Editorial
Universidad de **Nariño**

Enseñanza de los fundamentos de programación de computadoras y transposición didáctica

Enseñanza de los fundamentos de programación de computadoras y transposición didáctica

Jesús Insuasti, Ph.D.

Editorial
Universidad de Nariño

Doctorado en Ciencias de la Educación – RUDECOLOMBIA
Colección Tesis Doctorales

Insuasti, Jesús
Enseñanza de los fundamentos de programación de computadoras y transposición didáctica / Jesús Insuasti. -- 1ª. ed. -- San Juan de Pasto : Editorial Universidad de Nariño, 2021
178 p. : il. byn., tablas

Incluye bibliografía p. 161-171 y datos del autor p. 7
ISBN: 978-628-7509-66-5 Impreso

1. Educación--Investigación--Ciencias de la computación 2. Educación superior--Currículos 3. Didáctica 4. Estrategias de enseñanza 5. Programación de computadoras--Enseñanza-aprendizaje

378.007 I599 - SCDD-Ed. 22

Universidad de Nariño FUNDADA EN 1904 | Universidad de Nariño ACREDITADA EN ALTA CALIDAD RESOLUCIÓN No.10567 - MAYO 23 DE 2017 | Sección de Biblioteca "Alberto Quijano Guerrero"

Autor: Jesús Insuasti, Ph.D.
Universidad de Nariño.
Doctorado en Ciencias de la Educación – RUDECOLOMBIA. Colección Tesis Doctorales
Enseñanza de los fundamentos de programación de computadoras y transposición didáctica
ISBN 978-628-7509-66-5
Edición: 1ra edición.

Diseño de carátula: Basado en imagen "Code" by jenwikehuger. It is licensed with CC BY-SA 2.0. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/2.0/

Revisión de estilo: Manuel E. Martínez R. manrique@udenar.edu.co
Jenny Guerrero, jennycarg@gmail.com

Digitalización por *Editorial Universitaria* – Universidad de Nariño
Impreso en Pasto, Colombia

Primera Impresión, 2021
ISBN 978-628-7509-66-5

Acerca del autor

Jesús Insuasti es profesor tiempo completo en categoría asociado adscrito al Departamento de Sistemas de la Universidad de Nariño, llevando más de veinte años al servicio de la institución. En su labor se destacan dos grandes pasiones desde una mirada académico-científica: las Ciencias de la Educación y las Ciencias de la Computación.

En este sentido, Jesús Insuasti proviene de una cuna de formación pedagógica en la Escuela Normal Nacional de Pasto (Colombia) —como se denominaba en aquel entonces—, es Especialista y Magister en Docencia Universitaria, además tiene el título de Doctor *cum laude* en Ciencias de la Educación; todos estos títulos de educación superior fueron otorgados por la Universidad de Nariño (Pasto, Colombia).

Por otra parte, Jesús Insuasti es Ingeniero de Sistemas de la Universidad de Nariño, es Desarrollador 5 estrellas certificado por *Microsoft Corporation* (D.F., México), tiene certificación *English Proficiency* otorgada por *San Jose State University* (Silicon Valley, USA) como parte del programa de formación de Becas Fulbright, tiene el título de *Master of Science —with distinction— in Internet Systems* otorgado por *The University of Liverpool* (Liverpool, UK), y es actualmente candidato a Doctor en Ingeniería – Sistemas e Informática de la Universidad Nacional de Colombia (Medellín, Colombia). Sus estancias de investigación han sido realizadas en el Departamento de Informática de la Universidad de Cádiz (Cádiz, España) y en *Computer Science Department* en *Technische Universiteit Eindhoven* (Eindhoven, The Netherlands).

Palabras del autor

Quiero aprovechar esta oportunidad para resaltar la magna labor de la Facultad de Educación de la Universidad de Nariño en la formación de profesionales en dicho campo del conocimiento. En mi caso particular, tuve el privilegio de formarme como Especialista y Magister en Docencia Universitaria, y como Doctor en Educación, gracias a esta casa de estudios.

En este orden de ideas, agradezco inicialmente a la profesora Martha Alicia López Lasso por su valiosa contribución en mi formación en Docencia para la Educación superior, por su confianza y amistad. De igual forma, expreso mi enorme gratitud y admiración al Dr. Roberto Ramírez Bravo; quién ha sido mi maestro en la Especialización, la Maestría y el Doctorado. Él ha representado para mí un ejemplo a seguir en el campo de las Ciencias de la Educación, y ha sido un privilegio contar con su acompañamiento. Agradezco también a la Dra. Gabriela Hernández Vega, actual directora del Programa de Doctorado en Ciencias de la Educación de la Universidad de Nariño, por su contribución académica, su apoyo y su amistad. Mi infinita gratitud al Dr. D. Juan Manuel Dodero Beardo, Profesor Titular adscrito al Departamento de Ingeniería Informática de la Universidad de Cádiz, por sus valiosas orientaciones dado su conocimiento y experiencia, por su dedicación en el transcurso de mi estancia de investigación, y sobre todo por su calidad humana. Doy las gracias al Programa de Movilidad Académica entre Universidades de América Latina y andaluzas asociadas a la AUIP (Asociación Universitaria Iberoamericana de Postgrado) por su generoso apoyo en la financiación de dicha estancia, así como el apoyo de la ORIC (Oficina de Relaciones Internacionales y Convenios) de la Universidad de Nariño por el acompañamiento para la consecución de la beca de movilidad de la AUIP.

Por otra parte, agradezco a los profesores de la Universidad de Cádiz (España) y de la Universidad Nacional, sede Medellín (Colombia) por su tiempo en el desarrollo de las entrevistas a expertos. De igual manera, les doy las gracias a nueve profesores de las siguientes universidades internacionales:

University of Texas at Austin, University of Illinois at Urbana-Champaign, City University of Hong Kong, Paris 13, Swiss Federal Institute of Technology at Zurich, University of Science and Technology of China, Northwestern University, Huazhong University of Science and Technology, The Ohio State University at Columbus. También quiero agradecer a cuatro profesores de las siguientes universidades colombianas: Universidad Nacional de Colombia en Medellín, Politécnico Grancolombiano en Bogotá, Universidad de Nariño en Pasto y Universidad de Medellín; a todos ellos mi gratitud por su participación en las encuestas, entrevistas y el desarrollo de la guía de vigilancia epistemológica.

Finalmente, no hubiese sido posible llegar a esta instancia sin el apoyo incondicional de toda mi familia. A aquellos familiares que aún permanecen conmigo en este plano de existencia y a aquellos que ya se han marchado: infinitas gracias.

獻給我的妻子 Lorena 及兒子 Miguel …

他們是我的靈感

Dedicado a mi esposa Lorena y a mi hijo Miguel …
son mi inspiración.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE SIGLAS Y ACRÓNIMOS

ABP: Aprendizaje Basado en Problemas
ACM: *Association for computing machinery*
ACM SIGCSE: *ACM’s special interest group on computer science education*
ARWU: *Academic ranking of world universities*
CE: *Computer engineering (discipline)*
CS: *Computer science (discipline)*
IDE: *Integrated development environment*
ICFES: Instituto colombiano para la evaluación de la educación
IEEE: *Institute of electrical and electronics engineers*
IS: *Information systems (discipline)*
IT: *Information technology (discipline)*
MEN: Ministerio de Educación Nacional
OOP: *Object-oriented programming*
PbE: *Programming by example*
SE: *Software engineering (discipline)*
TIC: Tecnologías de la información y la comunicación

PRÓLOGO

El pensamiento como contenido objetivo o noema es el término empleado en Fenomenología para referirse al objeto o contenido de un pensamiento, juicio o percepción, que lo diferencia del acto intencional o noesis. El concepto de pensamiento ha sido empleado en la Ciencia de la Computación para delimitar lo que se conoce como pensamiento computacional, término acuñado en 1980 por Seymour Papert y reutilizado en 2006 por Jeannette Wing para referirse al proceso involucrado en la formulación de un problema y sus soluciones, representadas de forma que puedan trasladarse a un procesador automático de información como es una computadora. Alrededor del término pensamiento computacional ha surgido un cierto criticismo, pues muchos investigadores lo contemplan, bien como un eufemismo del soslayado término informática, bien como una forma de sortear la omnipresencia de la programación en la ciencia y la tecnología de las computadoras.

Preocupados por la enseñanza de los fundamentos de la informática, en abril de 2008 un grupo de colegas interesados en el tema fundamos el capítulo español del grupo de ACM de especial interés en la educación en ciencia de la computación (SIGCSE), que humildemente acepté presidir desde la Universidad de Cádiz. Nos preocupaban aspectos epistemológicos de la ciencia informática y, en particular, la debilidad de objetivos, métodos y análisis, así como el escaso interés por la enseñanza de sus fundamentos en muchas disciplinas, despreocupadamente ajenas a los incipientes avances en Ciencia de la Computación que se habían producido en los tiempos finiseculares recién concluidos. La despreocupación por la informática era evidente, ya desde entonces, tanto en educación superior como en enseñanzas medias. A través de la colaboración con el capítulo español de SIGCSE, en abril de 2013 me contactó desde Colombia el autor de este libro, planteando y compartiendo sus inquietudes alrededor de la enseñanza de los fundamentos de la programación de computadoras y del enfoque novedoso de la transposición didáctica, aplicado a la programación de compu-

tadoras y, por ende, a la informática en su concepción más amplia. Ni que decir tiene que la propuesta era fascinante.

Este mundo pandémico o post-pandémico de 2021 ha contribuido a acelerar el vaciado de la caja de Pandora que, sobre hombros de gigantes como Donald Knuth, Seymour Papert o la propia Jeannette Wing, tímidamente desvelamos. Antes, durante y después de su pasantía internacional en la Universidad de Cádiz a finales de 2013, el autor de este libro ha versado su investigación en una parte relevante del andamiaje epistemológico necesario para el objetivo de hacer de la informática, de la ciencia de la computación y de los fundamentos de la programación de computadoras un eje fundamental para el aprendizaje de la ciencia del siglo XXI.

El dilema Husserliano noesis-noema puede servir muy bien para delimitar el amplio terreno de investigación sobre pensamiento computacional, más allá de la tradicional conducción realizada por la industria para construir el cuerpo de conocimiento de la Informática. En la búsqueda de que el saber que se enseña en las escuelas no se desvíe del saber erudito o científico, la vigilancia epistemológica por la que tan acertadamente se preocupa este trabajo pone el acento una transposición didáctica rigurosa como punto de partida.

El trabajo que tengo el honor de prologar es un ejercicio de investigación de ida y vuelta. En la ida, contribuye con una mejora sustancial para aquellas disciplinas que, en ese ejercicio de transposición, conforman la noosfera que detenta la responsabilidad de diseñar los planes de estudio en contextos específicos, actualmente ajenos a los fundamentos de la informática. En el viaje de vuelta, sus resultados contribuyen a incrementar el rigor del saber experto en informática, sin perder la esencia de la ciencia informática por mor de tenerse que ocupar en exceso de tan relevante fenomenología en los planes de estudio.

Los resultados de la investigación, dado el origen de su autor, recuerdan aquellos cantes que, tras pasar por el triángulo geográfico que vio nacer al flamenco, formado por Cádiz, Triana y Ronda, partieron hacia América y volvieron mejorados en su letra y en su música, imbuidos de los ritmos y melodías indianas, para dejar su sello inigualable en unos palos flamencos

conocidos como *cantes de ida y vuelta*, en forma de guajiras cubanas, de milongas argentinas, de rumbas afro-cubanas o, como en esta ocasión, de singulares colombianas.

Cádiz, a 12 de marzo de 2021

Juan Manuel Dodero Beardo
Profesor Catedrático del área de Lenguajes y Sistemas Informáticos,

INTRODUCCIÓN

Teniendo en cuenta dos grandes campos del conocimiento, por un lado, las ciencias de la educación y, por otro lado, la computación y sus disciplinas asociadas, el presente documento representa el informe final de una investigación que integra dichos campos para una situación particular: el estudio de la enseñanza de los fundamentos de programación de computadoras y la transposición didáctica.

La transposición didáctica es una propuesta teórica del profesor Yves Chevallard en ciencias de la educación, originalmente aplicada a la enseñanza de la matemática. Esta propuesta afirma que el conocimiento experimenta una serie de transformaciones al ser representado en objetos que son llevados a contextos diferentes. Partiendo de la primera definición del verbo "Transponer" dada por la Real Academia Española, la cual dice "poner a alguien o algo más allá, en lugar diferente del que ocupaba", un objeto de saber (saber experto, profesional, erudito o sabio) se transpone cuando una comunidad opta por tomarlo como tema factible de ser enseñado de acuerdo con sus intereses; aquí, el fenómeno produce el objeto a enseñar a partir del objeto de saber. Cuando el objeto a enseñar es formalmente llevado al aula de clase, y el profesor interactúa con éste, el objeto a enseñar se transpone generando el objeto de enseñanza que finalmente es "consumido" por los estudiantes. En esta breve descripción, se puede notar la existencia de tres objetos relacionados, pero con identidades particulares: el objeto de saber, el objeto a enseñar y el objeto de enseñanza.

Desde el punto de vista de la producción académica, muchas experiencias de la transposición didáctica en la enseñanza de la matemática e incluso de la biología han sido documentadas; no obstante, la producción académica sobre la aplicación de dicha teoría en la enseñanza de la computación y sus disciplinas asociadas es incipiente a la fecha.

Esta investigación significó un aporte al conocimiento de la aplicación de la transposición didáctica en un escenario específico de la computación, que es la enseñanza de los fundamentos de programación de computadoras en el nivel de pregrado universitario. Para tal efecto, se diseñó y desarrolló dentro del paradigma cualitativo con un enfoque hermenéutico. Con respecto a los aspectos metodológicos, la definición de objetivos, preguntas orientadoras y categorías permitieron vislumbrar las técnicas de recolección de información y su posterior procesamiento; de igual manera, las unidades de análisis y de trabajo fueron conformadas junto con el diseño, validación y despliegue de los instrumentos de recolección de información. Finalmente, los hallazgos por cada categoría fueron descritos en detalle, para llegar al epílogo de la investigación donde son sintetizados los aportes al conocimiento.

1. EL RETO INVESTIGATIVO

En el mundo académico de las disciplinas asociadas a la computación, es común observar a la programación de computadoras como el área de formación donde el ingenio y la capacidad de abstracción son factores claves de éxito. Dicha área contempla una fundamentación teórica, más una serie de conocimientos en lenguajes de programación y herramientas, así como habilidades cognitivas orientadas a la solución de problemas desde un enfoque sistémico. Desde esta óptica, el paso introductorio al dominio de dicha área en las disciplinas asociadas a la computación es el curso de fundamentos de programación.

Posterior a la fundamentación teórica, una actividad común en la enseñanza de los fundamentos de programación es abordar los conceptos básicos del lenguaje (o de los lenguajes) de programación para luego guiar a los estudiantes a través de estrategias donde se contemple la totalidad del proceso de programación de computadoras.

La enseñanza de dichos conceptos básicos de los fundamentos de programación, tales como variables, constantes, expresiones aritmético-lógicas, instrucciones, comparaciones, ciclos y funciones, es un punto clave en la formación de los estudiantes dentro del área de la programación de computadoras; no obstante, estos conceptos básicos son totalmente abstractos. Tener éxito en la programación de computadoras es ser capaz de resolver problemas reales a través del código del lenguaje computacional, haciendo uso de dichos conceptos básicos. Es por tal razón que enseñar a programar computadoras es una tarea compleja dado que los estudiantes requieren del desarrollo de habilidades orientadas a la solución de problemas usando conceptos con un nivel de abstracción elevado; situación que sugiere un reto al momento de enseñar dichos conceptos.

1.1 Una problemática latente

Existe evidencia que sugiere que aprender a programar no es una tarea fácil, por tanto, la enseñanza de los fundamentos de programación conlleva

retos didácticos para el desarrollo de la abstracción y del pensamiento algorítmico (Soloway & Spohrer, 1989, p. 2). Para estos autores, la creación y el control de ambientes y soluciones computacionales a través de la programación son aspectos que para un individuo pueden ser difíciles de realizar. Por esto, la enseñanza de los fundamentos de programación de computadoras es considerada como una actividad compleja y retadora. Si enseñar tiene una dificultad por su nivel de abstracción de los objetos de saber, el aprendizaje del estudiante se ve afectado por tal situación. De hecho, las altas tasas de deserción en fundamentos de programación de computadoras son evidencias de algo que no está bien al impartir dichos cursos. Como ejemplo, un estudio estima que hasta un 80% de los estudiantes en Estados Unidos abandonan sus primeras clases de programación debido a la dificultad que enfrentan para aprender a programar, junto con las dificultades manifestadas por los profesores al momento de enseñar ciertos temas (Carter & Jenkins, 2002).

De igual forma, la enseñanza de un primer curso de programación ha sido objeto de estudio (Ali & Mensch, 2008) cuyo común denominador es la dificultad de enseñar programación de computadoras dada la forma de abordar los objetos de saber de naturaleza abstracta. Esto refuerza la idea de que enseñar a programar se considera una tarea difícil en relación con el aprendizaje que, para la mayoría de los estudiantes, es asociado a objetos de elevada complejidad para ser interpretados en la realidad. Desde la percepción estudiantil, una de las principales razones de deserción de los estudiantes de su proceso de formación son los cursos de fundamentos de programación de computadoras (Anewalt, 2008; Porter & Calder, 2004).

El fenómeno de deserción de los estudiantes de los cursos de fundamentos de programación de computadoras no solo se presenta en los Estados Unidos, dicho comportamiento se ha reproducido en otros países. Es el caso de Colombia, donde los niveles de deserción estudiantil de las áreas del conocimiento de las ingenierías, la arquitectura, el urbanismo y afines —carreras en las que se imparten los programas relacionados con las disciplinas asociadas a la computación— se encuentran en un segundo lugar con un porcentaje de 54% aproximadamente (Ministerio de Educación Nacional, 2013).

Para los profesores de las disciplinas asociadas a la computación, la dificultad de enseñar tales cursos ha sido objeto de debate. Adicionalmente, hay un amplio reconocimiento entre académicos en computación sobre la necesidad de intervenir este problema; tal es el caso de los manifiestos de ACM —*Association for Computing Machinery*— en materia de reformas curriculares y planteamientos de didácticas para hacer frente a la problemática (ACM & IEEE *Computer Society*, 2013). Varios programas en el escenario internacional han adoptado fuertes medidas para compensar estos puntos de dificultad; algunos cambiaron el lenguaje en que se enseña normalmente, se realizaron cambios en la selección del texto guía a utilizar o se tomaron medidas adicionales en otros aspectos de la academia; sin embargo, la dificultad en la enseñanza de los cursos mencionados persistió (Marreno & Settle, 2005).

Pero ¿Por qué realmente es difícil enseñar a programar? Quizás esta pregunta la han formulado algunos profesores que están (o estuvieron) a cargo de fundamentos de programación de computadoras. Sin duda, la respuesta a dicho interrogante aún no ha sido encontrada en su totalidad. A continuación, se presentan algunos factores que, dados los antecedentes investigativos, influyen negativamente en la enseñanza de los citados cursos en el campo de las ciencias computacionales en el ámbito global, y su incidencia en los aprendizajes de los estudiantes.

Un interesante estudio realizado por Baldwin y Kulijis logró identificar algunos factores que dificultan la enseñanza de los fundamentos de programación de computadoras y su impacto directo en los aprendizajes. En dicho estudio, los investigadores señalaron sobre la dificultad que enfrentan los estudiantes durante el aprendizaje de los cursos en mención de la siguiente manera: “la mayoría de los estudiantes, […], encuentran difícil y compleja la tarea cognoscitiva relacionada a la programación de computadoras” y explicaron: “el aprendizaje demanda complejas habilidades cognitivas tales como la planificación, razonamiento y resolución de problemas en programación de computadoras” (Baldwin y Kulijas, 2001, p. 1). Por su parte, los profesores a cargo de los cursos asociados al tema objeto de estudio manifestaron en dicho estudio que, la dificultad de desarrollar habilidades cognitivas asociadas a la solución de problemas a partir de elementos abstractos está siempre presente. Con esto se entiende que existen habilidades de pensamiento que, de no haber sido desarrolladas previamente, se convertirían en factores negativos para el aprendizaje. Entre algunas habilidades de pensamiento se destacan: la capacidad de abstracción, la capacidad de análisis —descomposición en partes—, la capacidad de síntesis —la recomposición de un todo—; todas estas anteriores son habilidades cognitivas fundamentales para la resolución de problemas, cualidades que

son muy complejas de ser desarrolladas cuando se enseña a programar computadoras.

Otros estudios proporcionan un análisis más exhaustivo de los factores que contribuyen a esta dificultad en la enseñanza de dichos cursos. Dann, Cooper y Pausch (2006) los enumeran así: Mecanismos frágiles en la elaboración de programas de computadora, en particular el uso de la sintaxis de los lenguajes de programación; la incapacidad para ver el resultado de los cálculos a la par cuando un programa de computadora se ejecuta, la falta de creatividad a fin de generar espacios de motivación para la programación y, por parte de los estudiantes, la dificultad de comprensión de la lógica compuesta y el desconocimiento en las técnicas de diseño. Lo interesante de este estudio en particular, es la utilización de un lenguaje nuevo de programación donde el factor de motivación se encuentra en cada escenario de uso del lenguaje. El lenguaje en particular es Alice (Carnegie Mellon University, 2013), y fue creado para propósitos educativos en forma exclusiva.

En la fase de planeación para la creación de un programa de computadora, el objetivo del profesor es desarrollar el grado de comprensión en sus estudiantes acerca de los conceptos de los fundamentos de programación de computadoras, tales como la estructura y el flujo total del programa. En el vocabulario de dichos cursos, el término estructura se usa frecuentemente y se practica al momento de escribir programas —que corresponde a la fase de codificación–. En realidad, la estructura otorga al programa de computadoras el orden sobre cómo debe ejecutarse en una compujtadora. En la práctica, la enseñanza del concepto de estructura no ha sido tan sencilla para los estudiantes que no están familiarizados con la construcción de un *software*. Adicionalmente, la evolución misma de los lenguajes de programación ha forzado a generar sólidas estructuras a fin de entender cómo fueron codificados los programas de computadoras, y esto constituye un reto adicional en su enseñanza (Schneider, 1999). He aquí un factor que dificulta enseñar los fundamentos de programación, dado que, para muchos profesores les resulta difícil evidenciar en clase cómo un programa de computadora realiza comparaciones, saltos al interior del código, ciclos e iteraciones, cómo encapsular la información y cómo los objetos pueden realizar herencia y polimorfismo. Entonces, la manera de abordar estas temáticas dentro de las aulas de clase por parte de los profesores influye en forma directa en los aprendizajes de los estudiantes dada la complejidad de los objetos de saber experto.

Frente a esta situación, se entiende que los estudiantes de pregrado que inscriben los cursos de fundamentos de programación de computadoras no tienen experiencia previa en programación con lenguajes computacionales; de hecho, esta experiencia no es el requisito previo. Aquí es posible identificar una brecha entre aquellos estudiantes que en el nivel de bachillerato —*high school*, preparatoria, colegio, bachillerato o Educación Básica Secundaria— han estudiado algunas materias referentes a la programación de computadores y aquellos que no tuvieron esa oportunidad en su ciclo de formación previa. Según las lineamientos nacionales, las competencias relacionadas con tecnología que deberían desarrollar los estudiantes al salir de la secundaria principalmente son: comprender la naturaleza, evolución de la tecnología, su apropiación y uso (Ministerio de Educación Nacional, 2008); en esta última competencia aparecen en forma explícita aptitudes derivadas que apuntan hacia interpretación de diseños, modelos y futuras implementaciones tecnológicas, punto a favor de la necesidad de los fundamentos de programación para alcanzar dichas competencias. La falta de experiencia previa para programar computadoras no parece ser un problema; sin embargo, sí es un inconveniente el bajo desarrollo de habilidades para la resolución de problemas. Dunican (2002) indica que los objetos de saber experto ofrecidos en las escuelas secundarias no incluyen los módulos de lógica y/o solución de problemas, poniendo a los estudiantes en una situación difícil cuando se inscriben en cursos sobre programación de computadoras en el nivel universitario, específicamente en las disciplinas asociadas a la computación. De igual forma, Stamouli, Doyle y Huggard (2004) también señalaron la falta de continuidad en los estudios de aquellos sujetos que salen de las escuelas secundarias e ingresan al primer año de su carrera universitaria. Esto evidencia que ese “salto abrupto” interfiere significativamente en su aprendizaje en la universidad.

Aunque el nivel de alfabetismo en TIC —Tecnologías de la Información y las Comunicaciones— no es tan alto entre algunos de los estudiantes que ingresan a carreras asociadas a la computación —dado que previamente han recibido formación al respecto en la básica secundaria—, la mayoría de ellos tiende a la carencia de experiencia específica en programación de computadoras. Aquí es preciso resaltar que saber manejar tecnología basada en computación es muy diferente a saber programarla con el fin de resolver problemas puntuales. Esto incluye no sólo las fases de diseño y construcción, sino también las tareas de compilación, depuración o ejecución de un programa de computadora, o, de hecho, la comprensión básica del

modelo computacional con sus componentes de hardware y software. Esta falta de comprensión del modelo mental de una computadora —es decir, cómo el ser humano ve a una computadora desde el punto de vista meramente conceptual—, a menudo resulta frustrante cuando los resultados no muestran lo que el estudiante había planeado previamente (Ben-Ari, 1998). En complemento a lo anterior, otra dificultad que enfrentan los estudiantes de programación de computadoras es la necesidad de imaginar y comprender muchos términos abstractos que no tienen equivalentes en la vida real: ¿Cómo se relaciona una variable, un tipo de datos o una dirección de memoria a un objeto de la vida real? De esta manera, muchos de los conceptos de programación de computadoras tienden a ser difícil de entender dado que no son fácilmente adaptables a la vida real (Dunican, 2002).

En consecuencia, muchos estudiantes de fundamentos de programación asumen una actitud de poco esfuerzo por comprender los conceptos que se abordan en el contenido de dichos cursos (Stamouli et al., 2004; Thomas et al., 2002). La forma en que los objetos de saber experto son abordados impacta en el aprendizaje de los estudiantes, dado que conceptualmente es difícil buscar símiles de dichos objetos con elementos de la vida real; punto clave para los estudios en didáctica donde se aborda esta problemática de índole conceptual. Entonces, es aquí donde la capacidad de abstracción juega un papel fundamental en la forma de diseñar e implementar programas de computadora.

Piteira y Costa (2013, p. 76) a través de su investigación relacionada con el estudio de las dificultades de la enseñanza de tales cursos, encontraron que se destacan en términos generales los siguientes 11 problemas asociados a los conceptos básicos de programación dado su gran nivel de abstracción: variables, estructuras de selección, estructuras cíclicas, parámetros, arreglos, punteros y referencias, tipos de estructuras de datos, tipos de datos abstractos, manejo de entrada y salida, manejo de errores, y uso de bibliotecas del lenguaje.

Por su parte, Shuhidan, Hamilton, y D'Souza (2011, p. 217) encontraron que a pesar de la creencia de que el conocimiento básico es fácil de aprender, los estudiantes novatos enfrentan grandes problemas de índole conceptual en materia de fundamentos de programación de computadoras; paralelo a esto, la forma de evaluar el aprendizaje incide en la forma en que los estudiantes aprenden –específicamente, en su forma de estudiar–.

Entonces, para estos autores, la enseñanza de los cursos citados enfrenta problemas conceptuales debido a la complejidad de sus constructos teóricos, difíciles de abordar sin el desarrollo previo de la capacidad de abstracción del estudiante.

Dentro de la interacción de la enseñanza y el aprendizaje, la visión por parte de los que enseñan los fundamentos de programación de computadoras en el nivel de pregrado evidencia un escenario de un pobre aprendizaje debido a que los conceptos básicos en cuanto a lógica y abstracción no tienen bases sólidas. De igual forma, los autores afirman que estas dificultades de enseñanza conllevan a que "algunas personas pueden usar lenguajes de programación, pero muy pocas de ellas lo hacen y con un gran esfuerzo" (Guzdial & Guo, 2014, p. 10).

Con este panorama, es posible observar que la enseñanza de los fundamentos de programación de computadoras no es, en términos generales, una tarea fácil; ya que el reto de enseñar involucra generar espacios para potenciar características cognitivas desde el punto de vista de la abstracción, la descomposición, la resolución de problemas y la reutilización. Con base en las anteriores referencias, una de las características comunes que puede ser destacada frente a esta problemática centra su atención en la forma en que se abordan los conceptos básicos en el contenido de dichos cursos; al parecer, esta problemática de enseñanza y de aprendizaje se relaciona con la naturaleza misma de los objetos del saber experto que pretenden ser llevados al aula y convertirlos en objetos factibles de ser enseñados. Se debe entonces realizar aportes a fin de enriquecer las prácticas de enseñanza de los fundamentos de programación de computadoras, atendiendo la naturaleza abstracta de los objetos de saber experto, los cuales son llevados a las aulas de clase para ser enseñados.

Considerando lo anterior, es interesante preguntar:

¿Cómo enriquecer las prácticas de enseñanza de los fundamentos de programación de computadoras a través de la transposición didáctica?

¿Cuáles son las características de los objetos seleccionados de saber de los fundamentos de programación de acuerdo con los referentes internacionales?

¿Cuáles son los niveles de transposición didáctica de los objetos seleccionados de saber de los fundamentos de programación de computadoras?

¿Cuáles son las prácticas de enseñanza de los objetos seleccionados de saber transpuestos de los fundamentos de programación de computadoras?

1.2 El fin investigativo

La transposición didáctica enriquece experiencias de enseñanza de los fundamentos de programación de computadoras.

Esta investigación se sustenta en la posibilidad de motivar a los estudiantes en el aprendizaje de la programación de computadoras a través del manejo de conceptos situados en el contexto y orientados a problemáticas reales con un enfoque práctico. La transposición didáctica contribuye sustancialmente en dicha motivación al aprendizaje de la programación de computadoras a través de un ejercicio de diseño de contenidos junto con sus estrategias de enseñanza adecuadas. Por tratarse de conceptos de carácter universal, al transponer dichos saberes al interior del aula de clase, su enseñanza debe respetar su esencia epistemológica garantizando su fácil comprensión por parte de los estudiantes.

De esta manera, la investigación busca:

Enriquecer las experiencias de enseñanza en los fundamentos de programación de computadoras a través de la transposición didáctica.

Caracterizar los objetos seleccionados de saber de los fundamentos de programación de computadoras, de acuerdo con los referentes internacionales.

Analizar los niveles de transposición didáctica de los objetos seleccionados de saber de los fundamentos de programación de computadoras.

Proponer prácticas de enseñanza para los objetos seleccionados de saber transpuestos de los fundamentos de programación de computadoras.

1.3 Motivación para la investigación

La computación contempla una serie de disciplinas cuyos fundamentos y métodos están asociados al software, los dispositivos hardware, y el apropiado tratamiento de información (Sommerville, 2011, p. 9). En este sentido, dichas bases teóricas deben ser soportadas por un conocimiento en algoritmia y programación de computadoras, los cuales permiten llevar a cabo los proyectos en escenarios reales.

En este estadio, gran parte de la deserción estudiantil en el nivel de pregrado en computación está asociada a fundamentos de programación y en los cursos de fundamentación matemática y física (Timarán, 2010). Varios factores han sido identificados, tales como: deficientes bases conceptuales del bachillerato sobre todo en el área de matemáticas, prácticas pedagógicas inadecuadas, debilidades en los hábitos de estudio y aprendizaje, entre otros. Al mismo tiempo, una sólida educación en matemáticas y fundamentos de programación tiene una considerable incidencia en las disciplinas asociadas a la computación (Carter, 2006 p. 27); he ahí la razón por la cual los fundamentos de programación de computadoras son un elemento esencial en la formación de profesionales en las disciplinas asociadas a la computación.

De manera general, se puede definir la computación como cualquier actividad orientada a objetivos que requiera, se beneficie de, o crea, sistemas de computadoras (ACM, AIS, & IEEE Computer Society, 2005, p. 9). Por lo tanto, la computación incluye el diseño y la construcción de sistemas de hardware y software para una amplia gama de propósitos: procesar, estructurar y administrar varios tipos de información; llevar a cabo estudios científicos usando computadoras; hacer que los sistemas computacionales se comporten de manera inteligente; crear y usar medios de comunicación y entretenimiento; encontrar y recopilar información relevante para cualquier propósito, entre otros. Las aplicaciones de la computación en el mundo actual son prácticamente innumerables, y las posibilidades son enormes. Por esta razón, los fundamentos de programación forman parte esencial de la computación la cual está inmersa en cada aspecto de la vida del ser humano. Para las recomendaciones curriculares en computación propuestas abiertamente en el 2005 por ACM, AIS e IEEE Computer Society, los programas académicos han demostrado que pueden formar estudiantes con habilidades sobresalientes en dichas temáticas, y que efectivamente éstas constituyen pilares esenciales para la construcción de sistemas de software complejos que requieren experiencia en ingeniería de software (ACM, AIS, & IEEE Computer Society, 2005, p. 33).

Las buenas estrategias para la enseñanza de los fundamentos de programación de computadoras en el nivel de pregrado en computación pueden resolver los problemas citados. Al considerar que la labor de enseñar dichos cursos tiene una dificultad particular, la exploración de alternativas de enseñanza gana relevancia. Así, la utilización de la transposición didáctica permite un análisis más exhaustivo sobre la naturaleza de los objetos de saber que se pretende llevar hacia el ambiente educativo. Al ser los niveles de abstracción uno de los focos problemáticos que son manejados por los objetos de saber, es importante realizar el proceso de vigilancia epistemológica en el contexto de aplicación de dichos objetos, con el fin de orientar prácticas de enseñanza con incidencia positiva frente a los aprendizajes.

La transposición didáctica es ampliamente utilizada en la enseñanza de la matemática, e incluso de la biología (Hazzan, Meerbaum-Salant & Dubinsky, 2010, p. 33); no obstante, su aplicación en escenarios de la computación y sus disciplinas asociadas es incipiente. Por tal razón, esta investigación pretendió realizar aportes a la enseñanza de los fundamentos de programación de computadoras a través de la transposición didáctica.

Dada la revisión sistemática de literatura, es frecuente encontrar que la dificultad principal al enseñar fundamentos de programación radica en el hecho de trabajar con conceptos muy abstractos; en este sentido, la transposición didáctica puede hacer aportes importantes al trabajar con la naturaleza de dichos objetos de saber experto a fin de ser convertidos en objetos factibles de ser enseñados, basándose en experiencias exitosas de diferentes contextos. Cabe resaltar la importancia del desarrollo de las investigaciones propuestas por Carvalho y otros (2014) junto con Maréchal y otros (2012) las cuales involucran enfoques interdisciplinares en aras de enriquecer la producción de conocimiento desde varias perspectivas. En la presente investigación, dos campos de saber se entrelazan: las ciencias de la educación y la computación. Esta particularidad del Programa de Doctorado ha demostrado su gran capacidad de inclusión desde el punto de vista multidisciplinar.

Según Bogdanovych y Trescak (2017), un factor clave para conseguir buenos resultados en cuanto al aprendizaje de los fundamentos de programación es la motivación, sobre todo, entendiendo el contexto de los actuales estudiantes. Por tal razón, estos autores promueven escenarios basados en juegos con el uso de claros ejemplos visuales. Dentro del pensamiento motivacional al momento de enseñar los fundamentos de programación, el éxito de quienes comienzan a aprenderlos está fuertemente ligado a los factores de motivación, especialmente interna, relacionada con el interés real (Andrzejewska, 2018).

Finalmente, este tipo de investigaciones son necesarias para mejorar procesos de enseñanza-aprendizaje, cualificar el programa y fundamentar la didáctica de los docentes que ejercen en el mismo. En este particular, la investigación constituye un importante insumo para la formulación de estrategias de enseñanza que son aplicables al Programa de Ingeniería de Sistemas de la Universidad de Nariño. Factor importante que puede ser tomado como referencia al momento de realizar procesos de reforma curricular y autoevaluación, entre otros.

2. FUNDAMENTACIÓN TEÓRICA

Quizás la primera actividad que se realiza en toda investigación se relaciona con determinar el estado del arte del objeto de estudio. En este sentido, la fundamentación teórica inició con la identificación y el análisis de experiencias que involucran asuntos de enseñanza de los fundamentos de programación en el escenario mundial.

El desarrollo de la investigación implicó su ubicación dentro de un marco de legalidad. Este punto es desarrollado en el marco normativo del presente capítulo. Por tratarse de una investigación al interior del país, su normatividad es tomada de las leyes colombianas; no obstante, se logró la participación de profesores extranjeros, haciendo que el contexto de la investigación se extienda más allá del escenario nacional.

Por último, se realizó un compendio de teorías, análisis y reflexiones para dar soporte teórico a la investigación. Esta parte de los conceptos fundamentales de didáctica, de la teoría de la transposición didáctica en términos generales, de la teoría de la transposición didáctica en la enseñanza de la computación, de los fundamentos de programación, y de las disciplinas asociadas a la computación.

Con esta parte final del capítulo, se cimienta la investigación a partir de los conceptos que en ella se manejan. Al mismo tiempo, sugerencias sobre los puntos esenciales fueron desarrolladas en el proceso investigativo, de acuerdo con el diseño metodológico.

2.1 Antecedentes

Las experiencias citadas aquí, tratan sobre los problemas de la enseñanza de fundamentos de programación de computadoras. En términos generales, los factores estudiados por dichas experiencias incluyen: el grado de abstracción de los objetos de saber, la creatividad para generar espacios de motivación, la permanencia estudiantil en el campus o fuera del campus, el dominio del idioma inglés —tanto de profesores como estudiantes—, el in-

terés por las temáticas, los respectivos conocimientos previos en programación de computadoras, entre otros. De esta síntesis, las experiencias muestran un panorama de conceptos que no son tan fáciles de enseñar, y que, su incidencia en los aprendizajes tiene efectos asociados a la problemática de bajo rendimiento y deserción académica. Una semilla germinal de dicha problemática apunta hacia la forma en que son abordados los objetos de saber experto dentro de los escenarios académicos; lo que se traduce en un problema de enseñanza.

Aquí se presenta un compendio sobre los hallazgos referentes a los problemas de enseñanza de los fundamentos de programación, un planteamiento de posibles soluciones y las experiencias previas en materia del estudio de la transposición didáctica en la enseñanza y aprendizaje de la computación.

Si bien es cierto que se ha logrado identificar algunos factores que inciden negativamente en el aprendizaje de los objetos de saber de fundamentos de programación, la producción a manera de propuestas didácticas para abordar dicha problemática no ha sido prolífera. En este sentido, desde los escenarios de la didáctica, se han formulado propuestas a fin de mejorar el aprendizaje de los objetos de saber en otras áreas del saber —como la matemática, la física y la química—; no obstante, en materia de fundamentos de programación, la producción de didácticas no es abundante como se evidencia en las áreas mencionadas.

Una serie de estudios realizados, identifican la dificultad de aprender un nuevo lenguaje de programación de computadoras y las medidas sugeridas para simplificar este proceso de aprendizaje. Estas sugerencias van desde cambiar el lenguaje de programación, hasta unas más detalladas que tienen que ver con el modelo conceptual y el paradigma de la enseñanza de los lenguajes de programación de computadoras.

Herbert (2007, p. 37) señala:

> *(...) para facilitar los procesos de enseñanza-aprendizaje de los fundamentos de la programación de computadoras, se deben mantener los siguientes elementos: minimizar la sintaxis de programación, retroalimentar visualmente la ejecución de programas informáticos, acortar el ciclo creativo de conceptualización, y mejorar la implementación y los resultados obtenidos tras la ejecución de un programa informático*[1].

Un estudio explica acerca del uso del modelo conceptual para la enseñanza orientada a la comprensión del lenguaje de programación por parte del estudiante. Dicho estudio argumenta que los modelos conceptuales en la enseñanza pueden servir para mejorar la comprensión de los estudiantes de programación. Los métodos utilizados para mejorar el desarrollo de modelos conceptuales precisos incluyen: un diseño de la interfaz para que los usuarios pueden interactuar activamente con ella; el uso de metáforas y analogías para explicar los conceptos; y el uso de relaciones espaciales para que los usuarios puedan desarrollar capacidades para la simulación mental (Baldwin & Kulijas, 2001, p. 1).

Dann, Cooper y Pausch (2006) señalan algunos temas que los estudiantes deberían aprender en fundamentos de programación, y que es responsabilidad de los profesores saber enseñarlos: pensamiento algorítmico, expresión, abstracción y apreciación de la realidad en detalle. Además, explican que, para resolver los problemas relacionados con cursos introductorios de programación de computadoras, el profesorado debe incluir ejemplos de la vida real en los cuales hayan participado a fin de capturar la atención de los estudiantes.

En cuanto a la forma cómo son abordados los objetos de saber experto dentro del aula, Herbert (2007, p. 127) escribió que “la mejor manera de enseñar las ideas de los fundamentos de programación de computadoras es mediante una exposición suave ante los estudiantes, para luego ir añadiendo más detalles hasta abarcar en profundidad un objeto de saber”. Este tipo de enfoque para aprender los fundamentos de programación se conoce como el enfoque de espiral. El proceso puede ser largo y a veces tedioso; por tanto, los profesores necesitan motivar a los estudiantes a lo largo del camino para mantenerlos interesados en los objetos de saber.

Desde el punto de vista tecnológico, los nuevos lenguajes de programación han evolucionado a tal punto que sus entornos integrados de desarrollo —del inglés: IDE – *Integrated Development Environment*— son una ayuda significativa en el proceso de construcción de programas de computadora.

1 Párrafo traducido del inglés por parte del autor.

Estas herramientas ayudan a mejorar el uso de la sintaxis de los lenguajes de programación y potencian la reutilización de activos de *software*. Con esto se tiene que las diferentes propuestas de orden didáctico siempre sugieren el uso de este tipo de herramientas para facilitar los procesos de aprendizaje en la construcción de programas de computadora.

Por otro lado, Guibert, Girard y Guitet (2004) hicieron hincapié en la experiencia positiva del uso de programación, por ejemplo —del inglés PbE – *Programming by Example*— donde los estudiantes diseñan métodos para proporcionar retroalimentación continua durante la ejecución del programa. El hecho de proveer dicha retroalimentación permanente involucra al estudiante en el programa, haciéndolo a la vez consciente de lo que está sucediendo durante la ejecución del programa en la computadora. En este punto, es pertinente el uso de depuradores —del inglés: *debuggers*— integrados al entorno de programación a fin de hacer conciencia sobre lo que el programa hace paso a paso.

