sentance, S., y Kölling, M. (2018). Blackbox, Five Years On: An Evaluation of a Large-Scale Programming Data Collection Project. *Proceedings of the 2018 ACM Conference on International Computing Education Research*, 196–204. https://doi.org/10.1145/3230977.3230991

Brown, N. C. C., Kölling, M., McCall, D., y Utting, I. (2014). Blackbox: A Large Scale Repository of Novice Programmers' Activity. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, 223–228. https://doi.org/10.1145/2538862.2538924

Brown, R., y McNeill, D. (1966). The “Tip of the Tongue” Phenomenon. *Journal of Verbal Learning and Verbal Behavior*, *5*(4), 325–337.

Brown, S. R. (2000). Tip-of-the-Tongue Phenomena: An Introductory Phenomenological Analysis. *Consciousness and Cognition*, *9*(4), 516–537. https://doi.org/10.1006/ccog.2000.0421

Bruning, R. H., Schraw, G. J., Norby, M. M., y Ronning, R. R. (2004). Cognitive Psychology and Instruction. *British Journal of Educational Psychology (Vol*, *45*, 426.

Bryman, A. (2012). *Social research methods* (Fourth edition). Oxford University Press.

Buckley, J. J., Siler, W., y Tucker, D. (1986). A fuzzy expert system. *Fuzzy Sets and Systems*, *20*(1), 1–16.

Butler, M., y Morgan, M. (2007). *Learning challenges faced by novice programming students studying high level and low feedback*. Proceedings of the Conference of the Australasian Society for Computers in Learning in Tertiary Education (ASCiLiTE).

Byrne, P., y Lyons, G. (2001). The effect of student attributes on success in programming. *SIGCSE Bull. 2*, *33*(3).

Calloni, B. A., Bagert, D. J., y Haiduk, H. P. (1997). Iconic Programming Proves Effective for Teaching the First Year Programming Sequence. *Proceedings of the Twenty-Eighth SIGCSE Technical Symposium on Computer Science Education*, 262–266. https://doi.org/10.1145/268084.268189

Carbone, A., y Kaasbøll, J. J. (1998). A survey of methods used to evaluate computer science teaching. *ACM SIGCSE Bulletin*, *30*(3).

Carbone, A., Mitchell, I., Gunstone, D., y Hurst, J. (2002). *Designing programming tasks to elicit self-management metacognitive behaviour. International Conference on Computers in Education Proceedings*.

Cho, K., y Cho, M. (2007). *Self-Monitoring in Learning to Write*. Evaluation, 13–22.

Cho, K., Cho, M.-H., y Hacker, D. J. (2010). Self-monitoring support for learning to write. *Interactive Learning Environments*, *18*(2), 1080/10494820802292386.

Choi, J. Y. (2006). Metacognitive Evaluation Method in Consecutive Interpretation for Novice Learners. *Meta*, *51*(2). http://www.erudit.org/revue/meta/

Christensen, H. B., y Caspersen, M. E. (2002). *Frameworks in CS1 – a Different Way of Introducing Event-driven Programming*. Design, 75–79.

Claudine Kirsch. (2012). Developing children's language learner strategies at primary school. *Education 3-13*, *40*(4), 379–399.

Confrey, J. (1990). What constructivism implies for teaching. En M. Davis y Noddings (Eds.), *Constructivist views on the teaching and learning of mathematics*.

Cook, W. R. (2008). High-level problems in teaching undergraduate programming languages. *ACM SIGPLAN Notices*, *43*(55.).

Cordula Artelt, y Wolfgang Schneider. (2014). Cross-Country Generalizability of the Role of Metacognitive Knowledge in Students' Strategy Use and Reading Competence. *Teachers College Record*, *117*(1), 1–32.

Corman, L. S. (1986). *Cognitive Style, Personality type, and Learning Ability as Factors in Predicting the Success of the Beginning Programming Student*. 80–89.

Costelloe, E. (2004). *Teaching programming the state of the art*.

http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Teaching+Programming+State+of+the+Art#5

Craig, K., Hale, D., Grainger, C., y Stewart, M. E. (2020). Evaluating metacognitive self-reports: Systematic reviews of the value of self-report in metacognitive research. *Metacognition and Learning*, *15*(2), 155–213. https://doi.org/10.1007/s11409-020-09222-y

Danielle S. McNamara, y Amy M. Shapiro. (2005). Multimedia and hypermedia solutions for promoting metacognitive engagement, coherence, and learning. *Journal of Educational Computing Research*, *33*(1), 1–29.

Dasuki, S., y Quaye, A. (2016). Undergraduate Students' Failure in Programming Courses in Institutions of Higher Education in Developing Countries: A Nigerian Perspective. *The Electronic Journal of Information Systems in Developing Countries*, *76*. https://doi.org/10.1002/j.1681-4835.2016.tb00559.x

D'Avanzo, C. (2003). Application of Research on Learning to College Teaching: Ecological Examples. *BioScience*, *53*(11).

