ENSEÑANZA DE LOS FUNDAMENTOS DE PROGRAMACIÓN DE COMPUTADORAS Y TRANSPOSICIÓN DIDÁCTICA

JESÚS INSUASTI PORTILLA

UNIVERSIDAD DE NARIÑO
RUDECOLOMBIA
DOCTORADO EN CIENCIAS DE LA EDUCACIÓN
SAN JUAN DE PASTO
2019

ENSEÑANZA DE LOS FUNDAMENTOS DE PROGRAMACIÓN DE COMPUTADORAS Y TRANSPOSICIÓN DIDÁCTICA

Tesis doctoral de
JESÚS INSUASTI PORTILLA

Director
Dr. ROBERTO RAMÍREZ BRAVO
Universidad de Nariño, Colombia

Tutor Internacional
Dr. D. JUAN MANUEL DODERO BEARDO
Universidad de Cádiz, España

UNIVERSIDAD DE NARIÑO
RUDECOLOMBIA
DOCTORADO EN CIENCIAS DE LA EDUCACIÓN
SAN JUAN DE PASTO
2019

NOTA DE RESPONSABILIDAD

Las ideas y conclusiones aportadas en el siguiente trabajo son responsabilidad exclusiva del autor.

Artículo 1ro del Acuerdo No. 324 de octubre 11 de 1966, emanado del Honorable Consejo Directivo de la Universidad de Nariño.

Nota de aceptación:

Calificación: 5.0 / 5.0

Mención: Tesis Laureada

_____

_____

_____

_____

Luis Joyanes Aguilar, Ph.D.

_____

Sandra Cristina Riascos, Ph.D.

_____

Delio Gómez López, Ph.D.

San Juan de Pasto, 15 de noviembre de 2019

Agradecimientos

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 en esta ocasión, como doctor en educación.

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 desde la especialización, la maestría y el doctorado. Como director de mi investigación, el Dr. Roberto Ramírez Bravo 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 pasantía internacional, 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 mi pasantía internacional, 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 la entrevista a expertos. De igual manera, agradezco a los 9 profesores de: *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.* Agradezco también a los 4 profesores nacionales de: 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!

RESUMEN

La investigación titulada “ENSEÑANZA DE LOS FUNDAMENTOS DE PROGRAMACIÓN DE COMPUTADORAS Y TRANSPOSICIÓN DIDÁCTICA” presenta una propuesta enfocada a la enseñanza de los objetos de saber asociados a los conceptos de variables, constantes, expresiones, comparaciones, ciclos y funciones de los fundamentos de programación de computadoras en las disciplinas asociadas a la computación. La contribución doctoral contempla el desarrollo de una metodología que extiende el concepto de transposición didáctica a fin de potenciar la enseñanza en contexto, y a su vez, el aprendizaje en los estudiantes de dichos cursos.

Palabras Clave:

Transposición, Didáctica, Programación, Computadoras

ABSTRACT

The doctoral thesis called "TEACHING COMPUTER PROGRAMMING FUNDAMENTALS AND DIDACTIC TRANSPOSITION" shows a proposal focused on teaching the knowledge objects related to the concepts of variables, constants, expressions, comparisons, loops, and functions of the computer programming fundamentals in the disciplines associated with computing. The doctoral contribution contemplates the development of a methodology that extends the concept of didactic transposition to promote teaching in context and, in turn, the students' learning in such courses.

Keywords:

Didactic, Transposition, Programming, Computers.

TABLA DE CONTENIDO

PRÓLOGO 16

ESQUEMA DE LA TESIS 18

1. EL RETO INVESTIGATIVO 19

1.1. Descripción del problema 20

1.2. Formulación del problema 27

1.3. Preguntas orientadoras 27

1.4. Tesis 27

1.5. Objetivo general 28

1.6. Objetivos específicos 28

1.7. Justificación 29

2. FUNDAMENTACIÓN TEÓRICA 32

2.1. Antecedentes 32

2.2. Marco normativo 41

2.3. Contexto de la investigación 42

2.4. Base teórica-conceptual 44

2.4.1. Elementos fundamentales de la didáctica 45

2.4.2. Acerca de la transposición didáctica 48

2.4.3. Discusión sobre el sistema didáctico y la noosfera 52

2.4.4. La transposición didáctica en computación 57

2.4.5. Las disciplinas asociadas a la computación 59

2.4.6. Los fundamentos de programación de computadoras 64

3. EL DESARROLLO METODOLÓGICO 66

3.1. Clasificación de la investigación 68

3.2. Técnicas de recolección y técnicas de procesamiento 73
3.3. Unidades de análisis y unidades de trabajo 77
3.4. Recolección y procesamiento de información 86
4. OBJETOS POR TRANSPONER 88
4.1. Análisis de la revisión documental 91
4.2. Análisis de las encuestas a profesores 96
4.3. Conceptualización de los objetos de saber 103
5. NIVELES DE TRANSPOSICIÓN DIDÁCTICA 110
5.1. Análisis de las entrevistas a expertos 110
5.2. Acerca de la vigilancia epistemológica 111
5.3. Extendiendo la teoría de transposición didáctica 113
Transposición didáctica *In Extensa Sensu* 114
Experiencias a través del proyecto *Open Discovery Space* 124
Experiencias en educación superior 125
6. ESTRATEGIAS DE ENSEÑANZA 129
6.1. Aplicación de entrevista semi-estructurada a los profesores 129
6.2. Análisis hermenéutico concluyente 130
Experiencia en enseñanza 131
Lenguaje en contexto 132
Reto de enseñanza 133
Reto de aprendizaje 135
Referencias bibliográficas 137
Eventos motivacionales 138
Valoración de aprendizaje 139

Trabajo independiente 141

6.3. Estrategias de enseñanza propuestas 143

Definición de estrategia de enseñanza 146

Comparación entre estrategias de enseñanza 148

Estrategia de enseñanza propuesta para el objeto de saber: variable 153

Estrategia de enseñanza propuesta para el objeto de saber: constante 156

Estrategia de enseñanza propuesta para el objeto de saber: expresión aritmético-lógica 159

Estrategia de enseñanza propuesta para el objeto de saber: condicional 163

Estrategia de enseñanza propuesta para el objeto de saber: ciclo 166

Estrategia de enseñanza propuesta para el objeto de saber: función 169

7. EPÍLOGO 173

REFERENCIAS 176

ANEXO A – Instrumentos de recolección de información 190

ANEXO B – Validación de instrumentos 193

ANEXO C – Informe de pasantía internacional 195

ANEXO D – Publicación internacional de artículo según SCOPUS© 199

ANEXO E – Síntesis de respuestas de la encuesta a profesores 200

ANEXO F – Síntesis de respuestas de la entrevista semiestructurada a expertos 215

ANEXO G – Síntesis de respuestas de la guía de vigilancia epistemológica 221

ANEXO H – Síntesis de respuestas de la entrevista semi-estructurada a profesores 236

ANEXO I – Diseño y desarrollo didáctico para cada estrategia de enseñanza basada en Transposición Didáctica *in Extensa Sensu* 239

LISTA DE TABLAS

Tabla 1. Síntesis de definiciones y elementos de didáctica 46
Tabla 2. Matriz metodológica 76
Tabla 3. Técnicas de recolección y procesamiento de información 77
Tabla 4. Listado de profesores extranjeros a los que se envió la encuesta 79
Tabla 5. Estructura académica y cursos del top 50 del mundo 81
Tabla 6. Listado de profesores colombianos a los que se envió la encuesta 83
Tabla 7. Lenguajes de programación usados en fundamentos de programación 88
Tabla 8. Definiciones propuestas para los 6 conceptos seleccionados 95
Tabla 9. Codificación de profesores encuestados 97
Tabla 10. Puntos focales en la práctica de la enseñanza de los fundamentos de programación 130
Tabla 11. Definiciones acerca del concepto de estrategia de enseñanza 144
Tabla 12. Conceptos e interpretaciones 145
Tabla 13. Matriz de comparación de estrategias de enseñanza 149
Tabla 14. Estrategias de Enseñanza 150
Tabla 15. Síntesis de clasificación de teoría pedagógica y didáctica 151
Tabla 16. Estrategia de enseñanza propuesta para variable 153
Tabla 17. Estrategia de enseñanza propuesta para constante 156
Tabla 18. Estrategia de enseñanza propuesta para expresión aritmético-lógica 159
Tabla 19. Estrategia de enseñanza propuesta para condicional 163
Tabla 20. Estrategia de enseñanza propuesta para ciclo 166
Tabla 21. Estrategia de enseñanza propuesta para función 169

LISTA DE FIGURAS

Figura 1. Síntesis de conceptos asociados a la didáctica 47

Figura 2. El sistema didáctico 49

Figura 3. El sistema de enseñanza 50

Figura 4. Teoría de situaciones didácticas aplicadas al sistema didáctico 54

Figura 5. Disciplinas asociadas a la computación 61

Figura 6. Resultados de búsqueda en corpus lingüístico 92

Figura 7. Nubes de palabras a partir del corpus lingüístico 93

Figura 8. Ontología computacional a través del esquema preconceptual del objeto de saber: variable 104

Figura 9. Ontología computacional a través del esquema preconceptual del objeto de saber: constante 105

Figura 10. Ontología computacional a través del esquema preconceptual del objeto de saber: expresión aritmético-lógica 106

Figura 11. Ontología computacional a través del esquema preconceptual del objeto de saber: condicional 107

Figura 12. Ontología computacional a través del esquema preconceptual del objeto de saber: ciclo 108

Figura 13. Ontología computacional a través del esquema preconceptual del objeto de saber: función 109

Figura 14. Trasposición Didáctica *In Extensa Sensu* 114

Figura 15. Primera transposición por recomendaciones curriculares de ACM 116

Figura 16. Segunda transposición debido a la noosfera 118

Figura 17. Tercera transposición debido a la práctica de enseñanza 119

Figura 18. Cuarta transposición debido a la evaluación y la retroalimentación 122

Figura 19. Esquema preconceptual para la representación ontológica del concepto de Estrategia de Enseñanza 147

Figura 20. Formato de diseño de estrategias de enseñanza basado en Transposición Didáctica *In Extensa Sensu* 152

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

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 la 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, donde originalmente fue aplicada a la enseñanza de la matemática. Esta propuesta afirma que el conocimiento experimenta una serie de transformaciones al ser representado en objetos, los cuales 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 dicho objeto, el objeto a enseñar se transpone generando el objeto de enseñanza; dicho objeto de enseñanza es el 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 busca aportar 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 diseña y desarrolla una investigación dentro del paradigma cualitativo con un enfoque hermenéutico; dentro de los aspectos metodológicos, la definición de objetivos, preguntas orientadoras y categorías han permitido 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 son descritos en detalle, para llegar al epílogo de la investigación donde son sintetizados los aportes al conocimiento.

ESQUEMA DE LA TESIS

Investigación

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

OBJETIVO GENERAL:
Enriquecer experiencias de enseñanza en los fundamentos de programación de computadoras a través de la transposición didáctica.

OBJETIVOS ESPECÍFICOS:
1. Caracterizar los objetos seleccionados de saber de los fundamentos de programación de computadoras, de acuerdo con los referentes internacionales.
2. Analizar los niveles de transposición didáctica de los objetos seleccionados de saber de los fundamentos de programación de computadoras.
3. Proponer prácticas de enseñanza para los objetos seleccionados de saber transpuestos de los fundamentos de programación de computadoras.

JUSTIFICACIÓN:
- Altos niveles de abstracción
- Dificultades de enseñanza
- Deserción estudiantil
- Base fundamental en construcción de software

MARCO DE REFERENCIA:
- Antecedentes
- Marco normativo
- Contexto de investigación

BASE TEÓRICA-CONCEPTUAL:
- Didáctica
- Transposición didáctica
- Enseñanza en computación
- Fundamentos de programación

METODOLOGÍA DE INVESTIGACIÓN:
- Paradigma: Cualitativo.
- Enfoque: Hermenéutico.

Obj. Específico	Categoría	Técnica Recolección	Técnica Análisis
1	Características de objetos de saber	Revisión documental	Esquemas preconceptuales
		Encuestas	Análisis hermenéutico
2	Niveles de transposición didáctica	Guia de vigilancia epistemológica	Esquemas preconceptuales
			Análisis hermenéutico
3	Prácticas de enseñanza	Entrevista semiestructurada	Esquemas preconceptuales
			Análisis hermenéutico

PROFESORES

País	Universidad
	The University of Texas at Austin
	University of Illinois, Urbana-Champaign
	City University of Hong Kong
	Paris 13
	Swiss Federal Institute of Technology Zurich
	University of Science and Technology of China
	Northwestern University
	Huazhong University of Science and Technology
	The Ohio State University - Columbus
	Universidad Nacional de Colombia-Medellín
	Politécnico Grancolombiano-Bogotá D.C.
	Universidad de Nariño-Pasto
	Universidad de Medellin-Medellin

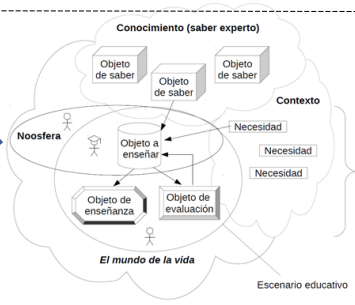
EXPERTOS

País	Universidad
	Universidad de Cádiz
	Universidad Nacional de Colombia-Medellín

TRANSPOSICION DIDACTICA EN "*Extensa Sensu*"

Contribución y propuesta

PROPUESTAS DE ESTRATEGIAS DE ENSEÑANZA DE EJEMPLO:
- Variable
- Constante
- Expresión aritmético-lógica
- Condicional
- Ciclo
- Función



Transposición Didáctica en *"Extensa Sensu"*

PUNTOS FOCALES:
- Experiencia en enseñanza
- Lenguaje en contexto
- Reto de enseñanza
- Reto de aprendizaje
- Referencias bibliográficas
- Eventos motivacionales
- Valoración de aprendizaje
- Trabajo independiente

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

ESTRATEGIA DE ENSEÑANZA

Dimensión Conceptual:
- 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

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. Descripción del problema

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) y presentan una tendencia similar en materia de 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 en mención (Anewalt, 2008; Porter & Calder, 2004).

El fenómeno de deserción de los estudiantes de dichos cursos 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 −donde están los programas relacionados con las disciplinas asociadas a la computación− se encuentran en un segundo lugar con un porcentaje promedio 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 Kulijas 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 constituyen 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 dicho programa en una computadora. En la práctica, la enseñanza del concepto de estructura no ha sido tan sencilla para los estudiantes que no estén familiarizados con la construcción de software. Adicionalmente, la evolución misma de los lenguajes de programación ha forzado a generar solidas estructuras a fin de entender cómo fueron codificados los programas de computadoras, y esto constituye un reto adicional en la enseñanza de los cursos en mención (Schneider, 1999). He aquí un factor que dificulta la enseñanza de 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 tales cursos no tienen experiencia previa en programación de computadoras; de hecho, la experiencia en programación 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 competencias 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 en programación de computadoras no parece ser un problema; sin embargo, si es un problema 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 dichos cursos 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 estudios universitarios; en este apartado, se evidencia que ese "salto abrupto" interfiere significativamente en los aprendizajes en los estudios universitarios.

Aunque el nivel de alfabetismo en TIC −Tecnologías de la Información y las Comunicaciones− puede ser alto entre algunos de los estudiantes que ingresan a carreras de las disciplinas 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 programar dicha tecnología a 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, como 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 desprecio en el esfuerzo por comprender los conceptos de que se abordan al interior 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 al interior de dichos cursos; al parecer, ésta 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 del ser enseñados. Se debe entonces realizar aportes a fin de enriquecer las prácticas de enseñanza de tales cursos, atendiendo la naturaleza abstracta de los objetos de saber experto, los cuales son llevados a las aulas de clase para ser enseñados.

1.2. Formulación del problema

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

1.3. Preguntas orientadoras

¿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?

1.4. Tesis

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

Esta tesis se sustenta en la posibilidad de motivar a los estudiantes 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 tal asunto 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.

Al desarrollar esta investigación, se pretende lograr una extensión de la teoría de transposición didáctica, donde se explorarán técnicas computacionales con el fin de soportar el método de investigación, a fin de poder proponer un modelo general que se pueda aplicar para el diseño de estrategias de enseñanza con los objetos de saber de los fundamentos de programación de computadoras.

1.5. Objetivo general

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

1.6. Objetivos específicos

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.

1.7. Justificación

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 dichas habilidades 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 problémicos 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 pretende 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 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 aprender programación 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 se constituye en 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 desarrollados 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 interés 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 al problema 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 prolifera. 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 si 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) señaló que para hacer más fácil los procesos de enseñanza-aprendizaje de los fundamentos de programación de computadoras, se debe mantener los siguientes elementos: minimizar la sintaxis de programación, proporcionar retroalimentación visual a la ejecución de los programas de computadoras, acortar el ciclo creativo de conceptualización, y mejorar la implementación y los resultados obtenidos tras la ejecución de un programa de computadora.

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) señalaron 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, se explicó 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 del estudiante de hoy.

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. 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, p. 1) 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". 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 objetos basados en el objeto persona −peso, talla− puede ser más comprensibles 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: 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) basaron 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 primeramente 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 visión 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 didacta de transponerlo para que dichos conceptos sean factibles de ser enseñados. En este aparte, Gallego afirma 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 didactas 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).

En este sentido, Gallego se interna en 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 didacta 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 solo 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. Marco normativo

Esta investigación se desarrolló en el contexto colombiano, pero con una alta participación de profesores extranjeros, los principales marcos de legitimidad sobre los saberes impartidos en las disciplinas asociadas a la computación son tomados a partir de los referentes conceptuales propuestos por ACM, IEEE Computer Society y AIS −*Asociation for Information Systems*−. No se trata de normas, sino del marco de referencia epistemológico para la organización de los saberes asociados a la computación.

En Colombia, el mayor aporte legislativo sobre la cual se basa la presente investigación es la ley 749 de 2002, por la cual se organiza el servicio público de la

educación superior en las modalidades de formación técnica profesional y tecnológica, donde se define la acción de la educación técnica, tecnológica y profesional asociada a las ciencias computacionales. Esta ley hace mención sobre los contenidos que deben ser abordados en un marco de competencias genéricas en el contexto nacional e internacional.

Por otra parte, la Ley 1324 de 2009 por la cual se fijan parámetros y criterios para organizar el sistema de evaluación de resultados de la calidad de la educación en Colombia, dicta normas para el fomento de una cultura de la evaluación, en procura de facilitar la inspección y vigilancia del estado y se transforma el ICFES. Esta ley es considerada relevante para la investigación dado que establece los lineamientos sobre qué evaluar, y en especial, demuestra el interés del Ministerio de Educación Nacional, por las pruebas estandarizadas Saber y otras pruebas como un instrumento de análisis de la evaluación externa e independiente para fomentar el mejoramiento continuo de la calidad de la educación y de las evaluaciones y su desarrollo. Estas pruebas sirven para evaluar oficialmente la educación formal impartida a quienes terminan el nivel de educación media y superior.

2.3. 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 menester indagar en diferentes contextos acerca de lo que ocurre en la enseñanza de dicho curso. Por tal razón, se pensó la investigación para ser realizada desde la Universidad de Nariño con la participación de profesores en diferentes universidades del escenario mundial.

Esta investigación toma como objeto de estudio la enseñanza de los fundamentos de programación dentro de cursos formales en el 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[1]. En adición, se incluyó también una muestra de universidades nacionales. Los criterios de selección se presentan 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 todos involucran la fundamentación en dicha área según los hallazgos al momento de analizar sus contenidos de estudio.

La 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 las 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

[1] 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.

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 abordan en el capítulo del 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, pero 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, además de la entrevista a dos expertos, uno de España y uno 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).

2.4. 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 basa la 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 dicha teoría. 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 presenta las definiciones y características de las disciplinas asociadas a la computación; estas disciplinas 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.

2.4.1. 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, practica 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 ha 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 de didáctica 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.

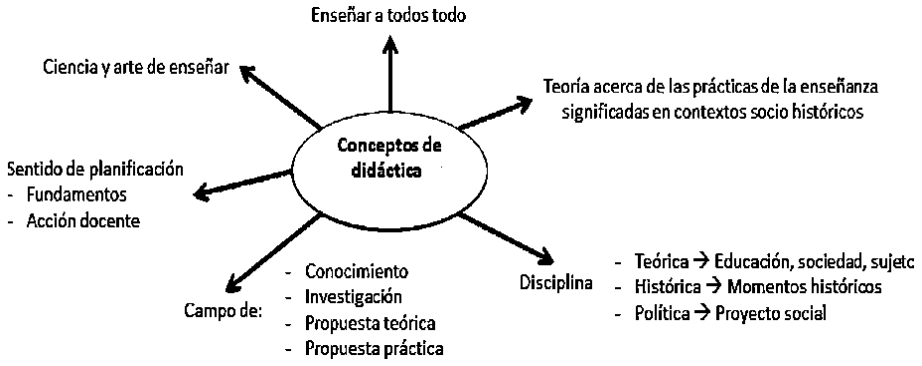


Figura 1. Síntesis de conceptos asociados a la didáctica (Fuente: esta investigación)

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, la propuesta de investigación pretende abordar el escenario principal de la enseñanza sin descuidar su incidencia en los aprendizajes de los estudiantes.

El concepto de didáctica se aborda 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.

2.4.2. 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 a 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. Al tenor de 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 llama 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 inmersos 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 incrustado en un ambiente con características específicas.

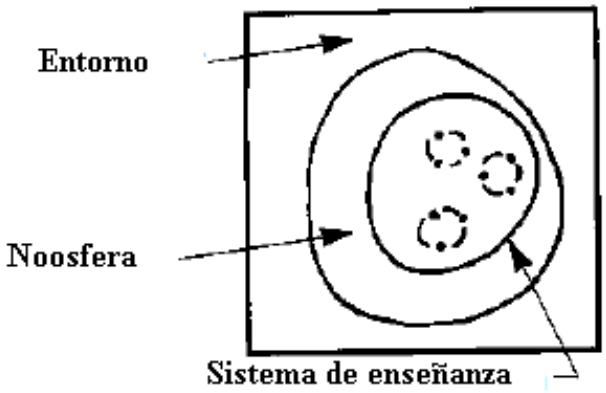


Figura 3. El sistema de enseñanza (Chevallard, 1985, p. 28**)**

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 literatura de esta investigación, la cual evidencia una producción académica incipiente cuando se trata de transposición didáctica aplicada a la computación.

Sobre la base de la teoría de Chevallard, esta investigación 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.

2.4.3. 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 Rio 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).

En 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 en el sentido de ser 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 como 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 de falta de profundidad 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, donde presentan 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.

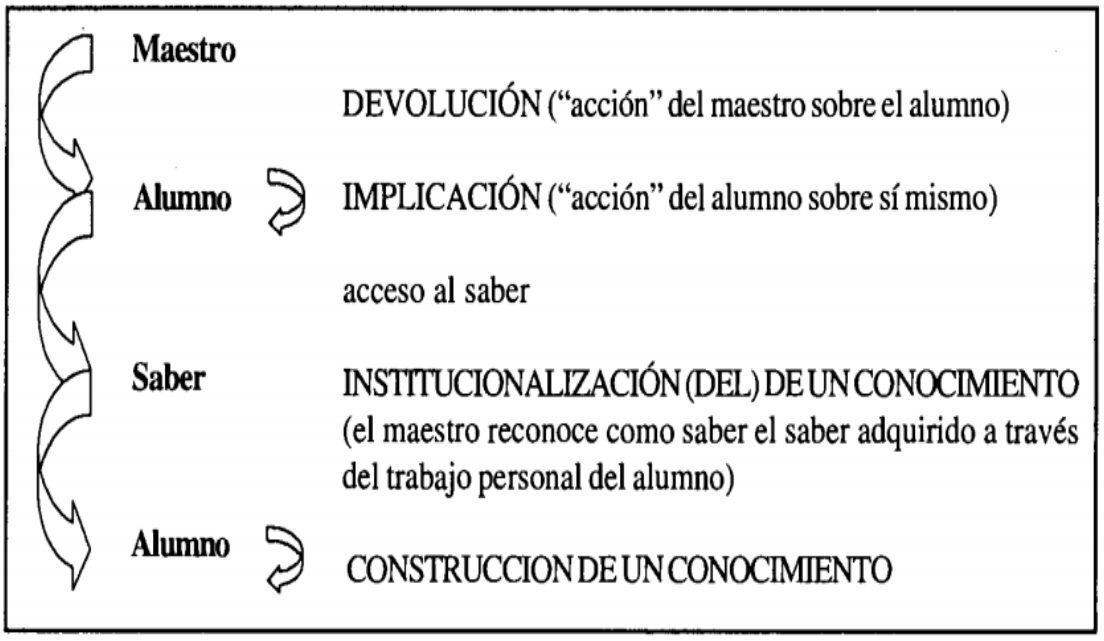


Figura 4. Teoría de situaciones didácticas aplicadas al sistema didáctico (D'Amore & Fandiño, 2001, p. 50**)**

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

En esta investigación, 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 Chevallard. Se pretende con esta investigación develar esas bondades en el contexto real y muy particular 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).

2.4.4. La transposición didáctica en computación

Pero ¿Cómo se debe entender la transposición didáctica? Para ello, es importante citar a su autor: Yves Chevallard. 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 (Chevallard, 1985).

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, dicha investigación la propone Hazzan, Meerbaum-Salant y Dubinsky (2010), Dicha investigación 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.