En un estudio realizado por Hartman, Nievergelt y Reichert (2001) se sugiere el uso de máquinas de estados finitos para la enseñanza de los fundamentos de programación de computadoras y se señala además que "se debe ver a la programación practicada como un ejercicio educativo donde es mejor aprendida en un ambiente de juego, libre de la preocupación utilitaria, diseñada para ilustrar los conceptos seleccionados en la configuración más simple" (p.1) El estudio sugiere además, introducir los fundamentos de programación para principiantes en un entorno de juego, donde a través de acciones limitadas es posible aprender a controlar rutinas simples, tales como condicionales, ciclos e iteraciones. El propósito del uso de las máquinas de estado finito y los juegos de azar es estimular el aprendizaje de objetos de saber experto a través de la lúdica y la motivación.

Así mismo, la articulación del mundo real con los fundamentos de programación es un elemento crucial para mejorar el aprendizaje de sus objetos de saber. El uso de objetos que se asemejan a instancias de vida real ayuda a la comprensión conceptual de las características mencionadas en fundamentos de programación. En muchos casos, los lenguajes de programación introducen objetos abstractos en representación de los objetos de la vida real. Estas abstracciones son utilizadas para facilitar el aprendizaje de los nuevos conceptos de programación orientada a objetos —del inglés: *Object-Oriented Programming*–. Por ejemplo, una persona puede ser considerada como un objeto desde el punto de vista de la programación de computadoras; así, la persona tiene propiedades —peso, talla, entre otras—, se le pueden crear métodos —tales como correr, caminar— y funciones para dichos objetos. De esta forma, un programa que muestra las manipulaciones de diversos elementos basados en el objeto persona —peso, talla— puede ser más comprensible que aquellos programas estáticos que muestran manipulación de texto aislado y cálculos sencillos.

En el campo de la didáctica, la analogía es una de las técnicas de enseñanza empleada para el aprendizaje de los fundamentos de programación de computadoras. Esta técnica es particularmente útil al enseñar los fundamentos de programación para el desarrollo de conceptos tales como entrada/salida, tipos de datos, búsquedas, clasificación, etc.; la analogía utiliza ejemplos ilustrativos de conceptos que los estudiantes han visto antes, de tal suerte que dichos conceptos familiares se relacionan con los nuevos conceptos. En la analogía, el concepto familiar es identificado como la fuente y el nuevo concepto como el objetivo, y cuando se hace la analogía, la fuente se asigna al objetivo (Blanchette & Dunbar, 2000). Dunican (2002) describe varias analogías, por ejemplo, el uso de juguetes infantiles para enseñar las declaraciones de misión; el uso de cajas para determinar el número más pequeño y el más grande en una lista; y el uso de un casillero de correspondencia para explicar el concepto de manipulación de datos en una matriz.

Otra importante faceta didáctica es el concepto de pertinencia. En tal sentido, los estudiantes deben ver un fin en lo que están aprendiendo. Sheard y Hagan (1998) informan sobre la respuesta positiva de los estudiantes después de los ejercicios de programación de computadoras con interfaces visualmente atractivas, al estilo de los videojuegos. Dichas interfaces fueron utilizadas para mostrar los beneficios del paradigma orientado a objetos. Esta situación proporcionó una oportunidad para explicar las ventajas de diseño y de programación orientada a objetos de otros estilos de programación de aplicaciones complejas. Una vez más, los objetos de saber son transformados en objetos atractivos y motivadores de aprendizaje.

En el contexto curricular se han creado diversos tipos de intervenciones para ayudar a los estudiantes a desarrollar habilidades de programación. Las intervenciones variaron entre los cambios del currículo, la pedagogía y la evaluación, para conseguir el apoyo adicional a los nuevos estudiantes. En este sentido, Van Roy, Armstrong, Flatt y Magnusson (2003, p. 270) ba-

saron las unidades académicas —entendiéndose como cursos, materias o asignaturas— en conceptos, en lugar de tipos de lenguajes o paradigmas individuales —tales como programación orientada a objetos, programación lógica o la programación funcional, entre otros–. El hecho de haber enseñado durante dos años con este enfoque en cuatro universidades, tanto de Europa como de Estados Unidos, ha permitido descubrir que los estudiantes razonen de manera amplia y profunda sobre el diseño de sus programas, su corrección y su complejidad.

Dos enfoques diferentes para el diseño curricular han sido utilizados y probados en diversas instituciones: el enfoque de enseñanza de orientación a objetos en primera instancia y el enfoque de la enseñanza de la programación estructurada; estos dos enfoques se han divulgado como casos exitosos. Sin embargo, Sheard y Hagan (1998, p. 315) observaron que los estudiantes "comenzaron a sentirse perdidos, [...] casi al mismo tiempo cuando se introdujo un nuevo paradigma al curso". En consecuencia, se decidió utilizar en primer lugar el enfoque *bottom-up* —programación estructurada en primera instancia— de forma tradicional e introducir luego los conceptos orientados a objetos después de que los estudiantes han ganado una comprensión en materia de expresiones, declaraciones y parámetros. Este fue uno de los cambios introducidos al curso desde el punto de vista curricular y no didáctico para dicho estudio, y así como resultado, "fue encontrado un aumento significativo en el rendimiento después del cambio curricular en materia de fundamentos de programación" (Sheard & Hagan, 1998, p. 319).

Los conceptos de programación también han sido abordados en investigaciones diferentes a la producción de *software*. Tal es el caso de la interacción de los fundamentos de programación y la robótica. Misirli y Komis (2014) presentan un marco de trabajo sólido para establecer la relación entre los fundamentos de programación y la robótica en escenarios de Educación Básica Primaria. Por otra parte, una experiencia en Argentina toma una muestra de estudiantes de un curso de desarrollo *Web* usando fundamentos de programación en el lenguaje *Python* (Reynoso *et al.*, 2013); los hallazgos son útiles no sólo para la transposición didáctica en la enseñanza de cursos que tengan en cuenta el equilibrio de las preferencias de los estudiantes, sino también para desarrollar nuevos métodos y programas de enseñanza que se centran en los procesos cognitivos de los estudiantes.

Finalmente, otro enfoque se basa en el uso de tecnología para la enseñanza. Clancy, Titteron, Ryan, Slotta y Linn (2003) describen un modelo basado en laboratorios para la enseñanza de las ciencias computacionales. Su modelo incluye tres componentes: un constructor del curso en línea para el profesor, un entorno de aprendizaje basado en *Web* para la entrega de todas las actividades del estudiante y un portal del curso que sirvió como un sistema de gestión de aprendizajes. La evaluación del sistema demuestra que se mejora el rendimiento de los estudiantes, cambiando a su vez su percepción que tenían acerca del curso. Dicha evaluación muestra que los estudiantes consideran agradable el curso. Sin embargo, el nuevo modelo no tuvo ningún impacto en la tasa de deserción del curso.

En el contexto hispanoamericano, son escasos los estudios en transposición didáctica aplicada a la computación. Quizás el caso más destacado en Hispanoamérica es aquel que realiza reflexiones hacia la construcción de una didáctica de la informática en el contexto argentino, dado que, desde la perspectiva de Chevallard, se requiere conocer con claridad el campo de estudio de la informática partiendo desde su indagación como ciencia, disciplina, tecnología y metodología (Caraballo & Cicala, 2006). En este estudio, la valoración de contenidos y el ejercicio de la vigilancia epistemológica son puntos clave en la legitimación de la informática en espacios curriculares para la formación profesional.

En contraste a lo anterior, existen estudios relacionados con transposición didáctica en ciencia aplicada diferente a la computación. Son los casos de la ciencia experimental como la biología y la química.

El abordaje de los problemas didácticos desde el estudio de la enseñanza en una ciencia experimental concluye en el reconocimiento de transformar el saber experto en saber factible de ser enseñado, el autor de este estudio afirma que "Toda transposición o recontextualización didáctica transforma la ciencia practicada por los científicos en ciencia escolar o escolarizada o, de otra manera, en ciencia didactizada." (Gallego, 2004). En este estudio se plantea que existen diferentes visiones acerca del saber específico, dichas visiones van desde quienes elaboraron los conceptos científicos hasta aquellos que hacen el trabajo didáctico de transponerlo para que dichos conceptos sean factibles de ser enseñados. También afirma este autor que, la multiplicidad de transposiciones para cada interpretación de las intencionalidades curriculares se encuentra determinada por las concepciones

epistemológicas, didácticas y pedagógicas de quien o quienes profesionalmente ejercen como educadores de las ciencias. En cuanto al saber original, se hace alusión a los artículos publicados en las revistas especializadas en las que se sometieron esos modelos de transposición a consideración de la comunidad de especialistas. Así, por ejemplo,

> *(...) podría traerse a cuento la historia del desarrollo de la construcción, admisión y sustitución de los modelos atómicos, desde J. J. Thompson, pasando por Rutherford, hasta Bohr, si se quiere. Se es consciente de que el punto de vista que se toma es restringido al caso de la física. Sin embargo, en las otras ciencias de la naturaleza la idea de originales podría extenderse a libros en los que los nuevos modelos cumplieron históricamente el mismo cometido* (Gallego, 2004, p. 308).

Siguiendo a Gallego (2004), se debe tomar en cuenta las lógicas de pensamiento de los ejercicios didácticos donde la transposición se hace evidente a fin de realizar contrastes. Dicha contrastación se realiza a través de la formulación y práctica de las estrategias de enseñanza correspondientes. De esta manera, el desempeño profesional del educador de las ciencias se convierte en trabajo investigativo. Estas reflexiones fueron concebidas y desarrolladas al interior del Grupo de Investigación: Representaciones y Conceptos Científicos —IREC— de la Universidad Pedagógica Nacional en Colombia.

La investigación propuesta por Cuéllar, Pérez y Quintanilla (2005) realiza el estudio sobre los libros de texto de la enseñanza del modelo de Rutherford, tanto en libros de Educación Media como de Educación Superior. En ella se resalta el ejercicio de la vigilancia epistemológica como argumento de crítica a la producción escrita usada en formación media y profesional. Los autores de dicha investigación invitan al profesor a cargo de materias de química a tomar una actitud investigadora, no sólo frente al saber disciplinar, sino a su ejercicio docente en cuanto a las concepciones epistemológicas, pedagógicas y didácticas que orientan su actividad. Este aporte se realizó en el contexto colombiano, a fin de establecerse la imagen de ciencia que se socializa en el sistema educativo del país. Para tal efecto, se seleccionaron quince libros de texto, diez de Educación Superior, los más utilizados en las Universidades Pedagógica Nacional y Distrital de Bogotá y cinco de Educación Media.

Finalmente, en esta revisión de literatura se puede evidenciar que existen antecedentes importantes desde el punto de vista de cómo enseñar dichos

cursos. Sin embargo, solo una investigación en el contexto argentino ha logrado relacionar la transposición didáctica a la enseñanza de las ciencias computaciones en términos generales. Hasta la fecha no se ha encontrado evidencia de una investigación que asocie la transposición didáctica en la enseñanza específica del tema en cuestión.

2.2 Contexto de la investigación

Siendo globalmente reconocida la importancia de los fundamentos de programación como curso introductorio en las disciplinas asociadas a la computación, es conveniente indagar en diferentes contextos acerca de lo que ocurre en la enseñanza de dicho curso. Por tal razón, se diseñó la investigación para ser realizada desde la Universidad de Nariño con la participación de profesores en diferentes universidades del mundo.

Esta investigación toma como objeto de estudio la enseñanza de los fundamentos de programación dentro de cursos formales a nivel universitario. Dada la gran cantidad de programas universitarios relacionados con la computación donde se imparten dichos cursos, el contexto general de la investigación es global; de esta forma se tomó muestra de las universidades del mundo en el campo de la ingeniería y la computación según el *ranking* mundial dado por la ARWU[2]. En adición, se incluyó también una muestra de universidades nacionales. Los criterios de selección se presentarán más adelante.

La computación y sus disciplinas asociadas tienen presencia en la mayoría de las instituciones de Educación Superior en el mundo, con nombres como: ingeniería de sistemas, ingeniería en computación, ciencia de computadoras, ingeniería de *software*, ingeniería informática, entre otros. Estos programas académicos, que comulgan con las definiciones disciplinares de la computación, involucran sin excepción en su currículo la formación en programación de computadoras, probablemente, algunos programas profundizan más que otros en esta materia; sin embargo, cierto es que to-

2 El *Ranking* Académico de Universidades Mundiales (ARWU) fue publicado por primera vez en junio de 2003 por el Centro de Universidades de Clase Mundial (CWCU), Escuela de Posgrado de Educación (anteriormente el Instituto de Educación Superior) de la Universidad de *Shanghai Jiao Tong*, China. Desde 2009, *ShanghaiRanking Consultancy* ha publicado y registrado el Ranking Académico de Universidades Mundiales (ARWU). *ShanghaiRanking Consultancy* es una organización totalmente independiente sobre estudios de Educación Superior y no está legalmente subordinada a ninguna universidad o agencia del gobierno.

dos involucran la fundamentación en dicha área según los hallazgos al momento de analizar sus contenidos de estudio.

Esta investigación hizo uso de la infraestructura y los recursos de la Universidad de Nariño en la ciudad de Pasto – Colombia; no obstante, la recolección de información involucra universidades localizadas en diferentes países teniendo en cuenta la clasificación "*Academic Ranking of World Universities in Engineering/Technology and Computer Sciences*" de 2016, así como algunas universidades colombianas. De esta forma, se pudo indagar sobre aspectos importantes en la enseñanza de los fundamentos de programación en el escenario mundial, tomando como base una muestra de la población de profesores que imparten dichos cursos en el *top* 50 de las universidades del mundo con un enfoque en ingeniería, tecnología y computación; y junto a profesores del contexto colombiano, los cuales pertenecen al *top* 84 de universidades colombianas que han obtenido los resultados por encima del promedio nacional en las pruebas de estado Saber Pro en el área específica de Diseño de *Software* en el 2016. Estos aspectos se abordarán en el capítulo sobre el diseño metodológico.

En resumen, se puede decir que el contexto de la investigación es nacional e internacional. El desarrollo de ella fue llevado a cabo en la Universidad de Nariño; sin embargo, contó con la participación de 9 profesores cuyas nacionalidades son de Estados Unidos, China, Francia, Suiza y Australia respectivamente, además de la participación de cuatro profesores colombianos, también se entrevistarona dos expertos, uno de España y otro de Colombia. Adicionalmente, la temática abordada en la investigación es de dominio general si se aborda desde las disciplinas asociadas a la computación. Como se ha mencionado anteriormente, los fundamentos de programación de computadoras es un curso introductorio común en los programas de formación de pregrado de ciencia de computadoras, ingeniería de computadoras, ingeniería de *software*, sistemas de información y de tecnología de información, según las recomendaciones de ACM, IEEE *Computer Society*, y AIS (2005).

2.3 Base teórica-conceptual

La investigación se fundamenta en una revisión sistemática de literatura basada en la recopilación de experiencias documentadas y en una serie de teorías alrededor del objeto de estudio. El primer concepto importante que

se tiene en cuenta es el concepto de didáctica, dado que la investigación centra su quehacer en la enseñanza. Con esto, es necesario el estudio de diferentes definiciones de dicho concepto a partir de propuestas de varios autores a lo largo de la historia, hasta elaborar una síntesis del concepto de didáctica que se maneja en el desarrollo de esta investigación.

En segunda instancia, la teoría de la transposición didáctica aparece en escena como eje fundamental de la investigación. En sus inicios, dicha teoría se basa en el principio de que "toda práctica de enseñanza de un objeto presupone, en efecto; la transformación previa de su objeto en objeto de enseñanza" (Verret, 1975, p. 140). De esta forma, la teoría propuesta por el profesor Yves Chevallard es de gran importancia ya que conforma el sustento teórico en el cual se fundamenta esta propuesta y el aporte al conocimiento derivado del proceso investigativo.

En tercer lugar, aparece el desarrollo de un discurso de reflexión sobre los críticos y los defensores de la teoría de Chevallard y el concepto de noosfera, derivado de la misa. En este punto, se determinan los aspectos positivos y negativos de la propuesta de Chevallard.

En cuarto lugar, se hace alusión a los estudios de transposición didáctica en la enseñanza de la computación. Finalmente, se presentan las definiciones y características de las disciplinas asociadas a la computación; éstas en conjunto definen el *corpus* de conocimiento que ha sido considerado como el referente mundial para la construcción curricular —al menos en los contenidos— de los programas académicos asociados a la computación. En esencia, las disciplinas determinan los enfoques profesionales de las carreras que se imparten en el mundo con relación a la computación.

2.3.1 Elementos fundamentales de la didáctica

Un criterio válido para acercarse a la definición del concepto de didáctica es revisar lo que algunos autores han producido al respecto. A continuación, la Tabla 1 presenta en orden cronológico una muestra de aportes conceptuales realizados por diferentes autores desde sus perspectivas hacia la elaboración del concepto de didáctica.

Tabla 1. Síntesis de definiciones y elementos de didáctica

Autor	Definición de didáctica	Elementos
Comenio (1640)	Artificio universal para enseñar	Rapidez, alegría, eficacia
Mattos (1963)	Disciplina pedagógica, práctica y normativa	Técnica de enseñanza, dirigir y orientar estudiantes
Stocker (1964)	Teoría de instrucción	Todos los niveles de escolaridad
Tomaschewsky (1966)	Teoría general de enseñanza	Leyes de instrucción, educación en la clase.
Larroyo (1970)	Estudio de métodos y procedimientos	Enseñanza y aprendizaje
Nerici (1973)	Ciencia y arte de enseñar	Técnicas, principios y procedimientos
García (1974)	Enseñanza con una finalidad instructiva	Trabajo discente y docente
Fernández (1974)	Ciencia que estudia el trabajo docente y discente	Métodos de enseñanza y aprendizaje, instrucción.
Mello (1974)	Planificación	Fundamentos de la acción, orientación de aprendizajes
Titone (1981)	Ciencia de dirección del proceso de enseñar	Fines inmediatos y remotos, eficacia instructiva y formativa
Pérez (1982)	Ciencia y tecnología de comunicación intencional	Proceso enseñanza-aprendizaje, formación intelectual
Bruera (1985)	Saber desde las estructuras epistemológicas de las ciencias	Propuestas instrumentales, situación de clase
Gimeno (1985)	Disciplina científica para guiar la enseñanza	Saberes, guiar la acción
Darós (1987)	Ciencia y tecnología (conjunto de técnicas)	Contenidos, procesos, facilitar aprendizajes
Benedito (1987)	Ciencia y tecnología, a partir de la teoría y la práctica	Comunicación intencional, enseñanza-aprendizaje
Aebli (1988)	Ciencia auxiliar de la pedagogía	Tareas educativas más generales, formación intelectual, metodologías para procesos formativos
Rosales (1988)	Ciencia del proceso de enseñanza sistemática	Optimización del aprendizaje
Zabalza (1990)	Campo de conocimientos	Procesos de enseñanza-aprendizaje
Vasco (1990)	Sector del saber pedagógico que se ocupa de la enseñanza	Practica de enseñar
Díaz (1998)	Ciencia imprescindible	Praxis del enseñante, cualificación de su actuación
Díaz Barriga (1998)	Disciplina teórica, histórica y política	Concepciones amplias de la educación, la sociedad y el sujeto
De Camilloni (2004)	Disciplina sobre la teoría de enseñanza	Acción pedagógica, campo social y del conocimiento
Litwin (2004)	Teoría acerca de las prácticas de enseñanza	Contextos socio-históricos
Grisales-Franco (2012)	Procesos de enseñanza para posibilitar aprendizajes	Lenguaje, comprensión, interpretación, síntesis.

Fuente: esta investigación

Con la tabla anterior, es posible sugerir los elementos fundamentales que componen una didáctica. Para ello, tras el rastreo relacionado con el concepto de didáctica, se han retomado elementos valiosos desarrollados como resultado de acontecimientos históricos, políticos y culturales. La Figura 1 es una representación gráfica sobre el concepto de didáctica que se asume dentro de esta investigación. Este concepto contempla la idea central de la enseñanza de todo a todos; entendiéndose por el espacio de interacción donde se enseñan los saberes de todas las profesiones y disciplinas hacia las personas. En este sentido, dicho concepto no se limita a alguna profesión, carrera o disciplina en particular, sino que, por el contrario, representa una postura general independientemente del saber.

Figura 1. Síntesis de conceptos asociados a la didáctica (Fuente: esta investigación)

Lo anterior sugiere que es posible hablar de didáctica como ciencia o arte de enseñar; donde se desarrollan teorías sobre prácticas de enseñanza. También es considerada como disciplina en estadios teóricos, históricos y políticos. De igual manera, como se trata de abordar prácticas de enseñanza de aula, el concepto de didáctica es asociado con propuestas de corte teórico-práctico en pro del mejoramiento del aprendizaje. Si bien es cierto que los conceptos de enseñanza y aprendizaje no están aislados el uno del

otro, esta investigación pretendió abordar el escenario principal de la enseñanza sin descuidar su incidencia en los aprendizajes de los estudiantes. El concepto de didáctica se abordó a partir de un constructo teórico que está fundamentado en principios pedagógicos para orientar las prácticas de enseñanza en un contexto determinado. Las visiones presentadas por los autores citados contemplan aspectos diversos para la construcción del concepto de didáctica como arte o como ciencia, la trascendencia del ejercicio de la enseñanza hacia ambientes externos al aula y su relación con el contexto social, entre otros.

2.3.2 Acerca de la transposición didáctica

La transposición didáctica es el fenómeno por el cual el saber profesional —conocimiento científico, académico, experto o erudito— se transforma en contenido factible de ser enseñado (Chevallard, 1985), no sólo mediante la simplificación o eliminación de algunas características difíciles o abstractas, sino más bien por un proceso más complejo que da forma a los conocimientos profesionales para adaptarse al contexto educativo, debido a las acciones conscientes e inconscientes de quién está a cargo del diseño de contenidos y de la enseñanza per sé.

El sistema didáctico es una tríada compuesta por el profesor (P), el estudiante (E) y el conocimiento (S) como se muestra en la Figura 2. Sin embargo, el conocimiento (S) presenta una serie de objetos que experimentan cambios. El conocimiento cambia; y lo hace frecuentemente cuando diferentes grupos de personas interactúan con él. Ese cambio se conoce como migración del conocimiento debido a la interacción de diferentes grupos de personas. En concordancia con la teoría de Chevallard, esta migración se llama transposición; implica necesariamente que el significado de la parte de conocimiento que migra cambiará ya que está vivo en diferentes grupos de personas. La transposición didáctica consiste en la migración del conocimiento en la comunidad de referencia, llamado saber experto, hacia el conocimiento que está vivo en el aula y se denomina saber enseñado (Tiberghien, Vince & Gaidioz, 2009, p. 6).

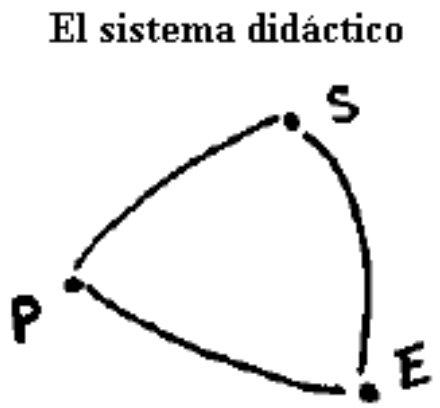


Figura 2. El sistema didáctico (Chevallard, 1985, p. 26)

De acuerdo con la teoría de la transposición didáctica, cuando el conocimiento experto —conocimiento erudito, profesional o saber sabio— se lleva hacia un entorno educativo, sus objetos experimentan transformaciones debido a la noosfera —el lugar donde los temas educativos se crean y se diseñan— y el ejercicio de la docencia en sí. La Figura 3 muestra el escenario en el que tal transformación del conocimiento se lleva a cabo; por lo tanto, un conjunto de sistemas didácticos está inmerso en el sistema de enseñanza para un contexto específico. El sistema de enseñanza pertenece a una institución educativa, donde la noosfera toma decisiones con respecto a los contenidos a enseñar. Además, todo está insertado en un ambiente con características específicas.

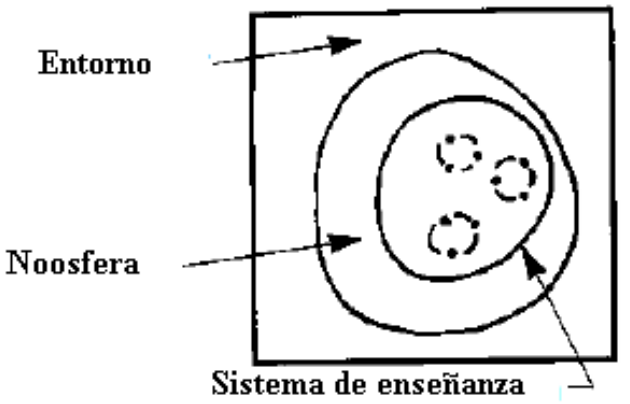


Figura 3. El sistema de enseñanza (Chevallard, 1985, p. 28)

En este sentido, un componente importante que forma parte del sistema de enseñanza y que es abordado en esta investigación es la noosfera. Para Chevallard, la noosfera representa ese escenario de interacción entre el sistema de enseñanza y la sociedad con sus exigencias particulares. En dicho escenario se presentan conflictos, se llevan a cabo negociaciones, y se supone que ahí se maduran las soluciones sobre la incidencia de la educación en la sociedad. En el espacio de la noosfera se proponen doctrinas, líneas completas de desarrollo educativo con un sentido político y social. El autor de la teoría de la transposición didáctica dice: "estamos en una esfera donde se piensa –según modalidades tal vez muy diferentes– el funcionamiento didáctico. Para esta instancia, sugerí el nombre paródico de noosfera" (Chevallard, 1985, p. 28).

La idea de que los seres humanos viven y comparten la noosfera como el espacio de pensamiento, tiene sus raíces en las ciencias cognitivas, de la misma forma en que la humanidad vive y comparte la biósfera, el entorno planetario natural. El concepto de noosfera es elaborado por uno de los cosmisistas rusos Vladimir Vernadsky en la década de 1920. Como científico natural, define la noosfera como la "esfera del pensamiento y las técnicas científicas manifestadas" y la reconoce como un "nuevo factor geológico sin precedentes en su poder". (Vernadsky, 2004, p. 16) Hoy en día, el término cubre toda la esfera de las actividades cognitivas y emocionales humanas.

En su particular explicación del lugar del pensamiento humano dentro de la naturaleza, Pierre Teilhard de Chardin (1955) postuló tal convergencia a través del surgimiento de capas sucesivas de "organización telúrica" desde el interior del planeta hacia la superficie del planeta, pasando por materia geológica inorgánica —concepto de geosfera—, a la complejidad orgánica de los seres vivos —concepto de biosfera–. En su lógica, de Chardin extiende el concepto hasta llegar a la trans-individualidad del pensamiento humano reflexivo. A esto, él le llamó noosfera.

Con estos referentes, se entiende la noosfera como el escenario de interacción de pensamiento donde se generan relaciones cognitivas. Los seres humanos crean ideas, conceptos, relaciones; en fin, producen pensamiento. La posición del ser humano frente al conocimiento hace que éste se transforme. Según la teoría de transposición didáctica, el conocimiento experimenta tres transformaciones a través de objetos claramente diferenciables: el objeto del conocimiento —el conocimiento experto en el estado

original—, el objeto de enseñar —que representa un contenido seleccionado para enseñar en los centros educativos y es generalmente propuesto por la noosfera— y el objeto de enseñanza —que representa el conocimiento en directa apropiación por parte de los estudiantes dentro de un salón de clases, es decir, el conocimiento puesto en práctica a través de la enseñanza–.

La teoría de trasposición didáctica propuesta por el profesor Chevallard ha sido ampliamente aplicada en la enseñanza de la matemática, tanto en el escenario europeo como en algunas partes de América Latina. Existe una amplia producción académica que da cuenta de su aplicación en la enseñanza de la matemática y de otras ciencias naturales; lo que contrasta con su baja aplicación en el campo de la enseñanza de la computación. Esta afirmación se puede corroborar en la revisión sistemática de las referencias usadas como fuentes en esta investigación, las cuales evidencian una producción académica incipiente cuando se trata de transposición didáctica aplicada a la computación.

Sobre la base de la teoría de Chevallard, se muestra una visión extendida del fenómeno con la adición de nuevos elementos, de acuerdo con el escenario de la educación en computación. En este orden de ideas, un nuevo enfoque sobre la teoría de transposición didáctica se propone a través de esta investigación, y se denomina: Transposición Didáctica In Extensa Sensu, que implica características propias, desde la perspectiva de la enseñanza de la Computación.

El concepto de transposición didáctica In Extensa Sensu propone una noosfera de interacción permanente en el acto educativo, además de la existencia de un objeto adicional que ayuda a extender el concepto original de transposición didáctica; además del objeto de conocimiento, del objeto de enseñar, y del objeto de enseñanza, la propuesta extendida incluye el objeto de evaluación junto con sus características particulares que ayuda a retroalimentar el sistema didáctico.

2.3.3 Discusión sobre el sistema didáctico y la noosfera

Si bien es cierto que la teoría de transposición didáctica ha sido usada en diferentes escenarios educativos, empezando por la enseñanza de la matemática y de las ciencias naturales, esta teoría no deja de causar inquietudes en la comunidad académica con fundamento en la pedagogía crítica. Una

interesante reflexión presentada por el profesor Cardelli en la Universidad Nacional de Río Cuarto en Argentina, muestra su inconformidad con las definiciones presentadas por Chevallard, haciendo una crítica hacia las limitaciones de la transposición didáctica. En su artículo, el profesor Cardelli menciona cómo en el contexto de la República Argentina, el concepto de transposición didáctica se queda corto en alcance, dado que la noosfera –y en especial la participación del profesorado– de la que habla Chevallard realmente no tiene incidencia en la estructuración de los saberes cuando existen grandes fuerzas hegemónicas que mantienen la ideología de un poder político y cultural (Cardelli, 2004).

Según Cardelli, el análisis de Chevallard no es lo suficientemente concreto porque deja afuera los elementos de hegemonía e ideología que son constitutivos del proceso mismo. La primera parte del proceso transpositivo, el que convierte un saber en saber a ser enseñado, se realiza sólo en determinadas coyunturas históricas, como fue el caso de los Contenidos Básicos Comunes implementados a partir de la Ley Federal en Argentina (Cardelli, 2004, p. 52).

Desde esta perspectiva, Cardelli afirma que el papel de la noosfera se orienta hacia el confinamiento de la acción de los educadores a la ejecución de lo que ya fue diseñado en escenarios de gobierno-estado. Con esto en mente, Cardelli establece una posición crítica frente a la posición hegemónica de un gobierno-estado con intereses muy particulares que quizás no comulguen con las expectativas de una población que quiere educarse. Ésta es la razón por la cual el profesor Cardelli discrepa de los conceptos de transposición didáctica por considerarla una teoría limitada que no tiene en cuenta aspectos socio-culturales, políticos e incluso económicos en un contexto determinado. En este orden de ideas, el sistema didáctico propuesto por Chevallard –el cual se compone del estudiante, el profesor y el saber– ha sido debatido fuertemente si se parte del hecho de la dificultad de atomizar sus componentes dada la complejidad del sujeto pedagógico, que a fin de cuentas es el protagonista en las dinámicas educativas (Puiggrós, 1990); por lo tanto, no se reduce al estudiante, al profesor y al saber cómo entes aislados con interacción en un escenario educativo sin contexto, sino que se aborda al sujeto pedagógico como el vínculo integrador entre los seres humanos que interactúan al interior de un currículo.

Por otra parte, hay autores que contrastan con el pensamiento que puede considerarse poco profundo del sistema didáctico de Chevallard, tal como lo

afirma Cardelli, y, por el contrario, le otorgan un valor que quizás no haya sido evidenciado en la práctica. Tal es el caso de una investigación realizada en Italia, que presenta un análisis en profundidad de lo que implica un saber institucionalizado junto con la relevante participación de la noosfera como organismo de vigilancia y promoción de saberes (D'Amore & Fandiño, 2001).

Para este estudio desarrollado en el ámbito del Programa de Investigación de la Universidad de Bolonia, la riqueza del triángulo que representa el sistema didáctico distingue aquí tres categorías que entran en juego: Los elementos que se pueden identificar con los "polos" del esquema precedente; las relaciones entre los elementos que se pueden identificar con los "lados"; y los procesos que identifican la modalidad de funcionamiento del sistema, tales como devolución, contrato didáctico, transposición, etc., los cuales están ligados al funcionamiento del sistema.

Al momento de poner en práctica esta composición particular del sistema didáctico de Chevallard, surge implícitamente la teoría de situaciones didácticas donde los componentes del triángulo cobran vida a través de sus interacciones conjuntas.

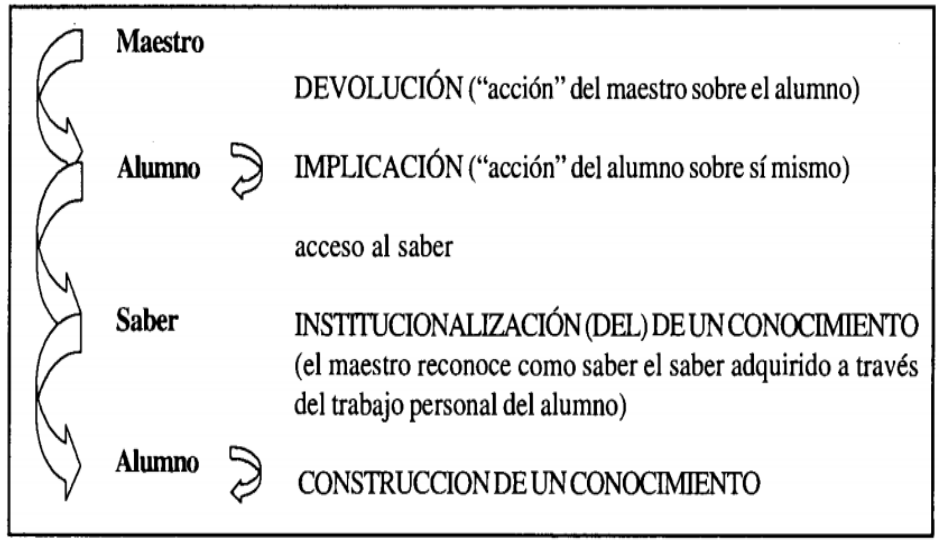


Figura 4. Teoría de situaciones didácticas aplicadas al sistema didáctico (D'Amore & Fandiño, 2001, p. 50)

Para la teoría de las situaciones didácticas en la enseñanza de la matemática, la visión sobre la enseñanza-aprendizaje es una construcción colabora-

tiva de una comunidad educativa que permite “comprender las interacciones sociales entre alumnos, docentes y saberes matemáticos que se dan en una clase y condicionan lo que los alumnos aprenden y cómo lo aprenden” (Brousseau, 1986). Es precisamente esa construcción colaborativa de la comunidad educativa que valida la importancia de la noosfera y su incidencia en los procesos de formación de corte constructivista.

Finalmente, Gómez (2005) hace una reflexión sobre los estudios realizados en la Facultad de Educación de la Universidad Tecnológica de Pereira en Colombia, donde resalta el valor de la noosfera como agente de cambio en la construcción de saberes en contextos específicos.

> *La noosfera se compone simultáneamente de representantes del sistema de enseñanza y de representantes de la sociedad: miembros de la asociación de docentes, profesores, padres de alumnos, especialistas de la disciplina que militan alrededor de su enseñanza, representantes de los organismos políticos. La noosfera es entonces ‘la esfera de gentes que piensan’ para retomar la expresión del autor [Chevallard]* (Gómez, 2005, p. 88).

En esta lógica, Gómez rescata la idea original de Chevallard donde el fin último es siempre la realización de un proyecto social. Es en este punto donde la noosfera recobra su valor y su importancia en las implicaciones sociales y políticas de un determinado contexto, donde el control social sobre el saber enseñado es fruto de innumerables interacciones que se suscitan al interior de las aulas y que pasaron por la visión de la noosfera.

Justo ahí, es posible develar el poder de la transposición didáctica para el emprendimiento de proyectos de liberación, de emancipación, de crítica hacia una sociedad que no se siente satisfecha con el papel de la academia según el desempeño de sus profesionales. Esta situación fácilmente puede considerarse como semilla germinal de una postura de pensamiento crítico-social, donde los procesos de reacomodación del tejido social sean factor importante en el proyecto de desarrollo alternativo de una sociedad.

Se considera que, la noosfera juega un papel esencial y es tomada como parte fundamental de las propuestas que se han producido al concluir la investigación. Es cierto que existen detractores de la teoría de transposición didáctica, y en especial del modelo de sistema didáctico propuesto

por Chevallard; no obstante, la importancia de configurar y determinar las acciones de una noosfera coherente al contexto ayudaría a resquebrajar las visiones reduccionistas que se tienen sobre el sistema didáctico de ese autor. Se pretendió con esta investigación develar esas bondades en el contexto real y particularmente sobre la educación en la computación.

Así, esta propuesta comulga con el pensamiento de humanización de la ciencia. Autores como Morín asumen una postura crítica frente a la fragmentación de saberes causado por un corazón deshumanizado de apropiación de las nuevas ciencias, donde no se las vuelve eternas ni universales, sino que son aisladas asumiendo una posición de soberanía en campos limitados. A esta situación, se le ha llamado “complejidad desarmada” (Morin, 2007). Complementario a este pensamiento, la presente investigación usa la transposición didáctica para la generación de procesos de libertad conceptual, orientada a la creación en aras de contrarrestar hegemonías de los saberes y situarlos en contexto. Es este el escenario preciso para la transposición didáctica que, sin perder rigurosidad en los saberes, gracias a la vigilancia epistemológica, ayuda a permear los lenguajes adaptables a las personas siguiendo los principios de democratización de saberes en entornos de complejidad donde se comulgue con los postulados de Gramsci, donde precisamente la comunidad educativa en el contexto es factor clave en el desarrollo de las sociedades (Ramírez, 2008).

2.3.4 La transposición didáctica en computación

Pero ¿Cómo se debe entender la transposición didáctica? Para ello, es importante citar a su autor: Yves Chevallard (1985) La transposición didáctica es el fenómeno por el cual el contenido profesional —científico, sabio, experto o erudito— se convierte en contenido factible de ser enseñado.

A partir de la formulación de la teoría de transposición didáctica a mediados de la década de los ochentas del siglo pasado, se ha disparado una serie de investigaciones y estudios desde la adopción de la teoría en escenarios académicos. Tal es el caso de la formación en matemáticas y lenguaje, donde la producción de conocimiento en materia de la enseñanza de dichos saberes desde una mirada a través de la transposición didáctica ha sido prolífica. En contraste a esta abundante producción de conocimiento, los estudios de transposición didáctica en la enseñanza de la computación se encuentran en un estado primigenio.

A pesar de sus valiosos aportes al campo de la didáctica, el grupo ACM SIGCSE —Special Interest Group in Computer Science Education— solamente tiene documentada hasta la fecha una sola investigación relacionada con el estudio de la transposición didáctica en computación, realizada porHazzan, Meerbaum-Salant y Dubinsky (2010), pero este trabajo no profundiza en el área específica de los fundamentos de programación de computadoras. Sus autores realizaron el análisis del fenómeno de transposición didáctica en materia del diseño de software; sin embargo, sus estudios no contemplan el proceso de enseñanza de los objetos de saber propios de los fundamentos de programación por tratarse de una fase posterior al diseño. La investigación de Hazzan, Meerbaum-Salant y Dubinsky se desarrolló en Technion, el Instituto Tecnológico de Israel con estudiantes de pregrado.

Ahora bien, partiendo de la transposición didáctica, el trabajo pionero por su naturaleza de estudio que se encuentra en el repositorio de investigaciones de ACM SIGCSE se titula “Didactic Transposition in Computer Science Education”. En este artículo, los autores introducen dicho concepto dentro del contexto de la enseñanza en computación (Hazzan, Dubinski & Meerbaun-Salant, 2010). A pesar de que en dicho trabajo se aborda la transposición didáctica, su aplicación se realiza en forma general dado que involucra el tema de métodos de construcción de software, cuyo campo de conocimiento es bastante amplio. Por otra parte, este estudio se basa en las experiencias en el nivel de educación básica secundaria —high school— y pregrado universitario —undergraduate— sin tocar los temas específicos propios de la programación de computadoras.

De esta manera, el conocimiento experto —científico— por parte de la industria del software es transformado a fin de producir objetos de aprendizaje para ser aplicados en la academia. Los resultados de la investigación son interesantes en el sentido de dar pautas para mejorar aspectos en la práctica de la docencia por parte de los profesores al incorporar elementos que no se tuvieron en cuenta en pasadas ocasiones. Tomando la directriz del fenómeno: “La enseñanza realmente no puede ser separada de aprendizaje” (Chevallard, 1988), los hallazgos de dicha investigación permiten articular actividades en el aula que potencian el aprendizaje de los estudiantes en la medida en que se recuperan elementos “perdidos” en la transformación del saber experto en el saber enseñado.

Con esto, se tiene una gran oportunidad de investigación en un campo que, para la educación en computación, ha sido explorado en forma incipiente. Los vacíos investigativos en materia del estudio de la transposición didáctica en la enseñanza de la computación obligan a la construcción de propuestas de enseñanza que permitan potenciar los aprendizajes de los objetos de saber que experimentan dicho fenómeno expresado en la teoría.

2.3.5 Las disciplinas asociadas a la computación

ACM e IEEE *Computer Society* tienen una larga historia apoyando esfuerzos para establecer lineamientos curriculares globales para programas de pregrado en computación. Dichas asociaciones tradicionalmente publican sus lineamientos en períodos aproximados de 10 años, iniciando con la publicación de *Curriculum* 68 hace más de 4 décadas. Como el campo de la computación ha crecido y se ha diversificado, también lo han hecho las recomendaciones para los planes de estudio promulgando una división a través del concepto de disciplina, así que ahora hay recomendaciones para las disciplinas: Ingeniería en Computación, Sistemas de Información, Tecnología de Información, Ingeniería de *Software* y Ciencia de la Computación.

Para *ACM* e IEEE *Computer Society*, las disciplinas son las más grandes divisiones epistemológicas en materia de conocimiento asociado a la computación. Cada una de ellas tiene un quehacer particular; se presenta a continuación la forma en que estas cinco disciplinas fueron definidas (ACM & IEEE *Computer Society*, 2013):

Ingeniería en Computación (*Computer Engineering - CE*). Esta disciplina se refiere al diseño y construcción de equipos y sistemas basados en computadoras. Implica el estudio de *hardware*, *software*, comunicaciones, y la interacción entre ellos. Su plan de estudios se centra en las teorías, principios y prácticas de la ingeniería y las matemáticas de la ingeniería eléctrica tradicional aplicándolas a los problemas de diseño de computadoras y dispositivos basados en computadoras.

