PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

FACULTAD DE INGENIERIA

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

REALIZACIÓN DE UN PROTOTIPO FUNCIONAL APLICANDO TÉCNICAS DE VISUALIZACIÓN PARA LA CONSTRUCCIÓN DEL ÁRBOL SINTÁCTICO ASCENDENTE Y DESCENDENTE EN LA ASIGNATURA DE COMPILADORES

ARIAS TABANGO ANDRÉS SEBASTIÁN

SANTIANA ÁVILA LIZETH CAROLINA

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS Y COMPUTACIÓN

Quito, julio del 2018

DEDICATORIA

Dedico este trabajo principalmente a mis padres y hermano quienes han sido mi motivación y apoyo en esta etapa de mi vida y ser vivo ejemplo de que todo esfuerzo hecho con el corazón vale la pena.

Sebastián Arias

DEDICATORIA

Dedico este trabajo final de la carrera a mis padres por brindarme la mejor educación y por brindarme siempre su apoyo en todos los obstáculos que han presentado durante toda vida, también les agradezco por ser un ejemplo de padres y por mostrar que todo lo que se desee se puede lograr; a mi hermana por siempre estar junto a mí, motivarme cada momento de vida y por ser un ejemplo a seguir.

Carolina Santiana

AGRADECIMIENTO

Agradecemos a la Pontificia Universidad Católica del Ecuador y a los docentes de la Escuela de Sistemas por habernos guiado durante nuestra etapa universitaria y haber despertado esa hambre de conocimiento.

Especial agradecimiento a nuestro director y revisores quienes apoyaron este proyecto de principio a fin con su conocimiento y consejos.

Contenido

INDICE DE TABLAS

INDICE DE FIGURA

RESUMEN

Con el fin de comprender de una manera más simple la construcción del árbol sintáctico ascendente y descendente, se ha realizado este trabajo de titulación para proponer un modelo visual que aporte a la comprensión de varios conceptos y brinde al estudiante la oportunidad de interactuar para poner en práctica los mismos.

Para esto se realizó la revisión sistemática de varias herramientas existentes para obtener información de los puntos clave a mejorarse con el modelo visual. A su vez; se aplicó el test de aprendizaje de Kolb a los estudiantes de los últimos semestres de la Escuela de Sistemas de la PUCE con el fin de determinar el estilo de aprendizaje predominante para así explotar las habilidades que los estudiantes tienen. Además; se realizó una síntesis de las técnicas de visualización, que serían la base del modelo visual; y, finalmente se tomaron ciertas características del modelo propuesto para realizar un prototipo funcional que aplique las mismas en la creación del árbol sintáctico en la materia de compiladores.

Al finalizar el trabajo de titulación, se presentarán las correspondientes conclusiones y recomendaciones que ayudarán para futuras investigaciones e implementaciones de herramientas de software teniendo como base el modelo visual.

ABSTRACT

With the purpose of learning in an easier way the construction of the syntax tree, this work proposes a new visual model that helps the comprehension of multiple concepts and gives the student the chance of interacting with a software to practice his knowledge.

To make this happen, a systematic review of the existing software tools was developed to obtain the main parts where the visual model can help to get a better educative experience. In order to get this, the students of the last semester of the Systems School of PUCE took the Kolb learning techniques test to get the predominant learning style in the faculty and exploit these students' abilities. Also, the visualization techniques were reviewed to have a basic idea for the visual model.

Finally, some features of the visual model were taken to implement a functional prototype that applies them in the creation of the syntax tree process.

You will find the conclusions and recommendations to be considered in further investigations and software implementations using this visual model at the end of this work.

1. CAPITULO I: MARCO TEÓRICO

1.1. INTRODUCCIÓN

Dentro de las asignaturas de Ingeniería en Sistemas, existen varias de ellas que son más difíciles que comprender, ya que los conceptos que se presentan son más abstractos y no tan prácticos como en otras materias. Uno de estos casos es la de compiladores donde los temas, como la creación del árbol sintáctico quedan netamente explicados teóricamente y con prácticas que son desarrolladas con papel y lápiz que muchas veces dificultan más el entendimiento de este tema en concreto.

En la actualidad existen herramientas que ayudan y simplifican la labor de procesar gramáticas para pasar por el analizador léxico y a continuación construir el árbol sintáctico pero el problema es que muchas de ellas como LISA, CUP, LEX, etc. que son muy antiguas lo que implica que conseguirlas sea un reto, su instalación es compleja ya que se debe preparar un ambiente que brinde compatibilidad con versiones antiguas del lenguaje usado y finalmente usarlas es casi imposible por la falta de documentación en algunos casos o por que utilizan sintaxis propias de la herramienta en lugar de usar formatos generales que son normalmente explicados durante las clases.

Estas herramientas tienen un claro enfoque profesional lo que complica la aplicación de estas a trabajos prácticos que ayuden a reforzar el aprendizaje de los estudiantes, como por ejemplo LISA. En su mayoría carecen de visualizaciones, como los es CUP, que no permitan seguir el proceso de aprendizaje de una manera más comprensible y hace aún menos atractivas para el campo educativo y no aportan a la enseñanza de materias complejas como compiladores.

En esta era donde los avances tecnológicos son cada vez más notables y nuevos campos de investigación van apareciendo, el aporte de las Tecnologías de la Información y Comunicación va a ser fundamental para "la creación de nuevos modelos de aprendizaje, nuevos procedimientos y estrategias de búsqueda, organización, procesamiento y utilización de la información" (Nicolalde Rodríguez & Urquiza Fuentes, 2017, p. 2) y estos avances en conjunto con la experiencia de los docentes creará una nueva era en la que las clases impulsen a los estudiantes a aprender.

Por estos motivos, este trabajo de titulación parte con la revisión de algunas herramientas existentes como LISA, CUPV, VCOCO, EvDebugger, entre otras, las cuales se usan en el campo de compiladores, tomando en cuenta variables que serán enumeradas y explicadas más adelante para así determinar los posibles campos de mejora en los que se pueden aplicar técnicas de visualización. Para esto además se realizará una encuesta para determinar el estilo de aprendizaje según Kolb que predomina en los estudiantes de ingeniería para así elegir la técnica de visualización que mejor se adapte a las capacidades de los estudiantes.

1.2. JUSTIFICACIÓN

Durante varios años la enseñanza se ha visto afectada por la tecnología, ya que esta ha facilitado a los maestros el plasmar ideas y conceptos de lo que desean enseñar a sus estudiantes. En la actualidad; con el fin de llamar la atención de los estudiantes, se cuenta con herramientas de software que permiten hacer llamativos los diferentes temas para lograr que la atención de los estudiantes se pueda mantener por algún tiempo; y, a la vez ayuda a despertar su curiosidad en los temas que están siendo expuestos.

Pero estas herramientas cumplen su propósito en cuanto a temas teóricos; en el caso de la materia de compiladores que requiere de una parte práctica que refuerce la teoría, solamente se cuenta con herramientas que datan de los años 80 o 90; que tienen salidas con código, lo cual muchas veces es incomprensible para los estudiantes ya que requiere un conocimiento más especializado y no aportan para un fin didáctico que enriquezca los conocimientos y solamente causan frustración durante el proceso de aprendizaje.

Por los motivos expuestos, el presente proyecto analizará las diferentes técnicas de visualización que existen en la actualidad con el fin de determinar cuál o cuáles son las más adecuadas para generar un modelo de visualización de los árboles sintácticos ascendentes y descendentes para crear un software que haga que la experiencia de los estudiantes sea más placentera y que aporte a su aprendizaje de una manera interactiva que ayude a aplicar los conocimientos teóricos y resuelva dudas en cuanto a las prácticas que se realicen en la materia de compiladores.

Se debe recalcar que este análisis puede ser tomado como base para nuevos temas de investigación relacionados a las diferentes técnicas de visualización aplicadas al aprendizaje; ya que, es un tema que está creciendo con gran fuerza gracias a la popularidad de las plataformas E-Learning.

1.3. PLANTEAMIENTO DEL PROBLEMA

Actualmente los estudiantes que toman la materia de compiladores ven sus herramientas de aprendizaje limitadas a los apuntes de la clase, material de apoyo proporcionado por el docente y libros que, en lugar de aportar al entendimiento de la materia, generan confusión. Estos no consideran la opción de utilizar una herramienta de software que genere árboles sintácticos

porque son difíciles de conseguir y están estigmatizadas como difíciles de usar por sus interfaces complejas de entender y que son poco amigables hacia el usuario.

Este problema se da porque las herramientas como ANTLR, BISON, CUP o LISA fueron pensadas para personas que tienen claro los conceptos relacionados a los compiladores y sobre todo no fueron creadas como una herramienta educativa por lo que sus outputs son poco comprensibles; a menos que, sean interpretados por un técnico especializado. Por lo indicado; estas no pueden ser utilizadas de manera óptima en el aprendizaje, porque frustran a los estudiantes, convirtiéndolas en una desventaja en el proceso educativo.

Estos motivos incentivan a plantear soluciones para mejorar esta experiencia educativa, teniendo como principales campos al diseño del software y a la visualización de los arboles sintácticos. El primer campo se centra en cómo crear un software fácil de utilizar por cualquier estudiante de la materia de compiladores o interesado en aprender sobre estos; mientras que el segundo, se especializa en la mejora de la experiencia visual en la creación de los árboles sintácticos.

En conclusión, se tiene que las herramientas de software que existen en la actualidad no suplen las necesidades que los estudiantes tienen al momento de estudiar de manera práctica la creación de los arboles sintácticos porque no fueron creadas con la finalidad de ser herramientas de aprendizaje; siendo esta la problemática a resolver, teniendo en cuenta los dos campos antes explicados.

1.4. OBJETIVOS

1.4.1. OBJETIVO GENERAL

Realizar un prototipo funcional, aplicando técnicas de visualización de algoritmos y programas, para la construcción del árbol sintáctico ascendente y descendente en la asignatura de compiladores que facilite el aprendizaje del mismo.

1.4.2. OBJETIVOS ESPECIFICOS

- Realizar una revisión sistemática de las técnicas de visualización para la construcción del árbol sintáctico ascendente y descendente.
- Describir de forma correcta los parámetros que se van a estudiar para la visualización de algoritmos.
- Describir de forma correcta los parámetros que se van a estudiar de cada una de las herramientas existentes.

1.5. REVISIÓN SISTEMÁTICA DE HERRAMIENTAS EXISTENTES

Para que la revisión de las herramientas sea fructífera y aporte a este trabajo de titulación se definieron las siguientes variables:

1. **Escritura de las gramáticas:** Conocer si la herramienta usa una notación estándar como es el caso de BNF (Backus-Naur Form) o EBNF (Extender Backus-Naur Form) como metalenguaje para expresar las gramáticas. Esto es importante tomando en cuenta que la gramática es el punto de inicio de un compilador.

2. **Creación del Árbol Sintáctico:** Esta variable ayuda a determinar si la herramienta es capaz de crear una representación gráfica (visualización) del árbol sintáctico que represente a la sentencia a ser evaluada.
3. **Ambiente de Uso:** específica el campo en el que la herramienta es utilizada, puede ser profesional o educativa.
4. **Tipo de Visualización:** Clasifica a las visualizaciones provistas por la herramienta como dinámicas (animaciones o tiempo real) y Estáticas (imágenes después de ejecución).
5. **Sincronía de la Visualización:** Determina el nivel de concordancia entre la ejecución de la gramática y las visualizaciones creadas.
6. **Disponibilidad:** Se refiere a la facilidad de conseguir la herramienta, es decir una versión ejecutable para evaluar y también si existe documentación de esta.

Se han considerado distintas herramientas que se han dividido en tres grupos, el primer grupo está conformado por LISA, CUPV, VCOCO y EvDebugger que han sido las escogidas por (Nicolalde Rodríguez & Urquiza Fuentes, 2017)en *"Revisión de herramientas de visualización de la Traducción Dirigida por Sintaxis"* que fue planteada para los traductores dirigidos por la sintaxis pero integran el análisis sintáctico y por ende la creación del árbol de análisis sintáctico. En el segundo grupo se han considerado herramientas de creación de compiladores que sean relativamente actuales considerando que se tenga acceso a documentación y a la herramienta y estas son BisonC++, Coco/R, Javacc y Jaccie. Finalmente se evaluaron tres herramientas online que a partir de una expresión escrita en notación de corchetes por niveles crean una visualización de árbol sintáctico. Este último grupo aporta un concepto previo de lo que se puede conseguir con este trabajo de titulación.

Antes de comenzar con el análisis de cada una de las herramientas, hay que acotar que se ha realizado esta revisión guiándose en la metodología propuesta por (Webster & Watson, 2002).

1.5.1. Grupo Uno

1.5.1.1. LISA

Como se describió en 2006 en la página web *"Intro to LISA"*, LISA es un sistema que genera automáticamente un compilador/interprete a partir de las gramáticas especificadas por el lenguaje. Proporciona "generadores de Árbol Léxico (AL), generadores de Árbol Sintáctico (AS) […], herramientas gráficas" (Nicolalde Rodríguez & Urquiza Fuentes, 2017, p. 4) y muestra las gramáticas en notación BNF que previamente deben ser especificadas en un lenguaje formal. Dada la complejidad de su uso, hace notar que es para uso profesional mas no educativo mientras que se puede considerar que la sincronización de visualización es parcial ya que si permite ver el árbol sintáctico del lenguaje ejecutado, pero al no existir interacción con el usuario también se considera la visualización como estática a pesar de tener una opción de animación. A pesar de ser creada con el fin de generar automáticamente compiladores, permite observar de manera gráfica las tareas realizadas por el analizador sintáctico a partir de una animación del autómata que está siendo utilizado. La generación gráfica del árbol sintáctico se realiza una vez se haya compilado y ejecutado el programa y se muestra como en la figura 1.1. LISA da la opción de animar a esta visualización de modo que se puede observar de qué manera se creó el árbol sintáctico; pero esta animación no se ejecuta en conjunto con el código por lo que no se sabe que reglas fueron aplicadas para llegar al resultado final. De la animación solo se puede configurar el tiempo de transición de nodo a nodo.