Davey, B. (1983). Think Aloud: Modeling the Cognitive Processes of Reading Comprehension. *Journal of Reading*, *27*(1), 44–47.

David R. Thompson. (2012). Promoting metacognitive skills in intermediate Spanish: Report of a classroom research project. *Foreign Language Annals*, *45*(3), 447–462.

De Backer, L., Van Keer, H., y Valcke, M. (2015). Promoting university students' metacognitive regulation through peer learning: The potential of reciprocal peer tutoring. *Higher Education*, *70*(3), 469–486.

Donna Bryce, David Whitebread, y Denes Szucs. (2015). The relationships among executive functions, metacognitive skills and educational achievement in 5 and 7 year-old children. *Metacognition and Learning*, *10*(2), 181–198.

Downing, K., Kwong, T., Chan, S.-W., Lam, T.-F., y Downing, W.-K. (2009). Problem-based learning and the development of metacognition. *Higher Education*, *57*(5).

Efklides, A. (2001). Metacognitive experiences in problem solving: Metacognition, motivation, and self-regulation. En A. E. J. Kuhl; y R. M. Sorrentino (Eds.), *Trends and prospects in motivation research* (pp. 297–323).

Efklides, A. (2005). Metacognition and Affect: What Can Metacognitive Experiences Tell Us about the Learning Process?. *Educational Research Review*. http://dx.doi.org/10.1016/J.EDUREV.2005.11.001

Efklides, A. (2008). Metacognition: Defining its facets and levels of functioning in relation to self-regulation and co-regulation. *European Psychologist*, *13*, 277–287. https://doi.org/10.1027/1016-9040.13.4.277

Efklides, A. (2014). How Does Metacognition Contribute to the Regulation of Learning? An Integrative Approach. *Psychological topics*, *23*, 1–30.

Efklides, A., y Metallidou, P. (2020). *Applying Metacognition and Self-Regulated Learning in the Classroom*. http://dx.doi.org/10.1093/ACREFORE/9780190264093.013.961

Efklides, A., y Tsiora, A. (2002). Metacognitive experiences, self-concept, and self-regulation. *Psychologia An International Journal of Psychology in the Orient*, *45*(4).

http://www.mendeley.com/research/metacognitive-experiences-selfconcept-and-self regulation/

Efklides, A., y Vlachopoulos, S. P. (2012). Measurement of metacognitive knowledge of self, task, and strategies in mathematics. *European Journal of Psychological Assessment*, *28*(3), 227–239. https://doi.org/10.1027/1015-5759/a000145

Ericsson, K. A. (1987). Verbal reports on thinking. En H. Simon (Ed.), *Introspection in Second Language Research* (pp. 24–54). Avon: Multilingual Matters.

Ericsson, K. A., y Simon, A. H. (1993). *Protocol analysis: Verbal reports as data*.

Ericsson, K. A., y Simon, H. (1980). Verbal reports as data. *Psychological Review*, *87*(3).

Ericsson, K. A., y Simon, H. A. (1998). How to Study Thinking in Everyday Life: Contrasting Think-Aloud Protocols With Descriptions and Explanations of Thinking. *Mind, Culture and Activity*, *5*(3).

Estey, A., y Coady, Y. (2016). *Can Interaction Patterns with Supplemental Study Tools Predict Outcomes in CS1?* 236–241. https://doi.org/10.1145/2899415.2899428

Estey, A., Keuning, H., y Coady, Y. (2017). Automatically classifying students in need of support by detecting changes in programming behaviour. *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE*, 189–194. https://doi.org/10.1145/3017680.3017790

Everitt, B., Landau, S., y Leese, M. (2010). *Cluster Analysis Ed. 5*. John Wiley & Sons.

Evita Kusumawati, Budiyono, y Diari Indriati. (2021). The Development of the Valid Test Instruments to Measure Metacognition Problem Solvers. *Proceedings of the International Conference of Mathematics and Mathematics Education (I-CMME*

2021), 96–103. https://doi.org/10.2991/assehr.k.211122.013

Fay, A. L., y Mayer, R. E. (1994). Benefits of Teaching Design Skills Before Teaching Logo Computer Programming: Evidence for Syntax Independent Learning. *Journal of Educational Computing Research*, *11*(3).

Flavell, J. (1971). Stage-related properties of cognitive development. *Cognitive Psychology*, *2*(4), 421–453. https://doi.org/10.1016/0010-0285(71)90025-9

Flavell, J. H. (1976). Metacognitive aspects of problem solving. En L. B. Resnick (Ed.), *The nature of intelligence* (pp. 136–231). Erlbaum.

Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American Psychologist*, *34*(10), 906–911. https://doi.org/10.1037/0003-066X.34.10.906

Fleming, S. M. (2016). HMeta-d: Hierarchical Bayesian estimation of metacognitive efficiency from confidence ratings. *Neuroscience of Consciousness*, *2017*(1), 007-.

Fleming, S. M., y Daw, N. D. (2016). Self-evaluation of decision-making: A general Bayesian framework for metacognitive computation. *Psychological Review*, *124*(1), 91–114.

Fleming, S. M., Huijgen, J., y Dolan, R. J. (2012). Prefrontal contributions to metacognition in perceptual decision making. *The Journal of Neuroscience : The Official Journal of the Society for Neuroscience*, *32*(18), 6117–6125. https://doi.org/10.1523/JNEUROSCI.6489-11.2012

Fox, R. (2001). Constructivism Examined. *Oxford Review of Education*, *27*(1).

Frades, I., y Matthiesen, R. (2009). Overview on Techniques in Cluster Analysis. En

Methods of Molecular Biology (Vol. 593, pp. 81–107). Humana Press. https://doi.org/10.1007/978-1-60327-194-3_5

Freedman, J. L., y Landauer, T. K. (1966). Retrieval of long-term memory: "Tip-of-the-tongue" phenomenon. *Psychonomic Science*, *4*, 309–310.

Frenken, L. (2020). *Measuring Students' Metacognitive Knowledge of Mathematical Modelling*. http://dx.doi.org/10.1007/978-3-030-66996-6_18

Fu-Yun Yu. (2005). *Promoting Metacognitive Strategy Development through Online Question-Generation Instructional Approach*. 564–571.

Gibbs, G. (2007). *Analyzing Qualitative Data*. https://doi.org/10.4135/9781849208574

Ginat, D. (2001). Metacognitive awareness utilized for learning control elements in algorithmic problem solving. *SIGCSE Bull.*, *33*(3).

Ginat, D. (2002). *On varying perspectives of problem decomposition*. 33rd SIGCSE technical symposium on Computer science education.

Goetz, J. P., y Lecompte, M. (1998). *Conceptualización del proceso de investigación: Teoría y Diseño*. Etnografía y diseño cualitativo en investigación educativa.

Greenlaw, R., y Kantabutra, S. (2013). *Survey of Clustering: Algorithms and Applications*. *3*(2), 1–29.

Gregory Schraw, Lori Olafson, Michelle L. Weibel, y Daphne Sewing. (2011). *Metacognitive Knowledge and Field-based Science Learning in an Outdoor Environmental Education Program*. 57–77.

Gross, P., y Powers, K. (2005). *Evaluating assessments of novice programming environments*. Proceedings of the first international workshop on Computing

education research, 99–110.

Gruneberg, M. M., y Monks, J. (1974). ‘Feeling of knowing’ and cued recall. *Acta Psychologica*, *38*(4), 257–265. https://doi.org/10.1016/0001-6918(74)90010-9

Guibert, N., Guittet, L., y Girard, P. (2005). *A Study of the Efficiency of an Alternative Programming Paradigm to Teach the Basics of Programming*. (figure 1).

Gulbinskienė, D., Masoodi, M., y Šliogerienė, J. (2017). *Moodle as Virtual Learning Environment in Developing Language Skills, Fostering Metacognitive Awareness and Promoting Learner Autonomy*. *127*(3), 176–185.

Hagan, D., y Markham, S. (2000). *Does it help to have some programming experience before beginning a computing degree program? ACM SIGCSE Bulletin, 5th annual SIGCSE/SIGCUE conference on Innovation and technology in computer science education (pp*. 25–28).

Hart, J. T. (1965). Memory and the feeling-of-knowing experience. *Journal of Educational Psychology*, *56*(4), 208–216. https://doi.org/10.1037/h0022263

Hart, J. T. (1966). *Recall, Recognition, and the Memory-Monitoring Process (Dissertation Plus Addendum)* [PhD Thesis].

Hart, J. T. (1967). Memory and the memory-monitoring process. *Journal of Verbal Learning and Verbal Behavior*, *6*, 685–691.

Hashmi, A., Khalid, M., y Shoaib, A. (2019). A Cross-Sectional Study of Assessing Metacognitive Knowledge and Metacognitive Regulatory Skills among Prospective Teachers and Its Relation to their Academic Achievement. *Bulletin of Education and Research*, *41*(2), 215–234.

Havenga, M. (2015). The role of metacognitive skills in solving object-oriented programming problems: A case study. *The Journal for Transdisciplinary Research in Southern Africa*, *11*, 15. https://doi.org/10.4102/td.v11i1.36

Havenga, M., Mentz, E., Breed, B., Govender, D., Govender, I., Dignum, F., y Dignum, V. (2012, marzo). *A case study regarding teachers' problem-solving activities and approaches towards computer programming in diverse learning environments*.

