

**“APOYAR EL PROCESO DE ENSEÑANZA-APRENDIZAJE DEL CURSO
DISEÑO DE COMPILADORES DEL PROGRAMA DE INGENIERÍA DE
SISTEMAS DE LA UNIVERSIDAD DE NARIÑO PARA FACILITAR EL
APRENDIZAJE DEL PROCESAMIENTO DESCENDENTE EN LA
CONSTRUCCIÓN DE GRAMÁTICAS LL(1)”**

**JUAN ESTEBAN CASTRO GUERRERO
PETER D'LOISE CHICAIZA CORTEZ**

**UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2020**

**“APOYAR EL PROCESO DE ENSEÑANZA-APRENDIZAJE DEL CURSO
DISEÑO DE COMPILADORES DEL PROGRAMA DE INGENIERÍA DE
SISTEMAS DE LA UNIVERSIDAD DE NARIÑO PARA FACILITAR EL
APRENDIZAJE DEL PROCESAMIENTO DESCENDENTE EN LA
CONSTRUCCIÓN DE GRAMÁTICAS LL(1)”**

**JUAN ESTEBAN CASTRO GUERRERO
PETER D'LOISE CHICAIZA CORTEZ**

**Trabajo de grado presentado como requisito parcial para optar el título de
Ingeniero de Sistemas**

**Director:
MAGÍSTER NELSON JARAMILLO ENRÍQUEZ**

**UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2020**

ABSTRACT

The present research was developed considering the existent problem involving the assimilation of theoretical concepts and the application of these by former students of the subject 'Compiladores', part of the systems engineering program at the University of Nariño. It was found, through a survey carried out on the Compilers course of the University of Nariño from period B of 2019, together with different antecedents that seek to solve the same problem within subjects that cover common themes in the compiler area, that the use of software tools is the solution to to guide the student and support him in his learning process. However, in the local context, it was found that these tools lack functional components that can allow a transition from the theoretical to the practical, more specifically in the final objective that most of the courses that teach this subject propose, the development of a compiler.

Based on the above, this project outlined as main objective to support the teaching-learning process of the compilers course at the University of Nariño, more specifically on the topic of Descending Processing in the construction of LL Grammars (1), with Practice-oriented components that facilitate the assimilation of the theoretical - practical component of the subject.

RESUMEN

La presente investigación se desarrolló teniendo en cuenta la problemática existente dentro de la asimilación de los conceptos teóricos y la aplicación de estos por parte de los estudiantes que cursan la materia de Compiladores del programa de Ingeniería de Sistemas de la Universidad de Nariño. Se encontró, por medio de un sondeo realizado al curso de compiladores de la Universidad de Nariño del periodo B de 2019, en conjunto con distintos antecedentes que buscan dar solución al mismo problema dentro de asignaturas que abarcan temas en común del área de compiladores, que el uso de herramientas software es la solución para orientar al estudiante y apoyarlo en su proceso de estudio. Sin embargo, en el contexto local, se encontró que dichas herramientas carecen de componentes funcionales que permitan hacer la transición de lo teórico a lo práctico, más concretamente en el objetivo final que la mayoría de los cursos que imparten esta asignatura proponen, el desarrollo de un compilador.

En base a lo anterior, el proyecto se planteó como objetivo general el apoyar el proceso de enseñanza-aprendizaje del curso de compiladores de la Universidad de Nariño, más concretamente en la temática de Procesamiento descendente en la construcción de Gramáticas LL(1), con componentes orientados a la práctica que faciliten la asimilación del componente teórico - práctico de la materia.

CONTENIDO

	pág.
INTRODUCCIÓN	7
1. PROBLEMA DE INVESTIGACIÓN	9
1.1. TEMA DE INVESTIGACIÓN	9
1.2. ÁREA DE INVESTIGACIÓN	9
1.3. LÍNEA DE INVESTIGACIÓN	9
1.4. DESCRIPCIÓN DEL PROBLEMA	9
1.5. FORMULACIÓN DEL PROBLEMA	11
1.6. OBJETIVOS	11
1.6.1. Objetivo general	11
1.6.2. Objetivos Específicos	11
1.7. JUSTIFICACIÓN	12
1.8. DELIMITACIÓN	13
2. MARCO TEÓRICO	15
2.1. ANTECEDENTES	15
2.2. SUPUESTOS TEÓRICOS	17
2.3. DEFINICIÓN DE CONCEPTOS	18
2.3.1. Aprendizaje Basado en conceptos	18
2.3.2. Compilador	20
2.3.3. Análisis Léxico	24
2.3.4. Análisis Sintáctico	28
2.3.5. Procesamiento Descendente	35
2.3.6. Metacompiladores	39
2.4. FORMULACIÓN DE HIPÓTESIS	41
3. METODOLOGÍA	42
3.1. TIPO DE INVESTIGACIÓN	42
3.2. DISEÑO DE LA INVESTIGACIÓN	42
3.3. PROPÓSITO y CONTEXTO	43
3.4. PREGUNTAS DE INVESTIGACIÓN	45

3.5. MÉTODOS: RELACIÓN ENTRE ESTUDIO Y SUS PROTAGONISTAS (POBLACIÓN Y MUESTRA)	46
3.6. MÉTODOS: INSTRUMENTOS DE RECOLECCIÓN DE INFORMACIÓN	48
Observación	48
Revisión documental	48
Encuesta	49
3.7. MÉTODOS: PROCESAMIENTO DE INFORMACIÓN	57
3.8. MÉTODOS: VALIDEZ Y CONFIABILIDAD	60
4. RESULTADOS DE LA INVESTIGACIÓN	63
4.1. Herramienta de apoyo	63
4.1.1. Ítems contemplados para desarrollo	63
4.1.2. Proceso de desarrollo	64
4.1.3. Componentes del software	66
4.2. Resultados sintetizados de evaluación	75
4.3. Propuesta de formulario para cursos siguientes	78
5. ANÁLISIS Y DISCUSIÓN DE LOS RESULTADOS	79
6. CONCLUSIONES	83
7. RECOMENDACIONES	84
BIBLIOGRAFÍA	85

INTRODUCCIÓN

El desarrollo de aplicativos como apoyo a la didáctica del aprendizaje de diferentes áreas del conocimiento, con el avance tan acelerado de la tecnología computacional, adquieren gran importancia, y su correcta implementación puede apoyar y mejorar el proceso de aprendizaje del estudiante.

En este sentido los estudiantes de últimos semestres del programa de Ingeniería de Sistemas, han dirigido su atención a asignaturas, que dentro del ámbito de la ciencia de la computación tienen gran relevancia en la formación profesional; Ingeniería de Software, Diseño de Base de Datos, Programaciones, son entre otras, las áreas del conocimiento que el futuro ingeniero debe desarrollar y manejar. Pero en concreto para el presente estudio, se dirigió la atención hacia las materias que siguen la línea de estudio de los compiladores, tomando como base, la asignatura Diseño de Compiladores, de los últimos semestres de la carrera de Ingeniería de Sistemas de la Universidad de Nariño.

El objetivo principal de este curso es la adquisición de conocimientos que permitan la implementación de un compilador como trabajo final, tarea que de acuerdo a antecedentes y a un estudio realizado dentro de la facultad, termina siendo pesada y difícil de llevar cabo, tanto para los estudiantes como para los profesores, aspecto que se evidencia en el trabajo final presentado.

Como respuesta a lo anterior, se han propuesto varias soluciones, que pretenden hacer uso de medios tecnológicos para facilitar la construcción del compilador o la apropiación de teoría que debe ser aplicada en el desarrollo de este último; sin embargo, es evidente al analizar dichas soluciones, que el componente didáctico se encuentra ausente total o parcialmente, por lo que la disminución de la dificultad del estudiante, en el desarrollo del proyecto de esa talla, no es manifiesta como se desea.

La investigación de esta problemática se realizó debido al interés académico que supone el implementar herramientas informáticas que ayuden a conseguir objetivos e impartir conocimientos dentro de asignaturas que poseen gran carga académica; conocer la viabilidad que la creación de este tipo de soluciones posee, abre el camino para futuras implementaciones dentro de otras áreas del entorno académico. Por otra parte, cabe aclarar que, desde el ámbito profesional como ingenieros de sistema, una materia como la de compiladores crea un valor mayúsculo dentro de nuestra formación, pues los conceptos y temáticas tratadas dentro de ella abren camino hacia muchos proyectos relacionados con la ciencia de la computación y cualidades analíticas idóneos en los profesionales a formar.

Este proyecto se realizó teniendo en cuenta tres fases. En un primer momento, se dio la fase de exploración para la apropiación de temáticas, teoría y datos necesarios para la construcción de una solución informática. Seguido a esto se definió una fase de desarrollo, en la que se utilizó metodología scrum para el diseño y creación de la herramienta de prueba; por último, está la fase de validación en la que se presentó la herramienta al grupo de estudiantes selecto, en conjunto con una posterior prueba de su desempeño y el análisis respectivo de su desempeño.

1. PROBLEMA DE INVESTIGACIÓN

1.1. TEMA DE INVESTIGACIÓN

“APOYAR EL PROCESO DE ENSEÑANZA-APRENDIZAJE DEL CURSO DISEÑO DE COMPILADORES DEL PROGRAMA DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD DE NARIÑO PARA FACILITAR EL APRENDIZAJE DEL PROCESAMIENTO DESCENDENTE EN LA CONSTRUCCIÓN DE GRAMÁTICAS LL(1)”

1.2. ÁREA DE INVESTIGACIÓN

Lenguajes formales, Autómatas finitos y compiladores

1.3. LÍNEA DE INVESTIGACIÓN

Procesos educativos apoyados por las nuevas tecnologías de la información y la comunicación

1.4. DESCRIPCIÓN DEL PROBLEMA

En el desarrollo del curso de compiladores dictado en la universidad de Nariño , como una cátedra dentro del pensum del programa de ingeniería de sistemas, según un sondeo realizado al curso de compiladores de la Universidad de Nariño del periodo B de 2019, existe un problema visible tanto en el rendimiento académico como en la perspectiva de los estudiantes que han cursado esta asignatura, fundamentado en la dificultad que existe para asimilar los conocimientos teóricos. El sondeo reveló una falencia en el enfoque práctico de la asignatura, situación que no es ajena tanto para el campo académico nacional como internacional, siendo percibida en otros estudios como el realizado en la Universidad del Istmo, México, donde Arellano y Nieva describen que las dificultades para asimilar los conocimientos teóricos de la materia han generado que los estudiantes resuelvan el proyecto final de la materia (la creación de un semi-compilador) “como pueden”, fenómeno que evidencia una falencia en el proceso de aprendizaje dentro de la asignatura¹.

¹ ARELLANO Jesús, NIEVA Omar, ALGREDO Ignacio. *Programación Matemática y Software: Aprendizaje Basado en Proyectos Utilizando L-Systems en un Curso de Compiladores*. Oaxaca. Universidad del Istmo. 2013. Vol. 5.

Al igual que en el estudio mencionado y otros varios que denotan esta problemática, se recurre al uso de un aplicativo o herramienta software que permita fortalecer el enfoque práctico dentro del curso de compiladores, como una estrategia que garantiza la aplicación del conocimiento teórico dentro de un espacio real.

Estas herramientas, que existen en el mercado, son desarrolladas por las entidades que realizaron estos estudios, y presentan características y funcionalidades varias, entre las cuales se procura la transmisión y uso práctico de estructuras vistas en la temática de compiladores, en algunos casos de maneras más visuales y dinámicas que otras.

En el entorno local, y de acuerdo con el sondeo realizado al grupo de la asignatura de Diseño de Compiladores del periodo B del año 2019 en la Universidad de Nariño, se determinó que el problema se dimensiona en gran medida en la presentación del proyecto final del curso, que consiste en el desarrollo de un semi-compilador, aspecto que evidencia una falencia en la asignatura, el paso de lo teórico a lo práctico en este aspecto en particular.

Por otro lado, el análisis de las herramientas didácticas existentes, más apropiadas, muestran que carecen de una funcionalidad que podría ser clave en el aprendizaje de la materia: una forma de permitir a los estudiantes visualizar la manera en que las estructuras teóricas (máquinas de estado, pilas, gramáticas, etc.) se traducen a código, dentro de un entorno didáctico donde el conocimiento teórico sea totalmente interactivo.

Por lo anterior, la presente investigación estudió el uso de una herramienta que permite el paso de lo teórico a lo práctico, facilitando la implementación de gramáticas LL(1) en código Python, con el fin de disminuir la carga académica de los estudiantes en el curso de compiladores y a partir de esto fomentar la implementación de espacios para la codificación de elementos teóricos del curso de compiladores, fortaleciendo los conocimientos teóricos a través de espacios prácticos.

A falta de la existencia de una herramienta con dichas características, se propuso el desarrollo de un software con los atributos requeridos, como parte de la investigación.

1.5. FORMULACIÓN DEL PROBLEMA

¿El desarrollo de un aplicativo didáctico que apoya el aprendizaje de la temática procesamiento descendente en la construcción de gramáticas LL(1), permite mejorar la asimilación de los conceptos y su aplicación en la construcción de un compilador?

1.6. OBJETIVOS

1.6.1. Objetivo general

Evaluar el proceso de asimilación del conocimiento teórico – práctico de la materia Diseño de compiladores en la temática de procesamiento descendente, utilizando para el componente práctico una herramienta software desarrollado para apoyar su aprendizaje.

1.6.2. Objetivos Específicos

- Delimitar los componentes visuales presentados en herramientas didácticas para la asignatura de diseño de compiladores, adecuados para dinamizar la funcionalidad de la herramienta a desarrollar.
- Desarrollar un módulo que permita la interacción con el componente del análisis léxico, para afianzar los elementos teóricos y su aplicación práctica en el desarrollo de un compilador.
- Desarrollar un módulo que implemente el componente del análisis sintáctico, que permita afianzar los elementos teóricos y su aplicación práctica en el desarrollo de un compilador, utilizando el procesamiento descendente.
- Implementar en cada uno de los módulos de análisis, un generador de código que permita acercar los modelos lógicos, a un modelo práctico realizado en un lenguaje de programación.
- Generar un prueba-encuesta aplicable a los estudiantes usuarios del software que permita evidenciar la perspectiva de estos sobre

la utilidad y usabilidad de la herramienta software dentro del diseño y la construcción de un compilador.

- Proponer un formulario de evaluación de la herramienta dirigido a los estudiantes que cursarán la cátedra Diseño de compiladores, dónde se hará uso de la herramienta software de apoyo.

1.7. JUSTIFICACIÓN

El uso de herramientas software para facilitar el proceso de aprendizaje es una de las tendencias que ha impulsado la evolución de los medios de enseñanza. Siendo pertinente la producción de herramientas interactivas que apoyen al docente en su práctica dentro de distintas disciplinas. Pero al igual que la pedagogía es evaluada y sometida a mejoras que le permitan adaptarse a los cambios de la sociedad, estas herramientas software deben también evaluarse y evolucionar para adaptarse a los nuevos requisitos de la formación académica.

Y es bajo esta perspectiva que, para la universidad de Nariño, como ente generador de talento profesional relevante para el avance de la región, que esta investigación toma importancia. Más específicamente para la facultad de ingeniería en su programa de Ingeniería de Sistemas ya que promover y mejorar la aplicación de recurso software para el apoyo del aprendizaje a través de un enfoque práctico que difiere de las herramientas existentes en el mercado, para la asignatura de Compiladores, puede significar la formación de profesionales con saberes que les permitan incursionar más profundo dentro del conocimiento de las ciencias computacionales, resultando en una generación de individuos con cualidades más llamativas y necesarias en el mercado laboral.

Hecho fundamentado por la ACM en su publicación “Computer Engineering Curricula 2016”² donde se menciona la necesidad del entendimiento de los compiladores como recursos de uso cotidiano para los profesionales en computación, y más específicamente establece en la sección Resumen de cursos- Curriculum D, “principle of compiler system” como una cátedra integrada en el BoK de las ciencias computacionales.

² ACM, IEEE. *Computer Engineering Curricula 2016*. p. 151.

Por otro lado, Dick Grune en la publicación “Modern Compiler Design”³ concibe que la importancia del diseño de compiladores en la formación de profesionales de computación se debe a su presencia y utilidad, dentro de las ramas de las ciencias computacionales en general y la opción de adquirir algoritmos o percepciones lógicas aplicables en diversos ámbitos externos a los compiladores.

1.8. DELIMITACIÓN

En el proyecto se propuso apoyar el proceso de enseñanza-aprendizaje del curso Diseño de Compiladores del programa de Ingeniería de Sistemas de la Universidad de Nariño, para facilitar el aprendizaje del procesamiento descendente en la construcción de gramáticas LL(1), para lo cual se tuvo en cuenta, propiedades y conocimientos presentes en herramientas y recursos software desarrollados y enfocados en cursos que integran el diseño de compiladores, en la revisión documental de elementos literarios disponibles de la temática ya comentada, y en el desarrollo de recursos de software que complementen elementos prácticos no identificados en la revisión de estos recursos.