Ciencia de la Computación (*Computer Science - CS*). Esta disciplina abarca una amplia gama de saberes, desde sus fundamentos teóricos y algorítmicos para desarrollos de vanguardia en la robótica, visión por computador,

sistemas inteligentes, bioinformática, y otras áreas. Se puede pensar en el trabajo de los científicos de la computación para el diseño e implementación de *software*, la visualización de nuevas formas de usar las computadoras, y el desarrollo de formas eficientes de resolver problemas mediante computadoras.

Sistemas de Información (*Information Systems - IS*). Esta disciplina se centra en la integración de soluciones de tecnología de la información y procesos de negocio para satisfacer las necesidades de información de las empresas y otras organizaciones, lo que les permite alcanzar sus objetivos de una manera eficaz y eficiente. La perspectiva de esta disciplina en la tecnología de la información hace hincapié en la informática, y considera a la tecnología como un instrumento para la captura, procesamiento y distribución de información.

Tecnología de Información (*Information Technology - IT*). Esta disciplina se refiere a los programas de pregrado que preparan a los estudiantes para satisfacer las necesidades de tecnología informática de negocios, el gobierno, la salud, las escuelas y otros tipos de organizaciones. En algunas naciones, otros nombres se utilizan para este tipo de programas de grado; por ejemplo, en España se habla de Ingeniería Informática mientras en la mayor parte de América Latina fue denominada la disciplina como Ingeniería de Sistemas.

Ingeniería de *Software* (*Software Engineering - SE*). Esta disciplina se enfoca en el desarrollo y mantenimiento de sistemas de *software* que se comportan de forma fiable y eficiente, para que sean asequibles de desarrollar y mantener, a fin de satisfacer todas las necesidades que los clientes han definido para ellos. La Ingeniería de *software* es diferente en carácter a las otras disciplinas debido a la naturaleza intangible del *software* y a su naturaleza discontinua de operación. Se trata de integrar los principios de las matemáticas y la computación con las prácticas de ingeniería desarrolladas para artefactos tangibles y físicos.

Al analizar esta clasificación del conocimiento asociado a la computación, hay un factor aglutinante que está presente y de forma transversal a todas estas disciplinas y es la programación de computadoras. Como evidencia de esta afirmación se muestran las siguientes gráficas expuestas en la Figura 5:

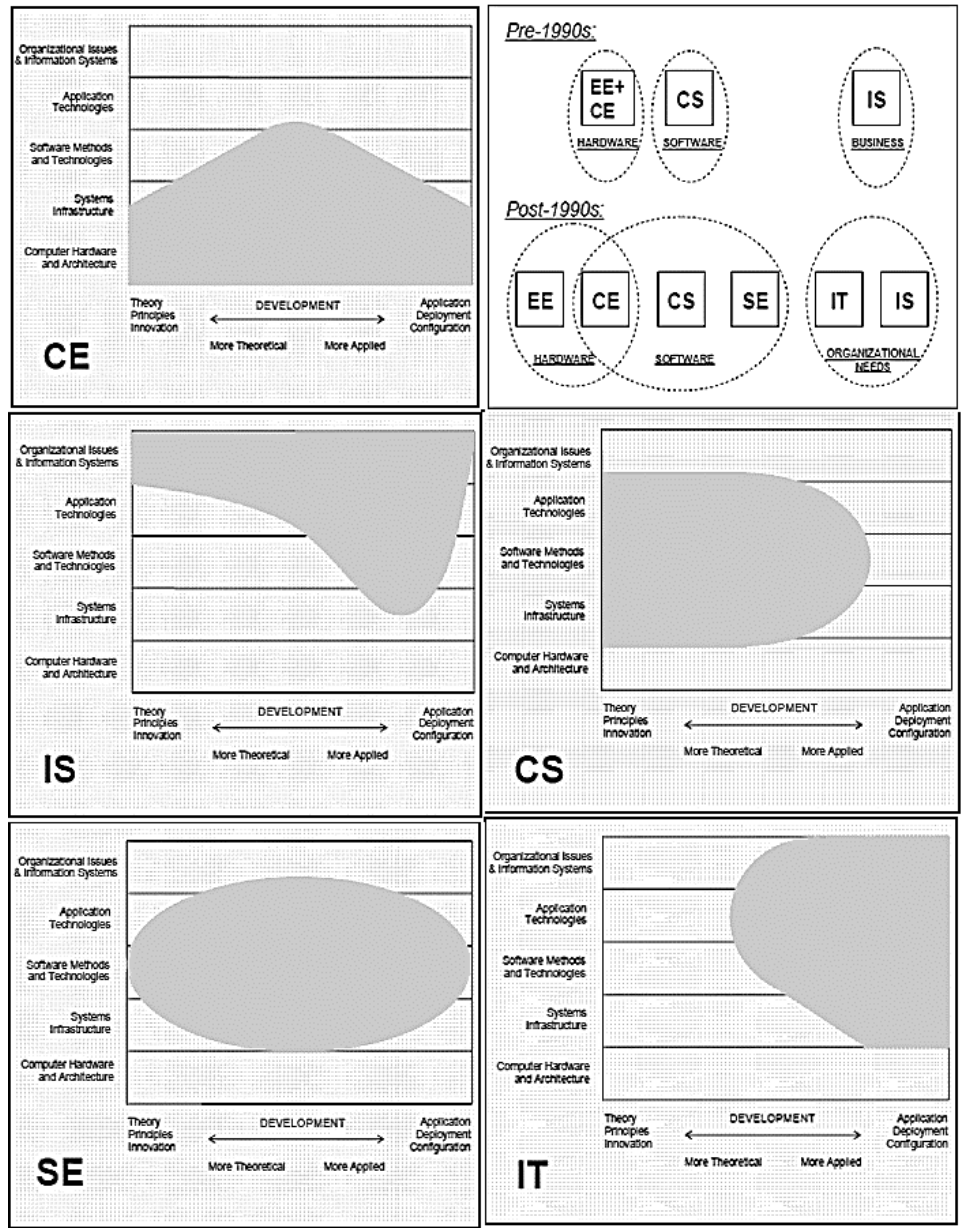


Figura 5. Disciplinas asociadas a la computación (Adaptado de ACM, AIS, & IEEE Computer Society, 2005)

En la Figura 5 se puede apreciar los campos de acción de las disciplinas asociadas a la computación, a saber: CE, IS, CS, SE, y IT (imagen compuesta a partir de ACM, AIS & IEEE Computer Society, 2005). Nótese que el campo de conocimiento denominado *Software Methods and Technologies* (Métodos de *Software* y Tecnologías) —el cual se resalta en rectángulos como se

observa en la Figura 5— es común a todas las disciplinas. Es justo en dicho campo de conocimiento donde los fundamentos de programación de computadoras se constituyen en el curso inicial, como la base para la construcción de *software*.

Las recomendaciones curriculares para la construcción del *Syllabi* (plan de estudios) de las disciplinas asociadas a la computación, a saber: ingeniería en computación, ciencia de la computación, ingeniería de *software*, sistemas de información y tecnología de información; tienen un común denominador en el área de los cursos introductorios ubicado en *Software Methods and Technologies* (rectángulos rojos en la Figura 5).

El área de los cursos introductorios se presenta a través de la existencia de diversos enfoques adecuados para cursos de inducción a las cinco disciplinas asociadas a la computación. Muchos de esos enfoques se centran en los temas en fundamentos de programación de computadoras junto con un subconjunto de los temas en los lenguajes de programación o ingeniería de *software*, dejando la mayor parte de los temas en estas otras áreas de conocimiento que se presentarán en cursos avanzados (ACM & IEEE *Computer Society*, 2013).

Entonces, se entiende que fundamentos de programación de computadoras dentro de los planes de estudio en las disciplinas asociadas a la computación que se han basado en las recomendaciones curriculares de ACM son ubicados en los dos primeros semestres de formación profesional de pregrado; y de acuerdo con su contenido, dichos cursos se constituyen como el prerrequisito de los cursos avanzados de programación y de construcción de *software* profesional.

En el contexto colombiano, las pruebas de estado profesional tienen un componente específico de evaluación para el área de construcción de software, el cual se aplica a las ingenierías: Ingeniería de Sistemas, Ingeniería de Software, Ingeniería de Sistemas y Computación, Ingeniería Informática, Ingeniería de Sistemas e Informática, e Ingeniería de Sistemas Informáticos. La guía de orientación de presentación del módulo de Diseño de Software para las pruebas Saber PRO 2015-2 involucra implícitamente los fundamentos de programación de computadoras a través de las áreas conceptuales de referencia.

Para abordar el módulo de diseño de software es necesario plantear problemas desde el punto de vista sistémico; conocer, entender y aplicar la teoría general de sistemas en cada una de las etapas del ciclo de vida de un sistema de información; comprender conceptos básicos de estructuras de datos y las primitivas de programación existentes, así como las bases de programación orientada a objetos, uso de lenguaje de modelado, diseño de interfaces gráficas, la teoría general de bases de datos y teoría general de sistemas, todo esto para la solución de problemas mediante algoritmos (ICFES, 2015). Por otra parte, la asociación colombiana de facultades de ingeniería, a través de sus recomendaciones curriculares para el capítulo de Ingeniería de Sistemas, ubica los cursos en mención como introductorios en la formación de profesionales en ingeniería, dado que sus contenidos son la base fundamental para la construcción de soluciones de software (ACOFI, 2005).

En resumen, cada país dispone de programas académicos cuya base epistemológica en el conocimiento de la computación parte de las definiciones de sus disciplinas asociadas. Por citar un ejemplo, es común en los Estados Unidos y en algunas partes de Europa oír hablar de Computer Science como el programa académico equivalente a la Ingeniería de Sistemas de nuestro país, y para el caso particular de España, el programa académico en Computer Science, o en Ingeniería de Sistemas, toma equivalencia como Ingeniería Informática. El asunto es que existen denominaciones diferentes para los programas académicos que han sido formulados a partir de los lineamientos de ACM e IEEE Computer Society. Lo cierto es que independientemente de su denominación, los fundamentos de programación son un curso introductorio en la mayoría de los casos.

2.3.6 Los fundamentos de programación de computadoras

Desde el punto de vista conceptual, el curso de fundamentos de programación de computadoras define la base de la programación estructurada e incluso la orientación a objetos, se fundamenta principalmente en la lógica algorítmica que le permite al programador construir una solución al problema planteado. Dentro de este curso, el estudiante construye soluciones algorítmicas a problemas de origen aritmético y algebraico, utiliza herramientas a manera de nomenclaturas especiales como *Flowchar*t, *Nassi-Schneidermann*, o *P-Code*, entre otros. En escenarios prácticos, los algoritmos planteados suelen ser implementados en lenguajes de programación, bien

sea de uso didáctico como *Alice*, o de uso profesional *como C/C++, C#, Java, Python, Matlab, Haskell*, por citar algunos de ellos.

El objetivo que persigue el curso de fundamentos de programación de computadoras es brindar las bases para la construcción profesional de *software* que tengan su fundamento en situaciones factibles de resolverse por medios computacionales. Es así como la competencia de programar computadoras es la más desarrollada dentro de estos cánones.

Por tratarse de un curso que se suele impartir en los primeros semestres por su carácter de fundamentación, se requiere de una base teórica en matemáticas, al menos para el uso de expresiones aritmético-lógicas. En este orden de ideas, la formación previa en bachillerato —o en *high school*, según el contexto norteamericano— es suficiente para enfrentar los retos que los fundamentos de programación de computadoras puedan plantear.

La fundamentación matemática también puede verse reflejada en el contexto colombiano a través de las pruebas de estado SABER PRO para Ingeniería de Sistemas. En los contenidos referenciales de dicha prueba, se puede evidenciar que un gran componente se orienta hacia la Ingeniería Aplicada, donde la programación y la algoritmia forman parte importante de dicha prueba. De un total de 180 preguntas, prácticamente 60 preguntas son orientadas a partir de la resolución de problemas mediante la aplicación de las ciencias naturales y las matemáticas utilizando un lenguaje lógico y simbólico, así como la utilización de la teoría, la práctica y las herramientas apropiadas para la solución de problemas de programación. Competencias que se generan a parir de una buena fundamentación en programación de computadoras (ICFES, 2010).

3. EL DESARROLLO METODOLÓGICO

Toda investigación requiere de un diseño metodológico que guíe su desarrollo. La línea de formación titulada “La Educación en América Latina desde su perspectiva Histórica, Pedagógica y Curricular” define las pautas para la delimitación de las investigaciones. Así, esta línea de formación promueve el desarrollo de investigaciones para las Ciencias de la Educación de corte cualitativo, priorizando los enfoques históricos, crítico-social y hermenéutico. En este sentido, la presente investigación se enmarca en el paradigma cualitativo con un enfoque hermenéutico.

Por tratarse de una investigación cualitativa, la identificación y el análisis de categorías son necesarias como parte del proceso investigativo. Estas categorías han sido formuladas de tal suerte que comulguen con los enunciados de los objetivos específicos de la investigación, que a su vez fueron producto derivado de las preguntas orientadoras de la investigación. De igual forma, las unidades de análisis y las unidades de trabajo fueron concebidas para la aplicación de los instrumentos de recolección de información.

Al final de este capítulo se presenta la tabla llamada matriz metodológica, la cual integra las preguntas orientadoras, los objetivos, las categorías, las técnicas de recolección de la información y las técnicas de análisis y procesamiento de esta. Dicha matriz orientó el desarrollo de la investigación.
Al tenor de las ciencias de la educación y la pedagogía, la posición que asume esta investigación se fundamenta en el concepto de didáctica. Teniendo en cuenta esta postura, se da respuesta a los siguientes interrogantes:

¿Qué se investiga?

El objeto de estudio de la investigación es la enseñanza de los fundamentos de programación de computadoras y el uso de la transposición didáctica en dicho escenario. Al tener una complejidad manifiesta en los contenidos de dicho tema, se opta por seleccionar un conjunto de objetos de saber que forman parte de su *corpus* de conocimiento. La investigación centra su

atención en dicho conjunto de objetos de saber desde una perspectiva de transposición didáctica.

¿Cómo se investiga?

La educación conforma el gran estadio donde la investigación se planea, se desarrolla y se concluye. Con esto, la forma cómo se investiga hereda las características propias del paradigma cualitativo, que es el paradigma predominante en la investigación propia de las ciencias sociales, ciencias humanas o de las ciencias del espíritu. Teniendo en cuenta las orientaciones fundamentadas por el paradigma cualitativo, el enfoque de esta investigación es hermenéutico; así, la investigación se construye a través de la interpretación y la argumentación en contexto de los estudios realizados, donde la cualidad de los acontecimientos cobra importancia para la aprehensión de lo que sucede al interior del aula de clase, explorando más allá de la descripción para lograr el entendimiento en profundidad de dichos acontecimientos.

¿Cuándo y dónde se investiga?

La investigación se nutre a través de las experiencias de enseñanza de profesores nacionales y extranjeros; ergo, el escenario mundial da respuesta a la pregunta acerca de dónde se investiga. Es precisamente la interacción en el aula de clase el momento cuando se hace explícito la labor de enseñanza. De esta forma, los profesores involucrados en la investigación brindan sus reflexiones sobre dichos momentos de enseñanza en el escenario real. Así se da respuesta al interrogante acerca de cuándo se investiga.

¿Quiénes participan en la investigación?

Se ha mencionado que profesores nacionales y extranjeros participaron en esta investigación. Cabe señalar que los instrumentos de recolección de información se diseñaron para lograr una participación masiva, teniendo en cuenta la definición de las unidades de análisis y las unidades trabajo definidas en la metodología; sin embargo, la muestra de profesores que participaron en la investigación respondió a su interés personal por participar.

¿Para qué se investiga?

El fin último de la investigación es enriquecer experiencias de enseñanza en los fundamentos de programación de computadoras a través de la transposición didáctica. Se sostiene el postulado de brindar aportes al conocimiento en materia de enseñanza de los fundamentos de programación de computadoras a través del uso de la transposición didáctica.

3.1 Racionalidad en la investigación

Paradigma de Investigación: cualitativo

Las palabras cualidad y calidad comparten su origen derivado de la locución latina qualitas, y ésta a su vez se deriva de qualis (cuál, qué). De modo que, a la pregunta por la naturaleza o esencia de un ser: ¿qué es?, ¿cómo es?, se dan las respuestas señalando o describiendo el conjunto de cualidades. Esto constituye el principio rector del paradigma cualitativo, donde importa la cualidad del objeto de estudio, sus manifestaciones en el mundo y su incidencia en un entorno social. De esta manera, la investigación cualitativa trata de identificar la naturaleza profunda de las realidades, su estructura dinámica, aquella que da razón plena de su comportamiento y manifestaciones. Es por eso que lo cualitativo (que es el todo integrado) no se opone a lo cuantitativo (que es sólo un aspecto de medición), sino que lo complementa, especialmente donde sea importante en el marco del diálogo de intersubjetividades (Campos, 2009).

Y es precisamente el concepto de integración y complementariedad lo que beneficia un estudio desde un enfoque sistémico, asumiendo las realidades en un estadio de complejidad y dependencia mutua con otros sistemas. Desde esta perspectiva, se considera que toda realidad, desde el átomo hasta la galaxia, está configurada por sistemas de muy alto nivel de complejidad, donde cada parte interactúa con todas las demás y con “el todo” (von Bertalanffy, 1976, p. 47). Otra razón más para considerar aspectos cualitativos de integración y complejidad, más allá de una explicación estrictamente experimental.

Dado que esta investigación se fundamenta en el enriquecimiento de la enseñanza de los fundamentos de programación de computadoras a partir de la transposición didáctica, es importante destacar la indagación sobre un fenómeno presente en las ciencias de la educación, más allá de la demostración o

la comprobación. En esta investigación se profundiza sobre experiencias de aula en diferentes contextos, donde las vivencias y los aprendizajes subyacen en el tejido de interacción social que dichos escenarios plantean.

La cualidad es lo relevante para esta investigación, más allá de la medición. En este sentido, se abordan experiencias humanas interpretadas como "experiencias de verdad" donde la metodología científica tradicional propia del enfoque cuantitativo positivista es incapaz de verificar dicha verdad en la riqueza profunda basadas en experiencias de vida humana (Gadamer, 1984).

Así, la investigación cualitativa está diseñada para revelar el rango de comportamiento del público objetivo y las percepciones que lo impulsan con referencia a temas específicos. Utiliza estudios en profundidad de grupos de personas para guiar y apoyar la construcción de hipótesis. Los resultados de la investigación cualitativa son descriptivos y no predictivos.

Los métodos de investigación cualitativa vieron su origen en las ciencias sociales y del comportamiento, tales como sociología, antropología y psicología. Hoy en día, los métodos cualitativos en el campo de la investigación incluyen encuestas con preguntas abiertas —a manera de cuestionarios—, entrevistas en profundidad con individuos, discusiones de grupo, etnografía, grupos focales, análisis hermenéutico, entre otros.

La investigación cualitativa es principalmente una investigación exploratoria tal como lo afirma Muñoz (2011). Se utiliza para obtener una comprensión de las razones subyacentes, las opiniones y las motivaciones, esto es: la cualidad. La investigación cualitativa también se utiliza para descubrir tendencias en pensamiento y opiniones, y profundizar en el problema.

Los datos cualitativos, generalmente, en forma de palabras a partir de observaciones, entrevistas o documentos, han sido la principal fuente de datos para las disciplinas de las ciencias sociales, humanas o del espíritu a lo largo de su historia (Miles & Huberman, 1994). Los investigadores cualitativos estudian las cosas en su entorno natural, tratando de dar sentido a los fenómenos dentro del contexto social o natural. En las ciencias sociales, humanas y del espíritu en particular, la investigación cualitativa se utiliza para explorar cualquier significado más profundo atribuido por los sujetos al tema bajo examen.

En este orden de ideas, esta investigación realiza una observación documental de personas en su labor —los profesores y sus prácticas de enseñanza—; interactúa con ellos sobre lo que saben, creen o sienten; y examina los artefactos que producen en su ejercicio de enseñanza. Estas son fuentes de información de tipo cualitativo con la cual la investigación se nutre.

Mientras que es común para los científicos naturales analizar sus datos usando números —p. ej., frecuencias de ocurrencia en una población—, los investigadores cualitativos consideran que el análisis numérico es limitado al momento de comprender en profundidad al objeto de estudio —que seguramente es un sujeto–: personas conscientes, agentes con ideas sobre su mundo y su sentir frente a él. En esta investigación, los datos son descripciones ricas, profundas y complejas sobre las experiencias en enseñanza de sujetos conscientes, en cuyas palabras se puede ver reflejado el sentir, las percepciones, las frustraciones y los anhelos con respecto al ejercicio de su labor; características que difícilmente pueden cuantificarse.

Enfoque de Investigación: hermenéutico

Históricamente, el término "hermenéutica" fue entendido como el arte de la interpretación de textos, es decir, la teoría de lograr una comprensión de documentos. La hermenéutica en este sentido tiene una larga historia, que se remonta al menos hasta la antigua Grecia. Sin embargo, en el período moderno, La Reforma asumió la responsabilidad de interpretar la Biblia de la Iglesia Católica para los cristianos en general, darle un nuevo enfoque. Esto se produjo especialmente en Alemania. Luego a principios del Siglo XIX los los primeros aportes de Friedrich Daniel Ernst Schleiermacherayudaron a la formalización de la hermenéutica. Mientras que en el siglo XX gracias a Martin Heidegger y Hans-Georg Gadamer floreció la hermenéutica desde una perspectiva filosófica.

La sistematización de la hermenéutica general propuesta por Schleiermacher, y que se describe en la edición de Bowie (1998), define el ejercicio hermenéutico como el arte de comprender, siendo ésta la base para teorías y metodologías para la interpretación de textos. De esta forma, se trata de interprerar un texto con el propósito de encontrarle su sentido en profundo. Para Gadamer, el ejercicio hermenéutico se trata de la integración del progreso científico y del pensamiento a partir de una hermenéutica filosófica, donde el arte de la interpretación apunta hacia el entendimiento bajo

el carácter fundamentalmente móvil de la existencia; de igual manera y dado el carácter específico y finito del ser humano, la experiencia se constituye como la pretensión del conocimiento objetivo. Es justo aquí donde el ejercicio hermenéutico tiene como principio supremo dejar abierto el diálogo orientado a la comprensión (Gadamer, 2008).

En este contexto, Gadamer relacionó su proyecto con la filosofía trascendental de Kant. Así como este filósofo intentó describir la posibilidad trascendental de las ciencias naturales, Gadamer buscó revelar la "posibilidad" de las ciencias humanas. De esta manera, dejó en claro que es de suma importancia para su proyecto explicar que la hermenéutica filosófica necesita una dimensión trascendental para no disolverse en un método de interpretación que compita con un número indefinido de otros métodos. En otro orden de ideas, Vattimo (1997) señala que, la hermenéutica no puede considerarse meramente como una teoría sobre la historicidad de la verdad; también debe considerarse como una verdad histórica radical.

En este sentido, esta investigación trata sobre la comprensión de situaciones propias de la enseñanza de los fundamentos de programación vividas al interior del aula de clase. Entonces, para comprender en profundidad un fenómeno es necesario recordar a Ricœur (1970, p. 33) cuando escribe: "La comprensión es hermenéutica: de ahora en adelante, buscar significado ya no se trata de deletrear la conciencia del significado, sino se trata de descifrar sus expresiones".

Por tal razón, esta investigación se valió de la hermenéutica como enfoque investigativo, justo cuando la información obtenida a partir de las experiencias de vida al interior del aula de clase debe ser interpretada en profundidad, a fin de lograr una comprensión de dichas experiencias que son producto de la labor de la enseñanza de los fundamentos de programación. Por tratarse de contextos diferentes, dada la participación de los profesores nacionales y extranjeros, el enfoque hermenéutico otorga validez metodológica, en el sentido de reconstruir los escenarios de enseñanza dando lectura a la descripción de cada experiencia, teniendo en cuenta la interacción con los estudiantes y sus contextos particulares.

Esta investigación parte de la visión gadameriana sobre la hermenéutica. Al respecto Habermas (1984) no acepta que la sociedad se comprenda y dialogue auténticamente desde la tradición, si ésta trae consigo toda su carga

de prejuicios, que puede traducirse en la fuente misma del error y ser el obstáculo principal para una auténtica comunicación.

Frente a la posición de Habermas, cabe resaltar que el interés de esta investigación es enriquecer experiencias de enseñanza sobre los fundamentos de programación de computadoras a través de la transposición didáctica. Con tal interés, la investigación parte de la tradición —es decir, de la forma tradicional de enseñar los fundamentos de programación— dándole su valor; sin embargo, el enfoque hermenéutico aplicado rescata el principio transformador habermasiano que pretende brindar alternativas que susciten tal enriquecimiento de experiencias a través de ejercicios de comunicación.

A partir del estado del arte y de los hallazgos como producto de la aplicación de los instrumentos de recolección de información, la investigación contempla la realización de un ejercicio de interpretación y valoración de dichos resultados. Así pues, se prepara, de esta forma, el terreno para la realización de un ejercicio hermenéutico donde las intersubjetividades entran a elaborar un complejo escenario de análisis.

3.2 Insumos y análisis de información

Es importante aclarar que la forma en que se recolectó la información de la investigación fue soportada por la plataforma tecnológica propia para estos fines, construida y desplegada por el autor de la investigación vía Web. Los recursos tecnológicos que fueron utilizados pertenecen al Grupo de Investigación GALERAS.NET adscrito al Departamento de Sistemas de la Universidad de Nariño.

Desde el punto de vista instrumental, se usan como técnicas de recolección de información la revisión documental guiada a través de fichas diseñadas para este propósito. De igual forma, fue complementada la recolección de información a través de encuestas puntuales y entrevistas estructuradas a expertos.

El análisis hermenéutico se apoya en la construcción de esquemas preconceptuales donde se establece relaciones a partir de la revisión documental. El uso de dichos esquemas parte de la iniciativa de establecer como punto de referencia un análisis de enunciados a partir de técnicas de lingüística computacional.

El uso de tales esquemas tiene como finalidad representar gráficamente los constructos teóricos propios de la labor de investigación en forma de ontologías computacionales. Según su creador, un esquema preconceptual es una forma de representar el conocimiento mediante el uso de un lenguaje controlado (Zapata, 2007). Además, los esquemas preconceptuales utilizan la notación simple, son fáciles de entender y son adaptables a cualquier dominio del conocimiento (Zapata, Arango & Gelbukh, 2006). Adicionalmente, se destaca el uso de los esquemas en mención por su facilidad de representar relaciones ontológicas, de tal suerte que son aprovechadas en las definiciones conceptuales de la investigación.

La revisión documental es una técnica de recolección de información, basada en la consulta sistemática de fuentes documentales para soportar procesos de investigación. En este sentido, las principales fuentes de revisión documental son: las recomendaciones curriculares propuestas por ACM, AIS e IEEE, los *syllabus* de fundamentos de programación que forman parte del grupo profesoral que participó en la investigación, el conjunto de libros de texto guía para fundamentos de programación, el *dossier* de respuestas dadas por la entrevista y las encuestas, y las respuestas del formato de vigilancia epistemológica.

La encuesta recrea un escenario de preguntas y respuestas hacia una población objetivo, con el propósito de abordar la opinión individual y colectiva sobre una temática específica. La encuesta suele utilizarse en diferentes escenarios de investigación, con grupos pequeños, con grupos grandes, con individuos geográficamente dispersos, o con grupos concentrados en un lugar. En este punto, fueron enviadas a través de correo electrónico un llamado a la acción para los profesores que deseaban participar en la investigación, donde previamente se contaba con la lista de contactos de los profesores a cargo de fundamentos de programación de 50 universidades en el mundo y de 84 instituciones de educación en Colombia. Cuando el grupo de profesores extranjeros y nacionales manifestaron su intención de participar en la investigación, se les envió la encuesta y el formato de vigilancia epistemológica para su desarrollo. Tanto la encuesta como el formato de vigilancia epistemológica son formularios que se diligencian a través de un servicio *Web*. Por su parte, la guía de vigilancia epistemológica ayuda a sistematizar el proceso de observación como acción examinadora para determinar la distancia entre el saber erudito y la interpretación dada en aula. El ejercicio debe ser constante y tendiente al uso de la preparación académica de contenidos y formas de ser impartidos

en el contexto real (Contreras, 2013). La entrevista semiestructurada hace uso de un cuestionario que podría ser previamente acordado con el entrevistado, pero se caracteriza principalmente por generar espacios de libertad para la profundización, extensión y explicación adicional de acuerdo con la dinámica de la entrevista. Este tipo de entrevista es muy relevante para adquirir información de orden cualitativo, para dejar abierta la puerta a un ejercicio de análisis hermenéutico.

De acuerdo con el enfoque de la investigación, la Tabla 2 muestra la distribución de los objetivos específicos junto con sus preguntas orientadoras las cuales se asociaron a la consecución de las categorías. De igual manera, en dicha tabla se presenta las técnicas de recolección de información y las técnicas de procesamiento de esta para el cumplimiento de los objetivos.

Tabla 2. Matriz metodológica

Objetivo Específico	Pregunta Orientadora	Categoría	Técnica de Recolección	Técnica de Procesamiento
Caracterizar los objetos seleccionados de saber de los fundamentos de programación de computadoras, de acuerdo con los referentes internacionales.	¿Cuáles son las características de los objetos seleccionados de saber de los fundamentos de programación, de acuerdo con los referentes ternacionales?	C1. Características de los objetos de saber seleccionados	Revisión documental Encuesta a profesores nacionales y extranjeros	Elaboración de esquemas preconceptuales Análisis hermenéutico
Analizar los niveles de transposición didáctica de los objetos seleccionados de saber de los fundamentos de programación de computadoras.	¿Cuáles son los niveles de transposición didáctica de los objetos seleccionados de saber de los fundamentos de programación de computadoras?	C2. Niveles de transposición didáctica de los objetos seleccionados	Guía para la vigilancia epistemológica a profesores nacionales y extranjeros	Elaboración de esquemas preconceptuales Análisis hermenéutico
Proponer prácticas de enseñanza para los objetos seleccionados de saber transpuestos de los fundamentos de programación de computadoras.	¿Cuáles son las prácticas de enseñanza de los objetos seleccionados de saber transpuestos de los fundamentos de programación de computadoras?	C3. Prácticas de enseñanza de objetos seleccionados	Entrevista semiestructurada a expertos nacionales y extranjeros	Elaboración de esquemas preconceptuales Análisis hermenéutico

Fuente: esta investigación

3.3 Unidades de análisis y unidades de trabajo

De acuerdo con las características de la investigación, las unidades de análisis son finitas e intencionadas, así que sus unidades de trabajo con respecto a los profesores en otros países son de tipo no probabilístico con sujetos voluntarios. Así, la Tabla 3 muestra la forma de abordar el estudio:

Tabla 3. Técnicas de recolección y procesamiento de información

Respecto a...	UNIDAD DE ANÁLISIS	UNIDAD DE TRABAJO	CRITERIO DE SELECCIÓN
Saberes propios de los fundamentos de programación de computadoras	*Knowledge Area*: *SDF - Software Development Fundamentals*	*Programming Fundamentals Concepts.*	El referente mundial de mayor uso
Profesores y expertos extranjeros	Expertos y profesores a cargo de cursos de fundamentos de programación de computadoras en programas de pregrado de universidades públicas y privadas en el exterior.	1 experto[3] y una selección de profesores que trabajen en universidades públicas o privadas, con producción académica relacionada con el objeto de estudio de la presente investigación.	*profesores del Top 50 según clasificación mundial de universidades en el campo de Ingeniería y Ciencia Aplicada.* (Quienes respondan la encuesta y desarrollen la guía de vigilancia epistemológica)
Profesores y expertos nacionales	Profesores de cursos de fundamentos de programación de computadoras de los programas de pregrado de universidades públicas y privadas en Colombia.	1 experto[4] y 4 profesores de fundamentos de programación de computadoras, con producción académica relacionada con el objeto de estudio de la presente investigación	Profesores de las universidades que obtuvieron un resultado por encima de la media nacional en Saber Pro de área específica de diseño de *software*. (Profesores que respondan la encuesta y desarrollen la guía de vigilancia epistemológica)

Fuente: esta investigación

3 El primer experto entrevistado fue seleccionado por el director de la carrera de Ingeniería Informática de la Universidad de Cádiz, que fue el lugar dónde se realizó la pasantía internacional.

4 El segundo experto entrevistado evidencia H-Index de 7 puntos (Según Scopus®) en el Departamento de Ciencias de la Computación y de la Decisión de la Universidad Nacional de Colombia, Sede Medellín.

Acerca de conceptos de fundamentos de programación

Es importante tener en cuenta que, para la identificación del conjunto de saberes propios de la programación de computadoras, la unidad de trabajo está basada en la producción teórica denominada *Computer Science Curricula* 2013. *Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* En este sentido, las definiciones científicas de los conceptos básicos de fundamentos de programación se tomarán a partir de este referente.

Acerca de los expertos

Uno de los puntos a desarrollar en la pasantía internacional, fue la identificación del experto en el contexto de la Universidad de Cádiz. Para ello, se entrevistó a una profesora titular con más de 20 años de experiencia que imparte el curso de fundamentos de programación de computadoras, la profesora tiene una alta productividad académica relacionada a la computación y a los fundamentos de programación.

Por su parte, el experto identificado en el Departamento de Ciencias de la Computación y de la Decisión de la Universidad Nacional de Colombia, Sede Medellín, tiene 16 años de experiencia en docencia e investigación en ciencias de la computación, es Investigador Senior según la Convocatoria 781 de 2017 de Colciencias, tiene también una alta productividad académica relacionada a la computación y a los fundamentos de programación

Acerca de los profesores extranjeros

Se tomó como referencia la clasificación mundial de universidades para el campo de conocimiento de ingeniería y computación dado por la Universidad de Shanghai Jiao Tong (Academic Ranking of World Universities - ARWU) del año 2016. Este listado puede observarse en la Tabla 4. En cumplimiento a las indicaciones de los instrumentos en cuanto a protección de la información y política de privacidad, los nombres de los profesores solo son mostrados con la inicial de su apellido. En negrilla se resaltan aquellos que participaron en esta investigación.

Tabla 4. Listado de los profesores extranjeros a quienes se les envió la encuesta

#	Universidad	País	Puntaje	Profesor
1	Massachusetts Institute of Technology (MIT)		100.0	?. G.
2	Nanyang Technological University		92.9	?. W.
3	*Stanford University*		92.9	?. G.
4	Tsinghua University		87.6	?. J.
5	*King Abdulaziz University*		87.4	?. M.
6	*National University of Singapore*		84.7	?. Y.
7	*The Imperial College of Science, Tech. & Medicine*		83.9	?. F.
8	University of California, Berkeley		82.8	?. G.
9	Harbin Institute of Technology		82.5	?. Z.
10	***The University of Texas at Austin***		**80.3**	**?. H.**
11	Swiss Federal Institute of Technology Lausanne		80.0	?. K.
12	*Georgia Institute of Technology*		79.8	?. O.
13	University of Illinois, Urbana-Champaign		79.4	?. V.
14	Zhejiang University		78.0	?. G.
15	*University of Michigan-Ann Arbor*		77.3	?. C.
16	Shanghai Jiao Tong University		77.2	?. L.
17	*Texas A&M University*		76.7	?. A.
18	Purdue University - West Lafayette		75.8	?. D.
19	University of Cambridge		75.1	?. R.
20	Southeast University		74.1	No identificado
21	*Xian Jiao Tong University*		74.0	?. H.
22	*South China University of Technology*		73.9	?. X.
23	*University of California, San Diego*		73.8	?. J.
24	City University of Hong Kong		73.6	?. F.
25	***Paris 13***		**72.7**	**?. L.**
26	*Carnegie Mellon University*		72.5	?. A.
27	Swiss Federal Institute of Technology Zurich		72.2	?. G.
28	***University of Science and Technology of China***		**72.1**	?. X.
29	*King Abdullah University of Science and Technology*		71.8	No identificado
30	*University of California, Los Angeles*		71.8	?. P.
31	The Hong Kong University of Science &Technology		71.5	?. L.
32	*Princeton University*		70.9	?. A.
33	Technical University of Denmark		70.8	?. H.
34	*University of California, Santa Barbara*		70.8	?. C.
35	The University of Manchester		70.7	?. L.
36	Northwestern University		70.4	?. H.
37	Harvard University		70.2	?. M.
38	University of Minnesota, Twin Cities		70.2	?. L.
39	Aalborg University		69.7	?. H.
40	***Huazhong University of Science and Technology***		**69.7**	?. H.
41	*Fudan University*		69.6	?. B.
42	*California Institute of Technology*		69.5	?. S.
43	The University of New South Wales		69.5	?. T.
44	University of Washington		69.5	?. S.
45	Universidad de Granada		68.5	?. C.
46	*National Taiwan University*		68.4	?. L.
47	***The Ohio State University – Columbus***		**68.2**	**?. M.**
48	University of Tehran		68.0	?. H.
49	Korea University		67.7	?. K.
50	*University of Toronto*		67.6	?. B.

Fuente: esta investigación

Con esta lista del top 50 del mundo, se identificó cada profesor con experiencia en impartir fundamentos de programación, o cursos relacionados —es importante mencionar que dicho curso aparece con nombres diferentes, tal como lo muestra la Tabla 5–. Al hacer un análisis sobre cada Syllabus, se logró determinar la relación directa con los fundamentos de programación dado el contenido explícito en cada uno de ellos. Nótese que 9 universidades con sus respectivos profesores; a todos los profesores —exceptuando 2 universidades donde no fue posible identificar el profesor que imparte dichos cursos— se les envió la invitación a participar en la investigación. De los 48 profesores invitados, solo 9 manifestaron su interés en participar. En negrilla se resaltan aquellos que participaron en esta investigación.

Tabla 5. Estructura académica y cursos del *top* 50 del mundo

#	Estructura Académica	Nombre del Curso
1	Electrical Engineering & Computer Science	Introduction to Computer Science and Programming
2	School of Computer Science and Engineering	Programming Environments & Tools
3	School of Engineering, Computer Science	Programming Abstractions
4	School of Software	Practical Training for Programming
5	Faculty of Computing and Information Technology	Principles of Programming
6	Computer Science	Programming Methodology
7	Department of Computing	Programming I
8	Computer Science	C for Programmers
9	School of Computer Science and Technology	Program Design and Programming Language
10	School of Information	Free and Open Source Software Development
11	SCHOOL OF COMPUTER AND COMMUNICATION SCIENCES	Algorithms
12	College of Computing	Programming Languages & Software Engineering
13	Computer Science	Intro computing: Non-tech, programming languages
14	College of Computer Science and Technology	Fundamental of Programming
15	School of Information	Programming education
16	School of Electronic, Information, and Electrical Engineering	Algorithms
17	Computer Science and Engineering	Programming languages
18	Department of Computer Science	Introduction to Computer Science
19	Faculty of Computer Science and Technology	Programming in Java
20	School of Computer Science and Engineering	program design and language
21	School of Electronic and Information Engineering	Computer programming
22	School of Automation Science and Engineering	Introduction to Computer Technology, System theory
23	Jacobs School of Engineering	Programming Languages
24	Department of Computer Science	Introduction to Computer Programming
25	Faculty of Engineering	Introduction à l'algorithmique et à la programmation
26	School of Computer Science	Programming Languages
27	Department of Computer Science	Introduction to Programming
28	School of Computer Science and Technology	Programming Languages
29	Computer Science Program	Programming Methodology and Abstractions
30	School of Engineering and Applied Science	Introduction to Computer Science, Programming Languages
31	Department of Computer Science and Engineering	Introduction to Computer Science, Introduction to Computing
32	Computer Science	Introduction to Programming Systems
33	Department of Applied Mathematics and Computer Science	Introduction to Programming
34	College of Computer Science	Computer Programming and Organization
35	School of Computer Science	Object Oriented Programming with Java 1
36	Department of Electrical Engineering and Computer Science	An intro to Computer Science for everyone
37	School of Engineering and Applied Sciences	INTRODUCTION TO COMPUTER SCIENCE I
38	Department of Computer Science and Engineering	Introduction to Computing and Programming Concepts
39	Faculty of Engineering and Sciences	Imperative Programming
40	Department of Computer Science and Technology	Senior Programming Design
41	Department of Information Management and Information Systems	Information System Development
42	Division of Engineering and Applied Science	Computer Algorithms
43	School of Computer Science and Engineering	Introduction to Programming
44	Computer Science and Engineering	Computer Programming I
45	Grado en Ingeniería Informática	Fundamentos de Programación
46	College of Electrical Engineering & Computer Science	Computer Programming
47	Department of Computer Science and Engineering	Introduction to Computer Programming in C++ for Engineers and Scientists
48	Department of Physics	Fundamentals of Computers and Programming
49	Division of Computer and Communication Engineering	Introduction to Computer, Data Structures and Algorithms
50	Department of Computer Science	Introduction to Computer Programming

Fuente: esta investigación

Acerca de los profesores colombianos

Fueron 187 las instituciones de Educación Superior las que presentaron las pruebas Saber Pro en 2016, y en el área específica de diseño de *software*, el promedio nacional fue de 152.14. Este valor es el resultado de la sumatoria de todos los puntajes en el área de todas las instituciones que presentaron la prueba, dividido por el número de dichas instituciones. En este sentido, sólo 84 instituciones en el país lograron un puntaje por encima del promedio nacional. La Tabla 6 muestra el listado de las instituciones junto con el profesor que fue identificado como responsable de impartir el curso de fundamentos de programación —o su equivalente–. Al igual que en el contexto internacional, los docentes en negrilla representan aquellos que desearon participar en la investigación; únicamente cuatro de los 84 docentes participaron en la encuesta. Similar a la aplicación de instrumentos para los profesores extranjeros, los nombres de los profesores son mostrados con la inicial de su primer apellido en cumplimiento a la protección de información y a la política de privacidad acordada con los individuos encuestados.