Henriksen, P., Kölling, M., McCall, D., y M, K. (2010). #246. *& lling. (2010). Motivating programmers via an online community. J. Comput. Small Coll.*, *25*(3).

Hipp, R. D. (2020). *SQLite* (3.31.1). https://www.sqlite.org/index.html

Hubertina Thillmann, Jill Gößling, Jessica Marschner, Joachim Wirth, y Detlev Leutner. (2012). *Metacognitive Knowledge About and Metacognitive Regulation of Strategy Use in Self-Regulated Scientific Discovery Learning: New Methods of Assessment in Computer-Based Learning Environments*.

Hull, A., y Du Boulay, B. (2009). Scaffolding Motivation and Metacognition in Learning Programming. *Artificial Intelligence in Education*, *200*.

Ido Roll, Vincent Aleven, Bruce M. McLaren, y Kenneth R. Koedinger. (2011). Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and Instruction*, *21*(2), 267–280.

Ismael, H. A. (2016). *The Role of Metacognitive Knowledge in Enhancing Learners Autonomy "*.

Jacobse, A. E., y Harskamp, E. G. (2012). Towards efficient measurement of metacognition in mathematical problem solving. *Metacognition and Learning*, *7*(2), 133–149.

https://doi.org/10.1007/s11409-012-9088-x

Jadud, M. C. (2006). Methods and Tools for Exploring Novice Compilation Behaviour. *Proceedings of the Second International Workshop on Computing Education Research*, 73–84. https://doi.org/10.1145/1151588.1151600

Jadud, M. C., y Dorn, B. (2015). Aggregate Compilation Behavior: Findings and Implications from 27,698 Users. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, 131–139. https://doi.org/10.1145/2787622.2787718

James, W. (1890). *The Principles of Psychology* (Número v. 1). H. Holt. https://books.google.co.cr/books?id=JLcAAAAAMAAJ

Kandel, A. (1991). *Fuzzy expert systems*. CRC press.

Kansanen, P. (1999). Teaching as Teaching-Studying-Learning Interaction. *Scandinavian Journal of Educational Research*, *43*(1), 81–89. https://doi.org/10.1080/0031383990430105

Kapa, E. (2001). A Metacognitive Support during the process of problem solving in computarized. *Eduational Studies in Mathematics*, *47*(3).

Keppens, J., y Hay, D. (2008). Concept map assessment for teaching computer programming. *Computer Science Education*, *18*(1), 1080/08993400701864880.

Kessel, R., Gecht, J., Forkmann, T., Drueke, B., Gauggel, S., y Mainz, V. (2014). Metacognitive monitoring of attention performance and its influencing factors. *Psychological Research-psychologische Forschung*, *78*(4), 597–607.

Khan, K., Rehman, Aziz, K., Fong, S., Sarasvady, S., y Vishwa, A. (2014). *DBSCAN: Past,*

present and future. 232–238.

Khosa, D. K., y Volet, S. (2013). Promoting effective collaborative case-based learning at university: A metacognitive intervention. *Studies in Higher Education*, *38*(6), 870–889.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464–1480. https://doi.org/10.1109/5.58325

Kohonen, T. (2013). Essentials of the self-organizing map. *Twenty-fifth Anniversay Commemorative Issue*, *37*, 52–65. https://doi.org/10.1016/j.neunet.2012.09.018

Kramarski, B. (2008). Promoting teachers' algebraic reasoning and self-regulation with metacognitive guidance. *Metacognition and Learning*, *3*(2), 83–99.

Kramarski, B., y Mizrachi, N. (2004). ENHANCING MATHEMATICAL LITERACY WITH THE USE OF METACOGNITIVE GUIDANCE IN FORUM DISCUSSION. *Learning*, *3*, 169–176.

Kramarski, B., y Ritkof, R. (2002). The effects of metacognition and email interactions on learning graphing. *Journal of Computer Assisted Learning2*, *18*(1).

Kurtz, B. L. (1980). Investigating the Relationship between the Development of Abstract Reasoning and Performance in an Introductory Programming Class. *SIGCSE Bull.*, *12*(1), 110–117. https://doi.org/10.1145/953032.804622

Larman, C., y Basili, V. (2003). Iterative and Incremental Development: A Brief History. *IEEE Computer*, *36*(6), 47–56.

Leppänen, L., Leinonen, J., y Hellas, A. (2016). Pauses and Spacing in Learning to Program. *Proceedings of the 16th Koli Calling International Conference on*

Computing Education Research, 41–50. https://doi.org/10.1145/2999541.2999549

Lior, R. (2008). *A survey of Clustering Algorithms*. 269–298.

Lister, R., Fidge, C., y Teague, D. (2009). Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *ITiCSE '09*.