El apoyo al proceso de enseñanza-aprendizaje del curso Diseño de compiladores, se centró específicamente, en facilitar la conceptualización y aplicación práctica de la temática, “Procesamiento descendente con gramáticas LL(1)”, que en el desarrollo del curso tiene como resultado la construcción de un semi-compilador, aspecto que en los recursos de software desarrollados para estos cursos, no se presentan en la medida requerida, por lo que en el transcurso de esta investigación, se desarrolló una herramienta de software con una funcionalidad centrada en la temática tratada en el curso de compiladores, la cual abarcar las siguientes funciones:

- **Analizador Léxico:**

Permite al usuario establecer el funcionamiento de un analizador léxico basado en la definición de Tokens. Como componente previo y necesario para el entendimiento y uso de gramáticas.

³ GRUNE Dick, REEUWIJK Kees, E.B.H, J.H. Jacobs, LANGENDOEN Koen. *Modern Compiler Design. Second Edition. New York: Business Media. 2012. p. 829.*

- **Analizador Sintáctico:**

Brinda al usuario un espacio restringido para la generación y validación de gramáticas LL(1) , donde se hace uso de los elementos definidos en el módulo anterior “Analizador Sintáctico”.

- **Generador de Código:**

proporciona un medio para la generación de un código funcional, representante de los elementos definidos previamente por el usuario “Analizador léxico” y “Analizador Sintáctico”. Siendo un componente fundamental en la ilustración del proceso práctico de aplicación de los conocimientos teóricos de la temática.

Estas funcionalidades se desarrollaron en el lenguaje de programación Python, y la librería PyQt dentro de un entorno de trabajo usando Visual Studio Code y SO Windows 10.

Para determinar en qué medida se logró el objetivo de facilitar el aprendizaje de esta temática, en la investigación se presentó la propuesta de software al curso concluido de Compiladores del año 2019 periodo B.Y a continuación se expuso un formulario en el que se evaluó la utilidad, rendimiento y usabilidad de la herramienta en cuanto construcción del compilador y la asimilación de sus elementos teóricos.

2. MARCO TEÓRICO

2.1. ANTECEDENTES

“DESARROLLO DE UN SOFTWARE EDUCATIVO QUE SIRVA DE APOYO PARA EL APRENDIZAJE DE LA ASIGNATURA COMPILADORES DEL PROGRAMA DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD DE SAN BUENAVENTURA CARTAGENA”

Estableciendo la trayectoria y evolución de los lenguajes de programación desde los inicios de la computación el autor hace énfasis en la persistencia de los compiladores como herramienta de uso cotidiano para los profesionales de la computación, siendo un hecho importante la comprensión completa de la teoría y lógica detrás de estos elementos. a través de una investigación que fundamenta una necesidad por mejorar el enfoque práctico con que se imparte el curso de compiladores a nivel de Colombia y más específicamente en la universidad de Buenaventura Cartagena. Para lo cual propone el desarrollo de un aplicativo (COMPIISOFT) que permita la impartición de conocimiento teórico y práctico de una manera más didáctica. El autor concluye con el resultado positivo dado por la perspectiva de la comunidad estudiantil con respecto a la herramienta desarrollada, además del proceso de integración más ameno por parte de estudiantes y docentes durante la impartición del curso. (*Johnny Cabarcas Machacon, 2013*)

“HERRAMIENTA DIDÁCTICA PARA ANÁLISIS SEMÁNTICO Y TRADUCCIÓN DE LENGUAJES FORMALES”

Este trabajo realizado en la serie de innovación docente los autores entienden la necesidad de innovar en el proceso de enseñanza aprendizaje para la temática de compiladores dentro de la materia "procesadores de lenguaje", haciendo uso de las herramientas tecnológicas, desarrollan un producto software que se implementa en un servidor web permitiendo como funcionalidad principal observar el funcionamiento de gramáticas LL(1), SLR(1), LR(1) y LALR(1), además de la construcción interactiva de dichas estructuras. Adicionalmente el aplicativo permite hacer seguimiento de la actividad del estudiante por parte del docente. Como resultado los autores enfatizan el aumento en el rendimiento académico de los estudiantes y la participación de estos en el curso en general. (*Jesús García Herrero, Antonio Berlanga, 2006*)

“METACOMPILADOR DIDÁCTICO JAVA”

En este trabajo se detalla la construcción de un metacompilador que sigue la categoría de gramáticas tipo LL(1), como herramienta didáctica para la materia de lenguajes y autómatas II. Dentro del desarrollo de este, se detalla la creación de dos componentes importantes: el diseño de un lenguaje de programación para el metacompilador y el desarrollo de un entorno de desarrollo integrado (IDE).

El hecho de contar no solo con un lenguaje de programación propio si no con un IDE para el uso del metacompilador, hace que este último se vuelva una herramienta didáctica sumamente eficiente para ser usada en el entorno educativo, puesto que provee al estudiante de un entorno que permite apropiarse de las temáticas vistas en una materia que sigue las líneas de estudio de los compiladores.

(Erick Leonel Rico Preciado, Ana Cristina Bueno Campos, José Gerardo Carpio Flores, Ruth Sáez de Nanclares Rodríguez, Martha Alicia Rocha Sánchez)

“ARTÍCULO: AN EDUCATIONAL TOOL FOR TEACHING COMPILER CONSTRUCTION”

El proyecto estudia el desarrollo de la herramienta didáctica LISA (Sistema de implementación de lenguaje basado en gramáticas atributadas, por su traducción al español) en conjunto al uso e impacto que tuvo está en el proceso educativo de la construcción de compiladores. El aplicativo permite experimentar y probar varios analizadores léxicos y sintácticos, junto a estrategias para la evaluación de atributos. LISA permite a los usuarios ver el proceso de compilación y obtener una comprensión intuitiva de la ejecución del compilador. Para cada una de las fases (léxica, sintaxis y semántica) existe una animación apropiada para mejorar la apropiación de conocimiento por parte del espectador.

Los autores afirman que sin el uso de una herramienta como lo es LISA, los temas discutidos en la línea de estudios de los compiladores se vuelven mucho más difíciles de entender y tratar. La inclusión de este tipo de software hace que la experiencia de enseñar y aprender en estos cursos, sea más intuitiva y entretenida. *(Marjan Mernik)*.

2.2. SUPUESTOS TEÓRICOS

- (Kundra y Sureka, 2016)⁴ analizan cómo los aspectos de la educación tradicional donde se presenta un fuerte factor de lectura y uso de documentación en el proceso de aprendizaje de las cátedras relacionadas a la construcción de compiladores, pueden ser causantes de la problemática planteada, por lo que es necesario adoptar otra metodología más centrada en las necesidades del estudiante. En este aspecto se propone el uso de aprendizaje basado en el caso como complementario al proceso ABP utilizado en muchos cursos de compiladores.
- (Mernik, 2013)⁵ dice que existe una brecha entre el espacio teórico y el práctico dentro de los cursos de compiladores. Principalmente porque las herramientas utilizadas para el apartado práctico tratan de manera diferente los conceptos, utilizan algoritmos complejos y no permiten el análisis de los mismos. Bajo esta perspectiva se plantea el supuesto de que las herramientas de ámbito profesional no son adecuadas para abordar el espacio práctico de las asignaturas de compiladores.
- En la publicación de (Vegdahl, 2001)⁶ se establece que el uso de herramientas que permitan un acercamiento más visual a las estructuras conceptuales dentro del espacio de los compiladores, puede significar un aporte positivo dentro del proceso de aprendizaje. Supuesto que se propone abordar de acuerdo a (Arias y Santiana, 2018), usando como base principios utilizados en otras herramientas, pero permitiendo la interacción dinámica del usuario con las mismas.

La visualización de código ejecutable de estructuras lógicas dentro del proceso de aprendizaje permite a los estudiantes afianzar su conocimiento sobre las mismas sin la necesidad de que estos deban profundizar en librerías terceras que aumentan la complejidad del proceso de construcción de un compilador.

⁴ KUNDRA Divya, SUREKA Ashish. *Arxiv: Application of Case-Based Teaching and Learning in Compiler Design Course. India. Cornell University. 2016. vol. 1.*

⁵ MERNIK Marjan, ZUMER Viljem. *IEEE Transactions On Education: An Educational Tool For Teaching Compiler Construction. IEEE. 2003. vol. 46.*

⁶ VEGDAHL Steven R. *Journal of computings science: Using Visualization Tools To Teach Compiler Design. ACM. 2001. vol. 16.*

2.3. DEFINICIÓN DE CONCEPTOS

2.3.1. Aprendizaje Basado en conceptos

Definición:

El aprendizaje basado en proyectos o ABP es una estrategia o metodología propuesta bajo la filosofía de que el alumno debe ser autónomo y responsable dentro de su proceso de aprendizaje, como un elemento activo del mismo, donde el trabajo en equipo juega un papel fundamental para su desarrollo⁷.

Dentro de esta metodología se busca que a través de una serie de tareas el estudiante pueda proponer una solución a una problemática, sea esta o no de un espacio real, lo que contempla que el estudiante organiza y planifica su trabajo, adquiere e investiga conocimientos, establece objetivos y propone procesos para su cumplimiento, finalizando con la discusión y conclusión de su aprendizaje, mientras el rol del docente se limita a la orientación del proceso y la facilitación recursos⁸.

Además, como elemento clave de este proceso, se da la consecución de un producto resultado del esfuerzo cooperativo de los estudiantes.

Atributos del aprendizaje basado en proyectos

Como parte de la fuente constructivista de la pedagogía, el ABP se caracteriza por unos atributos:

- **Estudiante activo:**
En esta metodología el estudiante es el motor principal del proceso de aprendizaje, haciéndolo consciente de su responsabilidad hacia sí mismo.
- **Inclusividad:**
Al desarrollarse en equipo, esta estrategia fomenta la constitución de equipos diversos en cuanto capacidades y aptitudes que se complementan o magnifican.

⁷ ARELLANO Jesús , NIEVA Omar, ALGREDO Ignacio. *Programación Matemática y Software: Aprendizaje Basado en Proyectos Utilizando L-Systems en un Curso de Compiladores*. Oaxaca. Universidad del Istmo. 2013 . Vol. 5.

⁸ CEU. *KIT DE PEDAGOGÍA Y TIC. Gobierno de las Canarias*. 2017

- **Socialización:**

Acorde a la agrupación de personas, genera espacios de participación y socialización en el que la comunicación y respeto son relevantes en cuanto al desempeño del grupo.

- **Flexibilidad**

Puede modificarse en conveniencia del proceso de aprendizaje o las necesidades de los estudiantes involucrados.

- **Interdisciplinariedad:**

Muchos proyectos requieren de la integración de distintos conocimientos provenientes de diversas disciplinas, de manera que puede ser necesario la composición de grupos con diferente base técnica.

Aplicación en ingeniería de sistemas:

Si bien el ABP empieza como una metodología ampliamente utilizada dentro de carreras sociales o relacionadas a la medicina, su orientación hacia la formación de estudiantes innovadores, autónomos y creativos, ha hecho que se aplique dentro de otras áreas, que recientemente han criticado su carácter rígido en cuanto a su métodos y evaluación, por ejemplo, las ingenierías, consecuentemente esto les ha conllevado la formación de estudiantes con un rigor más flexible⁹.

Observe el claro ejemplo de la universidad de Nariño, en su facultad de ingeniería, programa de ingeniería de sistemas donde varias cátedras como seminarios de programación, ingeniería de software, inteligencia artificial, diseño de compiladores, ingeniería legal y ética, entre otras, han adoptado en gran parte esta metodología con el fin de ubicar a sus estudiantes en espacios más allegados a la realidad, en los que se fomenta el trabajo en equipo, el autodidactismo y la autogestión. Y sobre todo la socialización, como parte integral de su función como institución, para la formación de profesionales no solo con fortaleza en lo técnico sino también en lo integral.

⁹ MIRÓ Carme, BARALDÉS Marissa, BENITO Helena ; GUTIÉRREZ. *Aula de innovación educativa: El ABP - origen, modelos y técnicas afines*. España: Universidad de Girona. 2012. vol. 216.

Si se habla sobre diseño de compiladores, cátedra a la que alude esta investigación, el desarrollo de un proyecto final para la construcción de un semi compilador, práctica que se efectúa no solo en la UDENAR sino también en otras entidades a nivel nacional e internacional, esta conlleva gran parte no solo en la cuestión evaluativa, sino también formativa de la asignatura, debido al nivel de conocimiento teórico de la cátedra y en adición de cátedras previas que debe aplicarse para su desarrollo. Además de habilidades sociales, investigativas y en planificación que requiere de los estudiantes.

2.3.2. Compilador

Definición

Un compilador puede designarse como un programa o sistema encargado de traducir un código en un lenguaje de programación, al que se denomina **lenguaje fuente** a un código equivalente (generalmente en un lenguaje de nivel inferior al lenguaje fuente) en **lenguaje objetivo o destino**. Si bien en un inicio estos se utilizan para la traducción de lenguajes de alto nivel como C o C++ a código de máquina (ejecutables), también existen traductores que se encargan de trasladar un código de un lenguaje de alto nivel a pseudo lenguajes o lenguajes de niveles ligeramente inferiores¹⁰.

A continuación, se puede observar un ejemplo del proceso básico:



Figura 1. Esquema base de un compilador. Adaptado de (Rojas y Mora, 2005).

Durante el proceso de traducción, el compilador analiza el código fuente, en busca de errores léxicos o sintácticos en el código entrante, antes de su “traducción”, de forma que no se terminara el proceso de traducción hasta que el codificador resuelva dichas ocurrencias.

¹⁰ LOUDEN Kenneth C. *Construcción de compiladores. Principios y práctica*. México: Parafino. 2004. p. 582.

Cabe resaltar que puede existir una confusión entre un intérprete y un compilador, siendo ambos softwares procesadores de lenguaje que tienen comportamiento similar, sin embargo, existen diferencias significativas que permiten distinguirlos. Empezando por el hecho de que un intérprete no genera directamente un “ejecutable” sino que más bien traduce las operaciones del código fuente y las ejecuta directamente en la máquina, de ahí la posibilidad de tener como entrada, además del código fuente, interacciones del usuario, conjunto a esto, un intérprete realiza ejecución “línea a línea” lo que permite detectar errores en el código fuente de una manera más completa¹¹.

Estructura de un compilador

Generalmente los compiladores suelen representarse como cajas negras, debido a su funcionamiento íntegro, pero en realidad existen diversas partes que conforman a un compilador con funciones claramente diferenciables.

Para poder entender la estructura de un compilador se requiere comprender que éstas están separadas de acuerdo a las etapas que conforman el proceso de traducción; análisis y síntesis.



Figura 2. Esquema de etapas en un compilador. Adaptado de (Rojas y Mora, 2005).

Análisis:

Como su nombre lo indica está conformada por diversos análisis que se encargan de verificar la estructura y el contenido del código fuente. Los elementos que constituyen esta etapa son:

- **Analizador lexicográfico:** Se encarga de sintetizar el código fuente en una secuencia ordenada equivalente conformada por categorías definidas. Además, se encarga de generar la tabla de control correspondiente.

¹¹ ROJAS Sergio, MORA Miguel. *Traductores Y Compiladores Con Lex/Yacc, Jflex/Cup Y Javacc*. España: Universidad de Málaga. 2005. p. 319.

- **Analizador Sintáctico:** Este elemento comprueba la estructura de la secuencia obtenida por el analizador lexicográfico, a través de la aplicación de reglas gramaticales.
- **Analizador semántico:** Después de validar la estructura de la secuencia entrante, este elemento garantiza el cumplimiento de las directrices del lenguaje, procurando la coherencia y el buen uso de sus elementos. Para esto hace uso de la tabla de símbolos.

Durante esta etapa los productos relevantes del proceso son la representación media, o secuencia de lexemas y la tabla de control, de los que se profundizará más adelante.

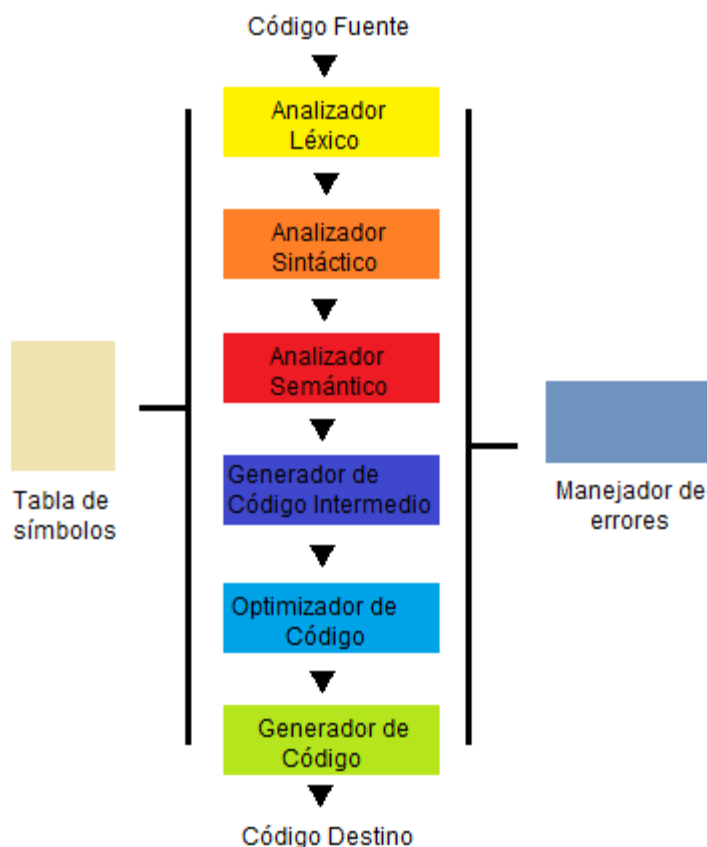


Figura 3. Esquema de un compilador por fases. Adaptado de (Rojas y Mora, 2005)

Síntesis:

La etapa de Síntesis se encarga de utilizar los elementos definidos en la representación intermedia y la tabla de símbolos, para generar el programa destino(ejecutable). En esta etapa ocurre lo que propiamente designamos como traducción. Elementos de la etapa:

- **Generador de código intermedio:** Utiliza la tabla de símbolos para la generación de un pseudocódigo independiente del código fuente, pero equivalente al mismo.
- **Optimizador de código:** Analiza el código intermedio para eliminar expresiones redundantes o elementos no utilizados.
- **Generador de código destino:** Con el código optimizado realiza la traducción final a código destino.