Esta herramienta se encuentra disponible tanto para descargar una copia totalmente funcional, en el sitio web también se incluye documentación y videos de ejemplo que ayudan a su utilización.

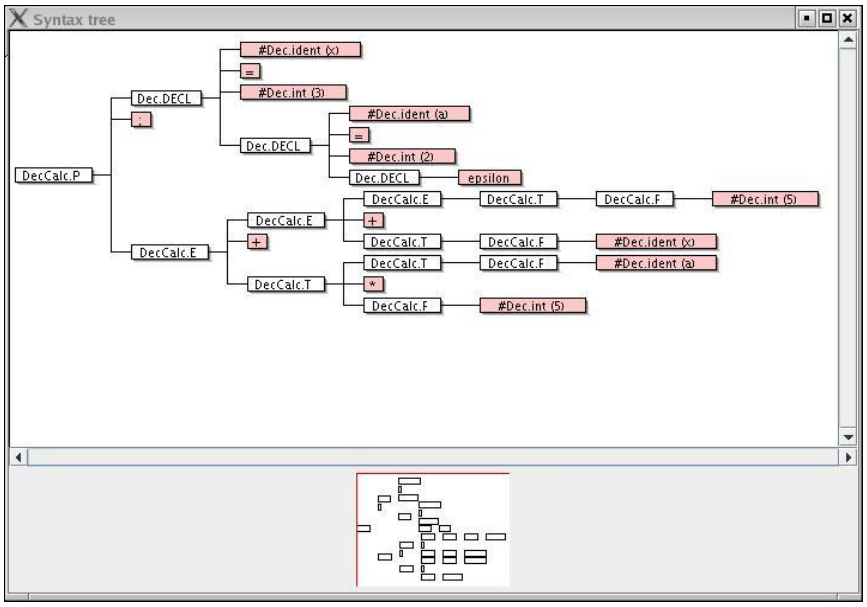


Figura 1.1. Visualización del árbol sintáctico presentada por LISA.

La opción que LISA posee una graficación del árbol sintáctico es de ayuda para ejecutar pruebas sobre un lenguaje que está siendo probado en cada ejecución ya que da una idea concreta de lo que está pasando en el analizador sintáctico y con la animación se puede interpretar las reglas que están siendo aplicadas en cada paso, pero como notamos todo está pensado en la proactividad y facilitar el trabajo a un profesional que tiene claro que es lo que va a ocurrir o por lo menos sabe a qué punto quiere llegar, siendo esa necesidad de conocimiento previo sobre el analizador sintáctico una desventaja de LISA para el ambiente educativo.

1.5.1.2. CUPV

Esta herramienta nace como extensión de CUP por lo que hereda entre otras características de crear el Árbol Sintáctico (AS) pero no posee la capacidad de realizar un análisis léxico por su propia cuenta por lo que el trabajo de Árbol Léxico (AL) debe ser delegado a otras herramientas que posean esta capacidad.

(Nicolalde Rodríguez & Urquiza Fuentes, 2017) afirman que "Además, su diseño fue pensado para ayudar la comprensión de la ejecución del analizador por parte de los estudiantes permitiéndoles visualizar varios aspectos críticos del proceso de compilación, por lo que el ámbito de uso de esta herramienta es educativo." (p.6), teniendo así la primera herramienta que fue creada con fines educativos que se revisó en este trabajo de titulación; por este motivo, presenta la visualización de algunos aspectos del compilador como "los valores semánticos, las reducciones realizadas por el analizador, los estados el analizador, el árbol sintáctico y la entrada" (Nicolalde Rodríguez & Urquiza Fuentes, 2017, p. 6) haciendo que el proceso de compilación de un lenguaje sea mucho más comprensible hacia los estudiantes.

CUPV utiliza el formato BNF como notación para la especificación de las gramáticas, en cuanto a la visualización ofrece la creación del árbol sintáctico por jerarquías y de una manera estática. La sincronización de la visualización con la ejecución del lenguaje compilado es parcial ya que muestra al lado de la visualización del AS la entrada a ser probada.

En cuanto a la documentación no se encontró manuales de CUPV, pero sí de CUP, y al ser una extensión se podría pensar que estos pueden ser de utilidad. En cuanto al software (Nicolalde Rodríguez & Urquiza Fuentes, 2017) afirman que "no han podido encontrar una versión del software accesible para poder ejecutarla" (p.8).

Al no encontrar la herramienta para su revisión, nos basamos en CUP para revisar la visualización del árbol sintáctico que esta provee. Esta visualización es muy básica ya que toma un XSLT generado por la herramienta y simplemente da una visualización generada en HTML y CSS como se muestra en la figura 1.2. Nuevamente nos encontramos con una visualización que no es interactiva con el estudiante y que solo aporta información para una posible revisión de que la gramática está siendo aplicada de manera correcta.

A pesar de ser construida pensada en un ambiente educativo, CUPV no entrega una visualización que aporte conocimiento de una manera interactiva al estudiante sobre el árbol sintáctico, ya que resulta ser simplemente informativa.

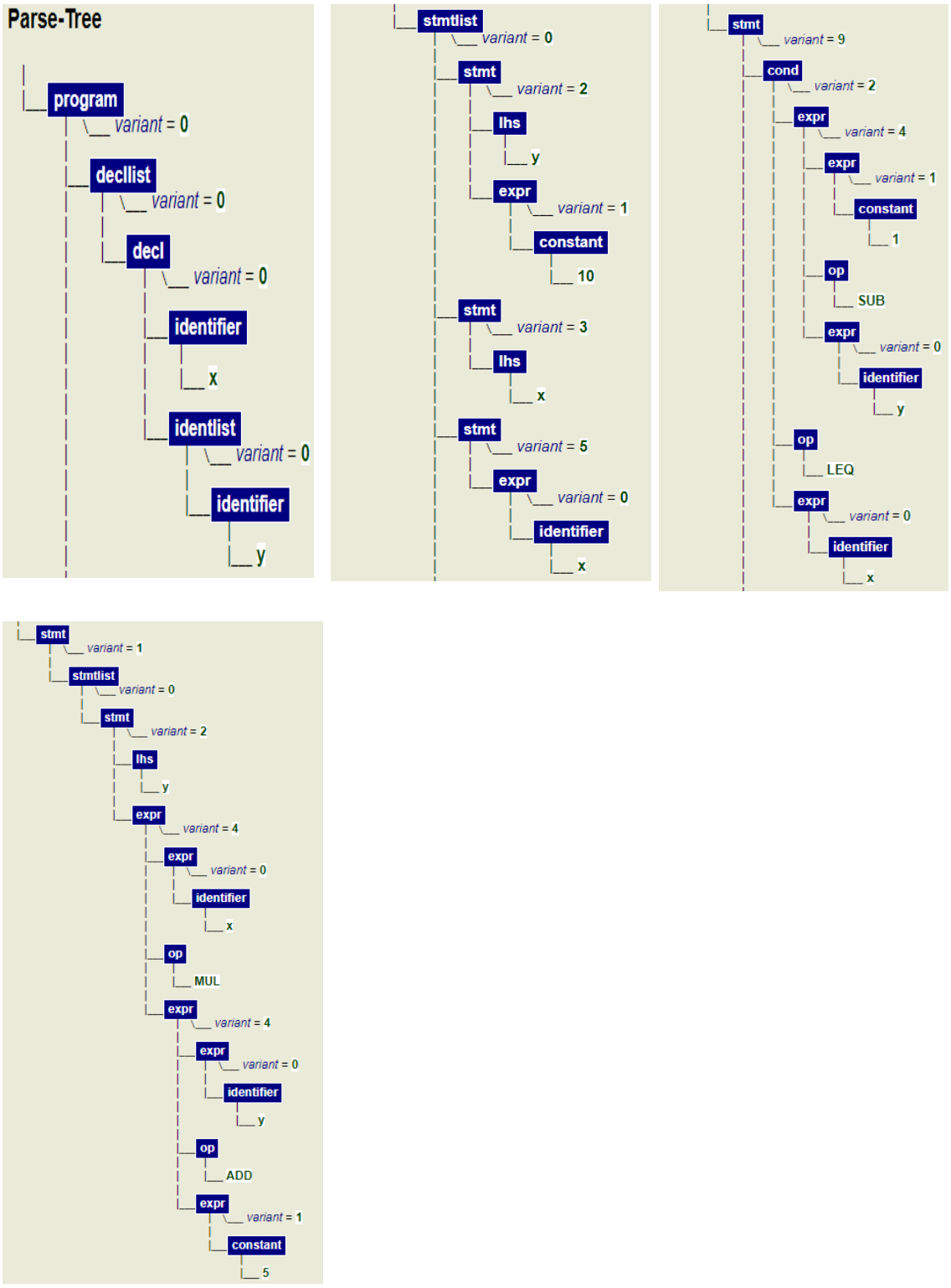


Figura 1.2. Visualización del árbol sintáctico presentada por CUP.

1.5.1.3. VCOCO

Es creada como extensión de COCO/R y similar al caso de CUPV, VCOCO hereda la característica de Árbol Léxico (AL) y Árbol Sintáctico (AS). La notación que utiliza es de formato EBNF para la especificación de las reglas del AS y fue creada para que los estudiantes puedan "ver el funcionamiento de un compilador como si se tratara del código de cualquier otro programa que se estuviera depurando" (Nicolalde Rodríguez & Urquiza Fuentes, 2017, p. 9) por lo que se ubica en un ambiente de uso educativo. VCOCO no crea el árbol sintáctico de una manera gráfica por lo que la sincronía entre visualización y ejecución es nula y la variable de tipo de visualización no se cubre por la herramienta.

En cuanto a la disponibilidad de la herramienta y su documentación (Nicolalde Rodríguez & Urquiza Fuentes, 2017) mencionan que no han encontrado una versión del software para ejecutar ni su documentación.

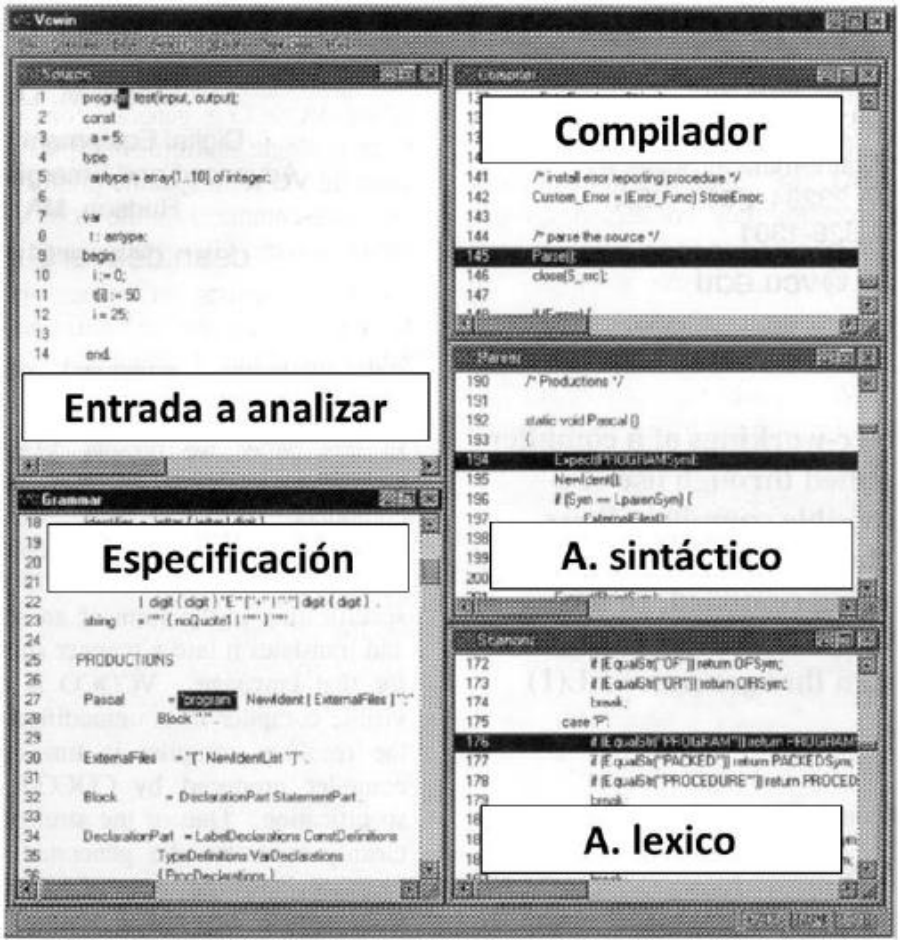


Figura 1.3. Elementos visualizados con VCOCO. Por (Resler & Deaver, 1998).

VCOCO a pesar de considerarla apta para el ambiente educativo, esta herramienta no aporta una visualización del árbol sintáctico ya que está centrada en la ejecución del compilador y los pasos que cumple en cada una de sus etapas a manera de código.