2.4.5. 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 que en Colombia se habla de 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, 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 la naturaleza discontinua de operación del software. 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:

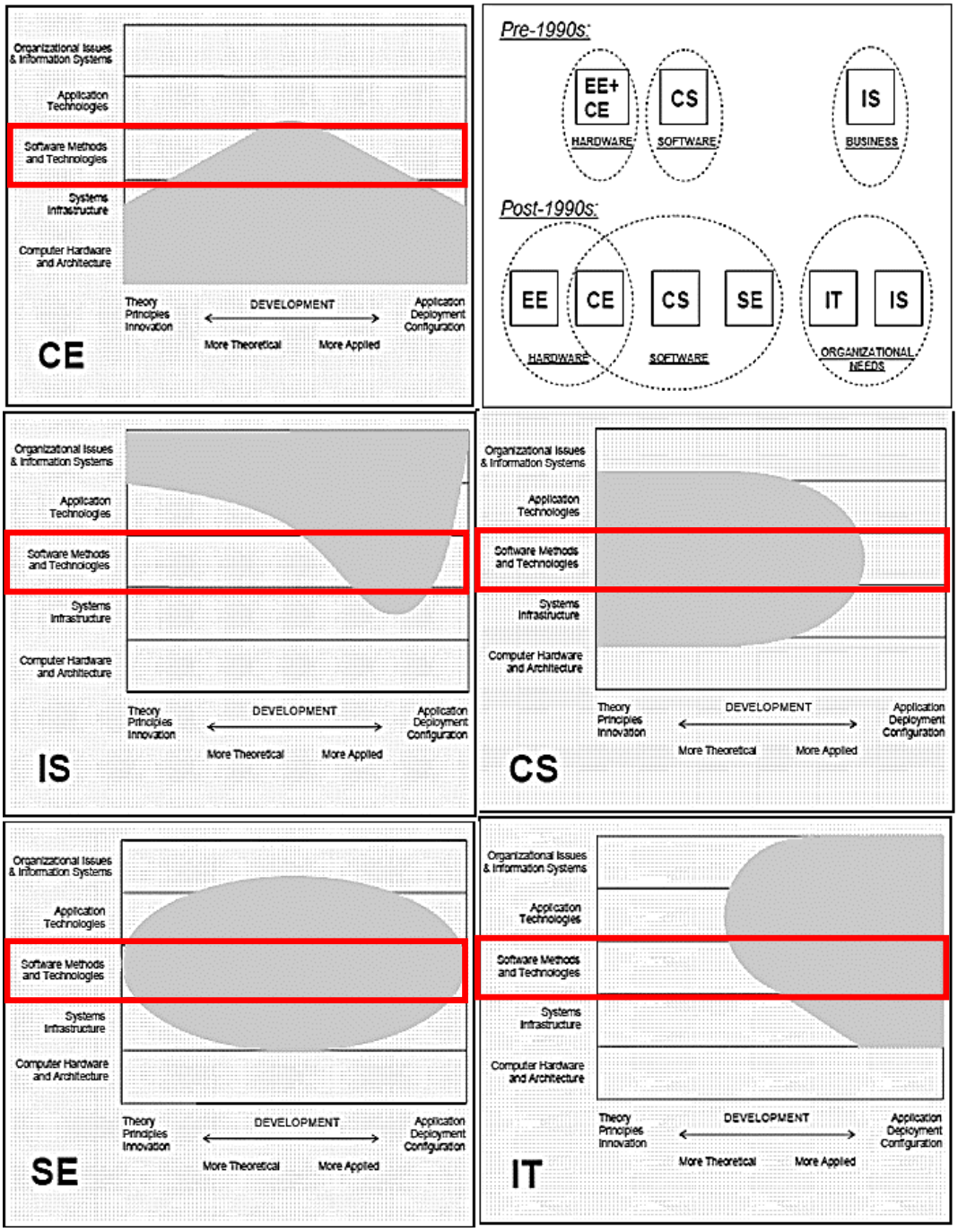


Figura 5. Disciplinas asociadas a la computación (Adaptado de ACM, AIS, & IEEE Computer Society, 2005**)**

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 de 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.

2.4.6. 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 notaciones 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 doctoral 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 la información. 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 participan 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 participan en la investigación se debe 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 la tesis 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. Clasificación de 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 por lo 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 dialogo de intersubjetividades (Campos, 2009).

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 puramente 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.e. 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 relacionado como la teoría de la interpretación, es decir, la teoría de lograr una comprensión de documentos, textos, enunciados, y así sucesivamente. La hermenéutica en este sentido tiene una larga historia, que se remonta al menos hasta la antigua Grecia. Sin embargo, en el período moderno, a raíz de La Reforma con su desplazamiento de la responsabilidad de interpretar la Biblia de la Iglesia a los cristianos en general, se le dio un nuevo enfoque. Este nuevo enfoque en la hermenéutica se produjo especialmente en Alemania. Los primeros aportes de Friedrich Daniel Ernst Schleiermacher a principios del siglo XIX ayudaron a su formalización, y luego floreció en la hermenéutica filosófica de Martin Heidegger y Hans-Georg Gadamer en el siglo XX.

La sistematización 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 en la interpretación de textos. De esta forma, se trata de la interpretación a fin de encontrar su sentido en profundidad. 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 relaciona su proyecto con la filosofía trascendental de Kant. Así como Kant intentó describir la posibilidad trascendental de las ciencias naturales, Gadamer intenta revelar la "posibilidad" de las ciencias humanas. De esta manera, Gadamer deja 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. para Vattimo (1997), 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 hace uso 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 de hermenéutica. En complemento a dicha visión, 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 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 de 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− en este escenario dándole su valor; sin embargo, el enfoque hermenéutico aplicado rescata el principio transformador habermasiano donde pretende brindar alternativas que suscite 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. Técnicas de recolección y técnicas de procesamiento

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 usa como técnicas de recolección de información la revisión documental guiada a través de fichas de revisión. De igual

forma, fue complementada la recolección de información a través de encuestas puntuales y entrevistas estructuradas a expertos, instrumentos que se relacionan en el Anexo A.

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, la cual se basa 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 a los profesores que deseen 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 manifiestan su intención de participar en la investigación, se les envía la encuesta y el formato de vigilancia epistemológica para su desarrollo. Tanto la encuesta como el formato de vigilancia epistemológica son formularios que se los diligencia 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. Es muy relevante para adquirir información del 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 asocian para 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 la información 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 internacionales?	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[2] 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)

[2] 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.

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[3] 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

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.

[3] 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 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, la entrevista fue realizada a una profesora titular con más de 20 años de experiencia quien 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 profesores extranjeros a los que se 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, solo 84 instituciones de educación 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; solo cuatro de los 84 docentes participaron en la encuesta. Similar a la aplicación de instrumentos a los profesores extranjeros, los nombres de los profesores solo 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 profesores colombianos a los que se 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**	**167**	**?. 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

Con esto se tiene que, de 48 invitaciones enviadas a los profesores extranjeros −dado que no se logró identificar a los profesores en 2 universidades extranjeras del 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 escenario 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 −los dos últimos expertos realizaron apreciaciones por escrito, las cuales se enviaron vía email, como consta en el Anexo B−.

3.4. Recolección y procesamiento de información

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, esta circunstancia permitió enfocar la atención tanto en cada *syllabus* como 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. Los conceptos son: variables, constantes, expresiones aritmético-lógicas, condicionales, ciclos y funciones. Independientemente de la metodología usada por el profesor, e independientemente 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 síntesis, 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 se enmarca en el estudio de la transposición didáctica en la enseñanza de los fundamentos de programación de computadoras, hace énfasis en los 6 conceptos: variable, constante, expresión aritmético-lógica, condicional, ciclo y función.

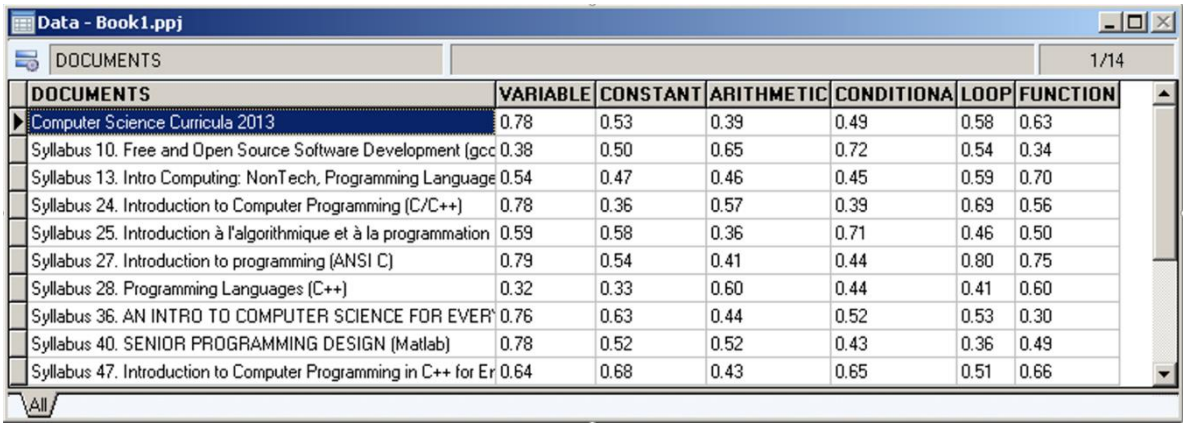
4.1. Análisis de la revisión documental

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 permitan organizar el conocimiento en dichas disciplinas.

El hecho de 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

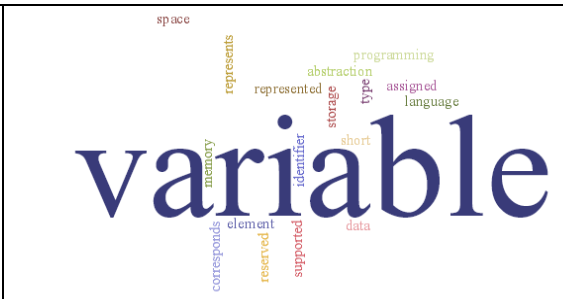
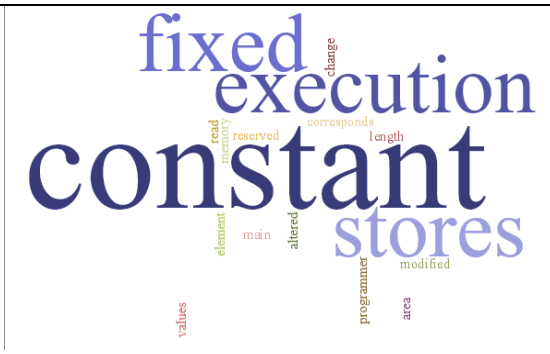
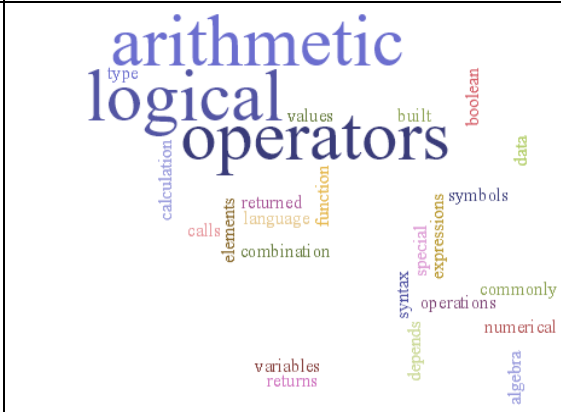
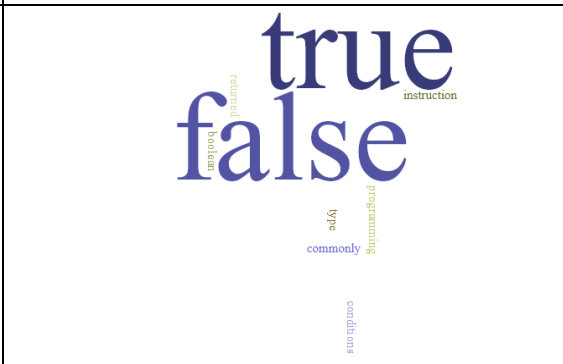
DOCUMENTS	VARIABLE	CONSTANT	ARITHMETIC	CONDITIONA	LOOP	FUNCTION
Computer Science Curricula 2013	0.78	0.53	0.39	0.49	0.58	0.63
Syllabus 10. Free and Open Source Software Development (gcc	0.38	0.50	0.65	0.72	0.54	0.34
Syllabus 13. Intro Computing: NonTech, Programming Language	0.54	0.47	0.46	0.45	0.59	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, los conceptos 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 produce 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*	

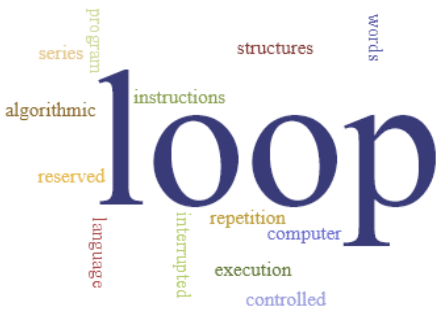
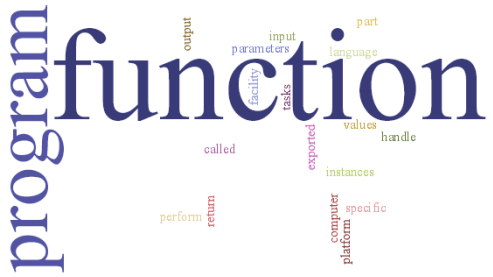
Concepto: *Loop* Palabras clave: *series, program, structures, words, algorithmic, instructions, reserved, language, interpreted, repetition, computer, execution, controlled*	
Concepto: *Function* Palabras clave: *part, output, input, parameters, program, called, facility, tasks, values, handle, exported,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

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

Según Hunston (2006), “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 puede 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 procede a realizar la construcción del concepto a partir del *corpus* lingüístico. A través de un ejercicio de interpretación de palabras −no todas fueron posibles de ser 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	Ciclo: un ciclo es una estructura

that allow the controlled repetition of a series of instructions within a computer program.	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. Análisis de las encuestas a profesores

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 ellos en escenarios reales. El ejercicio de la docencia es altamente valorado para esta investigación, por tal razón se preguntó sobre las estrategias motivacionales para enseñar dichos conceptos, se preguntó además por los aspectos más fáciles y los más difíciles de enseñarlos de acuerdo con cada profesor. El instrumento utilizado está disponible en el Anexo A y se denomin*a "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 evidencia del

proceso se consigna en el Anexo B. De igual manera, se relaciona el informe de la pasantía internacional en el Anexo C junto con la publicación internacional de un artículo asociado a la presente investigación en el Anexo D. 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 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 permite establecer la codificación de los 13 profesores encuestados. Con esta codificación se realizó el análisis de las respuestas a través de interpretación hermenéutica. Dicho análisis se presenta por medio de una serie de tablas discriminadas por cada concepto. A continuación, se realiza una tabulación de las respuestas de los profesores encuestados discriminadas por concepto. Todas las respuestas de las encuestas se encuentran compiladas en el Anexo E.

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 que pueden cambiar durante la ejecución de un programa de computadora. Como estrategias motivacionales, algunos profesores describen el uso de cajas donde objetos −en este caso valores− pueden estar contenidos en ellas. Los profesores en general reconocen sus limitaciones en el campo de la didáctica; de hecho, algunos reconocen que manejan una forma tradicional de enseñar, aunque tienden a reducir el concepto de pedagogía hacia meramente el acto de enseñar. Finalmente, no se resaltan 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 abordan 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 manifiestan 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 usan prácticamente recursos similares a los planteados para explicar el concepto de variable. De igual forma, no se hace explícita la dificultad al enseñar el concepto de variable exceptuando por direccionamiento de memoria y su nomenclatura en bajo nivel para el manejo. Adicionalmente, un profesor manifiesta que algunos de sus estudiantes no comprenden la utilidad de las constantes, dado que ellos argumentan que podrían usar simplemente variables a los cuales no se les afectará su valor a través de expresiones de asignación; aquí, el profesor lo expresa 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 sustenta 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 usan 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 éstos son abordados desde la matemática −en este particular, el profesor que trabaja con MATLAB® rescata el gran potencial de la herramienta−. De igual manera, la mayoría de los profesores asegura 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 reflejan su analogía donde un camino 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 manifiestas en la enseñanza del concepto se presentan al momento de usar la sintaxis propia de los lenguajes de programación; algunos lenguajes pueden hacer uso de contracciones de nomenclatura para expresar una estructura condicional; por otra parte, algunos profesores aseguran 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 reitera 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 argumentan que los ciclos son muy útiles en los procesos de simulaciones físicas; el hecho de poder hacer iteraciones controladas sobre un modelo matemático se constituye en la base de los procesos de simulación. El ciclo cobra importancia para tal razón. 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 reflejan uno de los mayores retos en su labor. Desde una perspectiva motivacional, los profesores argumentan su búsqueda por dividir un programa de computadoras en tareas puntuales; algunos de ellos se basan en el concepto de función matemática, otros lo orientan hacia funciones gráficas, otros hacia tareas específicas de simulación o incluso de videojuegos. Con respecto a las funciones, la mayoría de los profesores encuestados concuerda 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 argumentan 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 afirman que los saltos suelen perder la secuencialidad de la ejecución del programa en algunos estudiantes.

Con la información consignada en el Anexo E es 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 representado 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 basa 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 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. Las ontologías que se presentan a continuación constituyen una base de datos que describe los conceptos del dominio específico, y cómo los conceptos 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 para 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 usar 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 llega a formular los siguientes constructos teóricos, en forma de ontologías computacionales, a partir de las experiencias de enseñanza descritas por los profesores. Estos constructos teóricos surgen del análisis lingüístico, ellos contemplan diferentes opiniones de los profesores alrededor de cada concepto y se toma en cuenta también las definiciones que aparecen según la bibliografía correspondiente en los *syllabus*.

El término filosófico ontología fue primeramente adaptado a la computación por Gruber (1993) como una "especificación explícita de conceptualización" para la comunidad de Inteligencia Artificial. El 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 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

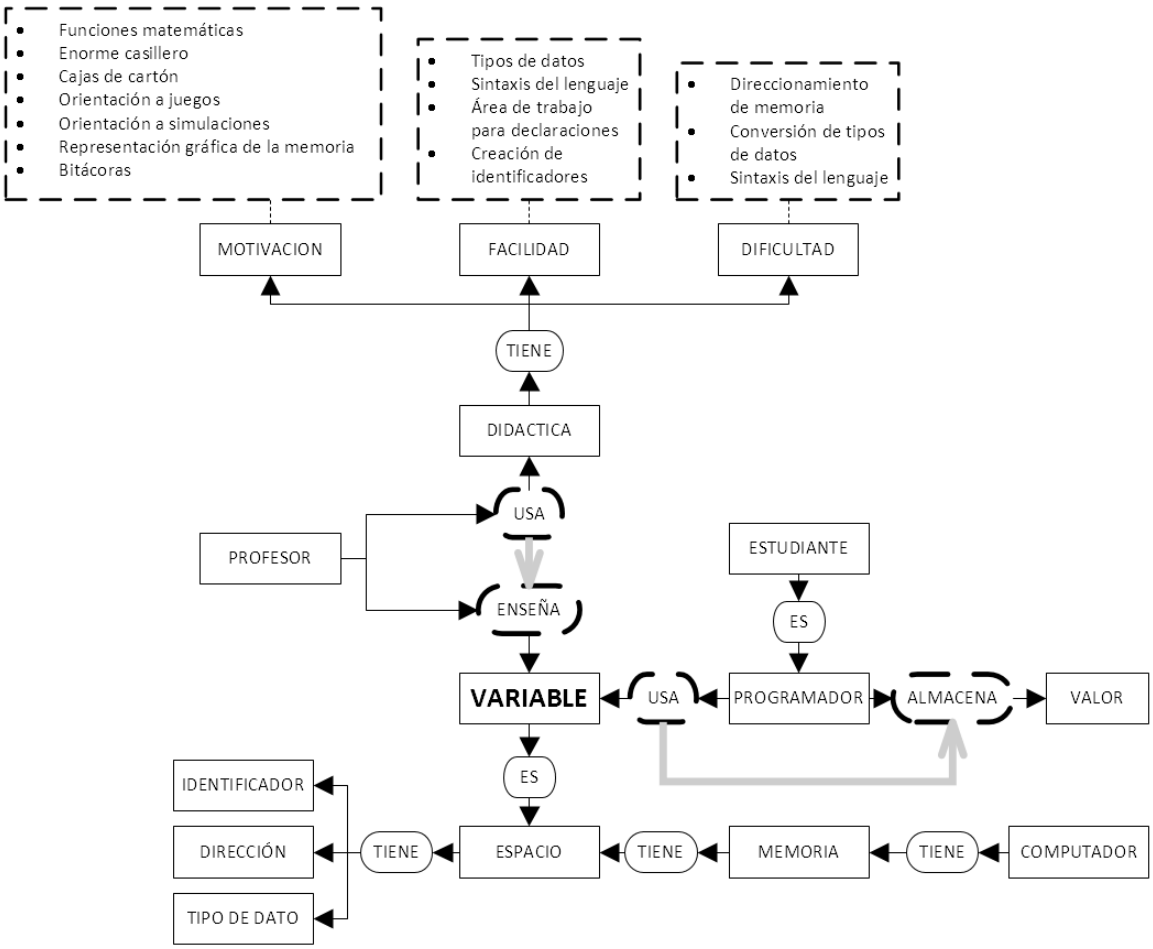


Figura 8. Ontología computacional a través del esquema preconceptual del objeto de saber: variable (Fuente: esta investigación)

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

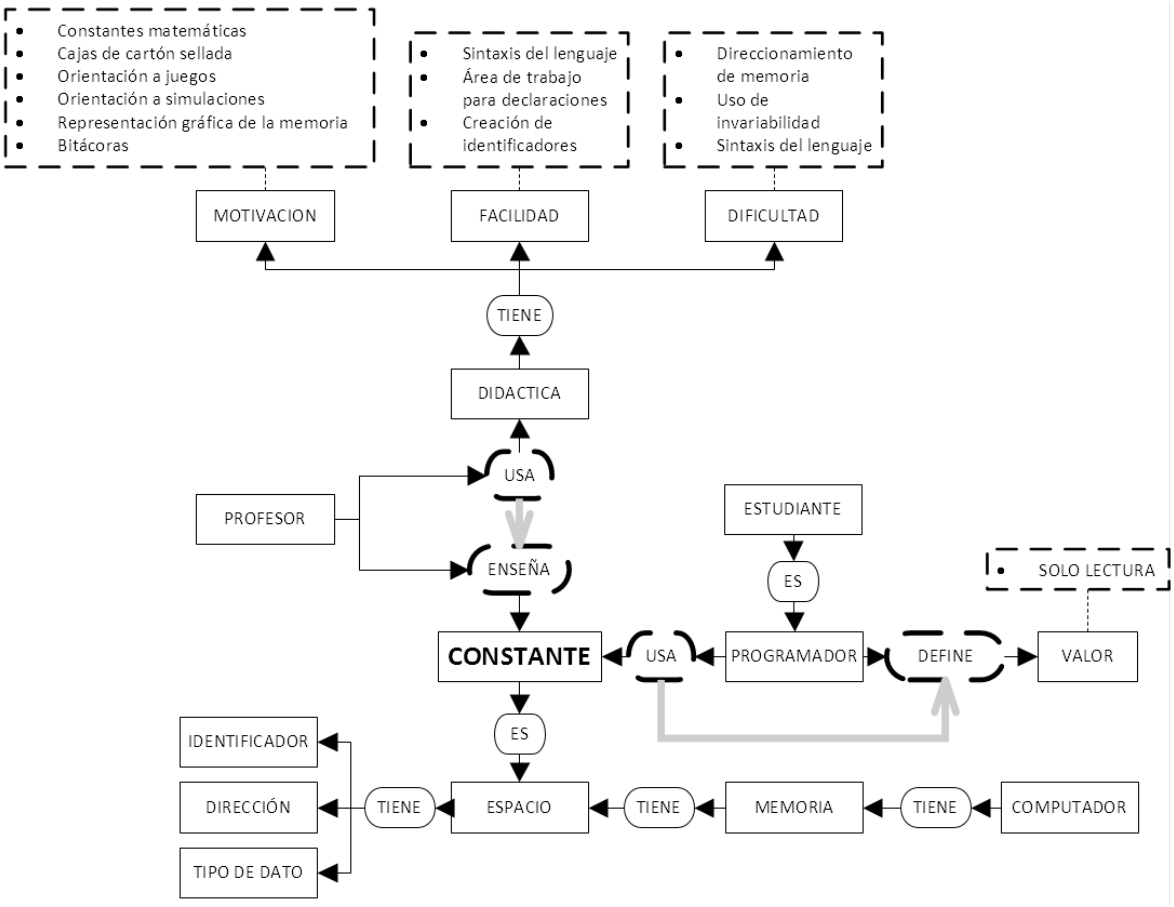


Figura 9. Ontología computacional a través del esquema preconceptual del objeto de saber: constante (Fuente: esta investigación)