De igual manera en que existen elementos del compilador separados de acuerdo a las etapas del proceso de compilación también existen elementos que están presentes durante todo el proceso:

Tabla de control: Es una abstracción de los elementos que conforman el código fuente directamente relacionada con la secuencia de lexemas. Contiene elementos como valores, posiciones, tipos entre otros elementos representativos de los componentes de la secuencia. Si bien esta es creada en la etapa de análisis lexicográfico, se hace uso de esta durante las etapas siguientes de la traducción con el fin de validar estructura y posterior construcción de código destino.

Manejador de errores: Ya que el proceso de traducción tiene distintas etapas en las que se validan distintos elementos de la secuencia de entrada, el manejador de errores se encarga de permitir notificar de los errores que surgen durante el proceso de compilación al usuario y su posible solución. En algunos casos también puede contar con algoritmos para la autocorrección de errores.

Los compiladores pueden variar en su tamaño y complejidad. Sin embargo, no es algo muy común que los profesionales de la computación escriban por sí solos el código completo para la definición de un programa de esta naturaleza. Debido a la extensión del conocimiento necesario para la adecuada implementación de estos y las destrezas en otras prácticas relacionadas al diseño de compiladores¹².

¹² LOUDEN Kenneth C. *Construcción de compiladores. Principios y práctica.* México: Parafino. 2004. p. 582.

Sin embargo, esto no demerita la importancia de conocer los estamentos teóricos que se integran en los compiladores, ya que este tipo de conocimientos son aplicables ampliamente en otros campos de la computación. Hecho que es reconocido por la ACM dentro del currículum propuesto para las carreras relacionadas a las ciencias computacionales. Por lo cual la cátedra de diseño de compiladores, o denominada en otros espacios como autómatas y lenguajes formales es ampliamente integrada en distintos pensum de carreras técnicas y profesionales afines a la computación. No siendo la excepción el espacio de la universidad de Nariño donde la cátedra de diseño de compiladores aborda los elementos que constituyen la etapa de análisis de un compilador.

2.3.3. Análisis Léxico

Definición

El análisis lexicográfico, análisis léxico o rastreo es la primera etapa del proceso de traducción llevado por un compilador, este recibe como entrada un código en lenguaje fuente que está limitado por algún estándar de caracteres (como ASCII), y lo divide (generalmente carácter a carácter o usando algún otro método de recorrido), para formar fragmentos que tienen un sentido, de acuerdo a unas reglas o la aplicación de patrones. Estos fragmentos posteriormente son unidos para formar una secuencia equivalente al código fuente, la cual sirve como entrada para el analizador sintáctico.

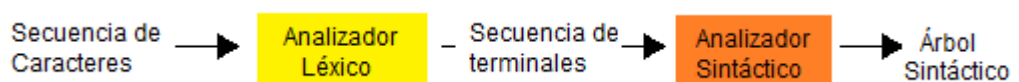


Figura 4. Primera etapa de análisis. Adaptado de (Rojas y Mora, 2005)

El tipo de algoritmos utilizados para este análisis no son únicamente utilizados dentro del ámbito de los compiladores, sino que estos pueden ser de gran utilidad para el desarrollo de lenguajes de consulta, búsqueda y recuperación siendo muy relevantes para la programación orientada por patrones. Donde el reconocimiento de secuencias clave es la esencia de su función.

Funciones del analizador léxico

Si bien la traducción del código entrante a la secuencia de lexemas es la función principal del análisis léxico, esta no es la única ya que este también se encarga de:

- Elimina comentarios
- Elimina separadores: espacios en blanco, tabulaciones y saltos de línea
- Construcción de tabla de símbolos (Elemento muy importante dentro del proceso de compilación)
- Notifica ocurrencias de errores léxicos.

Token, patrón y lexema

Los conceptos fundamentales tratados por el análisis léxico:

- **Patrón:** una expresión regular o regla
- **Token:** Categoría léxica asociada a un patrón
- **Lexema:** Secuencia de caracteres que coincide con un patrón

Por ejemplo. Se define un analizador léxico que reconoce expresiones numéricas y símbolos:

Token	Patrón	Lexema
NUM	(0-9)+	1 , 2 , 3 ...100...n
OP	(+,-,*,/)	+ , - , * , /

Ejemplo de patrón, token y lexema

Si se tiene la **secuencia de entrada:**

987 - 16 * 45 + 100

Se obtendrá la **secuencia de componentes léxicos:**

NUM OP NUM OP NUM OP

Como resultado de la aplicación de los patrones en el análisis léxico se obtiene la secuencia de componentes léxicos, que está conformada por los tokens a los que pertenecen los lexemas reconocidos. Los elementos como la posición y el valor de los elementos se almacenan en la tabla de símbolos.

¿Por qué es necesario el analizador léxico?

En la aplicación habitual de los compiladores estos no utilizan un orden lineal secuencial para el proceso de traducción. Lo que indica que el análisis léxico no se realiza completamente antes del análisis sintáctico o semántico. Por el contrario, el proceso de traducción consta de eventos cíclicos en los que el análisis léxico funciona en una pequeña instancia, reconociendo un elemento léxico, este alimenta al análisis sintáctico y este realiza un conjunto de validaciones bajo las que decide o no si el proceso continúa, y así este se repite hasta terminar de analizar el código entrante o hasta incurrir en un error. Esto es a lo que se denomina traducción orientada por sintaxis, ya que es el analizador sintáctico quien controla el proceso en general.

Este hecho ha causado que algunos autores dudan de la necesidad del análisis léxico, pero según Rojas y Mora, la construcción del analizador léxico provee de algunos atributos deseables para un sistema¹³.

- **Simplicidad en diseño:**

La separación del análisis léxico y el análisis sintáctico permite la modularidad en cuanto al desarrollo de estos elementos, facilitando la modificación, ampliación y mantenimiento de los mismo, lo que permite mantener estructuras más pequeñas y menos repetitivas.

- **Eficiencia:**

La división del analizador léxico permite la aplicación de algoritmos de recorrido y estructuras optimizadas para el proceso de reconocimiento. Punto en el que suele consumirse la mayor parte del tiempo del proceso de compilación.

- **Portabilidad:**

Al mantener el analizador léxico como un módulo independiente la reutilización e incluso reemplazo del mismo puede ser un proceso simple y rápido.

¹³ ROJAS Sergio, MORA Miguel. *Traductores Y Compiladores Con Lex/Yacc, Jflex/Cup Y Javacc*. España: Universidad de Málaga. 2005. p. 319.

Autómatas finitos- proceso de construcción

Para la implementación de analizadores léxicos existen diversas estructuras lógicas que permiten la integración de patrones para reconocimiento, pero entre las más usadas son los autómatas finitos debido a su rapidez y facilidad de implementación. 10

Los autómatas finitos o máquinas de estados finitos, como los miembros más básicos dentro de la teoría de autómatas, son estructuras matemáticas utilizadas para definir patrones en secuencias de reconocimiento. Estos autómatas son estructuras sencillas y realmente prácticas para la implementación de analizadores léxicos.

Para definir un autómata es necesario:

- Un conjunto finito de símbolos de entrada
- Un conjunto finito de estados
- Una función de transición
- un estado inicial
- Y al menos un estado de aceptación

Con estos elementos definir un autómata finito es un proceso sencillo.

La forma más ilustrativa para representar un autómata finito es a través de un diagrama. tome por ejemplo un analizador que reconoce identificadores, el autómata finito para su representación sería el siguiente:

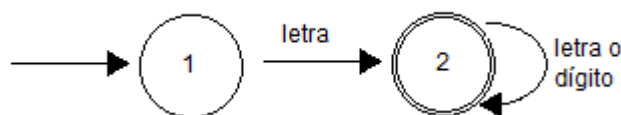


Figura 5. Diagrama de un autómata finito para identificadores. Adaptado de (Rojas y Mora, 2005)

En este diagrama los estados del autómata se representan por círculos, en este caso el autómata cuenta con dos estados.

El estado inicial se caracteriza por estar indicado por una flecha que no tiene conectado ningún otro estado. en el gráfico el estado 1 es el estado inicial.

Las demás flechas que tienen conectados dos estados, son las representantes de las funciones de transición, iniciando el cambio de un estado a otro, y el símbolo que efectúa la transición. En el gráfico el estado 1 está conectado al estado 2, y la flecha tiene como enunciado letra, esto indica que si en la secuencia de entrada se encuentra el símbolo letra se dará la transición del estado 1 al 2.

Por último, el estado de aceptación se representa con un círculo dentro de otro círculo. Para este caso el estado dos es el estado de aceptación.

El proceso llevado por el autómata en la figura para realizar el reconocimiento, dada una secuencia de entrada var1. Sería:

1 -v-> 2 -a-> 2 -r-> 2 -1-> 2

Como el estado dos es el estado de aceptación la secuencia es correcta.

Si, dada una secuencia, que se analiza por el autómata finito, este no termina en un estado de aceptación, la secuencia se rechaza.

2.3.4. Análisis Sintáctico

Definición:

El análisis sintáctico o análisis gramatical es la etapa en la que se verifica la estructura del programa fuente, utilizando para ello la secuencia dada por el analizador léxico, En este proceso se construye como representación del programa fuente, un árbol sintáctico, el que no es un elemento instanciado como tal sino más bien una estructura emergente resultado del proceso y la secuencia con que se aplican análisis a la secuencia de componentes léxicos.

durante este proceso se presenta el desafío del tratamiento de errores, ya que mientras que, en el análisis lexicográfico, el tratamiento de errores se reduce a denotar que expresión no pertenece a ninguno de los patrones definidos, en el análisis sintáctico el compilador debe ser capaz de detectar el error y recuperarse del mismo para continuar con el análisis, de manera que este logre detectar la mayor cantidad de errores posibles. para esto existen distintos procedimientos para el tratamiento de errores,

pero ya que estos requieren de un conocimiento avanzado en el ámbito de los compiladores, no se hace mención de este en la investigación¹⁴.

Gramáticas Libres de contexto

El análisis sintáctico es llamado también análisis gramatical, debido a que el comportamiento de éste está definido por una gramática libre de contexto o gramática formal.

Las gramáticas libres de contexto son la especificación jerárquica de la estructura de un lenguaje, a través de reglas recursivas. Al igual que en las expresiones regulares las reglas están definidas sobre el alfabeto del lenguaje, hacer uso de gramáticas puede facilitar significativamente la construcción de un analizador ya que su traducción al espacio computacional es relativamente sencilla.

Para definir una gramática libre de contexto son necesarios 4 componentes:

1. **(T)** Un conjunto de símbolos terminales, estos vienen dados por los que en el análisis léxico denominamos como tokens. Siendo símbolos elementales del lenguaje.
2. **(N)** Un conjunto de NO terminales, también llamados variables sintácticas. Estos no terminales hacen alusión a un conjunto de cadenas terminales. Generalmente se simbolizan con letras mayúsculas entre los símbolos < y >.
3. **(P)** Un conjunto de producciones, conformadas por un lado izquierdo y un lado derecho que están unidas por una flecha. El lado izquierdo de la producción se denomina encabezado y su lado derecho es un conjunto de símbolos terminales y no terminales a los que se denomina cuerpo.
4. **(S)** Un no terminal inicial.

Tome por ejemplo la gramática E, que permite reconocer expresiones aritméticas simples:

Donde:

T = {id, num, +, *, (,)}

¹⁴ AHO Alfred, LAM Monica SETHI Ravi ULLMAN Jeffrey. *Compiladores, principio técnicas y herramientas*. México: PEARSON EDUCATION. 2008. p. 801.

N = {E, T, F}

S = E

P =

1. $\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle$
2. $\langle E \rangle \rightarrow \langle T \rangle$
3. $\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle$
4. $\langle T \rangle \rightarrow \langle F \rangle$
5. $\langle F \rangle \rightarrow \text{id}$
6. $\langle F \rangle \rightarrow \text{num}$
7. $\langle F \rangle \rightarrow (\langle E \rangle)$

Derivación:

Es el proceso llevado a cabo para representar una secuencia de entrada con un árbol sintáctico, para ello se hace uso de las reglas de producción, en las que uno o más símbolos que se encuentran en el encabezado de una producción son reemplazados por el cuerpo de la misma dentro de la secuencia.

Existen varios procesos de derivación como:

Derivación por la izquierda: Empieza el proceso de sustitución por el elemento más a la izquierda de la secuencia entrante

Derivación por la derecha: Empieza el proceso de sustitución por el elemento más a la derecha de la secuencia entrante

Dependiendo del proceso de derivación la estructura de análisis o árbol gramatical es diferente. Como se puede observar en el ejemplo, utilizando la gramática E dada anteriormente:

Dada una secuencia de entrada en lenguaje fuente:

12 * (27 +16)

Se obtiene a través de análisis léxico la secuencia de componentes:

NUM * (NUM + NUM)

Desarrollando el análisis sintáctico de la secuencia de componentes léxicos aplicando la gramática **e** se tiene:

Con derivación por la izquierda:

Producción aplicada	Resultado
2	T
3	T * F
4	F * F
6	NUM * F
7	NUM * (E)
1	NUM * (E + T)
2	NUM * (T + T)
4	NUM * (F + T)
6	NUM * (NUM + T)
4	NUM * (NUM + F)
6	NUM * (NUM + NUM)

Con derivación por la derecha:

Producción aplicada	Resultado
2	T
3	T * F
7	T * (E)
1	T * (E + T)
4	T * (E + F)
4	T * (E + NUM)
2	T * (T + NUM)
4	T * (F + NUM)
6	T * (NUM + NUM)
4	F * (NUM + NUM)
6	NUM * (NUM + NUM)

Como se puede observar las formas secuenciales en las que se aplican las producciones en cada uno de los análisis son diferentes, aunque permiten reconocer la misma estructura.

Árboles Sintácticos

Obtener las formas secuenciales, o los procesos de derivación, para representar el proceso de análisis sintáctico puede ser poco ilustrativo para indicar el orden de derivación y las diferencias presentes dependiendo del método de derivación utilizado. Siendo por esto recomendable hacer uso de árboles sintácticos.

un árbol sintáctico corresponde a una sentencia haciendo uso de reglas gramaticales para la definición de su estructura. Estas estructuras como se ha mencionado son muy útiles para representar el proceso de derivación a partir de una sentencia¹⁵.

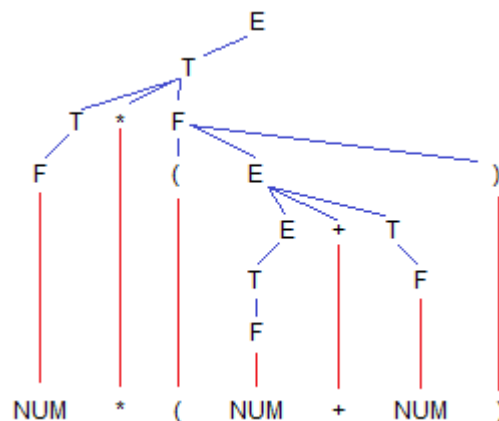
En la estructura de los árboles sintácticos se contempla:

Nodo raíz: Corresponde a un nodo que representa al no terminal inicial de una gramática. A partir del cual se desprende el proceso de derivación.

Nodos internos: Son elementos relacionados a los símbolos no terminales de la gramática, que se obtienen a través de la aplicación de producciones.

Nodos hoja: indican la aparición de un símbolo terminal, de igual manera se obtienen a través de la derivación aplicando reglas gramaticales.

Se puede tomar como ejemplo, los árboles de derivación resultantes del análisis sintáctico del ejemplo anterior.



Como se puede observar en el árbol, la derivación en el nodo raíz, pero esta parte hacia la izquierda. Como representante del proceso de derivación por la izquierda.

¹⁵ LOUDEN Kenneth C. *Construcción de compiladores. Principios y práctica.* México: Parafino. 2004. p. 582.

No terminales extraños

Dentro de los procesos de derivación a partir de gramáticas existen distintos desafíos, principalmente relacionados con el diseño y construcción de las gramáticas en sí, debido a que estas estructuras complejas pueden llegar a contener elementos que imposibiliten el proceso de análisis.