Tabla 6. Listado de los profesores colombianos a quienes se les envió la encuesta

#	Universidad	Puntaje	Profesor
1	Universidad Icesi-Cali	202	?. M.
2	Universidad de los Andes-Bogotá D.C.	197	?. L.
3	Universidad de San Buenaventura-Cali	190	?. S.
4	Universidad del Valle-Cali	190	?. G.
5	Universidad EIA-Medellín	189	No identificado
6	Universidad Nacional de Colombia-Bogotá D.C.	189	?. P.
7	Universidad Nacional de Colombia-Medellín	189	?. H.
8	Corporación Universitaria Lasallista-Caldas	187	?. R.
9	Universidad de Caldas-Manizales	186	No identificado
10	Universidad de Antioquia-Medellín	185	?. J.
11	Universidad Distrital “Francisco José de Caldas”-Bogotá D.C.	185	?. J.
12	Pontificia Universidad Javeriana-Bogotá D.C.	184	?. P.
13	Unipanamericana - Fundación Universitaria Panamericana-Bogotá	181	No identificado
14	Universidad Francisco de Paula Santander-Cúcuta	181	No identificado
15	Pontificia Universidad Javeriana-Cali	180	?. G.
16	Universidad Pontificia Bolivariana-Bucaramanga	179	?. C.
17	Universidad de la Sabana-Chía	178	?. G.
18	Universidad del Cauca-Popayán	176	?. P.
19	Universidad Tecnológica de Bolívar-Cartagena	175	?. G.
20	Escuela Colombiana de Ingeniería “Julio Garavito”-Bogotá D.C.	174	?. S.
21	Politécnico Colombiano “Jaime Isaza Cadavid”-Medellín	174	?. G.
22	Universidad Autónoma del Caribe-Barranquilla	174	?. C.
23	Universidad EAFIT-Medellín	174	?. P.
24	Universidad el Bosque-Bogotá D.C.	173	No identificado
25	Universidad Autónoma de Manizales-Manizales	172	?. C.
26	Universidad Pontificia Bolivariana-Medellín	172	No identificado
27	Universidad Católica de Colombia-Bogotá D.C.	171	?. G.
28	Universidad de Cartagena-Cartagena	170	?. M.
29	Universidad del Norte-Barranquilla	170	?. C.
30	Universidad del Quindío-Armenia	170	?. E.
31	Universidad Sergio Arboleda-Bogotá D.C.	169	?. S.
32	Politécnico Grancolombiano-Bogotá D.C.	169	?. M.
33	Universidad de San Buenaventura-Bogotá D.C.	168	No identificado
34	Escuela de Administración y Mercadotecnia del Quindío-Armenia	168	No identificado
35	Politécnico Grancolombiano-Medellín	167	?. G.
36	Universidad de Ibagué-Ibagué	167	?. D.
37	Universidad de Nariño-Pasto		?. I.
38	Universidad EAN-Medellín.	167	?. G.
39	Universidad de Medellin-Medellin	167	?. D.
40	Universidad Santiago de Cali-Cali	167	?. G.
41	Universidad de San Buenaventura-Medellín	166	?. N.
42	Universidad Industrial de Santander-Bucaramanga	166	?. R.
43	Universidad Manuela Beltrán --Bogotá D.C.	165	No identificado

44	Universidad Pedagógica y Tecnológica de Colombia-Tunja	164	?. M.
45	Universidad Autónoma de Occidente-Cali	164	?. H.
46	Universidad EAN-Bogotá D.C.	164	?. T.
47	Universidad Nacional de Colombia-Manizales	163	?. G.
48	Corporación Universitaria Adventista-Medellín	163	No identificado
49	Corporación Universitaria del Meta-Villavicencio	162	?. C.
50	Fundación Universidad de Bogotá “Jorge Tadeo Lozano”-Bogotá	161	?. C.
51	Universidad Cooperativa de Colombia-Bogotá D.C.	161	?. M.
52	Tecnológico de Antioquia-Medellín	161	?. G.
53	Universidad Autónoma de Bucaramanga -Bucaramanga	161	?. M.
54	Fundación Universitaria Konrad Lorenz-Bogotá D.C.	161	?. T.
55	Institución Universitaria de Envigado-Envigado	160	?. G.
56	Universidad Católica de Oriente-Rionegro	160	No Identificado
57	Universidad del Magdalena - Santa Marta	159	?. M.
58	Universidad Santo Tomas-Tunja	159	?. A.
59	Universidad Libre-Bogotá D.C.	159	?. V.
60	Colegio Mayor del Cauca-Popayán	159	?. A.
61	Escuela Tecnológica Instituto Técnico Central -Bogotá D.C.	159	No identificado
63	Universidad Central-Bogotá D.C.	158	No identificado
64	Universidad Francisco de Paula Santander-Ocaña	157	?. G.
65	Corporación Universidad de la Costa -Barranquilla	157	?. H.
66	Institución Universitaria Antonio José Camacho -Cali	157	No identificado
67	Universidad de Cundinamarca-Ubaté	157	?. T.
68	Corporación Universidad Piloto de Colombia-Bogotá D.C.	156	?. B.
69	Corporación Universitaria Minuto de Dios -Bogotá D.C.	156	?. P.
70	Instituto Tecnológico Metropolitano-Medellín	156	?. J.
71	Unidad Central del Valle del Cauca-Tuluá	155	?. T.
72	Universidad de Cundinamarca-Fusagasugá	155	?. H.

73	Universidad del Sinú “Elías Bechara Zainúm” - Cartagena	155	No identificado
74	Universidad Tecnológica de Pereira - Pereira	155	?. V.
75	Universidad de la Amazonía-Florencia	155	?. T.
76	Universidad de Pamplona-Pamplona	155	?. C.
77	Universidad ECCI-Bogotá D.C.	155	No identificado
78	Universidad Pontificia Bolivariana-Montería	154	?. S.
79	Universidad Surcolombiana-Neiva	154	?. B.
80	Universidad de los Llanos-Villavicencio	154	?. P.
81	Universidad de Santander - Bucaramanga	154	?. E.
82	Fundación Universitaria Juan de Castellanos-Tunja	154	No identificado
83	Universidad Santo Tomás-Bogotá D.C.	154	?. G.
84	Universidad Simón Bolívar-Barranquilla	153	No identificado

Fuente: esta investigación

Estos datos confirman que, de 48 invitaciones enviadas a los profesores extranjeros —dado que no se logró identificar a los profesores en 2 universidades extranjeras de un total de 50—, 9 de ellos manifestaron su intención de participar. Curiosamente, en el contexto nacional, de las 70 invitaciones enviadas a los profesores nacionales —dado que no se logró identificar a los profesores en 14 instituciones de Educación Superior colombianas del total de 84—, solamente 4 de ellos manifestaron su intención de participar. Analizando este contraste, la investigación tiene mayor cantidad de información del escenario extranjero que del nacional. Se garantizó que las invitaciones y su participación se hiciese en un período de tiempo de actividad académica —evitando el período de vacaciones y recesos académicos—; no obstante, los resultados de participación marcan la diferencia al evidenciar un mayor interés por parte de los profesores extranjeros que de los nacionales.

Acerca de la validación de los instrumentos de recolección de información

Los instrumentos de recolección de información planteados para la presente investigación fueron validados por 3 expertos; el primer experto es Doctor en Ciencias de la Educación de la Universidad de Barcelona —realiza

apreciaciones verbales sobre el uso del lenguaje y la intencionalidad de los instrumentos—, el segundo experto es Doctor en Ingeniería – Línea Automática de la Universidad Nacional de Colombia sede Manizales y el tercer experto es Doctor en Ingeniería de la Universidad de los Andes.

3.4 Destilación de información por medios computacionales

Desde una mirada instrumental, existen varias herramientas computacionales para trabajar con información de corte cualitativo, entre ellas están ATLAS.ti®, Ethnograph®, NVIVO®, QDA Miner®, y Nud*ist®, entre otros. En términos generales, estas herramientas computacionales permiten llevar a cabo las tareas básicas en investigaciones cualitativas, a fin de integrar, en una red estructural compleja, las realidades evidenciadas en documentos. A través de técnicas de categorización, estructuración y teorización, y con los operadores booleanos, semánticos y de proximidad, de esta forma, estas herramientas permiten organizar la información en un esquema comprensible por parte de los investigadores (Martínez, 2004).

En el desarrollo de las técnicas de análisis de información, se debe tener en cuenta que la destilación de información será soportada con los formatos propios de dicha técnica; no obstante, el análisis hermenéutico de grandes cantidades de información textual fue soportado por la herramienta computacional QDA Miner® (producto registrado por PROVALIS Research, una empresa canadiense dedicada al desarrollo de herramientas de soporte a la investigación cualitativa).

Por otra parte, para el diseño de los esquemas preconceptuales que soportan las definiciones ontológicas de los conceptos desarrollados se utiliza graficadores especializados como soporte tecnológico. En este sentido, el diseño de los esquemas preconceptuales se desarrolla con la herramienta Microsoft Visio®. Cabe resaltar que la plantilla de elaboración de los esquemas preconceptuales para Microsoft Visio® es producida por el Grupo de Investigación en Lenguaje Computacional de la Universidad Nacional de Colombia, Sede Medellín.

Finalmente, el diseño e implementación de la plataforma para la aplicación y obtención de la información concerniente a las encuestas a profesores extranjeros y nacionales, fue realizada a través de *Microsoft Visual Studio® 2017 Enterprise Edition*, con el uso de Microsoft *SQL Server® 2016* como

motor de base de datos. El despliegue de dicha plataforma fue realizado en uno de los servidores del grupo de investigación GALERAS.NET, el *software* en cuestión está debidamente licenciado a dicho grupo de investigación. Todos los instrumentos de recolección de información, a saber: entrevista, encuesta y formato de vigilancia epistemológica están escritos en inglés, con el propósito de poder abarcar diferentes contextos en su aplicación. Las plataformas de aplicación de instrumentos y procesamiento de información fueron debidamente registradas en la Dirección Nacional de Derecho de Autor del Ministerio del Interior y de Justicia, en la República de Colombia. El *software* producido dispone actualmente del Registro de Soporte Lógico emitido por la entidad competente.

4. OBJETOS POR TRANSPONER

En el contexto de la enseñanza de los fundamentos de programación, varios conceptos son requeridos. En el estudio realizado por Piteira y Costa (2013, p. 76), existen 11 conceptos identificados, como principales, al momento de enseñar dicha temática, estos conceptos son: variables, estructuras de selección, estructuras cíclicas, parámetros, arreglos, punteros y referencias, tipos de estructuras de datos, tipos de datos abstractos, manejo de entrada y salida, manejo de errores, y uso de bibliotecas del lenguaje. Estos conceptos por su alto grado de abstracción han sido identificados como conceptos que presentan retos al momento de ser enseñados.

A partir del diálogo con los expertos, se sugirió identificar únicamente los conceptos universales que forman parte de los fundamentos de programación sin importar qué tipo de lenguaje de programación se utilice para la implementación de soluciones computacionales. Para tomar esta decisión, se realizó previamente un análisis de los principales lenguajes de programación, de dicho análisis se logró obtener la lista de los libros de texto de dichos lenguajes de programación que forman parte de la bibliografía de fundamentos de programación del *top* 50 de universidades —listado base que se tomó para determinar la población intencional de profesores extranjeros–. Los lenguajes de programación usados se pueden observar a través de la Tabla 7; en negrilla se resaltan los nombres de los cursos y los lenguajes utilizados los cuales corresponden a los profesores que participaron en esta investigación.

Tabla 7. Lenguajes de programación usados en fundamentos de programación

#	Nombre del curso (extranjeros)	Lenguaje de programación usado
1	Introduction to Computer Science and Programming	Alice
2	Programming Environments & Tools	Python
3	Programming Abstractions	C/C++
4	Practical Training for Programming	JavaScript

5	Principles of Programming	Python
6	Programming Methodology	C++
7	Programming I	C/C++
8	C for Programmers	ANSI C
9	Program Design and Programming Language	JavaScript
10	Free and Open Source Software Development	gcc
11	Algorithms	BlueJ
12	Programming Languages & Software Engineering	Java
13	Intro Computing: Non-Tech, Programming Languages	C++
14	Fundamental of Programming	ANSI C
15	Programming education	C/C++
16	Algorithms	Python
17	Programming languages	C#
18	Introduction to Computer Science	C++
19	Programming in Java	Java
20	Program design and language	C#
21	Computer programming	MATLAB
22	Introduction to Computer Technology, System theory	ANSI C
23	Programming Languages	Scheme/Mozart
24	Introduction to Computer Programming	C/C++
25	Introduction à l'algorithmique et à la programmation	ISO/IEC C++ 2011
26	Programming Languages	ISO/IEC C++ 2011
27	Introduction to programming	ANSI C
28	Programming Languages	C++
29	Programming Methodology and Abstractions	C/C++
30	Introduction to Computer Science, Programming Languages	BlueJ
31	Introduction to Computer Science, Introduction to Computing	C#
32	Introduction to Programming Systems	ISO/IEC C++ 2003
33	Introduction to Programming	Java
34	Computer Programming and Organization	Python
35	Object Oriented Programming with Java 1	Java
36	AN INTRO TO COMPUTER SCIENCE FOR EVERYONE	C#
37	INTRODUCTION TO COMPUTER SCIENCE I	ANSI C
38	Introduction to Computing and Programming Concepts	Java
39	Imperative Programming	ANSI C
40	SENIOR PROGRAMMING DESIGN	MATLAB
41	Information System Development	MATLAB
42	Computer Algorithms	Alice
43	Introduction to Programming	Java
44	Computer Programming I	C/C++
45	Fundamentos de Programación	C++
46	Computer Programming	ANSI C
47	Introduction to Computer Programming in C++ for Engineers and Scientists	C++
48	Fundamentals of Computers and Programming	C/C++
49	Introduction to Computer, Data Structures and Algorithms	C#
50	Introduction to Computer Programming	ISO/IEC C++ 2003
#	Nombre del curso (nacionales)	Lenguaje de programación usado
7	**Fundamentos de Programación**	***Python***
32	**Programación de Computadores**	***C++***
37	**Fundamentos de Programación**	***C++***
39	**Fundamentos de Programación**	***Java***
	El resto de los cursos en Colombia	*C, Python, y Java*

Fuente: esta investigación

Al identificar los cursos y el tipo de lenguaje de programación utilizado, se pudo enfocar la atención tanto en cada *syllabus* como en los textos guía y los recursos asociados a cada curso. Este fue el punto de partida para iniciar la construcción de un *dossier* de documentos con el fin de establecer un marco común de trabajo.

Al realizar el análisis de los *syllabus*, se encontró que 6 conceptos son comunes al momento de trabajar en la implementación de soluciones computacionales con dichos lenguajes. Éstos son: variables, constantes, expresiones aritmético-lógicas, condicionales, ciclos y funciones. Independientemente de la metodología usada por el profesor, de las herramientas y lenguajes de programación usados en el curso, los anteriores 6 conceptos son de carácter universal y deben ser abordados en dicho espacio académico. En sínte-

sis, se puede afirmar que un estudiante que termina el curso de fundamentos de programación debe haber desarrollado como mínimo competencias cognitivas y aptitudinales en los 6 conceptos anteriormente expuestos.

Por tal razón, esta investigación basada en el estudio de la transposición didáctica en la enseñanza de los fundamentos de programación de computadoras hace énfasis en los siguientes 6 conceptos: variable, constante, expresión aritmético-lógica, condicional, ciclo y función.

4.1 Análisis de la revisión documental

Cuando se habla de programas de pregrado relacionados con la computación, tradicionalmente se asocia este concepto a las cinco disciplinas propuestas por la ACM, AIS e IEEE *Computer Society*, a saber: Ingeniería en Computación, Ciencias de la Computación, Tecnología de Información, Sistemas de Información e Ingeniería de *Software* (ACM, AIS & IEEE *Computer Society*, 2005). De hecho, este equipo de trabajo conformado por las tres asociaciones ha establecido tradicionalmente recomendaciones curriculares que permiten organizar el conocimiento en dichas disciplinas.

En este sentido, debido a que los fundamentos de programación es una temática común a las cinco disciplinas, dicha formación se imparte en la mayoría de los programas académicos relacionados con la computación que han tomado las recomendaciones curriculares anteriormente citadas. De esta forma, cada disciplina profundiza en las definiciones de los conceptos que le corresponden. Así, cada una de ellas ha establecido en los últimos 10 años una serie de documentos que sirven como referente internacional sobre los conceptos que deberían ser manejados al interior de los cursos.

Este es el caso de la disciplina denominada Ciencia de la Computación —del inglés *Computer Science*–. Las indicaciones de los expertos sugieren partir del estudio de las últimas recomendaciones curriculares para dicha disciplina, ya que los fundamentos de programación tienen mayor profundidad en esta disciplina en particular. Al hacer el estudio del documento de recomendaciones curriculares para Ciencia de la Computación, un capítulo importante describe el cuerpo de conocimiento para dicha disciplina (ACM & IEEE *Computer Society*, 2013). Este documento, además, contiene información sobre los *syllabus* de las universidades cuyos profesores contestaron la encuesta y suministraron dichos documentos, donde se puede

evidenciar el contenido de los programas de fundamentos de programación. Con esta información, el documento en mención —con los *syllabus*: 9 extranjeros y 4 nacionales— y la revisión de la bibliografía sugerida, se ha logrado elaborar la caracterización de los 6 objetos de saber presentes en esta investigación. Para ello, se ha hecho uso de la herramienta *QDA Miner*® con el fin de elaborar un análisis categorial de cada concepto usando mecanismos de lingüística computacional, junto con *software* producido por el grupo de investigación GALERAS.NET como herramientas auxiliares de lingüística de *corpus*. La Figura 6, muestra la frecuencia porcentual de uso de cada código de búsqueda a través del corpus lingüístico creado por los documentos en mención.

Data - Book1.ppj

DOCUMENTS 1/14

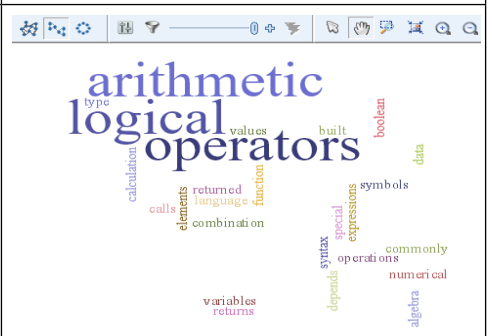
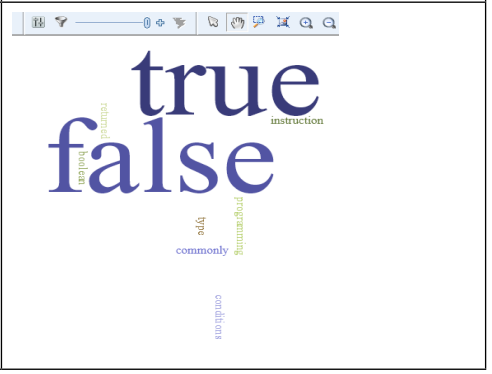
DOCUMENTS	VARIABLE	CONSTANT	ARITHMETIC	CONDITIONA	LOOP	FUNCTION
Computer Science Curricula 2013	0.78	0.53	0.39	0.49	0.58	0.63
Syllabus 10. Free and Open Source Software Development (gcc	0.38	0.50	0.65	0.72	0.54	0.34
Syllabus 13. Intro Computing: NonTech, Programming Language	0.54	0.47	0.46	0.45	0.59	0.70
Syllabus 24. Introduction to Computer Programming (C/C++)	0.78	0.36	0.57	0.39	0.69	0.56
Syllabus 25. Introduction à l'algorithmique et à la programmation	0.59	0.58	0.36	0.71	0.46	0.50
Syllabus 27. Introduction to programming (ANSI C)	0.79	0.54	0.41	0.44	0.80	0.75
Syllabus 28. Programming Languages (C++)	0.32	0.33	0.60	0.44	0.41	0.60
Syllabus 36. AN INTRO TO COMPUTER SCIENCE FOR EVER	0.76	0.63	0.44	0.52	0.53	0.30
Syllabus 40. SENIOR PROGRAMMING DESIGN (Matlab)	0.78	0.52	0.52	0.43	0.36	0.49
Syllabus 47. Introduction to Computer Programming in C++ for Er	0.64	0.68	0.43	0.65	0.51	0.66

All

Figura 6. Resultados de búsqueda en corpus lingüístico (Fuente: esta investigación)

Este análisis de corte cuantitativo permite evidenciar que, sin importar el lenguaje de programación, los conceptos clave que han sido identificados en la presente investigación prevalecen en cada *syllabus*. De igual manera, son también desarrollados en los textos guía y en los recursos que provee cada profesor a sus estudiantes.

Al realizar los análisis de enlazamiento de nube de palabras, una de las herramientas produjo las siguientes nubes conceptuales por medio de la detección automatizada de sintagmas nominales y sintagmas verbales a partir del *corpus* lingüístico, tal como lo muestra la Figura 7.

Concepto: *Variable* Palabras clave: *space, represents, represented, storage, abstraction, programming, type, assigned, language, memory, identifier, short, data, supported, corresponds, element, reserved*	
Concepto: *Constant* Palabras clave: *fixed, execution, change, read, memory, reserved, corresponds, length, element, main, altered, stores, modified, programmer, area, values*	
Concepto: *Arithmetic-logical expression* Palabras clave: *type, arithmetic, logical, Boolean, values, built, data, symbols, calculation, calls, elements, returned, function, language, combination, special, expressions, syntax, operations, commonly, numerical, depends, variables, returns, algebra*	
Concepto: *Conditional* Palabras clave: *true, false, instruction, returned, Boolean, commonly, type, programming, conditions*	

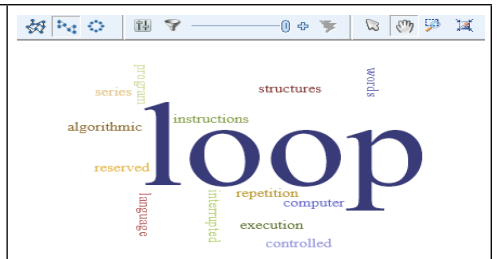
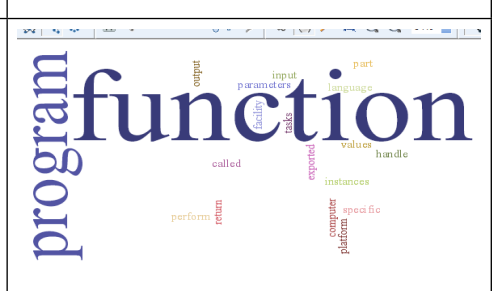
Concepto: *Loop* Palabras clave: *series, program, structures, words, algorithmic, instructions, reserved, language, interpreted, repetition, computer, execution, controlled*	
Concepto: *Function* Palabras clave: *part, output, input, parameters, program, called, facility, tasks, values, handle, exported, instances, perform, return, computer, specific, platform.*	

Figura 7. Nubes de palabras a partir del corpus lingüístico (Fuente: esta investigación)

Según Hunston (2006, p. 51)

> *La lingüística de corpus puede considerarse un método sofisticado para encontrar respuestas a los tipos de preguntas que los lingüistas siempre han formulado. Un corpus grande puede ser un banco de pruebas para las hipótesis y se puede usar para agregar una dimensión cuantitativa a muchos estudios lingüísticos. También es cierto, sin embargo, que el software de corpus presenta al investigador el lenguaje en una forma que normalmente no se encuentra y, que esto puede resaltar patrones que a menudo pasan desapercibidos. La lingüística de corpus también ha llevado a una reevaluación de cómo es el lenguaje.*

A partir de los constructos lingüísticos de la Figura 7, se pudo obtener las definiciones que caracterizan los conceptos seleccionados con el *software* especializado. Una vez identificadas las palabras claves y su peso correspondiente en la nube semántica, se procedió a realizar la construcción del concepto a partir del *corpus* lingüístico. A través de un ejercicio de interpretación de palabras —no todas fueron incluidas— que son parte de los elementos de las nubes semánticas, se proponen las siguientes definiciones de acuerdo con la Tabla 8:

Tabla 8. Definiciones propuestas para los 6 conceptos seleccionados

Concepto en inglés	Concepto en español
Variable: a variable is a reserved memory space to store a value that corresponds to a data type supported by the programming language.	Variable: una variable es un espacio de memoria reservado para almacenar un valor que corresponde a un tipo de datos compatible con el lenguaje de programación.
Constant: a constant is a value that cannot be modified during the execution of a program, it can only be read.	Constante: una constante es un valor que no se puede modificar durante la ejecución de un programa, solo se puede leer.
Arithmetic-Logical Expression: an arithmetic-logical expression is a combination of variables, constants, operators and function calls built according to the language syntax that returns a value.	Expresión Aritmético-Lógica: una expresión aritmético-lógica es una combinación de variables, constantes, operadores y llamadas a función, construidas de acuerdo con la sintaxis del lenguaje que devuelve un valor.
Conditional: a conditional is an algorithmic structure from which a Boolean value (true or false) is returned. According to the returned value, the execution of a program can be varying.	Condicional: un condicional es una estructura algorítmica a partir de la cual se devuelve un valor booleano (verdadero o falso). De acuerdo con el valor devuelto, la ejecución de un programa puede variar.
Loop: a loop is an algorithmic structure that allow the controlled repetition of a series of instructions within a computer program.	Ciclo: un ciclo es una estructura algorítmica que permite la repetición controlada de una serie de instrucciones dentro de un programa de computadora.
Function: a function is a part of a computer program that perform specific tasks and have the facility to be called in different instances within the program; they can handle input parameters and can return output values.	Función: una función es una parte de un programa de computadora que realiza tareas específicas y tiene la facilidad de ser llamado en diferentes instancias dentro del programa; pueden manejar parámetros de entrada y pueden devolver valores de salida.

Fuente: esta investigación

4.2 Opiniones profesorales

A los profesores extranjeros como a los nacionales que mostraron interés en participar en la investigación, se les aplicó una encuesta *online*, donde se pudo indagar sobre aquellas actividades desarrolladas en clase con el fin de descubrir las características de los objetos de saber en cuestión a partir del ejercicio de la docencia. La intención es caracterizar los objetos de saber a través de la práctica de la enseñanza de estos en escenarios reales. El ejercicio de la docencia tiene un gran valor para esta investigación, por tal razón se preguntó sobre las estrategias motivacionales para enseñar dichos conceptos, también por los aspectos más fáciles y los más difíciles de enseñar de acuerdo con cada profesor. El instrumento utilizado se denomina *"survey about characterization of knowledge objects"*. Es importante resaltar que, todos los instrumentos de recolección de información fueron

sometidos a un proceso de validación por parte de expertos. La Tabla 9 establece la codificación de los profesores encuestados y el lenguaje de programación que han usado en el ejercicio de la docencia.

Tabla 9. Codificación de los profesores encuestados

	País	Universidad	Código profesor	Lenguaje de programación
Contexto Internacional		The University of Texas at Austin	10.H	*gcc*
		University of Illinois, Urbana-Champaign	13.V	*C++*
		City University of Hong Kong	24.F	*C/C++*
		Paris 13	25.L	*ISO/IEC C++ 2011*
		Swiss Federal Institute of Technology Zurich	27.G	*ANSI C*
		University of Science and Technology of China	28.X	*C++*
		Northwestern University	36.H	*C#*
		Huazhong University of Science and Technology	40.H	*MATLAB*
		The Ohio State University - Columbus	47.M	*C++*
Contexto Nacional		Universidad Nacional de Colombia-Medellín	7.H	*Python*
		Politécnico Grancolombiano-Bogotá D.C.	32.M	*C++*
		Universidad de Nariño-Pasto	37.I	*C++*
		Universidad de Medellin-Medellin	39.D	*Java*

Fuente: esta investigación

Esta tabla permitió establecer la codificación de los 13 profesores encuestados. Con esta codificación se realizó el análisis de las respuestas a través de un enfoque hermenéutico. Dicho análisis se presentó por medio de una serie de tablas discriminadas para cada concepto. Cabe resaltar que se realizó una tabulación de las respuestas de los profesores encuestados discriminadas por concepto.

Dadas las respuestas al concepto de variable, se evidencia una cercanía al concepto que se maneja en la matemática. Su representación abstracta proviene del concepto de almacenamiento de valores en memoria y puede cambiar durante la ejecución de un programa de computadora. Como estrategias motivacionales, algunos profesores describieron el uso de cajas donde objetos —en este caso valores— pueden estar contenidos en ellas. Los

profesores en general reconocieron sus limitaciones en el campo de la didáctica; de hecho, algunos admitieron que manejan una forma tradicional de enseñar y tienden a reducir el concepto de pedagogía hacia meramente el acto de enseñar. Finalmente, no se resaltaron mayores dificultades de enseñanza, exceptuando los aspectos relacionados al direccionamiento de memoria y su nomenclatura en código de bajo nivel.

Al igual que el concepto de variable, los profesores abordaron el concepto de constante desde una perspectiva matemática, argumentando que sus valores almacenados no pueden ser alterados durante la ejecución de un programa de computadoras. En sus respuestas, los profesores manifestaron gran cercanía con el concepto de constante, que se maneja en la matemática, y usan ejemplos a partir de ella, así se enseña el concepto de constante en los fundamentos de programación. Desde el punto de vista motivacional, se usaron prácticamente recursos similares a los planteados para explicar el concepto de variable. De igual forma, no se hizo explícita la dificultad al enseñarlo a excepción del direccionamiento de memoria y su nomenclatura en bajo nivel para el manejo. Adicionalmente, un profesor manifestó que, algunos de sus estudiantes no comprenden la utilidad de las constantes y argumentan que, podrían usar simplemente variables a la cuales no se les afectará su valor a través de expresiones de asignación; aquí, el profesor lo expresó como una dificultad de enseñar: no el concepto de constante *per sé*, sino su utilidad en los fundamentos de programación.

Según las respuestas de los profesores relacionadas al concepto de expresión aritmético-lógica, el éxito del aprendizaje de dichas expresiones se sustentó a través de la realización de una gran cantidad de ejercicios. Es un común denominador el ejercicio riguroso con expresiones aritmético-lógicas a partir de problemas que, por lo general, son planteados a partir de la matemática, la física y la ingeniería. Para la motivación, se usaron estrategias como resolución de problemas en grupos, olimpiadas de matemáticas computacionales, talleres de resolución de problemas de física, estimación del valor de una variable a través de expresiones anidadas. Algunos profesores otorgan bonificaciones en las calificaciones como factor motivacional cuando estas actividades de aula se desarrollan adecuadamente. Se encuentra en sus respuestas que también es común la facilidad de enseñar los conceptos si son abordados desde la matemática —en este particular, el profesor que trabaja con MATLAB® rescató el gran potencial de la herramienta–. De igual manera, la mayoría de los profesores aseguró que no

existe mayor dificultad al momento de enseñar el concepto de expresión aritmético-lógica, salvo por la complejidad de las expresiones cuando éstas crecen en número de variables, constantes y operadores utilizados.

Para abordar el concepto de condicional, las respuestas de los profesores al respecto reflejaron el uso de una analogía que consiste en un camino que se bifurca y que viajar por una opción o por la otra depende de un proceso de toma de decisiones —decisión no humana— basada en lógica matemática, la cual se expresa a través del algebra de Boole. Según los profesores, los condicionales son estructuras de control de ejecución dentro de un programa de computadoras, se basan en la comparación de una situación de naturaleza booleana cuyos únicos posibles resultados son verdadero —*true*— o falso —*false*–. Las dificultades manifiestadas en la enseñanza del concepto se presentaron al momento de usar la sintaxis propia de los lenguajes de programación; puesto que algunos de estos pueden hacer uso de contracciones de nomenclatura para expresar una estructura condicional; por otra parte, algunos profesores aseguraron que la enseñanza de los condicionales se facilita en tanto haya una sólida fundamentación en lógica matemática, y, en especial, en álgebra Booleana y el conocimiento de las tablas de verdad.

Se reiteró que la rigurosidad en los ejercicios de aplicación es factor clave para lograr el aprendizaje de los conceptos; similar a la enseñanza de las expresiones aritmético-lógicas. La enseñanza del concepto de ciclo se fortalece a través de la realización de gran cantidad de ejercicios. Algunos profesores argumentaron que, los ciclos son muy útiles en los procesos de simulaciones físicas; el hecho de poder hacer iteraciones controladas sobre un modelo matemático constituye la base de los procesos de simulación. El ciclo cobra importancia para tal razón. En este sentido, motivacionalmente, algunos profesores hacen uso de analogías para expresar situaciones de iteraciones controladas. El no control de las iteraciones conlleva a estancamientos en ciclos infinitos por parte de los estudiantes; estos estancamientos se reconocen como retos en el aprendizaje, más no en la enseñanza.

Finalmente, las respuestas de los profesores acerca de la enseñanza del concepto de función reflejaron uno de los mayores retos en su labor. Desde una perspectiva motivacional, los profesores argumentaron su búsqueda por dividir un programa de computadoras en tareas puntuales; algunos de ellos se basaron en el concepto de función matemática, otros lo orientaron hacia funciones gráficas, otros hacia tareas específicas de simulación o in-

cluso de videojuegos. Con respecto a las funciones, la mayoría de los profesores encuestados concordó en que se trata de un reto de enseñanza el manejo de los argumentos de la función, en especial, el paso de parámetros por valor o por referencia. Algunos profesores argumentaron que suelen encontrar dificultades en algunos estudiantes con el uso de parámetros por referencia, debido al hecho de manejar direccionamiento de memoria en cada salto. Incluso, hay quienes afirmaron que los saltos suelen perder la secuencialidad de la ejecución del programa en algunos estudiantes.

Con la información recolectada fue posible hacer un análisis general sobre qué estrategias motivacionales están usando los profesores, así como los aspectos más fáciles y difíciles de enseñar sobre los objetos de saber. La síntesis del análisis está representada a través de estructuras ontológicas expresadas en esquemas preconceptuales desde la visión de la lingüística computacional.

Se han propuesto muchas definiciones de "ontología" en la literatura (Gruber & Olsen, 1994), y se hacen clasificaciones de diferentes tipos de vocabularios, tesauros y bases de conocimiento, enfocadas en criterios tales como su uso previsto, grado de formalización o interpretación filosófica (Smith *et al.*, 2007). Dado que una ontología permite representar el conocimiento a través de conceptos y relaciones, esta investigación se basó en dicha definición para la construcción de sus definiciones conceptuales.

Por su origen, las ontologías surgen de la rama de la filosofía conocida como metafísica, que se ocupa de cuestiones como "¿qué existe?" y "¿cuál es la naturaleza de la realidad?" Una de las cinco ramas tradicionales de la filosofía, la metafísica, se ocupa de explorar la existencia a través de propiedades, entidades y relaciones, como las que existen entre los particulares y los universales, las propiedades intrínsecas y extrínsecas, o la esencia y la existencia. La metafísica ha sido un tema constante de discusión desde la historia registrada.

Desde mediados de la década de 1970, los investigadores en el campo de la inteligencia artificial —IA— han reconocido que la ingeniería del conocimiento es la clave para construir sistemas de inteligencia artificial grandes y poderosos. Los investigadores de IA argumentaron que podían crear nuevas ontologías como modelos computacionales que permiten ciertos tipos de razonamiento automatizado, que solo tuvieron un éxito marginal. En la década de 1980, la comunidad de IA comenzó a usar el término ontología

para referirse tanto a una teoría de un mundo modelado como a un componente de sistemas basados en el conocimiento. Algunos investigadores, inspirándose en ontologías filosóficas, vieron la ontología computacional como un tipo de filosofía aplicada (Gruber, Liu & Özsu, 2008).

Las ontologías computacionales proporcionan una representación simbólica de los objetos conceptuales —clases, propiedades y relaciones— con el fin de hacer manifiesto un conocimiento en un dominio específico. El modelado de la ontología computacional es posible efectuarlo a través de definiciones matemáticas, lógicas y estructurales (Lim, Liu, & Lee, 2011). Con las características de los esquemas preconceptuales, las ontologías computaciones son factibles de ser modeladas. De hecho, las ontologías que se presentan a continuación constituyen una base de datos que describe los conceptos del dominio específico, y cómo se relacionan entre sí.

En los últimos años, el desarrollo de ontologías como explicaciones formales explícitas de los términos en un dominio del conocimiento y sus relaciones entre ellos, se ha gestado en laboratorios de Inteligencia Artificial. Las ontologías se han vuelto comunes en la *World Wide Web*. Las ontologías en la Web van desde grandes taxonomías que categorizan sitios web —como en *Google*, *Bing*, o *Yahoo*!— hasta categorizaciones de productos para la venta y sus características —como en *Amazon.com*–. El Consorcio WWW —W3C— ha desarrollado el marco de descripción de recursos, o RDF —de sus siglas en inglés— un lenguaje para codificar el conocimiento en las páginas web con el propósito de que sea comprensible para los agentes electrónicos que buscan información (Brickley & Guha, 1999). La Agencia de Proyectos de Investigación Avanzada de Defensa —DARPA—, junto con el W3C, ha desarrollado el Lenguaje de marcado de agentes de DARPA —DAML— extendiendo RDF con construcciones más expresivas destinadas a facilitar la interacción del agente en la Web (Hendler & McGuinness, 2000). Muchas disciplinas ahora desarrollan ontologías estandarizadas para que los expertos del dominio puedan usarlas a fin de compartir y anotar información en sus campos. La medicina, por ejemplo, ha producido vocabularios grandes, estandarizados y estructurados como SNOMED (Price & Spackman, 2000) y la red semántica del *Unified Medical Language System* (Humphreys & Lindberg, 1993).

4.3 Conceptualización de los objetos de saber

Haciendo uso del análisis hermenéutico se formularon los siguientes constructos teóricos, en forma de ontologías computacionales, a partir de las experiencias de enseñanza descritas por los profesores. Estos constructos teóricos surgieron del análisis lingüístico y contemplan diferentes opiniones de los profesores alrededor de cada concepto tomando en cuenta también las definiciones que aparecen según la bibliografía correspondiente en los *syllabus*.

El término filosófico ontología fue por primera vez adaptado a la computación por Gruber (1993) como una "especificación explícita de conceptualización" para la comunidad de Inteligencia Artificial. Este término ha sido usado y definido de varias maneras diferentes por las comunidades de representación y razonamiento del conocimiento, desde entonces. Una ontología representa los conceptos dentro de un dominio y especifica cómo se relacionan los conceptos entre sí a través de axiomas lógicos expresados en un lenguaje formal (por ejemplo, lenguaje lógico de descripción de OWL por W3C para Web semántica). Es así como se fue acuñando el término desde la filosofía hacia las ciencias computacionales; en este último escenario, se las conoce como ontologías computacionales.

Las ontologías computacionales ayudan a generar un marco de entendimiento sobre cada concepto con sólidas relaciones lingüísticas. La conceptualización de los objetos de saber busca descubrir la naturaleza de dichos objetos con el fin de establecer el marco de referencia a partir del cual se deben enseñar dichos objetos. Por cada objeto de saber —variable, constante, expresión aritmético-lógica, condicional, ciclo y función— se propone una definición ontológica que sirve de base conceptual para determinar los niveles de transposición didáctica para cada objeto; de igual manera, las estrategias de enseñanza propuestas a la luz de la teoría de transposición didáctica trabajan con dichas definiciones ontológicas.

A continuación, cada objeto de saber se expresa en forma de ontología computacional usando esquemas preconceptuales a partir de los insumos de la investigación. Esto con el fin de manejar un lenguaje controlado dentro de un dominio de conocimiento específico, en este caso: los fundamentos de programación de computadoras.

Conceptualización del objeto de saber: "Variable"

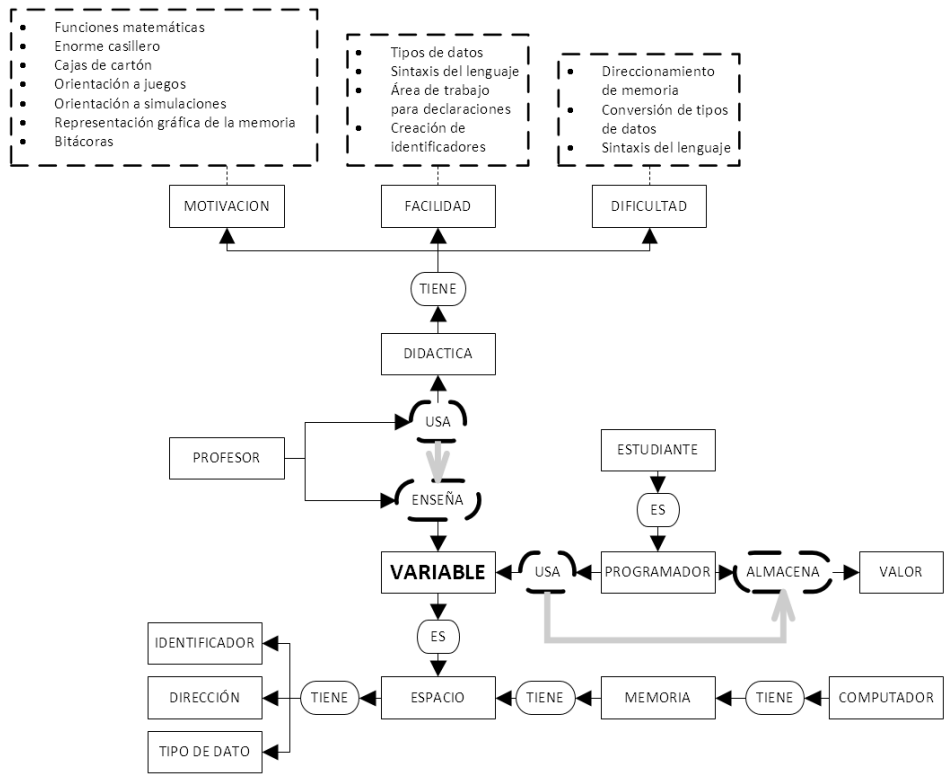


Figura 8. Ontología computacional a través del esquema preconceptual del objeto de saber: "Variable". (Fuente: esta investigación)

La Figura 8 representa la naturaleza del objeto de saber "Variable" a través de su definición, como un espacio de memoria de las computadoras que tiene un identificador, direccionamiento y tipo de dato. Esta variable es usada por los programadores para almacenar valores. En el contexto educativo, el estudiante asume su rol de programador, donde el profesor enseña dicho concepto a través de una didáctica con una serie de elementos motivacionales, y teniendo en cuenta aspectos que facilitan y dificultan su enseñanza.