1.5.1.4. EvDebugger

De todas las herramientas revisadas por (Nicolalde Rodríguez & Urquiza Fuentes, 2017), es la primera que no utiliza un formato estándar sea BNF o EBNF, EvDebugger utiliza un formato propio para definir las gramáticas. Esta herramienta está orientada a la educación porque posee

un depurador visual que tiene como objetivo ayudar a los estudiantes a diseñar sus propios procesadores de lenguajes, esto también demuestra que tiene una sincronía entre ejecución del lenguaje y visualización completa ya que muestra las gramáticas aplicadas para crear el árbol de una forma dinámica. A pesar de los esfuerzos realizados (Nicolalde Rodríguez & Urquiza Fuentes, 2017) afirman no haber encontrado una versión de este software para ejecutar; y en cuanto a la documentación tampoco se halló manuales ni nada parecido.

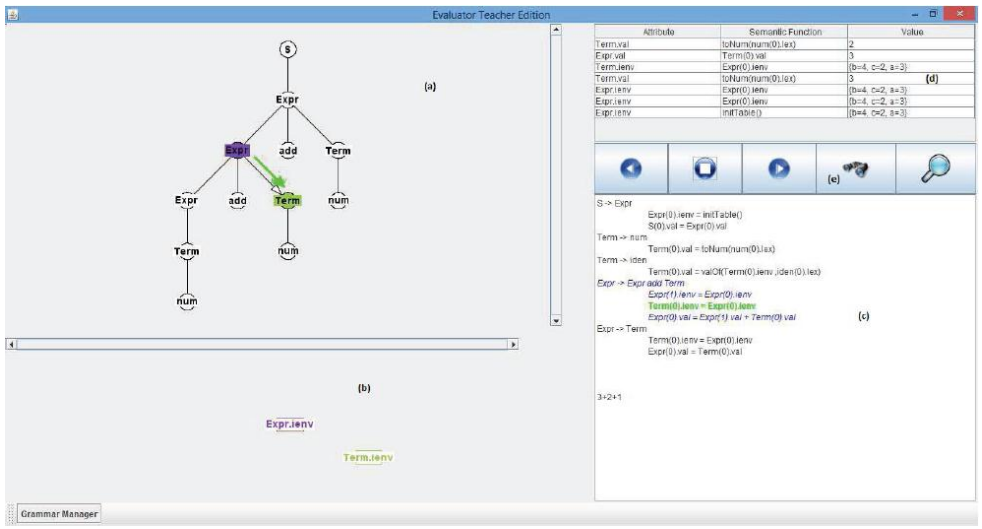


Figura 1.4. Herramientas del depurador visual. Por (Rodríguez-Cerezo, Enriquez, & Sierra, 2014).

Como se puede observar en la figura 1.4, EvDebugger presenta una sincronía completa de la visualización con las gramáticas aplicadas, ya que el proceso de creación del árbol sintáctico puede ser ejecutado paso por paso donde la herramienta explica las reglas aplicadas en el proceso para construir el árbol mientras flechas de colores guían al usuario en el orden en que los diferentes nodos van apareciendo de una manera didáctica y que aporta al estudiante; pero

su falta de disponibilidad es una falencia grave ya que no permite explotar su potencial en las aulas.

1.5.2. Grupo Dos

Estas herramientas fueron elegidas ya que mantienen actualizándose constantemente y tienen versiones descargables para ser ejecutadas y su documentación disponible. Por lo que la variable de disponibilidad es cumplida por todas ellas.

1.5.2.1. BisonC++

BisonC++ es un generador de analizadores de propósito general, convierte las gramáticas libres de contexto del tipo LALR en clases en C++. Trabaja con el formato BNF para las gramáticas, pero no es capaz de crear el árbol sintáctico de manera visual así que la sincronía entre la ejecución del lenguaje de prueba y la visualización es nula. Por la manera en que trabaja denota que fue creado para un ámbito profesional más no educativo.

Al ser creado con un fin profesional, su aplicabilidad está enfocada en crear clases de C++ que se encarguen del análisis sintáctico, desde interpretar lenguajes básicos utilizados por una calculadora hasta lenguajes complejos de programación. En su sitio de GitHub explícitamente indican que es indispensable tener un conocimiento bueno de C++, lo que es un prerrequisito que puede causar inconvenientes. No cuenta con ningún tipo de visualización interactiva que ayude a los estudiantes por lo que no se puede destacar ninguna función gráfica en cuanto a la construcción del árbol sintáctico.

1.5.2.2. COCO/R

Es un generador de compiladores que toma los atributos gramaticales descritos en formato EBNF y que sigue el formato LL(k), al igual que BisonC++ no realiza la construcción del árbol sintáctico de una manera visual, por lo que no cumple con las variables de tipo de visualización y sincronía de visualización, pero las salidas que entrega pueden ser utilizadas para construirlo como un módulo independiente, inclusive su documentación afirma que COCO/R ha sido utilizado con este fin. Es una herramienta que está diseñada para el uso profesional y fines investigativos.

Esta herramienta al ser creada pensando en un ambiente profesional no da las facilidades para que un estudiante pueda visualizar e interactuar con el árbol sintáctico de una manera didáctica. También tiene como desventaja que, al ser pensada solo para profesionales o investigadores, su sintaxis es algo compleja de aprender a pesar de tener una documentación extensa, en la figura 1.4 se puede apreciar la sintaxis que utiliza para crear reglas de una gramática.

```
Cocol =
  {ANY}              // using clauses in C#, import clauses in Java,
                     // #include clauses in C++
  "COMPILER" ident
  {ANY}              // global fields and methods
  ScannerSpecification
  Parserspecification
  "END" ident '.'.
```

```
ScannerSpecification =
  ["IGNORECASE"]
  ["CHARACTERS" {SetDecl}]
  ["TOKENS"  {TokenDecl}]
  ["PRAGMAS" {PragmaDecl}]
  {CommentDecl}
  {WhiteSpaceDecl}.
SetDecl        = ident '=' Set.
Set            = BasicSet {('+'|'-') BasicSet}.
BasicSet       = string | ident | char [".." char] | "ANY".
TokenDecl      = Symbol ['=' TokenExpr '.'].
TokenExpr      = TokenTerm {'|' TokenTerm}.
TokenTerm      = TokenFactor {TokenFactor} ["CONTEXT" '(' TokenExpr ')'].
TokenFactor    = Symbol
               | '(' TokenExpr ')'
               | '[' TokenExpr ']'
               | '{' TokenExpr '}'.
Symbol         = ident | string | char.
PragmaDecl     = TokenDecl [SemAction].
CommentDecl    = "COMMENTS" "FROM" TokenExpr "TO" TokenExpr ["NESTED"].
WhiteSpaceDecl = "IGNORE" (Set | "CASE").
```

```
ParserSpecification = "PRODUCTIONS" {Production}.
Production  = ident [Attributes] [SemAction] '=' Expression '.'.
Expression  = Term {'|' Term}.
Term        = [[Resolver] Factor {Factor}].
Factor      = ["WEAK"] Symbol [Attributes]
            | '(' Expression ')'
            | '[' Expression ']'
            | '{' Expression '}'
            | "ANY"
            | "SYNC"
            | SemAction.
Attributes  = '<' {ANY} '>' | "<." {ANY} ".>".
SemAction   = "(." {ANY} ".)".
Resolver    = "IF" '(' {ANY} ')' .
```

Figura 1.5. Sintaxis de COCO/R. Por (Mössenböck, s. f.)

Al no poseer ninguna salida gráfica del árbol sintáctico, no se puede considerar una opción viable para aplicar en un campo educativo que fortalezca al entendimiento de la construcción del árbol sintáctico.

1.5.2.3. JAVACC

Escrito totalmente en Java, es un generador de analizadores que produce su código totalmente en Java. Utiliza el formato EBNF como notación para las gramáticas a ser usadas y en su página oficial dice ser el compilador el generador de analizador sintáctico más utilizado en proyectos

realizados con Java. Por si solo no construye el árbol sintáctico de una manera visual causando que la sincronía entre ejecución de lenguaje y visualización sea nula, pero tiene un complemente denominado JJTree que es capaz de construir el árbol sintáctico de forma gráfica utilizando una representación de niveles como se muestra en la figura 1.6, lo que hace que el tipo de visualización sea estática. Posee la característica de reportar errores del lenguaje en ejecución y generar documentación del analizador creado; tiene un enfoque profesional, pero al estar escrito en un lenguaje popular como Java puede ser utilizado por estudiantes con una investigación previa de la herramienta.

El complemento JJTree en efecto puede construir el árbol sintáctico de una manera que puede considerarse gráfica, pero sigue siendo muy básica como para considerarlo una ayuda al aprendizaje de la construcción del árbol sintáctico en compiladores, por lo que no es un aporte al ambiente educativo. En la figura 1.6 se muestra la salida que entrega JJTree en la construcción del árbol sintáctico de un lenguaje ejecutado.

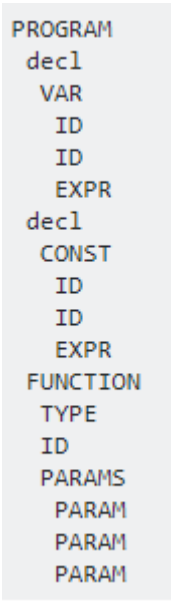


Figura 1.6. Árbol sintáctico construido por JJTree.

Como se observa, el árbol creado por JJTree entrega la información ordenada por niveles, de una forma que es útil para su revisión, pero no aporta ninguna interacción o explicación de cómo fue creado que pueda ser útil para los estudiantes.

1.5.2.4. JACCIE

Jaccie es un compilador de compiladores basado en Java, que presenta un ambiente interactivo para el usuario en cada una de las etapas de análisis léxico y sintáctico. En cuanto al analizador sintáctico es capaz de crear parsers de tipo LL(1), LR(0), SLR(1), LALR(1) y LR(1) utilizando gramáticas libres de contexto. Dentro de la interfaz presenta editores específicos para cada etapa del compilador que son de gran ayuda para la definición correcta de las diferentes reglas que se vayan a utilizar en la gramática del lenguaje como se muestra en la figura 1.7 y 1.8.

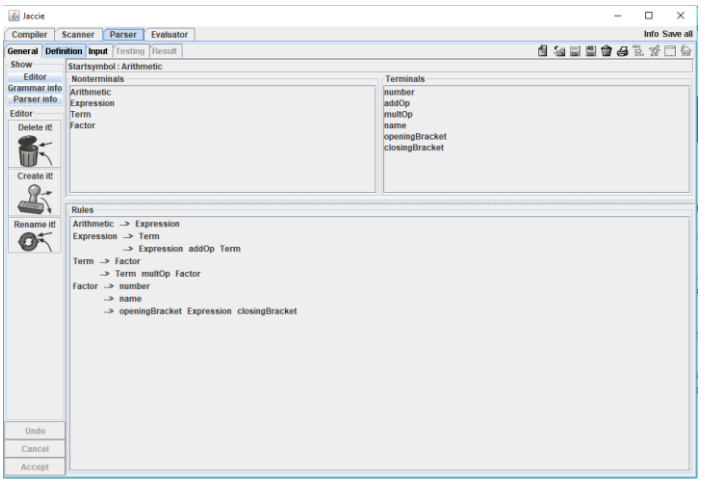


Figura 1.7. Editor analizar sintáctico de JACCIE.

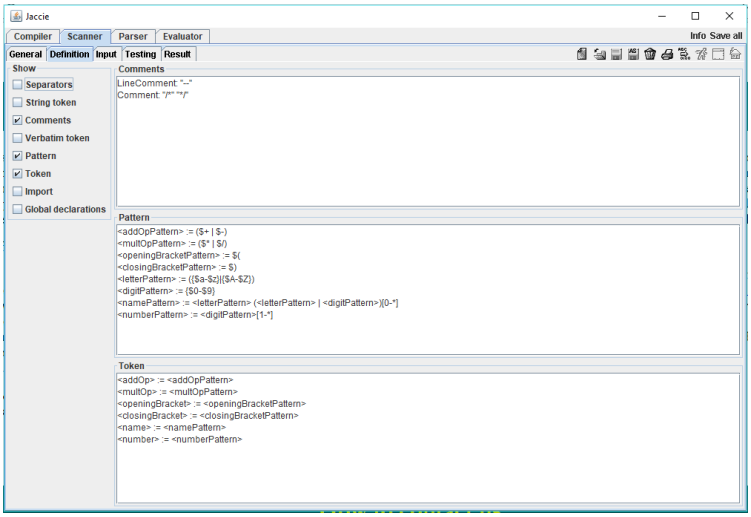


Figura 1.8. Editor analizar léxico de JACCIE.

JACCIE crea el árbol sintáctico de una manera gráfica totalmente interactiva, teniendo una visualización dinámica y una sincronía completa, permite al usuario interactuar con la creación del árbol sintáctico paso a paso o generarlo de manera completa y explicando las gramáticas aplicadas para su construcción y los estados en los que se encuentra el proceso como se muestra en la figura 1.9.