Es por eso que para evitar los errores más comunes en la definición de gramáticas se propone el tratamiento de los no terminales extraños, entre los que se encuentran:

- **No terminales Muertos:**

Son símbolos de una gramática que al ser utilizados dentro del proceso convencional de derivación pueden llevar a la creación de loops infinitos de análisis, ya que estos carecen de una definición de cierre que permita completar el proceso.

Un caso de ejemplo la gramática, A que permite reconocer secuencias de caracteres abc:

1. $\langle S \rangle \rightarrow a \langle B \rangle$
2. $\langle B \rangle \rightarrow b \langle C \rangle$
3. $\langle C \rangle \rightarrow c \langle S \rangle$

Si se utiliza la secuencia mínima reconocida por la gramática 'abc':

- 1 a $\langle B \rangle$
- 2 a b $\langle C \rangle$
- 3 a b c $\langle S \rangle$

En el proceso de derivación se reconoce la secuencia, pero esta mantendrá una variable gramatical abierta, correspondiente al no terminal $\langle S \rangle$. Para evitar este problema simplemente basta con eliminar el no terminal $\langle S \rangle$ de la 3ra producción de la gramática.

Aunque el ejemplo es bastante sencillo y el error de la gramática es fácilmente reconocible, dentro de gramáticas destinadas al reconocimiento de lenguajes, donde las producciones y sus símbolos son cuantiosos, el reconocimiento de este problema puede resultar complejo. Es por eso que existen algoritmos para su

evaluación y solución, como el utilizado dentro de la herramienta desarrollada por GADUN, descrito por LEWIS II¹⁶.

- **No terminales inalcanzables:**

Como su nombre lo indica estos símbolos pertenecen a producciones que no pueden ser alcanzadas a través de la derivación desde el símbolo no terminal inicial.

Un ejemplo de esta situación sería la siguiente gramática:

1 $\langle S \rangle \rightarrow a \langle B \rangle$
2 $\langle B \rangle \rightarrow b \langle D \rangle$
3 $\langle C \rangle \rightarrow c \langle D \rangle$
4 $\langle D \rangle \rightarrow d$

En esta gramática, el símbolo no terminal $\langle C \rangle$ no puede alcanzarse a través del símbolo no terminal inicial $\langle S \rangle$, y por consiguiente las producciones en las que este es encabezado tampoco son accesibles.

Este problema es muy común cuando se hacen cambios en las producciones de una gramática en especial tras aplicar algoritmos para eliminar no terminales muertos. De igual manera existen algoritmos para la detección de estos elementos. Como el también propuesto por LEWIS II¹⁷.

¹⁶ LEWIS II Phillip, ROSEKRANS Daniel, STEARNS Richard. *Compiler Design Theory*. Massachusetts. Addison-wesley. 1976. p. 647.

¹⁷ LEWIS II Phillip, ROSEKRANS Daniel, STEARNS Richard. *Compiler Design Theory*. Massachusetts. Addison-wesley. 1976. p. 647.

2.3.5. Procesamiento Descendente

Definición:

El procesamiento descendente es una de las grandes categorías en las que se dividen los métodos de análisis o parsing. Refiriéndose principalmente al orden en que las producciones se disponen en el árbol de derivación. En un sentido más técnico en el procesamiento descendente la etapa de reconocimiento de producciones a aplicar se produce en la etapa más alta o temprana del árbol sintáctico¹⁸.

Además, este procedimiento también se caracteriza por el uso de una máquina de pila, aplicando restricciones que hacen que el proceso se denomine como procesamiento descendente determinístico.

Se debe aclarar que este tipo de procesamiento hace uso de gramáticas libres de los ya definidos no terminales extraños.

principios del procesamiento descendente:

- Hace uso de símbolos especiales para el control de la pila, como los no terminales y el fondo de pila.
- Empieza el proceso de derivación a través de la inclusión del símbolo inicial de la gramática como símbolo inicial en el tope de la pila.
- La intersección entre el tope de la pila y un elemento de la secuencia léxica representa un paso dentro del proceso de sintaxis.
- El proceso de derivación está dado por la comparación de los no terminales en el cabezal de las producciones y el primer elemento de su cuerpo, por lo que la definición total de las producciones es poco relevante para el proceso
- Solamente la intersección entre el símbolo de fin de cadena de la secuencia léxica y el fondo de la pila representan un estado de aceptación de secuencia.
- Al utilizar como motor de procesamiento una máquina de pila, cada parser de procesamiento descendente cuenta con una tabla de control que describe detalladamente su comportamiento.

¹⁸ LEWIS II Phillip, ROSEKRANS Daniel, STEARNS Richard. *Compiler Design Theory*. Massachusetts. Addison-wesley. 1976. p. 647.

Tipos de gramáticas para el procesamiento descendente:

El procesamiento descendente puede presentar varias ventajas para el espacio académico, ya que su implementación es relativamente sencilla, son rápidas en su operación y requieren un uso ligero de recursos computacionales. Pero desafortunadamente no todas las gramáticas libres de contexto pueden ser aplicadas en este método.

Las gramáticas utilizadas dentro del procesamiento descendente son:

- Gramáticas S
- Gramáticas Q
- Gramáticas LL(1)

Gramáticas S:

Estas gramáticas deben cumplir con dos condiciones; primero el lado derecho de cada producción(cuerpo) debe empezar por un símbolo terminal, y segundo, si existen dos producciones en la gramática con el mismo símbolo en el lado izquierdo(encabezado), entonces sus lados derechos deben empezar por símbolos terminales diferentes.

Un ejemplo de una gramática S, sería el siguiente:

1. $\langle S \rangle \rightarrow ab \langle R \rangle$
2. $\langle S \rangle \rightarrow b \langle R \rangle b \langle S \rangle$
3. $\langle R \rangle \rightarrow a$
4. $\langle R \rangle \rightarrow b \langle R \rangle$

Como se puede observar, reconocer este tipo de gramáticas es muy sencillo ya que todas las producciones de la gramática cuentan con un lado derecho que empieza por símbolo terminal, y aquellas producciones con un lado izquierdo igual tienen lados derechos con sus primeros símbolos diferentes.

Gramáticas Q:

Las gramáticas Q no cumplen con la primera condición de las gramáticas S, puesto que algunas de sus producciones pueden tener lados derechos nulos(ϵ).

Como se puede observar:

1. $\langle S \rangle \rightarrow a \langle A \rangle \langle S \rangle$

2. $\langle S \rangle \rightarrow b$
3. $\langle R \rangle \rightarrow c \langle A \rangle \langle S \rangle$
4. $\langle R \rangle \rightarrow \epsilon$

Este símbolo especial denominado épsilon o vacío, es utilizado en las gramáticas para indicar un estado en el que no se debe realizar operaciones sobre la máquina de pila.

Para la implementación de estas gramáticas se hace uso del concepto de conjunto siguiente. término que se explicará más adelante.

Gramáticas LL(1):

Tome como ejemplo de LL(1) la gramática:

1. $\langle S \rangle \rightarrow \langle A \rangle b \langle B \rangle$
2. $\langle S \rangle \rightarrow d$
3. $\langle A \rangle \rightarrow \langle C \rangle \langle A \rangle b$
4. $\langle A \rangle \rightarrow \langle B \rangle$
5. $\langle B \rangle \rightarrow c \langle S \rangle d$
6. $\langle B \rangle \rightarrow \epsilon$
7. $\langle C \rangle \rightarrow a$
8. $\langle C \rangle \rightarrow d$

En este caso la gramática no solamente contempla producciones nulas, sino que también utiliza producciones donde su lado derecho empieza por un símbolo no terminal. Lo que incluye el concepto de conjunto primero, elemento que se contempla a continuación

Implementación de un parser de procesamiento descendente a partir de una gramática.

Como se ha indicado el procesamiento descendente tiene objetivo la construcción de parser que funcionan con una máquina de pila, a partir de las reglas dadas por una gramática, sea esta S, Q o LL(1). Ya que la implementación a partir de gramáticas LL(1) es la más completa, es la que se explicará a continuación.

Para la máquina de pila:

1. El conjunto de entrada para la máquina es el alfabeto de la gramática aumentado el símbolo de fin de cadena.

2. El conjunto de símbolos de pila consiste en los símbolos no terminales de la gramática, los símbolos terminales que no se encuentran en la primera posición del lado derecho de las producciones, y el símbolo de fondo de pila.
3. El estado inicial del procesamiento consiste en la pila con el símbolo de fondo de pila y el símbolo no terminal inicial.
4. El control de la máquina está descrito por la tabla de control, que tiene como cabeceras de las columnas los símbolos de entrada y como etiquetas de las filas, los símbolos de pila.

En consideración de estas reglas, la ejecución de las producciones en la máquina de pila se propone así.

Sea **a** un símbolo de entrada y **** el símbolo en el tope de la pila. Se aplicará a producción con lado izquierdo **a** donde **a** pertenezca a su conjunto selección.

Si la producción tiene la forma:

a. $\langle B \rangle \rightarrow a\lambda$

Se extrae el símbolo en el tope de la pila y se avanza al siguiente elemento de entrada en la secuencia léxica. Si λ es vacío el proceso culmina. Sino se ingresa los símbolos de λ en la pila en orden inverso.

b. $\langle B \rangle \rightarrow \epsilon$

Se extrae el símbolo en el tope de la pila y se retiene el elemento de entrada en la secuencia léxica.

c. $\langle B \rangle \rightarrow \langle C \rangle \lambda$

Se extrae el símbolo en el tope de la pila y se retiene el elemento de entrada en la secuencia léxica. Si λ es vacío el proceso culmina. Sino se ingresa los símbolos de λ en la pila en orden inverso.

Si el símbolo **a** no pertenece a ninguno de los conjuntos selección de las producciones con lado izquierdo ****. El proceso culmina con un mensaje de error.

Como se ha mencionado entonces el **conjunto selección** es un concepto clave para el procesamiento descendente, ya que este indica el conjunto

de símbolos de entrada bajo los cuales debe aplicarse la producción a la que pertenece.

Para obtener el conjunto selección se aplican las siguientes reglas.

- a. Si la producción tiene la forma:

$$\langle B \rangle \rightarrow \lambda$$

$$\text{Selección}(n) = \text{PRIMERO}(\lambda)$$

Donde n indica la posición de la producción en la gramática

λ es una combinación de símbolos terminales y no terminales no nulos

- b. Si el lado derecho de la producción tiene la forma:

$$\langle B \rangle \rightarrow \beta$$

$$\text{Selección}(n) = \text{PRIMERO}(\beta) \cup \text{SIGUIENTE}(B)$$

Donde n indica la posición de la producción en la gramática

β es una combinación de símbolos no terminales anulables. O un espacio vacío.

Siendo PRIMERO, el conjunto de símbolos terminales que se pueden obtener a partir de la derivación de un conjunto de símbolos terminales y no terminales, y SIGUIENTE el conjunto de símbolos terminales que pueden aparecer después de un símbolo no terminal.

2.3.6. Metacompiladores

Definición:

Los metacompiladores son compiladores de compiladores. Estos aceptan la descripción de un lenguaje y generan un compilador acorde al lenguaje definido¹⁹. También pueden denominarse generadores de parsers.

Como se ha indicado en previos ítems los compiladores son programas realmente complejos y es por esta razón que hasta ahora los metacompiladores son parciales. Las herramientas disponibles en el mercado permiten la creación de fragmentos de funciones específicas del

¹⁹ LOUDEN Kenneth C. *Construcción de compiladores. Principios y práctica*. México: Parafino. p. 84. 2004.

compilador, como el análisis léxico o el análisis sintáctico, ejemplos de ellos son Bison, JFlex, Yacc, Cup. Entre otros.

Como se ha mencionado existen diversas herramientas orientadas a la creación de compiladores, por lo que muchas de ellas comparten aspectos específicos o se diferencian especialmente por estos. uno de los más comunes es el método de entrada o algoritmo de análisis utilizado.

DFA:

Deterministic Finite Automaton. Cuando es usado como algoritmo de análisis, indica que los compiladores utilizan una máquina de estado finito para realizar el reconocimiento, generalmente se usa para la generación de herramientas de análisis léxico.

BNF:

Backus Naur form. Es un metalenguaje utilizado para construir gramáticas que describan las características de un lenguaje. No solamente es utilizado por los metacompiladores, puede ser utilizado también para explicar la naturaleza de un lenguaje como una práctica de diseño.

YACC

Yet Another Compiler Compiler. De por sí YACC es un recurso factible para la construcción de compiladores a partir de la definición de gramáticas, aun así, este es implementado por otras herramientas de forma que presentan una funcionalidad más compleja

2.4. FORMULACIÓN DE HIPÓTESIS

Los estudiantes de la asignatura de diseño de compiladores perciben el uso de una herramienta que permita visualizar las estructuras teóricas utilizadas en el procesamiento descendente y además su construcción a través de métodos interactivos; reflejando como resultado útil la generación de un código ejecutable en Python, como un elemento adecuado para apoyar el proceso de enseñanza aprendizaje llevado en la asignatura.

3. METODOLOGÍA

3.1. TIPO DE INVESTIGACIÓN

De acuerdo a Hurtado, la investigación proyectiva procura cómo deberían ser las cosas. Para esto analiza y plantea los elementos necesarios para que un evento ocurra adecuadamente²⁰.

Si bien dentro de la investigación proyectiva se destaca la creación de artefactos que permiten abordar una problemática con el fin de reflejar un cambio dentro de esta. Los medios previos para la construcción de los mismos tales como observación, documentación análisis de datos, teorización y planificación son los que permiten contemplar este tipo de investigación completamente.

Bajo este enunciado la investigación planteada en este documento se denomina como una investigación proyectiva, reconociendo el proceso de investigación documentación y análisis realizado para entender no solo la problemática sino también el contexto dentro de la comunidad académica e investiga al respecto, para la obtención de una tentativa de solución cuya construcción y validación tomaron también parte dentro de este proyecto.

3.2. DISEÑO DE LA INVESTIGACIÓN

Para el diseño de esta investigación, se tuvo en cuenta que en el marco de la investigación proyectiva Hurtado reconoce 8 fases que componen su metodología²¹.

Fase exploratoria: Análisis del contexto de la problemática a través de propuestas de otros autores.

Fase descriptiva: Utiliza el conocimiento planteado para la construcción de la problemática local y justifica su importancia.

Fase comparativa: Compara el evento local con las propuestas externas para establecer puntos de interés.

Fase analítica: Análisis de los síntomas del evento y explicaciones teóricas

²⁰ HURTADO Jacqueline. *Metodología de la investigación Holística*. Colombia. SYPAL. 2000. p. 664.

²¹ HURTADO Jacqueline. *Metodología de la investigación Holística*. Colombia. SYPAL. 2000. p. 664.

Fase explicativa: Utiliza el conocimiento adquirido para explicar el evento local.

Fase predictiva: Elaboración de estimaciones causales, posibles fenómenos, limitantes, plantea medios para la medición de los mismos.

Fase interactiva: Aplica los instrumentos y recoge datos del problema.

Fase confirmatoria: Análisis y conclusiones sobre el evento.

Fase evaluativa: Identifica limitaciones y recomendaciones para la continuación del estudio.

3.3. PROPÓSITO y CONTEXTO

Después de desarrollar las fases exploratoria y descriptiva de la investigación para la presente investigación se propone un objetivo pragmático, del cual se desprenden los objetivos específicos, donde la obtención de un producto útil para la resolución de la problemática presentada es un elemento común.

A partir de este supuesto se puede afirmar que este proyecto tiene por fin obtener información que avale si la solución propuesta en una herramienta didáctica puede utilizarse para cambiar las condiciones del proceso de aprendizaje dentro de la asignatura de diseño de compiladores en la universidad de Nariño, y a partir de esta proponer métodos, herramientas y alternativas para abarcar la problemática con medidas más complejas en un contexto más amplio. Por consiguiente, se delimita un contexto bajo el cual se desarrollan las siguientes etapas de la investigación.

La Universidad de Nariño es una entidad pública comprometida en el ámbito de la educación superior. Siendo una institución reconocida y con un gran trasfondo cultural e histórico dentro de la región nariñense al sur de Colombia. Bajo la misión de instruir profesionales con calidad técnica internacional que se destaquen por su formación integral en la que se reconoce el aporte de conocimiento científico, técnico y cultural, que están empeñados en impulsar el avance de la región. Aplicando la filosofía del respeto a la autonomía no solo del estudiante sino también del docente en su práctica, lo que conlleva a un espacio de formación peculiar.

Dispone de una alta diversidad de facultades y programas, como la medicina, el derecho, las artes, las ciencias sociales y por supuesto la ingeniería.

Centrándose en la facultad de ingeniería en su programa de ingeniería de sistemas, un marco donde se sigue un cuerpo de conocimientos fundamentado por organizaciones internacionales afines a las ciencias computacionales, como ACM, ACOFI e IEEE. Coherente al objetivo de la institución destaca por la formación de ingenieros hábiles en la construcción de software, no por el conocimiento de múltiples herramientas sino por la destreza que demuestran estos profesionales en cuanto a la aplicación de algoritmia y lógica para la solución de problemas.