Locke, J. (1924). *An Essay Concerning Human Understanding*. Clarendon Press. https://books.google.co.cr/books?id=dSZHAAAAIAAJ

Lockl, K., y Schneider, W. (2007). Knowledge About the Mind: Links Between Theory of Mind and Later Metamemory. *Child Development*, *78*(1), 148–167. https://doi.org/10.1111/j.1467-8624.2007.00990.x

Loksa, D., Margulieux, L., Becker, B. A., Craig, M., Denny, P., Pettit, R., y Prather, J. (2022). Metacognition and Self-Regulation in Programming Education: Theories and Exemplars of Use. *ACM Trans. Comput. Educ.*, *22*(4). https://doi.org/10.1145/3487050

Luxton-Reilly, A., Simon, U., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., y Szabo, C. (2018). Introductory programming: A systematic literature review. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, undefined-undefined. https://doi.org/10.1145/3293881.3295779

Ma, L., Ferguson, J., Roper, M., Ross, I., y Wood, M. (2009). Improving the mental models held by novice programmers using cognitive conflict and jeliot visualisations. *SIGCSE Bull.*, *41*(3).

Ma, L., Ferguson, J., Roper, M., y Wood, M. (2007). Investigating the viability of mental

models held by novice programmers. En *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (Vol. 103).

Malik, S. I. (2016). *Enhancing practice and achievement in introductory programming using an ADRI editor*. https://doi.org/10.1109/TALE.2016.7851766

Malik, S. I., y Coldwell-Neilson, J. (2017). A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*, *22*(3), Article 3. https://doi.org/10.1007/s10639-016-9474-0

Maniscalco, B., y Lau, H. (2012). A signal detection theoretic approach for estimating metacognitive sensitivity from confidence ratings. *Consciousness and Cognition*, *21*(1), 422–430.

Manoli Pifarré, y Ruth Cobos. (2010). *Promoting metacognitive skills through peer scaffolding in a CSCL environment*. *5*(2), 237–253.

Marion Händel, Cordula Artelt, y Sabine Weinert. (2013). Assessing metacognitive knowledge: Development and evaluation of a test instrument. *Journal for educational research online*, *5*(2), 162–188.

Mason, R., Cooper, G., y de Raadt, M. (2012). Trends in Introductory Programming Courses in Australian Universities: Languages, Environments and Pedagogy. *Proceedings of the Fourteenth Australasian Computing Education Conference - Volume 123*, 33–42.

McShane, M., English, J., y Nirenburg, S. (2022). *Knowledge Engineering in the Long Game of Artificial Intelligence: The Case of Speech Acts*.

Mehmood, E., Abid, A., Farooq, M. S., y Nawaz, N. A. (2020). Curriculum, Teaching and

Learning, and Assessments for Introductory Programming Course. *IEEE Access*, *8*, 125961–125981. https://doi.org/10.1109/ACCESS.2020.3008321

Meijer, J., Sleegers, P., Elshout-Mohr, M., van Daalen-Kapteijns, M., Meeus, W., y Tempelaar, D. (2013). The development of a questionnaire on metacognition for students in higher education. *Educational Research*, *55*(1), 31–52. https://doi.org/10.1080/00131881.2013.767024

Merriam, S. B., y Tisdell, E. J. (2016). *Qualiatative research: A guide to design and implementation* (Fourth edition). Jossey-Bass, a Wiley brand.

Michael Hogan, Christopher P. Dwyer, Owen Harney, Chris Noone, y Ronan Conway. (2014). *Metacognitive Skill Development and Applied Systems Science: A Framework of Metacognitive Skills, Self-regulatory Functions and Real-World Applications*. 75–106.

Michelle Bauml y Sarah Quebec-Fuentes. (2016). Promoting Metacognitive Decision-Making in Teacher Education. *Theory Into Practice*, *55*(3), 242–249. https://doi.org/10.1080/00405841.2016.1173997

Montague, M., y Applegate, B. (2010). Middle School Students' Mathematical Problem Solving: An Analysis of Think-Aloud Protocols. *Learning Disability Quarterly*, *16*(1). http://www.jstor.org/stable/1511157

Morrison, M., y Newman, T. S. (2001). A Study of the Impact of Student Background and Preparedness on Outcomes in CS I. *SIGCSE Bull.*, *33*(1), 179–183. https://doi.org/10.1145/366413.364580

Moser, J. S., Moran, T. P., Schroder, H. S., Donnellan, M. B., y Yeung, N. (2013). On the

relationship between anxiety and error monitoring: A meta-analysis and conceptual framework. *Frontiers in Human Neuroscience*, *7*, 466–466.