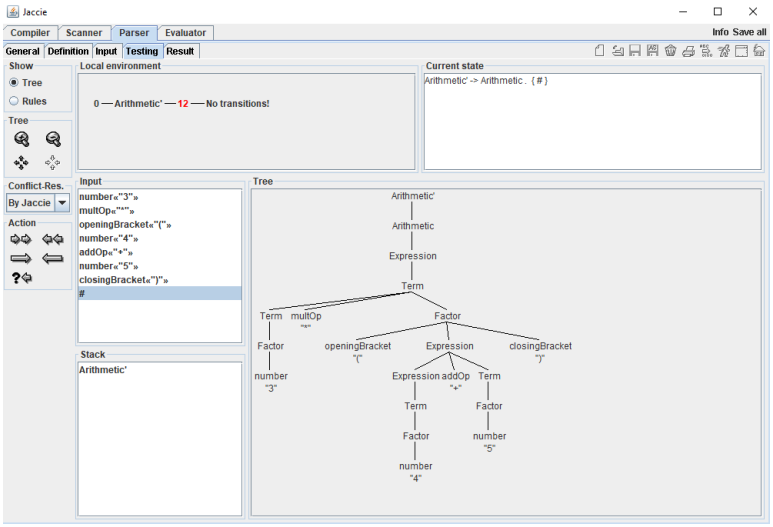


Figura 1.9. Representación gráfica del árbol sintáctico de JACCIE.

Como característica extra, JACCIE permite exportar código Java de todo el compilador creado para que pueda ser utilizado en otro proyecto.

1.5.3. Grupo Tres

Este grupo fue tomado en cuenta ya que posee la capacidad de creación y presentación visual de un árbol sintáctico, pero en contraparte las tres herramientas a ser analizadas no utilizan un formato estándar, sea BNF o EBNF, sino que han sido desarrolladas para usar una notación de corchetes por niveles.

También se determinó que estas herramientas caben perfectamente tanto en el campo educativo como en el profesional ya que son fáciles de utilizar y dan una visualización útil y no se necesitan preparar ambientes específicos para usarlas, solo se necesita un navegador. A continuación, se describirán.

1.5.3.1. mshang.ca/syntree

Esta herramienta posee una visualización 100% sincronizada con la ejecución del lenguaje a ser evaluado, una característica óptima para la comprensión del árbol sintáctico. Este está disponible online y su código y documentación están subidos en GitHub. Al utilizar notación en corchete es fácil de utilizar siempre y cuando la estructura del árbol a construir este clara por quien utiliza la herramienta. A continuación, un ejemplo:

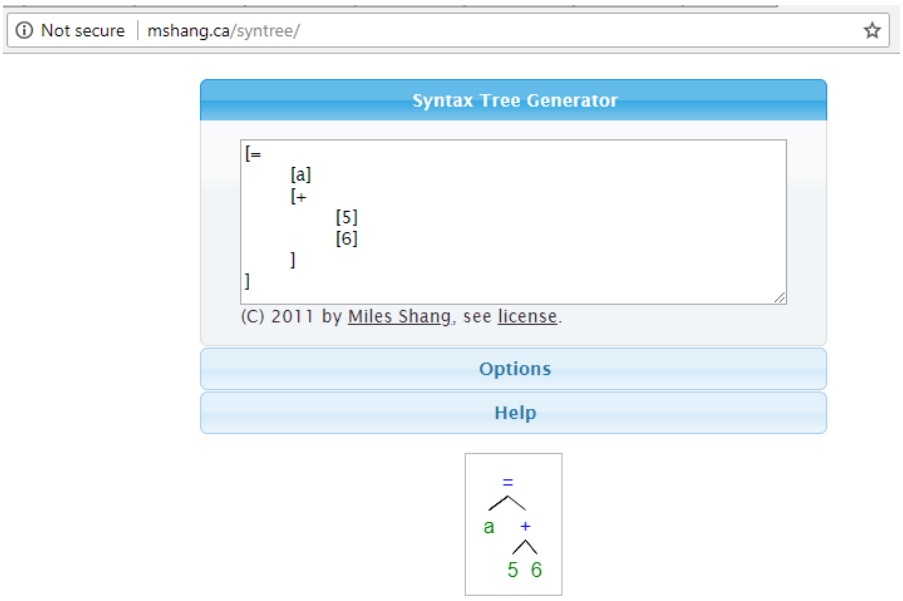


Figura 1.10. Estructura de árbol creada por mshang.ca/syntree/

Como se aprecia en la figura 1.10, esta herramienta es capaz de generar de manera gráfica un árbol sintáctico, pero solo aporta la visualización no tiene ningún tipo de analizador de gramática siendo esta su principal desventaja al no brindar una experiencia completa.

1.5.3.2. ironcreek.net/phpsyntaxtree

Al igual que la anterior herramienta esta también es online y permite crear el árbol sintáctico, pero solo una vez que la expresión escrita en la notación de corchetes haya sido terminada, por lo que la sincronización entre visualización y ejecución del código es parcial. En este caso la herramienta se encuentra disponible en línea y no se dispone de más documentación que un pequeño apartado en la página web, en cuanto a su código al día de hoy parece haber sido eliminado por el creador ya que no se encuentra en su respectivo repositorio de Google Code lo que dificulta su utilización como parte de algún otro proyecto.

Como ocurre en la herramienta anterior, no se tiene una experiencia completa en cuanto al análisis de la gramática, en la figura 1.11 se muestra la estructura de árbol que esta herramienta propone.

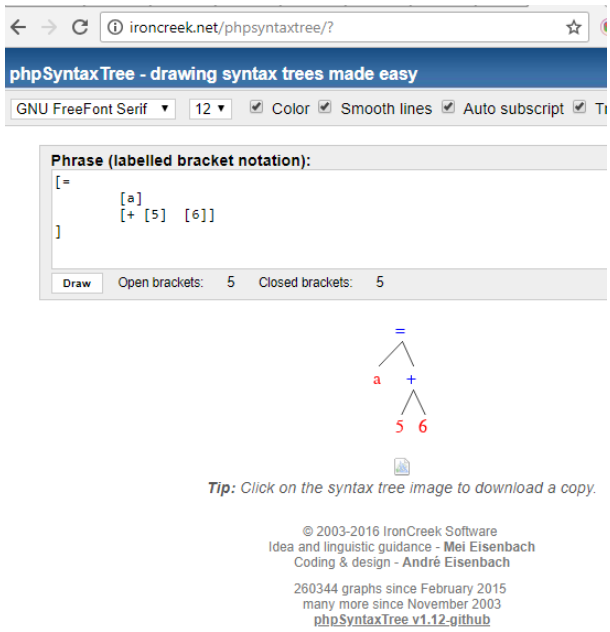


Figura 1.11. Estructura de árbol creada por ironcreek.net/phpsyntaxtree

Una de las ventajas de esta herramienta es que permite exportar la estructura del árbol creado.

1.5.3.3. yohasebe.com/rsyntaxtree

Esta herramienta también cumple con la creación del árbol sintáctico una vez se haya completado de escribir la sentencia en la notación de corchetes, hay que agregar que la escritura de las sentencias en esta herramienta es más cómoda ya que utiliza un editor de texto para código. Coincidiendo con la anterior herramienta tiene una sincronización entre la visualización y ejecución del código parcial, pero añade algo útil al momento de escribir el código, que es un revisor de "gramática" que revisa si los corchetes están siendo utilizados de manera correcta, algo muy útil que hace la diferencia en este grupo de herramientas revisadas. Está disponible para ser usada y su documentación está disponible en GitHub.

Como se observa en la figura 1.12, esta herramienta tiene muchas personalizaciones para la visualización del árbol como personalizar los conectores y sus alturas, el tipo de letra y su tamaño entre otros detalles que son útiles al momento de exportar la imagen como PNG, PDF, SVG o directamente en GYAZO. Lo que nos hace interpretar que esta herramienta está perfilándose al futuro y puede ser de gran ayuda en la visualización de la creación de los árboles sintácticos.

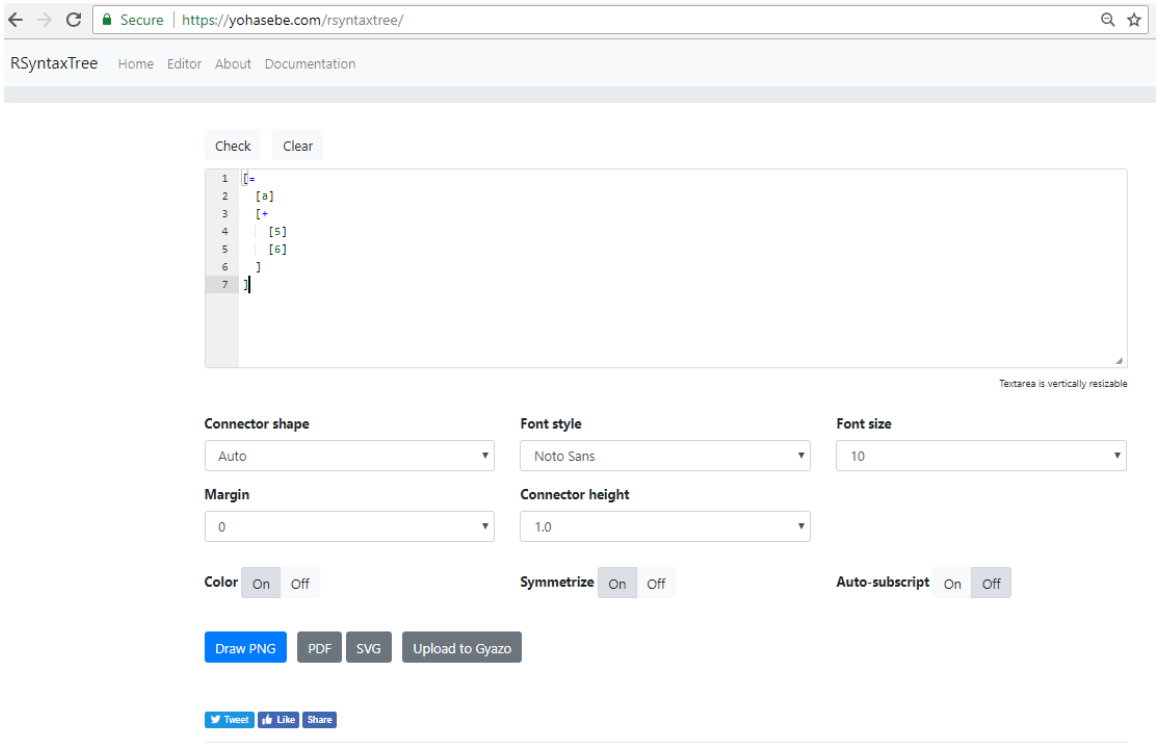


Figura 1.12. Panel de configuración de yohasebe.com/rsyntaxtree

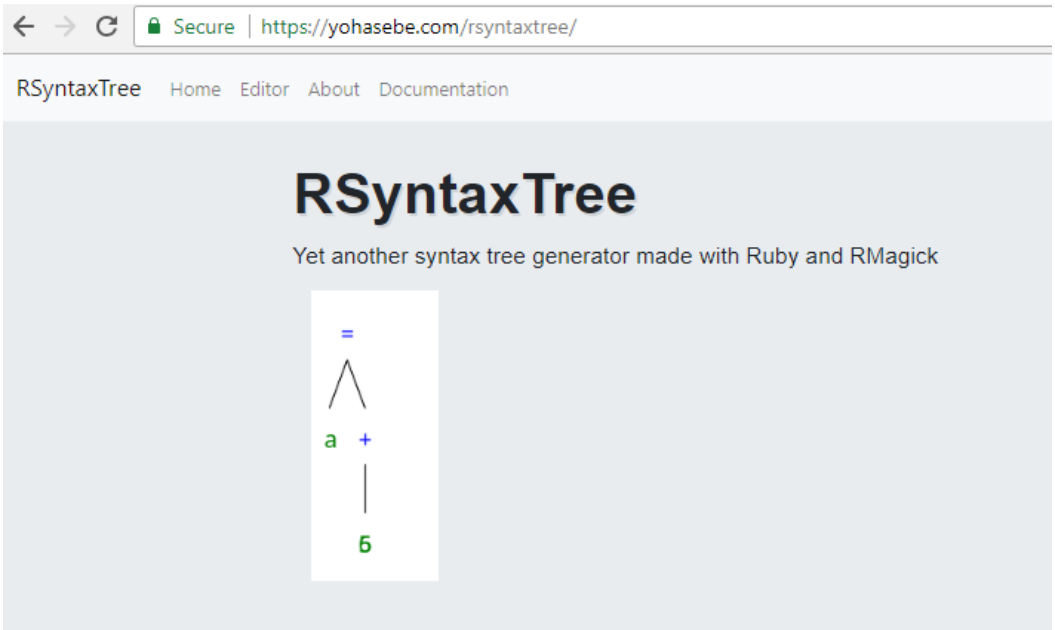


Figura 1.13. Estructura de árbol creada por yohasebe.com/rsyntaxtree

1.5.4. Resumen de la Revisión de las Herramientas

Descripción	Nombre de la herramienta	Escritura de las Gramáticas			Creación del Árbol Sintáctico	Ambiente de Uso		Tipo de Visualización		Sincronia de la Visualización			Disponibilidad	
		BNF	EBNF	Otra		Profesional	Educativo	Dinámica	Estática	Completa	Parcial	Nula	Documentación	Herramienta
Revisión de herramientas de visualización de la Traducción Dirigida por Sintaxis	LISA	X	-	-	X	X	-	-	X	-	X	-	X	X
	CUPV	X	-	-	-	-	X	-	X	-	X	-	-	-
	VCOCO	-	X	-	-	-	X	-	-	-	-	X	-	-
	ExDebugger	-	-	X	X	-	X	X	-	X	-	-	-	-
Herramientas Actuales	BISONC++	X	-	X	-	X	-	-	-	-	-	-	X	X
	COCO/R	-	X	-	-	X	-	-	-	-	-	-	X	X
	JAVACC	X	-	-	X	X	-	-	X	-	-	X	X	X
	JACCIE	-	-	X	X	X	X	X	-	X	-	-	X	X
Contructores árbol sintactico en línea	mshang.ca/syntree	-	-	X	X	X	X	X	-	X	-	-	X	X
	ironcreek.net/phpsyntaxtree	-	-	X	X	X	X	-	X	-	X	-	-	-
	yohasebe.com/rsyntaxtree	-	-	X	X	X	X	-	X	-	X	-	X	X

Tabla 1.1. Esta tabla resume la revisión de las herramientas de visualización y creación del árbol sintáctico y señala con una X que variable a ser evaluada cumple.