Para finalizar la contextualización, la asignatura de diseño de compiladores impartida en este programa a estudiantes que cursan 8 - 7 semestre, sigue la normativa dada por la ACM. Lo que indica una revisión de los conceptos teóricos relacionados a análisis léxico, sintáctico y semántico, acompañada de una parte práctica de los dos primeros análisis. Representa el espacio donde se presenta la problemática discutida por este proyecto

3.4. PREGUNTAS DE INVESTIGACIÓN

Para orientar el proceso de investigación en sus etapas de explicativa y predictiva, en concordancia con los objetivos específicos propuestos en la investigación. Se definen las siguientes preguntas de investigación.

- ¿El conocimiento de herramientas de apoyo al aprendizaje de los compiladores existentes, permiten establecer los elementos a incluir en el aplicativo didáctico a desarrollar?
- ¿El desarrollo de un módulo del aplicativo para el análisis léxico que permita, entender sus conceptos en forma práctica y mirar su construcción en código fuente resultado, va a permitir al estudiante entender, su estructuración y construcción en un compilador?
- ¿El desarrollo de un módulo del aplicativo para el análisis sintáctico que permite, entender los conceptos de procesamiento descendente y gramáticas LL(1) y mirar su construcción en código fuente resultado, va a permitir al estudiante entender, su estructuración y construcción en un compilador?
- ¿La inclusión de un generador de código Python tanto en el análisis léxico como en el análisis sintáctico de los módulos en construcción, van a permitir al estudiante acercarse más a la estructuración y construcción de un compilador?
- ¿La aplicación de la herramienta didáctica a estudiantes que ya cursaron la asignatura de Diseño de Compiladores permite establecer su validez en la conceptualización y construcción de un compilador?

3.5. MÉTODOS: RELACIÓN ENTRE ESTUDIO Y SUS PROTAGONISTAS (POBLACIÓN Y MUESTRA)

En esta sección se propone el grupo de población y muestra, así como sus características, en adición de por qué se selecciona o cómo aporta al proyecto.

El grupo de estudiantes seleccionados como población del estudio está conformado por los estudiantes que cumplen con los atributos:

Atributo	Valor
Institución educativa	Universidad de Nariño
Facultad	Ingeniería
Programa	Ingeniería de sistemas
Cursó cátedra	Diseño de compiladores
Año en que cursó la cátedra	2019
Estatus de conclusión de asignatura	Aprobado
Grupo de estudio del curso	1, 2
Semestre en que curso	8
Modalidad	Presencial

Se debe aclarar que las cualidades propuestas para el grupo de la población están dadas en fin de permitir abordar la problemática teniendo en cuenta:

- Cercanía con respecto al contexto de la problemática planteada
- Reciente participación dentro de la problemática
- Objetividad con respecto al proceso de aprendizaje llevado en la asignatura y para la validación de la herramienta propuesta
- Claridad y uniformidad en cuanto a la base de conocimientos previos requeridos por la cátedra y los adquiridos en la misma
- Variables de los medios utilizados durante el desarrollo del curso

Con estos elementos contemplados se obtiene una población neta de 20 estudiantes que cumplen con las características solicitadas. Para el proceso de

muestreo se hace uso de las herramientas utilizadas dentro del muestreo aleatorio simple.

$$no = \frac{Z^2 * p * q}{e^2} \quad \text{ecuación para obtención de muestra}$$

$$n' = \frac{no}{1 + \frac{(no - 1)}{N}} \quad \text{ecuación para obtención de muestra ajustada}$$

Donde

no = Tamaño de muestra sin considerar la población

n' = Tamaño de muestra ajustado a la población

N = indica el tamaño de la población

Z = Es el valor del nivel de confianza

P = Proporción de éxito

Q = Probabilidad de fracaso, complementaria a P

e = Máximo margen de error permitido

Tras la aplicación de estas ecuaciones se obtiene una muestra de 20 estudiantes, observando que se mantiene el número de individuos de la población debido a al número reducido de personas que la integran.

Los parámetros utilizados para su obtención se explicarán en la sección de validez y confiabilidad.

3.6. MÉTODOS: INSTRUMENTOS DE RECOLECCIÓN DE INFORMACIÓN

Observación

De acuerdo con Gallardo, la observación participativa es un tipo de observación no estructurada en la que el investigador tiene objetivos meramente exploratorios ya que carece de conocimientos que le permitan establecer hipótesis que dirijan su proceso²². A través de esta técnica el observador pretende obtener información que le permita entender los elementos presentes en el evento local.

En base a esta aclaración, el proceso de observación llevado a cabo desde la perspectiva de estudiantes de la asignatura de diseño de compiladores en la universidad de Nariño, permitió evidenciar la existencia de dificultades al abordar ciertas temáticas del curso, lo que se veía reflejado en una actitud negativa con respecto a la cátedra de compiladores.

Este hecho se confirma de acuerdo a la perspectiva docente donde se pudo visualizar que los estudiantes presentan bajo rendimiento académico y un promedio del grupo “insatisfactorio”. Además, se responde que la cuestión no es un asunto cerrado del grupo observado sino que este fenómeno se repite ha repetido dentro de un espacio cronológico y en otras instituciones de educación superior.

Con esta información se planteó como medio de delimitación del evento local el uso de un formulario orientado a evidenciar que temáticas presentan dificultad dentro de la cátedra diseño de compiladores. De manera complementaria también se realizó una revisión documental, para entender el contexto general de la problemática y orientar un proceso de investigación más estructurado.

Revisión documental

Se hace uso de este método para obtener información válida relacionada con el uso de herramientas en espacios de formación académica para generar cambios en el proceso de aprendizaje.

Con la información obtenida se obtuvo una perspectiva de la implementación adecuada de herramientas didácticas, los atributos que estas poseen y como

²² GALLARDO Yolanda. *APRENDER A INVESTIGAR, MÓDULO 3 RECOLECCIÓN DE LA INFORMACIÓN*. Colombia: ICFES. 1999.

está orientado su uso. Además, se revisan proyectos similares o relacionados a la temática de compiladores con el fin de visualizar las distintas alternativas propuestas por otros profesionales para solucionar la problemática común.

Por último, se utiliza varias investigaciones para obtener un sumario de herramientas software utilizadas dentro de la práctica de diseño de compiladores para evaluar los puntos de la problemática y además para obtener atributos deseables en la herramienta desarrollada.

Entendido este punto, se enfatiza en que el único punto discriminante para la documentación revisada es la temática, ya planteada, y el origen de información. Todos los documentos revisados son obtenidos de bases de datos científicas como IEEE explore, ACM digital library IJJSR o están avalados por una IES.

Encuesta

Gallardo define a la encuesta como “Técnica destinada a obtener información primaria, a partir de un número representativo de individuos de una población, para proyectar sus resultados sobre la población total”²³. Permitiendo unidades de estudio mayores que las de la observación al igual que la cantidad de variables que pueden estudiarse.

Para la construcción de una encuesta se debe establecer:

- **Definición de los conceptos:** Describir sus conceptos y construir indicadores
- **Diseño del cuestionario de la encuesta:** Construcción de preguntas
- **Diseño de la muestra:** Selección y entendimiento de la muestra
- **Trabajo de campo de la encuesta:** Presentación del cuestionario
- **Registro de la encuesta:** Registro de datos en un formato

Para la estructuración del cuestionario presentado a partir del 26 de octubre de 2020, para para la validación de la hipótesis de la investigación, se cuenta con los siguientes puntos:

²³ GALLARDO Yolanda. *APRENDER A INVESTIGAR, MÓDULO 3 RECOLECCIÓN DE LA INFORMACIÓN*. Colombia: ICFES. 1999.

- **Definición de los conceptos:**

Como una abstracción al modelo de evaluación de materiales educativos computarizados de Galvis²⁴, se establece los espacios fundamentales de evaluación:

Visibilidad: Indica el grado en que la herramienta representa a través de su interfaz y elementos gráficos los conceptos teóricos y la coherencia con la que estos se presentan.

Interactividad: El grado en que la interacción con la herramienta permite al usuario apropiarse y comprender conceptos teóricos.

Además, para obtener el grado en que los estudiantes perciben a la herramienta como un apoyo, se propone el ítem de **apoyo**, que define la facilidad con la que se puede interactuar con la herramienta y la ayuda que perciben los estudiantes con las distintas funcionalidades de la herramienta, especialmente la generación de código para la construcción de un semi-compilador.

Conjunto a estos conceptos, se hace uso de una lista de temas en los cuales el estudiante debe percibir la visibilidad de teoría su correspondiente interacción y como esto influye en su proceso de aprendizaje.

- Definición y elementos del análisis léxico.
- Gramática y elementos del análisis sintáctico.
- Construcción de un compilador.

En base a estos elementos se propone las siguientes escalas e ítems de evaluación:

²⁴ COVA Ángela, ARRIETA Xiomara, AULAR Judith, REVISIÓN DE MODELOS PARA EVALUACIÓN DE SOFTWARE EDUCATIVOS. Primera edición. Venezuela: Revista Electrónica de Estudios Telemáticos. 2008. vol 7.

Visibilidad y Coherencia

En esta sección se evaluó las variables de Galvis, **contenido y actitud de los estudiantes**, de acuerdo con el grado en que la herramienta permite visualizar los conceptos teóricos relacionados al procesamiento descendente y la coherencia con la que estos se presentan dentro de ella.

Escala de evaluación:

Rango	Evaluación
0 - 1	El ítem no se visibiliza, carece de sentido o presenta información incoherente.
2 - 3	El ítem es visible, pero se presenta de manera inadecuada o la información presentada es incompleta o insuficiente
4	El ítem es visible, presenta información en forma aceptable y suficiente. Presentar características menores para mejorar
5	El ítem es visible, presenta información de manera adecuada y coherente

Ítems evaluados:

- ☐ Visibilidad de proceso y elementos involucrados en el análisis léxico
- ☐ Visibilidad de proceso y elementos involucrados en el análisis sintáctico
- ☐ Visibilidad de continuidad dentro del proceso de construcción de un compilador

Comprensión y Análisis

Equivalente a la variable de Galvis de **estrategias de instrucción** evalúa el grado en que las propuestas de interacción con la herramienta permiten al usuario apropiarse y comprender conceptos teóricos relacionados con el procesamiento descendente.

Escala de evaluación:

Rango	Evaluación
0 - 1	El ítem no está presente en la herramienta o carece de interacción
2 - 3	El ítem está presente, pero su interacción es poco adecuada para su entendimiento y análisis
4	El ítem presenta una interacción aceptable y permite comprender su concepto. Presentar características menores para mejorar
5	El ítem presenta una interacción muy adecuada para la comprensión de su concepto

Ítems evaluados:

- ☐ Análisis de proceso y elementos involucrados en el análisis léxico
- ☐ Análisis de conceptos relacionados con gramáticas
- ☐ Análisis de conceptos relacionados con procesamiento descendente
- ☐ Análisis de conceptos involucrados en los análisis léxicos y sintáctico a través de código

Apoyo

Esta sección evalúa la facilidad con la que se puede interactuar con la herramienta y el apoyo que perciben los estudiantes con las distintas funcionalidades de la herramienta, especialmente la generación de código para la construcción de un semi-compilador. Como una abstracción de las variables propuestas por Galvis; **Objetivo del material y Realimentación de desempeño.**

Escala de evaluación:

Rango	Evaluación
0 - 1	El ítem no está presente en la herramienta o tiene una funcionalidad confusa y que puede dificultar el proceso de aprendizaje
2 - 3	El ítem presenta una funcionalidad precaria o que presenta poca utilidad para el estudiante
4	El ítem presenta una funcionalidad que puede facilitar el aprendizaje y análisis para el estudiante. Presentar características menores para mejorar
5	El ítem presenta una funcionalidad que facilita completamente el aprendizaje y análisis para el estudiante

Ítems evaluados:

- ☐ Apoyo en la aplicación de conceptos y construcción de analizadores léxicos
- ☐ Apoyo en la aplicación de conceptos relacionados con gramáticas y su construcción
- ☐ Apoyo en la obtención de bases para la construcción de un semi-compilador
- ☐ Apoyo en la apropiación de conceptos a través de código

- **Diseño del cuestionario de la encuesta:**

Para la construcción de preguntas se hace uso de los ítems dados en la definición de conceptos.

En la siguiente tabla se consigna la rastreabilidad de los ítems con respecto a las preguntas.

VISIBILIDAD Y COHERENCIA	
Ítem 1	Visibilidad de proceso y elementos involucrados en el análisis léxico
Rango 1	1 - 2
Pregunta clave 1	Los elementos gráficos de la herramienta permiten visualizar los componentes de un analizador léxico y su función dentro de un compilador.
Ítem 2	Visibilidad de proceso y elementos involucrados en el análisis sintáctico
Rango 2	3 - 4
Pregunta clave 2	Los elementos gráficos de la herramienta permiten visualizar los componentes de un analizador sintáctico y su función dentro de un compilador.
Ítem 3	Visibilidad de continuidad dentro del proceso de construcción de un compilador
Rango 3	5 - 6
Pregunta clave 3	Los elementos gráficos de la herramienta permiten visualizar las etapas del proceso de análisis llevado por un compilador desde análisis léxico hasta sintáctico.
COMPRENSIÓN Y ANÁLISIS	
Ítem 1	Análisis de proceso y elementos involucrados en el análisis léxico
Rango 1	1 - 5

Pregunta clave 1	La herramienta permite interactuar con los analizadores léxicos definidos de manera didáctica dando a entender la participación de los elementos definidos
Ítem 2	Análisis de conceptos relacionados con gramáticas
Rango 2	6 - 8
Pregunta clave 2	La herramienta permite plantear gramáticas como estructuras para la construcción de lenguajes
Ítem 3	Análisis de conceptos relacionados con procesamiento descendente
Rango 3	9 - 14
Pregunta clave 3	La herramienta permite explorar el concepto de primero, siguiente y selección en una gramática.
Ítem 4	Análisis de conceptos involucrados en los análisis léxicos y sintáctico a través de código
Rango 4	15 - 17
Pregunta clave 4	El código generado por la herramienta permite entender de manera básica el proceso de reconocimiento llevado por un analizador sintáctico
APOYO	
Ítem 1	Apoyo en la aplicación de conceptos y construcción de analizadores léxicos
Rango 1	1 - 2
Pregunta clave 1	La herramienta permite la creación de analizadores léxicos de manera sencilla y en poco tiempo.
Ítem 2	Apoyo en la aplicación de conceptos relacionados con gramáticas y su construcción
Rango 2	3 - 5
Pregunta clave 2	El uso de una tabla de control y la explicación básica de las producciones de una gramática le permiten mejorar su experiencia de aprendizaje en

	procesamiento descendente.
Ítem 3	Apoyo en la obtención de bases para la construcción de un semi-compilador
Rango 3	6 - 8
Pregunta clave 3	El código generado es una base sencilla y comprensible para la construcción de compiladores más sencillos.
Ítem 4	Análisis de conceptos involucrados en los análisis léxicos y sintáctico a través de código
Rango 4	9 - 10
Pregunta clave 4	Observar el código ejecutable de las estructuras definidas en clase ayuda a entender la transición entre teórico y lo práctico

El cuestionario Final cuenta con 37 preguntas de las cuales 34 pertenecen a las secciones definidas. Aprobadas por el docente asesor. Las preguntas sobrantes se establecen como preguntas no determinantes para la investigación.

Para observar la estructura completa de preguntas presentadas, revise el documento en la carpeta de anexos con el nombre “ANEXO_A Formulario de validación GADUN.pdf”.

- **Diseño de la muestra:** La muestra obtenida es la descrita en la sección de métodos (Población y muestra)
- **Trabajo de campo de la encuesta:**
El proceso de ejecución del test comprende una etapa preparatoria en la que a los estudiantes se les presenta un conjunto de medios didácticos y recursos complementarios con el objetivo de que se visualice el espacio de aplicación del software. Comprendido por un plazo de una semana. Paso siguiente se habilita el formulario a través de la plataforma de Google forms, donde se han implementado las preguntas de evaluación.
- **Registro de la encuesta:** Para el registro de los datos de la encuesta se utiliza un libro de Google sheets en las que cada pregunta comprende una

de las preguntas del formulario y las filas indican cada una de las respuestas.

3.7. MÉTODOS: PROCESAMIENTO DE INFORMACIÓN

El procesamiento de la información consiste en la operación que implica la recolección, ordenamiento y síntesis de datos, de manera que estos puedan presentarse en un formato útil para el investigador²⁵.

Recolección:

Como se indicó en la sección de métodos el medio utilizado para la recolección de los datos para el formulario de validación es a través de la herramienta de Google forms, a partir de las respuestas dadas se genera un libro de Google sheets.

Ordenamiento:

En cuanto al ordenamiento y validación los datos obtenidos se disponen en filas y columnas, donde cada columna indica una pregunta del formulario y la fila indica la secuencia de respuestas dadas por cada estudiante.