Conceptualización del objeto de saber: “Constante”

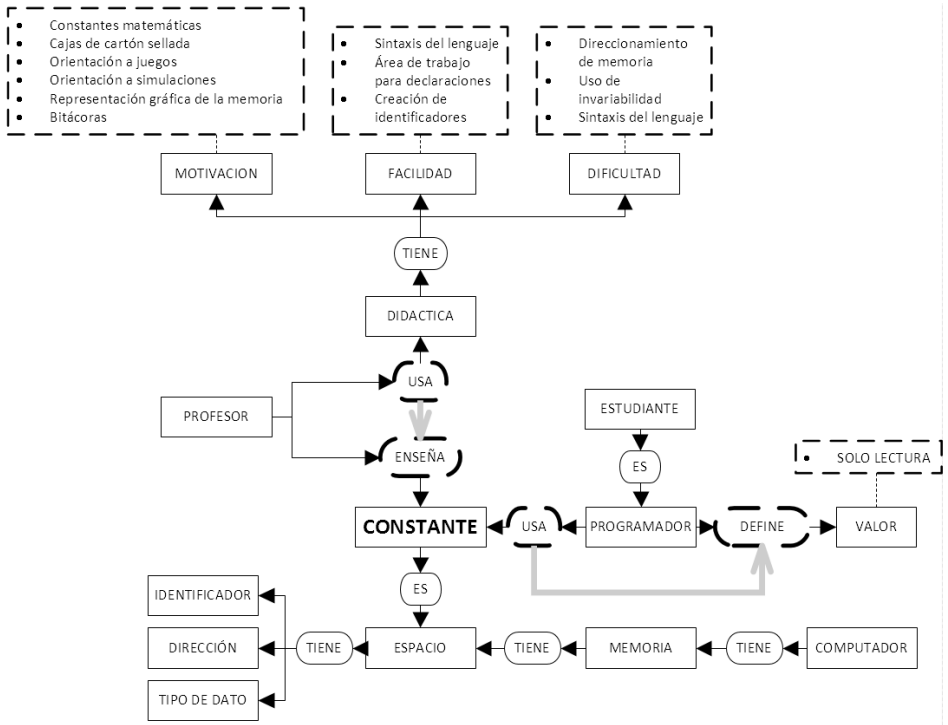


Figura 9. Ontología computacional a través del esquema preconceptual del objeto de saber: “Constante” (Fuente: esta investigación)

La Figura 9 representa la naturaleza del objeto de saber “Constante” a través de su definición, como un espacio de memoria de las computadoras que tiene un identificador, direccionamiento y tipo de dato. Esta constante es usada por los programadores para definir valores de solo lectura, es decir que son inmodificables a lo largo del programa. Al interior de la clase, el estudiante adopta responsabilidades asociadas a la labor de programación, con las actitudes propias de un programador profesional. El concepto asociado de constante se aborda dentro de esa lógica dentro del campo laboral profesional de un programador.

Conceptualización del objeto de saber: “Expresión aritmética-lógica”

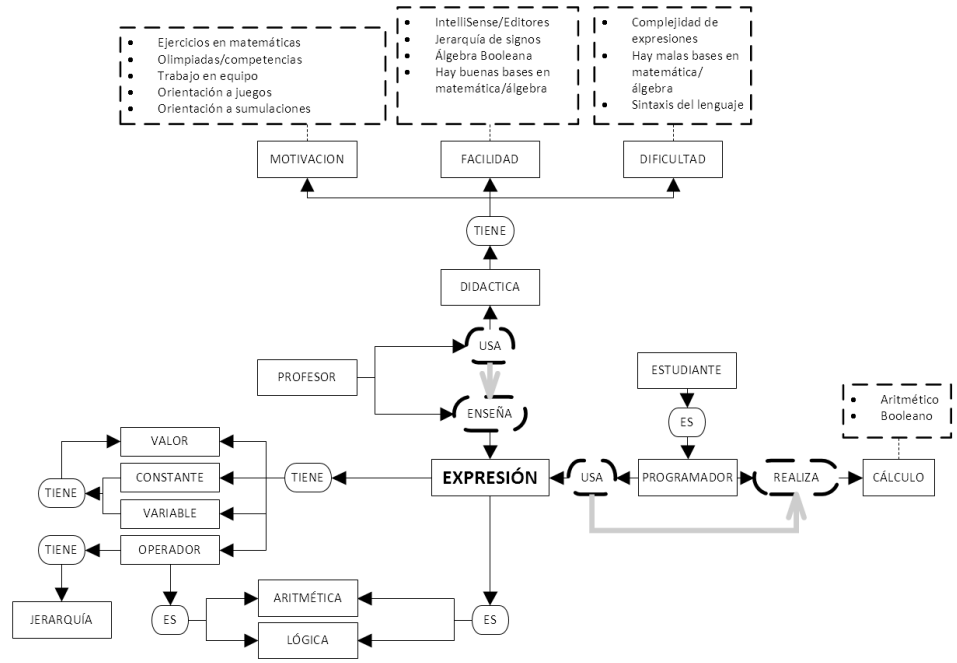


Figura 10. Ontología computacional a través del esquema preconceptual del objeto de saber: Expresión Aritmética-Lógica. (Fuente: esta investigación)

La Figura 10 representa la naturaleza del objeto de saber “Expresión aritmética-lógica” a través de su definición, como un conjunto de valores, variables, constantes y operadores que permiten realizar un cálculo aritmético-lógico. Es importante tener en cuenta que los operadores tienen una jerarquía y su escritura se define según la sintaxis de un lenguaje específico. Al igual que las representaciones conceptuales anteriores, el profesor genera un ambiente de motivación para que sus estudiantes asuman su rol como profesionales en asuntos de codificación de programas computacionales.

Conceptualización del objeto de saber: “Condicional”

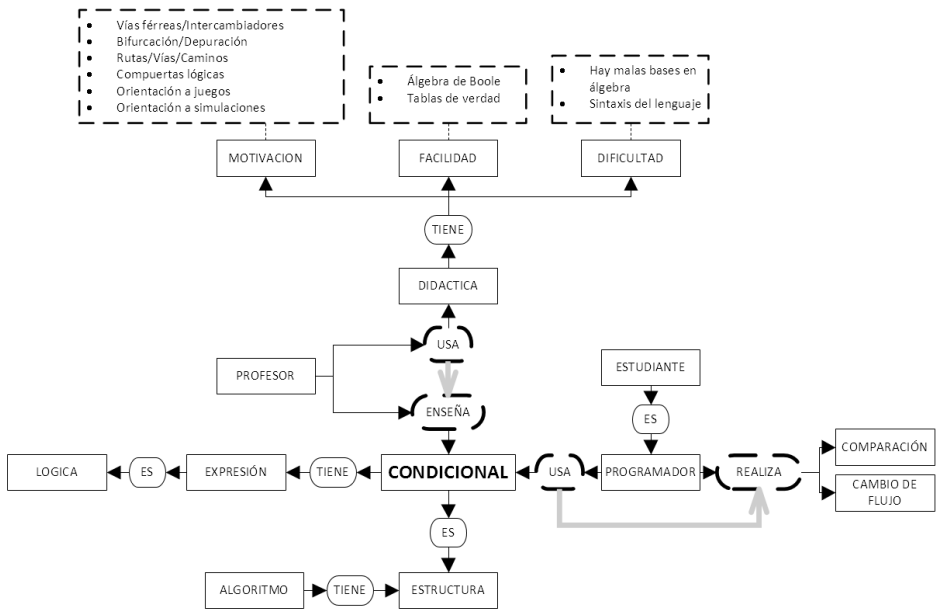


Figura 11. Ontología computacional a través del esquema preconceptual del objeto de saber: “Condicional”. (Fuente: esta investigación)

La Figura 11 representa la naturaleza del objeto de saber “Condicional” a través de su definición, como una estructura algorítmica que permite la bifurcación —cambio de flujo— en la ejecución de un programa a partir de la evaluación de una expresión lógica —comparación–. En el desarrollo teórico, independientemente del lenguaje de programación, es requerida la fundamentación en álgebra Booleana; las expresiones dependen de la sintaxis del lenguaje de programación. A partir de los condicionales, todos los conceptos anteriores son usados. Por lo tanto, exige mayor esfuerzo motivacional por parte del profesor; por su parte, los estudiantes en su rol de programadores asumen situaciones donde una decisión debe tomarse por medio de la lógica Booleana. Así, los estudiantes requieren un conocimiento previo en lógica proposicional.

Conceptualización del objeto de saber: “Ciclo”

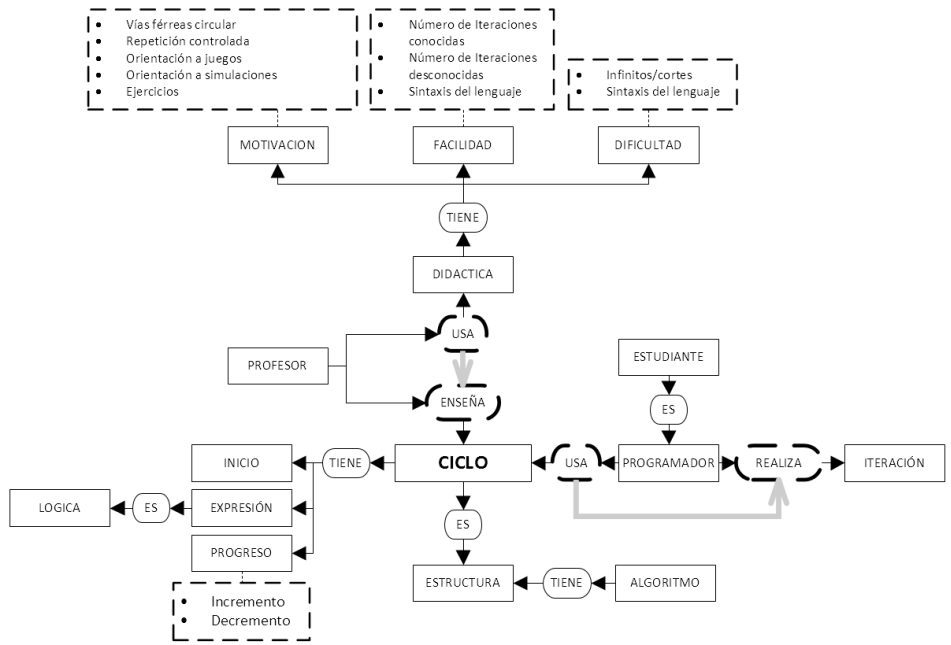


Figura 12. Ontología computacional a través del esquema preconceptual del objeto de saber: “Ciclo”. (Fuente: esta investigación)

La Figura 12 representa la naturaleza del objeto de saber “Ciclo” a través de su definición, como una estructura algorítmica que permite la iteración —repetición— de una o más instrucciones del programa a partir de un punto de partida —inicio—, la evaluación de una expresión lógica —comparación de finalización— y el progreso de éste a través de incrementos o decrementos de la variable de control. Dentro del desarrollo teórico, independientemente del lenguaje de programación, es requerida la fundamentación en álgebra Booleana; las expresiones dependen de la sintaxis del lenguaje de programación. En este punto, la abstracción toma un papel protagónico al momento de diseñar estructuras cíclicas.

Conceptualización del objeto de saber: “Función”

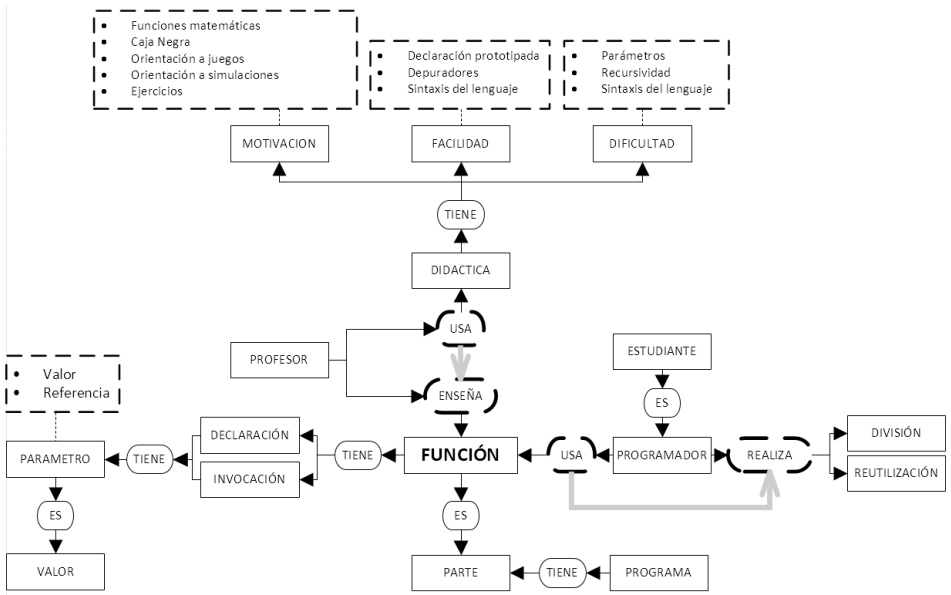


Figura 13. Ontología computacional a través del esquema preconceptual del objeto de saber: “Función” (Fuente: esta investigación)

La Figura 13 representa la naturaleza del objeto de saber “Función” a través de su definición, como una parte del programa que realiza tareas específicas y tiene la facilidad de ser llamado en diferentes instancias dentro del programa; pueden manejar parámetros de entrada y pueden devolver valores de salida. El concepto Función es el de mayor complejidad de los objetos seleccionados en esta investigación, porque involucra la integración de todos los conceptos anteriores. Por lo tanto, varios ejercicios prácticos son necesarios para afianzar su aprendizaje. Es aquí donde los estudiantes en su rol de programadores profesionales explotan su potencial creativo a través del uso de funciones.

5. NIVELES DE TRANSPOSICIÓN DIDÁCTICA

A partir de la caracterización de los objetos de saber, el siguiente paso consiste en determinar los niveles de transposición didáctica que se llevan a cabo en la labor de enseñanza. Así, es necesario apoyarse en la retroalimentación dada por los expertos. Además, gracias a las encuestas y al formato de vigilancia epistemológica se recolecta la información acerca de cómo los profesores que participaron en esta investigación transpusieron los objetos de saber; finalmente, se retoma la teoría original de transposición didáctica para la construcción de una versión extendida que se adapta al contexto propio de la enseñanza en programas académicos relacionados a la computación.

5.1 Oponiones de expertos

La pasantía internacional fue desarrollada en la Universidad de Cádiz —España—, en el Departamento de Ingeniería Informática de la Escuela Superior de Ingeniería. Esta actividad académica permitió el espacio para realizar la primera entrevista semiestructurada, la cual fue aplicada a una profesora adscrita al Departamento de Ingeniería Informática. Dicha entrevista tuvo una duración aproximada de una hora y media, en la cual se permitió el registro de notas sin registro audiovisual. Por otra parte, el segundo experto fue entrevistado en su escenario de labor académica en la Universidad Nacional de Colombia, sede Medellín. La entrevista semiestructurada tuvo una duración de 50 minutos sin registro audiovisual.

En resumen, las entrevistas a los expertos permitieron tener un conocimiento sobre un caso específico de práctica docente. Gracias a la amplia experiencia de los expertos en la enseñanza de los fundamentos de programación, se logró determinar los elementos que, a su juicio, son relevantes en el ejercicio de la docencia de los fundamentos de programación de computadoras. Dada sus opiniones, la práctica en dicha asignatura constituye un factor clave para promover el aprendizaje; esto se logró evidenciar cuando los expertos enfatizaron el uso y el desarrollo de una amplia gama de ejerci-

cios adaptados a los intereses de los estudiantes, con el ánimo de ocupar su tiempo en actividades que les sea provechosas para su aprendizaje.

El diálogo establecido con los expertos en las entrevistas brindó orientaciones adicionales hacia dónde enfocar los asuntos de vigilancia epistemológica, el momento de preguntarles a los profesores que participarían en la investigación. El siguiente punto explica cómo se abordó la indagación sobre aspectos relacionados a la vigilancia epistemológica.

5.2 Acerca de la vigilancia epistemológica

Durante el desarrollo de la presente investigación, se encontraron varias situaciones retadoras. Quizás, una de las situaciones de mayor complejidad a enfrentar está relacionada con la objetividad de los procesos de transposición de los objetos de saber, sin que éstos pierdan su esencia y su rigor otorgados a partir de la naturaleza del saber experto. Es aquí donde la vigilancia epistemológica entra en escena.

Autores como Lerner (2001, p. 52), afirman que es posible "mantener una vigilancia epistemológica que garantice la semejanza fundamental entre lo que se enseña y el objeto o práctica social que se pretende que los alumnos aprendan". En este sentido, se debe entender a la vigilancia epistemológica como el mecanismo para determinar la distancia entre los objetos de saber producidos en el estado del arte de una ciencia o disciplina, y los que se enseña en los ambientes académicos. Según el creador de la teoría de transposición didáctica, la vigilancia epistemológica hace alusión al método de observación que desarrolla el didacta en forma permanente, para garantizar que se supere adecuadamente la distancia que existe entre el saber científico y el saber enseñado (Chevallard, 1985, p. 16).

Se puede afirmar que la vigilancia epistemológica persigue un objetivo principal, que es velar porque el saber enseñado, en lo substancial, no pierda la esencia del saber experto. Con esta idea, la vigilancia epistemológica trata de evitar las deformaciones producidas por la transposición didáctica y garantizar la calidad de la enseñanza (Contreras, 2013, p. 41).

La vigilancia epistemológica implica adoptar una postura cuidadosa al hablar sobre el desarrollo histórico y contextual de los conceptos originales —el saber experto— a fin de comprender su naturaleza. Esta capacidad

debería permitir a los profesores no sólo reconocer simplificaciones excesivas, distorsiones o errores en los conceptos que se han diseñado para ser enseñados, sino también evitar incurrir en ellos durante el desarrollo de las clases en vivo. En el caso particular de la enseñanza de los fundamentos de programación, tal postura evitaría una referencia excesivamente simplificada o incluso tergiversada al razonamiento original de los creadores de dichos conceptos.

Para tal menester, se elaboró un instrumento de recolección de información para determinar elementos constitutivos a partir del ejercicio de vigilancia epistemológica según la visión de los profesores.

A partir del instrumento de recolección de información, una última actividad consistió en la alimentación de una discusión a través de entradas en un foro programado por el investigador. Hay que reconocer que dicha estrategia no tuvo acogida por parte de los profesores, ya que sólo fue realizada la entrada inicial del foro por parte del profesor de la Universidad de Nariño —37.I—. El autor de la investigación interpretó que esta situación ocurrió debido al período de tiempo en el cual fue planeado el desarrollo del foro, dado que los profesores argumentaron que por estar en el final del semestre su tiempo era bastante limitado para dedicarse a otras actividades. Con el fin de no perder la oportunidad de recibir una retroalimentación importante con respecto a la discusión, se optó por enviar nuevamente las entradas del foro vía correo electrónico.

5.3 Extendiendo el concepto de transposición didáctica

Esta investigación, introdujo una visión extendida de la transposición didáctica en la educación de las ciencias de la computación y su fundamento para la construcción de objetos de enseñanza. Esta nueva construcción hizo parte del trabajo elaborado en la estancia de investigación en Cádiz —España—; estancia que fue financiada por la AUIP –Asociación Universitaria Iberoamericana de Postgrado–. El trabajo fue realizado a través de la recolección de información relevante sobre las experiencias de los profesores que imparten dicho curso en la Universidad de Cádiz. El Departamento de Informática e Ingeniería de la Universidad de Cádiz y especialmente el grupo de investigación SPI&FM –*Software Process Improvement and Formal Methods*– formó parte del programa *Open Discovery Space*. En este contexto, los profesores y los estudiantes investigadores trabajaron juntos en el

diseño, construcción, y despliegue de algunos objetos de enseñanza para ser aplicados en Educación Básica; haciendo que la aplicación del proceso de ingeniería de *software* ya implique acciones de transposición didáctica. De acuerdo con los hallazgos, los procesos formales de Ingeniería de *Software* para construir objetos de saber tienen una relación directa con su uso posterior en contextos educativos basados en la teoría de transposición didáctica, situación que posibilita la oportunidad del desarrollo de nuevos aprendizajes en el aula.

Transposición didáctica *In Extensa Sensu*

El término en latín *In Extensa Sensu* fue usado con el propósito de crear una visión extendida del fenómeno en un estadio específico en la educación de las ciencias computacionales. La visión extendida del fenómeno de transposición didáctica en las ciencias computacionales está representada en la Figura 14.

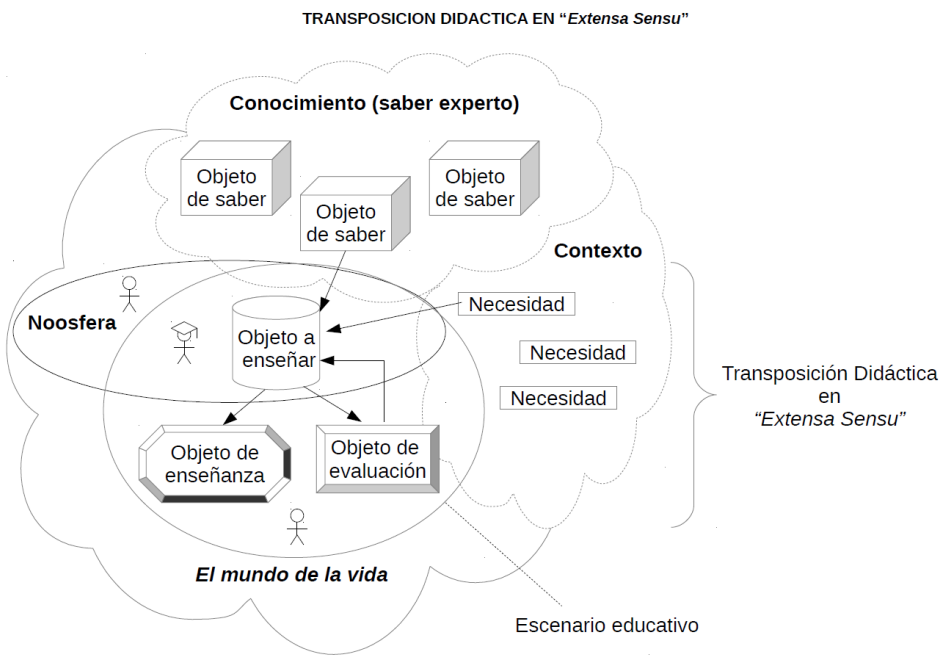


Figura 14. Trasposición Didáctica In Extensa Sensu (Fuente: esta investigación)

De acuerdo con la figura anterior, todo forma parte del mundo de la vida. Este término fue intencionalmente usado en este enfoque debido a la fuerza de su significado. Husserl (1991) fue quien acuñó el término en alemán "*lebenswelt*" desde una perspectiva fenomenológica para indicar las relaciones complejas de subjetividades en el mundo alineadas con la sociedad, la cultura y el individuo. En este contexto complejo, existe una primera "nube" de conocimiento; pero para este caso en particular se habla de una "nube" de conocimiento experto en ciencias de la computación.

En el cuerpo de conocimiento de las ciencias de la computación, hay varios objetos de saber relacionados con temas específicos; para esta investigación el conjunto de objetos de saber que fueron seleccionados tiene relación directa con los fundamentos de programación. Aquí se encuentra el primer paso del fenómeno de transposición didáctica debido a los lineamientos recomendados por ACM.

La primera transposición identificada en este trabajo se llama "Transposición debido a las recomendaciones curriculares de ACM". En esta parte, el fenómeno de transposición didáctica *In Extensa Sensu* inicia con la identificación de los conceptos básicos y en términos generales que pertenecen a temas específicos en las ciencias de la computación. En este sentido, un conjunto de expertos en el campo –especialmente de SIGCSE– han establecido el cuerpo de conocimiento para tal dominio. Hablando del conocimiento acerca de las ciencias de la computación, es notoria la participación de la industria en la construcción de dicho cuerpo de conocimiento; en el documento de recomendaciones curriculares para las ciencias computacionales, el equipo de trabajo que las propusieron dice:

> *Perspectivas industriales: Como era de esperar, la retroalimentación de la industria era muy diferente, pero invariablemente, se proporciona de buena gana —incluso con entusiasmo— y con un profundo sentido de convicción. Se tendía a confirmar la importancia de reclutar a estudiantes de alta calidad que luego tenía que estar en sintonía con una buena ética de trabajo y recibir educación sólida en los fundamentos de la materia* (*ACM & IEEE Computer Society*, 2008, p. 11).

Con esto en mente, lo que es recomendado por ACM tiene una estrecha relación con la intencionalidad de responder a las preguntas: ¿Qué enseñar?, y ¿Para qué enseñar? Desafortunadamente, hay algunos aspectos que se pierden en este proceso, por ejemplo: las respuestas a las preguntas, ¿Cómo lo hizo —el autor/creador original–?, y ¿Por qué lo hizo —el autor/creador original–? Parecen menos relevantes que las preguntas iniciales. Al pare-

cer, las primeras preguntas —¿Qué? y ¿Para qué? — son “más útiles” en estos tiempos de acuerdo con el modelo hegemónico económico. El primer paso de la transposición didáctica *In Extensa Sensu* se puede observar en la Figura 15.

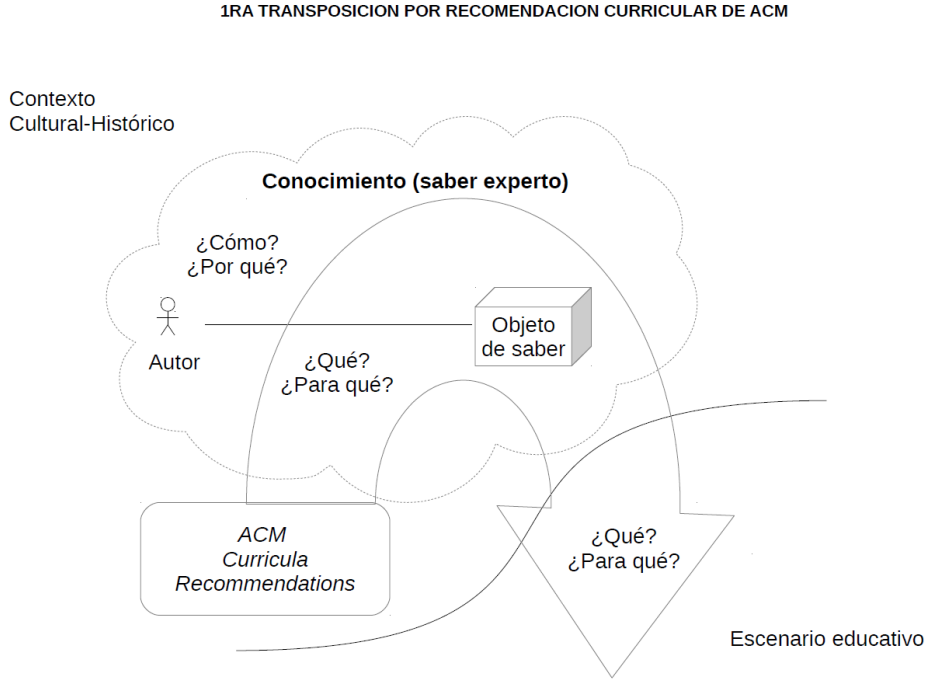


Figura 15. Primera transposición por recomendaciones curriculares de ACM (Fuente: esta investigación)

¿Para quién se desarrolla el conocimiento desde un punto de vista utilitario? ¿Está hecho sólo para las clases hegemónicas, las clases sociales que tienen el control de las grandes compañías del sector productivo, o para aquellos personajes que “gobiernan” políticamente el planeta? El conocimiento tiene una motivación altruista, desde la humanidad y para la humanidad. En este estadio, el fin último parece ser el desarrollo del conocimiento para el bien del mundo; no obstante, la sombra utilitaria cubre con su manto el escenario educativo. En dicho escenario, el contenido “más útil” es implantado; sin embargo, hay algunos aspectos perdidos sobre el contexto histórico-cultural donde el autor/creador de dicho objeto de saber estuvo inmerso.

Esta investigación abordó 6 objetos de saber dados a partir del conocimiento experto o erudito. En este sentido, fue de gran interés analizar su naturaleza, sus orígenes, es decir, el por qué fueron creados dichos objetos de saber. Este conocimiento fue importante tenerlo en cuenta, mucho más allá de su utilidad instrumental. El estudio de su naturaleza otorga valor adicional a lo que los estudiantes aprenden sobre los objetos de saber nombrados El conocimiento sobre cómo el autor creó un objeto de saber y por qué lo hizo, más allá de su utilidad instrumental, es relevante de acuerdo con los hallazgos descritos en este capítulo. Dicho conocimiento podría promover nuevos aprendizajes en los estudiantes porque es posible explorar formas alternativas para “reconstruir” los objetos de saber en sí. Algunas luces sobre este asunto se presentarán a continuación.

Cuando un objeto de saber es seleccionado para propósitos de enseñanza, la segunda transposición toma lugar en el camino hacia los ambientes educativos. Este segundo paso es llamado “Transposición debido a la noosfera”, y está representado en la Figura 16.

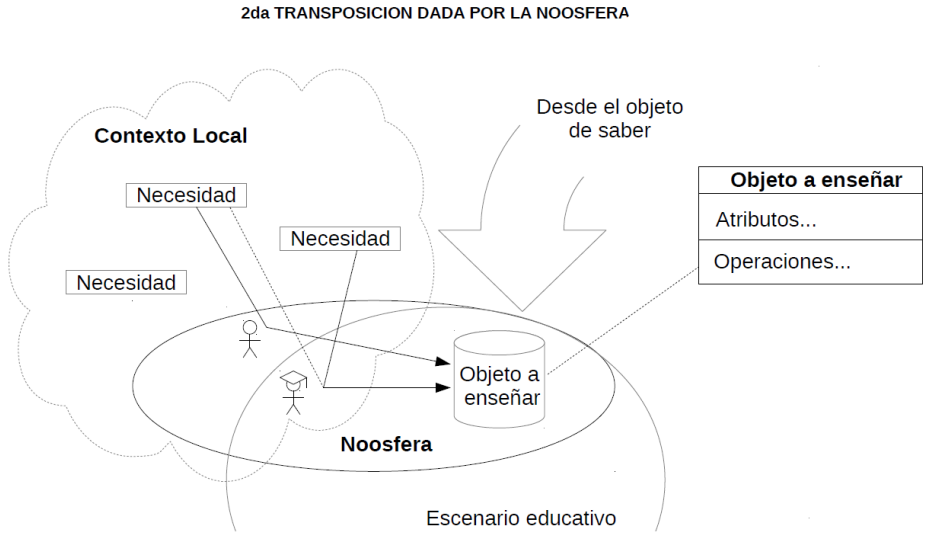


Figura 16. Segunda transposición debido a la noosfera (Fuente: esta investigación)

Aquí, el concepto de noosfera toma protagonismo. De acuerdo con Chevallard (1985, p. 28) la noosfera es la “esfera” donde las prácticas educativas son pensadas y diseñadas. Dicha “esfera” o escenario está compuesto por personas a cargo de diseñar e implementar planes de estudio para contextos específicos —tales como personal administrativo, directores de departamentos, profesores, pedagogos, consultores de la industria y empleados, entre otros–. A partir de la noosfera, sus integrantes toman decisiones al

seleccionar los objetos de saber con el fin de transformarlos en objetos factibles de enseñarse; y a la vez, captura las necesidades del contexto. En este punto, un nuevo objeto es creado y es derivado del objeto de saber; no obstante, es un objeto totalmente nuevo –llamado objeto a enseñar– el cual es diferente a su predecesor. Un objeto a enseñar tiene atributos y operaciones que lo hacen posible de ser enseñado. En este sentido, un objeto a enseñar contempla características especiales de acuerdo con el modelo de aprendizaje —tales como aprendizaje activo, aprendizaje basado en problemas, modelo basado en competencias, etc.—, y fue diseñado específicamente para responder a necesidades del contexto —esta es la idea principal—; sin embargo, las necesidades pueden variar de acuerdo con la perspectiva del individuo; por ejemplo: hablando de subjetividades, los intereses personales de quien enseña podrían ser diferentes a los de aquellos que trabajan en la industria y el sector productivo. Este nuevo objeto ha definido claramente sus metas educativas debido a las orientaciones de la noosfera. Así, la esencia de la segunda transposición se basa en las subjetividades de las personas en la noosfera; de hecho, esta subjetividad ya está presente al momento de diseñar los objetos a enseñar.

En esta parte de las situaciones que experimenta el conocimiento, después de que un objeto a enseñar es diseñado, un nuevo objeto aparece en escena: el objeto de enseñanza; el cual constituye la tercera transposición que se puede observar a través de la Figura 17.

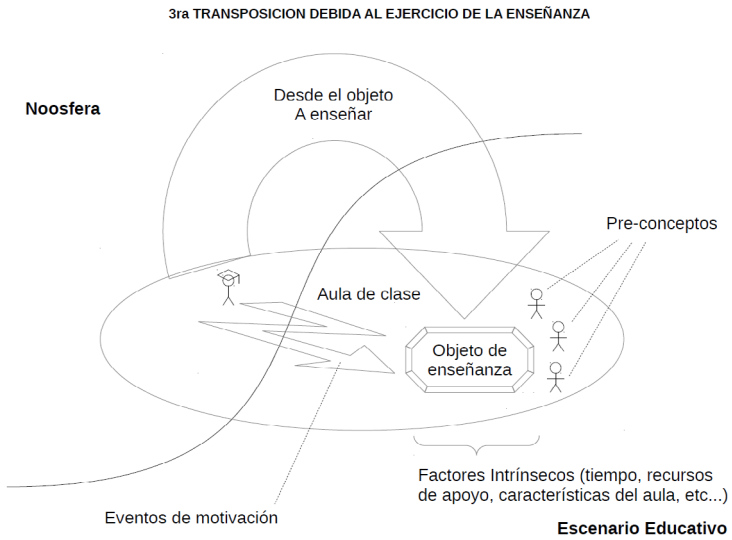


Figura 17. Tercera transposición debido a la práctica de enseñanza
(Fuente: esta investigación)

Como el paso anterior, el objeto de enseñanza es un objeto diferente al objeto predecesor, ambos están obviamente relacionados, pero son diferentes; esta característica es la esencia de la transposición didáctica.

La tercera transposición involucra al objeto a enseñar dentro de un escenario práctico: el aula de clase. En las aulas de clase, las prácticas de enseñanza son transformadas en situaciones reales basadas en diseños teóricos previos. Pero este encuentro con la realidad conlleva factores circunstanciales tales como, manejo del tiempo, recursos disponibles, características propias del espacio físico —iluminación, acústica, ventilación, higiene, condiciones climáticas, etc.—, entre otros. En consecuencia, la falta o la deficiencia de estos factores podrían afectar el aprendizaje en los estudiantes.
Por otro lado, los estudiantes tienen preconceptos antes de asistir a un curso de fundamentos de programación. Estos preconceptos se asocian por lo general a formaciones previas en ciertos lenguajes de programación. Una cita conocida de Friedman y Koffman (1978) dice acerca de la dificultad de enseñar fundamentos de programación a estudiantes que han tenido cierta experiencia previa con algunos lenguajes de programación, dadas sus características particulares. Estos preconceptos están asociados al conocimiento previo en material de computación, y especialmente sobre programación de computadoras. En algunos casos, un grupo de estudiantes podría tener un conocimiento previo acerca de cómo crear programas de computadoras; este es el caso de quienes reciben algún tipo de formación relacionada a los fundamentos de las ciencias computacionales en Educación Básica, Primaria y Secundaria, más allá de la simple instrucción ofimática o de paquetes informáticos en específico. En este sentido, las habilidades personales están fuertemente relacionadas con el aprendizaje de los estudiantes; de hecho, una fundamentación teórica sobre el desarrollo de las habilidades estudiantiles fue propuesta por Vygotsky (1987) durante sus reflexiones sobre la zona de desarrollo próximo. En esta teoría, fue declarada la diferencia entre el desempeño de los aprendizajes debido a la presencia o ausencia de apoyo externo. Por ello, las competencias personales son factores clave a fin de determinar la habilidad de solución de problemas. De esta forma, la noción previa sobre el conocimiento y las habilidades de los estudiantes, se convertirán en un insumo para el desarrollo de las prácticas de enseñanza.

Todos estos elementos anteriores deberían estar integrados en la instancia de un objeto de enseñanza. En este punto, el trabajo real en el aula repre-

senta el tercer nivel del fenómeno de Transposición Didáctica porque las integraciones de nuevos elementos producen cambios en el objeto inicial —el objeto a enseñar— que fue seleccionado para ser enseñado. Dado que el objeto de enseñanza es diferente al objeto a enseñar, hay algunos elementos que se pierden en dicha transición. De hecho, la sola interacción entre quien enseña y quienes aprenden, implica que algo pueda perderse por el principio de la imperfección del acto comunicativo, el cual se manifiesta de forma consciente o inconsciente. Hay teorías al respecto de la imperfección del acto comunicativo, las cuales explican por qué un mensaje de quien enseña puede ser malentendido por quienes reciben dicho mensaje; por ejemplo, Kinsella y Marcus (2009, p. 197) encontraron que hay algunas "fallas" en el lenguaje que lo hace imperfecto de acuerdo con la evidencia empírica; algunas de ellas incluyen: ambigüedad léxica, ambigüedad sintáctica, irregularidad morfológica, errores extra-gramaticales, redundancia morfológica, movimiento, y condiciones de localidad, entre otras. Con esto en mente, el aprendizaje del estudiante se afecta por este comportamiento dinámico del objeto de enseñanza debido al fenómeno de transposición en el mundo real.

Finalmente, la última transposición es presentada en la Figura 18; llamada "Transposición debido a la evaluación y la retroalimentación", y ha sido definida para crear espacios de seguimiento a los aprendizajes de los estudiantes. En esta situación, un nuevo objeto es creado y depende de la imagen transpuesta del objeto de enseñanza, es decir, se trata del objeto de evaluación.

El objeto de evaluación es el último objeto transpuesto en este enfoque. Este objeto se basa en un conjunto de instrumentos que permiten valorar el aprendizaje de los estudiantes, más allá de cuantificarlo en una calificación. Debido a la naturaleza de este objeto, la esencia de la transposición en este punto se refleja en la construcción de recursos de valoración a fin de cualificar los logros del estudiante en relación con el aprendizaje esperado; aquí aparecen elementos de subjetividad que deberían ser cuidadosamente direccionados.

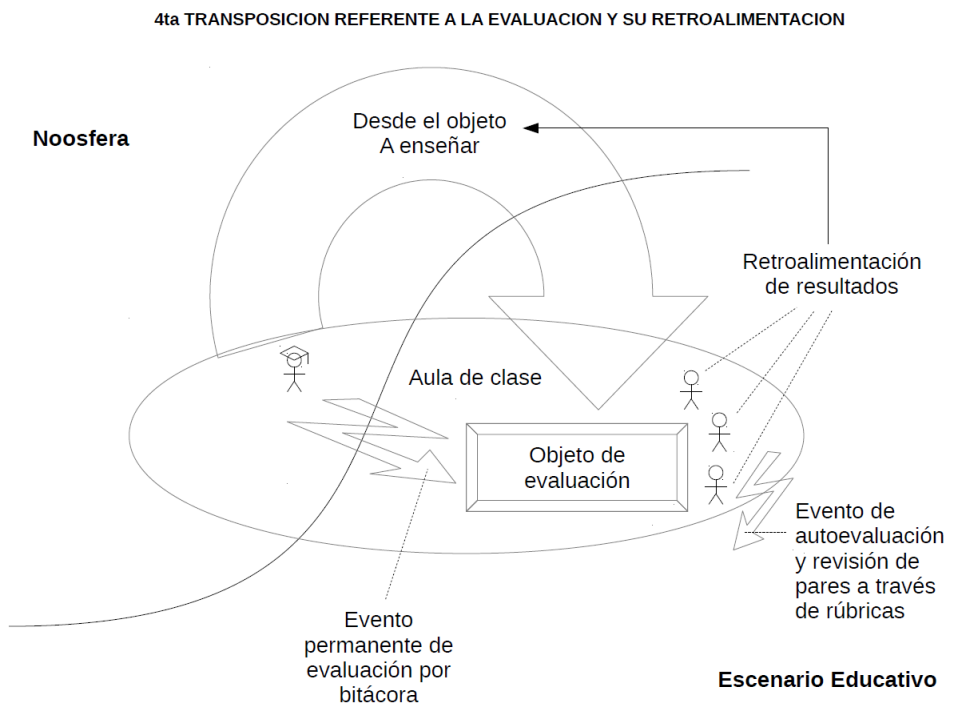


Figura 18. Cuarta transposición debido a la evaluación y la retroalimentación (Fuente: esta investigación)

Un punto importante para resaltar aquí son los niveles de responsabilidad en las tareas de autoevaluación que los estudiantes deberán enfrentar. Esto implica que las acciones que promueven una cultura de autoevaluación y mejoramiento continuo deberán ser trabajadas en las aulas de clase. Esta no es una labor fácil; de hecho, se requiere que los estudiantes tengan un conocimiento profundo sobre sus propias capacidades dentro del aula de clase, que pueda ser articulado en sus proyectos de vida. Se está hablando de una cultura que permita, con determinación, valorar nuestras mismas potencialidades y oportunidades de mejoramiento. He ahí la razón por la cual los eventos motivacionales en el objeto anterior deben estar fuertemente acoplados en la práctica de la enseñanza.

Así como el objeto de enseñanza, el objeto de evaluación tiene sus propios eventos. En este caso hay eventos de autoevaluación y revisión por pares a través de rúbricas, dado que algunos estudios empíricos en el campo de la enseñanza de los fundamentos de programación recomiendan tanto el trabajo individual como el trabajo en parejas dentro y fuera del aula de clase —tareas, desarrollo de proyectos, análisis de lecturas, trabajos de consulta,

etc.–. Un estudio interesante sobre los beneficios de la programación en parejas con el fin de lograr habilidades de programación individual reveló que los estudiantes sienten más confianza en su trabajo al haber usado esas técnicas de aprendizaje (Braught, Wahls & Marlin, 2011).

Al cambiar el punto de vista, la persona quien enseña tiene la responsabilidad de ejecutar una acción permanente sobre el proceso de valoración. Esto se recomienda dado que la evaluación del aprendizaje no debería verse como una colección de eventos valorativos aislados —un *quiz*, un examen, un cuestionario, una tarea, etc.—; en esta línea de pensamiento, la valoración implica un proceso holístico, integral y continuo que contempla varios aspectos de la formación del ser humano relacionados con el aprendizaje disciplinar especifico. Dado esto, factores humanos están siempre presentes en el proceso de valoración en relación con las actitudes de los estudiantes. Quizás este es el objeto más complejo y desafiante para ser diseñado ya que el proceso de valoración tiene un alto grado de subjetividad. A fin de cuentas, la retroalimentación basada en los logros de los estudiantes es un nuevo insumo para afianzar y mejorar el diseño de los objetos a enseñar; y esto hace parte del dinámico mundo educativo.

Experiencias a través del proyecto *Open Discovery Space*

Rushkoff (2010) plantea el siguiente interrogante ¿Estamos educando a los ciudadanos y profesionales del futuro para saber cómo manejar la tecnología, o para ser manejados por la tecnología? La respuesta a esta pregunta pasa por decidir hasta qué punto es necesaria la generalización de las habilidades de informática para todos los niveles de la educación, con el fin de ser capaz de programar en lugar de ser programado. La generalización de las ciencias computacionales y las habilidades en informática está apuntada por Wing (2006) como pensamiento computacional. Pensar como un científico en computación significa algo más que programar una computadora.