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 aritmético-lógica

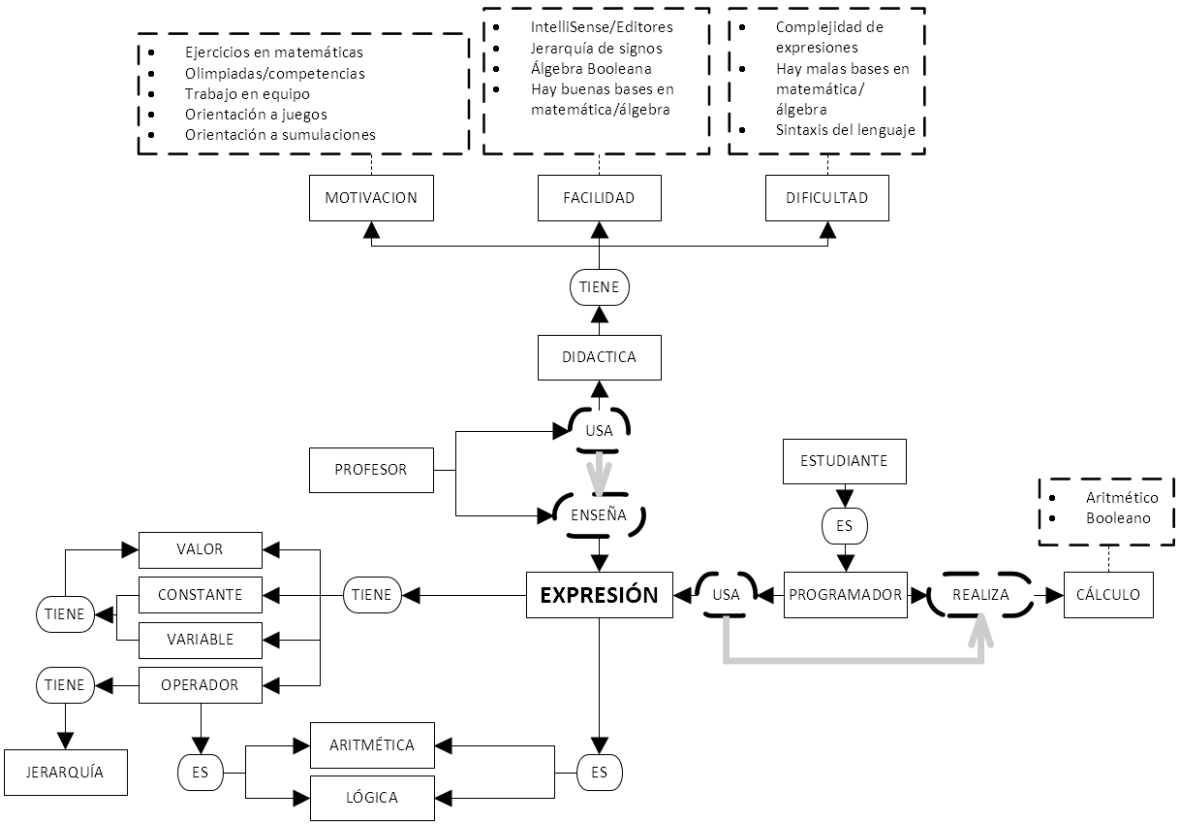


Figura 10. Ontología computacional a través del esquema preconceptual del objeto de saber: expresión aritmético-lógica (Fuente: esta investigación)

La figura 10 representa la naturaleza del objeto de saber Expresión aritmético-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

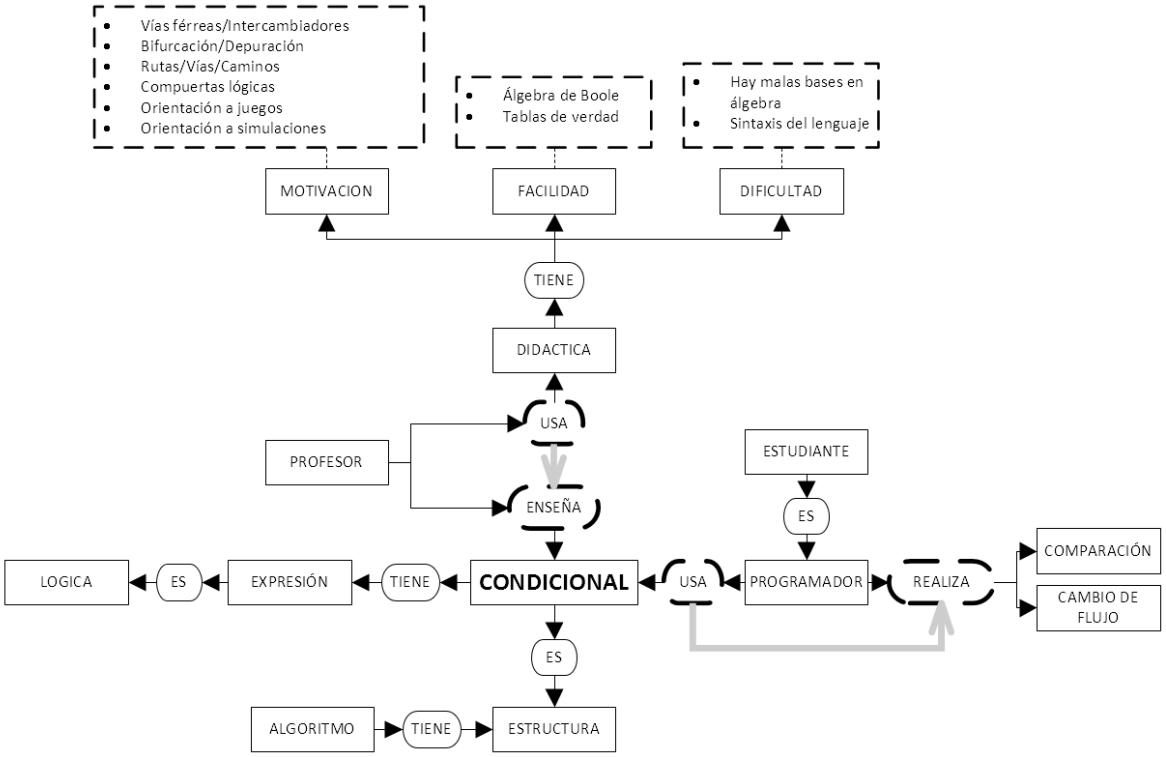


Figura 11. Ontología computacional a través del esquema preconceptual del objeto de saber: condicional (Fuente: esta investigación)

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

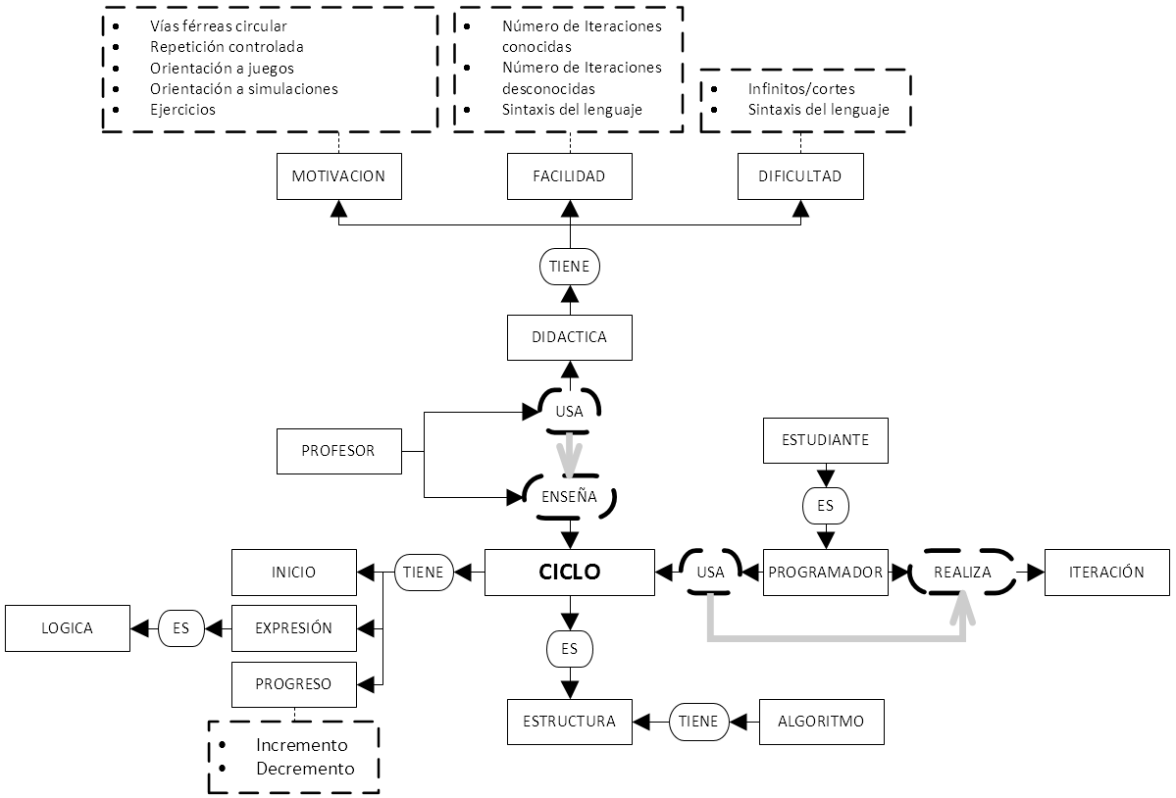


Figura 12. Ontología computacional a través del esquema preconceptual del objeto de saber: ciclo (Fuente: esta investigación)

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 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. En este punto, la abstracción toma un papel protagónico al momento de diseñar estructuras cíclicas.

Conceptualización del objeto de saber: función

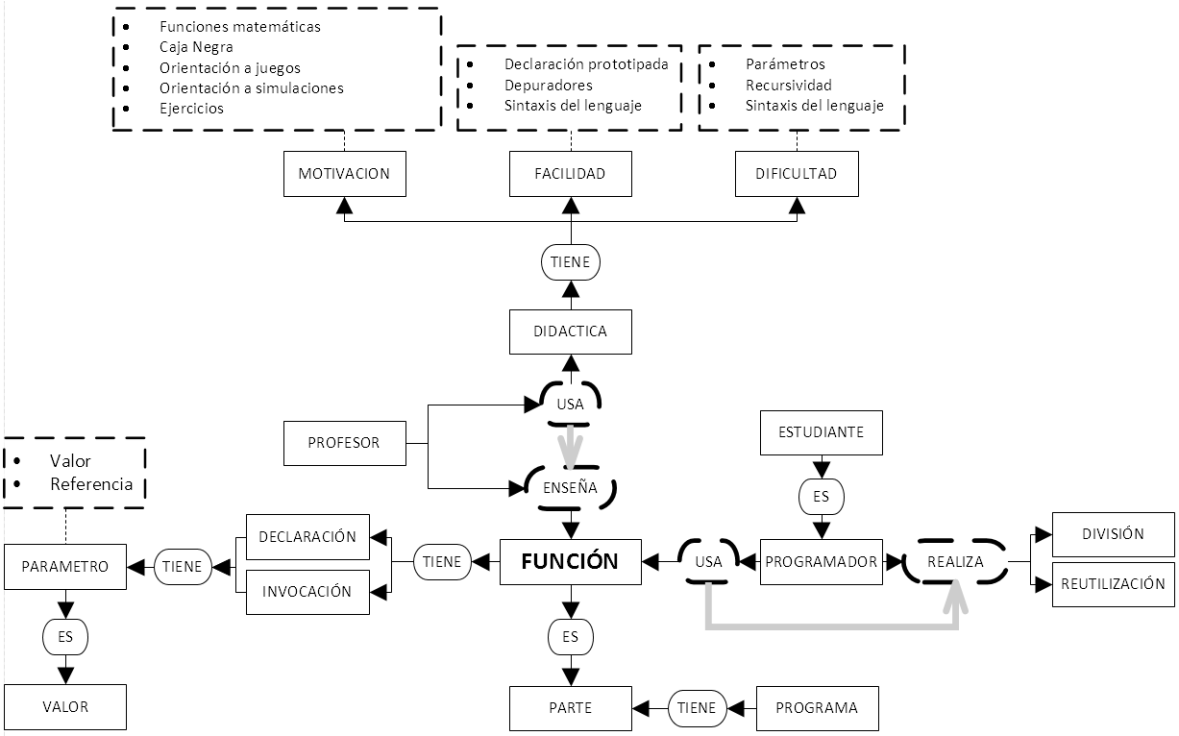


Figura 13. Ontología computacional a través del esquema preconceptual del objeto de saber: función (Fuente: esta investigación)

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 de función sea el de mayor complejidad de los objetos seleccionados en esta investigación, éste 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. Análisis de las entrevistas a 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. La síntesis de respuestas dada por ambos expertos se presenta en el Anexo E, según el instrumento de recolección de información diseñado que corresponde al Anexo B –*Semi-structured interview on teaching practices*−.

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 logra 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 ejercicios 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 aborda la indagación sobre aspectos relacionados a la vigilancia epistemológica.

5.2. Acerca de la vigilancia epistemológica

Durante el desarrollo de la presente investigación, se han encontrado 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 solo 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. Dicho instrumento puede observarse a manera de guía a través del Anexo A −*Epistemological Surveillance Guide*−. Al aplicar dicha guía de vigilancia epistemológica, se obtuvieron los resultados que han sido plasmados en el Anexo F.

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 solo fue realizada la entrada inicial del foro por parte del profesor de la Universidad de Nariño −37.I−. El autor de la investigación interpreta esta actitud cuyo origen podría deberse 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 es 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 la teoría de transposición didáctica

Esta investigación introduce 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 hace 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 de 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*– tomó parte en el 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 a 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.

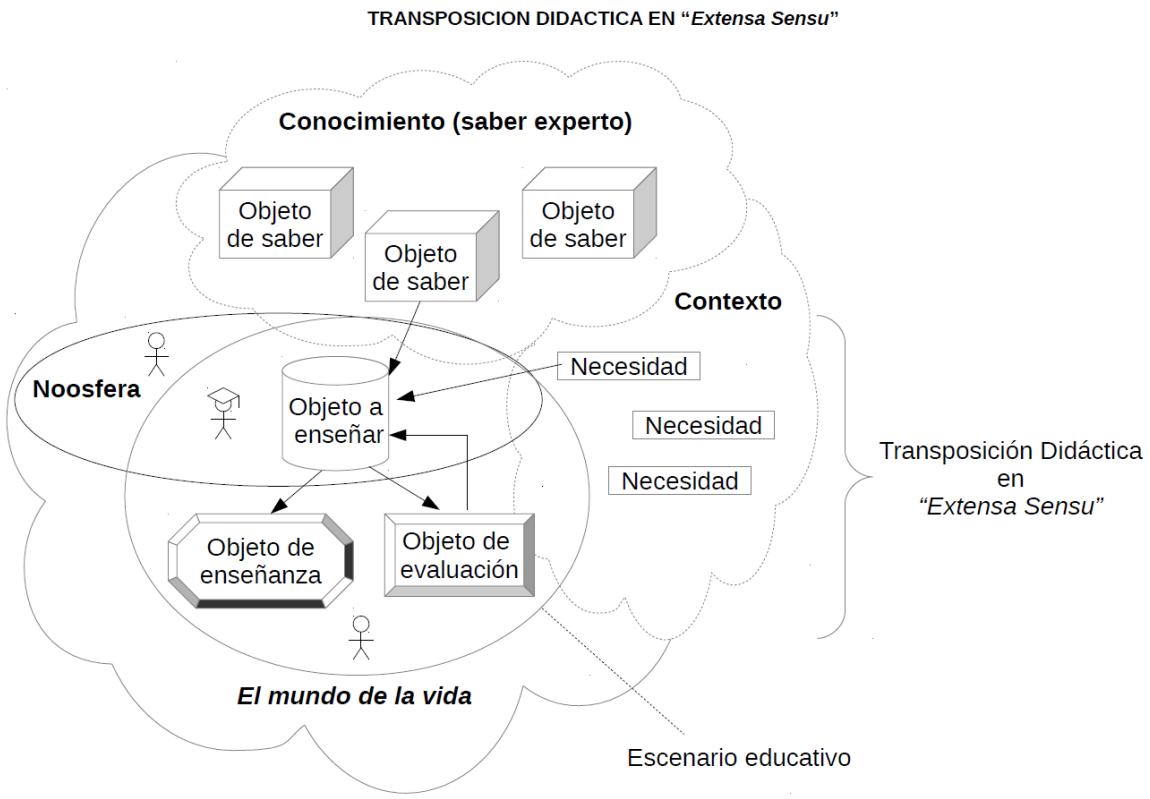


Figura 14. Trasposición Didáctica *In Extensa Sensu* (Fuente: esta investigación)

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 parecer, 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.

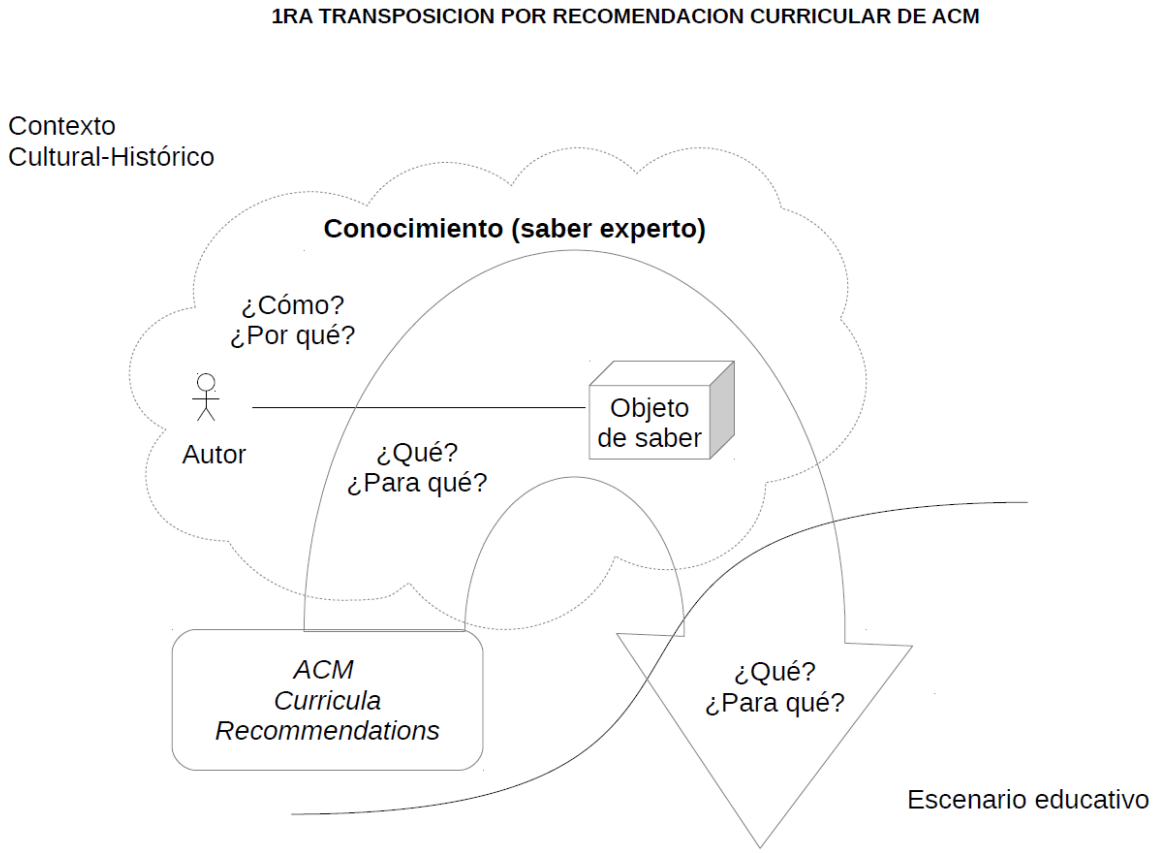


Figura 15. Primera transposición por recomendaciones curriculares de ACM (Fuente: esta investigación)

¿Para quién se desarrolla el conocimiento desde un punto de vista utilitario? ¿Está hecho solo para las clases hegemónicas, las clases sociales que tienen el control de las grandes compañías del sector productivo, o de 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 aborda 6 objetos de saber dados a partir del conocimiento experto o erudito. Para ella, es de gran interés analizar su naturaleza, sus orígenes de por qué fueron creados dichos objetos de saber. Este conocimiento es 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 de dichos objetos de saber.

El conocimiento sobre como el autor creó un objeto de saber y por qué lo hizo más allá de su utilidad instrumental si 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 en 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.

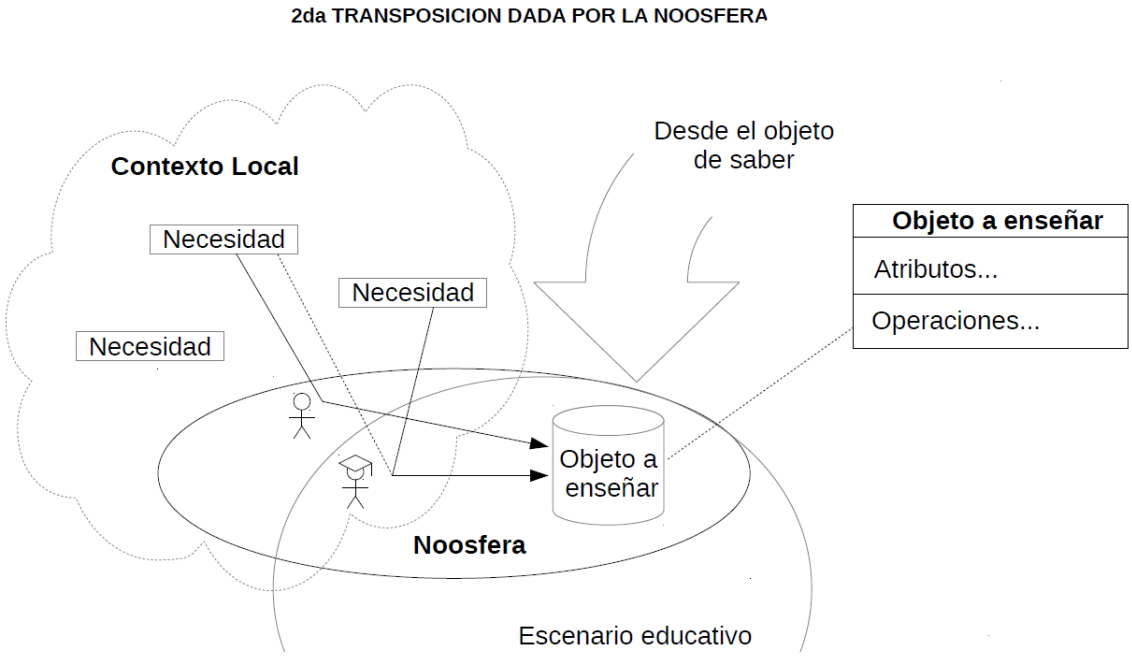


Figura 16. Segunda transposición debido a la noosfera (Fuente: esta investigación)

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.

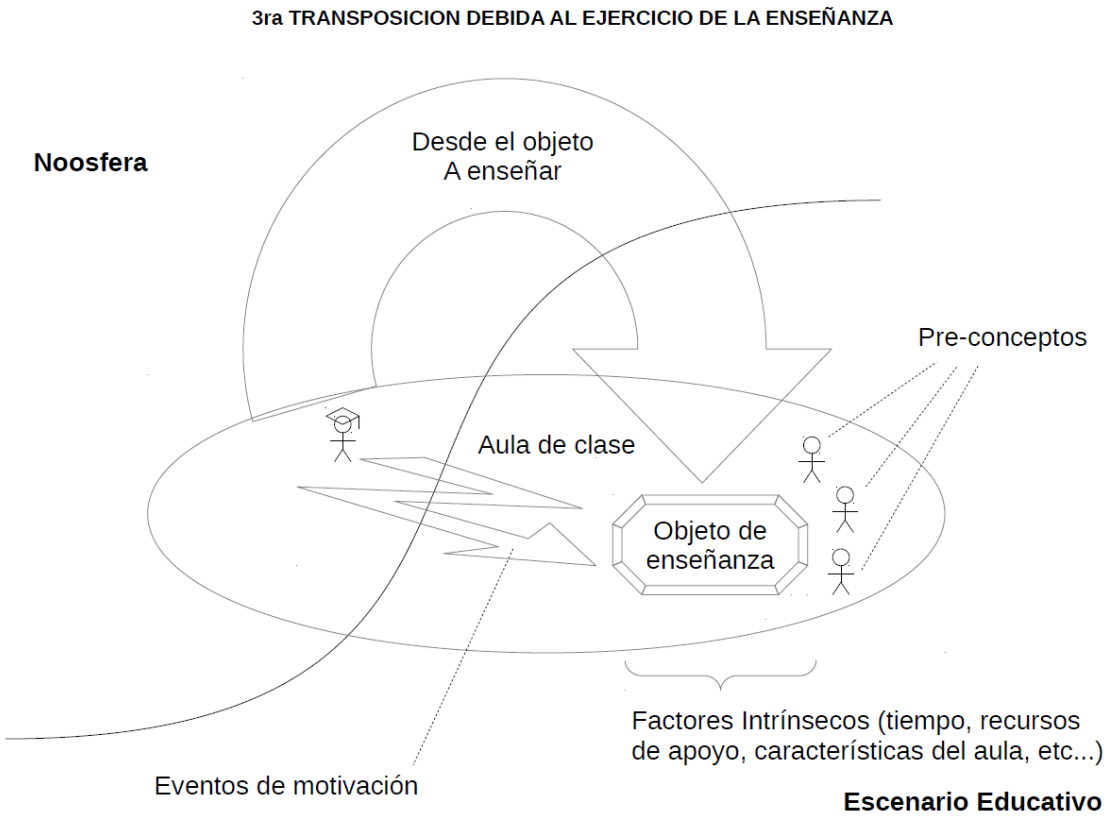


Figura 17. Tercera transposición debido a la práctica de enseñanza(Fuente: esta investigación)

Como el paso anterior, el objeto de enseñanza es un objeto diferente al objeto predecesor, están obviamente relacionados, pero son objetos 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. He aquí porque las competencias personales son factores clave a fin de determinar la habilidad de solución de problemas independientemente. 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 representa 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 porque 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; se llama “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 el cual depende de la imagen transpuesta del objeto de enseñanza: 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.