1.6. ANÁLISIS DE LAS POSIBLES ÁREAS DE MEJORA EN LAS HERRAMIENTAS DE CREACIÓN DE ÁRBOLES SINTÁCTICOS

Con los resultados obtenidos en la revisión anterior, se pueden observar ciertas áreas en las que se puede mejorar las herramientas según las variables establecidas. Para este análisis se tomarán todas las herramientas como un conjunto, y de esta manera se podrán considerar mejor las áreas en las que se puede incursionar para hacer una mejora.

La primera variable de la revisión anterior fue la escritura de las gramáticas. De las 11 herramientas revisadas a penas 4 utilizan el formato BNF o EBNF que es un estándar que posee mucha documentación y es utilizado en el curso de compiladores. Esta es un área de mejora importante porque facilitaría a los estudiantes el uso de la herramienta ya que estarían familiarizados con la escritura de las gramáticas.

Otra área de mejora que se debe considerar es el ambiente de uso, ya que de las 8 herramientas que componen el grupo uno y dos apenas 3 fueron creadas para el ámbito educativo y los 5 restantes para el uso profesional. Mientras que de las herramientas que componen el grupo tres por sus características se las puede aplicar tanto de manera profesional como de manera educativa. Esto nos demuestra que las herramientas que se utilizan en la actualidad son pensadas en su mayoría para profesionales que las aplican en la creación de programas específicos, pero no para que un estudiante la utilice para comprender de mejor manera los conceptos de compiladores, especialmente la creación de los árboles sintácticos.

La anterior área se complementa con la falta de visualizaciones porque de las 8 herramientas analizadas al principio tan solo 4 tienen visualizaciones del árbol sintáctico; y de estas 4, dos

permiten la posibilidad de graficar una visualización estática y las otras dos grafican el árbol sintáctico permitiendo una visualización dinámica. Lo que deja una brecha enorme entre la ejecución del lenguaje de muestra y la creación del árbol sintáctico, esto en términos de aprendizaje debería estar de la mano. Siendo el tipo de visualizaciones una tercera área de mejora en las herramientas para la construcción del árbol sintáctico.

A partir de las visualizaciones se establece que tan solo 3 herramientas de las 11 revisadas cumplen con una sincronía entre visualización y ejecución del lenguaje completa, reflejando que esta área debe ser tomada en cuenta de manera prioritaria ya que es algo fundamental que la visualización del árbol pueda reflejarse a penas el estudiante ingrese una cadena de prueba para que comprenda de mejor manera la creación del árbol sintáctico.

Finalmente, en el área de disponibilidad de la herramienta y documentación se debe tener en cuenta que las herramientas revisadas que se ofrecen en un ámbito educativo no están disponibles para su uso y tampoco su documentación. Por lo que esta área de mejora es una prioridad para este trabajo de titulación.

1.6.1. Resumen áreas de mejoras

1. Escritura de las gramáticas utilizando un formato estándar BNF o EBNF.
2. Ambiente de uso de la herramienta enfocado a la educación.
3. Generar una visualización dinámica de la creación del árbol sintáctico.
4. Interacción entre el usuario y la visualización del árbol sintáctico.
5. Sincronía entre la visualización y la ejecución del lenguaje (sentencia de prueba).
6. Disponibilidad de la herramienta y de la documentación.

2. CAPITULO II: METODOLOGÍAS DE DESARROLLO E INVESTIGACIÓN

2.1. METODOLOGÍAS ÁGILES PARA EL DESARROLLO

Para este trabajo de titulación se decidió utilizar una metodología ágil ya que el tiempo para realizar el prototipo es corto y el equipo de desarrollo está compuesto por apenas dos personas. También por la apertura que estas metodologías tienen a los cambios sobre la marcha da una facilidad a los retos que se pueden encontrar en el camino del desarrollo.

Además, permiten gestionar las tareas de una manera organizada y simple, mantiene al equipo de desarrollo informado de todos los avances o posibles inconvenientes; siendo este un punto clave en este trabajo de titulación por la complejidad que conlleva la realización del prototipo.

Finalmente, se consideró el objetivo del trabajo de titulación que es generar un prototipo funcional que facilite el aprendizaje de la creación del árbol sintáctico en compiladores, que sea fácil de usar y sobre todo que sea intuitivo para que no necesite una extensa documentación para su uso, y las metodologías ágiles se ajustan a generar productos de software funcionales dejando de lado las extensas documentaciones.

A continuación, se tiene el concepto de metodología ágil para que pueda entenderse de mejor manera como se realizó el prototipo.

2.1.1. Concepto

El desarrollo de software tiene varias dimensiones que deben ser consideradas a lo largo del proceso y el detalle de estas depende de la metodología a utilizar. Las metodologías ágiles están

diseñadas para pequeños proyectos; ya que simplifican los procesos de las metodologías tradicionales, pero no renuncian a ciertas prácticas que ayudan a asegurar la calidad del producto.

Estas metodologías están regidas por "El Manifiesto Ágil" que resume todo lo que conlleva la filosofía ágil; este a su vez, valora ciertos puntos importantes que debe cumplir una metodología para que pueda ser considerada ágil. En la actualidad han tomado fuerza ya que los equipos de trabajo normalmente consideraban que las metodologías tradicionales eran difíciles de implementar y sobre todo muy costosas y encontraron la solución con las metodologías ágiles; al menos, en los proyectos de desarrollo pequeños que están caracterizados por plazos de entrega cortos y requisitos cambiantes.

El presente trabajo de titulación ha optado por Scrum como metodología ágil en cuanto al desarrollo; ya que, por el tiempo destinado al proyecto, no se puede dedicar mucho del mismo a la realización de documentación puesto que el desarrollo del prototipo es solo uno de los objetivos dentro del trabajo de titulación.

A continuación, será descrita brevemente.

2.1.2. Metodología Scrum

2.1.2.1. Concepto

SCRUM nace en 1986 de la investigación realizada por Takeuchi y Nonaka a empresas de tecnología como Canon, Honda y HP; que estaban generando productos en un tiempo menor, con un costo bajo y una buena calidad, comparándolas con varias empresas dentro del mismo entorno. Esto; los llevó a la conclusión, de que estas empresas habían sustituido los equipos

especializados que existían en cada una de las fases del proceso, por un equipo multidisciplinario que trabajaba en el proyecto desde su inicio hasta su culminación.

Este trabajo en equipo; fue comparado con una estrategia en el juego de Rugby, donde los jugadores toman una pelota perdida, y por su juego en equipo logran ponerla en juego nuevamente, el nombre de esta estrategia es SCRUM.

En 1996, Jeff Sutherland y Ken Schwaber presentaron las prácticas que se usaban como proceso formal para el desarrollo de software y que pasarían a incluirse en la lista de Agile Alliance.

2.1.2.2. Beneficios Scrum

La metodología Scrum en la actualidad posee cualidades que hacen sentir al equipo de trabajo más cómodo; ya que, no posee ciertas fases que caracterizaban a las metodologías tradicionales, "Scrum es menos burocrática y está más orientada a la productividad; dejando de lado, por lo menos, sin otorgar una excesiva importancia a la documentación"(Fuentes, 2015, p. 127) esto aporta una ventaja en cuanto a la escalabilidad de los proyectos ya que nada está preestablecido y da libertad a los integrantes del equipo para adaptarse a los cambios.

El equipo al estar en constante interacción tiene como ventajas una clara comunicación, Scrum también aporta flexibilidad al permitir adaptaciones sobre la marcha del proyecto y da la opción de que el resultado sea un software que pueda funcionar de manera incremental.

2.1.2.3. Componentes Scrum

Scrum está formado por cinco componentes que son: backlog, team de desarrollo, Sprints, meetings diarios y presentación de demos. Estos componentes en conjunto; crean una armonía; que hace a esta metodología tan poderosa y aplicable para proyectos de desarrollo de software;

más adelante, se describirán las funciones de cada uno; con el fin de, tener una idea global de lo que conforma Scrum.

Todo proyecto se empieza con la finalidad de resolver un problema o suplir necesidades; el backlog, es el conjunto de estas y también incluye todas las nuevas ideas que pueden surgir para la implementación. No incluye la información sumamente técnica; ya que, al inicio del proyecto no representa mayor importancia, "lo que si son necesarios son los flujos de las funcionalidades y los requisitos necesarios para el entendimiento inicial"(Fuentes, 2015, p. 128). En este componente es importante que los interesados estén presentes cuando sea posible.

En el team de desarrollo se tiene una estructura horizontal; es decir, que no existen jerarquías, lo que evita documentación técnica de tareas a realizarse como ocurre en las metodologías tradicionales. El team cumple sus tareas y se comunica constantemente para evitar riesgos; esto es posible, porque "las recompensas y todos los fracasos son errores de responsabilidad del equipo" (Fuentes, 2015, p. 129), por lo que es importante el compromiso del team con el proyecto.

Los Sprints, son el período de tiempo que se designa para cumplir las tareas que se hayan seleccionado del backlog, los sprints usualmente duran 15 días. Las tareas a ser desarrolladas se eligen en las reuniones de sprint, "estas son tomadas en cuenta dependiendo de la prioridad, complejidad, cantidad y calidad de los requisitos del software"(Fuentes, 2015, p. 129).

Los meetings diarios ayudan a estar pendientes del estado de avance del proyecto; principalmente, sirven para evitar el retraso del proyecto, manteniendo el compromiso del equipo y para corregir inmediatamente cualquier problema que haya aparecido. En estos

meetings diarios el equipo responde tres preguntas con el fin de saber exactamente como marcha el proyecto:

1. ¿Qué se realizó el día anterior?
2. ¿Cuáles son las actividades del día?
3. ¿Qué problemas han aparecido y de qué manera afectan a la solución de los problemas existentes?

Finalmente, tenemos las presentaciones de demos; estas toman lugar en la reunión de fin de sprint, estos demos cumplen dos funciones puntuales. La primera, es la prueba y demostración de los avances, que puede ser realizada por el equipo y los interesados en el proyecto; y, la segunda es la reflexión de los errores y posibles mejoras a implementar. La segunda parte se realiza solamente con el equipo.

2.1.2.4. Conclusión de la metodología Scrum

Scrum es una metodología que permite adaptarse a cambios sin sacrificar calidad y terminar proyectos en períodos de tiempo cortos con costes bajos. También, hace posible "desarrollar software incrementalmente en entornos complejos donde los requisitos no están claros o cambian con mucha frecuencia"(Fuentes, 2015, p. 127).

2.2. ESTUDIO TEÓRICO

En el estudio teórico es un tema clave en todo trabajo de investigación porque es una guía durante todo el proceso, también da directrices para trabajar con la información que se vaya recopilando a medida que se lleva a cabo la investigación.

Cuando se inicia una investigación se debe tener claro lo que se quiere realizar ya que de esto dependerá el método a elegir. Este trabajo de titulación tiene como fin aplicar las técnicas de visualización a la creación del árbol sintáctico en la materia de compiladores para de esta manera ayudar a los estudiantes en la comprensión del mismo.

Para llevar a cabo lo propuesto, se han revisado trabajos relacionados a compiladores que analicen las visualizaciones que muestran algunas herramientas y además los investigadores revisaron otras más nuevas que permiten tener una idea más clara sobre la actualidad de las herramientas de compiladores y especialmente sobre las visualizaciones que estas brindan.

Previo a la revisión, se definieron variables que representan las características a tomar en cuenta para que las herramientas de software sean útiles para el entendimiento de la creación del árbol sintáctico. Estas a su vez ayudaron a determinar las áreas de mejora que se tiene en el campo y que se deben considerar cuando se inicie la realización del prototipo funcional.

Y para complementar este trabajo de titulación, los investigadores realizaron una encuesta a los estudiantes de la facultad de ingeniería para determinar el estilo de aprendizaje de Kolb que predomina para así enfocarse en la técnica de visualización que mejor se apegue al estilo de aprendizaje predominante y conseguir los mejores resultados en los estudiantes al entendimiento de la creación de los arboles sintácticos en la materia de compiladores.

Para alcanzar todo esto se utilizó el método inductivo-deductivo que utilizados por separado son contrarios, pero en conjunto se complementan de una forma única dando un valor agregado a la investigación para que esta pueda ser tomada como base a futuros trabajos relacionados al

tema o a la mejora del prototipo. En la metodología de investigación, se describe brevemente la metodología aplicada.

2.2.1. Metodología de la Investigación

Antes de explicar el método inductivo-deductivo es esencial conocer sus componentes, más adelante se los explica brevemente para tener una idea clara de cada uno para posteriormente entender de mejor manera su trabajo en conjunto.