Murphy, E., Crick, T., y Davenport, J. H. (2017). An Analysis of Introductory Programming Courses at UK Universities. *The Art, Science, and Engineering of Programming*, *1*(2). https://doi.org/10.22152/programming-journal.org/2017/1/18

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., y Balik, S. (2003). Improving the CS1 Experience with Pair Programming. *SIGCSE Bull.*, *35*(1), 359–362. https://doi.org/10.1145/792548.612006

Negnevitsky, M. (2005). *Artificial Intelligence*. Pearson Education UK. https://books.google.co.cr/books?id=dZ97zwEACAAJ

Nelson, T. O., y Narens, L. (1990). Metamemory: A theoretical framework and new findings. *The Psychology of Learning and Motivation*, *26*.

Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press.

Ozturk, N. (2017). Assessing Metacognition: Theory and Practices. *International Journal of Assessment Tools in Education*, *4*, 134–134. https://doi.org/10.21449/ijate.298299

Papert, S. (1999). *Introduction: What is Logo? And Who Needs It? The introduction to Logo Philosophy and Implementation*. a collection of essays on LOGO experiences in a number of different countries around the world.

Pears, A., Seidman, S., Malmi, L., Bennedsen, J., Devlin, M., y Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*.

Pedone, R., Semerari, A., Riccardi, I., Procacci, M., Nicolo, G., y Carcione, A. (2017). Development of a self-report measure of metacognition: The metacognition self-assessment scale (MSAS). Instrument description and factor structure. *Clinical Neuropsychiatry*, *14*, 185–194.

Piaget, J. (1977). *Recherches sur l'abstraction réfléchissante. L'abstraction de l'ordre et des relations spatiales*. PUF.

Pintrich, P. R., y Groot, E. D. (1990). Motivational and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology*, *82*, 33–40.

Ponmalai, R., y Kamath, C. (2019). *Self-Organizing Maps and Their Applications to Data Analysis*. https://doi.org/10.2172/1566795

Porter, R. (2006). *Design Patterns in Learning to Program*. Trial.

Porter, R., y Calder, P. (2004). *(2004)*. Patterns in Learning to Program - An Experiment? In,.

Prather, J., Becker, B. A., Craig, M., Denny, P., Loksa, D., y Margulieux, L. (2020). What Do We Think We Think We Are Doing?: Metacognition and Self-Regulation in Programming. *ICER 2020 - Proceedings of the 2020 ACM Conference on International Computing Education Research*, undefined-undefined. https://doi.org/10.1145/3372782.3406263

Qin, Y., Bothell, D., y Anderson, J. R. (2007). ACT-R Meets fMRI. En *Web Intelligence Meets Brain Informatics* (pp. 205–222). Springer Berlin Heidelberg.

Quille, K., y Bergin, S. (2018). *Programming: Predicting student success early in CS1. A*

re-validation and replication study. 15–20. https://doi.org/10.1145/3197091.3197101

R Core Team. (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Rapkin, B. D., y Luke, D. A. (1993). Cluster analysis in community research: Epistemology and practice. *American Journal of Community Psychology*, *21*(2), 247–277. https://doi.org/10.1007/BF00941623

Reder, L., y Schunn, C. (1996). Metacognition Does Not Imply Awareness: Strategy Choice Is Governed by Implicit Learning and Memory. En *Implicit Memory and Metacognition*.

Reynolds, J. C. (2008). Some thoughts on teaching programming and programming languages. *ACM SIGPLAN Notices*, *43*(11).

Robins, A. (2010). Learning edge momentum: A new account of outcomes in CS1. *Computer Science Education*, *20*(1), 37–71.

Robins, A., Rountree, J., y Rountree, N. (2003). *Learning and Teaching Programming: A Review and Discussion*. *13*(2).

Robson, C., y McCartan, K. (2017). *Real World Research, 4th Edition*.

Rountree, N., Rountree, J., Robins, A., y Hannah, R. (2004). Interacting factors that predict success and failure in a CS1 course. *ACM SIGCSE Bulletin*, *36*(101.).

Salem, A.-B. M., y Nikitaeva, A. Y. (2019). Knowledge Engineering Paradigms for Smart Education and Learning Systems. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics*

(MIPRO), 1571–1574. https://doi.org/10.23919/MIPRO.2019.8756685

Salomon, G., y Perkins, D. N. (1987). *Transfer of Cognitive Skills from Programming: When and How? Journal of Educational Computing Research, 3(2)*. 149–169. http://doi.apa.org/?uid=1989-10093-001

Sara Cotterall, y Garold Murray. (2009). Enhancing metacognitive knowledge: Structure, affordances and self. *System*, *37*(1), 34–45.

Saraç, S., y Karakelle, S. (2012). On-line and Off-line Assessment of Metacognition. *Learning*, *4*(2).