	Pregunta		
Participante	La herramienta permite v	La herramienta permite v	La herramient
Estudiante001	5	5	
Estudiante002	4	5	
Estudiante003	5	5	
Estudiante004	5	5	
Estudiante005	5	5	
Estudiante006	5	5	
Estudiante007	5	4	
Estudiante008	4	5	

A partir de este formulario se crea dos tablas de datos ordenados:

La primera tabla utiliza un proceso de categorización simple, se realiza el conteo de las respuestas que pertenecen a un rango de valores específica en una respuesta dada, las frecuencias obtenidas se relacionan a la pregunta perteneciente y al ítem evaluado. A partir de esta se dispone a hacer un análisis estadístico de la perspectiva de los estudiantes.

²⁵ HERNÁNDEZ Roberto, FERNÁNDEZ Carlos, BAPTISTA Pilar. Metodología de la Investigación. México: MCGRAW-HILL. 2006.

Ítem	Pregunta	Rango 1	Rango 2	Rango 3	Rango 4	Total
I1	P1	I1P1F1	I1P1F2	I1P1F3	I1P1F4	I1P1F1 + I1P1F2 + I1P1F3 + I1P1F4 = T1
I1	P2	I1P2F1	I1P2F2	I1P2F3	I1P2F4	I1P2F1 + I1P2F2 + I1P2F3 + I1P2F4 = T2
I1	Pn	I1PnF1	I1PnF2	I1PnF3	I1PnF4	I1PnF1 + I1PnF2 + I1PnF3 + I1PnF4 = Tn

Tabla de datos ordenados, análisis estadístico

En esta tabla **ítem** indica el ítem al que pertenece la pregunta, **Pregunta** indica la cuestión evaluada, **Rango 1** indica la frecuencia obtenida para el primer rango (0 - 1), de la misma manera hasta el **Rango 4(5)**, estos de acuerdo a las escalas propuestas para el formulario, y **Total** es la cantidad de participantes que evaluaron el ítem.

Como segunda instancia del ordenamiento se aplica escala de Likert, para obtener la calificación obtenida por la herramienta.

Sección		
ÍTEM	Puntos por Ítem	Puntaje Obtenido
I1	(#PI1 * Participantes)* 5	Sumatoria(Valor PI1)
I2	(#PI2 * Participantes)* 5	Sumatoria(Valor PI2)

Tabla de datos ordenados, escala de Likert

Para esta tabla, la **Sección** indica la variable evaluada, visibilidad, coherencia o apoyo. **Ítem** indica el ítem evaluado perteneciente a la sección. **En puntos por ítem**, se registra la cantidad de puntos ideal que se debe obtener por el ítem, se calcula a través del producto entre el **número de preguntas(#PI)** por ítems y el **número de participantes del formulario (Participantes)**, a este valor se multiplica por **5(valor de evaluación ideal)**. Mientras que el **puntaje obtenido**

es la sumatoria de las evaluaciones dadas por los estudiantes en todas las preguntas pertenecientes al ítem.

Síntesis:

Para sintetizar los datos obtenidos en la primera tabla se hace uso de los ítems dados el punto de métodos. Ya que cada pregunta tiene una relación directa con cada uno de los ítems de evaluación.

En el proceso se utiliza una sumatoria básica, primero se agrupa las preguntas que pertenecen al mismo ítem y se suman las frecuencias de cada uno de sus rangos, agrupados por rango, seguido se obtiene la razón entre el dato obtenido y la sumatoria de las frecuencias totales de las preguntas agrupadas, y por último se multiplica el valor por 100 para obtener su valor porcentual.

Ítem	Rango 1	Rango 2	Rango 3	
I1	$\frac{(I1P1F1 + I1P2F1 + I1PnF1)}{(T1 + T2 + Tn)} * 100$	$\frac{(I1P1F2 + I1P2F2 + I1PnF2)}{(T1 + T2 + Tn)} * 100$	$\frac{(I1P1F3 + I1P2F3 + I1PnF3)}{(T1 + T2 + Tn)} * 100$	$\frac{(I1P1F3 + I1P2F3 + I1PnF3)}{(T1 + T2 + Tn)} * 100$
I2	$\frac{(I2P1F1 + I2P2F1 + I2PnF1)}{(T1 + T2 + Tn)} * 100$	$\frac{(I2P1F2 + I2P2F2 + I2PnF2)}{(T1 + T2 + Tn)} * 100$	$\frac{(I2P1F3 + I2P2F3 + I2PnF3)}{(T1 + T2 + Tn)} * 100$	$\frac{(I2P1F3 + I2P2F3 + I2PnF3)}{(T1 + T2 + Tn)} * 100$

Tabla de datos estadísticos sintetizados

3.8. MÉTODOS: VALIDEZ Y CONFIABILIDAD

Según las ecuaciones presentadas en 3.5 Población y muestra:

$$no = \frac{Z^2 * p * q}{e^2} \quad \text{ecuación para obtención de muestra}$$

$$n' = \frac{no}{1 + \frac{(no - 1)}{N}} \quad \text{ecuación para obtención de muestra ajustada}$$

Donde

no = Tamaño de muestra sin considerar la población

n' = Tamaño de muestra ajustado a la población

N = indica el tamaño de la población

Z = Es el valor del nivel de confianza

P = Proporción de éxito

Q = Probabilidad de fracaso, complementaria a P

e = Máximo margen de error permitido

Como se explicó en el apartado de población, de los estudiantes tomados como población. Se establecen los siguientes parámetros.

N	20
Z	1.96
P	0.5
Q	0.5
e	0.02

Tome en cuenta que según Mezo²⁶, para la obtención de una encuesta apropiada se recomienda un **nivel de confianza de 95%(z=1,96)** y un **error permitido de 2%(0.02)**. Además, para la obtención de la probabilidad de éxito se debe utilizar datos obtenidos por parte de antecedentes en la investigación, de no tenerlos se recomienda usar un valor superior a 0.4 e inferior a 0.6.

$$no = 2401$$

²⁶ MEZO Josu. *Encuestas y margen de error: una guía práctica*. Madrid: Cuadernos de Periodistas. Publicación número 30. 2015.

$$n' = 19,84297521$$

Tras la aplicación del formulario de validación se obtiene la respuesta de 18 estudiantes sobre los 20 planteados. Con esta situación se alteran las características de confiabilidad.

N	20
Z	1.31
P	0.5
Q	0.5
e	0.05

Partiendo desde una clara disminución en el nivel de confianza, obteniendo un 90%(z=1,31) de confiabilidad y un aumento en el margen de error a 5%. De acuerdo a Mezo se aleja del nivel deseado de confiabilidad, y disminuye la replicabilidad de los resultados, pero se mantiene dentro de márgenes aceptables.

En cuanto a la validez, Gallardo dice que para evaluar el grado de validez de un instrumento se debe tomar en cuenta los siguientes aspectos²⁷:

- **Validez de contenido:**

Dominio del instrumento sobre el evento medido.

En el formulario presentado para la evaluación de la herramienta y en el marco de la investigación se entiende los aspectos fundamentales que abarcan la problemática. Desde las características de la herramienta como un elemento clave en el proceso de aprendizaje hasta los elementos que los estudiantes utilizan para alterar el evento.

- **Validez de criterio**

Validez del instrumento en comparación con propuestas externas.

²⁷ GALLARDO Yolanda. *APRENDER A INVESTIGAR, MÓDULO 3 RECOLECCIÓN DE LA INFORMACIÓN*. Colombia: ICFES. 1999

La validación de la investigación toma elementos propuestos por Mernik²⁸ donde se realiza la sustracción de la perspectiva de los estudiantes con respecto al uso de LISA como herramienta para el apoyo en el curso de compiladores. Obteniendo valores significativos con un respaldo entre el 95%.

A partir de estos se propuso tomar dentro de la investigación valores significativos como aquellos con al menos un 90% de apoyo por la población, los otros elementos se sometieron a una interpretación más analítica.

- **Validez de concepto**

Puntuaciones de la prueba.

Para esto se hizo uso de la escala de Likert, que permite obtener de manera cuantitativa una valoración de las variables propuestas.

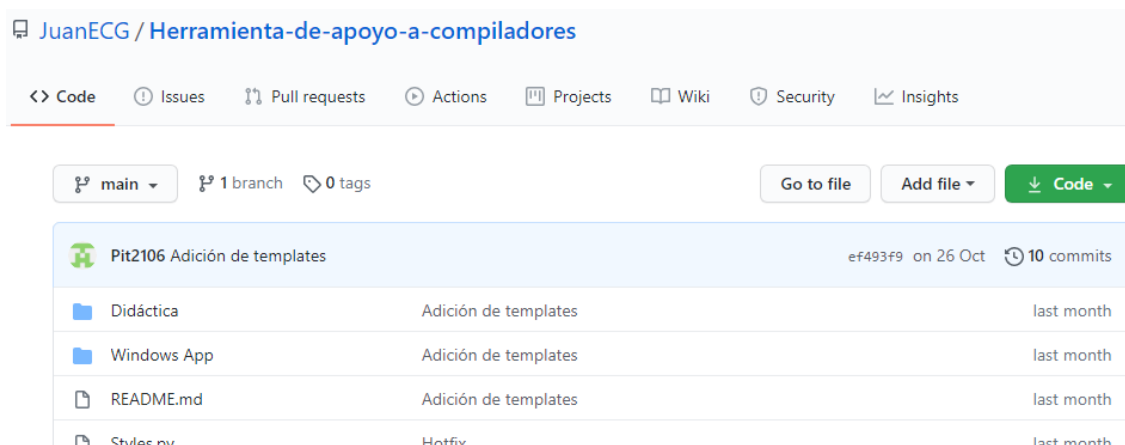
²⁸ MERNIK Marjan, ZUMER Viljem. *IEEE Transactions on Education: An Educational Tool For Teaching Compiler Construction*. IEEE. 2003. vol. 46

4. RESULTADOS DE LA INVESTIGACIÓN

4.1. Herramienta de apoyo

Durante el desarrollo de esta investigación se desarrolló una herramienta didáctica con el objetivo de aplicar un proceso alternativo para la representación de estructuras utilizadas para el procesamiento descendente y la obtención de su código ejecutable.

El producto de dicho proceso se publicó en un repositorio de GitHub con el fin de permitir el acceso a la misma por parte de los estudiantes que presentaran la encuesta de validación. [Repositorio herramienta de apoyo a compiladores.](#)



Además de albergar la herramienta ejecutable, este repositorio contiene tanto el código fuente de la herramienta como los recursos didácticos utilizados para la misma (video, ejercicios, manual). Todos estos liberados para el uso de los estudiantes.

4.1.1. Ítems contemplados para desarrollo

Para definir los elementos que contemplaría la herramienta se hizo uso de un conjunto de datos obtenidos tras la revisión documental inicial. De los cuales se obtuvo información secundaria sobre distintas herramientas que permiten la construcción de compiladores.

En el directorio de anexos dentro del fichero "ANEXO_B Revisión documental y de herramientas.docx" se puede encontrar mayor información.

Con la información obtenida se sintetizó una lista de ítems deseables para la herramienta didáctica.

- Integración de componentes para el ámbito educativo
- Uso de métodos de visualización para la construcción de estructuras
- Uso de medios de visualización para análisis
- Definición de estructuras lógicas para análisis léxico y sintáctico de manera simple
- Visualización de conceptos relacionados
- Observación del funcionamiento de análisis léxico
- Proceso de depuración o paso a paso dentro de la herramienta
- Disponibilidad de herramienta y documentación

A partir de estos ítems se ejecutó la construcción de la herramienta didáctica que permite construir analizadores léxico - sintácticos utilizando una interfaz que evidencia la aplicación de conceptos teóricos relacionados al procesamiento descendente, en un ámbito simple donde también es posible el análisis de estas estructuras. Como último factor se especificó la generación del código de las estructuras dadas a través del lenguaje Python, como un recurso fácil de entender e implementar.

4.1.2. Proceso de desarrollo

Para el desarrollo de la herramienta se aplicó la metodología de desarrollo ágil SCRUM, como una alternativa adecuada para desarrollar productos útiles en cortos periodos de tiempo.

El periodo de desarrollo estimado fue de 4 meses, durante los cuales se desarrollaron sprints o iteraciones con duración de 30 días.

No	Descripción de la Historia de Usuario	Valor que agrega al proceso del cliente [1-10]	Urgencia con la que se debe desarrollar [1-10]	Prioridad	Complejidad
1	Como usuario quiero gestionar un token a través de una expresión regular o un conjunto de elementos	10	10	100	Alta

	terminales.				
2	Como usuario quiero generar una prueba sobre los tokens que he definido.	9	7	63	Alta
3	Como usuario quiero gestionar los elementos del conjunto subconjunto cargador por defecto.	7	8	56	Alta
4	Como usuario quiero gestionar un No Terminal.	8	7	56	Baja
5	Como usuario quiero gestionar la tabla de producciones.	8	6	48	Media
6	Como usuario quiero añadir y eliminar elementos dentro de las producciones de la tabla de producciones del analizador sintáctico.	7	6	42	Alta
7	Como usuario quiero validar la gramática descrita por las filas/producciones de la tabla de producciones del analizador sintáctico.	9	5	45	Alta
8	Como usuario quiero examinar la tabla de control cuando esta sea habilitada.	9	5	45	Alta
9	Como usuario quiero agregar un mensaje de error personalizado a las casillas de la tabla de control que lo permitan.	5	5	25	Muy baja
10	Como usuario quiero obtener información de cada elemento que no resulte en un error de la tabla de control del analizador sintáctico.	7	4	28	Media
11	Como usuario quiero generar una prueba sobre la gramática construida cuando esta se haya validado.	10	4	40	Alta

12	Como usuario quiero enviar tokens del analizador léxico al analizador sintáctico.	7	7	49	Baja
13	Como usuario quiero abrir, crear y guardar archivos que permitan llevar cuenta del trabajo realizado.	9	8	72	Alta
14	Como usuario quiero generar código que ejecute los elementos definidos en los dos analizadores.	10	6	60	Muy alta
15	Como usuario quiero obtener ayuda respecto a un inconveniente en específico o información adicional sobre el código y el proyecto en general.	7	4	28	Baja

Product backlog del proceso de desarrollo.

El artefacto principal que orientó el desarrollo fue el sprint backlog, con el cual se pudo obtener una vía de prioridad en el desarrollo de la herramienta.

La información detallada del proceso de desarrollo se puede encontrar en la carpeta ANEXOS los documentos “ANEXO_C Documentación metodología SCRUM”. Y “ANEXO_D Tablas de gestión de desarrollo”.

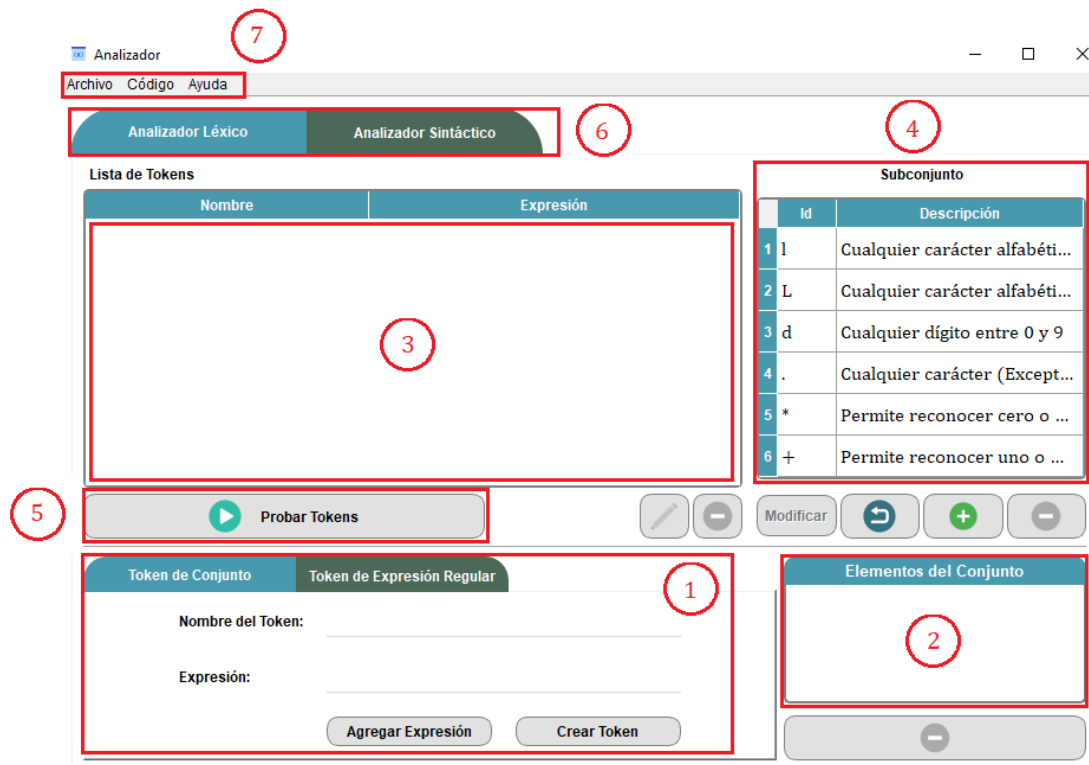
4.1.3. Componentes del software

El software desarrollado para esta investigación, al que se ha denominado GADUN(Generador de Analizadores Didáctico para la Universidad de Nariño) abarca elementos del análisis descendente y sus relacionados, con el objetivo de ilustrar cómo la construcción de un compilador es un proceso continuo.