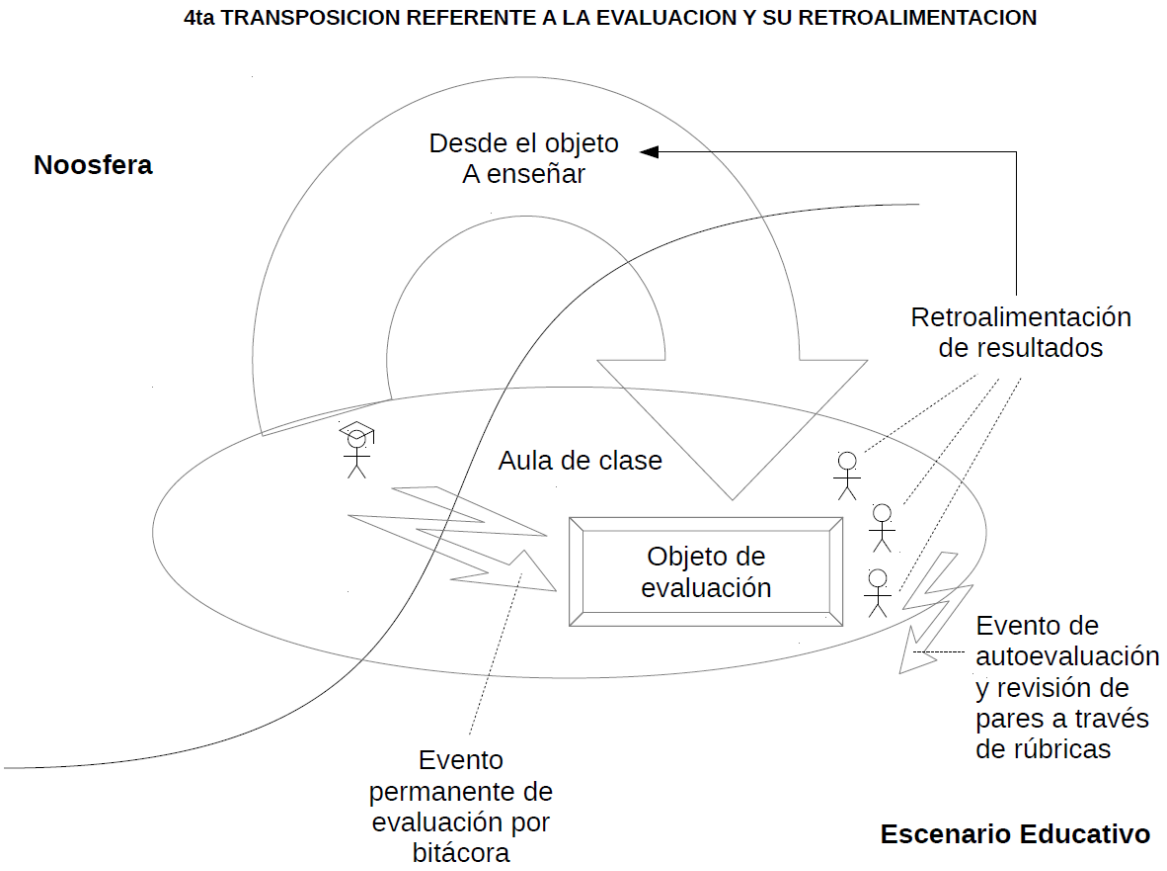


Figura 18. Cuarta transposición debido a la evaluación y la retroalimentación (Fuente: esta investigación)

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, y que puede ser articulado en sus proyectos de vida. Se está hablando de una cultura que permita a conciencia 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 revelo 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 la pregunta si 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 sobre 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 indagados 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 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 altamente 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 para cada sesión 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 y permanente.

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. Aplicación de entrevista semi-estructurada a los profesores

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 de las entrevistas a expertos; la diferencia radica en la profundidad de las respuestas, dado que los dos 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. Una síntesis de las respuestas está condensada en el Anexo G.

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, ellos 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 manifestaron 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 manifestaron los profesores.

Ante esta realidad, la pregunta que surge es: ¿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 con una revisión al final del curso sobre aquellos puntos que él consideró que no se hizo 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 tiene 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 de las prácticas en la labor docente.

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 solo podrá mantener un solo 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 enseñar dicha disciplina: 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[4]. 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 niveles de 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 logra 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.

[4] 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.

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 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 solo 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 les 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. Es muy diferente impartir un curso de fundamentos de programación hace 30 años que impartirlo ahora; no solo 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 da 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 afirman que fundamentos de programación tienen 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, éstos deben trascender más allá de una simple recompensa. Según esto, el incentivo debería orientarse hacia potenciar la automotivación en el estudiante, de tal suerte 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 alto 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.

De acuerdo con 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 de manera 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 solo los escenarios donde se encuentre acompañado por el profesor, sino que se integra a dicho esfuerzo el trabajo que el estudiante hace sin la presencia del profesor. 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 colocar 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, cuando se trate de un especio 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, en dicho documento afirma que:

> “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 con base 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 involucra 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 con base 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. Definiciones acerca del concepto 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	Facilitar el aprendizaje en los estudiantes y el complimiento de las metas establecidas serán el	Objetivos

	enseñanza	objetivo final	
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 con base en la revisión sistemática de literatura. Dicha definición se presenta a continuación.

Definición de estrategia de enseñanza

En el desarrollo de esta investigación se propone 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.

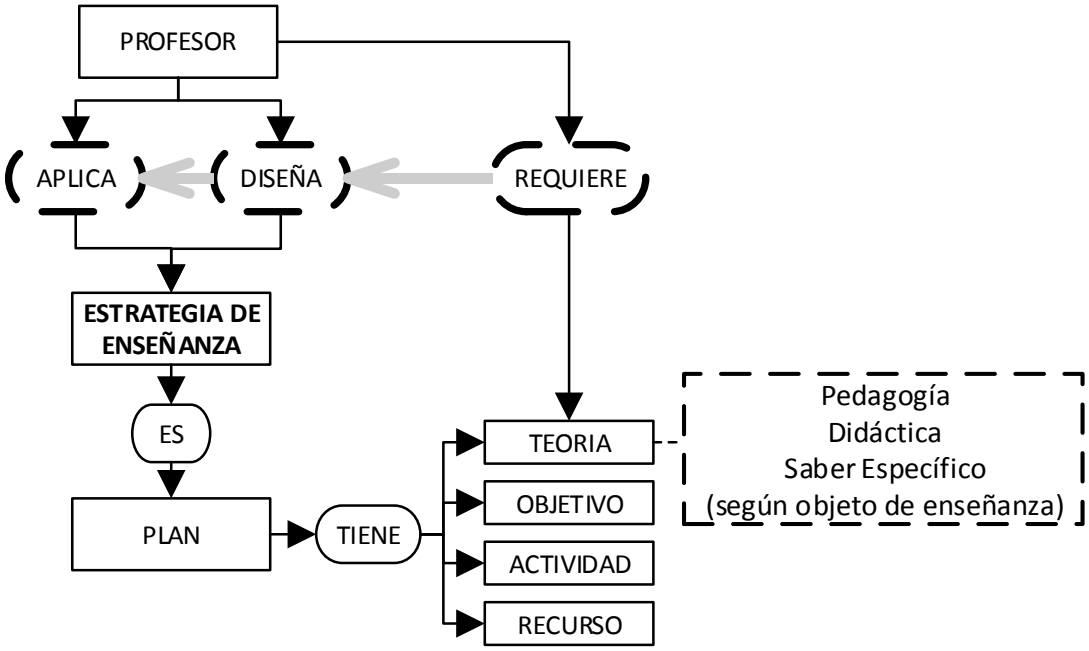


Figura 19. Esquema preconceptual para la representación ontológica del concepto de Estrategia de Enseñanza (Fuente: esta investigación)

En forma textual, una estrategia de enseñanza es un plan compuesto por una fundamentación teórica, objetivos, actividades y recursos, el cual ha sido diseñado y aplicado por el docente con base 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 y 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 menester 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 particular, se han tomado las definiciones de estrategia de enseñanza a partir de la visión de Díaz y Hernández (1999). Los posibles valores de la columna Estrategia de Enseñanza son 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 son 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 de lenguaje propio de dichas ciencias. En aras de proponer estrategias de enseñanza dentro de un estadio de la educación en ciencias computacionales, se propone 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.

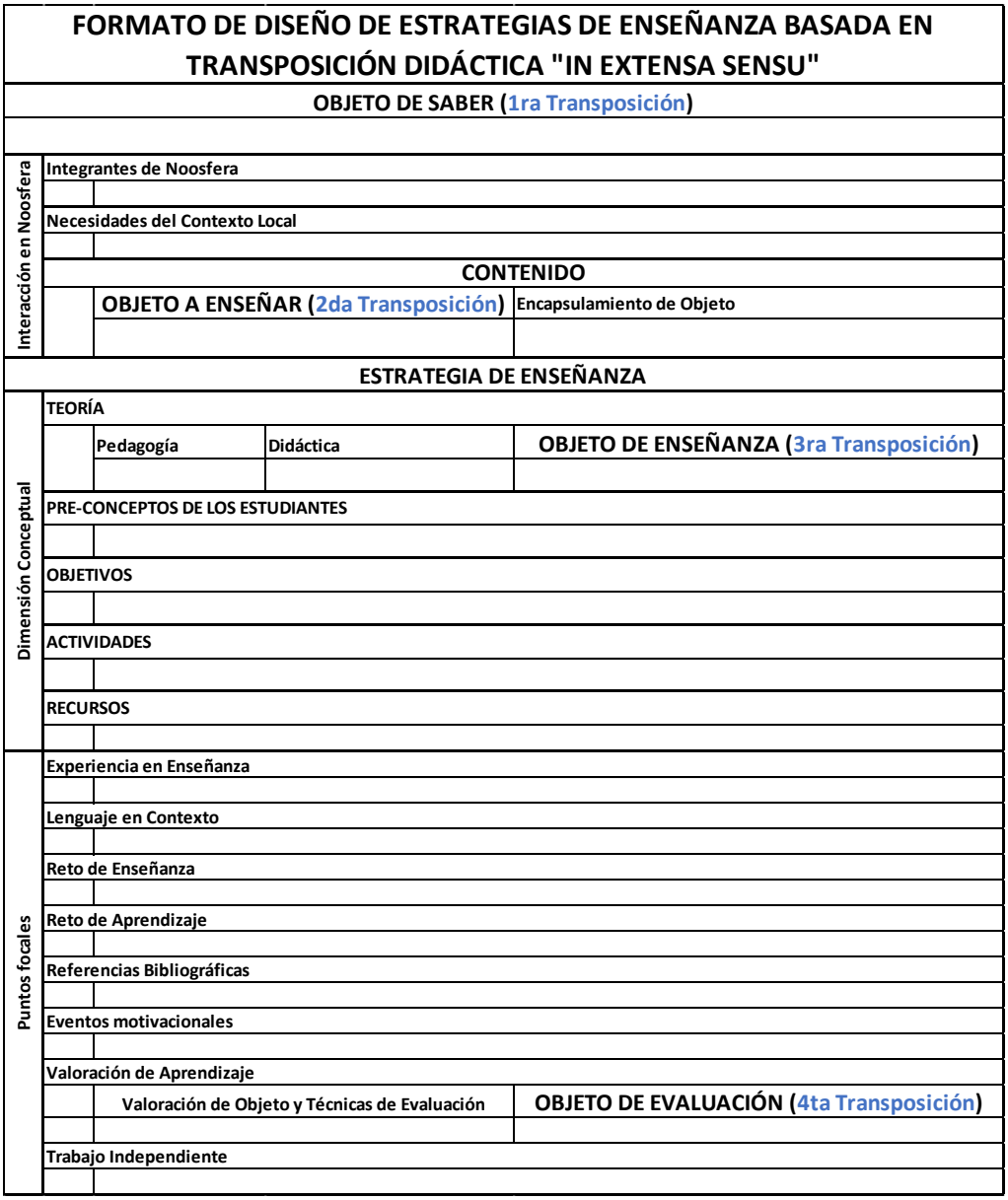


Figura 20. Formato de diseño de estrategias de enseñanza basado en Transposición Didáctica *In Extensa Sensu* (Fuente: esta investigación)

Estrategia de enseñanza propuesta para el objeto de saber: variable

Tabla 16. Estrategia de enseñanza propuesta para variable

<table>
<tr><th colspan="4">FORMATO DE DISEÑO DE ESTRATEGIAS DE ENSEÑANZA BASADA EN TRANSPOSICIÓN DIDÁCTICA IN EXTENSA SENSU</th></tr>
<tr><th colspan="4">OBJETO DE SABER: VARIABLE (1ra Transposición)</th></tr>
<tr><td colspan="4">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)</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 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)</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 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.</td><td>Atributos:
- Visibilidad
- Tipo de dato
- Valor
Operaciones:
- Declarar(…)
- AsignarValor(…)</td></tr>
<tr><td colspan="4">ESTRATEGIA DE ENSEÑANZA</td></tr>
</table>

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).

Pun

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 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 **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. Ver diseño y desarrollo didáctico en Anexo I.

Estrategia de enseñanza propuesta para el objeto de saber: constante

Tabla 17. Estrategia de enseñanza propuesta para constante

<table>
<tr><th colspan="4">FORMATO DE DISEÑO DE ESTRATEGIAS DE ENSEÑANZA BASADA EN TRANSPOSICIÓN DIDÁCTICA IN EXTENSA SENSU</th></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="6">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</td><td>Atributos:
- Visibilidad
- Tipo de dato
- Valor
Operaciones:
- Declarar(…)</td></tr>
</table>

		establecer valores inmodificables en tiempo de ejecución.	

ESTRATEGIA DE ENSEÑANZA

Dimensión conceptual

TEORÍA

	Pedagogía	Didáctica	OBJETO DE ENSEÑANZA (3ra Transposición)
	Cognitiva-Constructivista	Aprendizaje Activo-Aprendizaje Basado en Problemas	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.

PRECONCEPTOS DE LOS ESTUDIANTES

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.

OBJETIVOS

Comprender la importancia de constantes en la representación del mundo real, teniendo en cuenta que ellas pueden almacenar información del contexto que nunca cambiará.

ACTIVIDADES

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.	
	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 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. Ver diseño y desarrollo didáctico en Anexo I.

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 de 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(…)
ESTRATEGIA DE ENSEÑANZA			
Dimensión conceptual	**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. Ver diseño y desarrollo didáctico en Anexo I.

Estrategia de enseñanza propuesta para el objeto de saber: condicional

Tabla 19. Estrategia de enseñanza propuesta para condicional

<table>
<tr><th colspan="4">FORMATO DE DISEÑO DE ESTRATEGIAS DE ENSEÑANZA BASADA EN TRANSPOSICIÓN DIDÁCTICA IN EXTENSA SENSU</th></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 de 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</td><td>Atributos:
- Expresión lógica
Operaciones:
- EsVerdadera(…)
- EsFalsa(…)</td></tr>
</table>

		booleano de la expresión lógica de control	

ESTRATEGIA DE ENSEÑANZA

Dimensión conceptual

TEORÍA

	Pedagogía	Didáctica	OBJETO DE ENSEÑANZA (3ra Transposición)
	Cognitiva-Constructivista	Aprendizaje Activo-Aprendizaje Basado en Problemas	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.

PRECONCEPTOS DE LOS ESTUDIANTES

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.

OBJETIVOS

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.

ACTIVIDADES

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
	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 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	El conjunto de **condicionales** que serán utilizadas en el

		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.	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. Ver diseño y desarrollo didáctico en Anexo I.

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. Ver diseño y desarrollo didáctico en Anexo I.

Estrategia de enseñanza propuesta para el objeto de saber: función

Tabla 21. Estrategia de enseñanza propuesta para función

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. Ver diseño y desarrollo didáctico en Anexo I.

7. EPÍLOGO

Finalizada la investigación, a través de la revisión de literatura junto con el análisis de la información recolectada dan lugar a una serie de actividades que contribuyan 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 con 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 altamente 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 permitan 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 propuesta, este tipo de reto se supera 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 de los estudiantes 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 provocado buenos resultados en cuestión de 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* se constituye como el aporte principal de esta investigación, definiéndose como el constructo teórico acerca de 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 permitiendo siempre el espacio de mejoras posibles.

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 permitan organizar el conocimiento. Dentro del desarrollo de esta investigación, los esquemas preconceptuales y las técnicas de lingüística computacional han permitido culminar exitosamente los objetivos específicos planteados. De esta forma, La forma cómo fue construida la teoría de Transposición Didáctica "*In Extensa Sensu*" constituye el método para la aplicación de sí misma.

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 historia 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 Cohorte 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.

ANEXO A – Instrumentos de recolección de información

SURVEY ABOUT CHARACTERIZATION OF KNOWLEDGE OBJECTS

Dear Professor.

This survey is designed to collect valuable information about how to teach 7 topics related with the Programming Fundamentals Course. The following question is raised: **Which are the characteristics of knowledge objects of programming fundamentals in accordance with international benchmarks?** The information you provide will be used strictly for this research in anonymous way.

Please, try to complete the table below:

When you are teaching the concept of…	**Describe any motivational strategy that you use in classroom**	**Which are the EASIEST aspects of teaching such concept?**	**Which are the MOST DIFFICULT aspects of teaching such concept?**
VARIABLE			
CONSTANT			
ARITHMETIC-LOGIC EXPRESSION			
CONDITION (COMPARISION)			
LOOP			
FUNCTION (PROCEDURE)			

Thank you very much for your cooperation. ☺

Universidad de Nariño

SEMI-STRUCTURED INTERVIEW ON TEACHING PRACTICES

Dear Professor.

This interview is designed to collect valuable information about how to teach the Programming Fundamentals Concepts. The following question is raised: **Which are the teaching practices of knowledge objects related with programming fundamentals?** The information you provide will be used strictly for this research.

1. Please, describe in general the programming fundamentals course according to the syllabus.
2. How long have you taught the programming fundamentals course?
3. Do you use analogies to explain the concepts about variables, constants, expressions, conditions, loops, and functions? If so, please describe them.
4. According to your teaching experience, which are the most difficult topics to teach in the programming fundamentals course?
5. According to your teaching experience, which are the most frequent problems of students' learning?
6. Do you consider that the textbooks used by students are suitable for their learning? Why?
7. Do you use motivation strategies? If so, please describe them.
8. How do you assess the students' learning?
9. Please, describe the kind of students' work that they do out of the classroom (i.e. homework, readings, etc.).
10. Do you have any additional comment on teaching programming fundamentals?

Thank you very much for your cooperation. ☺

Universidad de Nariño

EPISTEMOLOGICAL SURVEILLANCE GUIDE

Dear Professor

This form has been designed to keep track on your interpretation of the knowledge objects of the computer programming fundamentals. Your answers will be taken anonymously, so answer with complete sincerity in accordance with your teaching experience in computer programming fundamentals.

	The textbook's definition of ... says:	**Write in your own words, the definition of...**	**Read the other concepts proposed by colleagues around the world, and write your comments**
VARIABLE			Go to Forum
CONSTANT			Go to Forum
ARITHMETIC-LOGIC EXPRESSION			Go to Forum
CONDITION (COMPARISION)			Go to Forum
LOOP			Go to Forum
FUNCTION (PROCEDURE)			Go to Forum

Thank you very much for your cooperation. ☺

ANEXO B – Validación de instrumentos

San Juan de Pasto, marzo 31 de 2016

Profesor
JESUS INSUASTI
Universidad de Nariño
Ciudad

Asunto: Validación de Instrumentos de recolección de Información

Cordial saludo.

A continuación, presentamos nuestras impresiones sobre la matriz metodológica utilizada en su investigación.

- Los objetivos específicos son claros y muestran una secuencia para realizar el análisis de los resultados. Así, la interpretación de cada uno de los instrumentos permitiría enlazar la caracterización de los objetos de saber, los niveles de transposición didáctica y las prácticas de enseñanza como un proceso coherente y conducente a conclusiones plenamente identificables.

- La elección de los objetos de saber a ser analizados representan los conceptos esenciales a ser transmitidos en el proceso de aprendizaje, de cuyo éxito depende que los estudiantes adquieran los fundamentos de programación de computadoras.

- Con respecto a la encuesta sobre la caracterización de los objetos de saber, este instrumento permite evaluar el concepto que tienen expertos internacionales sobre cada objeto. Es de destacar que la primera pregunta facilita la interpretación del encuestado sobre su visión del concepto básico de cada objeto y permitiría deducir si este aspecto es claro. Las otras dos preguntas son más directas sobre la caracterización de los objetos y el encuestado no tendría lugar a confusiones. Es así que considero que este instrumento permite alcanzar el objetivo planteado.

- La guía de vigilancia epistemológica explora las bases bibliográficas de los docentes, su interpretación de los objetos y finalmente, los contrasta con conceptos de otros expertos. Los resultados indicarían en gran medida la coherencia entre el saber personal del docente, la validez de las fuentes que utiliza y su capacidad de interpretar definiciones diversas para adaptarlas a su propio concepto. Con este instrumento se esperaría resultados muy interesantes al poder comparar

la variedad de entendimientos de un mismo objeto y poder deducir las principales falencias o fortalezas en el proceso de enseñanza de cada objeto de saber.

- La entrevista presenta un orden claro y facilita las respuestas del docente sobre sus prácticas de enseñanza. Además, el análisis posterior puede ser realizado de forma sencilla por ser preguntas bien estructuradas. En este sentido, el objetivo propuesto es claramente alcanzado.

Agradecemos su confianza y esperamos que las impresiones sean enriquecedoras a su interesante proceso de investigación.

Cordialmente,

Javier Revelo, PhD
Director
Departamento de Electrónica
Universidad de Nariño

Andrés Pantoja, PhD
Docente Tiempo Completo
Departamento de Electrónica
Universidad de Nariño

ANEXO C – Informe de pasantía internacional

Cádiz, 9 de diciembre de 2013

Doctora Gabriela Hernández
Coordinadora del Programa de Doctorado en Ciencias de la Educación
Universidad de Nariño
Pasto, Colombia

En calidad de tutor internacional de la pasantía de investigación del estudiante Jesús Insuasti Portilla, identificado con pasaporte Colombiano AJ237522, me permito rendir el informe de actividades realizadas en su estancia en la Universidad de Cádiz, la cual inició el 15 de octubre y finalizó el 15 de noviembre del presente año.

Es importante resaltar que dicha estancia de investigación fue favorecida por una beca de la segunda convocatoria del Programa de Movilidad Académica de la AUIP entre universidades Andaluzas y Latinoamericanas para el año 2013 (se adjunta la notificación de dicha beca).

Fueron establecidos 7 elementos según el plan de actividades previamente acordado, a saber:

Actividad	Descripción
1	Documentación de Experiencias en materia de construcción de contenidos digitales educativos
2	Observación de pruebas piloto
3	Entrevistas, temática: "La Educación en las Ciencias de la Computación"
4	Estudio de caso: Un análisis de transposición didáctica sobre un objeto de estudio específico.
5	Preparación de artículo de avance de investigación donde se incluya las actividades 1, 2, 3, y 4.
6	Visualización de posibles convenios o desarrollo de proyectos en conjunto.
7	Conclusiones

A continuación se describe en detalle el desarrollo de cada actividad.

Actividad 1:

La oportunidad de interactuar con el proyecto parcialmente financiado por CIP (*Competitiveness and Innovation Framework Programme*) de la Unión Europea denominado *Open Discovery Space (ODS),* ha permitido entablar lazos de cooperación para el desarrollo de actividades orientadas al uso de tecnologías computacionales aplicadas a ambientes educativos. En esta materia, la estancia de investigación permitió la vinculación de Jesús Insuasti dentro de los procesos de revisión del material educativo en construcción por parte de la Universidad de Cádiz para la comunidad "Colegios de España". En este sentido, Jesús se fundamentó en la metodología de trabajo de los proyectos a desarrollarse en el marco de ODS, con el fin de habilitar su participación en comunidades digitales similares.

Actividad 2:

Si bien es cierto que la Universidad de Cádiz ha iniciado su participación en *ODS*, en estos momentos el equipo de trabajo ha podido iniciar con la gestión de talleres "*workshops*" con una muestra del profesorado de los colegios de enseñanza primaria y secundaria de la ciudad. Sin embargo, es importante anotar que la implementación del material educativo ha sido re-programada para el próximo año. Por tal razón, la segunda actividad que hace referencia a la observación de pruebas piloto fue limitada a la exploración de la construcción de los materiales educativos y no a su aplicación en escenarios reales.

Actividad 3:

Con el fin de reforzar la propuesta de tesis doctoral en el campo de la didáctica en los cursos de fundamentos de programación, Jesús realizó una serie de entrevistas a los profesores expertos que imparten dicha asignatura. Para tal efecto, se realizó una consulta con el director de carrera de Ingeniería Informática de la Universidad de Cádiz para identificar los profesores que imparten la asignatura de introducción a la programación en Ingeniería Informática a nivel de grado, teniendo en cuenta su experiencia en la materia.

Un profesor fue entrevistado, cuya información suministrada constituye en un insumo fundamental para el interés investigativo de la tesis doctoral de Jesús. El fenómeno de transposición didáctica fue evidenciado en las respuestas de la entrevista con grandes posibilidades de potenciar las prácticas de enseñanza de los fundamentos de programación en Ingeniería Informática y todas las carreras cuya disciplina sea a fin a las ciencias computacionales.

Desde el punto de vista metodológico, Jesús realizó entrevistas semiestructuradas, con toma de notas y sin registro de grabación digital. Jesús manifestó que éste diálogo le ha permitido formular el eje teórico central de su tesis doctoral en curso que se titula: "Transposición didáctica en los fundamentos de programación de computadoras: Una propuesta de enseñanza para las disciplinas asociadas a la computación".

Actividad 4:

El estudio concreto del fenómeno de transposición didáctica fue realizado a través de una metodología "experimental" de indagación en un caso específico. Para ello, Jesús me entrevistó con el fin de hacer un seguimiento a objetos de enseñanza específicos. A través de esta experiencia, fue posible determinar las pautas que se deben tener en cuenta al momento de caracterizar las fases de transformación del objeto de saber, el objeto a enseñar y el objeto enseñado.

Este diálogo sobre el seguimiento de un objeto en particular ha permitido fundamentar nuevos aspectos a la teoría original de la transposición didáctica de Yves Chevallard, lo cual se constituye en interesantes aportes en el campo específico de la enseñanza de los fundamentos de programación en el marco de las ciencias computacionales. Con estos insumos, Jesús presenta un concepto que denomina "Transposición Didáctica en *Extensa Sensu*", el cual se explica a través del artículo de avance de investigación que fue construido para dar cumplimiento a la siguiente actividad.