Schellings, G., Van Hout-Wolters, B., Veenman, M., y Meijer, J. (2013). Assessing metacognitive activities: The in-depth comparison of a task-specific questionnaire with think-aloud protocols. *European Journal of Psychology of Education*, *28*, 963–990. https://doi.org/10.1007/s10212-012-0149-y

Schmitz, T. W., Kawahara-Baccus, T. N., y Johnson, S. C. (2004). Metacognitive evaluation, self-relevance, and the right prefrontal cortex. *NeuroImage*, *22*(2). http://www.ncbi.nlm.nih.gov/pubmed/15193625

Schneider, W. (2008). The Development of Metacognitive Knowledge in Children and Adolescents: Major Trends and Implications for Education. *Mind, Brain, and Education*, *2*(3), 114–121. https://doi.org/10.1111/j.1751-228X.2008.00041.x

Schneider, W., Tibken, C., y Richter, T. (2022). The development of metacognitive knowledge from childhood to young adulthood: Major trends and educational implications. *Advances in child development and behavior*, *63*, 273–307. https://doi.org/10.1016/bs.acdb.2022.04.006

Schraw, G. (1998). Promoting general metacognitive awareness. *Instructional Science*, *26*(1), 3–16.

Schraw, G., y Dennison, R. S. (1994). Assessing metacognitive awareness. *Contemporary Educational Psychology*, *19*(4), 460–475.

Schraw, G. J. y Impara, J.C. (2000). *Issues in the measurement of metacognition*. Buros Institute of Mental Measurements.

Schraw, G., y Moshman, D. (1995). Metacognitive theories. *Educational Psychology Review*, *7*(4), 351–371. https://doi.org/10.1007/BF02212307

Schubert, E., Sander, J., Ester, M., Kriegel, H.-P., y Xu, X. (2017). *DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN*. *42*(3), 19-.

Schulte, C., y Bennedsen, J. (2006). *What do teachers teach in introductory programming? Proceedings of the second international workshop on Computing education research*. Canterbury, United Kingdom: ACM. doi:10.

Schwartz, B. (2006). Tip-of-the-tongue states as metacognition. *Metacognition and Learning*, *1*, 149–158. https://doi.org/10.1007/s11409-006-9583-z

Shackelford, R., y Badre, A. N. (1993). Why Can't Smart Students Solve Simple Programming Problems? *Int. J. Man Mach. Stud.*, *38*, 985–997.

Sheard, J., Simon, S., Hamilton, M., y Lönnberg, J. (2009). *Analysis of research into the teaching and learning of programming*. ACM.

Sidi, Y., Ackerman, R., y Erez, A. (2017). Feeling Happy and (Over)Confident: The Role of Positive Affect in Metacognitive Processes. *Cognition and Emotion*, *32*(4), 876–884. https://doi.org/10.1080/02699931.2017.1346501

Siler, W., y Buckley, J. J. (2005). *Fuzzy expert systems and fuzzy reasoning*. John Wiley & Sons.

Simon, C., A., R., y M. (2008). *De, Lister, R., Hamilton, M., & Sheard*. ACM.

Staats, W. J., Dean, D., Miles, D., Blum, T., y Lu, S.-Y. (2003). 2003. *Developing techniques for measuring and enhancing students' cognitive and metacognitive skills. Frontiers in Education 2003 FIE 2003 33rd Annual (Vol*, *1)*, T3B-2–7.

Stein, L. (1998). What We Swept Under the Rug: Radically Rethinking CS1. *Computer Science Education*, *8*(2).

Sulcer, B., Roscoe, R. O. D., y B, S. (2008). *Measuring self-regulated learning skills through social interactions in a teachable agent environment*. Learning, 1–30.

Tavana, M., y Hajipour, V. (2019). A practical review and taxonomy of fuzzy expert systems: Methods and applications. *Benchmarking: An International Journal*, *27*(1), 81–136.

Thomas, G., Anderson, D., y Nashon, S. (2008). Development of an Instrument Designed to Investigate Elements of Science Students' Metacognition, Self-Efficacy and Learning Processes: The SEMLI-S. *International Journal of Science Education*, *30*(13), 1080/09500690701482493.

Thomas, G. P., y McRobbie, C. J. (2001). Using a metaphor for learning to improve students' metacognition in the chemistry classroom. *Journal of Research in Science Teaching*, *38*(2).

Thomas, L., Ratcliffe, M., Woodbury, J., Jarman, E., Thomas, L., Ratcliffe, M., Woodbury, J., y Jarman, E. (2002). *Learning styles and performance in the introductory*

programming sequence. Proceedings of the 33rd SIGCSE technical symposium on Computer science education.

Tiago Roberto Kautzmann, y Patrícia Augustin Jaques. (2019). Effects of adaptive training on metacognitive knowledge monitoring ability in computer-based learning. *Computers in Education*, *129*, 92–105.

Tobias, S., y Everson, H. (2000). *4. Assessing Metacognitive Knowledge Monitoring*.