2.2.1.1. Método Inductivo

Este método utiliza razonamiento inductivo que "observa, estudia y conoce las características genéricas o comunes que se reflejan en un conjunto de realidades para elaborar una propuesta o ley científica de índole general"(José Luis Abreu, 2014, p. 6). El método inductivo se puede definir como aquel que plantea un razonamiento ascendente que inicia la observación y estudio desde lo particular hasta llegar a lo general y que está enfocado en el fin.

2.2.1.2. Método Deductivo

El razonamiento utilizado en este método se basa en "determinar las características de una realidad partícular que se estudia por derivación o resultado de los atributos o enunciados contenidos en proposiciones o leyes científicas de carácter general formuladas con anterioridad"(José Luis Abreu, 2014, p. 6). Una vez determinadas estas características se puede llegar mediante deducciones a una conclusión general que aporta y ayuda a avanzar en el conocimiento de las realidades estudiadas.

2.2.1.3. Método Inductivo-Deductivo

Conformado por las dos metodologías anteriores, el método inductivo-deductivo parte de las observaciones de particularidades que sirven como soporte para proponer generalidades; y estas a su vez permiten realizar predicciones que toman fuerza al ser corroboradas o la pierden cuando son desmentidas.

2.2.1.4. Aplicación del método en el trabajo de titulación

En este trabajo de titulación, se parte de variables específicas que ayudan a observar las carencias existentes en las herramientas de compiladores actuales en cuanto a una labor educativa con el estudiante. Estas variables ayudaron a determinar las generalidades, o áreas de mejora, que se tienen en estas herramientas. Estas generalidades fueron usadas para proponer la predicción de que si se aplicaran técnicas de visualización a las herramientas de compiladores en la creación del árbol sintáctico el entendimiento de los estudiantes en este tema aumentaría.

2.2.2. Revisión Sistemática de la Literatura

Para la revisión sistemática de la literatura se utilizó la metodología propuesta por Jane Webster y Richard T. Watson en su artículo “Analyzing The Past To Prepare For The Future: Writing A Literature Review” que está orientado en brindar una alternativa para la revisión de literatura para investigaciones en el área de tecnología, dando herramientas y procedimientos para que la información recolectada sea de mayor utilidad y faciliten el procesamiento de toda la bibliografía recolectada.

Gracias a la revisión sistemática de la literatura la elección de las herramientas a analizar fue rápida ya que se tomó en cuenta todas las alternativas que aparecieron y se consideraron

solamente las más cercanas a la fecha actual; también ayudo a establecer las variables necesarias para la revisión de las mismas en cuanto a su aporte al aprendizaje de los estudiantes.

Finalmente, la revisión de literatura aporto para que las técnicas de visualización analizadas estén acorde a los parámetros provistos por el estilo de aprendizaje de Kolb predominante en los estudiantes de la facultad de ingeniería en sistemas.

2.2.3. Estilos de aprendizaje de Kolb

Para determinar la mejor opción de técnica de visualización a ser aplicada en el prototipo, se decidió realizar un test de estilos de aprendizaje de Kolb a los estudiantes de la facultad de ingeniería en sistemas a través de una encuesta para así descubrir el estilo predominante de aprendizaje en los estudiantes para de esta manera explotar las cualidades del estilo y brindar una experiencia mucho más enriquecedora en el prototipo.

Más adelante se habla de los estilos de aprendizaje de Kolb con más detalle para entender su aporte en este trabajo de titulación.

2.2.3.1. Modelo de aprendizaje de Kolb

Kolb en su teoría de aprendizaje plantea un modelo de aprendizaje a partir de experiencias y señaló que todo individuo para aprender necesita de cuatro capacidades que considera básicas que son:

1. Experiencia Concreta (EC)
2. Observación Reflexiva (OR)
3. Conceptualización Abstracta (EA)
4. Experimentación Activa (EA)

Y a partir de estas capacidades propuso cuatro estilos de aprendizaje que están compuestos por dos capacidades y presentan distintas características como se observa a continuación:

Estilo de Aprender	Características
Divergente	Modalidades EC y OR. Agilidad imaginativa, visualiza situaciones concretas de diversas perspectivas, formula ideas, emotivo(a), se interesa por las personas.
Asimilador	Modalidades CA (conceptualización activa) y OR (observación reflexiva). Habilidad para crear modelos teóricos, razonamiento inductivo, le interesan menos por las personas y más por los conceptos abstractos.
Convergente	Modalidades CA y EA. Aplicación práctica de las ideas, pruebas de inteligencia de una contestación, solucionar un problema o pregunta, razonamiento hipotético deductivo, poco emotivo(a), prefiere los objetos a las personas.
Acomodador	Modalidades EC (experiencia concreta) y EA (experiencia activa). Hacer, llevar a cabo planes, involucrarse en experiencias nuevas, arriesgado(a), intuitivo(a), depende de otras personas, cómodo con la gente.

Tabla 2.1. Los estilos de aprender y sus características generales. Por Kolb, 1984.

Para que el aprendizaje ocurra, Kolb indica que deben existir al menos dos dimensiones que son descritas como percepción del medio y procesamiento creando de esta manera una versión de mapa cartesiano que se divide en cuatro cuadrantes como se describe en la siguiente imagen:

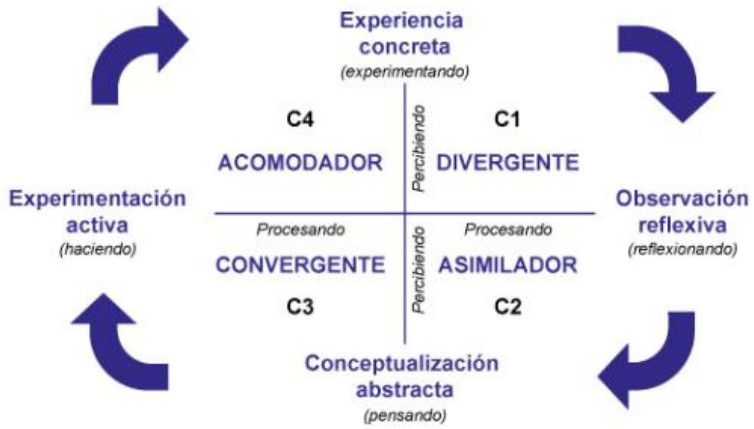


Figura 2.1. Estilos de aprendizaje. Por Kolb, 1984.

A partir de esta relación y la encuesta (anexo 1) realizada a los estudiantes de ingeniería en sistemas, se puede recopilar la información suficiente para poder tener idea de que técnicas de visualización favorecerían de mejor manera a los estudiantes que posean el estilo de aprendizaje predominante en la facultad.

3. CAPÍTULO III: TÉCNICAS DE VISUALIZACIÓN

3.1. INTRODUCCIÓN

El recurso conocido como visualización es una metodología que se utiliza para dar a conocer un algoritmo o programa que muchas veces es confuso para el mundo. En otras palabras, la visualización no es otra cosa que hacer visibles los conceptos abstractos que muchas veces son poco entendibles. "…La importancia de la visualización radica en su fuerza expresiva. Suele decirse que una imagen vale más que mil palabras. Las características de las imágenes hacen que la búsqueda de la información se reduzca significativamente, a pesar de que en una sola imagen puede aparecer resumida información equivalente a enormes cantidades de datos numéricos. Este alto poder de abstracción es otro punto a favor en el uso de imágenes", García D., Hernández D. y Olivar C., (2006). La visualización se ha impulsado aceleradamente en los últimos cuarenta años; volviéndose más completa día tras día, ya que gracias a una imagen se puede trasmitir un mensaje, una idea, un concepto o incluso un tema de manera gráfica y que se vuelva mucho más entendible para las personas a las cuales se les esté trasmitiendo. Los algoritmos de visualización se emplean para diseñar la forma en que se desea presentar dicha información (en redes, jerarquías, entre otros). Esta forma de representar los datos es una de las técnicas más usadas actualmente y sobre todo en campos de diversas empresas", Montero I., (2016).

3.2. TIPOS DE VISUALIZACIONES

Dentro del tema de visualización se encuentran algunos tipos de visualizaciones que permiten entender los conceptos de los sistemas de Informática de manera mucho más gráfica, animada y visual. A continuación, se muestran los principales tipos de visualización del área de Computación e Informática.

3.2.1. Visualización de datos

La visualización de datos va en conjunto con la interfaz del usuario gráfico, las bases de datos, el espacio navegable y la simulación, que es una de las formas culturales realmente nuevas que los ordenadores han hecho posible. Este tipo de visualización ha existido desde el ciclo 18 gracias a Edward Tufte. Con ayuda de un computador se puede visualizar conjuntos de datos mucho más grandes; en comparación a los siglos pasados, crear visualizaciones dinámicas (animadas e interacciones), introducir datos en tiempo real y graficarlos en un análisis matemático; utilizando toda una variedad de métodos, como la minería de datos.

La visualización de datos; también conocida como visualización de la información, proviene de la combinación de la Computación, Estadística y Diseño. Por lo cual se puede decir que "...una buena visualización de los datos ayuda a revelar relaciones y patrones que de otra forma pasarían desapercibidos", Vásquez R., Pérez C. y Torres J. (2013). Entonces, la visualización de datos es un medio para lograr un mayor y mejor entendimiento de la información que nos rodea en general.

Gracias a la invención de los ordenadores y de la tecnología, se han creado nuevos Sistemas y Herramientas Informáticos para el análisis de datos o información en distintas técnicas; es decir,

que tanto los avances tecnológicos como la teoría han generado múltiples soluciones o formas gráficas a los problemas que se han producido a lo largo de la historia. Con lo cual se puede decir que la tecnología es una fuente infinita que se va a expandir en el futuro tecnológico.

3.2.2. Visualización de Software

Hoy en día las computadoras se han vuelto una herramienta necesaria para el aprendizaje de la programación; y, de la misma manera para la comprensión de los algoritmos y programas que se utilizan en las diferentes asignaturas. "…Si bien los entornos de los lenguajes de programación ofrecen capacidades cada vez más útiles al programador, no son aún lo suficientemente amigables con el usuario como es deseable", Moroni N. y Señas P. (2009). Por lo cual, las visualizaciones de software han sido pensadas para el entendimiento de una manera más sencilla de los algoritmos y programas.

Existen diferentes conceptos que definen que son las visualizaciones de software, pero para nosotros los más importantes son los siguientes: "…La visualización de software es el uso de gráfica de computadoras y animación interactiva para ayudar a ilustrar y presentar programas, procesos y algoritmos. Se basa en el uso de diseño gráfico, de animación, de sonido, de video y tiene como característica sobresaliente la interacción entre el usuario y la computadora, apuntando a una mayor comprensión y a un uso efectivo del software",Moroni N. y Señas P. (2009). Otra definición que va acorde con el tema que se está tratando en la investigación nos dice que "…Es una disciplina de la Ingeniería del Software cuyo objetivo es mapear ciertos aspectos de software en una o más representaciones multimediales", Berón M., Riesco D. y Montejano G. (2010). A su vez; existe otra definición más técnica y simple, para la compresión

que dice que: “Visualizar software es hacer que un programa tenga un aspecto visible diferente al que tiene su código fuente”, García D., Hernández D. y Olivar C., (2006).

Si bien las visualizaciones están orientadas a la comprensión de programas, el principal reto consiste en realizar un procedimiento que vincule el Dominio del Problema con el del Programa. Mediante este tipo de visualización se puede ver a los programas de una forma más gráfica y dinámica, del mismo modo que se lo hace con los algoritmos. El principal motivo de introducirse en la visualización de software, es volver descifrables las herramientas ya existentes en la actualidad para el estudio de los diversos algoritmos y programas que se utilizan dentro del área de Programación.

3.2.3. Visualización de Algoritmos

Dentro de las visualizaciones, encontramos las visualizaciones de algoritmos, las cuales son muy beneficiosas para el aprendizaje y la enseñanza de los mismos. La visualización de algoritmos; a diferencia de otras visualizaciones, tiene aspectos distintos que se deben tomar en cuenta: definir claramente y con anterioridad lo que se quiere visualizar y tener la información necesaria para hacerlo. Uno de los conceptos más claros y simples de este tema nos habla de que “...La visualización de algoritmos es el campo de la visualización de software que representa la semántica del programa de computadora”, Moroni N. y Señas P. (2002).

Con el ánimo de mejorar el entendimiento de los algoritmos y programas, se quiere implementar “un sistema debe proveer multi ventanas en las que se debe exhibir la estructura estática del algoritmo, la estructura estática de los datos, el menú de selección de la representación de datos

y aquellas ventanas que permitan el ingreso de los valores de los datos y de su representación con los cuales se realizará la corrida del programa", Moroni N. y Señas P. (2002). Este tipo de visualizaciones son mayormente utilizadas para la docencia, ya que hace que los algoritmos sean amigables para el aprendizaje del usuario; por lo cual, "...Las representaciones, animaciones, y la propia herramienta de visualización deben cumplir una serie de características, entre ellas la capacidad de interacción por parte del usuario y la calidad y claridad de lo representado, mostrando de la forma más intuitiva posible y acompañando las ejecuciones de texto explicativo", García D., Hernández D. y Olivar C., (2006).