Actividad 5:

Con el trabajo teórico realizado en la estancia de investigación, Jesús ha manifestado que ha podido trazar el eje fundamental de su tesis doctoral, la cual responde a una versión extendida de la teoría original del profesor Yves Chevallard. De esta forma, fue conformada una visión muy propia para la enseñanza de las ciencias computacionales a la cual se le dio el nombre de Transposición Didáctica en "Extensa Sensu".

Con esto, hemos elaborado un artículo de avance de investigación de su tesis doctoral junto con los hallazgos de los trabajos realizados en la Universidad de Cádiz a través de algunas de mis investigaciones. Este producto de la estancia de investigación se pretende publicar en una revista de difusión mundial llamada *ACM Inroads* sobre los avances en el campo de educación de las ciencias computacionales. Actualmente se ha iniciado los trámites para su publicación y su contenido se adjunta al presente informe. En síntesis, el artículo presenta la visión extendida del fenómeno de transposición didáctica en la enseñanza de los fundamentos de programación y describe una serie de experiencias de los proyectos realizados en temas afines.

Cabe resaltar que, en acuerdo con los términos de referencia contractual de la beca que le ha sido otorgada a Jesús, toda publicación que se genere como producto de la estancia de investigación hará en forma explícita el reconocimiento a la AUIP, anotando que la estancia de investigación recibió el apoyo financiero por dicha entidad.

Actividad 6:

En cuanto al estudio de viabilidad de posibles convenios de cooperación entre la universidad Andaluza (en este caso la Universidad de Cádiz) y la universidad Latinoamericana (en este caso la Universidad de Nariño – Colombia), por fortuna está vigente un convenio marco de cooperación entre dichas instituciones de educación superior; dicho convenio marco fue firmado por los rectores de las universidades en el 2012. Sin embargo, no hay hasta la fecha un convenio específico que permita la interacción directa de las dos instituciones.

Dada esta situación, Jesús logró entablar diálogos con los decanos, tanto de la escuela Superior de Ingeniería como de la Facultad de Ciencias de la Educación de la Universidad de Cádiz, a fin de mirar estrategias que permitan posibles actividades de cooperación académica. Jesús ha informado que los decanos entrevistados manifiestan su disponibilidad de estudiar las propuestas escritas por parte de la Universidad de Nariño.

Actividad 7:

Finalmente, Jesús manifiesta haber logrado avances significativos en la elaboración teórica de su tesis doctoral en curso. Además, Jesús adquirió la membresía para trabajar en conjunto dentro de una nueva comunidad de experiencias de enseñanza de las ciencias computacionales denominada *Ibero-American Computing* disponible en http://portal.opendiscoveryspace.eu/beta/members/174170.

Adicionalmente, Jesús ha logrado ingresar a la comunidad de investigación *SIGCSE Chapter* Spain –*Special Interets Group in Computer Science Education*–. Esta vinculación fue realizada

con el fin de participar en proyectos de investigación en materia de educación en ciencias computacionales, para ser trabajados tanto en España como en Colombia.

Teniendo en cuenta el desarrollo de las actividades anteriormente descritas, doy por cumplida la pasantía internacional de Jesús Insuasti Portilla, como requisito de su formación doctoral.

Agradezco la atención prestada.

Atentamente,

Juan Manuel Dodero, PhD
Profesor Titular
Departamento de Ingeniería Informática
Universidad de Cádiz
C/ Chile 1, 11002 Cádiz
Tel +34 956 015 784 Fax +34 956 015 139
juanma.dodero@uca.es

ANEXO D – Publicación internacional de artículo según SCOPUS©

Scopus

Documents

Insuasti, J., Beardo, J.M.D.

About didactic transposition: Teaching programming fundamentals at different levels of the school system

(2015) *Proceedings - 2015 International Conference on Learning and Teaching in Computing and Engineering, LaTiCE 2015*, art. no. 7126238, pp. 91-94.

DOI: 10.1109/LaTiCE.2015.53

[a] Systems Department, University of Nariño, Pasto, Colombia
[b] Department of Computer Science, University of Cádiz, Cádiz, Spain

Abstract
This paper introduces a new approach for the analysis of didactic transposition phenomenon in Computer Science education, and its foundations for the construction of teaching objects. Thus, two main scenarios were explored: teaching experiences on programming fundamentals at undergraduate level, and the construction and deployment of teaching objects which have been framed within the European project called Open Discovery Space for K-12 education, by using computing means. The description of experiences in both scenarios is documented according to the following schema: A theoretical background, the findings, and the statement of new proposals to improve teaching practices and the criteria for constructing teaching objects as conclusions of this paper. © 2015 IEEE.

Sponsors:
Publisher: Institute of Electrical and Electronics Engineers Inc.

Conference name: 2015 3rd International Conference on Learning and Teaching in Computing and Engineering, LaTiCE 2015
Conference date: 9 April 2015 through 12 April 2015
Conference code: 112965

ISBN: 9781479999675
Language of Original Document: English
Abbreviated Source Title: Proc. - Int. Conf. Learn. Teach. Comput. Eng., LaTiCE
Document Type: Conference Paper
Source: Scopus

ANEXO E – Síntesis de respuestas de la encuesta a profesores

Respuestas sobre el concepto de Variable

RESPUESTAS SOBRE CONCEPTO DE VARIABLE			
Profesor	**Estrategia Motivacional**	**Facilidad de Enseñar**	**Dificultad de Enseñar**
10.H	*I usually use the metaphor of mathematics. Especially the representation of first grade functions. For example: in saying that a straight line whose function in the Cartesian plane is described as y = 2x, when graphing and using tabulation, students recognize the use of the concept of mathematical variable.*	*I think it's very easy to teach the types of data that variables can store.*	*Perhaps, when thoroughly explaining aspects related to memory addressing, a level of difficulty could be presented.*
13.V	*In the classroom, I usually represent the memory of a computer as a large mailbox. Each box represents a variable, which has the possibility of being filled by some mail (some value)*	*I think the classification of data types associated with the concept of variable is easy to explain with clear examples.*	*Honestly, I do not find difficult aspects to explain for this concept.*
24.F	*I think that my teaching strategies are something traditional. I explain the concept of variable with boxes represented on a board, a value is located in each box.*	*In general, it is easy to teach this concept.*	*Nothing*
25.L	*Honestly, my teaching practice has not had in-depth pedagogical reflection. I don't know about motivational strategies that promote better learning in students. According to this, I consider that my teaching practice is very traditional.*	*I believe the understanding of the variable concept depends on previous education in mathematics, and especially in algebra. With a good foundation in such education, I think it is very easy to teach such concept.*	*On some occasions, students often get lost when making conversions of numeric and alphanumeric data types. Casting usually generates difficulties in the students; so, the teaching of conversion of data types requires special attention.*
27.G	*To explain the concept of variable I usually work with several small cardboard boxes. The action of storing or recording values in the variable is represented as placing paper numbers inside those boxes. It is easy to explain how a variable can change its stored content as many times as necessary.*	*I find teaching the concept of variable pretty easy.*	*I don't see what thing associated with the variables can be difficult to teach.*
28.X	*It's difficult to say. Honestly, I don't think I have motivational activities to explain this concept.*	*All the programming languages that inherit the ECMA syntax are easy to teach. In this case, the declaration of identifiers for variables*	*Explaining the reasons why there are rules in C++ for the nomenclature of identifiers usually has a bit of difficulty.*

		becomes easy to teach when the standard is known previously for students.	
36.H	*According to my experience, students like games. So, I usually use the development of games as motivation. The variables I use refer to solving problems in the world of computer games.*	*I consider that everything related to the teaching of variables is easy if it is given an adequate use from motivation.*	*Nothing*
40.H	*I believe the best motivation for students is the fact that their programs are useful. In that sense, I approach the teaching of the variable concept with things that are of interest to students. In my case, I use simulations of physical phenomena.*	*The use of the workspace area for declaration of variables and constants is very easy to use. In this sense, the teaching of the variable concept isn't difficult.*	*None*
47.M	*During my experience in teaching, I consider that I have created my own way of teaching in the lectures. I have no previous education in pedagogy; however, my students agree that such lectures are pleasant.*	*I have been teaching for more than 10 years, teaching this concept is very easy for me*	*No problem at all*
7.H	Al no tener sólidas bases en pedagogía, la explicación que suelo hacer sobre variables es a través de varios ejemplos, junto con su representación en la memoria. En este punto introductorio, no toco temáticas de direccionamiento de memoria	Considero que es fácil enseñar el concepto de variable en su totalidad.	La verdad, no veo dificultad alguna al momento de enseñar este concepto.
32.M	Suelo representar la memoria de un computador como un enorme casillero. Cada casilla representa un espacio de memoria el cual puede ser ocupado por un valor con un solo tipo de dato vigente. Esta analogía hace que mis estudiantes comprendan mejor el concepto de variable en la memoria	La regla sobre nomenclatura de identificadores es muy clara en el lenguaje. De esta manera, considero que enseñar el concepto de variable es fácil a partir de las reglas de definición de los identificadores.	Al abordar justamente los temas relacionados con direccionamiento de memoria, se suscitan situaciones que complican a los estudiantes. Las representaciones hexadecimales de las direcciones de memoria suelen confundir a algunos estudiantes.
37.I	Considero que utilizo un método de enseñanza tradicional. Si veo muy importante el desarrollo de ejercicios tanto en clase como tareas que se les deja a los estudiantes. Para el caso de variables, creo que lo que me ha	En los primeros semestres, si los estudiantes vienen con buenas bases matemáticas del bachillerato, enseñar el concepto de variable	No encuentro elementos que sean difíciles de explicar

	servido es comparar el concepto con el que maneja la matemática.	en fundamentos de programación es fácil.	
39.D	En clase, destino un área del tablero para ubicar los identificadores de las variables y las constantes a manera de bitácora, donde sus valores (exceptuando el de las constantes) va variando según el paso de instrucción del programa.	La sintaxis misma del lenguaje permite tener claridad sobre la declaración de identificadores con sus respectivos tipos de datos. En síntesis, es fácil enseñar el concepto de variable en Java.	Tengo la impresión que los estudiantes tienen cierta dificultad cuando se trata de manipular conversión de tipos de datos entre variables. He tratado de explicar los conceptos en forma clara, sin embargo, siempre suele haber esa dificultad en el estudiante.

Fuente: esta investigación

Respuestas sobre el concepto de Constante

RESPUESTAS SOBRE CONCEPTO DE CONSTANTE			
Profesor	**Estrategia Motivacional**	**Facilidad de Enseñar**	**Dificultad de Enseñar**
10.H	*Like the previous situation, I use mathematics and especially physics to explain the concept of constant. I speak of the number PI, the constant of acceleration of gravity in the earth, among others.*	*I think it's very easy to teach the types of data that constants can store.*	*Like variables, constants require memory addressing. Concept that could be a bit difficult to digest for new students*
13.V	*To explain the concept of constant, I remind students of the concept of mathematical constant, such as PI, E, etc.*	*Pretty easy to explain it!*	*Nothing.*
24.F	*Starting from the teaching of the variable concept, I teach in the same way emphasizing that a constant cannot change its value.*	*In general, it is easy to teach this concept.*	*Nothing*
25.L	*Traditional lecture. The same explanation for teaching the concept of variable.*	*I don't find how to teach the concept of constant can generate some difficulty. It is easy for me to teach it.*	*How curious it is when some students don't understand the nature of a constant. Many have told me that they don't see the need to create constants, it would be enough to only create variables that won't change their value.*
27.G	*Taking into account the previous exercise, to teach the concept of constant I use a special cardboard box that can only be opened, and its contents filled only once. Such box has a special seal that prevents opening it at any time*	*Everything in constants is easy to teach.*	*None*

	easily. With this simile, I usually explain the concept of constant.		
28.X	*None*	*I do not find anything that harms the teaching of this concept.*	*None*
36.H	*I propose the creation of constants such as maximum number of players, number of lives, etc. They are common concepts in the world of computer games.*	*Everything about constants is easy to teach.*	*Nothing*
40.H	*In class I propose the programming of small simulations. Thus, the motivation arises from the definition of each simulation. The constants can be represented as maximum number of tries of the experiment, gravity constants, etc.*	*See the previous answer.*	*None*
47.M	*I make a textual explanation with examples in the lectures.*	*Pretty easy!*	*None, I think it's pretty easy!*
7.H	Suelo basarme en la definición de constante en física. Pongo ejemplos tales como: la constante de gravitación universal, el número de Avogadro, la constante de Euler, etc.	Enseñar constantes es fácil.	No los hay.
32.M	En términos simples, en clase se trabajan las constantes con la misma nomenclatura y metodología de definición de variables, haciendo énfasis en que el valor de una constante se define al inicio y nunca su valor podrá ser cambiado.	Enseñar constantes es supremamente fácil.	No observo aspectos que puedan ser difíciles de explicar en cuanto a constantes se refiere.
37.I	Pasa lo mismo que en el concepto anterior. Yo creo que la forma mejor de explicar las constantes es tomando ejemplos del mundo matemático.	Como lo mencioné anteriormente, las buenas bases matemáticas hacen que el concepto de constante sea fácil de explicar.	Tampoco encuentro elementos que sean difíciles de explicar
39.D	La forma se explicó en el anterior concepto.	Simplemente se hace referencia al uso de palabras reservadas para hacer que un identificador sea inmodificable su valor contenido. Considero que esto es fácil de enseñar.	No veo aspectos difíciles de enseñar en materia de constantes en Java a pesar de que el concepto de constante sea representado por las palabras reservadas "*static final*"

Fuente: esta investigación

Respuestas sobre el concepto de Expresión Aritmético-Lógica

RESPUESTAS SOBRE CONCEPTO DE EXPRESIÓN ARITMÉTICO-LÓGICA			
Profesor	**Estrategia Motivacional**	**Facilidad de Enseñar**	**Dificultad de Enseñar**
10.H	*Again, I resort to the development of mathematical exercises such as solving small systems of linear equations to clear the values of an unknown quantity. In this sense, at least arithmetic expressions can be manipulated. For logical expressions, I simply use the syllogisms.*	*In general, students who face programming fundamentals come with a previous education in mathematics, the use of the analogy facilitates the students' learning on my opinion.*	*When using complex expressions (several operators, variables, values, and parentheses), students usually show difficulties.*
13.V	*When these topics are discussed in class, I usually do practical workshops for solving arithmetic expressions and logical expressions. In groups of up to 3 students, they solve a list of expressions, then they exchange them between each group to review successes and failures.*	*The more exercises students perform on analyzing arithmetic expressions and logical expressions, the more they will learn. In this sense, it is easy to teach these concepts when there are many exercises.*	*Perhaps students face naming problems due to unbalanced parentheses in long expressions.*
24.F	*I use writing exercises and solving arithmetic expressions and logical expressions. To do this, I usually introduce the concept of sign hierarchy and Boolean algebra.*	*I don't have any problem by teaching the concept*	*Perhaps, the solution of arithmetic and logical expressions of great complexity produces some difficulty in the students. But I don't consider that it is something that harms the teaching process.*
25.L	*Traditional lecture. The same explanation for teaching the concept of variable.*	*Pretty easy. I just recall some mathematical foundations in order to build arithmetic and logical expressions.*	*If you allow me to comment, the previous mathematical education that students have must be solid. I believe that a large part of the learning difficulties faced by students is due to their poor formation in mathematical bases. On the other hand, I don't consider that there are hard aspects of teaching in this regard.*

27.G	*The motivational agent that I use to teach these concepts is through the development of a Math Olympiad. Through groups, students must solve a series of arithmetical and logical expressions in limited time. At the end, I obtain the results that represent bonuses for their final grade.*	*It's easy to teach as long as there is a good foundation in Math.*	*Nothing*
28.X	*I grant some bonuses to those students who solve the largest number of proposed expressions in the shortest time.*	*Teaching the handling of arithmetic expressions and logical expressions becomes easy when there is a good foundation in Math.*	*When working with other numerical systems than decimal, for example octal, binary, or hexadecimal, students usually have some difficulties in solving arithmetic expressions. However, this situation doesn't constitute a weakness of teaching according to my opinion.*
36.H	*Many computer games require special simulations for physical effects (gravity, trajectory calculation, collisions, etc.) It is right here where the use of strong arithmetic expressions and logical expressions is required.*	*When working with complex expressions in arithmetic and Boolean notation, a great help is the integrated development environment. With its characteristic called IntelliSense, it automatically determines if there is an unbalance of parentheses, or if the expressions are syntactically badly assembled. In this sense, the teaching of syntax is simplified.*	*Nothing.*
40.H	*In the case of simulation of physical phenomena, the use of physical functions is required. In this way, arithmetic expressions and logical expressions are taught within an orientation towards simulations based on physics.*	*In the particular scenario of using MATLAB as a programming language, the use of arithmetic expressions and logical expressions is widely supported by the environment. Taking advantage of the potential of the MATLAB environment,*	*In essence, MATLAB is a matrix calculator. If students do not have previous training in the use of this type of calculator, there may be difficulties in learning arithmetic and logical expressions.*

		teaching in this area becomes very simple.	
47.M	*lots of exercises*	*Pretty easy!*	*Not at all*
7.H	Realizamos con los estudiantes múltiples ejercicios con el fin de despejar una incógnita, o saber qué valor tiene una variable a través de expresiones aritmético-lógicas complejas. Al parecer, ese ambiente de competitividad en el aula de clase es motivante para los estudiantes.	Para mis estudiantes, es muy comprensible la jerarquía de signos (operadores aritméticos y operadores lógicos) así como la jerarquía de paréntesis. Esta parte considero que es fácil de enseñar dada la formación previa con que vienen los estudiantes.	He notado dentro de mis clases que al crecer en complejidad las expresiones aritmético-lógicas, los estudiantes suelen encontrar dificultades al momento de manejarlas. Yo veo que se trata de un asunto de mejorar la atención por parte de los estudiantes al momento de trabajar con expresiones muy complejas.
32.M	Expongo problemas de origen matemático para que los estudiantes puedan interpretarlos a través de expresiones aritmético-lógicas. Por su resolución en clase, otorgo puntos para las evaluaciones parciales.	Debo reconocer que, gracias a las opciones de corrección sintáctica de los entornos integrados de desarrollo, la construcción de expresiones aritmético-lógicas con una buena estructura sintáctica es fácil de enseñar.	La verdad enseñar la sintaxis de las expresiones aritmético-lógicas es relativamente fácil. Sin embargo, he notado que mis estudiantes tienen problemas con la simbología de los operadores y de las contracciones o "formas cortas" de representar incrementos, decrementos e incluso condicionales.
37.I	Para enseñar expresiones aritmético-lógicas suelo realizar muchos ejercicios en clase, al mismo tiempo que permito que los estudiantes resuelvan ejercicios en clase con el fin de otorgar décimas de un examen parcial o final. He notado que un estímulo para los estudiantes consiste en tener la posibilidad de asegurar buenas calificaciones, y el desarrollo de ejercicios en clase es un espacio para ello.	En esta parte, yo creo que los lenguajes de programación de hoy en día ayudan mucho, en especial los entornos integrados de desarrollo. Ellos ayudan a corregir errores de escritura y brinda sugerencias para ello.	Al momento en que empiezo a trabajar con fórmulas matemáticas complejas, he notado que los estudiantes manifiestan alguna dificultad.

39.D	En clase armo equipos de trabajo de hasta tres estudiantes. Cada equipo debe resolver un conjunto de ejercicios sobre expresiones aritmético-lógicas.	Muchos ejercicios se requieren para afianzar el aprendizaje de la buena escritura de expresiones aritmético-lógicas. Yo veo que ejercitar a los estudiantes en estos conceptos hace fácil la enseñanza de los mismos.	He notado que la combinación de funciones inmersas en expresiones aritmético-lógicas genera dificultad en los estudiantes. Por otra parte, si existen conversión de tipos de datos inmersas en las mismas expresiones, los estudiantes suelen presentar mayor dificultad.

Fuente: esta investigación

Respuestas sobre el concepto de Condicional.

RESPUESTAS SOBRE CONCEPTO DE CONDICIONAL			
Profesor	**Estrategia Motivacional**	**Facilidad de Enseñar**	**Dificultad de Enseñar**
10.H	*At the moment of approaching the conditionals, I make the analogy of representing a conditional as railway switches. in a certain position, the train can take a road, but by altering the position of the switch, the train will travel in another direction.*	*The handling of two possible options that are present in a simple conditional is easy to explain. There are only two possible paths: if the condition is true or if the condition is false.*	*In general, I could say that the domain of Boolean algebra as well as the management of arithmetic operators' priorities is required. If the students do not have good bases on these abstract concepts, learning is a bit difficult.*
13.V	*I do not consider that I have a motivational strategy to explain this concept.*	*Everything related to conditional, is easy to explain.*	*Nothing.*
24.F	*The concept of bifurcation is easily explained. It simply emphasizes the existence of two paths, on the one hand students can divert the execution of the code in a program, if the condition to be evaluated is true. The other path is taken if the same condition is false.*	*It isn't difficult to explain the concept.*	*Nothing*
25.L	*Traditional lecture. The same explanation for teaching the concept of variable.*	*I notice that the teaching of conditionals is facilitated when there are good bases about Boolean algebra. In this vein, teaching conditional is very simple.*	*None*

27.G	*I have always explained the Boolean algebra through the use of logic gates. In this way, I teach the concept of conditional which is supported first in the foundation of logic gates. First off, I teach the Boolean algebra using logic gates.*	*It's easy to teach as long as there is a good foundation in Boolean algebra.*	*When Boolean expressions grow, there is often complexity for students to understand them.*
28.X	*I usually incorporate elements of algorithmic complexity. In this sense, I usually reward those students whose algorithms perform better. I measure nested comparisons through cyclomatic complexity.*	*Starting from the teaching of Boolean algebra, it is the easiest way to teach conditional.*	*I believe the explanations given for the short form of conditionals usually have undesirable results in some students. Some of them do not usually understand it easily.*
36.H	*The rules of the game are represented as conditionals*	*With clarity about the rules of a game, it is possible to explain the concept of conditional. Simply the completion of a conditional performs certain tasks that should not be done otherwise.*	*Nothing*
40.H	*Conditionals are taught from the implementation of constraints in a physical simulation. For instance, a trajectory of an object is different when it is on earth, on the moon, on Jupiter; when gravity and atmospheric resistance are conceived as constants for each location*	*MATLAB supports both structured programming and object-oriented programming. In this sense, I believe that with a good appropriation of the tool, the teaching of concepts such as conditionals, loops and functions is very easy.*	*None*
47.M	*practical examples*	*Pretty easy!*	*None*
7.H	Suelo explicar la naturaleza de la instrucción con el fin de bifurcar los caminos de ejecución de un programa en *Phyton*. A través de ejemplos, he notado que los estudiantes comprenden bien el concepto.	Enseñar condicionales para mí es muy fácil. La verdad no encuentro elementos en dicha temática que sean difíciles de explicar.	Quizás sea conveniente profundizar un poco más en el álgebra Booleana, a fin de abordar adecuadamente la temática.
32.M	No tengo una estrategia especial de motivación para enseñar este concepto. Uso la clase magistral.	En términos generales, considero que es fácil enseñar el uso de condicionales.	Según la herencia del lenguaje con ECMA, hay formas cortas de representar los condicionales. Dicha forma suele generar problemas en algunos

			estudiantes.
37.I	Para enseñar los condicionales, simplemente me dedico a explicar el concepto basándome en varias expresiones lógicas de la vida real; por ejemplo: si (está lloviendo) entonces no podré viajar; de lo contrario, iniciaré mi viaje en bicicleta. Cosas como esa.	Realmente considero que enseñar condicionales es una tarea sencilla, sin mayores dificultades.	Por la sintaxis muy particular de C++, algunos estudiantes suelen cometer errores de formulación de estructuras condicionales.
39.D	En este particular no creo tener una estrategia motivacional para su explicación. Trato de introducir el significado directamente en la clase magistral.	Es fácil explicar condicionales en Java.	No encuentro aspectos difíciles de enseñar que incidan en el aprendizaje de los estudiantes sobre el concepto de condicionales.

Fuente: esta investigación

Respuestas sobre el concepto de Ciclo.