Tobin, G. A., y Begley, C. M. (2004). Methodological rigour within a qualitative framework. *Journal of advanced nursing*, *48*(4), 388–396.

Veenman, M. (2005). *The assessment of metacognitive skills: What can be learned from multi method designs? In C* (Artelt y B. Moschner, Eds.). Lernstrategien und Metakognition: Implikationen fu¨ r Forschung und Praxis.

Veenman, M., y Elshout, J. J. (1999). Changes in the relation between cognitive and metacognitive skills during the acquisition of expertise. *European Journal of Psychology of Education*, *14*(4), 509–523. https://doi.org/10.1007/BF03172976

Veenman, M., Hout-Wolters, B. V., y Afflerbach, P. (2006). Metacognition and learning: Conceptual and methodological considerations. *Metacognition and Learning*, *1*.

Veenman, M. V. J., Elshout, J. J., y Meijer, J. (1997). The generality vs domain-specificity of metacognitive skills in novice learning across domains. *Learning and Instruction*, *7*(2), 187–209. https://doi.org/10.1016/S0959-4752(96)00025-4

Vico, G. (1710). De antiquissima italorum sapientia. *Stamperia de Classici latini.*

Vrugt, A., y Oort, F. J. (2008). Metacognition, achievement goals, study strategies and academic achievement: Pathways to achievement. *Metacognition and Learning*,

3(2), 123–146.

Vygotsky, L. S. (1978). *Mind in Society*. Harvard University Press; JSTOR. https://doi.org/10.2307/j.ctvjf9vz4

Vygotsky, L. S. (1981). *The genesis of higher mental functions*.

Watson, C., y Li, F. W. B. (2014). Failure Rates in Introductory Programming Revisited. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, 39–44. https://doi.org/10.1145/2591708.2591749

Watson, C., Li, F. W. B., y Godwin, J. L. (2013). Predicting Performance in an Introductory Programming Course by Logging and Analyzing Student Programming Behavior. *2013 IEEE 13th International Conference on Advanced Learning Technologies*, 319–323.

Wernke, S., Anschuetz, A., y Moschner, B. (2011). Assessing Cognitive and Metacognitive Learning Strategies in School Children: Construct Validity and Arising Questions. *International Journal*, *6*(2).

Westbrook, S. (2005). *A multi-institutional investigation of computer science seniors' knowledge of programming concepts*. Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education - SIGCSE '05.

Westhuizen, D. van der, y Chetty, J. (2013). *METACOGNITIVE AWARENESS AND SKILL AMONGST STUDENTS STUDYING COMPUTER PROGRAMMING*.

Westphal, B. T., Harris, F. C., y Fadali, M. S. (2003). *GRAPHICAL PROGRAMMING: A VEHICLE FOR TEACHING COMPUTER PROBLEM SOLVING*. 33rd Annual Frontiers in Education.

Wolfe, J. (2004). Why the Rhetoric of CS Programming Assignments Matters. *Computer Science Education*, *14*(2).

Wolz, U. (2001). *Teaching design and project management with lego RCX robots*. ACM.

Wolz, U., Leitner, H. H., Malan, D. J., y Maloney, J. (2009). Starting with scratch in CS 1. *ACM SIGCSE Bulletin*, *41*(2.).

Wundt, W. (1912). *Lectures on Human and Animal Psychology (Psychology Revivals)*. Routledge. https://doi.org/10.4324/9781315780009

Yeh, S. W., y Lo, J. J. (2005). Assessing metacognitive knowledge in web-based CALL: a neural network approach. *Computers & Education*, *44(2)*, 97–113. https://doi.org/10.1016/j.compedu.2003.12.019

Yeung, y Summerfield, C. (2012). Metacognition in human decision-making: Confidence and error monitoring. *Philosophical Transactions of the Royal Society B*, *367*(1594), 1310–1321.

Yi Tian, Yu Fang, y Jian Li. (2018). The Effect of Metacognitive Knowledge on Mathematics Performance in Self-Regulated Learning Framework-Multiple Mediation of Self-Efficacy and Motivation. *Frontiers in Psychology*, *9*, 2518–2518.

Yin, H. (2008). The Self-Organizing Maps: Background, Theories, Extensions and Applications. En J. Fulcher y L. C. Jain (Eds.), *Computational Intelligence: A Compendium* (pp. 715–762). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-78293-3_17

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353. https://doi.org/10.1016/S0019-9958(65)90241-X

Zadeh, L. A. (1999). From computing with numbers to computing with words. Manipulation of measurements to manipulation of perceptions. *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials. IPMM'99 (Cat. No.99EX296)*, *1*, 3–4.

Zadeh, L. A. (2008). Is there a need for fuzzy logic? *Information Sciences*, *178*(13), 2751–2779. https://doi.org/10.1016/j.ins.2008.02.012