GADUN se puede dividir en 3 Módulos distintos, que dependen entre ellos.

- **Análisis Léxico**

En esta etapa se integra elementos tratados en el curso de diseño de compiladores como lo son, el uso de expresiones regulares, operaciones con cadenas y Tokens. De manera que se obtiene una interfaz simple en su interacción, pero con abundantes conceptos teóricos integrados.



La sección de análisis léxico contiene 7 áreas relevantes.

1. Área de definición de tokens:

Token de Conjunto **Token de Expresión Regular**

Nombre del Token: _____

Expresión: _____

Agregar Expresión **Crear Token**

En esta área se facilitan al usuario los medios para definir tokens tanto a través de expresiones regulares como expresiones

específicas como un medio sencillo para la definición del comportamiento del analizador léxico

2. Área de validación de tokens

Elementos del Conjunto	Validar Expresión

Ya que en el proceso de construcción de tokens pueden ocurrir distintos errores, esta sección permite tener un control de las expresiones utilizadas en los tokens de conjunto y validar expresiones regulares para los tokens de expresión regular.

3. Área de visualización de tokens definidos por usuario

Lista de Tokens

	Nombre	Expresión
1	número	d+

Presenta una lista de los tokens creados por el usuario, indicando el identificador y la expresión regular del elemento.

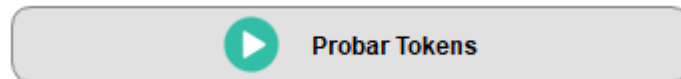
4. Área de subconjuntos

Subconjunto

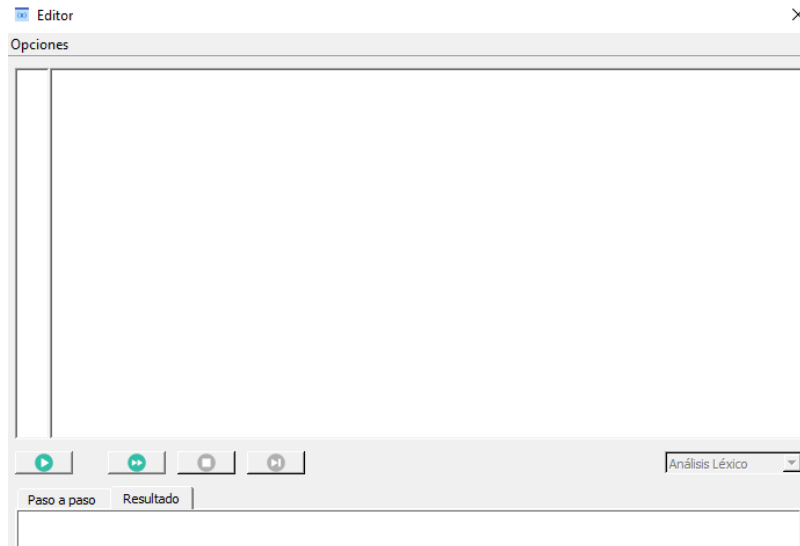
Id	Descripción
1 l	Cualquier carácter alfabéti...
2 L	Cualquier carácter alfabéti...
3 d	Cualquier dígito entre 0 y 9

Presenta una lista de “atajos” de los que dispone el usuario para facilitar la creación de tokens de expresión regular complejos

5. Área de prueba de tokens



Permite desplegar un editor que incluye el analizador léxico definido.



Con este editor el usuario puede probar el funcionamiento de sus analizadores. Además de poder realizar ejecuciones paso a paso que permiten observar el funcionamiento detallado de los mismos.

6. Área de navegación entre analizadores



Consiste en un tab simple para la navegación rápida entre las secciones de definición de analizadores.

7. Área de gestión de proyectos

Archivo	Código	Ayuda
Nuevo...		Ctrl+N
Abrir...		Ctrl+O
Guardar...		Ctrl+S
Guardar como...		Ctrl+A
Salir		Ctrl+Q

Permite acceder a funciones para gestionar el proyecto, además de la función de generación de código y opciones de ayuda.

- **Análisis Sintáctico**

En esta sección el usuario define los elementos para análisis sintáctico a través de procesamiento descendente con gramáticas LL(1). Además, cuenta con una sección de análisis para el usuario, a la que se generaliza como tabla de control.

Archivo Código Ayuda

Analizador Léxico

Analizador Sintáctico

Producciones

Tabla de Control

Lado Izquierdo

Lado Derecho

1

+

Validar Producciones

No Terminales

1

Agregar

Eliminar

Terminales

número

Archivo Código Ayuda

Analizador Léxico

Analizador Sintáctico

Producciones

Tabla de Control

	exp:ATOM	agr:(agr:)	pnto:.	^
<S>	P:1	P:2	Error	Error	Error
pnto:.	Error	Error	Error	PA	Error
agr:)	Error	Error	PA	Error	Error
√	Error	Error	Error	Error	Aceptado

Producciones

1. <S> → exp;

2. <S> → agr:(

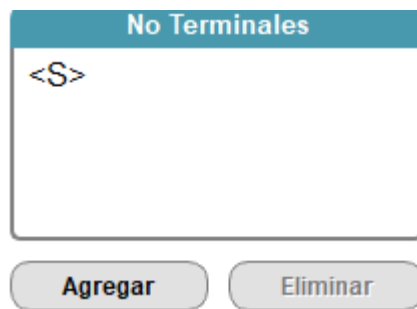
Selecciona un ítem de la tabla o de la lista para obtener información de este

7

Abrir Editor

8

1. Área de definición de no terminales



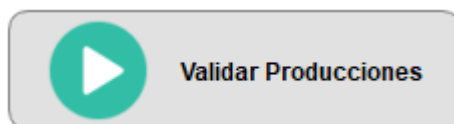
Permite la definición de símbolos no terminales para la construcción de la gramática. Simplemente solicita el ingreso de una cadena como identificador del no terminal.

2. Área de definición de producciones

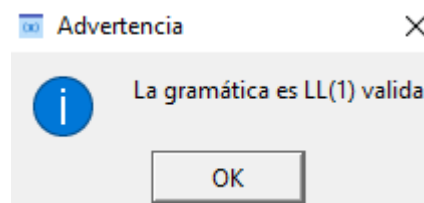


Esta área facilita la creación de producciones, utilizando los no terminales y terminales definidos, a través de funciones de arrastre de elementos.

3. Área de validación de gramática



Con este botón el usuario ejecuta el proceso de validación de la gramática definida, de esta manera verifica que cumpla con las características de una gramática LL(1).



Si la gramática no es válida, se orienta al usuario con el error encontrado.

4. Área de navegación entre zona de definición zona de análisis



Un tab simple que permite a navegación entre la zona para definir la gramática(producciones) y el área para el análisis de la misma (Tabla de control).

5. Área de tabla de control

	exp:ATOM	agr:(agr:)	pnto:.	↵
<S>	P:1	P:2	Error	Error	Error
pnto:.	Error	Error	Error	PA	Error
agr:)	Error	Error	PA	Error	Error
▽	Error	Error	Error	Error	Aceptado

En esta área se presenta la tabla de control generada para la gramática del usuario. Como un elemento que permite analizar el comportamiento de la misma.

6. Área de producciones definidas

Producciones

- <S> → exp:ATOM
- <S> → agr:(<S> pnto:.. <S> agr:)

Indica la lista de las producciones definidas por el usuario y que integran la gramática.

7. Área de análisis de producciones

Explicación de Producción

Selección

Código

Lado Izquierdo

Lado Derecho

Aplicación

<S>

> pnto:.. <S> agr:)

Anulable:

No

re(agr:) <S> pnto:.. <S>) avanza

Al seleccionar celdas de la tabla de control o producciones definidas, despliega información de análisis, como su definición uso de conjuntos selección y código representativo.

8. Área de prueba

Abrir Editor

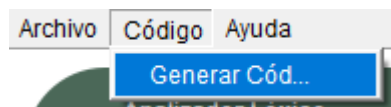
Al igual que en la sección de análisis léxico permite utilizar el editor de la herramienta para probar el funcionamiento de la gramática definida.

- **Generación de código**

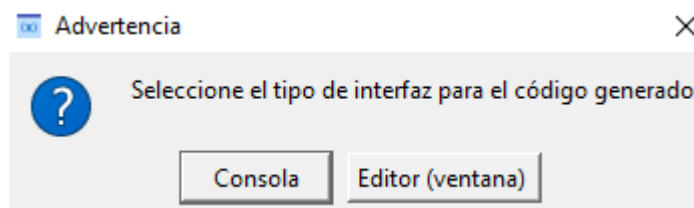
Con las herramientas para la definición de analizadores léxico y sintáctico, el usuario es capaz de crear fácilmente los elementos de análisis de su compilador, en un ambiente que le permite observar las estructuras que está definiendo. Para finalmente obtener un código ejecutable de las estructuras que ha construido.

Para esto se hace uso del módulo generador de código de GADUN.

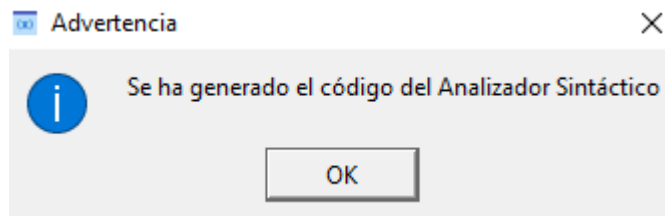
La interacción en la interfaz con este módulo es bastante simple.



A través del área de gestión del proyecto en la sección de código se utiliza la opción de generar código. Con lo que se despliega una serie de mensajes para generar el código.



Selección de método de interacción con los analizadores



Mensaje de notificación del resultado del proceso.

De este modo la herramienta permite la construcción de analizadores de una manera sencilla, brindando al usuario un entorno amigable y orientado al aprendizaje.

La información completa sobre el uso de la herramienta se encuentra diligenciada en el documento “ANEXO_F Manual de usuario” dentro del directorio de ANEXOS.

4.2. Resultados sintetizados de evaluación

Tras la aplicación de síntesis se obtiene las siguientes tablas de resultados:

Tabulación de datos de perspectiva

Visibilidad Y Coherencia				
ÍTEM	Rango 1 (0-1)	Rango 2 (2-3)	Rango 3 (4)	Rango 4 (5)
Visibilidad de proceso y elementos involucrados en el análisis léxico	0,00	0,00	22,22	77,78
Visibilidad de proceso y elementos involucrados en el análisis sintáctico	0,00	0,00	36,11	63,89
Visibilidad de continuidad dentro del proceso de construcción de un	0,00	2,78	38,89	58,33

Comprensión y Análisis				
ÍTEM	Rango 1 (0-1)	Rango 2 (2-3)	Rango 3 (4)	Rango 4 (5)
Análisis de proceso y elementos involucrados en el análisis léxico	0,00	7,78	38,89	53,33
Análisis de conceptos relacionados con gramáticas	0,00	7,41	20,37	72,22
Análisis de conceptos relacionados con procesamiento descendente	0,00	10,19	23,15	66,67
Análisis de conceptos involucrados en los análisis léxicos y sintáctico a través de código	0,00	1,85	22,22	75,93

Apoyo				
ÍTEM	Rango 1 (0-1)	Rango 2 (2-3)	Rango 3 (4)	Rango 4 (5)
Apoyo en la aplicación de conceptos y construcción de analizadores léxicos	0,00	2,78	11,11	86,11
Apoyo en la aplicación de conceptos relacionados con gramáticas y su construcción	0,00	1,85	18,52	79,63
Apoyo en la obtención de bases para la construcción de un semi-compilador	0,00	0,00	27,78	72,22
Apoyo en la apropiación de conceptos a través de código	0,00	0,00	22,22	77,78

Tabulación de datos en escala de Likert

Visibilidad y Coherencia		
ÍTEM	PUNTOS POR ÍTEM	PUNTAJE OBTENIDO
Visibilidad de proceso y elementos involucrados en el análisis léxico	180	172
Visibilidad de proceso y elementos involucrados en el análisis sintáctico	180	167
Visibilidad de continuidad dentro del proceso de construcción de un	180	164

Comprensión y Análisis		
ÍTEM	PUNTOS POR ÍTEM	PUNTAJE OBTENIDO
Análisis de proceso y elementos involucrados en el análisis léxico	450	401
Análisis de conceptos relacionados con gramáticas	270	251

Análisis de conceptos relacionados con procesamiento descendente	540	493
Análisis de conceptos involucrados en los análisis léxicos y sintáctico a través de código	270	256

Apoyo		
ÍTEM	PUNTOS POR ÍTEM	PUNTAJE OBTENIDO
Apoyo en la aplicación de conceptos y construcción de analizadores léxicos	180	174
Apoyo en la aplicación de conceptos relacionados con gramáticas y su construcción	270	258
Apoyo en la obtención de bases para la construcción de un semi-compilador	270	255
Apoyo en la apropiación de conceptos a través de código	180	172

Puntaje total por sección		
Sección	Puntos por sección	Puntaje obtenido
Visibilidad	540	503
Comprensión	1530	1401
Apoyo	900	859

4.3. Propuesta de formulario para cursos siguientes

Para los cursos que abarquen la temática de diseño de compiladores y donde se haga uso de la herramienta GADUN como apoyo al proceso de aprendizaje. Se establecieron ítems para la construcción de un formulario que permita evaluar la perspectiva de los estudiantes con respecto al uso de la herramienta.

Para ello se hizo uso en los ítems presentados por el formulario para validación por parte del curso concluido de diseño de compiladores, incluyendo las secciones de evaluación de visibilidad, interacción y apoyo. En conjunto a estos se proponen las escalas de evaluación. Ya que estos ítems son trasladados desde el formulario presentado para la validación de la herramienta por parte del curso concluido de compiladores, no se ahondará en este documento.

El documento “ANEXO_G Propuesta para formulario de evaluación en cursos siguientes” contiene la información comentada.

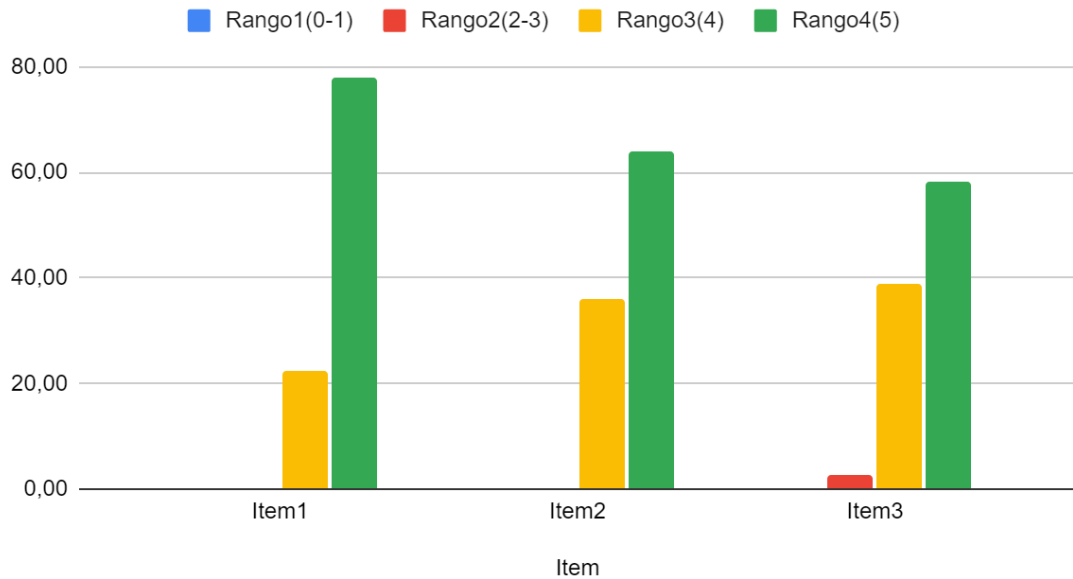
De manera adicional se propuso una sección de integración, que permita evaluar de forma cuantitativa la medida en que los estudiantes han hecho uso de la herramienta.

Variable	Indicador
Apoyo académico	Uso de herramienta para comprobación de ejercicios
Motivación	Uso de la herramienta para la creación de elementos del compilador
Actividad del usuario	Tiempo estimado dedicado al uso de la herramienta
Metodología	Uso de ejercicios didácticos para la apreciación de conceptos.

De esta manera el evaluador puede observar que tan integrado está el uso de la herramienta con la cátedra.