RESPUESTAS SOBRE CONCEPTO DE CICLO			
Profesor	**Estrategia Motivacional**	**Facilidad de Enseñar**	**Dificultad de Enseñar**
10.H	*I usually represent a cycle as a train track in a closed circular way. The number of turns that the train can give determines the nature of the loop. With this metaphor, I also explain how a loop can be stopped in a controlled way or in a sudden way.*	*It is easy to explain how a loop gives a known number of iterations.*	*I believe that the control of a loop whose number of iterations is not previously known, generates a bit of difficulty.*
13.V	*I only explain that a group of instructions have the possibility of being executed a controlled number of times, from a conditional. As simple as that. I think my students have not had a hard time understanding that.*	*Loops inherit the behavior of a conditional for the effects of iteration, and not bifurcation. In this vein, it is also easy to teach.*	*Sometimes, students get stuck in infinite loops in practice. I do not see it as a learning problem, but as an oversight of implementation. It is not a big deal!*
24.F	*I just take a group of instructions and have them repeat it a controlled number of times.*	*I believe when I'm teaching such concept, students get it easily*	*Nothing*
25.L	*Traditional lecture. The same explanation for teaching the concept of variable.*	*Simply I explain that an instruction or a group of instructions of a programming language can be repeated a controlled number of times. The concept that I teach in class is*	*None*

		usually well accepted by students.	
27.G	*At this point, I could say that I don't have an explicit motivational strategy. I simply explain the concept as a flow control structure, it involves one or several instructions that can be repeated in a controlled manner.*	*Based on the previous concept, the definition of the loop structure is easy to teach, because a conditional is required.*	*In practice, students usually get stuck in infinite loops. However, I do not consider that it was a failure of the teaching.*
28.X	*Also, I reward students with their algorithms reach a better performance. The foundation in this regard is algorithm complexity.*	*Everything related to loops is easy to explain, in my opinion.*	*None*
36.H	*Every video game maintains the player's interaction with the computer. Thus, this interaction is anticipated when repetition loops are fulfilled. For example, the automatic generation of enemies or obstacles is programmed during a game with loops.*	*The art of managing a timeline with repetition events in a game helps immensely in understanding what a programming loop is.*	*Nothing.*
40.H	*The simulation scenario is immersed within a loop. the actions of such simulations are repeated within the loop's logic. A loop can provide tabular values according to physical functions in 2D or 3D environments.*	*just like I said before.*	*None*
47.M	*practical examples*	*Pretty easy!*	*Not at all*
7.H	Muchos ejercicios	La sintaxis del lenguaje es muy clara al momento de definir ciclos de lógica positiva. Esta es una importante característica que hace que enseñar dichos conceptos sea fácil.	No considero que en mi práctica haya aspectos difíciles de enseñar en los ciclos.
32.M	Tampoco tengo estrategias motivacionales para este concepto. Lo explico en clase de forma tradicional.	Quisiera reportar que la mayoría de mis estudiantes han tenido una respuesta favorable al momento de explicar la naturaleza de los ciclos cuando se realiza a conciencia un diseño previo de estructuras cíclicas.	Tomo nuevamente la descripción de cierta complejidad al momento de realizar las contracciones de incrementos y decrementos aplicadas a los ciclos. Realmente no considero que sea complejo explicarlo, pero por las experiencias

			observadas en los estudiantes, en estos conceptos aplicados suelen cometer errores.
37.I	Al igual que con las expresiones aritmético-lógicas, aquí toca desarrollar una cantidad de ejercicios para afianzar los conceptos. Entre más se entrene a los estudiantes, mejor aprenderán el uso de ciclos.	Iniciar enseñando las partes que definen un ciclo: inicio de variable, comparación o condición, e incremento (o decremento) ayuda a la comprensión al momento de los ejercicios. Trato de hacer que los estudiantes siempre recuerden eso.	Más de un estudiante, en cada curso que he dictado, se atasca en ciclos infinitos.
39.D	En clase ubico una parte de mi tablero para establecer el número de vueltas (iteraciones) que un determinado ciclo efectúa. De esta manera, los estudiantes tienen la posibilidad de observar cómo funciona un ciclo en ejecución.	Es fácil explicar ciclos usando Java como lenguaje de programación.	No encuentro aspectos difíciles de enseñar en ciclos con Java.

Fuente: esta investigación

Respuestas sobre el concepto de Función

RESPUESTAS SOBRE CONCEPTO DE FUNCIÓN			
Profesor	**Estrategia Motivacional**	**Facilidad de Enseñar**	**Dificultad de Enseñar**
10.H	*In class, I'm back to using mathematics. Specifically, through graphical functions and tabulation, I can explain how a function takes values and processes them to return a new numerical value.*	*The declaration of a void function is pretty easy to explain*	*Explaining how the parameters of a function travel by value or by reference is not so easy*
13.V	*In order to explain the concept of function, in the classroom I usually associate this concept with a black box with inputs, process and outputs. We call the inputs parameters, the process is simply the body of the function, and the outputs are the results that the function returns. I usually think in the following way: [I] -> [O = F (I)] -> [O]*	*In my opinion, the easiest part to explain about the functions is that they represent fragments of the program. That explanation is very understandable by the students.*	*Students manifest certain difficulties in learning when I work with pointers as parameters in functions. In some occasions, the fact of passing memory addresses towards a function, usually confuses the students.*

24.F	*As I mentioned from the beginning, I consider that I have traditional teaching strategies. I dare to say that perhaps this concept is the most difficult to teach. In this moment, I only try to explain it in the best possible way by looking at the concept of function as a part of the main program and having the behavior of a mathematical function.*	*It has a high degree of abstraction; however, I think there is an effort to explain it when considering its mathematical relationship.*	*Nothing*
25.L	*Traditional lecture. The same explanation for teaching the concept of variable.*	*I think the concept of function has a higher level of complexity when it is taught. In my case, I see it is easy to understand that functions are part of a program, and those parts can be reused and they are available for programmers as a programming resource.*	*I've noticed that some students have problems with calls to the functions that can be performed in the main code of a program. Those "jumps" usually generate loss of sequentially of execution of a program. Teaching that could be a little bit hard.*
27.G	*Like loop structures, I consider I don't have a motivational strategy to explain the concept of function. I only emphasize that it is about partitioning a program according to its functionality, and these parts can be reused.*	*At the moment of teaching the sending of parameters to the functions by reference, that is to say the sending of memory addresses, it has an implicit difficulty given their level of abstraction. I think teaching this part is challenging. For everything else, I consider it easy to teach.*	*I've been able to realize that my students have certain difficulties when I teach the sending of pointers as parameters. Issues associated with memory addressing are usually not easily understood at the first sight.*
28.X	*From an algorithmic perspective, I encourage my students to build their own functions instead of using existing ones in standard libraries. Perhaps this can be a way of motivation, which is oriented to the development of creativity.*	*When I have to teach functions in C ++, I usually start with the explanation of the function prototype. I consider it the most organized way to work with source code. I encourage my students to prototyping functions.*	*The exchange of values between memory addresses that interact as parameters of a function tends to confuse students a little. Maybe, this is the hardest thing to teach.*

36.H	*Computer games require many actions programmed for each character, time management, movements, animations and special abilities. Each of these characteristics can be programmed by one or several functions. In this way, students are encouraged to create their own functions for common actions.*	*With the use of debuggers, students can track step by step the calls that the program makes to the functions. In this way, it is easy to teach the "leaps" the functions do when invoked.*	*Nothing.*
40.H	*I use the philosophy of "divide-and-conquer" in order to split the functionality of the simulations. Through this philosophy, the concept of function is taught.*	*just like I said before.*	*Perhaps some characteristics of the environment that could generate some difficulty in students are related to the use of toolboxes. Depending on the degree of training in MATLAB, the teaching of the functions immersed in the toolboxes could generate some complications in the students.*
47.M	*practical examples*	*Pretty easy!*	*None*
7.H	Muchos ejercicios en clase.	la declaración de funciones es muy simple. Considero que es la parte más fácil de explicar.	Posiblemente el movimiento de valores que viajan desde y hacia una función de Python podría perder un poco a los estudiantes.
32.M	En esta parte si suelo apoyarme en depuradores con el fin de que los estudiantes puedan ver el movimiento de los parámetros al examinar el contenido de las variables por inspección. Honestamente, no sé si sea motivacional o no, pero los estudiantes si comprenden en realidad dicho concepto después de utilizar los depuradores.	El control de llamadas podría argumentar que es la parte más fácil de explicar el concepto de Función	En mi caso, es sin duda, el traspaso de parámetros o argumentos de una función. Los estudiantes suelen tener algunos problemas entendiendo como se puede transportar parámetros por referencia (como direcciones de memoria de apuntadores)
37.I	Las funciones tienen una parte complicada y es el paso de argumentos, esto suele perder a algunos estudiantes. Cuando yo enseño esa parte, trato de representar los argumentos como	La parte de declaración de funciones es fácil de enseñar. Se debe tener un cuidado especial al momento de usar argumentos.	El paso de punteros como argumentos entre funciones suele confundir a los estudiantes.

	“mensajes” en el tablero.		
39.D	Finalmente, otra área del tablero la destino para llevar un control de las llamadas y paso de argumentos de las funciones. Esta área la dibujo a manera de tabla donde se pueden consignar los datos de cada llamada a función junto con sus argumentos.	Gracias a la sintaxis de Java, la explicación de cómo declarar y usar funciones sea hace muy sencilla.	Curiosamente, he encontrado cierta dificultad al momento de explicar recursividad. Esa capacidad de auto-invocación suele generar cierta confusión en los estudiantes sobre todo cuando hay parámetros de por medio.

Fuente: esta investigación

ANEXO F – Síntesis de respuestas de la entrevista semiestructurada a expertos

Respuesta de la experta de la Universidad de Cádiz, España.

1. Please, describe in general the programming fundamentals course according to the syllabus.
La experta afirmó que se trata de una asignatura obligatoria en el grado (carrera) de Ingeniería informática. Ésta se imparte en el primer año con una asignación de 6 créditos ECTS. De naturaleza teórico-práctica, la asignatura busca generar competencias específicas en materia de fundamentos de programación de computadoras dentro del marco de solución de problemas. Como resultado de la revisión del *syllabus* junto con la experta, se logró identificar los siguientes contenidos generales: Conceptos básicos y definiciones. La Programación de Ordenadores. Objetivos de la Programación. Clasificación de los Lenguajes de Programación. Traductores: compiladores e Intérpretes. Paradigmas de Programación. Concepto de algoritmo. Proceso de creación de un programa. Datos y tipos de datos. Herramientas de descripción de Algoritmos. Características de la Programación Estructurada. Estructura secuencial. Estructura selectiva. Estructuras repetitivas. Estructuras anidadas. Descomposición de problemas y abstracción. Subalgoritmos: funciones, procedimientos. Ámbito y persistencia de las variables. Correspondencia entre argumento y parámetro formal: paso por valor y por referencia, efectos laterales. Funciones y procedimientos como parámetros. Tipos de datos estructurados. Vectores y matrices. Cadenas de caracteres. Registros. Archivos. Tipos enumerados y subrango. Resolución de programas en lenguaje de programación C. Además de estos contenidos, también se estudian en lenguaje de programación C las funciones de manejo de memoria dinámica. La experta afirmó que estos contenidos son generales, y que la implementación en un lenguaje en particular no afecta su contenido dada su orientación hacia el manejo de abstracción de problemas y diseño algorítmico.
2. How long have you taught the programming fundamentals course?
La experta afirmó que lleva más de diez años en la labor de la enseñanza de este tipo de asignaturas y otras más.
3. Do you use analogies to explain the concepts about variables, constants, expressions, conditions, loops, and functions? If so, please describe them.
De acuerdo con su experiencia, la experta afirmó “al inicio de mi labor de enseñanza, utilizaba un estilo tradicional en clase, donde prevalecía mi trabajo sobre el de mis alumnos”. En el presente (al momento en el que fue realizada la entrevista), la experta manifestó que se apoya mucho con la plataforma tecnológica de la Universidad de Cádiz (en especial, el campus virtual de la asignatura). En este sentido, el uso de medios síncronos y asíncronos de interacción con sus alumnos facilita el proceso de enseñanza, y a su vez, propicia un espacio de trabajo autónomo para ellos; la experta afirmó que ha sido notoria la positiva incidencia en los aprendizajes de sus alumnos al usar la plataforma tecnológica como apoyo a su labor en docencia. La experta consideró además que, si bien es cierto que el uso de analogías podría motivar a los estudiantes, desde su perspectiva los mejores resultados en aprendizaje se obtienen de la práctica con disciplina y constancia. De ahí que la experta otorga un valor superlativo a la selección de ejercicios y problemas adecuados al contexto de los estudiantes, de tal suerte que ellos estén ocupando su potencial creativo en solucionar situaciones que les son de su interés.
4. According to your teaching experience, which are the most difficult topics to teach in the programming fundamentals course?
Según la experta, resulta complejo definir temas que sean difíciles de enseñar. La experta aseguró que un(a) profesor(a) cualquiera podría encontrar algo que es difícil de enseñar en la medida en que dicho(a) profesor(a) encuentre a su vez, dificultades en su propia resignificación conceptual frente al dicho tema. Esto es, si se tiene suficiente claridad acerca de un concepto,

existe una gran probabilidad de que dicho concepto sea fácilmente enseñado por el individuo, todo dentro del marco de una práctica de enseñanza adecuada a partir de la didáctica de dicho individuo. Esta respuesta generó un interesante diálogo en profundidad entre la experta y el investigador; aprovechando la semi-estructuralidad de la entrevista, nuevas preguntas surgieron sobre el papel fundamental del profesor, llegando a la conclusión siguiente: El hecho de que un individuo tenga un dominio sobre un tema específico, no necesariamente lo faculta para que pueda enseñarlo. En este sentido, un profesional en alguna rama del saber no está automáticamente habilitado para enseñar temáticas de dicha rama. Es precisamente ahí donde las aptitudes en el ejercicio de la enseñanza (la didáctica) juega un papel fundamental; y dichas aptitudes para la enseñanza, por lo general, no suelen estar presentes en todas las personas, o quizás lo estén en diferentes grados de desarrollo de la habilidad de enseñar. Ella afirmó con simpatía: "Dejemos a los pedagogos que se encarguen de averiguar eso!"

5. According to your teaching experience, which are the most frequent problems of students' learning?

Partiendo de su experiencia de enseñar la asignatura de introducción a la programación (fundamentos de programación), la experta hizo énfasis en que, durante varias generaciones de sus estudiantes, curiosamente los asuntos que se consideran "programación de bajo nivel" usando el lenguaje C ha suscitado problemas en algunos estudiantes. Indagando aún más en la pregunta a través de subpreguntas asociadas, se pudo evidenciar que conceptos de programación de bajo nivel tales como: direccionamiento dinámico de memoria, punteros, instrucciones de pre-procesamiento, compilación condicional, llamadas a funciones de lenguaje ensamblador, manejo de interrupciones a la ROM BIOS, entre otros, suelen generar dificultades en los estudiantes debido a que prioritariamente se está enseñando los conceptos de fundamentos de programación a través de la implementación de algoritmos con un lenguaje de alto nivel. Los conceptos de programación de bajo nivel han sido generados para lenguajes no estructurados como el lenguaje ensamblador, y que formó parte de una generación de programadores en otros tiempos (décadas de los 60s y 70s del siglo pasado). Según la experta, al parecer los estudiantes de hoy miran la programación de bajo nivel como "poco necesaria" para los entornos de desarrollo actuales. A través de un diálogo con el investigador, la experta recordaba que "en otros tiempos, teníamos que preocuparnos por programar la funcionalidad total del ratón desde el bajo nivel para un entorno gráfico. Hoy por hoy, los escenarios de escritorio ya dan por hecho su existencia, así que su funcionalidad básica no requiere programación alguna".

6. Do you consider that the textbooks used by students are suitable for their learning? Why?

Según la experta, la bibliografía que da soporte a su asignatura es buena y es suficiente. Es importante aclarar que la experta enfatizó en que existen libros de texto como bibliografía básica (de hecho, el libro de texto principal fue producido por profesores a través del Servicio de Publicaciones de la de la Universidad de Cádiz), y que dichos textos se deben tomar como libros de referencia para afianzar los temas vistos en clase. En este sentido, la lectura complementaria es importante; pero quizás lo es aún más, la práctica de la programación a través del desarrollo de múltiples ejercicios.

7. Do you use motivation strategies? If so, please describe them.

En cuanto a motivación, la experta afirmó que el mayor interés que se puede generar en los estudiantes se logra por medio de ejercicios prácticos, donde los estudiantes encuentren el sentido real de lo que están aprendiendo. La experta sostuvo que sus estudiantes buscan aplicaciones en el mundo real donde puedan "sacar provecho" a sus conocimientos. Con esto, la experta sustenta la orientación hacia la práctica en contexto real a través de su labor docente, mediante el diseño de ejercicios prácticos, atractivos, e interesantes para sus estudiantes.

8. How do you assess the students' learning?

De acuerdo con la reglamentación vigente en aquel entonces, se elaboran cuestionarios de evaluación: Se realizarán 4 cuestionarios durante el curso correspondientes a los contenidos de la asignatura. Dichos cuestionarios consisten en preguntas tipo *test* que se resolverán a través

del campus virtual preferentemente en horario de clase. Los cuestionarios se evalúan automáticamente a través del campus virtual de la asignatura. Adicional a esto, se exige la entrega de ejercicios resueltos: Se solicita a los alumnos la entrega de 3 ejercicios que se deben resolver y entregar durante las sesiones de laboratorio. Los ejercicios deben desarrollarse siguiendo todos los pasos del proceso de creación de programas. Hay un examen final el cual consta de preguntas teóricas, prácticas y de resolución de problemas tanto en pseudocódigo como en lenguaje C. Los exámenes finales son realizados en las correspondientes convocatorias oficiales y evaluados por el profesorado de la asignatura en los plazos establecidos.
9. Please, describe the kind of students' work that they do out of the classroom (i.e. homework, readings, etc.)
Múltiples ejercicios son planteados a los alumnos, por lo general se prepara una serie de ejercicios de programación de acuerdo con los temas estudiados en clase; esto con el fin de reforzar los conceptos y ponerlos en práctica. Además, se propone una bibliografía básica, una bibliografía específica y una bibliografía de ampliación. Aprovechando el uso de preguntas derivadas, en este aparte, la experta manifestó adicionalmente que es importante seguir el pensamiento de que "la práctica hace al maestro". En este sentido, si bien es cierto que la lectura de bases teóricas es importante, para efectos de aprender a programar ordenadores se requiere más el desarrollo de varios ejercicios. Así, para que realmente un alumno aprenda a programar, debe necesariamente tener contacto permanente con ejercicios de programación. La experta utilizó la siguiente metáfora: "si quieres aprender a nadar, tienes que mojarte un poco!". Con esto se quiso decir que la mejor forma de aprender a programar, es con la práctica constante de la programación de ordenadores.
10. Do you have any additional comment on teaching programming fundamentals?
A juicio de la experta, la enseñanza de los fundamentos de programación debería centrar su atención a partir del planteamiento de problemas de pequeña-mediana envergadura. De esta forma, los estudiantes deberían desarrollar las competencias a fin de saber realizar programas de ordenadores para resolverlos implicando: 1. Saber aplicar los pasos adecuados para la realización de programas. 2. Tener en cuenta los objetivos de la programación. 3. Saber elegir y utilizar los tipos y estructuras de datos adecuadas. 4. Saber elegir y utilizar las estructuras de control adecuadas. 5. Saber realizar la descomposición adecuada e implementar las funciones y procedimientos necesarios correctamente. También afirmó que el aprendizaje de los fundamentos de programación de ordenadores (computadoras) es dependiente de la motivación que los alumnos tengan frente a dicha asignatura; esto hace más interesante la labor del profesor al momento de escoger los ejercicios y actividades más adecuadas para generar interés en sus alumnos.

Fuente: esta investigación

Respuestas del experto de la Universidad Nacional de Colombia, Sede Medellín

1. Please, describe in general the programming fundamentals course according to the syllabus.
Frente a la pregunta, el profesor afirma: "En la Universidad Nacional tenemos una formación de pregrado muy sólida. Desde los primeros semestres, nuestros estudiantes enfrentan grandes retos en su proceso de formación como ingenieros de sistemas e informática ... Según nuestra malla curricular, la asignatura Fundamentos de Programación es obligatoria y deberá cursarse en el primer período académico ... Ahora nosotros estamos enseñando los fundamentos de programación a través del lenguaje *Python…"* Haciendo una revisión del *syllabus*, la asignatura de fundamentos de programación busca analizar problemas de ingeniería y plantear soluciones a los mismos por medio de pseudocódigo, dichas soluciones se implementan posteriormente en un lenguaje de programación. Las temáticas incluyen: Algoritmos, variables sencillas y operaciones básicas, listas (es decir,

algoritmos con listas), instrucciones I/O, lectura y escritura de archivos, funciones simples y propias de *Python*, expresiones lógicas y condicionales simples, condicionales anidados, iteraciones simples (definidas e indefinidas) e iteraciones anidadas, subprogramas, funciones anidadas y recursividad. El profesor afirma, además: "teniendo en cuenta nuestra infraestructura y forma de trabajo, esta asignatura la ayudan a impartir varios profesores auxiliares… Por lo general, ellos [los profesores auxiliares] son estudiantes de maestría o doctorado y cumplen su función como contraprestación en labor docente por ser becarios de nuestra misma facultad".
2. How long have you taught the programming fundamentals course?
El profesor asegura: "Tengo 16 años enseñando en la Facultad de Minas …. Vengo del sector privado y el diseño y desarrollo de software ha sido mi campo de acción. Enseño muchas cosas más en el departamento, sin embargo, todo se relaciona al diseño y construcción de software. La asignatura de fundamentos de programación es básica y obligatoria para estos propósitos".
3. Do you use analogies to explain the concepts about variables, constants, expressions, conditions, loops, and functions? If so, please describe them.
El profesor hace alusión a usar una metodología de enseñanza tradicional, así el uso de las analogías en clase los sitúa en el marco del uso de ejemplos de la vida real. El profesor afirma: "Realmente, hago una exposición tradicional sobre dichos conceptos… No veo que use analogías, a no ser que el uso de ejemplos o ejercicios de la vida real se consideren como tales…. Hago muchos ejercicios en clase, y también, para que los jóvenes los resuelvan en horario extra-clase".
4. According to your teaching experience, which are the most difficult topics to teach in the programming fundamentals course?
Durante el desarrollo de la pregunta, se suscita el diálogo con el profesor, quien considera que realmente no es difícil enseñar los fundamentos de programación, siempre y cuando los estudiantes de dicho curso tengas buenas bases en la matemática y en la lógica. Dada su experiencia, el profesor afirma "De todo lo que se enseña con *Python*, hay algunas cosas que consideramos [el colectivo de profesores que imparte la materia] que son un poco retadoras, sobre todo cuando *Python* no es su [de los estudiantes] primer lenguaje de programación… tomará un tiempo ajustarse a su forma de hacer las cosas". El diálogo con el experto logró evidenciar que aquellos estudiantes que ya tienen una experiencia en programación estructurada, o incluso en programación orientada a objetos en otros lenguajes de programación tales como C++, C# o Java, suelen presentar ciertas dificultades al momento de enfrentar lenguajes como Python por sus características muy particulares. El experto asegura que, para el colectivo de profesores, hay complicaciones en la enseñanza dado que los estudiantes con formación previa en otros ambientes de desarrollo tienen cierta resistencia hacia las características propias del desarrollo basado en *Python.*
5. According to your teaching experience, which are the most frequent problems of students' learning?
Según el profesor, se han logrado detectar 5 situaciones donde los estudiantes presentan mayor dificultad. El profesor afirma que la primera situación hace referencia a la entrada de datos; según el profesor "las entradas de datos siempre son cadenas, por lo tanto, muchos estudiantes suelen olvidar los procesos de filtrado y conversión de tipos al usar funciones como *raw_input*". Otra situación que suele confundir a algunos estudiantes es el uso de las enumeraciones cíclicas. En este particular, el profesor argumenta "*Python* a menudo presenta maneras de hacer cosas diferentes de otros lenguajes de programación populares como C++ o Java... Cuando recorres una matriz en otros lenguajes, tú vas incrementando un número entero desde 0… *Python* ofrece ventajas al usar enumeraciones cíclicas; sin embargo, hemos [los profesores a cargo de la asignatura] notado que algunos estudiantes presentan dificultad usando estas formas". Como tercera situación, el profesor manifestó que la ejecución de comandos externos a través de la consola de *Python* es engorrosa. La cuarta situación se manifiesta a través del uso de excepciones. El profesor dice "*Python* es un lenguaje interpretado, lo que significa que el código se ejecuta línea por línea… Si se encuentra un error en una línea, se detiene la ejecución

adicional del código. Sin embargo, puede manejar excepciones conocidas en *Python* usando el bloque *try-except*. Esta ejecución interrumpida les suele causar problemas [a los estudiantes] por pérdida de secuencialidad". Finalmente, el profesor manifiesta que la separación modular de los programas suele presentar dificultades en los estudiantes. El profesor dice "es suficientemente claro el concepto de división modular, pero para algunos estudiantes les es difícil dividir un problema en partes funcionales… Esta situación requiere de experiencia… ellos [los estudiantes] necesitan tener un grado de madurez en desarrollo para dividir un problema en partes funcionales".
6. Do you consider that the textbooks used by students are suitable for their learning? Why?
De acuerdo con las respuestas dadas por el profesor, y al diálogo como consecuencia de esta respuesta, se logra evidenciar que, si bien es cierto que los textos propuestos como guía de la asignatura son suficientes y que disponen de la información necesaria para el desarrollo de esta, su uso se limita a ser una mera referencia para la formación de los estudiantes. El profesor afirma "Considerando los lineamientos que te dan aquí [la Universidad], se requiere presentar una bibliografía completa y coherente para el desarrollo de cualquier asignatura…. Junto con los compañeros [colectivo de profesores que imparte la asignatura] hemos llegado a un acuerdo sobre la lista de libros que los estudiantes puedan aprovechar…. Es importante que ellos [los estudiantes] tengan acceso a esta bibliografía, por tal razón procuramos que dicha bibliografía este físicamente presente en nuestra biblioteca". El profesor siempre hace hincapié en que lo más importante es el desarrollo de ejercicios para orientar el aprendizaje. Con esto, otorga una importancia mayúscula a la práctica en la asignatura.
7. Do you use motivation strategies? If so, please describe them.
En la respuesta se resalta la orientación motivacional del profesor hacia el planteamiento de bonificaciones académicas para sus estudiantes a partir del desarrollo de ejercidos extra-clase. El profesor describe esta situación de la siguiente manera: "Hay que tener presente que se trata de estudiantes de los primeros semestres de la carrera, y uno [el profesor] quisiera plantearles los beneficios que la investigación en ingeniería brinda, tanto para ellos [los estudiantes] como para los grupos de investigación, el departamento, la facultad y la universidad… Como investigador, mi forma de motivar a los estudiantes es a través de la participación en proyectos de investigación, pero es complicado involucrar estudiantes que recién inician su formación como profesionales… Para captar su interés y dedicación por la asignatura, doy bonificaciones por la entrega a tiempo de los ejercicios propuestos en clase y que ellos [los estudiantes] realizan como trabajo independiente".
8. How do you assess the students' learning?
Actividades previas y complementarias −APyC− (10 %). Trata de que cada estudiante debe realizar semanalmente las tareas dispuestas en la plataforma Moodle. Todas las tareas tienen el mismo peso. Ejercicios de programación −VPL− (5 %). Trata sobre el desarrollo de ejercicios de programación en *Python*. Evaluaciones durante talleres −ET− (50 %). La evaluación será desarrollada durante los talleres en la sala de cómputo de la Universidad Nacional de Colombia y utilizando los equipos disponibles en dicha sala. Todas las actividades tienen el mismo peso. Examen final (35 %). Se realizará al final del semestre durante el horario de la sesión práctica en la sala de cómputo de la Universidad Nacional de Colombia y utilizando los equipos disponibles en dicha sala.
9. Please, describe the kind of students' work that they do out of the classroom (i.e. homework, readings, etc.)
La asignatura tiene dos sesiones por semana: una clase magistral cuyo objetivo es explicar y complementar los aspectos más importantes del contenido; y un taller cuyo objetivo es aplicar los conocimientos adquiridos en la clase teórica a la solución de problemas prácticos utilizando el lenguaje de programación *Python*. El estudiante es responsable de estudiar de manera autónoma y previo a las sesiones el material de referencia sugerido en la bibliografía y demás

recursos complementarios. Las sesiones están diseñadas para que el estudiante desarrolle conceptos fundamentales sólidos sobre los temas tratados. El estudiante debe reforzar dichos conocimientos a partir de la lectura del material sugerido y la elaboración de ejercicios y tareas de programación. En este orden de ideas, el profesor asigna el trabajo a los estudiantes a través de las tareas y otorga bonificaciones a los ejercicios propuestos.
10. Do you have any additional comment on teaching programming fundamentals?
El profesor afirma: "Tengo políticas muy claras en cuanto al desarrollo de esta asignatura. Siempre suelo habilitar un foro de discusión para que los estudiantes puedan interactuar de forma colaborativa, con el fin de resolver dudas o inquietudes sobre los temas del curso… Prohíbo cualquier publicación relacionada con los temas de las evaluaciones. Los estudiantes que publiquen las soluciones de exámenes y tareas tendrán una nota de cero en la actividad correspondiente y serán procesados por fraude ante las instancias respectivas. El foro siempre es monitoreado constantemente por el colectivo de profesores… ellos [el colectivo de profesores] con una gran ayuda para controlar el progreso de los estudiantes". Teniendo presente esta afirmación, el profesor otorga gran importancia al colectivo de profesores que acompaña el proceso de formación de los estudiantes que cursan la asignatura de fundamentos de programación; además, hace referencia a la interacción y a la colaboración en foros de tal suerte que comulga con los principios del aprendizaje activo y colaborativo.