El enfoque llamado Transposición Didáctica *In Extensa Sensu* es una oportunidad metodológica para desarrollar experiencias educativas de pensamiento computacional entre los docentes y los alumnos que no están familiarizados con los temas de computación y en especial de programación de computadoras en su práctica docente cotidiana. Estas experiencias están siendo llevadas a cabo dentro del Proyecto ODS, que tiene por objeto el desarrollo de una infraestructura de aprendizaje abierto, socialmente, para potenciar la adopción de los recursos de *e-learning*. La Universidad de Cádiz coordina las comunidades en España de ODS, formadas por un selecto grupo de profesores de Educación Básica; a pesar de ser lo suficientemente maduros en el uso de las tecnologías informáticas para el aprendizaje, no están familiarizados con la enseñanza de las disciplinas de Ciencias de la Computación, la programación de computadoras y, sobre todo, acerca de la formación del pensamiento computacional. Las comunidades ODS de España usan recursos educativos que están relacionados con la computación y la educación de las ciencias computacionales —por ejemplo: *csunplugged.org* y *cse4k12.org*—, que son en su mayoría recursos apropiados para experimentar Transposición Didáctica.

Experiencias en Educación Superior

Dado el escenario para analizar el fenómeno de la transposición didáctica con el enfoque *In Extensa Sensu*, los profesores encargados de enseñar fundamentos de programación de los cursos de la Universidad de Cádiz fueron consultados con el fin de articular la construcción teórica de la investigación con los contextos reales. A través de las entrevistas, se reconoció que las recomendaciones curriculares de ACM son los pilares en la construcción de los planes de estudio de una carrera profesional asociada a las disciplinas de las ciencias computacionales. Se definió claramente cómo se podría formar la noosfera con la participación de profesionales de la industria; sin embargo, esta participación siempre está unida a la creación de contratos explícitos y acuerdos formales entre la universidad y la industria. En este sentido, la experiencia vivida en la Universidad de Cádiz permitió identificar el papel relevante del Coordinador de Carrera, quien está a cargo del diseño de la estructura principal de contenidos —diseño de planes de estudio— con el apoyo del grupo de profesores del Departamento de Informática e Ingeniería. Esta construcción de contenidos se basa principalmente en las recomendaciones de ACM, de algunas referencias locales —de España y de la Comunidad Autónoma de Andalucía—, y de algunas directrices de la Comunidad Europea. Además, la opinión de la industria es muy valorada; sin embargo, los profesores encargados de fundamentos de programación tienen la “última palabra” con el Coordinador de Carrera en la construcción de los contenidos de los cursos, teniendo en cuenta las aportaciones de la industria.

Esta es la primera etapa para el fenómeno de la Transposición Didáctica *In Extensa Sensu*, porque se seleccionan prácticamente todos los objetos de saber relacionados con los fundamentos de programación, el cual es enseñado sin antecedentes históricos. A través de las entrevistas, los profesores argumentaron acerca de sus intenciones sobre la enseñanza de algunos aspectos históricos alrededor de los objetos de saber; pero, por desgracia, el tiempo no es suficiente para cubrir esos temas. En este camino, la segunda etapa del fenómeno de la transposición didáctica se produce en la creación de los objetos a enseñar. Estos objetos tienen características intrínsecas debido al contexto académico en el que han sido diseñados; por ejemplo, las características intrínsecas incluyen atributos y operaciones. Algunos atributos pueden tener una estructura basada en competencias —incluyendo habilidades, resultados del aprendizaje, y los indicadores de rendimiento académico, entre otros–. También algunas operaciones intrínsecas incluyen horarios de clases, programación de las actividades de clase, lecturas complementarias y recursos educativos.

Dentro del aula de clase, los objetos a enseñar se convierten en objetos de enseñanza. En este sentido, los profesores entrevistados dijeron que un punto clave para mantener la atención de los estudiantes se basa en la definición de un "hilo conductor" para toda la clase. Tal "hilo" podría ser una historia, un desafío, un problema del contexto local, etcétera. Por lo tanto, los profesores entrevistados informaron que a través de un "hilo conductor" se puede establecer un sentido o propósito de la temática que los estudiantes están aprendiendo en el salón de clases; sin embargo, la preparación de cada sesión es una tarea que requiere bastante tiempo. Este es el momento para la creación de eventos de motivación que complementan la definición de un objeto de enseñanza; pero la definición de esos agentes motivacionales implica una gran experiencia en el campo educativo sobre lo que se está enseñando. Según los profesores entrevistados, otro punto importante a este nivel es la conciencia de quien está enseñando sobre los preconceptos manejados por los estudiantes. Con esto en mente, el conocimiento previo que tienen los estudiantes representa un impacto directo en cómo se debe conducir la sesión de clase; para el caso específico de los estudiantes de la Universidad de Cádiz, el grado de conocimiento previo de los que asisten a fundamentos de programación es prácticamente homogéneo.

Por último, los profesores entrevistados coinciden en que una de las mejores estrategias de evaluación es a través del trabajo en parejas —programación en pares—, y el trabajo en equipo. En este sentido, las estrategias de evaluación, tales como la autoevaluación y la coevaluación se utilizan a menudo para hacer la evaluación formativa y sostenible; esto se hizo en una experiencia interesante con una metodología de evaluación orientada al aprendizaje mediante el uso de *Open Data Framework* (Balderas *et al.*, 2013). De acuerdo con las entrevistas, el uso de la plataforma *Web* ha sido un éxito; de esta manera, la comunicación entre profesores y estudiantes es eficiente. A partir de este escenario, el proceso de valoración del aprendizaje de los estudiantes a través de las tareas, las pruebas, los cuestionarios y los proyectos tiene un comportamiento similar a un *Blog* de eventos continuo.

Por otra parte, hay muchas similitudes entre el proceso de enseñanza de los cursos básicos sobre la programación y la construcción de objetos de enseñanza —enmarcado dentro del proyecto ODS— desde un punto de vista de la transposición didáctica *in Extensa Sensu*. Al dialogar con el Ingeniero Informático —título en España que equivale al título de Ingeniero de Sistemas en Colombia— responsable del desarrollo de los proyectos enmarcados en ODS para la comunidad de las escuelas primarias y secundarias en España, estaba claro que la construcción de los objetos de enseñanza para las ODS implica procesos de *software* formal, en especial el uso de Metodologías Ágiles. Siguiendo un proceso de *software* para la construcción de objetos de enseñanza, implica la identificación y análisis de los requisitos de acuerdo con las necesidades de los potenciales usuarios. Esta etapa consiste en un fenómeno de transposición didáctica en la Educación Básica, ya que se parte del punto de vista del profesor que quiere enseñar algo y el punto de vista del programador de la aplicación, que está tratando de "materializar" la visión del profesor. Este diálogo entre ambos actores —el profesor y el programador— genera la idea de un objeto de enseñanza que es diferente del objeto de saber original. En este punto, hay algunos aspectos del objeto de saber original, que se pierden en esta transposición, según el contexto histórico-social y algunos detalles científicos que no son apropiadas, dado el nivel de la educación. Así se concluye que la construcción de objetos de enseñanza para las ODS no es una tarea fácil; de hecho, es muy necesaria la participación de profesores y expertos en pedagogía con el fin de producir material adecuado con fines educativos, especialmente, para la Educación Básica. En este contexto, la teoría original de transposición didáctica se puede aplicar para ODS.

En su momento, se ejecuta la segunda fase de ODS que tiene como objetivo hacer un despliegue completo para probar y evaluar las experiencias en algunas escuelas primarias y secundarias en España. Actualmente, no hay datos al respecto, ya que esta fase está en desarrollo. A pesar de esta situación, el fenómeno de la transposición didáctica se identificó claramente con grandes oportunidades para mejorar la construcción de objetos de enseñanza desde un punto de vista pedagógico.

6. ESTRATEGIAS DE ENSEÑANZA

Este capítulo recoge las experiencias en el campo de la enseñanza de los fundamentos de programación, y elabora una propuesta de prácticas en la labor de enseñanza a partir de la transposición didáctica en un sentido ampliado.

6.1 Posición profesoral frente a la práctica

Los 6 profesores que participaron en el desarrollo de la guía epistemológica manifestaron su colaboración con el desarrollo de una entrevista semi-estructurada. Cabe resaltar que aquellos profesores extranjeros fueron entrevistados a través de una sesión de videoconferencia de máximo 10 minutos, a través del servicio *Skype*® de *Microsoft Corporation*. A diferencia de los extranjeros, el profesor de la Universidad de Nariño fue entrevistado en forma presencial. Todos los profesores tenían conocimiento previo del cuestionario de 9 preguntas que se iba a desarrollar en la entrevista. Para tal entrevista, se utilizó el mismo cuestionario propuesto para las entrevistas a expertos; la diferencia consistió en la profundidad de las respuestas, dado que los 2 expertos entrevistados disponían del tiempo suficiente para profundizar en las respuestas a través del diálogo. Por su parte los profesores no contaban con el tiempo suficiente para entrar en detalles.

Adicionalmente, los profesores entrevistados manifestaron que no querían ser grabados. Esto obligó a la realización de una toma de notas de oraciones claves que constituyen el cuerpo de cada respuesta.

6.2 Análisis hermenéutico concluyente

Dada la síntesis de respuestas de las entrevistas a los profesores y teniendo en cuenta la intencionalidad del instrumento, los puntos focales de esta investigación en cuanto a la práctica de la enseñanza de los fundamentos de programación son 8 y se describen en la Tabla 10.

Tabla 10. Puntos focales en la práctica de la enseñanza de los fundamentos de programación

#	Pregunta intencionada	Punto focal
1	*How long have you taught the courses related to programming fundamentals?*	Experiencia en enseñanza
2	*Do you use analogies to explain the concepts about variables, constants, expressions, conditions, loops, and functions? If so, please describe them.*	Lenguaje en contexto
3	*According to your teaching experience, which are the most difficult topics to teach in the programming fundamentals?*	Reto de enseñanza
4	According to your teaching experience, which are the most frequent problems of students' learning?	Reto de aprendizaje
5	*Do you consider that the textbooks used by your students are suitable for their learning? Why?*	Referencias bibliográficas
6	*Do you use motivational strategies? If so, please describe them.*	Eventos motivacionales
7	*How do you assess the students' learning?*	Valoración de aprendizaje
8	*Please, describe the kind of students' work that they do out of the classroom (i.e. homework, projects, etc.)*	Trabajo independiente

Fuente: esta investigación

Experiencia en enseñanza

En aras de proponer prácticas de enseñanza apropiadas para los fundamentos de programación, la experiencia es considerada como factor clave. No obstante, esta experiencia sugiere un permanente ejercicio de reflexión sobre las prácticas que se están llevando a cabo. Esta reflexión la debe hacer cada profesor en respuesta a una actitud investigadora sobre su propio desempeño en las labores de docencia.

En los diálogos internos que se generaban durante la entrevista semi-estructurada, la mayoría de los profesores manifestaron que los ejercicios de autorreflexión sobre cómo se enseña son escasos, algunos de ellos argumentaban la falta de tiempo para realizar dichas reflexiones. Sin embargo, dos de ellos declararon que es una norma de sus departamentos establecer encuentros entre profesores para gestar los espacios de reflexión sobre sus prácticas de enseñanza; esto suele hacerse al final del año académico, según lo dicho por los profesores.

Ante esta realidad, la pregunta que surgió fue: ¿Los profesores hacen reflexión sobre su labor de enseñanza? En esta recopilación de información,

uno de ellos manifestó que son escasos los ejercicios de reflexión. Tal docente afirmó que, su reflexión simplemente se sustenta en una revisión al final del curso sobre aquellos puntos que consideró que no se expusieron bien al momento de la clase. Quizás dicha acción sea un primer paso -al menos se hace eso-. Pero quizás, más allá de revisar "lo que no salió bien", la reflexión sobre la práctica de la enseñanza apunta hacia escenarios de mayor profundidad epistemológica. Con esto se puede inferir que las competencias de los docentes universitarios no se deben limitar a "saber enseñar", sino que se propone una actitud investigativa y reflexiva sobre su propia labor. Para Zabalza (2003), las competencias que deberían tener los docentes universitarios son: planificar el proceso de enseñanza-aprendizaje; seleccionar y preparar los contenidos disciplinares; ofrecer información y explicaciones comprensibles y bien organizadas —competencia comunicativa—; manejar las nuevas tecnologías; diseñar la metodología y organizar las actividades —organización del espacio, selección del método, desarrollo de las tareas instructivas—; comunicarse-relacionarse con los alumnos; tutorizar, evaluar, reflexionar e investigar sobre la enseñanza, identificarse con la institución y trabajar en equipo.

A manera de reflexión: en el campo de las ciencias de la educación ¿De qué sirve tener una cantidad de años de experiencia enseñando si no se ha hecho un ejercicio de reflexión sobre las propias prácticas? Con esto, la experiencia en enseñanza es muy importante en cuanto sea sustentada en la reflexión sobre las prácticas docentes.

Lenguaje en contexto

A los profesores entrevistados se les preguntó sobre el uso de analogías al momento de enseñar algunos objetos de saber propios de los fundamentos de programación de computadoras; ante esto, la mayoría manifiesta que las analogías se explicitan a través de ejemplos de la vida real. En materia de fundamentos de programación, el "puente" que une al mundo computacional con el mundo real es a través de la matemática, y por ella se unen prácticamente todas las áreas de ciencia básica, como la física, la química, la biología, etcétera.

Por ser la matemática un lenguaje abstracto, se requiere manejar instancias al momento de enseñar los fundamentos de programación para que

el lenguaje sea fácilmente apropiado por los estudiantes. Quizás el recurso más apropiado para ello es la analogía.

La analogía es una comparación entre nociones –conceptos, principios, leyes, fenómenos, etc.– que mantienen una cierta semejanza entre sí. Constituyen un recurso frecuente en el contexto escolar, cuando el profesor, por ejemplo, pretende hacer más comprensible una idea compleja y utiliza, para ello, otra que resulta más conocida y familiar para el alumno. La noción o sistema que se quiere aclarar se denomina objeto o blanco, según los autores, mientras que el que se utiliza como referencia se denomina análogo, ancla o fuente. El uso de analogías aparece ligado, de una parte, al aprendizaje en el ámbito conceptual, por ejemplo, como ayuda en la compresión y desarrollo de nociones abstractas o como recurso dirigido a cambiar las ideas intuitivas ya existentes (Oliva, 2006).

No obstante, el uso de analogías debe ser llevado a cabo con sumo cuidado. El hecho de usarlas ya implica implícitamente un ejercicio complejo de transposición didáctica, dado que son lenguajes que se empiezan a adaptar con un fin didáctico.

El lenguaje en contexto apunta hacia el ejercicio de transposición didáctica que parte de un objeto de saber propio de los fundamentos de programación, y que será transpuesto en un objeto a enseñar, que posteriormente dentro del aula de clase será transpuesto en un objeto de enseñanza. La analogía forma parte importante de la tercera transposición; por ejemplo: al momento de enseñar los conceptos de variables, es posible representar cada posición de memoria a través del uso de un gran casillero con ubicaciones de contenido independiente; de esta forma, cada casilla sólo podrá mantener un valor en un determinado instante y dichas casillas podrán cambiar sus valores con el paso del tiempo. Con este ejemplo, se aclara la relación entre la analogía y su ubicación en la tercera transposición dada la acción que se lleva a cabo en el aula de clase.

Reto de enseñanza

Es común encontrar en el profesorado una sensación de confianza frente a los contenidos de fundamentos de programación. Se da por hecho que el profesor en general tiene un dominio suficiente sobre dichas temáticas dada su condición. Dada esta situación, a los profesores entrevistados se les

realizó el siguiente cuestionamiento derivado de la tercera pregunta que estaba programada en dicha entrevista: *With the simple fact of knowing a discipline, ¿is it enough to be able to teach it?* —Con el simple hecho de conocer una disciplina, ¿Es suficiente para poder enseñarla? –. Este interrogante generó un diálogo reflexivo con los profesores entrevistados. Fue muy interesante notar que, de forma unánime, los profesores reconocieron que se requieren competencias adicionales a las cognitivas en la disciplina que permita facultar a una persona para enseñarla, se habló puntualmente sobre las competencias pedagógicas y didácticas.

Si bien es cierto que los profesores reconocieron la necesidad de tener una formación básica en pedagogía[5]. Algunos de ellos restaron importancia a dicha formación, argumentando que puede ser fácilmente lograda a través de un curso de corta duración. Dado esto, se evidencia posiciones marcadas en cuanto a la importancia que le otorgan a las competencias pedagógicas y didácticas del profesorado en la Educación Superior; para algunos, estas competencias no tienen gran peso. En lo que a esta investigación se refiere, la importancia de una formación en didáctica es superlativa. Retomando la información recolectada, se logró evidenciar que, en términos generales, no se reconoce que haya algo difícil de enseñar dada la naturaleza del objeto de saber; algunos profesores afirman que suele haber dificultades al enseñar ciertos conceptos dada la insuficiente formación previa en los estudiantes. Esto es, se está atribuyendo la dificultad de enseñar las cosas en función de las deficiencias en aprendizajes previos de los estudiantes. Esta situación hace que se desligue la complejidad de enseñar las cosas por las condiciones propias del profesor.

Al reflexionar a partir de las respuestas dadas por los profesores y teniendo en cuenta los hallazgos en las diferentes instancias de esta investigación, el reto de enseñanza subyace en las habilidades en didáctica de los profesores frente a los objetos de saber. La capacidad para, didácticamente, poder enseñar los objetos de saber es el factor principal para el concepto de reto de enseñanza. Justo en este punto, la extensión de la teoría de transposición didáctica propuesta cobra relevancia, dado que los objetos de saber ya están experimentando transposiciones desde el momento en que el saber

5 Los profesores entrevistados denominaron “pedagogía” a todos los aspectos relacionados con el ejercicio adecuado de la enseñanza. Para esta investigación, el ejercicio adecuado de la enseñanza se define dentro de los cánones de la didáctica como parte integral del concepto de pedagogía.

erudito ha sido publicado. El arte de adaptar los lenguajes para los objetos de saber, los objetos a enseñar, y los objetos de enseñanza constituye el núcleo fundamental del reto de enseñanza.

En este sentido, enseñar teniendo en cuenta el contexto y las circunstancias que rodean a los estudiantes se torna en un arte. Es precisamente donde el concepto de reto de enseñanza establece las pautas para que el profesor potencie los escenarios propicios para facilitar el aprendizaje en sus estudiantes. Hoy, los estudiantes que ingresan a fundamentos de programación tienen un denominador común en cuanto a su generación se refiere. Tecnológicamente, existen nuevos imaginarios y expectativas por parte del grupo estudiantil. Frente a esta situación, es donde ese arte de enseñar apropiadamente necesita una lectura crítica sobre el grupo social al cual se enfrenta (Breed & Taylor, 2013).

Reto de aprendizaje

Se plantea un gran interrogante: ¿Cómo aprenden los estudiantes del curso de fundamentos de programación, hoy en día? Esta pregunta obliga a indagar sobre las características que identifican a los individuos según su generación. En este sentido, "los estudiantes han cambiado fundamentalmente con respecto a sus necesidades intelectuales, sociales, motivacionales y emocionales. El estudiante moderno no sólo usa la tecnología a diario, sino que se ha vuelto dependiente de ella" (Breed & Taylor, 2013, p. 868). Es cierto que los grupos sociales perciben una gran influencia a partir del contexto donde se desenvuelven; sin embargo, en asunto de adaptación tecnológica, los hallazgos indican que los contextos donde se imparten fundamentos de programación de computadoras tienen circunstancias similares en recursos y conectividad; al menos si se trata de acceso a Internet y el uso de dispositivos móviles de propósito general.

Las diferencias generacionales son notorias en el campo de la enseñanza de los fundamentos de la programación. A continuación, se cita textualmente un análisis de los estereotipos de individuos según estas generaciones. Teniendo en cuenta esta condición generacional en los estudiantes, aparece el concepto de reto de aprendizaje que se desarrolla en este punto. Una interesante reflexión argumenta:

> *Por medio de Internet [las nuevas generaciones] tienen acceso a un enorme horizonte de información, lo que, en vez de ser un problema, más bien se convierte en un beneficio que los lleva a la reflexión, y promueve un pensamiento crítico. Otro aspecto por desarrollar en las nuevas generaciones de alumnos es el sentido del servicio, ya que no basta con acumular experiencias o conocimientos de forma personal, sino que deben aprender a compartirlos con los demás* (López-Sámano, 2017).

De acuerdo con lo anterior, la tecnología forma parte fundamental en la vida del estudiante. Cabe mencionar que en el diálogo interno que se sostuvo en la entrevista a los profesores, todos afirmaron usar recursos digitales dentro del proceso de enseñanza de fundamentos de programación. Los profesores se apoyan en plataformas Web para administrar su curso y realizar un seguimiento a los estudiantes en cuanto a trabajos y tareas. En dicho diálogo, los profesores manifestaron, además, que los estudiantes se apoyan los unos a los otros y comparten recursos, tutoriales, ejercicios, y muchos más materiales digitales que se encuentran fácilmente en el servicio *Web* de Internet.

Son estas condicionales las que generan un espacio de reflexión para el profesorado. Era muy diferente impartir un curso de fundamentos de programación hace 30 años que impartirlo ahora; no sólo por la tecnología involucrada, sino principalmente por las condiciones y los estilos de aprendizaje de las generaciones de estudiantes involucrados. Curiosamente, algunos profesores en la entrevista afirmaban, con cierto humor, que sus estudiantes no podían creer que ellos hayan estudiado sin acceso a Internet. Ante esta situación, incluso uno de ellos dijo textualmente que se sentía como un "dinosaurio" frente a sus estudiantes. En realidad, los retos son tanto para los profesores como para los estudiantes, y es por tal razón que esta investigación dio una valoración importante a los conceptos de reto de enseñanza y reto de aprendizaje.

Referencias bibliográficas

Tomando como referente el punto anterior acerca del reto de aprendizaje, hay que tener presente que los estilos en que los estudiantes aprenden han cambiado significativamente. A partir de las entrevistas, se logró evidenciar que el uso de textos guías es importante para el desarrollo del curso, dichos

textos han sido tomados como referencia complementaria a la formación. No hay obligatoriedad de realizar una lectura exhaustiva de dichos textos -cabe resaltar aquí que, algunos de los textos de referencia tienen cerca de mil páginas-. Se pretende que los textos usados en clase sirvan de soporte a lo que se está desarrollando en el momento, y como afianzamiento dentro del trabajo independiente realizado por los estudiantes.

Unánimemente, los profesores afirmaron que, la enseñanza sobre fundamentos de programación tiene un alto componente práctico; así, entre más ejercicios se resuelvan, mucho mejor será el aprendizaje dado por dichas experiencias en programación.

Con estos antecedentes, se concluye que los recursos bibliográficos en cuanto a fundamentos de programación son suficientes. Pero en la reflexión sobre el reto de aprendizaje se ha logrado establecer que la forma en que los estudiantes adquieren los conocimientos, no necesariamente, apunta en forma estricta a la lectura de textos guías o de referencia. Según el reto de aprendizaje, los estudiantes disponen, hoy en día, de un abanico inconmensurable de recursos digitales para potenciar sus aprendizajes, más allá de la lectura basada en un texto guía.

Eventos motivacionales

Según el Diccionario de la Lengua Española, la tercera acepción de la palabra motivar dice: “Influir en el ánimo de alguien para que proceda de un determinado modo. Ej. El profesor motiva a los alumnos para que estudien.” (Real Academia Española, 2018). En este sentido, la motivación tiene un vínculo estrecho en los procesos de enseñanza y aprendizaje. En este escenario, la motivación puede manifestarse en forma interna —automotivación— o en forma externa —a través de otras personas o circunstancias—; sin embargo, según el interés del individuo, la motivación puede clasificarse como intrínseca o extrínseca, así: “la motivación intrínseca es la que se traduce en acciones realizadas por el interés que genera la propia actividad, en cambio, la motivación extrínseca es la que el individuo desarrolla para satisfacer otras necesidades diferentes a las relacionadas con la actividad principal” (Morales-Cadena et al., 2017).

Para Báez y Chacón (2013) la motivación es “uno de los factores que influye en la medida en que las personas tienen éxito o fracasan en cualquier proceso de aprendizaje”. Hay quienes afirman que los incentivos -como medio de motivación- en busca de satisfacción pueden ser convenientes, tal es el caso de Weller (2005) quien afirma que “el uso de incentivos se basa en el principio de que el aprendizaje se produce de manera más efectiva cuando el alumno experimenta sentimientos de satisfacción”. En contraposición, existen aportes en pedagogía que han tratado el tema de la motivación alejada del enfoque de estímulo-respuesta, es el caso de Bruner (citado por Brown, 2000, p. 165) quien afirma que “una de las maneras más efectivas de ayudar a los estudiantes a pensar y aprender es liberándolos del control de recompensas y castigos”.

En este diálogo, a pesar de que Weller resalta el uso de incentivos, afirma también que, éstos deben trascender más allá de una simple recompensa. Según esto, el incentivo debería orientarse hacia potenciar la automotivación en el estudiante, de tal manera que los estudiantes, por sí mismos, puedan encontrar satisfacción en el aprendizaje basado en la comprensión de que los objetivos son útiles para ellos o, también, sentir satisfacción en el puro placer de explorar cosas nuevas (Weller, 2005).

A partir del hecho de conocer que la motivación tiene importancia mayúscula en los procesos de enseñanza y de aprendizaje, el incentivo enfocado hacia el desarrollo de competencias de formación para la vida, -más allá de una simple recompensa trivial, momentánea y pasajera-, es de gran interés. Por tal motivo, se desarrolla el concepto de eventos motivacionales, que permitan a los profesores la generación de espacios de interacción para la construcción de saberes orientados hacia una explícita aplicación en el mundo real. Para brindar la oportunidad a los estudiantes de notar la importancia de lo que están aprendiendo dada la incidencia e impacto en el mundo de la vida dentro de un contexto dado.

Valoración de aprendizaje

Quizás, uno de los aspectos más documentados en experiencias en educación trata sobre la evaluación de aprendizajes. Esta gran productividad en cuanto a formas de evaluar podría interpretarse como un aspecto de alto interés por parte de la comunidad académica. Dada su naturaleza subjetiva, primero se debe realizar un acercamiento al concepto de evaluar aprendizajes; para tal menester, una definición interesante aplicada a la Educación Superior se cita a continuación, dado que involucra no

solo las acciones del estudiante, sino que también integra los procesos del profesorado.

> *La evaluación del aprendizaje es un proceso sistemático y permanente que comprende la búsqueda y obtención de información de diversas fuentes acerca de la calidad del desempeño, avance, rendimiento o logro del estudiante y de la calidad de los procesos empleados por el docente, la determinación de su importancia y pertinencia de conformidad con los objetivos de formación que se espera alcanzar, todo con el fin de tomar decisiones que orienten el aprendizaje y los esfuerzos de la gestión docente.* (Ianfrancesco, 2002).

De acuerdo con Ianfrancesco, la evaluación de aprendizajes necesariamente se fundamenta en procesos de comunicación interpersonal; donde la interacción entre evaluador y evaluado enriquece en sí el proceso de aprendizaje sobre el objeto de estudio. Esta relación va más allá de asignar una calificación a un trabajo, sino que ayuda a explorar posibilidades de aprendizaje inmersas en la propia evaluación.

Si bien es cierto, la producción académica sobre este punto hace referencia al concepto de evaluación. En forma muy particular, se prefiere hablar de valoración en lugar de evaluación de aprendizajes. Entiéndase por valoración de aprendizajes a aquellas situaciones con propósitos preestablecidos para el afianzamiento del aprendizaje. Para los intereses de esta investigación, realmente no se trata de cuantificar —llevar a un número o a una representación equivalente— lo que se ha aprendido, sino de dar relevancia -dar valor- a lo aprendido para los propósitos de formación. Según esta postura, hay que tener presente un contexto sociohistórico frente al proceso de valoración de aprendizajes. Tal como lo afirma Londoño (2007, p. 53),

> *La evaluación se realiza con referencia a normas y valores vigentes en la sociedad y las concepciones y valores de los implicados en la misma — instituciones, personas–. Aquello que se evalúa —conocimientos, habilidades, actitudes, modos de comportamiento, valores— y cómo se evalúa, dependen de lo que se considera valioso y pertinente en un contexto social e histórico determinado.*

Es precisamente la tarea de valorar lo que es importante en el aprendizaje de los fundamentos de programación, una labor propia de la noosfera que se ha conformado para dicha tarea. En este sentido, la visión de noosfera —integrada a las situaciones de transposición didáctica— de esta investigación no solamente se limita a seleccionar los contenidos apropiados para ser impartidos en un determinado curso, sino que establece una relación de valores que apuntan hacia los propósitos educativos previamente formulados por una institución educativa.

Así, el término valorar se desliga del concepto de calificar —que es propio de la evaluación–. En su lugar, el término valorar toma su significado literal que es dar valor a algo, y no es al aprendizaje como tradicionalmente se habla, sino dar valor a lo aprendido y lo que implica ese saber en el mundo de la vida.

Trabajo independiente

El origen del concepto de crédito académico se remonta a finales del siglo XIX, cuando en la Universidad de Harvard se implementó un sistema de hora-crédito con fines de medición del esfuerzo estudiantil (Gerard, 1955). Con estas experiencias, nació en Europa una iniciativa de establecer un marco común que facilite la movilidad académica en dicha zona. De esta manera, en la década de los 80 del siglo pasado fue donde la Declaración de la Sorbona en 1998 sentó las bases para la creación de un Espacio Europeo de Educación Superior y, un año después, en la Declaración de Bolonia finalmente se establecieron los principios y compromisos básicos para orientar dicho proceso. Uno de los aportes de esta última declaración fue la creación del sistema de créditos académicos ECTS —del inglés *European Credit Transfer and Accumulation System*— el cual facilita a los estudiantes los desplazamientos entre distintos países. Como los créditos se basan en los resultados del aprendizaje y la carga de trabajo de un curso, los estudiantes pueden transferir sus créditos ECTS de una universidad a otra, con el propósito de que se sumen a un programa concreto de grado o de formación. Por definición, “Los créditos ECTS representan la carga de trabajo y los resultados del aprendizaje —lo que el individuo sabe, comprende y es capaz de hacer— de un determinado curso o programa. 60 créditos equivalen a un año completo de estudios o trabajo”. (*European Commission*, 2018). Esa carga de trabajo de la que habla la definición anterior hace referencia al esfuerzo del estudiante por aprender. En dicho esfuerzo se contempla

no sólo los escenarios donde se encuentre acompañado por el profesor, sino que se integra a dicho esfuerzo el trabajo que el estudiante hace sin su presencia. Esta situación ha generado el concepto de trabajo autónomo o trabajo independiente.

De esta manera, el trabajo independiente es intencionado, con claros propósitos y es planeado de forma responsable. No se trata de mandar actividades simplemente para cumplir lo que está establecido en la asignación de créditos. Todas las actividades de clase y, fuera de ella, deben apuntar hacia los propósitos de formación. En este orden de ideas, el trabajo independiente debe ser preparado con antelación y siguiendo un hilo conductor para potenciar los aprendizajes. Tal como lo expresa Vergara en sus reflexiones sobre el trabajo independiente en la Educación Superior colombiana:

> *Se tiene que tomar conciencia, que demostrar la existencia o el cumplimiento de esta condición —un registro calificado para un plan de estudios—, que se llama organización de las actividades de la formación por créditos académicos, no es cumplir con la expedición de una resolución explicando a cuántas horas de acompañamiento docente corresponde el trabajo independiente, sino que verdaderamente se demuestre que el trabajo independiente es funcional y acertado en la formación del profesional* (Vergara, 2009).

La planeación del trabajo independiente tiene sus bases en la noosfera y se articula a través de los eventos motivacionales. Estas relaciones y conceptos se presentan en el siguiente punto de este capítulo, el cual propone estrategias de enseñanza basadas en la teoría desarrollada. Así, en cumplimiento con el último objetivo específico que se planteó, se propone a continuación unas estrategias de enseñanza de los objetos de saber seleccionados en los fundamentos de programación de computadoras.

Con estas consideraciones, un abanico de alternativas para el trabajo independiente debería ser propuesto, de tal suerte que los estudiantes puedan tener la oportunidad de seleccionar sus tareas y actividades a desarrollarse como su trabajo independiente. Dar esta oportunidad a los estudiantes, puede generar un incremento cualitativo acerca del disfrute del curso que se está estudiando (Fulton & Schweitzer, 2011, p. 11).

6.3 Estrategias de enseñanza propuestas

En el escenario de educación, las estrategias de enseñanza forman parte sustancial en la consecución de los propósitos educativos. Teniendo en cuenta que se involucran aspectos asociados a las prácticas de enseñanza para los fundamentos de programación de computadoras, es importante resaltar que se requiere una definición apropiada de estrategia de enseñanza, que sirva de soporte para establecer puntos de comparación entre las experiencias narradas en la entrevista semi-estructurada y basada en la información complementaria de los diferentes instrumentos de recolección de información.

A fin de apropiar un concepto de estrategia de enseñanza, se ha tomado la revisión sistemática de literatura como estrategia de recolección de información. Así, la Tabla 11 muestra una serie de definiciones obtenidas en dicha revisión:

Tabla 11. Conceptos acerca de estrategia de enseñanza

Autores	Definición	Conceptos comunes
(García & Cañal, 1995, p. 6)	"estrategia de enseñanza como un sistema peculiar constituido por unos determinados tipos de actividades de enseñanza que se relacionan entre sí mediante unos esquemas organizativos característicos"	Sistema, actividades, relaciones, esquemas organizativos.
(Larriba-Naranjo, 2001, p. 78)	"estrategia de enseñanza como el proceso de reflexión acerca de la enseñanza que se plasma, en la práctica, en un conjunto organizado de acciones educativas que implican la utilización y organización de unos recursos materiales y la realización de unas actividades determinadas"	Proceso, práctica, conjunto organizado, acciones, utilización, recursos, actividades.
(Anijovich & Mora, 2010, p. 58)	"una estrategia de enseñanza es un conjunto de decisiones que el docente toma para cada situación particular de práctica"	Conjunto, decisiones, docente, situación, práctica.
(Orlich *et al.*, 2010, p. 14)	*"Teaching Strategies helps all prospective teachers to acquire the basics of professional knowledge that are so necessary to facilitate learning for all our nation's children"*	*Professional knowledge, teachers, facilitate, learning.*
(Nurhayati, 2014, p. 20)	"*Teaching strategy is a plan for learning activities (the series of activities) that include the use of methods, techniques and also the resources or ability to achieve the learning goals*"	*Plan, learning activities, methods, techniques, resources, goals.*

Fuente: esta investigación

Con la información anterior, es posible establecer los conceptos comunes que han sido empleados para construir las definiciones de estrategia de enseñanza. En la elaboración de un concepto de estrategia de enseñanza apropiado para esta investigación, se ha podido determinar el conjunto de conceptos a ser representados bajo una sola definición; la Tabla 12 muestra dicho conjunto.

Tabla 12. Conceptos e interpretaciones

Concepto común	Interpretación	Análisis	Rol
Sistema, Proceso, Conjunto, *Plan*	Los autores utilizan estos conceptos para definir lo que es una estrategia de enseñanza	Se trata de la definición principal, se puede decir que se propone un sistema con sus respectivos elementos que han sido previamente diseñados y que siguen un plan determinado.	Definición
Actividades, acciones, relaciones, decisiones, *profesional knowledge*, *learning activities*	Los autores utilizan estos conceptos para definir los componentes o elementos que conforman la estrategia de enseñanza	De acuerdo con el concepto anterior, se trata de establecer aquellos elementos que forman parte de la definición de estrategia de enseñanza.	Componentes
Situación, práctica	Los autores usan estos conceptos para hacer énfasis en el escenario final donde la estrategia de enseñanza se aplica	Con este concepto se puede establecer el escenario de aplicación de una estrategia de enseñanza.	Escenarios de aplicación
Facilitate, learning, goals	Los autores usan estos conceptos para determinar el fin último de una estrategia de enseñanza	Facilitar el aprendizaje en los estudiantes y el complimiento de las metas establecidas serán el objetivo final	Objetivos
Utilización, recursos, *methods, techniques, resources*	Los autores utilizan estos conceptos para establecer los medios por el cual se explicita una estrategia de enseñanza	La estrategia de enseñanza requiere unos medios por los cuales se pueda evidenciar su diseño en la práctica.	Recursos
Docente, *teacher*	Los autores usan el concepto para explicar que es el docente quién ejecuta la estrategia de enseñanza	El docente está a cargo de diseñar y ejecutar la estrategia de enseñanza	Actores

Fuente: esta investigación

Con el análisis anterior, es posible construir una definición de estrategia de enseñanza basada en la revisión sistemática de literatura. Dicha definición se presenta a continuación.

Definición de estrategia de enseñanza

En el desarrollo de esta investigación se propuso el uso de un lenguaje controlado llamado esquemas preconceptuales a fin de generar una síntesis según los conceptos generados en la Tabla 18. Según Zapata (2007), un esquema preconceptual es una forma de representar el conocimiento mediante el uso de un lenguaje controlado. Además, los esquemas preconceptuales usan notación simple, son fáciles de entender y se adaptan a cualquier dominio del conocimiento (Zapata, Arango & Gelbukh, 2006). Así, una estrategia de enseñanza será definida ontológicamente según la Figura 19.

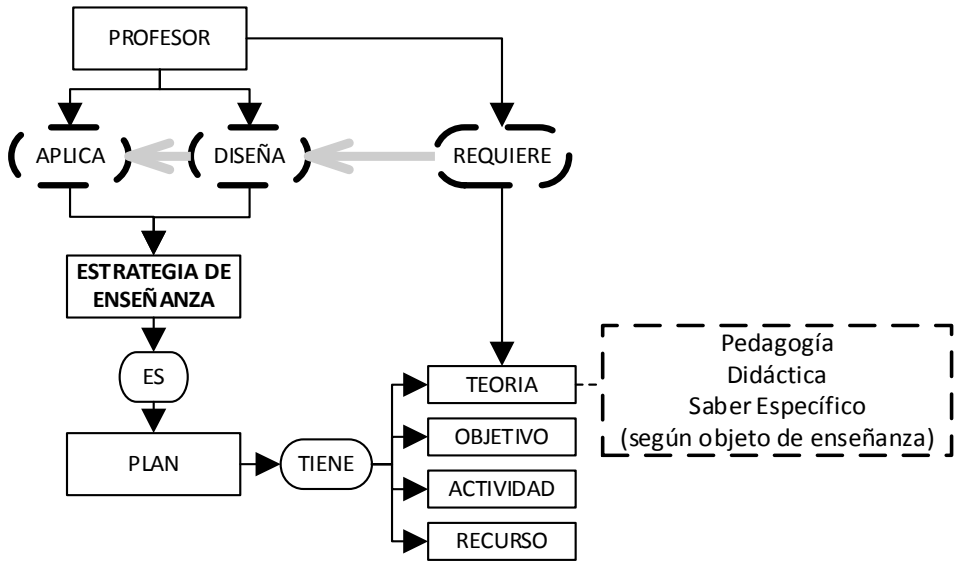


Figura 19. Esquema preconceptual para la representación ontológica del concepto de Estrategia de Enseñanza (Fuente: esta investigación)

En forma textual, una estrategia de enseñanza es un plan compuesto por una fundamentación teórica, objetivos, actividades y recursos, diseñado y aplicado por el docente fundamentado en su conocimiento pedagógico, didáctico y su saber específico (según el objeto de enseñanza). Con esta definición, cabe resaltar que ya se tiene un objeto de enseñanza específico, que son los fundamentos de programación de computadoras; por lo tanto, se asume esta definición de estrategia de enseñanza para dicho escenario, con el fin de determinar pautas para comparar entre diferentes estrategias de enseñanza que han sido previamente identificadas según la recolección de información.

La definición anterior se establece como el punto de partida para establecer puntos de comparación entre diferentes estrategias de enseñanza en los fundamentos de programación de computadoras. Los elementos que componen dicha definición fueron analizados a partir de las experiencias de los profesores encuestados y entrevistados.

Comparación entre estrategias de enseñanza

Existe a la fecha un amplio espectro de modelos de enseñanza, estrategias, metodologías y técnicas que funcionan en aulas complejas (Orlich *et al.*, 2010, p. 15). Con esa diversidad de estrategias, es conveniente utilizar un método de comparación entre ellas. A partir de la revisión de literatura, se encuentra que todos los estudios que realizan comparaciones entre dos o más estrategias de enseñanza parten de la elaboración de unos instrumentos intencionados, con el fin de explorar aquellos aspectos que se desean comparar. En el caso de experiencias de enseñanza de comprensión lectora, se plantean dos grupos separados con diferentes estrategias de enseñanza; posteriormente se obtienen los resultados con cuestionarios aplicados a los sujetos (Hudson *et al.*, 2013). Por otra parte, una experiencia de enseñanza en psicología muestra la aplicación de *pretest* y *postest* a fin de establecer puntos de comparación entre estrategias de enseñanza (Roselli, 2010). La forma de comparar dos o más estrategias de enseñanza, según la metodología propuesta en una tesis de maestría en métodos y técnicas de investigación social, apuntan hacia el desarrollo de cuestionarios que permitan evidenciar el tipo de enfoque pedagógico con que son llevadas a cabo las estrategias de enseñanza en la práctica (López, 2015).

Dadas las anteriores experiencias, se puede concluir que cada investigación establece sus propios lineamientos acerca de cómo comparar dos o más estrategias de enseñanza. En este sentido, se propone un esquema matricial expresado en la Tabla 13 para la realización del análisis comparativo de las experiencias de estrategias de enseñanza de los fundamentos de programación, a partir de la información recolectada.