5. ANÁLISIS Y DISCUSIÓN DE LOS RESULTADOS

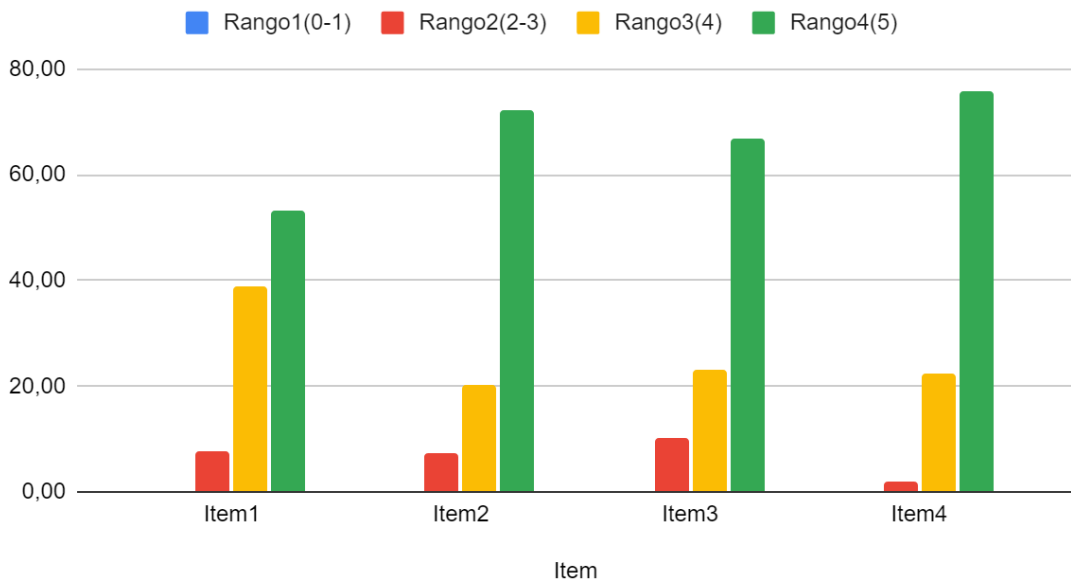
Visibilidad y Coherencia



Para la sección de visibilidad y coherencia se obtiene para los tres ítems perspectivas positivas que superan el 90% lo que denota que la mayor parte de la población percibe que la integración de conceptos a través de elementos visuales fue adecuada, sin embargo los ítems 2 y 3 presentan un alto porcentaje en el rango de posibles elementos mejorables, lo que indica que existe un margen significativo de mejora para la presentación. Por último, en el ítem 3, asociado a la pregunta clave *“Los elementos gráficos de la herramienta permiten visualizar las etapas del proceso de análisis llevado por un compilador desde análisis léxico hasta sintáctico”* que define la trazabilidad de los conceptos dentro de la construcción de un compilador. Existe población que indica una implementación inadecuada en la herramienta.

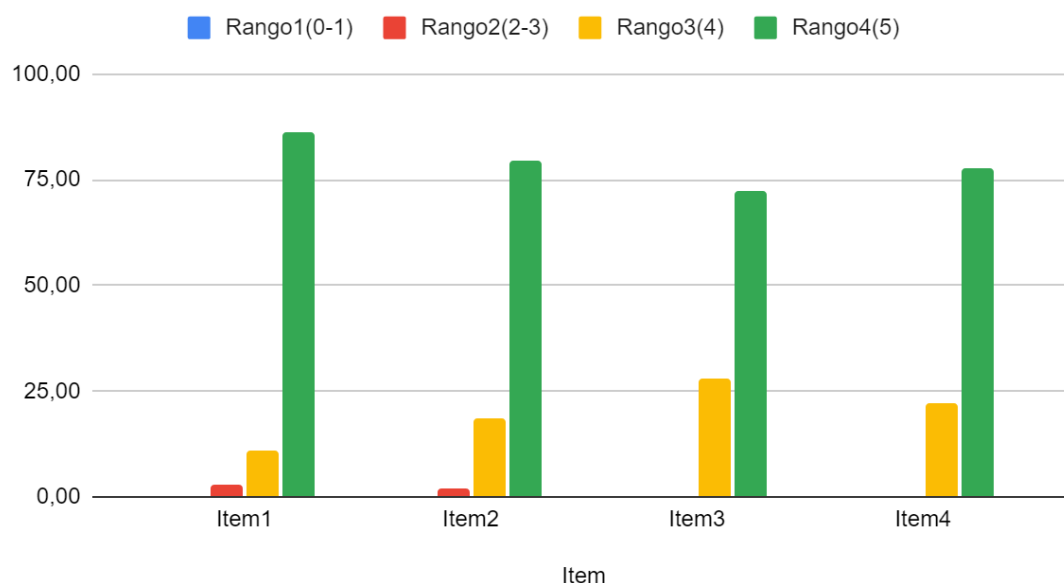
Hecho que puede atribuirse a una necesidad de mayor explicación en cuanto a los conceptos utilizados dentro de la herramienta, o que la limitación de temática abordada por la misma sea percibida como inadecuada para abordar el tema.

Comprensión y Análisis



Para comprensión y análisis existen varios puntos interesantes, primero entre los mayores elementos con puntos a mejorar y perspectivas de implementación inadecuada el Ítem 1 que indica la interacción dada para la sección de análisis léxico a través de tokens. Esto significa que una cantidad significativa de usuarios observó como mejorable la interacción con analizadores léxicos. Por otro lado Ítem 2 y 4, que representan la interacción con gramáticas y código fuente respectivamente, obtienen valores altos de percepción adecuada, indicando que existen perspectivas inadecuadas lo que puede indicar que es necesario aumentar los medios de interacción de estos ítems. y por último el ítem 3 con mayor nivel de perspectivas inadecuadas especifica que es necesario dar tratamiento más amplio al manejo de gramáticas de tipo diferentes a LL(1) para comprender el procesamiento descendente.

Apoyo



Para apoyar la dispersión de la perspectiva es menor. En cuanto a los ítems 1 y 2 relacionados a la construcción de analizadores léxico y sintáctico, respectivamente, demuestran los mayores valores en cuanto a percepción muy positiva, pero en contraste también presentan valoraciones inadecuadas. Esto nos indica que la mayor parte de la población ha encontrado útil el uso de la herramienta para la construcción de estos analizadores, pero que existen elementos para los que se podría presentar falencia, como efecto de los posibles elementos inadecuados percibidos en comprensión y análisis.

Mientras que el ítem 3 y 4 relacionados al uso del código generado, indica percepciones netamente positivas, evidenciando que la población percibe la utilidad de la herramienta como una base para la construcción de un semi compilador.

Rangos	Escala Visibilidad		Escala Comprensión		Escala Apoyo	
Muy adecuado	540		1530		900	
Adecuado	432 - 539	X	1224 - 1529	X	720 - 899	X
Inadecuado	216 - 431		612 - 1223		360 - 719	
Muy Inadecuado	0 - 215		0 - 611		0 - 359	

De acuerdo a la escala de Likert la herramienta obtiene a niveles generales un nivel adecuado en cada uno de los ítems evaluados. Esto significa que la herramienta presenta a través de su interfaz los conceptos teóricos tratados dentro de la asignatura relacionados con el procesamiento descendente y la forma en que estos son tratados y presentados es coherente a su definición en el espacio teórico, además la interacción con los mismos puede ayudar a la comprensión de estos conceptos en un nivel adecuado. y por último que el código generado puede servir como una base fácil de entender para la creación de semi compiladores complejos.

Tomando en cuenta el análisis estadístico y la escala de Likert se puede afirmar que los datos obtenidos en la etapa de validación del estudio apoyan a la hipótesis postulada, presentando una perspectiva positiva por parte del 90% de los estudiantes que cursaron la asignatura de diseño de compiladores, con respecto al empleo de la herramienta que permite la interacción didáctica con la creación de elementos del procesamiento descendente y la cual puede generar el código Python ejecutable de los elementos definidos. Esto no quiere decir que se desconozca los resultados con perspectivas negativas, sino que estos se tomaran como indicadores de características mejorables en la propuesta.

6. CONCLUSIONES

- De acuerdo a los datos obtenidos en los resultados se puede decir que los estudiantes perciben de manera positiva el uso de herramientas software como un apoyo desde la tecnología a la formación profesional. Sin embargo, desde la información obtenida de propuestas externas se debe aclarar que la aplicación de herramientas de apoyo debe estar apoyada por una metodología práctica que integre las necesidades del estudiante, de manera que este sea protagonista de su proceso de formación.
- El uso de una herramienta didáctica que permite la visualización de conceptos teóricos dentro de estructuras tangibles, sin la necesidad de usar un recurso externo complejo, como una librería o un lenguaje. Permite hacer del proceso de aprendizaje en la asignatura de compiladores, un proceso más eficiente, lo que a su vez permite abarcar de mejor manera las temáticas de la cátedra.
- La visualización de una estructura teórica como una gramática LL(1) definida en una máquina de pila a través de código funcional permite romper la barrera entre lo teórico y lo práctico en el diseño de compiladores, afianzando el conocimiento adquirido por el estudiante e impulsando lo al desarrollo de soluciones alternas y métodos propios.
- Se deben establecer límites claros en la funcionalidad proporcionada por la herramienta didáctica de manera que esta facilite la interacción del estudiante con los elementos teóricos de la asignatura de manera justa y suficiente, respetando el rol del estudiante y el docente en el proceso de aprendizaje.
- Existe una diferencia significativa entre los algoritmos utilizados por la herramienta didáctica, y el método de los estudiantes para obtener el conjunto selección siendo los primeros procesos más complejos y extenuantes, por lo que es pertinente confiar al docente la tarea de profundizar y orientar en la adquisición y entendimiento de los mismos.
- Se debe prestar especial atención y cuidado a los elementos gráficos que componen la herramienta; la correcta implementación de estos permite crear un ambiente didáctico en el que el usuario puede hacer mejor uso del software y sus funcionalidades.

7. RECOMENDACIONES

Ya que la herramienta didáctica GADUN se enfoca en el uso de elementos relacionados al procesamiento descendente, varias temáticas abordadas en las cátedras de compiladores o lenguajes formales y autómatas se pueden percibir como incompletas, es por esto que se recomienda para las siguientes investigaciones la ampliación de elementos temáticos de la asignatura.

Python es un lenguaje de programación altamente conocido y utilizado en el espacio de la programación, aun así, las herramientas didácticas se caracterizan por facilitar el proceso de aprendizaje, por lo que para investigaciones relacionadas o continuas a la presente se recomienda la implementación de generación de código a lenguajes diversos como java y C++, o los que se vea prudente a las necesidades de aprendizaje de la población.

El uso de funcionalidades estadísticas que permitan controlar el uso que se ha dado a la herramienta es un apartado muy interesante para las herramientas didácticas, brindando datos que permitan al investigador analizar comportamientos en sus usuarios. Por esta razón se recomienda tomar como una funcionalidad a considerar dentro de la construcción de herramientas de esta índole.

El espacio de aprendizaje es muy variado en cuanto a sus características por lo que puede usarse como un laboratorio para experimentar metodologías nuevas y propuestas para mejorar el proceso de enseñanza. Si bien en esta investigación se hizo uso de ABP debido a sus bondades, se invita a los investigadores y docentes a hacer uso de diversos métodos y herramientas para mantener un flujo de innovación y cambio dentro de los espacios de aprendizaje.

BIBLIOGRAFÍA

GRUNE Dick, REEUWIJK Kees, E.B.H, J.H. Jacobs, LANGENDOEN Koen. Modern Compiler Design. Second Edition. New York: Business Media. 2012. p. 829.

ROJAS Sergio, MORA Miguel. Traductores Y Compiladores Con Lex/Yacc, Jflex/Cup Y Javacc. España: Universidad de Málaga. 2005. p. 319.

LOUDEN Kenneth C. Construcción de compiladores. Principios y práctica. México: Parafino. 2004. p. 582.

AHO Alfred, LAM Monica, SETHI Ravi, ULLMAN Jeffrey. Compiladores, principio técnicas y herramientas. México: PEARSON EDUCATION. 2008. p. 801.

LEWIS II Phillip, ROSEKRANS Daniel, STEARNS Richard. Compiler Design Theory. Massachusetts. Addison-wesley.1976. p. 647.

HURTADO Jacqueline. Metodología de la investigación Holística. Colombia. SYPAL. 2000. p. 664.

MAXWELL Josep. Diseño de investigación cualitativa. GEDISA, Volumen 241006 de Herramientas Universitarias. 2019

GALLARDO Yolanda. APRENDER A INVESTIGAR, MÓDULO 3 RECOLECCIÓN DE LA INFORMACIÓN. Colombia: ICFES. 1999.

COVA Ángela, ARRIETA Xiomara, AULAR Judith, REVISIÓN DE MODELOS PARA EVALUACIÓN DE SOFTWARE EDUCATIVOS. Primera edición. Venezuela: Revista Electrónica de Estudios Telemáticos. 2008. vol. 7.

HERNÁNDEZ Roberto, FERNÁNDEZ Carlos, BAPTISTA Pilar. Metodología de la Investigación. México: MCGRAW-HILL. 2006.

GONZÁLES Antonio, SERRANO J. PEÑARROCHA Ignacio, PÉREZ Emilio. Un sistema para la evaluación del aprendizaje basado en proyectos. Universidad Jaume.2008.

GARCIA Jesús, BERLANGA Antonio. Herramienta didáctica para análisis semántico y traducción de lenguajes formales. Universidad Carlos III de Madrid. 2006.

CABARCAS Johnny. Desarrollo De Un Software Educativo Que Sirva De Apoyo Para El Aprendizaje De La Asignatura Compiladores Del Programa De Ingeniería De Sistemas De La Universidad De San Buenaventura Cartagena. Cartagena. Universidad De San Buenaventura. 2016

ÁVILA Lizeth, ARÍAS Andrés. Realización De Un Prototipo Funcional Aplicando Técnicas De Visualización Para La Construcción Del Árbol Sintáctico Ascendente Y Descendente En La Asignatura De Compiladores. Ecuador. Universidad Pontificia Católica de Ecuador. 2018.

MERNIK Marjan, ZUMER Viljem. IEEE Transactions On Education: An Educational Tool For Teaching Compiler Construction. IEEE. 2003. vol. 46.

PLANES Fernando, PÉREZ Aurora. Herramientas de ayuda para la construcción de compiladores. Universidad politécnica de Madrid. 2020.

CRUZ Raúl, MORENO Luis, MUÑOZ Guillermina y SÁNCHEZ Zindi. GALITN, un generador de analizadores léxicos compilables en C#. Instituto Técnico de Nogales. 2015.

AIKEN Alexander. Cool: A Portable Project for Teaching Compiler Construction. SIGPLAN. 1996.

SEQUEIRA Gladis, ZAJACZKOWSKI Silvia, KORNUTA Cristian, GÓMEZ Gabriela. Estrategias didácticas en el uso de herramientas software para favorecer la comprensión de los alumnos en la enseñanza sobre compiladores. Paraná. 2013.

VEGA Rafael. Compilador de pseudocódigo como herramienta para el aprendizaje en la construcción de algoritmos. UNIVERSIDAD DEL NORTE. 2008.

ARELLANO Jesús, NIEVA Omar, ALGREDO Ignacio. Programación Matemática y Software: Aprendizaje Basado en Proyectos Utilizando L-Systems en un Curso de Compiladores. Oaxaca. Universidad del Istmo. 2013. Vol. 5.

RICO Erick, BUENO Ana, CARPIO José, ROCHA Martha. Metacompileador didáctico generador de código JAVA. México. Jóvenes de la ciencia. 2014.

TONCHE Ronny, GUTIERREZ Alfonso. Diseño y desarrollo de un

compilador visual para la enseñanza de la robótica básica. México. Instituto Politécnico Nacional. 2010

PÁRRAGA Cristina. Compilador didáctico. Argentina. Universidad de Mendoza. 2013

GARCIA Horacio. Reconocedor de lenguajes con base en gramáticas formales. México. Instituto Politécnico Nacional. 2008.

DEMAILE Akim, LEVILLAIN Roland, PERROT Benoit. A set of tools to teach compiler construction. ITICSE. 2008.

VIJAYALASKHMI M., KARIBASAPPA K.G. Activity based teaching learning in formal languages and automata theory - An experience. IEEE. 2012

LI Xu. MARTIN Fred. Chirp on crickets: Teaching compilers using an embedded robot controller. ACM 2006.

Improving Teaching and Learning Computer Programming in Schools through Educational Software

JANCHESKI Metodija. Improving Teaching and Learning Computer Programming in Schools through Educational Software. Universidad Cyril and Methodius. 2017.

ORTIN Francisco, ZAPICO Daniel, CUEVA Manuel. Design Patterns for Teaching Type Checking in a Compiler Construction Course. IEEE. 2007.

ACEBAL César, IZQUIERDO Raúl, CUEVA Manuel. Good design principles in a compiler university course. ACM. 2002.

JONES James. PARTICIPATORY TEACHING METHODS IN COMPUTER SCIENCE. ACM. 1987.

ISLAM Zahurul, KHAN mumit. Teaching Compiler Development to Undergraduates Using a Template Based Approach. Universidad BRAC.2010.

AYALA Gustavo. POSKAL Pablo. GAMESS Eric. SNMP JManager: An Open Source Didactic Application for Teaching and Learning SNMP v1/2c/3 with Support for IPv4 and IPv6. Universidad Central de Venezuela. 2009.

VEGDAHL Steven R. Journal of computings science: Using Visualization Tools To Teach Compiler Design. ACM. 2001. vol. 16.

Recursos Web

ACM, IEEE. Computer Engineering Curricula 2016. p. 151.

GUÍA GENERAL PARA LA ELABORACIÓN DE TRABAJOS DE GRADO.
Ingeniería de Sistemas. Universidad de Nariño