Fuente: esta investigación

ANEXO G – Síntesis de respuestas de la guía de vigilancia epistemológica

RESPUESTAS SOBRE GUIA DE VIGILANCIA EPISTEMOLÓGICA - OBJETO DE SABER: VARIABLE				
Profesor	**Definición del texto guía**	**Escriba la definición usando sus propias palabras**	**Lenguaje**	**Referencia de Texto Guía usado en clase**
24.F	*"The basis of a program is data, in the form of numbers and characters. The storing, manipulation and output of this data gives functionality. Variables hold this data, think of it as a box used to store a single item" (Seacord, 2013, p. 35)*	*A Variable is a memory space to store data by using a programming language. Variables can store values according to their datatypes.*	*C/C++*	*Seacord, R. (2013). Secure Coding in C and C++. 2nd Edition. Addison-Wesley Professional. USA.*
25.L	*"This chapter introduces the basics of storing and using data in a program. To do so, we first concentrate on reading in data from the keyboard. After establishing the fundamental notions of objects, types, values, and variables, we introduce several operators and give many examples of use of variables of types char, int, double, and string" (Stroustrup, 2014, p. 59)*	*Variables allow storing data in memory during a program execution. Like in Math, variables in computing can vary their content, depending on the assignment of values.*	*ISO/IEC C++ 2011*	*Stroustrup, B. (2014). Programming: Principles and Practice Using C++ 2nd Edition. Addison-Wesley Professional. USA*
28.X	*"A variable provide us with named storage that our programs can manipulate. Each variable in*	*Through identifiers, variables are named to create memory spaces to store data.*	*C++*	*Lippman, S., Lajoie, J., & Moo, B. (2015). C++ Primer, 5th Edition. Pearson Education. USA.*

	C++ has a type" (Lippman, Lajoie & Moo, 2015, p. 41)			
36.H	*"A general-purpose programming language must allow a program to store and retrieve data. A variable is a name that represents data stored in memory during program execution" (Solis, 2011, p. 45)*	*Programmers use variables to store data in memory. Several situations in games require variables when data must vary; for instance, the position of an object in motion.*	*C#*	*Solis, D. (2011) Illustrated C# 2012, 4th Edition. Apress. USA*
40.H	*"To store a value in a MATLAB sesion, or in a program, a variable is used. The Workspace Window shows variables that have been created and their values" (Attaway, 2013, p. 6)*	*A variable is a memory space that is designated through an identifier to store values. Matlab creates the respective variable reserving the necessary memory space. Therefore, if the variable already exists, Matlab only changes its content.*	*MATLAB*	*Attaway, S. (2013). Matlab: A Practical Introduction to Programming and Problem Solving, 3rd Edition. Butterworth-Heinemann. USA.*
37.I	"Una variable es un objeto o tipo de datos cuyo valor puede cambiar durante el desarrollo del algoritmo o ejecución del programa. Dependiendo del lenguaje, hay diferentes tipos de variables, tales como enteras, reales, carácter, lógicas y de cadena. Una variable que es de un cierto tipo puede tomar únicamente valores de ese tipo" (Joyanes,	Una variable es declarada a través de un identificador que representa un espacio de memoria para almacenar datos, esto sucede mientras el programa que creo las variables está en ejecución. Los valores almacenados en las variables pueden cambiar durante la ejecución de un programa.	C++	Joyanes, L. (2008). Fundamentos de programación, $th Edition. McGraw-Hill, España.

	2008, p. 93)			
RESPUESTAS SOBRE GUIA DE VIGILANCIA EPISTEMOLÓGICA - OBJETO DE SABER: CONSTANTE				
Profesor	**Definición del texto guía**	**Escriba la definición usando sus propias palabras**	**Lenguaje**	**Referencia de Texto Guía usado en clase**
24.F	*"As Variables. Constants also hold data, but their values cannot be changed" (Seacord, 2013, p. 36)*	*Like variables, constants store data in memory, but their values can not be altered.*	*C/C++*	*Seacord, R. (2013). Secure Coding in C and C++. 2nd Edition. Addison-Wesley Professional. USA.*
25.L	*"Programs typically use a lot of constants. For example, a geometry program might use pi and an inch-to-centimeter conversion program will use a conversion factor such as 2.54. Obviously, we want to use meaningful names for those constants (as we did for pi; we didn't say 3.14159). Similarly, we don't want to change those constants accidentally" (Stroustrup, 2014, p. 95)*	*Constants in programming have the same connotation as in mathematics, they represent values that should not change.*	*ISO/IEC C++ 2011*	*Stroustrup, B. (2014). Programming: Principles and Practice Using C++ 2nd Edition. Addison-Wesley Professional. USA*
28.X	*"A constant is a assigned value to an identifier which cannot be modified" (Lippman, Lajoie & Moo, 2015, p. 59)*	*The value assigned to a constant in a C++ program cannot change.*	*C++*	*Lippman, S., Lajoie, J., & Moo, B. (2015). C++ Primer, 5th Edition. Pearson Education. USA.*

36.H	*"A local constant is much like a local variable, except that once it is initialized, its value can't be changed. Like a local variable, a local constant must be declared inside a block." (Solis, 2011, p. 73)*	*Constants represent values that cannot be changed; for instance, the size of the chessboard (No matter how many chess matches are played, the chessboard will always have the same size).*	*C#*	*Solis, D. (2011) Illustrated C# 2012, 4th Edition. Apress. USA*
40.H	*"Variables are used to store values that can change, or that are not known ahead of time. Most languages also have the capacity to store constants, which are values that are known ahead of time, and cannot possibly change" (Attaway, 2013, p. 14)*	*Matlab has predefined constants. In practice, they are enough to work with. However, if a constant defined by the programmer is required, it can be created through the assignment of a fixed value to a variable.*	*MATLAB*	*Attaway, S. (2013). Matlab: A Practical Introduction to Programming and Problem Solving, 3rd Edition. Butterworth-Heinemann. USA.*
37.I	"Una constante es un dato que permanece sin cambios durante todo el desarrollo del algoritmo o durante la ejecución del programa" (Joyanes, 2008, p. 93)	Para crear constantes se usan también identificadores. Las constantes almacenan datos que, una vez asignados, no podrán ser modificados.	C++	Joyanes, L. (2008). Fundamentos de programación, $th Edition. McGraw-Hill, España.

RESPUESTAS SOBRE GUIA DE VIGILANCIA EPISTEMOLÓGICA - OBJETO DE SABER: EXPRESIÓN ARITMÉTICO-LÓGICA

Profesor	**Definición del texto guía**	**Escriba la definición usando sus propias palabras**	**Lenguaje**	**Referencia de Texto Guía usado en clase**
24.F	*"Numbers and booleans are used as expressions" (Seacord, 2013, p. 41)*	*Expressions that can be computed (arithmetic and logical) are formed by numbers and Boolean values with their respective operators.*	*C/C++*	*Seacord, R. (2013). Secure Coding in C and C++. 2nd Edition. Addison-Wesley Professional. USA.*

25.L	*"We need to back off, stop programming for a while, and think about how we read and understand an input string and evaluate it as an arithmetic expression. They are composed by numbers and arithmetic operators to do some calculations with them" (Stroustrup, 2014, p. 184)* *"A logical expression is much like an arithmetic expression except that the operators are ! (not) , - (complement), & (and), I (or) , and " (exclusive or) . ! and - are prefix unary operators, A logical expression returns a boolean value" (Stroustrup, 2014, p. 217)*	*Arithmetic calculations can be made through expressions, they use numbers and math operators. On the other hand, logical expressions use Boolean values and Boolean operators to be evaluated in two possible return values: true or false. Both types of expressions represent statements of the mathematical and algebra scenario.*	*ISO/IEC C++ 2011*	*Stroustrup, B. (2014). Programming: Principles and Practice Using C++ 2nd Edition. Addison-Wesley Professional. USA*
28.X	*"The smallest unit of computation. An expression consists of one or more operands and usually one or more operators. Expressions are evaluated to produce results" (Lippman, Lajoie & Moo, 2015, p. 133)*	*In order to perform mathematical or logical calculations, expressions are used. An expression is composed of operands (numbers or Boolean values) and operators (arithmetic or logical-relational)*	*C++*	*Lippman, S., Lajoie, J., & Moo, B. (2015). C++ Primer, 5th Edition. Pearson Education. USA.*

36.H	*"An operator is a symbol that represents an operation that returns a single result. An operand is a data element used as input by the operator. An operator does the following: Takes its operands as input, Performs an action, Returns a value, based on the action. An expression is a string of operators and operands" (Solis, 2011, p. 202)*	*An expression represents an arithmetic calculation or a Boolean calculation. An expression is written as a sequence of values and symbols that comply with specific syntax rules. In games, expressions help to establish mathematical calculations. For example, trajectories, scores, physical effects, among others.*	*C#*	*Solis, D. (2011) Illustrated C# 2012, 4th Edition. Apress. USA*
40.H	*"Expressions can be created using values, variables that have already been created, operators, built-in functions, and parentheses. For numbers, these can include operators such as multiplication, and functions such as trigonometric functions." (Attaway, 2013, p. 10)* *"Conditions in if statements use expressions that are conceptually, or logically, either true or false. These expressions are called relational expressions, or sometimes Boolean or logical expressions. These expressions can use both relational*	*Matlab is a Matrix Calculator. In this vein, arithmetic expressions and logical expressions are part of their language core. Expressions are very close to those that are used in mathematics and algebra. They are composed of values and operators. In practice, every physical phenomenon has associated mathematical models that represent them. Mathematical models require the use of expressions for their definition.*	*MATLAB*	*Attaway, S. (2013). Matlab: A Practical Introduction to Programming and Problem Solving, 3rd Edition. Butterworth-Heinemann. USA.*

	operators, which relate two expressions of compatible types, and logical operators, which operate on logical operands" (attaway, 2013, p. 80)			
37.I	"Las expresiones son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales. Las mismas ideas son utilizadas en notación matemática tradicional" (Joyanes, 2008, p. 94)	Las expresiones aritmético-lógicas con parte fundamental de los lenguajes de programación ya que permiten la realización de cálculos. Estas se componen de valores númericos (en el caso de las expresiones aritméticas) o de valores booleanos (en el caso de las expresiones lógicas) los cuales son conectados a través de operadores segun corresponda.	C++	Joyanes, L. (2008). Fundamentos de programación, $th Edition. McGraw-Hill, España.
RESPUESTAS SOBRE GUIA DE VIGILANCIA EPISTEMOLÓGICA - OBJETO DE SABER: CONDICIONAL				
Profesor	**Definición del texto guía**	**Escriba la definición usando sus propias palabras**	**Lenguaje**	**Referencia de Texto Guía usado en clase**
24.F	*"Comparing values to other values in a meaningful way is fundamental for conditional statements... The List of conditionals above can be used in certain circumstances to control the flow of the program" (Seacord, 2013, p. 59)*	*Conditionals in programming are mechanisms to execute a series of instructions when a logical expression is fulfilled, these are accompanied by logical operators.*	*C/C++*	*Seacord, R. (2013). Secure Coding in C and C++. 2nd Edition. Addison-Wesley Professional. USA.*

25.L	*"This is the ability to test a variable against a value and act in one way if the condition is met by the variable or another way if not. They are also commonly called by programmers if statements" (Stroustrup, 2014, p. 226)*	*Conditional structures evaluate logical expressions, so a course of action is followed within the program, which is based on the result of such evaluation.*	*ISO/IEC C++ 2011*	*Stroustrup, B. (2014). Programming: Principles and Practice Using C++ 2nd Edition. Addison-Wesley Professional. USA*
28.X	*"An algorithm structure which expression is evaluated as true or false. Flow control depends on such evaluation." (Lippman, Lajoie & Moo, 2015, p. 175)*	*A conditional is a flow control structure to evaluate a logical expression.*	*C++*	*Lippman, S., Lajoie, J., & Moo, B. (2015). C++ Primer, 5th Edition. Pearson Education. USA.*
36.H	*"Selection statements: These statements allow you to select which statement or block of statements to execute. - if: Conditional execution of a statement - if ... else: Conditional execution of one statement or another" (Solis, 2011, p.74)*	*Programming games, we almost always need the ability to check certain conditions and change the behavior of the game according to the present condition. Conditional sentences give us this ability.*	*C#*	*Solis, D. (2011) Illustrated C# 2012, 4th Edition. Apress. USA*
40.H	*"A condition is a relational expression that is conceptually, or logically, either true or false. The action is a statement, or a group of statements, that will be executed if*	*Conditionals are control structures that include selection alternatives based on the result of a Boolean operation (The result of a Boolean operation is always true or false). In a conditional statement, a conditional expression is evaluated. If the expression is true,*	*MATLAB*	*Attaway, S. (2013). Matlab: A Practical Introduction to Programming and Problem Solving, 3rd Edition. Butterworth-Heinemann. USA.*

	the condition is true. When the if statement is executed, first the condition is evaluated. If the value of the condition is conceptually true, the action will be executed, and if not, the action will not be executed. The action can be any number of statements until the reserved word end; the action is naturally bracketed by the reserved words if and end" (Attaway, 2013, p. 83)	*the group or block of commands are executed. If the expression is false, MATLAB does not execute (skip) the group of commands in question.*		
37.I	"Las estructuras selectivas se utilizan para tomar decisiones lógicas; de ahí que se suelan denominar también estructuras de decisión o alternativas. En las estructuras selectivas se evalúa una condición y en función del resultado de la misma se rea liza una opción u otra. Las condiciones se especifican usando expresiones lógicas" (Joyanes, 2008, p. 130)	En programación se hace necesario ejecutar instrucciones dependiendo de ciertas situaciones que resultan de la evaluación de esprexiones lógicas. Las estructuras condicionales permiten realizar esa selección de instrucciones a ejecutarse.	C++	Joyanes, L. (2008). Fundamentos de programación, $th Edition. McGraw-Hill, España.
RESPUESTAS SOBRE GUIA DE VIGILANCIA EPISTEMOLÓGICA - OBJETO DE SABER: CICLO				
Profeso	**Definición del**	**Escriba la definición**	**Lenguaj**	**Referencia de Texto Guía**

r	texto guía	usando sus propias palabras	e	usado en clase
24.F	*"Iteration means looping, and looping quickly gives programs the ability to perform lots of similar operations very quickly." (Seacord, 2013, p. 72)*	*Loops are algorithmic structures that contain a set of instructions that are repeated a finite number of times by evaluating logical expressions.*	*C/C++*	*Seacord, R. (2013). Secure Coding in C and C++. 2nd Edition. Addison-Wesley Professional. USA.*
25.L	*"Loops are used to repeat a block of code. Being able to have your program repeatedly execute a block of code is one of the most basic but useful tasks in programming" (Stroustrup, 2014, p. 228)*	*Loops are control structures that repeat the execution of a group of instructions. Basically, a logical expression is required, since within it repeats the execution of one or more instructions while a specific condition is met.*	*ISO/IEC C++ 2011*	*Stroustrup, B. (2014). Programming: Principles and Practice Using C++ 2nd Edition. Addison-Wesley Professional. USA*
28.X	*"Iterative statements mean repetition of set of statements depending upon a condition test. Till the time of condition is true. (or false depending upon the loop). A set of statements are repeated again and again" (Lippman, Lajoie & Moo, 2015, p. 183)*	*Loop structures allow the execution of a instruction block several times. The number of times the instruction block will be executed can be specified explicitly through a logical condition. Each execution of the instruction block is known as an iteration.*	*C++*	*Lippman, S., Lajoie, J., & Moo, B. (2015). C++ Primer, 5th Edition. Pearson Education. USA.*
36.H	*"Looping statements repeatedly execute a section of code. The looping statements are the following: - while - do - for - foreach" (Solis, 2011, p. 242)*	*A loop allows executing one or more instructions several times, that is, it allows executing a block of instructions repeatedly, by verifying logical expressions. Games require the use of loops to execute several tasks that are repeated continuously.*	*C#*	*Solis, D. (2011) Illustrated C# 2012, 4th Edition. Apress. USA*

40.H	*"A conditional loop also repeats statements, but ahead of time it is not known how many times the statements will need to be repeated. With a conditional loop, for example, you might say "repeat these statements until this condition becomes false." The statement(s) that are repeated in any loop are called the action of the loop." (Attaway, 2013, p. 110)*	*Loops in Matlab are algorithmic structures that allow the repeated execution of a group of instructions according to the evaluation of Boolean expressions.*	*MATLAB*	*Attaway, S. (2013). Matlab: A Practical Introduction to Programming and Problem Solving, 3rd Edition. Butterworth-Heinemann. USA.*
37.I	"Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan bucles y se denomina iteración al hecho de repetir la ejecución de una secuencia de acciones" (Joyanes, 2008, p. 158)	Los ciclos son estructuras de control que permiten la ejecución controlada de código de programa varias veces. Los ciclos requieren variables de control que son iniciadas, incrementadas (o decrementadas) y son comparadas para determinar su fin.	C++	Joyanes, L. (2008). Fundamentos de programación, $th Edition. McGraw-Hill, España.

RESPUESTAS SOBRE GUIA DE VIGILANCIA EPISTEMOLÓGICA - OBJETO DE SABER: FUNCIÓN

Profesor	Definición del texto guía	Escriba la definición usando sus propias palabras	Lenguaje	Referencia de Texto Guía usado en clase
24.F	*"Functions are the building blocks of a program, they allow the reuse of code, the ability to kepp it readable and stops the programmer*	*A function is a module of a program, it is separated from the main body. A function performs a specific task and can return a value to the main part of the program or another*	*C/C++*	*Seacord, R. (2013). Secure Coding in C and C++. 2nd Edition. Addison-Wesley Professional. USA.*

	repeating code" (Seacord, 2013, p. 104)	*function or procedure that invokes it.*		
25.L	*By "parts of a program" we mean entities such as a function producing a result from a set of input arguments (e.g., a square root from a floating-point number), a function performing an action on a physical object (e.g., a function drawing a line on a screen) , or a function modifying some table within the program (e.g., a function adding a name to a table of customers)" (Stroustrup, 2014, p. 91)*	*A function in C ++ can be defined as a set or group of sentences that are well defined and that perform a particular task, the advantage of using them lies in the fact that we can program more robust applications by dividing the complexity of the problem through functions.*	*ISO/IEC C++ 2011*	*Stroustrup, B. (2014). Programming: Principles and Practice Using C++ 2nd Edition. Addison-Wesley Professional. USA*
28.X	*"A fragment of code that appears in your program in multiple places can be placed into a function definition and replaced by a function call instead, resulting in a smaller, more maintainable program. It is smaller because there are fewer lines of code. It is more maintainable because if you decide to change the code, you only have to do it in one place, instead of searching through*	*A function is a part of the program that can be invoked or called to execute as many times as necessary. A function can optionally do two things: receive data and return data.*	*C++*	*Lippman, S., Lajoie, J., & Moo, B. (2015). C++ Primer, 5th Edition. Pearson Education. USA.*

	the entire program for all occurrences of that code fragment" (Lippman, Lajoie & Moo, 2015, p. 202)			
36.H	*"A method is a named block of executable code that can be executed from many different parts of the program, and even from other programs. When a method is called, or invoked, it executes the method's code, and then returns to the code that called it and continues executing the calling code. Some methods return a value to the position from which they were called. Methods correspond to member functions in C++." (Solis, 2011, p. 55)*	*A function is an element of the program that contains code and can be executed, that is, it does an operation. A function can be called or invoked when necessary and then the code will be executed. Once the function finishes executing, the control of the program is returned to the next point where the function was called. In games, functions are widely used to perform specific tasks.*	*C#*	*Solis, D. (2011) Illustrated C# 2012, 4th Edition. Apress. USA*
40.H	*"In general, any function in MATLAB consists of: 1)The function header (the first line); this has: – the reserved word function – (if the function returns values, the name(s) of the output argument(s) followed by the assignment*	*In Matlab, a function is a set of instructions that are written separately from the program and that perform some specified task. Users can define functions and add them to functions own Matlab.*	*MATLAB*	*Attaway, S. (2013). Matlab: A Practical Introduction to Programming and Problem Solving, 3rd Edition. Butterworth-Heinemann. USA.*

	operator =) *– the name of the function* *(Important: This should be the same as the name of the M-file in which this function is stored in order to avoid confusion)* *– the input arguments in parentheses, if there are any (separated by commas if there is more than one)* *2) A comment that describes what the function does (this is printed if help is used)* *3) The body of the function, which includes all statements, including assigning values to all output arguments if there are any" (Attaway, 2013, p. 162)*			
37.I	"Matemáticamente una función es una operación que toma uno o más valores llamados argumentos y produce un valor denominado resultado —valor de la función para los argumentos dados—. Todos los lenguajes de programación tienen funciones incorporadas, intrínsecas o internas —en el Capítulo 3 se vieron algunos ejemplos—, y	Las funciones son partes de un programa las cuales ejecutaran una determinada tarea al momento en que son llamadas. Pueden devolver valores y pueden también recibir valores a manera de parámetros.	C++	Joyanes, L. (2008). Fundamentos de programación, $th Edition. McGraw-Hill, España.

	funciones definidas por el usuario." (Joyanes, 2008, p. 203)			

Fuente: esta investigación.