Tabla 13. Matriz de comparación de estrategias de enseñanza

Profesor	Estrategia de enseñanza	Teoría pedagógica y didáctica	Actividades destacadas	Recursos
24.F	Estructuras textuales, analogías	Conductista, Clase magistral	Representación de espacios de memoria a través de cajas o casilleros, acciones de direccionamiento de memoria. Ejercicios en clase.	Tablero Digital, Laboratorio de computadoras
25.L	Estructuras textuales	Conductista, Clase magistral	Exposición temática y desarrollo de ejercicios en clase.	Tablero tradicional, Proyector, Laboratorio de computadoras
28.X	Estructuras textuales	Conductista, Clase magistral	Exposición temática y desarrollo de ejercicios en clase.	Tablero digital, Laboratorio de computadoras
36.H	Estructuras textuales, analogías	Cognitivista, ABP	Diseño de juegos de computadoras. Desarrollo del proyecto de curso usando una bitácora de seguimiento.	Tablero digital, Bitácora de proyecto de curso, Laboratorio de computadoras
40.H	Estructuras textuales, organizador previo	Cognitivista, ABP	Resolución de problemas reales a través de modelos matemáticos. Planteamientos de retos matemáticos.	Guías de desarrollo, Tablero digital, Laboratorio de computadoras
37.I	Estructuras textuales, analogías	Cognitivista, ABP	Exposición temática y desarrollo de ejercicios en clase.	Tablero tradicional, Proyector, Laboratorio de computadoras

Fuente: esta investigación

La columna “estrategia de enseñanza” relaciona conceptos de didáctica que se formulan a la luz de la teoría. Existen varias concepciones alrededor del concepto de estrategia de enseñanza; en este caso, se consideraron las definiciones de Díaz y Hernández (1999). Los posibles valores de la columna Estrategia de Enseñanza fueron tomados según la Tabla 14:

Tabla 14. Estrategias de Enseñanza

Objetivos	Enunciado que establece condiciones, tipo de actividad y forma de evaluación del aprendizaje del alumno. Generación de expectativas apropiadas en los alumnos.
Resumen	Síntesis y abstracción de la información relevante de un discurso oral o escrito. Enfatiza conceptos clave, principios, términos y argumento central.
Organizador previo	Información de tipo introductorio y contextual. Es elaborado con un nivel superior de abstracción, generalidad e inclusividad que la información que se aprenderá. Tiende un puente cognitivo entre la información nueva y la previa.
Ilustraciones	Representación visual de los conceptos, objetos o situaciones de una teoría o tema específico (fotografías, dibujos, esquemas, gráficas, dramatizaciones, etcétera).
Analogías	Proposición que indica con una cosa o evento (concreto y familiar) es semejante a otro (desconocido y abstracto o complejo).
Preguntas intercaladas	Preguntas insertadas en la situación de enseñanza o en un texto. Mantienen la atención y favorecen la práctica, la retención y la obtención de información relevante.
Pistas tipográficas y discursivas	Señalamientos que se hacen en un texto o en la situación de enseñanza para enfatizar y/u organizar elementos relevantes del contenido por aprender.
Mapas conceptuales y redes semánticas	Representación gráfica de esquemas de conocimiento (indican conceptos, proposiciones y explicaciones).
Uso de estructuras textuales	Organizaciones retóricas de un discurso oral o escrito, que influyen en su comprensión y recuerdo.

Fuente: Díaz & Hernández (1999)

Con respecto a la teoría pedagógica y didáctica, los posibles valores fueron tomados a partir de la clasificación de teoría, enfoque y modelo pedagógico explicados en la Tabla 15.

Tabla 15. Síntesis de clasificación de teoría pedagógica y didáctica

TEORÍAS	ENFOQUES	MODELOS	DIDÁCTICAS
Conductista	Pedagogía Conductista	Educación Conductista	Clase magistral
Cognitivista	Pedagogía Constructivista Pedagogía Conceptual Inteligencias Múltiples	Educación Desarrollista	Aprendizaje basado en problemas Aprendizaje Activo Aprendizaje Colaborativo
Socio-Crítica	Pedagogía Liberadora Pedagogía Socio-crítica	Educación Social	Investigación-Acción Investigación-Acción-Participativa

Fuente: Teorías, Enfoques y Modelos basado en (Pérez, 2006), lista de didácticas por revisión documental.

Como conclusión parcial al momento de comparar las estrategias de enseñanza de los fundamentos de programación, es notorio que aún la mitad de los profesores se enmarcan en una teoría pedagógica conductista, y la otra mitad refleja algunas prácticas dentro de la teoría pedagógica cognitivista. A partir de estos hallazgos y teniendo presente la base teórica-conceptual desarrollada en esta investigación, el punto a continuación describe las estrategias de enseñanza propuestas para los objetos de saber seleccionados de los fundamentos de programación de computadoras. Las estrategias de enseñanza propuestas se clasifican dentro de la teoría cognitiva, de enfoque constructivista, y con una didáctica de aprendizaje basado en problemas, sin decir que esta condición sea obligatoria para los diseños que las personas deseen proponer.

En primera instancia, la extensión de la teoría de transposición didáctica denominada *In Extensa Sensu* integra características muy puntuales en la educación en ciencias computacionales, esta extensión hace uso del lenguaje propio de dichas ciencias. En aras de proponer estrategias de enseñanza dentro de un estadio de la educación en ciencias computacionales, se propuso el uso de un formato de diseño de dichas estrategias a la luz de la teoría y los conceptos trabajados. De esta forma, la Figura 20 presenta el diseño general.

FORMATO DE DISEÑO DE ESTRATEGIAS DE ENSEÑANZA BASADA EN TRANSPOSICIÓN DIDÁCTICA "IN EXTENSA SENSU"			
OBJETO DE SABER (1ra Transposición)			
Interacción en Noosfera	Integrantes de Noosfera		
	Necesidades del Contexto Local		
	CONTENIDO		
	OBJETO A ENSEÑAR (2da Transposición)	Encapsulamiento de Objeto	
Dimensión Conceptual	ESTRATEGIA DE ENSEÑANZA		
	TEORÍA		
	Pedagogía	Didáctica	OBJETO DE ENSEÑANZA (3ra Transposición)
	PRE-CONCEPTOS DE LOS ESTUDIANTES		
	OBJETIVOS		
	ACTIVIDADES		
	RECURSOS		
Puntos focales	Experiencia en Enseñanza		
	Lenguaje en Contexto		
	Reto de Enseñanza		
	Reto de Aprendizaje		
	Referencias Bibliográficas		
	Eventos motivacionales		
	Valoración de Aprendizaje		
	Valoración de Objeto y Técnicas de Evaluación	OBJETO DE EVALUACIÓN (4ta Transposición)	
	Trabajo Independiente		

Figura 20. Formato de diseño de estrategias de enseñanza basado en Transposición Didáctica In Extensa Sensu (Fuente: esta investigación)

Estrategia de enseñanza propuesta para el objeto de saber: variable

Tabla 16. Estrategia de enseñanza propuesta para variable

FORMATO DE DISEÑO DE ESTRATEGIAS DE ENSEÑANZA BASADA EN TRANSPOSICIÓN DIDÁCTICA *IN EXTENSA SENSU*		
OBJETO DE SABER: VARIABLE (1ra Transposición)		
Una **variable** es un espacio de memoria de las computadoras que tiene un identificador, direccionamiento y tipo de dato. Esta variable es usada por los programadores para almacenar valores. (Tomado de ontología computacional, p. 89 de este documento)		
Interacción en noosfera	**Integrantes de noosfera**	
	Profesores de fundamentos de programación Director del Departamento, Área o Carrera Personal del sector productivo y la industria	
	Necesidades del contexto local	
	A través de entrevistas, los profesores del curso de fundamentos de programación indagan las necesidades del contexto local con el personal del sector productivo y la industria. Junto con el director de departamento, se articulan dichas necesidades al plan o proyecto de formación institucional, determinando su correspondencia con la misión y visión institucional. Para esta propuesta, las necesidades del contexto local alrededor del concepto de **variable** pueden ser (dependerán de cada contexto), por ejemplo: Riesgo volcánico - tipos de datos numéricos (reales de doble precisión) Producción agrícola - tipos de datos numéricos (reales de doble precisión) Producción acuícola - tipos de datos numéricos (reales de doble precisión) Producción ganadera - tipos de datos numéricos (reales de doble precisión) Expresiones artísticas, cultura del carnaval, poesía, música – tipo de datos alfanuméricos (cadenas de caracteres)	
	CONTENIDO	
	OBJETO A ENSEÑAR (2da Transposición)	**Encapsulamiento de objeto**
	La **variable** representa un espacio en memoria que es capaz de almacenar un valor a la vez. Los valores responden a tipos de datos dependientes de los lenguajes de programación. El beneficio de usar variables radica en poder cambiar sus valores durante la ejecución de un programa de computadoras, facilitando así un comportamiento dinámico frente a valores y cálculos en tiempo de ejecución.	Atributos: Visibilidad Tipo de dato Valor Operaciones: Declarar (…) AsignarValor(…)

ESTRATEGIA DE ENSEÑANZA				
Dimensión conceptual	**TEORÍA**			
		Pedagogía	**Didáctica**	**OBJETO DE ENSEÑANZA (3ra Transposición)**
		Cognitiva-Constructivista	Aprendizaje Activo-Aprendizaje Basado en Problemas	Usemos la **variable** para almacenar datos del contexto, usemos además diferentes tipos de datos que describen situaciones de riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y expresiones artísticas y culturales.
	PRECONCEPTOS DE LOS ESTUDIANTES			
		Se espera que los estudiantes en esta instancia tengan un conocimiento previo matemático, en especial del álgebra y de resolución de ecuaciones con incógnitas. Dichos escenarios recrean el concepto de variable. Se realiza un sondeo para determinar qué tanto conocen sobre riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y expresiones artísticas y culturales.		
	OBJETIVOS			
		Comprender la importancia de **variables** en la representación del mundo real, teniendo en cuenta que ellas pueden almacenar información del contexto, y tiene la posibilidad de cambiar su valor de acuerdo con las dinámicas de este.		
	ACTIVIDADES			
		1 sesión de clase planeada así: La primera parte de exploración de los preconceptos matemáticos y del álgebra, donde se permita abordar el tema de las variables en dicho escenario. En esta sesión, el profesor indica a través de ejemplos el uso de variables matemáticas y en álgebra. Una serie de ejercicios en clase deberán ser desarrollados por los estudiantes para afianzar el concepto. Dichos ejercicios tendrán relación directa con las necesidades del contexto manifestadas en este formato. La segunda parte de declaración de variables computacionales a través de un lenguaje de programación, exploración de tipos de datos, e instrucciones de asignación de valores. Mediante depuradores se explora el almacenamiento de los valores en memoria y su cambio en tiempo de ejecución. Se desarrollan programas simples de almacenamiento y cambio de valores.		
	RECURSOS			
		Lenguaje de programación / depurador (depende del lenguaje de programación).		

Puntos focales			
	Experiencia en enseñanza		
		Las reflexiones sobre la práctica de enseñanza se orientarán hacia: Facilidad de representación de datos del mundo real por parte del estudiante	
	Lenguaje en contexto		
		Todas las definiciones de variables estarán inmersas en el vocabulario de las situaciones descritas en las necesidades del contexto. Por ejemplo: *intesidad_sísmica, pH_del_suelo, camada, coro*, etc.	
	Reto de enseñanza		
		Lograr que los estudiantes representen su entorno a través de declaración de variables y posibles valores de un estado determinado.	
	Reto de aprendizaje		
		Los estudiantes buscan material audiovisual en la Web sobre la creación de variables en un lenguaje determinado. Los estudiantes comparten entre sí el material encontrado, hacer uso de un foro de discusión sobre lo encontrado.	
	Referencias bibliográficas		
		... (Varía según el lenguaje de programación)	
	Eventos motivacionales		
		Lluvia de ideas para la exploración con los estudiantes sobre los posibles usos de las variables en los problemas del contexto. *Gamification* aplicada al concepto de variable (juego de pares / *game pairs*) usando la metáfora del juego de pares, se relaciona un valor, y el tipo de dato asociado en parejas. Los estudiantes juegan sobre un tablero de celdas ocultas, por turnos se descubren parcialmente las parejas (valor, tipo de dato), hasta destapar todas las celdas del tablero.	
	Valoración de aprendizaje		
		Valoración del objeto y técnicas de evaluación	**OBJETO DE EVALUACIÓN (4ta Transposición)**
		Autoevaluación Uso de rúbricas diseñadas por el docente donde se aplicará la revisión por pares. Grabación audiovisual de bitácora donde los estudiantes explican por qué es importante el concepto de **variable** en su proyecto de curso.	El conjunto de **variables** declaradas que serán utilizadas en el desarrollo del proyecto de curso. Revisión del *dossier* de ejercicios
	Trabajo independiente		
		Fragmentos de lecturas complementarias que relacionen variables al riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, expresiones artísticas y culturales de la región. *Dossier* de ejercicios de declaración de variables para los contextos mencionados Proyecto de curso	

Fuente: esta investigación.

Estrategia de enseñanza propuesta para el objeto de saber: constante

Tabla 17. Estrategia de enseñanza propuesta para constante

<table>
<tr><td colspan="4">FORMATO DE DISEÑO DE ESTRATEGIAS DE ENSEÑANZA BASADA EN TRANSPOSICIÓN DIDÁCTICA IN EXTENSA SENSU</td></tr>
<tr><td colspan="4">OBJETO DE SABER: CONSTANTE (1ra Transposición)</td></tr>
<tr><td colspan="4">Una constante es un espacio de memoria de las computadoras que tiene un identificador, direccionamiento y tipo de dato. Esta constante es usada por los programadores para definir valores de solo lectura, es decir que son inmodificables a lo largo del programa. (Tomado de ontología computacional, p. 90 de este documento)</td></tr>
<tr><td rowspan="8">Interacción en noosfera</td><td colspan="3">Integrantes de noosfera</td></tr>
<tr><td></td><td colspan="2">Profesores de fundamentos de programación
Director del Departamento, Área o Carrera
Personal del sector productivo y la industria</td></tr>
<tr><td colspan="3">Necesidades del contexto local</td></tr>
<tr><td></td><td colspan="2">A través de entrevistas, los profesores del curso de fundamentos de programación indagan las necesidades del contexto local con el personal del sector productivo y la industria. Junto con el director de departamento, se articulan dichas necesidades al plan o proyecto de formación institucional, determinando su correspondencia con la misión y visión institucional.
Para esta propuesta, las necesidades del contexto local alrededor del concepto de constante pueden ser (Obviamente, dependerán de cada contexto), por ejemplo:
Riesgo Volcánico - tipos de datos numéricos (reales de doble precisión)
Producción agrícola - tipos de datos numéricos (reales de doble precisión)
Producción acuícola - tipos de datos numéricos (reales de doble precisión)
Producción ganadera - tipos de datos numéricos (reales de doble precisión)
Expresiones artísticas, cultura del carnaval, poesía, música – tipo de datos alfanuméricos (cadenas de caracteres)</td></tr>
<tr><td colspan="3">CONTENIDO</td></tr>
<tr><td></td><td>OBJETO A ENSEÑAR (2da Transposición)</td><td>Encapsulamiento de objeto</td></tr>
<tr><td></td><td>La constante representa un espacio en memoria que es capaz de almacenar un valor a la vez al momento de la declaración. Los valores responden a tipos de datos dependientes de los lenguajes de programación. A diferencia de las variables, las constantes no permiten modificar su valor después de la declaración, el beneficio radica en poder establecer valores inmodificables en tiempo de ejecución.</td><td>Atributos:
Visibilidad
Tipo de dato
Valor
Operaciones:
Declarar(...)</td></tr>
</table>

<table>
<tr><td colspan="5">ESTRATEGIA DE ENSEÑANZA</td></tr>
<tr><td rowspan="11">Dimensión conceptual</td><td colspan="4">TEORÍA</td></tr>
<tr><td></td><td>Pedagogía</td><td>Didáctica</td><td>OBJETO DE ENSEÑANZA (3ra Transposición)</td></tr>
<tr><td></td><td>Cognitiva-Constructivista</td><td>Aprendizaje Activo-Aprendizaje Basado en Problemas</td><td>Usemos la constante para declarar valores fijos (inmodificables) del contexto, usemos además diferentes tipos de datos que describen situaciones de riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y expresiones artísticas y culturales.</td></tr>
<tr><td colspan="4">PRECONCEPTOS DE LOS ESTUDIANTES</td></tr>
<tr><td></td><td colspan="3">Se espera que los estudiantes en esta instancia tengan un conocimiento previo matemático, en especial de constantes físicas. Dichos escenarios recrean el concepto de constante. Se realiza un sondeo para determinar qué tanto conocen sobre riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y expresiones artísticas y culturales, en el sentido de encontrar valores que nunca cambian.</td></tr>
<tr><td colspan="4">OBJETIVOS</td></tr>
<tr><td></td><td colspan="3">Comprender la importancia de constantes en la representación del mundo real, teniendo en cuenta que ellas pueden almacenar información del contexto que nunca cambiará.</td></tr>
<tr><td colspan="4">ACTIVIDADES</td></tr>
<tr><td></td><td colspan="3">1 sesión de clase planeada así:
La primera parte de exploración de los preconceptos matemáticos, donde se permita abordar el tema de las constantes en dicho escenario. En esta sesión, el profesor indica a través de ejemplos el uso de constantes matemáticas tales como PI, Raíz de 2, el número e, etc. Una serie de ejercicios en clase deberán ser desarrollados por los estudiantes para afianzar el concepto. Dichos ejercicios tendrán relación directa con las necesidades del contexto manifestadas en este formato.
La segunda parte de declaración de constantes computacionales a través de un lenguaje de programación, exploración de tipos de datos. Mediante depuradores se explora el almacenamiento de los valores en memoria y su permanencia en tiempo de ejecución. Se desarrollan programas simples de uso de constantes y variables.</td></tr>
<tr><td colspan="4">RECURSOS</td></tr>
<tr><td></td><td colspan="3">Lenguaje de programación / depurador (depende del lenguaje de programación).</td></tr>
</table>

Puntos focales	**Experiencia en enseñanza**		
		Las reflexiones sobre la práctica de enseñanza se orientarán hacia: Facilidad de identificación de constantes del mundo real por parte del estudiante, entendiendo la naturaleza de inmodificabilidad de su valor.	
	Lenguaje en contexto		
		Todas las definiciones de constantes estarán inmersas en el vocabulario de las situaciones descritas en las necesidades del contexto. Por ejemplo: *PrimeraEscalaRichter*, *PI, edad_máxima*, etc.	
	Reto de enseñanza		
		Lograr que los estudiantes representen su entorno a través de declaración de constantes y posibles valores inmutables. Lograr reconocer aquellos valores del entorno que no cambian.	
	Reto de aprendizaje		
		Los estudiantes buscan material audiovisual en la Web sobre la creación de constantes en un lenguaje determinado. Los estudiantes comparten entre sí el material encontrado, hacer uso de un foro de discusión sobre lo encontrado.	
	Referencias bibliográficas		
		… (Varía según el lenguaje de programación)	
	Eventos motivacionales		
		Lluvia de ideas para la exploración con los estudiantes sobre los posibles usos de las constantes en los problemas del contexto. *Gamification* aplicada al concepto de constante (juego de pares / *game pairs*) usando la metáfora del juego de pares, se relaciona un valor, y el tipo de dato asociado en parejas. Los estudiantes juegan sobre un tablero de celdas ocultas, por turnos se descubren parcialmente las parejas (valor, tipo de dato), hasta destapar todas las celdas del tablero.	
	Valoración de aprendizaje		
		Valoración del objeto y técnicas de evaluación	**OBJETO DE EVALUACIÓN (4ta Transposición)**
		Autoevaluación Uso de rubricas diseñadas por el docente donde se aplicará la revisión por pares. Grabación audiovisual de bitácora donde los estudiantes explican por qué es importante el concepto de **constante** en su proyecto de curso.	El conjunto de **constantes** declaradas que serán utilizadas en el desarrollo del proyecto de curso. Revisión del *dossier* de ejercicios
	Trabajo independiente		
		Fragmentos de lecturas complementarias que relacionen constantes al riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, expresiones artísticas y culturales de la región. *Dossier* de ejercicios de declaración de constantes para los contextos mencionados. Proyecto de curso	

Fuente: esta investigación.

Estrategia de enseñanza propuesta para el objeto de saber: expresión aritmético-lógica

Tabla 18. Estrategia de enseñanza propuesta para expresión aritmético-lógica

FORMATO DE DISEÑO DE ESTRATEGIAS DE ENSEÑANZA BASADA EN TRANSPOSICIÓN DIDÁCTICA *IN EXTENSA SENSU*			
OBJETO DE SABER: EXPRESIÓN ARITMÉTICO-LÓGICA (1ra Transposición)			
Una expresión aritmético-lógica es un conjunto de valores, variables, constantes y operadores que permiten realizar un cálculo aritmético-lógico. Es importante tener en cuenta que los operadores tienen una jerarquía y su escritura se define según la sintaxis de un lenguaje específico. (Tomado del apartado Ontología computacional, p. 91 de este documento)			
Interacción en noosfera	**Integrantes de noosfera**		
		Profesores de fundamentos de programación Director del Departamento, Área o Carrera Personal del sector productivo y la industria	
	Necesidades del contexto local		
		A través de entrevistas, los profesores del curso de fundamentos de programación indagan las necesidades del contexto local con el personal del sector productivo y la industria. Junto con el director de departamento, se articulan dichas necesidades al plan o proyecto de formación institucional, determinando su correspondencia con la misión y visión institucional. Para esta propuesta, las necesidades del contexto local alrededor del concepto de expresión aritmético-lógica pueden ser (Obviamente, dependerán de cada contexto), por ejemplo: Riesgo volcánico – ecuaciones en sismología, modelos matemáticos, física de sismoresistencia Producción agrícola – modelos de producción, programación lineal, modelos de transporte Producción acuícola - modelos de producción, programación lineal, modelos de transporte Producción ganadera - modelos de producción, programación lineal, modelos de transporte Expresiones artísticas, cultura del carnaval, poesía, música – modelos estadísticos de publicidad y marketing	
	CONTENIDO		
		OBJETO A ENSEÑAR (2da Transposición)	**Encapsulamiento de objeto**
		La expresión aritmético-lógica consiste en una instrucción de procesamiento de valores numéricos y valores Booleanos. Dicho procesamiento se efectúa a través de operaciones aritméticas (suma, resta, potenciación, radicación, etc.) y también a través de operaciones lógicas (conjunción, disyunción, etc.) El beneficio radica en la capacidad de procesar valores de esta naturaleza. Con ellas se calculan valores y su sintaxis es dependiente de los lenguajes de programación.	Atributos: Conjunto de valores Conjunto de operadores Conjunto de llamadas a funciones Operaciones: Calcular (…)

Dimensión conceptual

ESTRATEGIA DE ENSEÑANZA

TEORÍA

	Pedagogía	Didáctica	OBJETO DE ENSEÑANZA (3ra Transposición)
	Cognitiva-Constructivista	Aprendizaje Activo-Aprendizaje Basado en Problemas	Usemos la expresión aritmético-lógica para representar modelos matemáticos y estadísticos aplicados a los contextos: riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y publicidad/ marketing aplicado a las expresiones artísticas y culturales.

PRECONCEPTOS DE LOS ESTUDIANTES

Se espera que los estudiantes en esta instancia tengan un conocimiento previo matemático y estadístico, en especial sobre representación de modelos y ecuaciones. Dichos escenarios recrean el concepto de expresiones aritmético-lógicas. Se realiza un sondeo para determinar qué tanto conocen sobre riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y publicidad y marketing en expresiones artísticas y culturales, en el sentido de encontrar expresiones matemáticas y estadísticas.

OBJETIVOS

Comprender la importancia de las expresiones aritmético-lógicas en la representación del mundo real, teniendo en cuenta que ellas pueden realizar cálculos matemáticos y de orden lógico o booleano.

ACTIVIDADES

2 sesiones de clase planeada así:
La primera sesión de exploración de los preconceptos matemáticos y estadísticos, donde se permita abordar el tema de las expresiones aritmético-lógicas en dicho escenario. En esta sesión, el profesor indica a través de ejemplos el uso de expresiones aritmético-lógicas basadas en modelos matemáticos y estadísticos según el contexto. Una serie de ejercicios en clase deberán ser desarrollados por los estudiantes para afianzar el concepto. Dichos ejercicios tendrán relación directa con las necesidades del contexto manifestadas en este formato. En esta sesión es importante hacer la traducción del lenguaje matemático al lenguaje computacional y viceversa. Se usarán expresiones de menor complejidad al inicio y se irá avanzando su nivel de complejidad en los ejercicios finales.
La segunda parte de programación de expresiones aritmético-lógicas a través de un lenguaje de programación, uso de funciones reservadas del lenguaje para cálculos. Mediante depuradores se explora el almacenamiento de los valores en memoria a través de los cálculos realizados. Se desarrollan programas simples de uso de constantes, variables y expresiones aritmético-lógicas.

RECURSOS

Lenguaje de programación / depurador (depende del lenguaje de programación).

Puntos focales

Experiencia en enseñanza

Las reflexiones sobre la práctica de enseñanza se orientarán hacia:
Facilidad de representar modelos matemáticos y estadísticos simples del mundo real por parte del estudiante usando expresiones aritmético-lógicas, entendiendo la potencia de las computadoras para efectuar cálculos a gran velocidad.

Lenguaje en contexto

Todas las definiciones de expresiones aritmético-lógicas estarán inmersas en el vocabulario de las situaciones descritas en las necesidades del contexto. Por ejemplo: (cálculo de magnitud Richter) $M_{\mathrm{L}} = \log_{10} A - \log_{10} A_0(\delta) = \log_{10}[A/A_0(\delta)]$, etc.

Reto de enseñanza

Lograr que los estudiantes representen su entorno a través de los modelos matemáticos y estadísticos. Lograr reconocer la importancia de representar la realidad y los posibles usos de dichas representaciones.

Reto de aprendizaje

Los estudiantes buscan material audiovisual en la Web sobre la representación de modelos matemáticos y estadísticos en un lenguaje determinado. Los estudiantes comparten entre sí el material encontrado, hacer uso de un foro de discusión sobre lo encontrado.

Referencias bibliográficas

... (Varía según el lenguaje de programación)

Eventos motivacionales

Lluvia de ideas para la exploración con los estudiantes sobre los posibles usos de las expresiones aritmético-lógicas en los problemas del contexto.
Gamification aplicada al concepto de expresión aritmético-lógica (juego de escaleras y toboganes / *ladders and chutes game*) usando la metáfora del juego de escaleras y toboganes, se resuelve una serie de expresiones aritmético-lógicas a través del tablero de juego en equipos, las expresiones se toman de una lista, a cada celda del tablero le corresponde una expresión aritmético-lógica compleja. El equipo que falla el cálculo regresa al tobogán más cercano. El juego termina cuando algún equipo llegue a la última posición del tablero.

Valoración de aprendizaje

	Valoración del objeto y técnicas de evaluación	OBJETO DE EVALUACIÓN (4ta Transposición)
	Autoevaluación Uso de rubricas diseñadas por el docente donde se aplicará la revisión por pares. Grabación audiovisual de bitácora donde los estudiantes explican por qué es importante el concepto de expresión aritmético-lógica en su proyecto de curso.	El conjunto de expresiones aritmético-lógicas que serán utilizadas en el desarrollo del proyecto de curso. Revisión del *dossier* de ejercicios

Trabajo independiente

Fragmentos de lecturas complementarias que relacionen expresiones aritmético-lógicas al riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, expresiones artísticas y culturales de la región.
Dossier de ejercicios de declaración de constantes para los contextos mencionados.
Proyecto de curso

Fuente: esta investigación.

Estrategia de enseñanza propuesta para el objeto de saber: condicional

Tabla 19. Estrategia de enseñanza propuesta para condicional

<table>
<tr><td colspan="4">FORMATO DE DISEÑO DE ESTRATEGIAS DE ENSEÑANZA BASADA EN TRANSPOSICIÓN DIDÁCTICA IN EXTENSA SENSU</td></tr>
<tr><td colspan="4">OBJETO DE SABER: CONDICIONAL (1ra Transposición)</td></tr>
<tr><td colspan="4">Un condicional es una estructura algorítmica que permite la bifurcación (cambio de flujo) en la ejecución de un programa a partir de la evaluación de una expresión lógica (comparación). Dentro del desarrollo teórico, independientemente del lenguaje de programación, es requerida la fundamentación en álgebra Booleana; las expresiones dependen de la sintaxis del lenguaje de programación. (Tomado del apartado Ontología computacional, p. 92 de este documento)</td></tr>
<tr><td rowspan="8">Interacción en noosfera</td><td colspan="3">Integrantes de noosfera</td></tr>
<tr><td></td><td colspan="2">Profesores de fundamentos de programación
Director del Departamento, Área o Carrera
Personal del sector productivo y la industria</td></tr>
<tr><td colspan="3">Necesidades del contexto local</td></tr>
<tr><td></td><td colspan="2">A través de entrevistas, los profesores del curso de fundamentos de programación indagan las necesidades del contexto local con el personal del sector productivo y la industria. Junto con el director de departamento, se articulan dichas necesidades al plan o proyecto de formación institucional, determinando su correspondencia con la misión y visión institucional.
Para esta propuesta, las necesidades del contexto local alrededor del concepto de condicional pueden ser (Obviamente, dependerán de cada contexto), por ejemplo:
Riesgo volcánico – parámetros de sismoresistencia, valoraciones en la escala de Richter e intensidad de Mercalli
Producción agrícola – parámetros de producción, parámetros de transporte
Producción acuícola - parámetros de producción, parámetros de transporte
Producción ganadera - parámetros de producción, parámetros de transporte
Expresiones artísticas, cultura del carnaval, poesía, música – análisis léxico de lenguaje ancestral</td></tr>
<tr><td colspan="3">CONTENIDO</td></tr>
<tr><td></td><td>OBJETO A ENSEÑAR (2da Transposición)</td><td>Encapsulamiento de objeto</td></tr>
<tr><td></td><td>El condicional es una instrucción de control de ejecución que realiza una evaluación de una expresión lógica o booleana, con el propósito de bifurcar el programa de acuerdo con los resultados de dicha evaluación. Los condicionales ayudan a cambiar la dinámica de ejecución de un algoritmo (o de un programa implementado) según el estado booleano de la expresión lógica de control</td><td>Atributos:
Expresión lógica
Operaciones:
EsVerdadera(…)
EsFalsa(…)</td></tr>
</table>

<table>
<tr><td colspan="4">ESTRATEGIA DE ENSEÑANZA</td></tr>
<tr><td rowspan="14">Dimensión conceptual</td><td colspan="3">TEORÍA</td></tr>
<tr><td></td><td>Pedagogía</td><td>Didáctica</td><td>OBJETO DE ENSEÑANZA (3ra Transposición)</td></tr>
<tr><td></td><td>Cognitiva-Constructivista</td><td>Aprendizaje Acti-vo-Aprendizaje Basado en Problemas</td><td>Usemos el condicional para tomar decisiones de ejecución del programa aplicados a los contextos: riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y el análisis léxico del lenguaje ancestral.</td></tr>
<tr><td colspan="3">PRECONCEPTOS DE LOS ESTUDIANTES</td></tr>
<tr><td></td><td colspan="2">Se realiza un sondeo para determinar qué tanto conocen sobre parámetros en riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, análisis léxico de lenguaje ancestral, en el sentido de aplicar condicionales.</td></tr>
<tr><td colspan="3">OBJETIVOS</td></tr>
<tr><td></td><td colspan="2">Comprender la importancia de los condicionales en la toma de decisiones de ejecución de un programa frente a una situación del mundo real, teniendo en cuenta que ellas pueden evaluar expresiones lógicas para determinas su verdad o falsedad.</td></tr>
<tr><td colspan="3">ACTIVIDADES</td></tr>
<tr><td></td><td colspan="2">3 sesiones de clase planeada así:
La primera sesión de exploración de los preconceptos del algebra de Boole y tablas de verdad, donde se permita abordar el tema de las expresiones lógicas en dicho escenario. En esta sesión, el profesor indica a través de ejemplos el uso de expresiones lógicas basadas y su uso a través de operadores lógicos. Una serie de ejercicios en clase deberán ser desarrollados por los estudiantes para afianzar el concepto. Dichos ejercicios tendrán relación directa con las necesidades del contexto manifestadas en este formato. Se debe fundamentar en el uso de lógica matemática. Se usarán expresiones de menor complejidad al inicio y se irá avanzando su nivel de complejidad en los ejercicios finales.
La segunda parte de programación de condicionales a través de un lenguaje de programación. Mediante depuradores se explora cómo se bifurca la ejecución del programa dada una situación de evaluación de expresión lógica. Se desarrollan programas simples de uso de constantes, variables y expresiones aritmético-lógicas y condicionales.
La tercera sesión se planea para la programación de avances del proyecto de curso donde se integre el concepto de condicionales</td></tr>
<tr><td colspan="3">RECURSOS</td></tr>
<tr><td></td><td colspan="2">Lenguaje de programación / depurador (depende del lenguaje de programación).</td></tr>
</table>

Puntos focales	**Experiencia en enseñanza**		
		Las reflexiones sobre la práctica de enseñanza se orientarán hacia: Facilidad de representar evaluación de parámetros simples del mundo real por parte del estudiante usando condicionales, entendiendo la potencia de los programas para cambiar su ejecución ante una determinada situación.	
	Lenguaje en contexto		

		Todas las definiciones de condicionales estarán inmersas en el vocabulario de las situaciones descritas en las necesidades del contexto. Por ejemplo: (cálculo de magnitud Richter)
		if (magnitud >= 1.0 && magnitud <= 1.9)
		{
		cout << “es un microtemblor”;
		}
		else
		{
		// seguir anidando los rangos de la escala
		}	

	Reto de enseñanza		
		Lograr que los estudiantes representen su entorno a través de reglas de parametrización expresadas en condicionales. Lograr reconocer la importancia de representar la realidad y los posibles usos de dichas representaciones.	
	Reto de aprendizaje		
		Los estudiantes buscan material audiovisual en la Web sobre el uso de condicionales en un lenguaje determinado. Los estudiantes comparten entre sí el material encontrado, hacer uso de un foro de discusión sobre lo encontrado.	
	Referencias bibliográficas		
		… (Varía según el lenguaje de programación)	
	Eventos motivacionales		
		Lluvia de ideas para la exploración con los estudiantes sobre los posibles usos de los condicionales en los problemas del contexto. *Gamification* aplicada al concepto de condicional (juego de laberinto / *maze game*) usando la metáfora del juego de laberinto, se resuelven una serie de expresiones lógicas a través de un laberinto, las expresiones lógicas se toman de una lista, a cada celda del tablero le corresponde una expresión lógica compleja. El equipo que falla el resultado lógico regresa una posición en el laberinto. El juego termina cuando algún equipo llegue a la última posición del tablero.	
	Valoración de aprendizaje		
		Valoración del objeto y técnicas de evaluación	OBJETO DE EVALUACIÓN (4ta Transposición)
		Autoevaluación Uso de rubricas diseñadas por el docente donde se aplicará la revisión por pares. Grabación audiovisual de bitácora donde los estudiantes explican por qué es importante el concepto de condicional en su proyecto de curso.	El conjunto de condicionales que serán utilizadas en el desarrollo del proyecto de curso. Revisión del *dossier* de ejercicios
	Trabajo independiente		
		Fragmentos de lecturas complementarias que relacionen condicionales al riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, análisis léxico de lenguaje ancestral. *Dossier* de ejercicios de declaración de condicionales para los contextos mencionados. Proyecto de curso	

Fuente: esta investigación.

Estrategia de enseñanza propuesta para el objeto de saber: ciclo

Tabla 20. Estrategia de enseñanza propuesta para ciclo

FORMATO DE DISEÑO DE ESTRATEGIAS DE ENSEÑANZA BASADA EN TRANSPOSICIÓN DIDÁCTICA *IN EXTENSA SENSU*			
OBJETO DE SABER: CICLO (1ra Transposición)			
Un ciclo es una estructura algorítmica que permite la iteración (repetición) de una o más instrucciones del programa a partir de un punto de partida (inicio), la evaluación de una expresión lógica (comparación de finalización) y el progreso de este a través de incrementos o decrementos de variable de control. Dentro del desarrollo teórico, independientemente del lenguaje de programación, es requerida la fundamentación en álgebra Booleana; las expresiones dependen de la sintaxis del lenguaje de programación. (Tomado de ontología computacional, p. 93 de este documento)			
Interacción en noosfera	**Integrantes de noosfera**		
		Profesores de fundamentos de programación Director del Departamento, Área o Carrera Personal del sector productivo y la industria	
	Necesidades del contexto local		
		A través de entrevistas, los profesores del curso de fundamentos de programación indagan las necesidades del contexto local con el personal del sector productivo y la industria. Junto con el director de departamento, se articulan dichas necesidades al plan o proyecto de formación institucional, determinando su correspondencia con la misión y visión institucional. Para esta propuesta, las necesidades del contexto local alrededor del concepto de ciclo pueden ser (Obviamente, dependerán de cada contexto), por ejemplo: Riesgo volcánico – procesamiento de señales sísmicas de trama continua Producción agrícola – algoritmos de cálculo de precio sombra Producción acuícola - algoritmos de cálculo de precio sombra Producción ganadera - algoritmos de cálculo de precio sombra Expresiones artísticas, cultura del carnaval, poesía, música – lingüística de *corpus* aplicada al lenguaje ancestral	
	CONTENIDO		
		OBJETO A ENSEÑAR (2da Transposición)	**Encapsulamiento de objeto**
		El ciclo es una estructura de control de ejecución controlado por una expresión lógica o booleana, el ciclo tiene la capacidad de repetir un número controlado de veces un grupo de instrucciones del programa. Los ciclos requieren usar al menos una variable de control con una instrucción que inicie su valor, otra que compare su valor actual con el fin esperado y otra que determine el cambio de dicha variable de control.	Atributos: Valor inicial Expresión lógica de fin de ciclo Declaración de cambio (incremento o decremento, o asignación del usuario) Operaciones: Iterar(…)

ESTRATEGIA DE ENSEÑANZA				
Dimensión conceptual	**TEORÍA**			
		Pedagogía	**Didáctica**	**OBJETO DE ENSEÑANZA (3ra Transposición)**
		Cognitiva-Constructivista	Aprendizaje Activo-Aprendizaje Basado en Problemas	Usemos el ciclo para poder repetir en forma controlada la ejecución de instrucciones del programa aplicados a los contextos: riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y sobre lingüística de corpus de lenguaje ancestral.
	PRECONCEPTOS DE LOS ESTUDIANTES			
		Se realiza un sondeo para determinar qué tanto conocen sobre algoritmos de iteración en riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y de lingüística de corpus de lenguaje ancestral, en el sentido de aplicar ciclos.		
	OBJETIVOS			
		Comprender la importancia de los ciclos en la iteración de instrucciones de un programa frente a una situación del mundo real, teniendo en cuenta que ellas pueden ejecutar en forma controlada fragmentos del programa.		
	ACTIVIDADES			
		4 sesiones de clase planeada así: La primera sesión de exploración de los tipos de ciclos. Se usarán expresiones de menor complejidad al inicio y se irá avanzando su nivel de complejidad en los ejercicios finales. Realizar estructuras cíclicas básicas, recorridos, incrementos, decrementos, sumatorias, productorias, etc. La segunda parte de programación de ciclos a través de un lenguaje de programación. Mediante depuradores se explora cómo se iteran las instrucciones en la ejecución del programa dada una situación de evaluación de expresión lógica. Se desarrollan programas simples de uso de constantes, variables y expresiones aritmético-lógicas, condicionales, y ciclos. La tercera y cuarta sesión se planea para la programación de avances del proyecto de curso donde se integre el concepto de ciclos.		
	RECURSOS			
		Lenguaje de programación / depurador (depende del lenguaje de programación).		

Puntos focales			
	Experiencia en enseñanza		
		Las reflexiones sobre la práctica de enseñanza se orientarán hacia: Facilidad de representar iteraciones del mundo real por parte del estudiante usando ciclos, entendiendo la potencia de las computadoras para cambiar ejecutar instrucciones velozmente.	
	Lenguaje en contexto		

		Todas las definiciones de ciclos estarán inmersas en el vocabulario de las situaciones descritas en las necesidades del contexto. Por ejemplo: (procesamiento de señales sísmicas)
		`for (int i = 0; i <= 100; i++)`
		`{`
		`  if (frecuencia[i] <= 0.1) y = 2.5 * x;`
		`  else y = 3.5 * x – 10;`
		`}`	

	Reto de enseñanza		
		Lograr que los estudiantes representen su entorno a través de estructuras iterativas expresadas en ciclos. Lograr reconocer la importancia de representar los fenómenos de realidad y los posibles usos de dichas representaciones.	
	Reto de aprendizaje		
		Los estudiantes buscan material audiovisual en la Web sobre el uso de ciclos en un lenguaje determinado. Los estudiantes comparten entre sí el material encontrado, hacer uso de un foro de discusión sobre lo encontrado.	
	Referencias bibliográficas		
		... (Varía según el lenguaje de programación)	
	Eventos motivacionales		
		Lluvia de ideas para la exploración con los estudiantes sobre los posibles usos de los ciclos en los problemas del contexto. *Gamification* aplicada al concepto de ciclo (juego de monopolio / *monopoly game*) usando la metáfora del juego de monopolio, se tiene un tablero circular con un dado para avanzar, cada casilla recrea la declaración de un ciclo, los jugadores deberán saber el resultado del cálculo del ciclo y su número de iteraciones, acertando el reto se tiene la oportunidad de un segundo lanzamiento, errando el reto, pierde un turno. Al lanzar el dado, si cae un número par se avanza dicha cantidad, si cae un número impar se retrocede dicha cantidad, el juego termina cuando un jugador llegue a la casilla final.	
	Valoración de aprendizaje		
		Valoración del objeto y técnicas de evaluación	**OBJETO DE EVALUACIÓN (4ta Transposición)**
		Autoevaluación Uso de rubricas diseñadas por el docente donde se aplicará la revisión por pares. Grabación audiovisual de bitácora donde los estudiantes explican por qué es importante el concepto de ciclo en su proyecto de curso.	El conjunto de ciclos que serán utilizadas en el desarrollo del proyecto de curso. Revisión del *dossier* de ejercicios
	Trabajo independiente		
		Fragmentos de lecturas complementarias que relacionen condicionales al riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, análisis léxico de lenguaje ancestral. *Dossier* de ejercicios de declaración de condicionales para los contextos mencionados. Proyecto de curso	

Fuente: esta investigación.