A través de la visualización de algoritmos, toda la información de un programa o software se vuelve más gráfico y visual. En la actualidad, lo más importante es "disponer de sistemas amigables con el usuario (facilidad de interacción hombre-máquina, en cuyo diseño se debe poner especial interés en la percepción humana). Pero; además, la experiencia de los usuarios con el lenguaje de programación y su nivel de familiaridad con la propia visualización de software resultante, influyen sobre la calidad del sistema de visualización", Moroni N. y Señas P. (2002); por lo cual, la opción de hacer animaciones o visualizaciones de algoritmos es la más beneficiosa.

En conclusión; la visualización de algoritmos es una rama interesante y muy poco explorada, que muchas veces es incompresible, ya que cualquier persona no entiende de inmediato el concepto de un algoritmo; por lo cual, el uso de la animación es una herramienta muy eficiente para el aprendizaje y enseñanza de conceptos de software.

3.3. VENTAJAS DE LAS VISUALIZACIONES

Hoy en día, las visualizacionesde datos, información, algoritmos y programas, son medios que se usan con mayor frecuencia y son muy comprensibles para los usuarios en el ámbito del aprendizaje y la docencia. La tecnología que va avanzando a la par de las visualizaciones, es parte fundamental para que esta herramienta sea la más utilizada hoy en día. Por lo cual, se pueden nombrar varios beneficios y ventajas que se dan al usarlas. A continuación, las ventajas encontradas en esta investigación.

Para las visualizaciones de datos se tienen las siguientes ventajas y creemos que es importante nombrarlas para el trabajo que se está realizando:

1. **Ayuda en la toma de decisiones acertadas:** Tener grandes cantidades de información ya no es un dolor de cabeza cuando se consigue un panorama más amplio y una interpretación de los datos. La visualización y transformación de la información es lo que permite tomar decisiones e informarlas en tiempo real.
2. **Monitoreo de información:**La visualización de los datos permite tener constancia en todo momento de lo que está pasando con los datos e información.
3. **Optimiza la colaboración/divulgación de la información**: Las aplicaciones móviles para tabletas o celulares fomentan la difusión de las representaciones visuales de datos entre los usuarios a quienes se les haga necesario tener la información de inmediato. Las aplicaciones web nos liberan de las aplicaciones de escritorio tradicionales y favorecen la movilidad y la interacción en tiempo real.

4. **Ahorro de tiempo:** La visualización de datos permite el ahorrar tiempo ya que la información se encuentra unificada instantáneamente a través del software que se esté utilizando.
5. **No se necesita ser experto en tecnología:**La visualización de datos permite a los usuarios utilizar la información de un modo intuitivo. Sin necesidad de conocimientos técnicos profundos, los usuarios inexpertos pueden crear visualizaciones de datos significativas. El análisis de datos ya no es una tarea exclusiva para los expertos informáticos.
6. **Detección de errores:** La visualización de datos en tiempo real,permite gracias a este es nada costoso análisis y muy gráfico,se revelará cualquier error de inmediato. Además, sabremos cuáles son las necesidades y gustos de los usuarios; en este caso los estudiantes, pudiendo adaptarse de acuerdo con las necesidades.

En definitiva, las visualizaciones de datos ofrecen muchas ventajas para los usuarios. "…La visualización de datos constituye el modo más sencillo en que nuestro cerebro puede procesar y transformar grandes cantidades de información", SAS, (2014). Cuando innumerables variables se convierten en un gráfico circular, de líneas, de burbujas o mapas de calor, las compañías pueden utilizar esta información de forma intuitiva para ver conexiones y formular preguntas que nunca antes alguien se había planteado.

Por otro lado; las Visualizaciones de Software y Algoritmos, presentan importantes beneficios sobre todo en la docencia. Se han encontrado los siguientes para el aprendizaje y enseñanza según Moroni N. y Señas P. (2002):

1. Logran un incremento de la motivación; a través de una presentación atrayente, en la que un algoritmo desafiante se transforma en uno más accesible y menos intimidante. Los estudiantes se sienten más motivados a interactuar con el material y a estudiar temas complejos.
2. Facilitan el desarrollo de destrezas ya que brindan la oportunidad de realizar prácticas adicionales. Los estudiantes tienen una nueva forma de experimentar los algoritmos. Además de resolver los ejercicios en papel y escribir los programas, ellos pueden percibir visualmente los algoritmos y estudiar sus características observando e interactuando con la animación.
3. Asisten en el desarrollo de habilidades analíticas y promocionan las predicciones ya que los estudiantes deben recopilar sus propios datos para el análisis del algoritmo y los subsecuentes diseños de algoritmos mejorados. Las animaciones de software ofrecen distintas ventajas comparadas frente a la ayuda ofrecida por la enseñanza tradicional, tal como el libro de texto y el pizarrón.
4. Ofrecen un buen soporte al docente y son de gran ayuda en la clase durante la explicación de la conducta dinámica de un algoritmo.
5. Permiten la exploración; jugando interactivamente, de las peculiaridades de un software, mejorando la comprensión individual de cada uno de los estudiantes. Permite a los mismos, manipular el software y sus entradas, formular hipótesis de la conducta del algoritmo y luego estudiar las acciones resultantes, verificando o refutando sus ideas. Esto los capacita para formar un modelo conceptual del algoritmo, a más de aprender el código. La interactividad agrega un nuevo nivel de efectividad al ambiente de aprendizaje y es una herramienta apropiada en el concepto de enseñanza ya que forza a

los aprendices a tomar parte de la lección, como opuesto a simplemente observar un movimiento. La interactividad ayuda a los estudiantes a adquirir una experiencia invalorable en la resolución de problemas.

6. Es importante el valor educativo de las técnicas de visualización de software. La animación de algoritmos y la visualización de programas ayudan a los estudiantes a comprender los conceptos de software y también a los docentes en su tarea de enseñar dichos conceptos. Distintas experiencias con respecto a la aplicación de la Visualización de Software en la enseñanza de la programación se han realizado en diversas Universidades extranjeras. Actualmente se emplean como recurso didáctico tanto en los cursos elementales como en los avanzados.

Aunque todas las animaciones de algoritmos tienen el mismo propósito, el uso de las mismas puede variar considerablemente. Algunas de las aplicaciones parecen ser las más comunes según Moroni N. y Señas P. (2009):

1. Para acompañar una lectura y ayudar a comprender los conceptos claves que explica el instructor durante la clase.
2. En un laboratorio formal donde los estudiantes interactúan con las computadoras.
3. Para uso informal; por parte de los estudiantes, fuera de la clase, en su tiempo libre, para ayudarles a comprender más acerca de un algoritmo.
4. Para realizar un aprendizaje personalizado e individualizado.

3.4. HERRAMIENTAS PARA LA VISUALIZACIÓN

Por medio de representaciones gráficas se logra la participación de personas con diferentes grados y tipos de educación y se facilita la sistematización de conocimientos y el consenso.

Las técnicas de visualización presentadas pertenecen a unos grandes tipos según Geilfus F. (2008):

- **Las matrices:** Son cuadros que permiten ordenar y presentar la información e ideas en forma lógica; con el fin de cruzar diferentes criterios (matrices de clasificación y de priorización), o de presentar ideas en forma jerárquica (matrices de planificación y otras). Sus aplicaciones son prácticamente ilimitadas y aquí se presentan numerosos ejemplos aplicables tanto al diagnóstico como a las fases de análisis, planificación y seguimiento de las acciones. Un ejemplo de matriz se encuentra en la figura 3.1.

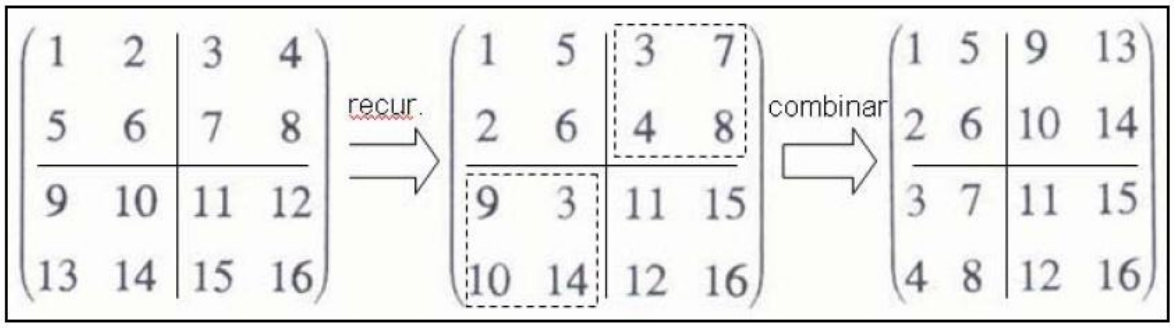


Figura 3.1. Tipo de visualización en matriz por Fernández, L., & Velázquez, J. Á

- **Los mapas y esquemas**: Son representaciones simplificadas de la realidad, tienen muchas aplicaciones en las fases de diagnóstico y análisis y muchas veces sirven de punto de partida para los procesos de desarrollo. A continuación, se puede ver este tipo de técnica en la figura 3.2.

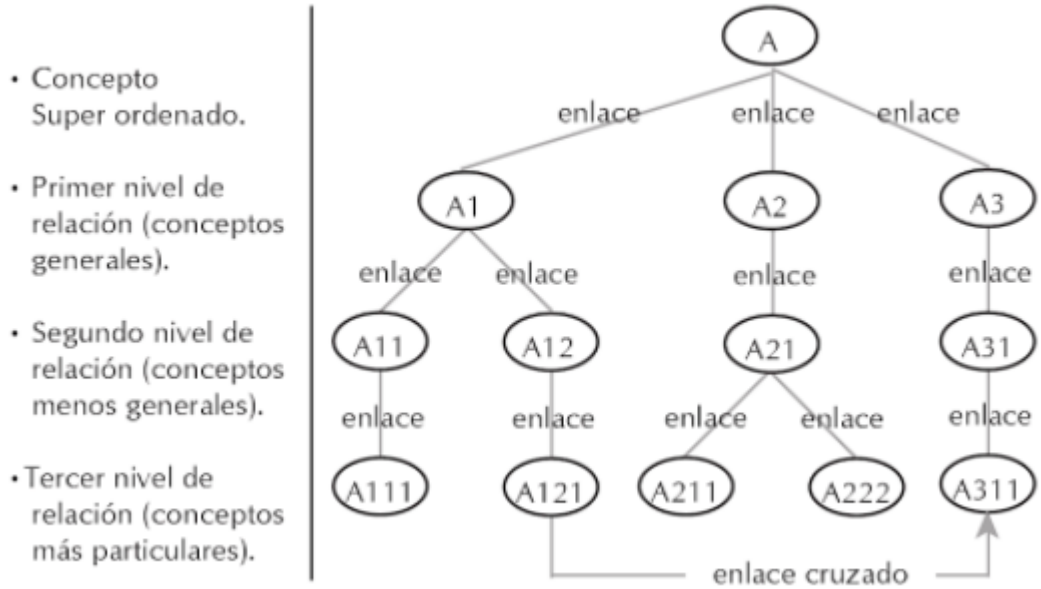


Figura 3.2. Estructura básica de un mapa conceptual por Agustín Campos Arenas.

- **Los flujogramas:** Son un tipo de diagrama en que se presentan en forma esquemática las relaciones entre diferentes elementos (simbolizadas por flechas), como relaciones de causa a efecto, secuencia de eventos, etc. En la figura 3.3. se puede ver claramente un ejemplo de un flujograma.

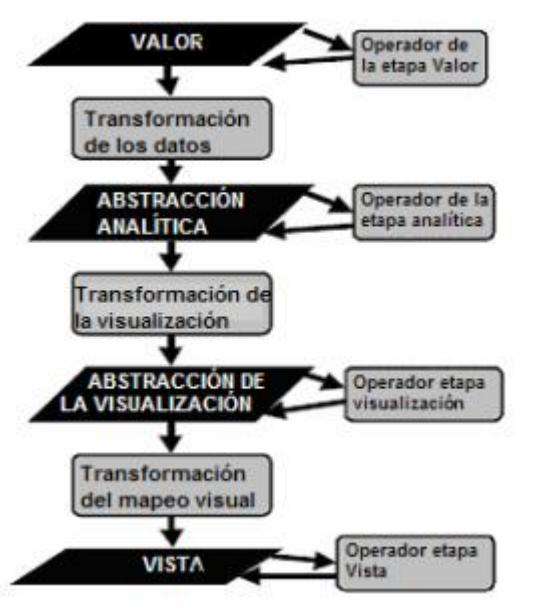


Figura 3.3. Modelo de Referencia de Estados de los Datos, de Chi (2000)

- **Los diagramas temporales:** Son representaciones de la presencia/ausencia; o, de la variación en intensidad de ciertos fenómenos en el tiempo.

CARACTERÍSTICAS DE LAS VISUALIZACIONES PARA EL SOFTWARE

Como se distingue el software que se va a realizar es para el aprendizaje de Arboles Sintácticos del Área de Compiladores lo cual se lo clasifica como un software educativo, ya que "…los programas educativos están pensados para ser utilizados en un proceso formal de aprendizaje y por ese motivo se establece un diseño especifico a través del cual se adquieran unos

conocimientos, unas habilidades, unos procedimientos, en definitiva, para que un estudiante aprenda.", Gros, B. (2000).

Se puede nombrar gran variedad de características que distingue el software de aprendizaje o enseñanza de software comercial, pero a continuación se seleccionará las más importantes y necesarias de tomar en cuenta antes de realizar un software educativo.