ANEXO H – Síntesis de respuestas de la entrevista semi-estructurada a profesores

TRANSCRIPCIÓN DE RESPUESTAS DE LA ENTREVISTA SEMI-ESTRUCTURADA	
Pregunta 1:	***How long have you taught the courses related to programming fundamentals?***
Profesor	**Transcripción de Respuesta**
24.F	*At the beginning, I started teaching programming in C… Then, the name of the course changed…. I've taught programming fundamentals for 17 years.*
25.L	*I've been teaching for 5 years.*
28.X	*My first classes were related to mathematics in Computer Science...I teach programming fundamentals for 12 years.*
36.H	*7 years.*
40.H	*I just can't remember how long I've been teaching this…Perhaps, more than 20 years…I'm not sure.*
37.I	Llevo trabajando con la Udenar más de 20 años… he impartido varias veces fundamentos de programación.
Pregunta 2:	***Do you use analogies to explain the concepts about variables, constants, expressions, conditions, loops, and functions? If so, please describe them.***
Profesor	**Transcripción de Respuesta**
24.F	*Sometimes, I remember using boxes to represent the concepts of variables and constants.*
25.L	*I don't think so...I only explain in the best possible way according to my experience.*
28.X	*No…I think my practice in teaching is based on a traditional lecture.*
36.H	*Always… I've based my teaching on programming exercises related to games.*
40.H	*I try to use real-life problems, especially in the field of physics…. I think I couldn't talk about analogies, but rather about solving real-life problems… I usually start my lectures with a Math introduction according to the lecture's topic.*
37.I	Quizás las analogías que hago se toman a partir del mundo de la matemática.
Pregunta 3:	***According to your teaching experience, which are the most difficult topics to teach in the programming fundamentals?***
Profesor	**Transcripción de Respuesta**
24.F	*Honestly, I don't find it difficult to teach programming fundamentals.*
25.L	*I don't see difficulty in teaching… Perhaps, the difficulty can arise in learning when students don't have sufficient foundations in Math.*
28.X	*I think that some low-level programming elements are a bit challenging to be taught by their level of abstraction.*
36.H	*Nothing.*
40.H	*Nothing….Teaching programming fundamentals with MATLAB is pretty easy ... The language is practically the mathematics itself.*
37.I	He notado que la formación en bachillerato es crucial… Enseñar la parte expresiones aritmético-logicas suele ser un poco complicado cuando la formación previa de los estudiantes no es buena.
Pregunta 4:	***According to your teaching experience, which are the most frequent problems of students' learning?***
Profesor	**Transcripción de Respuesta**
24.F	*The proper writing of arithmetic and logical expressions.*
25.L	*the sequentiality of a program...I've noticed that several students in the course of programming fundamentals tend to get lost in the middle of calls to functions, especially when they are recursive.*

28.X	*C++ Language's syntax.*
36.H	*Nothing.*
40.H	*The adaptation of physical models.*
37.I	La abstracción y representación de expresiones matemáticas.
Pregunta 5:	***Do you consider that the textbooks used by your students are suitable for their learning? Why?***
Profesor	**Transcripción de Respuesta**
24.F	*Yes. It is widely recommended by the academic community in my region.*
25.L	*Of course. The way in which the textbook was written is appropriate for students.*
28.X	*Yes. The textbooks were suggested by a group of professors of the Department.*
36.H	*Yes. However, I only use the textbook as a C# language reference. The important thing is what can be done from the knowledge of such language.*
40.H	*Yes. The authors use lots of practical exercises into a clever learning path.*
37.I	Tradicionalmete se ha trabajo con esta serie de libros en la Udenar…Hemos tenido buenos resultados al usar este libro de referencia en fundamentos de programación.
Pregunta 6:	***Do you use motivational strategies? If so, please describe them.***
Profesor	**Transcripción de Respuesta**
24.F	*I believe that the best motivation for students is through the development of activities to improve their grades.*
25.L	*The motivation is implicit in my speech within the lectures.*
28.X	*Students always come with expectations on computer programming… I try to maintain these expectations with practical exercises according to the students' interests.*
36.H	*I consider that the main motivation comes from the orientation towards the development of games.*
40.H	*I look at the motivation in the fact of being able to solve real problems in the field of math and physics through the programming fundamentals.*
37.I	Los fundamentos de programación son la entrada de los estudiantes hacia el mundo de la programación... La motivación está en poder evidenciar el potencial de los estudiantes en cuanto a la solución de problemáticas reales.
Pregunta 7:	***How do you assess the students' learning?***
Profesor	**Transcripción de Respuesta**
24.F	*1 mid-term exam, 1 final exam, and 2 workshops.*
25.L	*Faculty examination… 2 exams*
28.X	*3 exams… one for each month within a quarter.*
36.H	*Course project*
40.H	*Exercises and 1 final exam.*
37.I	He tenido la costumbre de realizar 2 evaluaciones parciales y una evaluación final... Un porcentaje de las evaluaciones parciales se promedia con ejercicios de trabajo independiente… Los porcentajes son acordados con los estudiantes al iniciar el semestre.
Pregunta 8:	***Please, describe the kind of students' work that they do out of the classroom (i.e. homework, projects, etc.)***
Profesor	**Transcripción de Respuesta**
24.F	*Set of exercises.*
25.L	*Homeworks...the amount depends on how the class progresses.*
28.X	*3 homeworks. One for each month within a quarter.*
36.H	*6 homeworks and a course project.*
40.H	*Lots of exercises.*

37.I	A los estudiantes se les deja 2 trabajos para entregar a mitad de semestre y al finalizar el semestre…. Por su parte, los estudiantes tienen que resolver varios ejercicios.
Pregunta 9:	***Do you have any additional comment on teaching programming fundamentals?***
Profesor	**Transcripción de Respuesta**
24.F	*Previously, you need good computer labs to program. Today, such infrastructure is no longer necessary…. Students with their mobile devices (laptops, tablets, smartphones, etc.) have all the resources they need to work on programming fundamentals.*
25.L	*I wonder where you want to go? Some questions are very similar to those I answered in the survey…. Good luck!*
28.X	*No.*
36.H	*No, I don't.*
40.H	*No comments.*
37.I	Para nosotros, el éxito en los fundamentos de programación radica en la lógica y en la algoritmia.

Fuente: esta investigación.

ANEXO I – Diseño y desarrollo didáctico para cada estrategia de enseñanza basada en Transposición Didáctica *in Extensa Sensu*

Diseño y desarrollo didáctico para la estrategia de enseñanza del objeto de saber: Variable
Diseño Didáctico
El diseño didáctico parte con los niveles de transposición que experimentan los objetos. Según esta investigación. El primer nivel de transposición hace referencia al objeto de saber; en este caso, **el objeto de saber: Variable**. Como referente conceptual, se toma la definición dada en la página 89, donde una variable es un espacio de memoria de las computadoras para almacenar un valor a través del uso de identificadores. Las variables tienen direccionamiento de memoria y poseen un tipo de dato. Para pasar al segundo nivel de transposición se hace necesario la discusión interna en la noosfera. Según esta propuesta, la noosfera debe estar conformada por el grupo de profesores a cargo de fundamentos de programación, la participación del director del programa académico (o su equivalente) y profesionales de la industria y del sector productivo. De esta forma, la noosfera ya conformada debe identificar las necesidades del contexto local y que puedan ser abordadas desde los fundamentos de programación. A manera de ejemplo, se toma el contexto local propio de la ciudad de Pasto (Colombia). Así, una situación real es el riesgo volcánico. Es aquí donde se efectúa la segunda transposición, cuyo objeto de saber Variable dado como definición general se sitúa al tenor del ejemplo del riesgo volcánico, así, el **objeto a enseñar: Variable** se transpone como: variable es un espacio de memoria de las computadoras para almacenar un valor a la vez a través del uso de identificadores. Un programa de computadoras hace operaciones de lectura y escritura de las variables a través del tiempo. Las variables tienen direccionamiento de memoria y poseen un tipo de dato. Dentro del diseño didáctico, el curso de fundamentos de programación se trabaja a través del desarrollo de un proyecto de curso. Entiéndase por proyecto de curso al ejercicio práctico de aplicación a mediano plazo (cuya duración se extiende a la del mismo curso de fundamentos de programación) donde los conocimientos del curso se implementan en resolver un problema específico. Depende del profesor otorgar su valoración porcentual frente a la calificación final del curso. Para este ejemplo, un proyecto de curso podría proponerse como la construcción de un conjunto de programas para sismología, iniciando con un programa básico para dispositivos móviles que permita la detección de movimientos sísmicos a través del acelerómetro.
Desarrollo Didáctico
El primer paso es establecer actividades que permitan develar los preconceptos de los estudiantes

al interior del aula de clase. Se hace necesario indagar sobre los que los estudiantes entienden sobre el concepto de variable. Para ello se prevé realizar una sesión de lluvia de ideas a partir de los aportes de los estudiantes a dicho concepto.

Es probable que los preconceptos de los estudiantes surjan a partir de su experiencia en matemática y álgebra. Usando dichos preconceptos se hacen ejercicios de recapitulación del concepto matemático y del algebra. Una vez concluidos estos ejercicios previos. Se procede a construir colectivamente el nuevo **objeto de enseñanza: Variable,** como el tercer nivel de transposición. De esta manera el profesor propone la definición: variable es un espacio de memoria de las computadoras para almacenar un valor a la vez a través del uso de identificadores, su sintaxis depende del lenguaje de programación. Un programa de computadoras en ejecución hace operaciones de asignación y recuperación de valores de las variables a través del tiempo. Las variables tienen direccionamiento de memoria y poseen un tipo de dato. Esta definición será debatida al final de clase con los aportes de los estudiantes.

Como motivación, el profesor propone una partida del juego de hacer parejas en un escenario de cartas ocultas, el juego se compone de una serie de fichas que contienen ejemplos de valores y tipos de datos. Este ejercicio debe estar articulado con el proyecto de curso a realizar, es decir, el programa para detección de movimientos sísmicos a través del acelerómetro de los dispositivos móviles. La idea es en equipos de estudiantes, cada equipo elabore 64 fichas en hojas de papel (32 de ellas contienen tipos de datos y los 32 restantes contienen valores que corresponden a cada tipo de dato). Una vez elaboradas las fichas, cada equipo las coloca invertidamente en forma de tabla de 8 x 8. Cada equipo inicia el juego. Con varias partidas del juego, las fichas se empiezan a intercambiar entre grupos (los cambios se hacen de las 64 fichas totales) y se juegan más partidas.

Después de los juegos de motivación, el profesor distribuye en sus estudiantes una guía de desarrollo grupal para afianzar el concepto de variable a través de ejercicios. Finalmente se procede a hacer las prácticas en las computadoras para aplicar el concepto de variable en un lenguaje de programación con varios ejemplos.

A manera de valoración del aprendizaje, se crea conjuntamente un **objeto de evaluación: Variable,** el cual contempla elementos que permitan valorar el aprendizaje de los estudiantes, este objeto de valoración se compone de un desarrollo de ejercicios individuales, un trabajo en colectivo (puede ser grupal o en parejas) unas rúbricas para autoevaluación y evaluación intergrupal. Finalizando esta actividad, el profesor elabora un informe de clase para ser analizado nuevamente por la noosfera a fin de retroalimentar el **objeto a enseñar**.

Diseño y desarrollo didáctico para la estrategia de enseñanza del objeto de saber: Constante
Diseño Didáctico

Muy similar al objeto de saber Variable, el objeto de saber: Constante, parte de la definición ontológica dada en la página 90. Así, el **objeto de saber: Constante** está dado por la siguiente definición: 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.

El segundo nivel de transposición se debe realizar a través de la participación de la noosfera. En este orden de ideas, el profesor (o los profesores) a cargo del curso de fundamentos de programación junto con el director del programa (o su equivalente) y representantes del sector de la industria y de la producción profesional arman en conjunto la definición que se debe enseñar. A esto se le llama el objeto a enseñar. Este objeto necesariamente deberá responder a necesidades o situaciones del contexto local. Como ejemplo para esta serie de diseños y desarrollos didácticos que acompañan a las estrategias, se ha tomado el riesgo volcánico como escenario de aplicación. Así, la temática relacionada a las constantes estará asociada a dicho escenario de aplicación. Aquí se da continuidad con el ejercicio básico que forma parte del proyecto de curso, el ejercicio se trata de la construcción de un programa para dispositivos móviles que permita la detección de sismos usando el acelerómetro.

De esta manera, se construye colectivamente el **objeto a enseñar**: Constante con la siguiente definición: Una constante es un espacio de memoria de las computadoras que almacena un valor inmodificable y corresponde a un identificador y tipo de dato. Durante la ejecución de un programa, el valor de dicha constante no puede ser alterado.

Desarrollo Didáctico

Se trabajan inicialmente los preconceptos de los estudiantes acerca de valores que no son modificables y que forman parte del escenario de sismología. Se pretende orientar el diálogo constructivo con los estudiantes hacia los valores propios de la escala de Richter y de Mercalli. En la parte motivacional, se prepara un cuestionario con un banco de preguntas relacionadas con la escala de Richter y Mercalli, este cuestionario se utiliza dentro de la metáfora de juego de mesa de toboganes y escaleras ha ser realizado por grupos. En el tablero y con avances a través de un dado, los jugadores avanzan por el juego donde por cada Celsa se realiza una pregunta del cuestionario. Al responderla satisfactoriamente, se permite al estudiante avanzar, de lo contrario se retrocede a la posición anterior al lanzamiento.

Una vez terminado el juego, el profesor entabla nuevamente el dialogo frente a la definición del concepto de constante dado por el objeto a enseñar. De esta forma y en forma colaborativa se propone un nuevo **objeto de enseñanza: Constante** que puede ser de la siguiente forma: Una constante es un espacio de memoria de las computadoras que almacena un valor inmodificable y corresponde a un identificador y tipo de dato, su sintaxis dependerá del lenguaje de programación

a usar. Durante la ejecución de un programa, el valor de dicha constante no puede ser alterado y solamente se inicializa su valor con la primera instrucción de asignación de valor. Cabe resaltar que el objeto de enseñanza se pretende que sea resultado de la construcción colectiva junto con los estudiantes, pero aquí se esboza aquella definición que es conveniente trabajarla al interior del aula.

A partir de este momento, se procede a desarrollar el conjunto de ejercicios referentes al manejo de constantes usando las computadoras y el lenguaje de programación específico.

A manera de valoración del aprendizaje, se crea conjuntamente un **objeto de evaluación: Constante,** el cual contempla elementos que permitan valorar el aprendizaje de los estudiantes, este objeto de valoración se compone de un desarrollo de ejercicios individuales, un trabajo en colectivo (puede ser grupal o en parejas) unas rúbricas para autoevaluación y evaluación intergrupal. Finalizando esta actividad, el profesor elabora un informe de clase para ser analizado nuevamente por la noosfera a fin de retroalimentar el **objeto a enseñar**.

Diseño y desarrollo didáctico para la estrategia de enseñanza del objeto de saber: Expresión Aritmético-Lógica

Diseño Didáctico

Se parte de la definición dada en la página 91 donde el **objeto de saber: Expresión Aritmético-Lógica** se define 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.

De la misma forma como los otros objetos, este objeto será sometido a las apreciaciones dadas en la noosfera. En el marco del ejercicio presentado en este anexo, las expresiones aritmético-lógicas que se usan deben estar relacionadas con la temática propuesta (por lo general se puede trabajar con las expresiones asociadas a a las ecuaciones de movimiento y cálculo de esfuerzos).

Al tenor del proyecto de curso, se plantea que la construcción del programa para dispositivos móviles sea capaz de interpretar las lecturas dadas por el acelerómetro haciendo uso de las ecuaciones relacionadas al campo de la sismología (por ejemplo, la función de Rayleigh y las funciones de desplazamiento de campo lejano).

Con estos elementos, es posible plantear la creación de un nuevo objeto a partir de la participación de la noosfera. Este nuevo **objeto a enseñar: Expresión aritmético-lógica** puede tener la siguiente definición: un conjunto de valores, variables, constantes y operadores que permiten realizar un cálculo aritmético-lógico. Dichas expresiones ayudan a representar la realidad a través de modelos matemáticos. Las expresiones aritmético-lógicas tienen operadores con una jerarquía y su escritura se define según la sintaxis de un lenguaje específico.

Es importante enfatizar en la importancia de representar la realidad a través de modelos matemáticos y que justamente las expresiones aritmético-lógicas ayudan a tal menester.

Desarrollo Didáctico

Al iniciar con las situaciones motivacionales, se parten de los preconceptos que dispongan los estudiantes en cuanto a conocimiento de física mecánica. En esta parte es conveniente el uso de simuladores de software, donde los estudiantes podrán interactuar alterando los parámetros de las situaciones asociadas a física mecánica. Existe actualmente una serie de objetos virtuales de aprendizaje que sirven para dichas actividades de enseñanza y aprendizaje. De esta manera, las actividades motivacionales parten del uso de simuladores.

A través de su uso, diferentes ejercicios pueden desarrollarse en clase para afianzar los conceptos de expresiones aritmético-lógicas. En esta ruta, el profesor en diálogo con los estudiantes pone a consideración un nuevo **objeto de enseñanza: Expresión aritmético-lógica**. La construcción de tal objeto contempla su definición como un conjunto de valores, variables, constantes y operadores que permiten realizar un cálculo aritmético-lógico. Dichas expresiones ayudan a representar la realidad a través de modelos matemáticos. Las expresiones aritmético-lógicas tienen operadores con una jerarquía y su escritura se define según la sintaxis de un lenguaje específico usándolas como asignación de valores procesables.

A partir de este momento, los estudiantes inician su labor de uso de expresiones aritmético-lógica que se requieran para el proyecto de curso. En esta parte se trata de colocar las expresiones necesarias para representar el movimiento de un objeto en un escenario gráfico a partir de los valores que se pueden leer del acelerómetro de los dispositivos móviles.

A manera de valoración del aprendizaje, se crea conjuntamente un **objeto de evaluación: Expresión aritmético-lógica**, el cual contempla elementos que permitan valorar el aprendizaje de los estudiantes, este objeto de valoración se compone de un desarrollo de ejercicios individuales, un trabajo en colectivo (puede ser grupal o en parejas) unas rúbricas para autoevaluación y evaluación intergrupal. Finalizando esta actividad, el profesor elabora un informe de clase para ser analizado nuevamente por la noosfera a fin de retroalimentar el **objeto a enseñar**.

Diseño y desarrollo didáctico para la estrategia de enseñanza del objeto de saber: Condicional

Diseño Didáctico

Para realizar el diseño didáctico del **objeto de saber: Condicional** se parte de la definición propuesta en la página 92 la cual asume el concepto 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 través de la noosfera, los integrantes inician la construcción del objeto a enseñar: Condicional en respuesta a una necesidad del contexto. Continuando con la situación usada en los previos diseños y desarrollos didácticos, a manera de ejemplo se tiene la situación de riesgo volcánico como centro de interés para impartir los conceptos dentro del aula. De esta forma, los condicionales pueden ser utilizados para clasificar el valor de un movimiento sísmico registrado dentro de una escala constante. Así, el **objeto a enseñar: Condicional** se puede transponer según la siguiente definición y dentro de la lógica de estos ejemplos: 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). Los condicionales contribuyen a tomar decisiones basadas en comparaciones Booleanas de valores con respecto a sus referentes programados.

Para los asuntos prácticos relacionados con estos ejemplos, el riesgo volcánico aparece en escena en cuanto a determinar la intensidad de un movimiento sísmico, por lo tanto, las dinámicas de clase en cuanto al desarrollo de la didáctica deben apuntar hacia situaciones relacionadas a la construcción del programa para dispositivos móviles que involucre la lectura de valores dados por su acelerómetro y su correspondiente clasificación en escala de Richter y de Mercalli.

Desarrollo Didáctico

Partir de los preconceptos es la actividad primera en ser realizada como trabajo de aula, según las directrices de la teoría didáctica desarrollada en este trabajo. De esta forma, los preconceptos asociados a los condicionales son de gran importancia para la construcción del **objeto de enseñanza: Condicional**. Los eventos motivacionales están orientados a resolver laberintos soportados en toma de decisiones de bifurcación. Este juego de mesa puede ser aplicado a grupos pequeños de estudiantes de tal suerte que cada estudiante contribuye con una decisión de direccionamiento a la vez (arriba, abajo, izquierda y derecha). Por turnos, los estudiantes toman decisiones en cuanto a la ruta que se debe seguir para resolver un laberinto; con cada bifurcación, a manera de instrucción deberán responder a una serie de preguntas programadas dentro de un cuestionario planteado por el profesor (dicho cuestionario siempre deberá estar relacionado con la temática de fundamentos de programación) al responder acertadamente, cada estudiante podrá realizar su operación de movimiento para avanzar en la resolución de dicho laberinto.

Al terminar el juego, su abre la discusión de construcción colectiva entre el profesor y los estudiantes orientados hacia una posible definición de dicho objeto: 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). Los condicionales contribuyen a tomar decisiones basadas en comparaciones Booleanas de valores con respecto a valores que determinan el comportamiento de

un programa de computadoras.

Después de construir una definición cercana, los estudiantes realizan sus ejercicios en las computadoras y van alimentando su proyecto de programación con las condiciones apropiadas para la clasificación de los valores sísmicos registrados dentro de la escala de Richter y la escala de Mercalli.

Finalmente, A manera de valoración del aprendizaje, se crea conjuntamente un **objeto de evaluación: Condicional**, el cual contempla elementos que permitan valorar el aprendizaje de los estudiantes, este objeto de valoración se compone de un desarrollo de ejercicios individuales, un trabajo en colectivo (puede ser grupal o en parejas) unas rúbricas para autoevaluación y evaluación intergrupal. Finalizando esta actividad, el profesor elabora un informe de clase para ser analizado nuevamente por la noosfera a fin de retroalimentar el **objeto a enseñar**.

Diseño y desarrollo didáctico para la estrategia de enseñanza del objeto de saber: Ciclo

Diseño Didáctico

La definición del concepto ayuda a establecer el punto de partida del diseño didáctico. Para tal efecto, el **objeto de saber: Ciclo** toma la definición que se encuentra plasmada en la página 93. 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.

Tomado dicho objeto, el primer segundo nivel de transposición aparece por interacción de la noosfera que busca contextualizar el concepto. En la lógica de los ejemplos recreados en esta serie de diseños y desarrollos didácticos, el riesgo volcánico nuevamente aparece en escena con el fin de orientar el ejercicio de la enseñanza hacia el enriquecimiento del proyecto de curso, en especial de su módulo inicial del programa para dispositivos móviles de detección sísmica a través del acelerómetro.

La interacción del objeto de saber en la noosfera obliga a un segundo nivel de transposición de dicho objeto. En la articulación de las necesidades del contexto y teniendo en cuenta este caso específico, un **objeto a enseñar: Ciclo** aparece en escena como una estructura algorítmica que permite la iteración controlada (repetición) de instrucciones del programa a través de una variable de control en una lógica de evaluación de un condicional.

Para efectos prácticos, se plantea el diseño didáctico orientado al módulo de programa para detección de sismos dentro de una estructura repetitiva y constante controlada por el usuario.

Desarrollo Didáctico

Los preconceptos que tienen los estudiantes acerca de los ciclos son de vital importancia al

momento de entrar en materia con dicho concepto. Quizás los estudiantes tengan una apreciación muy cercana al concepto de ciclo que se maneja dentro de los fundamentos de programación. Basados en sus preconceptos, el profesor alimenta la idea de realizar actividades que son repetitivas. Una propuesta puede ser el dibujo de fractales como aquellas figuras que se repiten constantemente y que van generando figuras dentro del campo del arte. Como estrategia motivacional, el profesor puede proponer el desarrollo de dibujos fractales para manejar el concepto de repetición controlada (el triángulo Sierpinski y el conjunto de Mandelbrot son ejemplos claros de cómo hacerlo).

Al terminal la sesión de motivación basada en dibujo artístico, el profesor junto con sus estudiantes entabla el diálogo a fin de construir un concepto de ciclo de los fundamentos de programación de computadoras. A partir de esto, se pretende acercarse hacia la definición dada por el **objeto de enseñanza: Ciclo** como una estructura repetitiva que permite la iteración controlada de las instrucciones del programa a través de variables de control en una lógica de evaluación de condiciones formuladas por expresiones booleanas.

Al terminar la fase de construcción colectiva del concepto, los estudiantes realizan varios ejercicios propuestos por el profesor para ser implementados en programas de computadoras. Al finalizar dichos ejercicios, los estudiantes implementan la lógica cíclica al programa que forma parte del proyecto de curso. Esto es, la constante lectura de movimientos sísmicos y que se encuentra controlado por un ciclo dentro del programa para dispositivos móviles.

Para la valoración del aprendizaje, se crea conjuntamente un **objeto de evaluación: Ciclo**, el cual contempla elementos que permitan valorar el aprendizaje de los estudiantes, este objeto de valoración se compone de un desarrollo de ejercicios individuales, un trabajo en colectivo (puede ser grupal o en parejas) unas rúbricas para autoevaluación y evaluación intergrupal. Finalizando esta actividad, el profesor elabora un informe de clase para ser analizado nuevamente por la noosfera a fin de retroalimentar el **objeto a enseñar**.

Diseño y desarrollo didáctico para la estrategia de enseñanza del objeto de saber: Función
Diseño Didáctico
En la página 94 se encuentra plasmada la definición del **objeto de saber: Función**, la cual asume 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. La noosfera hace su parte para transponer dicho objeto teniendo en cuenta el contexto trabajado en estos diseños y desarrollos didácticos. Siendo manifiesto el ejemplo del riesgo volcánico, las funciones a las que se hace referencia son basadas en el cálculo de desplazamiento de campo

lejano (con sus parámetros de funciones trigonométricas) y el cálculo de la velocidad de ondas según Rayleigh.

De esta forma, se puede proponer un **objeto a enseñar: Función** por parte de la noosfera el cual puede ser considerado 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. Para efectos de simulación, las funciones pueden representar cuerpos de transformación de valores dados por modelos matemáticos claramente establecidos.

Desarrollo Didáctico

Semejante al desarrollo didáctico del concepto de expresión aritmético-lógica de los fundamentos de programación de computadoras, los preconceptos de los estudiantes como punto de partida del abordaje del concepto de función pueden partir del símil de la matemática. En este punto se retoma el concepto de función en matemáticas y en física, más aún cuando se da una orientación en las funciones y ecuaciones propias de la sismología. Como estrategia motivacional, se necesitará el uso de simuladores de física y de modelos matemáticas apoyados en objetos virtuales de aprendizajes propios para tal dominio de conocimiento.

Son ejercicios puntuales de la vida real en asuntos de sismología con casos particulares registrados en el planeta los elementos que puede tener el profesor para despertar un interés en sus estudiantes.

Después de realizar las actividades en los simuladores, en diálogo constructivo se propone acercarse a una posible definición del **objeto de enseñanza: Función** como resultado de un tercer nivel de transposición. Así su definición puede ser una parte del programa que realiza tareas específicas y tiene la facilidad de ser llamado en diferentes partes dentro del programa; las funciones reciben datos de entrada y arrojan un resultado. Para efectos de simulación, las funciones pueden representar transformación de valores dados por modelos matemáticos claramente establecidos.

Muchos ejercicios se deben realizar para afianzar el concepto en la práctica. Finalizados los ejercicios en clase, los estudiantes deberán integrar los modelos de sismología al programa que forma parte de su proyecto de curso a través del uso de funciones.

Al final, la valoración del aprendizaje crea conjuntamente un **objeto de evaluación: Función**, el cual contempla elementos que permitan valorar el aprendizaje de los estudiantes, este objeto de valoración se compone de un desarrollo de ejercicios individuales, un trabajo en colectivo (puede ser grupal o en parejas) unas rúbricas para autoevaluación y evaluación intergrupal. Finalizando esta actividad, el profesor elabora un informe de clase para ser analizado nuevamente por la noosfera a fin de retroalimentar el **objeto a enseñar**.