Estrategia de enseñanza propuesta para el objeto de saber: función

Tabla 21. Estrategia de enseñanza propuesta para función

<table>
<tr><td colspan="4">FORMATO DE DISEÑO DE ESTRATEGIAS DE ENSEÑANZA BASADA EN TRANSPOSICIÓN DIDÁCTICA IN EXTENSA SENSU</td></tr>
<tr><td colspan="4">OBJETO DE SABER: FUNCIÓN (1ra Transposición)</td></tr>
<tr><td colspan="4">Una función es una parte del programa que realiza tareas específicas y tiene la facilidad de ser llamado en diferentes instancias dentro del programa; pueden manejar parámetros de entrada y pueden devolver valores de salida. (Tomado de ontología computacional, p. 94 de este documento)</td></tr>
<tr><td rowspan="8">Interacción en noosfera</td><td colspan="3">Integrantes de noosfera</td></tr>
<tr><td></td><td colspan="2">Profesores de fundamentos de programación
Director del Departamento, Área o Carrera
Personal del sector productivo y la industria</td></tr>
<tr><td colspan="3">Necesidades del contexto local</td></tr>
<tr><td></td><td colspan="2">A través de entrevistas, los profesores del curso de fundamentos de programación indagan las necesidades del contexto local con el personal del sector productivo y la industria. Junto con el director de departamento, se articulan dichas necesidades al plan o proyecto de formación institucional, determinando su correspondencia con la misión y visión institucional.
Para esta propuesta, las necesidades del contexto local alrededor del concepto de función pueden ser (Obviamente, dependerán de cada contexto), por ejemplo:
Riesgo volcánico – aplicación de transformadas de Fourier
Producción agrícola – aplicación de método de simplex de optimización
Producción acuícola - aplicación de método de simplex de optimización
Producción ganadera - aplicación de método de simplex de optimización
Expresiones artísticas, cultura del carnaval, poesía, música – construcción de servicio de lexicón para definición de términos del lenguaje ancestral</td></tr>
<tr><td colspan="3">CONTENIDO</td></tr>
<tr><td></td><td>OBJETO A ENSEÑAR (2da Transposición)</td><td>Encapsulamiento de objeto</td></tr>
<tr><td></td><td>La función constituye una parte del programa que ha sido diseñada para atender una situación específica (por ejemplo, un cálculo especial, una acción determinada). Las funciones, además de dividir la funcionalidad de un programa, tienen la ventaja de poder ser invocadas las veces que sea necesario; esta circunstancia permite su reutilización y de esta manera se potencia la modularización de un programa, haciendo más fácil su mantenimiento.</td><td>Atributos:
Argumentos o parámetros de entrada
Valor de devolución
Declaración de la firma
Cuerpo de la función
Operaciones:
Invocar(…)</td></tr>
</table>

<table>
<tr><td colspan="4">ESTRATEGIA DE ENSEÑANZA</td></tr>
<tr><td rowspan="11">Dimensión conceptual</td><td colspan="3">TEORÍA</td></tr>
<tr><td></td><td>Pedagogía</td><td>Didáctica</td><td>OBJETO DE ENSEÑANZA (3ra Transposición)</td></tr>
<tr><td></td><td>Cognitiva-Constructivista</td><td>Aprendizaje Activo-Aprendizaje Basado en Problemas</td><td>Usemos la función para dividir la funcionalidad de un programa aplicados a los contextos: riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y sobre lexicón de lenguaje ancestral.</td></tr>
<tr><td colspan="3">PRECONCEPTOS DE LOS ESTUDIANTES</td></tr>
<tr><td></td><td colspan="3">Se realiza un sondeo para determinar qué tanto conocen sobre métodos generales en riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y de lexicones de lenguaje ancestral, en el sentido de aplicar funciones.</td></tr>
<tr><td colspan="3">OBJETIVOS</td></tr>
<tr><td></td><td colspan="3">Comprender la importancia de las funciones en la división funcional del programa frente a una situación del mundo real, teniendo en cuenta que ellas pueden ser invocadas y reutilizadas por programa varias veces.</td></tr>
<tr><td colspan="3">ACTIVIDADES</td></tr>
<tr><td></td><td colspan="3">5 sesiones de clase planeada así:
La primera sesión de exploración de los tipos de funciones. Se realizará el uso de parámetros. Realizar definiciones de función básicas, sin retorno de valor y sin argumentos, sin retorno de valor y con argumentos, con retorno de valor y sin argumentos, y con retorno de valor y con argumentos.
La segunda parte de programación de funciones a través de un lenguaje de programación. Mediante depuradores se explora cómo se salta la ejecución de un programa dada una invocación a función. Se desarrollan programas simples de uso de constantes, variables y expresiones aritmético-lógicas, condicionales, ciclos, y funciones.
La tercera, cuarta y quinta sesión se planea para la programación de avances del proyecto de curso donde se integre el concepto de funciones.</td></tr>
<tr><td colspan="3">RECURSOS</td></tr>
<tr><td></td><td colspan="3">Lenguaje de programación / depurador (depende del lenguaje de programación).</td></tr>
</table>

Puntos focales

Experiencia en enseñanza	
	Las reflexiones sobre la práctica de enseñanza se orientarán hacia: Facilidad de representar funcionalidad del mundo real por parte del estudiante usando funciones, entendiendo la potencia de los programas para reutilizar su código fragmentado.
Lenguaje en contexto	

	Todas las definiciones de funciones estarán inmersas en el vocabulario de las situaciones descritas en las necesidades del contexto. Por ejemplo: (análisis de Fourier)
	`void` CSignal::Fourier(`int` tf)
	{
	`real` *TF = m_pprTransformada[tf];
	`int` c = tf * (m_iLongVent - m_iSupVent);
	...
	}

Reto de enseñanza	
	Lograr que los estudiantes representen su entorno a través de funciones usando el principio "Divide y Vencerás". Lograr reconocer la importancia de representar los fenómenos de realidad y los posibles usos de dichas representaciones a través de funciones de programas.
Reto de aprendizaje	
	Los estudiantes buscan material audiovisual en la Web sobre el uso de funciones en un lenguaje determinado. Los estudiantes comparten entre sí el material encontrado, hacer uso de un foro de discusión sobre lo encontrado.
Referencias bibliográficas	
	... (Varía según el lenguaje de programación)
Eventos motivacionales	
	Lluvia de ideas para la exploración con los estudiantes sobre los posibles usos de las funciones en los problemas del contexto. *Gamification* aplicada al concepto de función (juego de monopolio / *monopoly game*) usando la metáfora del juego de monopolio, se tiene un tablero circular con un dado para avanzar, cada casilla recrea la declaración de una función, los jugadores deberán saber el resultado del cálculo de la función, acertando el reto se tiene la oportunidad de un segundo lanzamiento, errando el reto, pierde un turno. Al lanzar el dado, si cae un número par se avanza dicha cantidad, si cae un número impar se retrocede dicha cantidad, el juego termina cuando un jugador llegue a la casilla final.
Valoración de aprendizaje	

	Valoración del objeto y técnicas de evaluación	**OBJETO DE EVALUACIÓN (4ta Transposición)**
	Autoevaluación Uso de rubricas diseñadas por el docente donde se aplicará la revisión por pares. Grabación audiovisual de bitácora donde los estudiantes explican por qué es importante el concepto de función en su proyecto de curso.	El conjunto de funciones que serán utilizadas en el desarrollo del proyecto de curso. Revisión del *dossier* de ejercicios

Trabajo independiente	
	Fragmentos de lecturas complementarias que relacionen funciones al riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y lexicón de lenguaje ancestral. *Dossier* de ejercicios de declaración de funciones para los contextos mencionados. Proyecto de curso

FORMATO DE DISEÑO DE ESTRATEGIAS DE ENSEÑANZA BASADA EN TRANSPOSICIÓN DIDÁCTICA *IN EXTENSA SENSU*

OBJETO DE SABER: FUNCIÓN (1ra Transposición)

Una función es una parte del programa que realiza tareas específicas y tiene la facilidad de ser llamado en diferentes instancias dentro del programa; pueden manejar parámetros de entrada y pueden devolver valores de salida. (Tomado de ontología computacional, p. 94 de este documento)

Interacción en noosfera

Integrantes de noosfera	
	Profesores de fundamentos de programación Director del Departamento, Área o Carrera Personal del sector productivo y la industria
Necesidades del contexto local	
	A través de entrevistas, los profesores del curso de fundamentos de programación indagan las necesidades del contexto local con el personal del sector productivo y la industria. Junto con el director de departamento, se articulan dichas necesidades al plan o proyecto de formación institucional, determinando su correspondencia con la misión y visión institucional. Para esta propuesta, las necesidades del contexto local alrededor del concepto de función pueden ser (Obviamente, dependerán de cada contexto), por ejemplo: Riesgo volcánico – aplicación de transformadas de Fourier Producción agrícola – aplicación de método de simplex de optimización Producción acuícola - aplicación de método de simplex de optimización Producción ganadera - aplicación de método de simplex de optimización Expresiones artísticas, cultura del carnaval, poesía, música – construcción de servicio de lexicón para definición de términos del lenguaje ancestral

CONTENIDO

	OBJETO A ENSEÑAR (2da Transposición)	**Encapsulamiento de objeto**
	La función constituye una parte del programa que ha sido diseñada para atender una situación específica (por ejemplo, un cálculo especial, una acción determinada). Las funciones, además de dividir la funcionalidad de un programa, tienen la ventaja de poder ser invocadas las veces que sea necesario; esta circunstancia permite su reutilización y de esta manera se potencia la modularización de un programa, haciendo más fácil su mantenimiento.	Atributos: Argumentos o parámetros de entrada Valor de devolución Declaración de la firma Cuerpo de la función Operaciones: Invocar(...)

ESTRATEGIA DE ENSEÑANZA				
Dimensión conceptual	**TEORÍA**			
		Pedagogía	**Didáctica**	**OBJETO DE ENSEÑANZA (3ra Transposición)**
		Cognitiva-Constructivista	Aprendizaje Activo-Aprendizaje Basado en Problemas	Usemos la función para dividir la funcionalidad de un programa aplicados a los contextos: riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y sobre lexicón de lenguaje ancestral.
	PRECONCEPTOS DE LOS ESTUDIANTES			
	Se realiza un sondeo para determinar qué tanto conocen sobre métodos generales en riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y de lexicones de lenguaje ancestral, en el sentido de aplicar funciones.			
	OBJETIVOS			
	Comprender la importancia de las funciones en la división funcional del programa frente a una situación del mundo real, teniendo en cuenta que ellas pueden ser invocadas y reutilizadas por programa varias veces.			
	ACTIVIDADES			
	5 sesiones de clase planeada así: La primera sesión de exploración de los tipos de funciones. Se realizará el uso de parámetros. Realizar definiciones de función básicas, sin retorno de valor y sin argumentos, sin retorno de valor y con argumentos, con retorno de valor y sin argumentos, y con retorno de valor y con argumentos. La segunda parte de programación de funciones a través de un lenguaje de programación. Mediante depuradores se explora cómo se salta la ejecución de un programa dada una invocación a función. Se desarrollan programas simples de uso de constantes, variables y expresiones aritmético-lógicas, condicionales, ciclos, y funciones. La tercera, cuarta y quinta sesión se planea para la programación de avances del proyecto de curso donde se integre el concepto de funciones.			
	RECURSOS			
	Lenguaje de programación / depurador (depende del lenguaje de programación).			

Puntos focales			
	Experiencia en enseñanza		
	Las reflexiones sobre la práctica de enseñanza se orientarán hacia: Facilidad de representar funcionalidad del mundo real por parte del estudiante usando funciones, entendiendo la potencia de los programas para reutilizar su código fragmentado.		
	Lenguaje en contexto		

	Todas las definiciones de funciones estarán inmersas en el vocabulario de las situaciones descritas en las necesidades del contexto. Por ejemplo: (análisis de Fourier)
	`void CSignal::Fourier( int tf )`
	`{`
	`  real *TF = m_pprTransformada[tf];`
	`  int c = tf * (m_iLongVent - m_iSupVent);`
	`  ...`
	`}`		

	Reto de enseñanza		
	Lograr que los estudiantes representen su entorno a través de funciones usando el principio “Divide y Vencerás”. Lograr reconocer la importancia de representar los fenómenos de realidad y los posibles usos de dichas representaciones a través de funciones de programas.		
	Reto de aprendizaje		
	Los estudiantes buscan material audiovisual en la Web sobre el uso de funciones en un lenguaje determinado. Los estudiantes comparten entre sí el material encontrado, hacer uso de un foro de discusión sobre lo encontrado.		
	Referencias bibliográficas		
	... (Varía según el lenguaje de programación)		
	Eventos motivacionales		
	Lluvia de ideas para la exploración con los estudiantes sobre los posibles usos de las funciones en los problemas del contexto. *Gamification* aplicada al concepto de función (juego de monopolio / *monopoly game*) usando la metáfora del juego de monopolio, se tiene un tablero circular con un dado para avanzar, cada casilla recrea la declaración de una función, los jugadores deberán saber el resultado del cálculo de la función, acertando el reto se tiene la oportunidad de un segundo lanzamiento, errando el reto, pierde un turno. Al lanzar el dado, si cae un número par se avanza dicha cantidad, si cae un número impar se retrocede dicha cantidad, el juego termina cuando un jugador llegue a la casilla final.		
	Valoración de aprendizaje		
	Valoración del objeto y técnicas de evaluación		**OBJETO DE EVALUACIÓN (4ta Transposición)**
	Autoevaluación Uso de rubricas diseñadas por el docente donde se aplicará la revisión por pares. Grabación audiovisual de bitácora donde los estudiantes explican por qué es importante el concepto de función en su proyecto de curso.		El conjunto de funciones que serán utilizadas en el desarrollo del proyecto de curso. Revisión del *dossier* de ejercicios
	Trabajo independiente		
	Fragmentos de lecturas complementarias que relacionen funciones al riesgo volcánico, producción agrícola, producción acuícola, producción ganadera, y lexicón de lenguaje ancestral. *Dossier* de ejercicios de declaración de funciones para los contextos mencionados. Proyecto de curso		

Fuente: esta investigación.

7. EPÍLOGO

Finalizada esta investigación, en la que se revisó una literatura específica y se realizó un análisis de la información recolectada, se diseñaron una serie de actividades que contribuyen a la creación de prácticas de enseñanza en fundamentos de programación de computadoras; prácticas que se sustentan en una visión extendida de transposición didáctica.

Es importante definir el grupo de personas que conforma la noosfera. Los profesores y directores de programas —o su equivalente— deberán estar presentes; adicionalmente, es de gran ayuda que algunos profesionales del sector productivo —la industria— puedan estar presentes, en la medida de lo posible, junto a personas con experiencia en educación y pedagogía. Aquí, se sustenta que tener un gran nivel de conocimientos de la disciplina no es suficiente para proponer los objetos a enseñar, que posteriormente se convertirán en objetos de enseñanza. Adicional a los conocimientos disciplinares, hay más aspectos que considerar para articular los objetos a enseñar dentro de un escenario educativo, situado dentro de un contexto específico, haciendo uso de estrategias de enseñanza apropiadas.

A partir de la revisión sistemática de literatura, los modelos de aprendizaje activo y de aprendizaje basado en problemas, proponen actividades que son muy recomendables en el marco de esta propuesta; no obstante, ésta no se limita exclusivamente a la integración de las actividades de dichos modelos. Se puede articular las actividades propias de cualquier otro modelo de aprendizaje a esta propuesta.

Adicionalmente, los escenarios de aprendizaje donde los estudiantes tienen la oportunidad de interactuar en la sesión de clase son desafiantes al tenor de esta propuesta, debido a la complejidad de la definición de acciones que permiten mantener el interés del estudiante cuando forma parte activa en las dinámicas del aula. El reto precisamente está en la creatividad y la imaginación del profesorado para motivar a los estudiantes del curso

de fundamentos de programación. En esta investigación, este tipo de reto se superó a través del concepto de evento motivacional.

De acuerdo con lo construido al interior de este documento, los recursos educativos juegan un papel importante para soportar las estrategias de enseñanza. Sin embargo, es precisamente la motivación la que nutre el interés del estudiantado hacia el aprendizaje. Los estudiantes motivados son individuos que tienen un interés marcado en seguir adelante con su proceso de aprendizaje. En este sentido, los eventos motivacionales de los que habla la propuesta de esta investigación pretenden crear una atmósfera cálida y positiva a fin de enfocar esfuerzos hacia el aprendizaje.

Experiencias interesantes se pueden sumar a esta propuesta al incorporar nuevas competencias dentro de los eventos motivacionales, mediante el análisis crítico de código fuente escrito por los compañeros en un ambiente de colaboración competitiva. Estas situaciones han dado buenos resultados en lo referente a motivación para el aprendizaje de acuerdo con las experiencias descritas por sus autores.

Por otra parte, la transposición didáctica contribuye a reducir la complejidad de abstracción de los objetos de saber. Con esto, su enfoque extendido —la transposición didáctica *In Extensa Sensu*— contribuye a la planeación de estrategias de actividades propias de la enseñanza en escenarios de educación en ciencias de la computación; en el caso particular de esta investigación: de los fundamentos de programación de computadoras.

Al reducir la complejidad de abstracción, las personas que intervienen en la noosfera prácticamente ayudan a construir puentes que conectan el saber erudito con los estudiantes que pretenden aprender. Así, la transposición didáctica entra en acción para tales fines.

La transposición didáctica *In Extensa Sensu* constituye el aporte principal de esta investigación, definiéndose como el constructo teórico sobre cómo transponer los objetos de saber experto en el campo de las ciencias computacionales, con una noosfera de contexto e involucrando un nuevo objeto a partir del objeto de enseñanza, como es el objeto de evaluación.

El objeto de evaluación entra en escena como agente de la valoración permanente del aprendizaje, el cual tiene incorporado como principio fundamental la revisión crítica por parte del profesor acerca de su naturaleza y la retroalimentación correspondiente por parte de los estudiantes. El objetivo final es el aprendizaje de los estudiantes; ergo, el objeto de evaluación tiene una alta responsabilidad para lograr tal objetivo. El objeto de evaluación surge derivado de la última transposición del objeto de enseñanza al tenor de esta propuesta y representa quizás un reto mayor para quienes participan de su diseño y despliegue, dado que el objeto de evaluación cierra el círculo de transposición, pero permitiendo siempre un espacio a posibles mejoras.

Como complemento al aporte de conocimiento, se hace un reconocimiento a las técnicas de lingüística computacional, con el propósito de contribuir al análisis de gran cantidad de información del orden textual y que ha sido recolectada por diferentes mecanismos, y que incluso, se presenta en diferentes idiomas provenientes de una heterogeneidad de contextos. La constitución de un *corpus* lingüístico es vital para la aplicación de dichas técnicas. Adicionalmente, el uso de esquemas preconceptuales es muy apropiado para la representación de ontologías computacionales que permiten organizar el conocimiento. Dentro del desarrollo de esta investigación, los esquemas preconceptuales y las técnicas de lingüística computacional permitieron culminar exitosamente los objetivos específicos planteados.

REFERENCIAS

ACOFI (2005). Marco de fundamentación conceptual especificaciones de prueba de estado en ingeniería de sistemas versión 6.0. Asociación Colombiana de Facultades de Ingeniería. Colombia.

ACM, AIS & IEEE Computer Society (2005). Computing Curricula 2005. ACM SIGCSE Bulletin, Vol. 34. [online] http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf. 02-03-2014.

ACM & IEEE Computer Society (2008). Computer Science Curriculum 2008: An interim revision of CS 2001. ACM's home page, http://www.acm.org//education/curricula/ComputerScience2008.pdf, 11-03-2015.

ACM & IEEE Computer Society (2013). Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science [online] disponible en: www.acm.org/education/CS2013-final-report.pdf. 28-01- 2014, USA.

Aebli, H. (1988). Factores de la enseñanza que favorecen el aprendizaje autónomo. Madrid, Narcea.

Ali, A., & Mensch, S. (2008). Issues and challenges for selecting a programming language in a technology update course. Proceedings of the Information Systems Education Conference, Phoenix, AZ 2008. http://isedj.org/isecon/2008/020/index.html. 28-09-2013.

Andrzejewska, M. (2018). 'Factors Affecting of Computer Science Students' Failure in an Introductory-level Programming Courses'. Repozytorium Uniwersytetu Rzeszowskiego (UR). Poland: Wydawnictwo Uniwersytetu Rzeszowskiego.

Anewalt, K. (2008). Making CS0 fun: An active learning approach using toys, games and Alice. Journal of Computing Sciences in Colleges, 23(3), 98-105.

Anijovich, R., & Mora, S. (2010). Estrategias de enseñanza. Otra mirada del quehacer en el aula.

Attaway, S. (2013). MATLAB: A Practical Introduction to Programming and Problem Solving, 3rd Edition. Butterworth-Heinemann. USA.

Ben-Ari, M. (1998). Constructivism in computer science education. Proceedings of the 29th SIGCSE Technical. Symposium on Computer Science Education, 257-261.

Benedito, V. (1987). Introducción a la Didáctica. Fundamentación teórica y diseño curricular. Barcelona: Barcanova.

Báez, L., & Chacón, L. (2013). Student-Teachers' Teaching Techniques: Actors in Pupils' Extrinsic Motivation as They Speak (Técnicas de enseñanza de los docentes practicantes: actores en la motivación extrínseca de los estudiantes a la hora de hablar). PROFILE: Issues in Teachers' Professional Development, 15(2), 69-84.

Balderas A., Ruiz-Rube I., Dodero J.M., Palomo-Duarte M., Berns A. (2013). 'A Generative Computer Language to Customize Online Learning Assessments'. Lecture Notes in Computer Science, vol 8095. Springer, Berlin, Heidelberg.

Baldwin, L.P., & Kulijas, J. (2001). Learning programming using program visualization techniques. Proceedings of the 34th Hawaii International Conference on System Sciences – 2001. http://www.computer.org/portal/. 29-09-2013.

Blanchette, I., & Dunbar, K. (2000). How analogies are generated: The roles of structural and superficial similarity. Memory and Cognition, 28, 108-124.

Bogdanovych, A., & Trescak, T. (2017). 'Teaching Programming Fundamentals to Modern University Students'. In Proceedings of the 8th International Conference on Computer Supported Education (CSEDU 2016), 2(1), 308-317.

Braught, G., Wahls, T., & Marlin, L. (2011). 'The Case for Pair Programming in the Computer Science Classroom', ACM Transactions on Computing Education, 11(1), New York, USA.

Breed, J., & Taylor, E. (2013). 'Learning paradigms for educating a new generation of computer science students'. International Journal of Educational and Pedagogical Sciences, 7(4), 861-869.

Brickley, D. & Guha, R. (1999). Resource Description Framework (RDF) Schema Specification. Proposed Recommendation, World Wide Web Consortium: http://www.w3.org/TR/PR-rdf-schema. 14-05-2015.

Brousseau, G. (1986). Fondements et méthodes de la didactique des mathématiques. Recherches en didactique des mathématiques, 7, 2, pp. 33-115.

Brown, D. (2000). Principles of language learning and teaching (4th ed.). New York, NY: Addison Wesley. Longman.

Bruera, R. (1985). La didáctica como ciencia cognitiva, España, Editorial: C.E.D.I.E.

Campos, A. (2009). Métodos mixtos de investigación. Integración de la investigación cuantitativa y la investigación cualitativa. Cooperativa Editorial Magisterio.

Caraballo, S. & Cicala, R. (2006). Vers une Didactique de l'Informatique, Enseignement de l'Informatique et des Technologies de l'Information et de la Communication, Disponible en : https://www.epi.asso.fr/revue/editic.htm#a0601c_esp. 18-03-2014.

Cardelli, J. (2004). Reflexiones críticas sobre el concepto de transposición didáctica de Chevallard. Cuadernos de Antropología Social, 19 (1), pp. 49-61.

Carnegie Mellon University (2013). Alice's Home Page, An Educational Software that teaches students computer programming in a 3D environment. http://www.alice.org/index.php?page=what_is_alice/what_is_alice. 28-09-2013

Carter, L. (2006). 'Why Students with an Apparent Aptitude for Computer Science Don't Choose to Major in Computer Science', SIGCSE '06, March 1–5 2006, pp.27-31, Houston (TX), USA, http://www.softwarebyrob.com/assets/whynotcs.pdf. 05-07-2013.

Carter, J., & Jenkins, T. (2002). Gender differences in programming? Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education. http://www.acm.org/dl. 28-09-2013.

Carvalho, S., Flores, E., Ramos, F., Batista, L., & Pereira, M. (2014). 'Hybrid Intelligent Tutoring System With Didactic Transposition Of The Subjects Guided By Expert Knowlegdment And Self Organizing Maps Neural Network', IEEE Latin America Transaction, 12(8), 1539–1544.

Chevallard, Y. (1985). La transposition didactique. Du savoir savant au savoir enseigné. Grenoble, La Pensée Sauvage.

Chevallard, Y. (1988). 'On didactic transposition theory: Some introductory notes', International Symposium on Research and Development in Mathematics, Bratislava, Checoslavaquia.

Clancy, M., Titteron, N., Ryan, C., Slotta, J., & Linn, M. (2003). New roles for students, instructors, and computers in a lab-based introductory programming course. Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, 132-136.

Contreras, F. (2013). Vigilancia Epistemológica. Horizonte de la Ciencia 3 (5), diciembre 2013. FE-UNCP/ISNN 2304 – 4330, pp. 39-43.

Comenio, J.A. (1640). Didáctica magna, Edición 2012, España, Ed. Bruguera.

Cuéllar, L., Pérez, R. & Quintanilla, M. (2005). La propuesta de Ernest Rutherford en los libros de texto en Colombia. Un análisis desde la his-

toria de las ciencias y la visión de Transposición Didáctica en ellos. Enseñanza de las ciencias, número extra. VII congreso.

D'Amore, B. & Fandiño, M. (2001). Un acercamiento analítico al 'triángulo de la didáctica'. Educación Matemática, Vol. 14 No. 1 abril 2002, pp. 48-47.

Dann, W., Copper, S., & Pausch, R. (2006). Learning to program with Alice. Upper Saddle River, NJ: Prentice Hall.

Darós, W. (1987). Diversas bases de una teoría didáctica. Revista de ciencias de la educación: Órgano del Instituto Calasanz de Ciencias de la Educación, 130, Argentina, Editorial Calasanz, 215-226.

De Camilloni, A. (2004). Corrientes didácticas contemporáneas, España, PAIDOS IBERICA.

De Chardin, P. T. (1955). Le phénomène humain. France, París: Editions du Seuil.

Díaz Barriga, A. (1998). La investigación en el campo de la didáctica. Modelos históricos, Perfiles Educativos, enero-junio, 79/80, México, UNAM.

Díaz, A., & Hernández, G. (1999). Estrategias docentes para un aprendizaje significativo. Diplomado en Informática para la enseñanza de la medicina. México: McGraw-Hill.

Díaz, H. (1998). La tutoría académica en la educación universitaria Documento de Trabajo, Diplomado en Didáctica Universitaria, Universidad de Medellín.

Dunican, E. (2002). Making the analogy: Alternative delivery techniques for first year programming courses. In J. Kuljis, L. Baldwin & R. Scoble (Eds), Proceedings de the 14th Workshop of the Psychology of Programming Interest Group, Brunel University, Junio 2002, 89-99.

European Commission. (2018). European Credit Transfer and Accumulation System (ECTS). Disponible [online] https://ec.europa.eu/education/resources/european-credit-transfer-accumulation-system_en. 25-04-2014.

Fernández, J. (1974). Didáctica, Universidad Nacional de Educación a Distancia, España, UNED.

Friedman, F. & Koffman, E. (1978). Teaching problem solving and structured programming in FORTRAN, Computers & Education, 2(3), 235-245.

Fulton, S., & Schweitzer, D. (2011). 'Impact of Giving Students a Choice of Homework Assignments in an Introductory Computer Science Class'.

International Journal for the Scholarship of Teaching and Learning, 5(1), 1-12.

Gadamer, H. (1984). Verdad y método: fundamentos de una hermenéutica filosófica. Salamanca: Sígueme.

Gadamer, H. (2008). Philosophical Hermeneutics. Ed. University of California Press; 30th Anniversary Edition.

Gallego, R. (2004). Un concepto epistemológico de modelo para la didáctica de las ciencias experimentales. Revista Electrónica de Enseñanza de las Ciencias, 3(3), pp. 301-319.

García, J., & Cañal, P. (1995). ¿Cómo enseñar? Hacia una definición de enseñanza por investigación. Investigación En La Escuela, 25(12).

García, V. (1974). Diccionario de Pedagogía, España, Labor, French & European Publications, Inc.

Gerard, D. (1955). Emergence of the credit system in American Higher Education. AAUP bulletin. (41) pp.654.

Gimeno, J. (1985). Teoría de la enseñanza y desarrollo del currículum, Madrid, Anaya.

Gómez, M. (2005). La transposición didáctica – Historia de un concepto, Revista Latinoamericana de Estudios Educativos. Vol. 1, Julio - diciembre 2005, pp. 83-115.

Grisales-Franco, M. (2012). 'Aproximación histórica al concepto de didáctica universitaria'. Educ. Educ. 15(2), 203-218.

Gruber, T. (1993). A Translation Approach to Portable Ontology Specification. Knowledge Acquisition 5: 199-220.

Gruber, T., Liu, L., & Özsu, M. (2008). Ontology. Encyclopedia of Database Systems. Tamer Eds. Springer-Verlag.

Gruber T., & Olsen G. (1994). An ontology for engineering mathematics in principles of knowledge representation and reasoning. In: J Doyle, P Torasso, E Sandewall (eds). Proceedings of the 4th International Conference (KR '94): Bonn, Germany, May 24–27, 1994. Morgan Kaufmann Publishers, Burlington, Massachusetts, USA, 1994, 258–69.

Guibert, N., Girard, P., & Guittet, L. (2004). Example-based programming: A pertinent visual approach for learning to program. Proceedings of the Working Conference on Advanced Visual Interfaces, 358 – 361.

Guzdial, M., & Guo, P. (2014). The Difficulty of Teaching Programming Languages, and the Benefits of Hands-on Learning. Communications of the ACM, 57(7).

Habermas, J. (1984). A Theory of Communicative Action, Vol 1. Reason and rationalization of society (T. McCarthy, Trans.) Boston: Beacon Press.

Hartman, W., Nievergelt, J., & Reichert, R. (2001). Kara, finite state machines, and the case for programming as part of general education. Proceedings of the IEEE Symposium on Human-Centric Computing Languages and Environment (HCC01). http://www.computer.org/portal/. 29-09-2014.

Hazzan, O, Meerbaum-Salant, O, & Dubinsky, Y. (2010). 'Didactic transposition in computer science education', ACM Inroads, 1, 4, p. 33-37, Scopus, EBSCOhost.

Hendler, J. & McGuinness, D. (2000). The DARPA Agent Markup Language. IEEE Intelligent Systems 16(6): 67-73.

Herbert, C. (2007). An introduction to programming with Alice. Boston, Massachusetts: Course Technology.

Hudson, M., Förster, C., Rojas-Barahona, C., Valenzuela, M., Valdés, P., & Ramaciotti, A. (2013). Comparación de la efectividad de dos estrategias metodológicas de enseñanza en el desarrollo de la comprensión lectora en el primer año escolar. Perfiles Educativos, 35(140), 100–118. https://doi.org/https://doi.org/10.1016/S0185-2698(13)71824-5. 16-05-2014.

Humphreys, B. & Lindberg, D. (1993). The UMLS project: making the conceptual connection between users and the information they need. Bulletin of the Medical Library Association 81(2): 170.

Hunston, S. (2006). 'Corpus Linguistics'. Corpus Approaches to Idiom. Elsevier Ltd. Birmingham, UK, 234-248.

Husserl, E. (1991). La crisis de las ciencias europeas y la fenomenología trascendental. Una introducción a la filosofía fenomenológica (Traducción por Jacobo Muñoz), Editorial Crítica, Barcelona, España.

lanfrancesco, G. (2002). La evaluación de los aprendizajes en la educación colombiana. En Programa de Desarrollo Pedagógico Docente de la Universidad de Antioquia. Medellín: Universidad de Antioquia.

ICFES (2010). Orientaciones para el examen de Estado de calidad de la Educación Superior SABER PRO (ECAES) INGENIERIA DE SISTEMAS. Instituto Colombiano para el Fomento de la Educación Superior.

ICFES (2015). Guía de Orientación – Módulo de Diseño de Software – Saber PRO-2015. Ministerio de Educación Nacional – ICFES, Colombia.

Joyanes, L. (2008). Fundamentos de programación, 4th Edition. McGraw-Hill, España.

Kinsella, A., & Marcus, G. (2009). 'Evolution, Perfection, and Theories of Language', Biolinguistics, 3(2-3), pp. 186 – 212. www.psych.nyu.edu/gary/marcusArticles/Kinsellla%20Marcus%202009.pdf. 26-07-2014.

Larriba-Naranjo, L. (2001). La investigación de los modelos didácticos y de las estrategias de enseñanza. Enseñanza, (19), 73–88.

Larroyo, F. (1970). Didáctica general contemporánea, Porrúa.

Lerner, D. (2001). Leer y escribir en la escuela: lo real, lo posible y lo necesario. México: SEP / Fondo de Cultura Económica.

Lim, E. Y., Liu, J. K., & Lee, R. T. (2011). Knowledge Seeker - Ontology Modelling for Information Search and Management. [electronic resource]: A Compendium. Berlin, Heidelberg: Springer Berlin Heidelberg.

Lippman, S., Lajoie, J., & Moo, B. (2015). C++ Primer, 5th Edition. Pearson Education. USA.

Litwin, E. (2004). Prácticas con Tecnologías, Praxis Educativa, 8, Argentina.

Londoño, J. (2007). Evaluación de aprendizajes. Un asunto vital en la Educación Superior. Revista Lasallista De Investigación, 4(2), 50-58.

López, D. (2015). Estudio comparativo de estrategias de enseñanza en los departamentos de psicología de la universidad de El Salvador y universidad Francisco Gavidia". Universidad de El Salvador.

López-Sámano, V. (2017). Retos de la enseñanza actual. Boletín del Colegio Mexicano de Urología, 32(3), 45-46.

Maréchal, L., Barthod, C., Goujon, L., & Büssing, T. (2012). 'Design and development of a mechatronic infant torso simulator for respiratory physiotherapy learning', Mechatronics, 22(1), 55-64.

Marrero, W., & Settle, A. (2005). Testing first: Emphasizing testing in early programming courses. Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, 4-8.

Martínez, M. (2004). Ciencia y arte en la metodología cualitativa. México: Trillas.

Mattos, L. (1963). Compendio de didáctica general. Editorial Kapelusz, Buenos Aires.

Mello, I. (1974). El proceso didáctico, Volumen 126 de la Biblioteca de Cultura Pedagógica, Argentina, Moreno.

Miles, M., & Huberman, A. (1994). Qualitative data analysis: an expanded sourcebook. Sage, Thousand Oaks.

Ministerio de Educación Nacional (2008). Ser competente en tecnología: ¡una necesidad para el desarrollo! Serie Guías No. 30. Orientaciones generales para la educación en tecnología.

Ministerio de Educación Nacional (2013). SPADIES – Sistema para la prevención de la deserción de la Educación Superior, Deserción por Co-

horte según área, http://spadies.mineducacion.gov.co/spadies/consultas_predefinidas.html?2. 03-10-2013.

Misirli, A., & Komis, V. (2014). Robotics and programming concepts in early childhood education: A conceptual framework for designing educational scenarios. In Research on e-Learning and ICT in Education (pp. 99-118). Springer New York.

Morales-Cadena, G., Fonseca-Chávez, M., Valente-Acosta, B., & Gómez-Sánchez, E. (2017). La importancia de la motivación y las estrategias de aprendizaje en la enseñanza de la medicina. Anales De Otorrinolaringología Mexicana, 62(2), 97-107.

Morin, E. (2007). El método, 6ª Edición, Cátedra.

Muñoz, N. (2011). 'El estudio exploratorio. Mi aproximación al mundo de la investigación cualitativa', Investigación y Educación en Enfermería, 29(3), 492-499.

Nerici, I. (1973). Hacia una didáctica general dinámica 2da edición, Biblioteca de Cultura Pedagógica, Argentina, Kapelusz.

Nurhayati, S. (2014). Teacher's Perception on The Student Needs in Learning English and Its Impact to The Teaching Strategies at Pastry Student of SMKN 2 Trenggalek, Chapter II. Literature Review. MEAUS Dissertation. TULUNGAGUNG: STATE ISLAMIC INSTITUTE (IAIN).

Oliva, J. (2006). Actividades para la enseñanza/aprendizaje de la química a través de analogías. Revista Eureka Sobre Enseñanza Y Divulgación De Las Ciencias, 3(1), 104-114.

Orlich, D., Harder, R., Callahan, R., Trevisan, M., & Brown, A. (2010). Teaching Strategies: A Guide to Effective Instruction, 9th Edition. (Wadsworth, Ed.). Boston, MA: Cengage Learning.

Pérez, A. (1982). Lecturas de aprendizaje y enseñanza, 9, Colección "Por un nuevo saber", México, Grupo Cultural Zero.

Pérez, G. (2006). Teorías y Modelos Pedagógicos. Medellín: Departamento de Publicaciones FUNLAM.

Piteira, M. & Costa, C. (2013). Learning Computer Programming: Study of difficulties in learning programming, ISDOC'13 - Proceedings of the International Conference on Information Systems and Design of Communication, Lisbon, Portugal.

Porter, R., & Calder, P. (2004). Patterns in learning to program: An experiment? Proceedings of the Sixth Conference on Australasian Computing Education, 30, 241-246.

Puiggrós, A. (1990). Sujetos, Disciplina y Currículum. Galerna, Buenos Aires.

Price, C. & Spackman, K. (2000). SNOMED clinical terms. BJHC&IM-British Journal of Healthcare Computing & Information Management 17(3): 27-31.

Ramírez, R. (2008). La pedagogía crítica: Una manera ética de generar procesos educativos. Folios - Segunda época, 28, Segundo semestre de 2008, pp. 108-119.

Reynoso, L., Grosclaude, E., Sánchez, L., & Álvarez, M. (2013). Technicians and their learning styles preferences and cognitive processes of formal inferences. In Cognitive Informatics & Cognitive Computing (ICCI* CC), 2013 12th IEEE International Conference on (pp. 51-60). IEEE.

Real Academia Española. (2018). Diccionario de la Lengua Española. Disponible [online] http://dle.rae.es/?id=PwDQ7LY. 06-02-2017.

Ricœur, P. (1970). Freud and philosophy: An essay on interpretation. New Haven, Conn.: Yale University Press.

Rosales, C. (1988). Didáctica: núcleos fundamentales, Educación hoy. Estudios, España, Narcea.

Roselli, N. (2010). Comparación experimental entre tres modalidades de enseñanza mediadas informáticamente. Revista de Investigación Educativa, 28(2), 265–282.

Rushkoff, D. (2010). Program or be Programmed: Ten Commands for a Digital Age. OR Books, New York.

Schleiermacher, F. (1998). Hermeneutics and Criticism. Ed. Andrew Bowie - Cambridge University Press.

Schneider, D. (1999). An introduction to programming using Visual Basic 6.0. Upper Saddle, River, NJ: Prentice Hall.

Seacord, R. (2013). Secure Coding in C and C++. 2nd Edition. Addison-Wesley Professional. USA.

Sheard, J., & Hagan, D. (1998). Experiences with teaching object-oriented concepts to introductory programming students using C++. Technology of Object-Oriented Languages and Systems-TOOLS 24, IEEE Technology, 310-319.

Shuhidan, S., Hamilton, M., & D'Souza, D. (2011). Understanding Novice Programmer Difficulties via Guided Learning, ITiCSE'11 - Proceedings of the 16th Annual Conference on Innovative and Technology in Computer Science, Darmstadt, Germany.

Smith, B. et al. (2007). The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nat Biotech 25:1251–5.

Solis, D. (2011). Illustrated C# 2012, 4th Edition. Apress. USA.

Soloway, E. & Spohrer, J. (1989). Studying the Novice Programmer, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Sommerville, I. (2011). Software Engineering. Pearson Addison Wesley 7th Edition.

Stamouli, I., Doyle, E., & Huggard, M. (2004). Establishing structured support for programming students. Proceedings of the 34th ASEE/IEEE Frontiers in Education Conference, Savannah, GA.

Stocker (1964). Principios de didáctica moderna, Argentina, Kapelusz.

Stroustrup, B. (2014). Programming: Principles and Practice Using C++ 2nd Edition. Addison-Wesley Professional. USA.

Thomas, L., Ratcliffe, M., Woodbury, J., & Jarman, E. (2002). Learning styles and performance in the introductory programming sequence. Proceedings of 33rd SIGCSE Technical Symposium, 34, 33-37.

Tiberghien, A., Vince, J., & Gaidioz, P. (2009). Design-based research: case of a teaching sequence on mechanics. International Journal of Science Education, 31(17), 2275-2314.

Timarán, R. (2010). "Una lectura sobre deserción universitaria en estudiantes de pregrado desde la perspectiva de la minería de datos", Revista Guillermo de Ockham, Colombia, Editorial Bonaventuriana.

Titone, R. (1981). Metodología Didáctica, Biblioteca de Educación, Argentina, Series Rial, Ediciones, S.A.

Tomaschewsky, K. (1966). Didáctica general 9na edición, Colección pedagógica, Grijalbo.

Van Roy, P., Armstrong, J., Flatt, M., & Magnusson, B. (2003). The role of language paradigms in teaching programming. Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, 269-270.

Vasco, C. (1990). Reflexiones sobre pedagogía y didáctica, Pedagogía y Currículo, 4, Ministerio de Educación Nacional.

Vattimo, G. (1997). Beyond Interpretation – The Meaning of Hermeneutics for Philosophy, 8. Cambridge (UK): Cambridge University Press.

Vergara, L. (2009). Reflexiones alrededor del trabajo independiente en la educación superior colombiana. Educación Y Humanismo, 11(17).

Vernadsky, V. (2004). Biosphere and Noosphere / Biosphere and noosphere, Iris press, Moscow.

Verret, M. (1975). Le temps des études, Paris, Librairie Honoré Champion.

Vygotsky, L. (1987). Zone of Proximal Development. Mind on Society: the development of high psychological processes, Harvard University Press, Cambridge (MA), USA.

von Bertalanffy, L. (1976). Teoría general de los sistemas: fundamentos, desarrollo, aplicaciones. Serie Ciencia y Tecnología. Fondo de Cultura Económica. España.

Weller, M. (2005). Los Angeles Business Journal. Disponible [online] http://www.ndsu.edu/ndsu/nlillebe/tandl/teachingtips/motivating/motivate.html. 03-06-2016.

Wilson, A. (2017). Biosphere, Noosphere, Infosphere: Epistemo-Aesthetics and the Age of Big Data. Parallax, 23(2), 202. doi:10.1080/13534645.2017.1299297.

Wing, J.M. (2006). Computational Thinking. Communications of the ACM, 49(3), pp. 33-35.

Zabalza, M. (1990). Diseño y desarrollo curricular, Narcea.

Zabalza, M. (2003). Competencias docentes del profesorado universitario. Calidad y desarrollo profesional. Madrid, España: Narcea.

Zapata, C., Arango, F. & Gelbukh, A. (2006). "Pre-conceptual Schema: a UML Isomorphism for Automatically Obtaining UML Conceptual Schemas," Lecture Notes in Computer Science (Artificial Intelligence Bioinformatics), vol. 4293, no. 65, pp. 27–37.

Zapata, C. (2007). "Definición de un esquema preconceptual para la obtención automática de esquemas conceptuales de UML," Universidad Nacional de Colombia, sede Medellín.