Las características que distinguen a un software educativos son según Marqués, P. (2000):

- **Finalidad Didáctica:** están elaborados con una intención pedagógica y en función de unos objetivos de enseñanza.
- **Uso del ordenador:** no requiere de mayor explicación. Sabemos que un software es para ser explorado a través del ordenador.
- **Interacción:** estimulan la participación del estudiante y el intercambio de información entre el estudiante y el ordenador.
- **Individualización del trabajo:** le permiten al estudiante o usuario trabajar de forma individual, de acuerdo con su propio ritmo de aprendizaje.
- **Facilidad de uso:** los conocimientos requeridos para el uso de estos programas son mínimo. El usuario o estudiante, sólo debe seguir las instrucciones que el programa le ofrece tanto para acceder a él como para navegar en él.
- **Multimedia:** Debe ser empleado por un ordenador como un soporte en el cual los alumnos desarrollan las actividades propuestas y las que ellos proponen.

- **Accesible:** Los conocimientos informáticos que requieren para su uso son mínimos.
- **Propositivo:** Capacidad de desarrollar habilidades como conocimientos y destrezas, para el logro de los objetivos de aprendizaje.
- **Personalizado:** Adaptación del software al ritmo de trabajo de cada uno de los estudiantes y de las actividades, tomando en cuenta sus conocimientos, habilidades y competencias.
- **Disciplinario:** Facilita el aprendizaje de los diferentes niveles educativos desde preescolar hasta profesional.

FUNCIONES DEL SOFTWARE EDUCATIVO

Quizás hayas utilizado algún programa de consulta como un diccionario, enciclopedia o atlas para buscar información relacionada con algún tema; aunque la función principal es informar, un software educativo va más allá de proporcionar información y entre sus funciones están las siguientes:

- Instruir: Debe ser explicado, ya sea explícita o implícitamente, lo que se va a realizar en las actividades asignadas.
- Motivar: Regularmente los estudiantes se sentirán atraídos a la información y actividades que se presentan en los programas educativos, pues una de sus funciones es captar la atención de los estudiantes a través de una interfaz llamativa y de fácil entendimiento.
- Guiar

- Evaluar
- Informar: Los programas de tutoriales y base de datos (*clasificación*) son los que más presentan esta función.
- Innovar: Los programas educativos siempre buscan una forma de innovar la forma de enseñar y aprender para que los alumnos retengan más los conocimientos. Además de hacer más simples pero efectivas las herramientas con las cuales se trabaja.

4. CAPÍTULO IV: TÉCNICAS DE VISUALIZACIÓN APLICADAS A ÁRBOLES SINTÁCTICOS

4.1. INTRODUCCIÓN

En este capítulo se explica el porqué de la selección de la técnica de visualización a ser utilizada en la realización del prototipo funcional. Es importante que queden claros los lineamientos que se utilizaron para elegir la mejor técnica de visualización, por este motivo se partió de los estilos de aprendizaje de Kolb que fueron explicados en el Capítulo II. Para esto se realizaron Test de Kolb mediante herramientas en línea a distintos estudiantes de la facultad de ingeniería con el propósito de determinar el estilo o estilos de aprendizaje que predominan entre los estudiantes de Ingeniería en Sistemas.

También se mostrarán a mayor detalle las características y la manera en la que la técnica de visualización a ser utilizada se adapta y explota de mejor forma el estilo predominante en los

estudiantes, para que de esta manera el prototipo sea más útil al momento de aprender sobre la construcción de árboles sintácticos en la materia de compiladores.

4.2. RESULTADOS TEST DE KOLB

4.2.1. Introducción

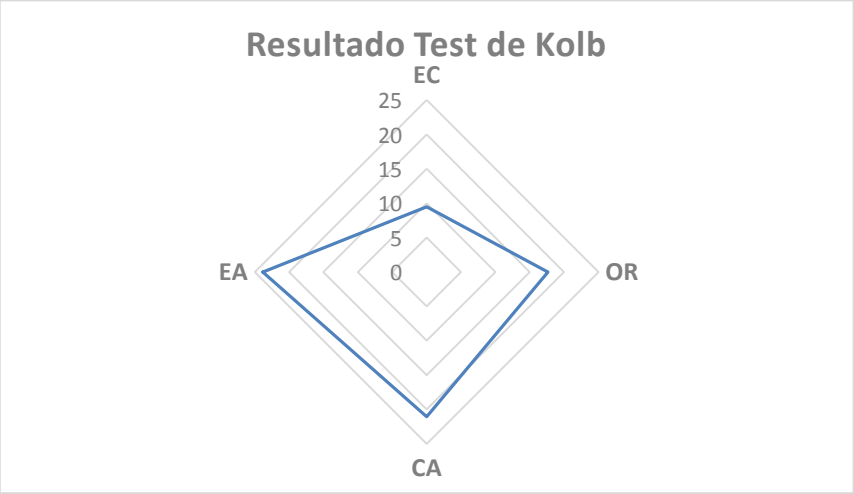
El Test de Kolb aplicado a los estudiantes cuenta con doce preguntas de fácil respuesta, ya que se pide que valoren en el rango de 0 (menor valor) al 3 (mayor valor) distintas situaciones relacionadas a la comprensión y aprendizaje del individuo evaluado; este test puede ser revisado a mayor detalle en el anexo 1.

Una vez finalizado el test se suman los valores de cada columna que representan una de las cuatro técnicas de aprendizaje propuestas por Kolb para finalmente representar en un gráfico de radar estos resultados y ver cuál de los cuatro cuadrantes tiene la mayor fortaleza.

Hay que acotar que los resultados fueron procesados de manera individual y posteriormente se determinó cual era el cuadrante predominante del sesgo a evaluado.

4.2.2. Análisis de resultados

Los resultados obtenidos del test aplicado a los estudiantes reflejo que la mayor parte de los estudiantes presentan un estilo de aprendizaje convergente que es percibido mediante la experimentación activa y la conceptualización abstracta. Obteniendo un promedio de 24 puntos en el apartado de experimentación activa siento este el mayor puntaje obtenido, seguido de la conceptualización abstracta con 21 puntos. A continuación, se muestra el resumen de los resultados obtenidos:



Sigla	Estilo de Aprendizaje	Puntaje Promedio
EC	Experiencia Concreta	9
OR	Observación Reflexiva	18
CA	Conceptualización Abstracta	21
EA	Experimentación Activa	24

El aprendizaje convergente tiene como punto más fuerte la aplicación práctica de ideas y las personas con este estilo de aprendizaje tienden a desempeñarse de mejor manera en tareas o pruebas que requieren una respuesta concreta; para llegar a esta respuesta utilizan razonamiento hipotético deductivo que se basa en plantar una hipótesis del problema y a partir de esta crear proposiciones o consecuencias que posteriormente se prueban con la experiencia. Hay que recalcar que las personas con el estilo de aprendizaje convergente se orientan más a las cosas que a las personas.

Cuando se habla de un estilo de aprendizaje convergente, se habla de que el individuo posee una habilidad predominante en las áreas de abstracción, conceptualización y experimentación activa.

Al hablar de abstracción, se trata de una actividad mental en la que el individuo aísla un elemento para enfocarse en el mismo; al tratarse de estudiantes de ingeniería en sistemas, el concepto de abstracción está presente a diario, desde la POO hasta la generación de modelos utilizando UML. Estos dos ejemplos de abstracción tienen en común que se pueden representar gráficamente para su comprensión, por lo que se puede decir que los individuos comprenden de mejor manera un concepto cuando este está ligado a un aprendizaje más visual, es decir mediante gráficos e imágenes, antes que representar un concepto teórico o información relevante.

A partir de la abstracción, el individuo puede ordenar y construir ideas para crear un concepto de lo que está observando, lo cual genera un aprendizaje visual y a su vez permite que este pueda ser aplicado en un problema o situación similar para llegar a una solución o repetir este ciclo de aprendizaje. Por lo tanto, al estimular la abstracción que un individuo con un estilo de aprendizaje convergente utilizando visualizaciones se puede mejorar su ciclo de aprendizaje fortaleciendo su conceptualización de ideas de un tema en específico, en este caso la construcción del árbol sintáctico en la materia de compiladores.

Los individuos con estilo de aprendizaje convergente tienen dos tipos de percepción:

- Conceptualización Abstracta: Está basada en el reflejo de las experiencias, el individuo teoriza, clasifica o generaliza la experiencia con el fin de generar nueva información. Esta etapa le sirve al individuo para organizar los conocimientos y captar una idea general e identificar patrones y normas.

- Experimentación Activa: En esta etapa el individuo pone a prueba los conocimientos recientemente adquiridos en un ambiente real generando a su vez una nueva experiencia que lo lleva nuevamente a la etapa de conceptualización abstracta.

Cuando un individuo con estilo de aprendizaje convergente tiene un concepto claro, está listo para aplicar los mismos a una situación real en forma de una experimentación activa para cerrar el ciclo de aprendizaje, y volver nuevamente a la etapa de abstracción siendo la experimentación activa la entrada de información hacia la abstracción; por lo tanto, esta experimentación debe ser lo más visual e interactiva posible para explotar estas cualidades del individuo, en especial la conceptualización abstracta que ayuda al aprendizaje desde generalidades que pueden ser representadas con facilidad utilizando visualizaciones gráficas.

4.3. SELECCIÓN DE TÉCNICAS DE VISUALIZACIÓN PARA EL PROTOTIPO

Basados en los resultados obtenidos del Test de Kolb aplicado, teniendo como estilo de aprendizaje predominante es el convergente, el cual está alineado las visualizaciones para que el aprendizaje sea mucho más gráfico y más comprensible para los estudiantes del área de informática.

Se toma la decisión de utilizar las visualizaciones de algoritmos como técnica de visualización para aplicar en el prototipo funcional para la construcción del árbol sintáctico, la cual se diseñará mediante diagramas flujos. Además, se utilizarán visualizaciones de software para la interacción del usuario con el prototipo con el fin de tener la mejor experiencia posible para el usuario y sobre todo que esta explote todos los aspectos en cuanto al estilo de aprendizaje.

4.4. FLUJOGRAMAS

4.4.1. Introducción

Los flujogramas o diagramas de flujo no son otra cosa más que representaciones gráficas de un proceso; representa las actividades que se deben realizar para llegar al resultado final del proceso. Esta herramienta visual también muestra las actividades que se deben realizar en cada etapa y las decisiones que se deben tomar para que el proceso culmine de manera correcta.

La representación gráfica de los flujogramas se realiza a través de una variedad de símbolos que representan a las distintas etapas que tiene el proceso, y a su vez los símbolos se conectan entre sí por líneas que representan los caminos que puede tomar esa etapa para continuar con el proceso.

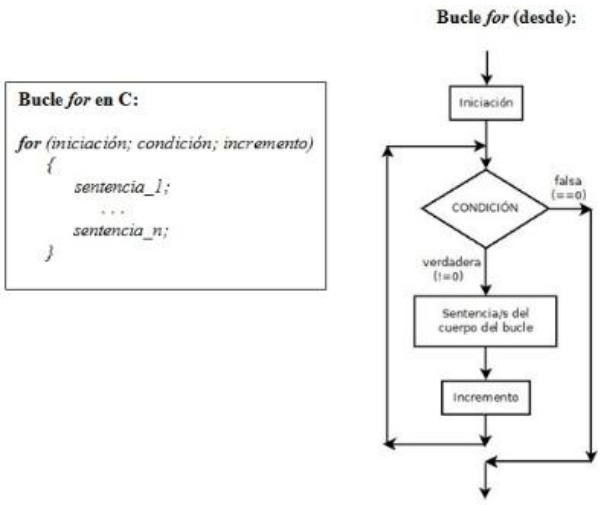


Figura 4.1. Flujograma que representa el proceso de un bucle *for.*

Esta técnica de visualización ayuda a diseñar una representación gráfica completa o parcial de un proceso y el flujo de la información que este maneja. También permite observar de manera

general todas las actividades que se realizan en el proceso con el fin de eliminar las innecesarias y asegurar el entendimiento del proceso de forma global, de igual forma ayudan a la formulación, análisis y solución de un problema. El flujograma ayuda a quien lo lee a comprender el sistema de información de acuerdo con las etapas o actividades que este incluya con el fin de aumentar el entendimiento del proceso y así poder incrementar la información de este para posibles mejoras en las actividades que lo componen.

4.4.2. Características

Para que un flujograma sea óptimo, debe cumplir las siguientes características:

- Sintetizar: Debe ser representado en una hoja si es posible, ya que esto facilita la comprensión y asimilación de este para que sea de utilidad.
- Simbolizar: Se debe diseñar con la simbología correcta en cada actividad para que se lo entienda de manera correcta y evitar confusiones.
- Visibilidad: Un diagrama debe permitir observar todos los pasos de un sistema o proceso sin necesidad de leer notas extensas.

Al cumplir estas características se puede asegurar que el flujograma va a ser de utilidad a quien lo vea e interprete, generando una idea clara del proceso y de las actividades que se deben cumplir para alcanzar el resultado final. También asegura que se han seguido todas las actividades en un orden y de manera correcta, da las bases para generar un conocimiento claro y lógico del proceso para finalmente establecer un enlace entre el razonamiento de la persona y el proceso a ser efectuado.

En el proyecto, se utilizará un flujograma de forma vertical que indica al flujo como una secuencia de operaciones que van de arriba hacia abajo, y será del tipo analítico ya que describirá cada actividad y para qué sirve realizarla con el fin de que el estudiante logre comprender de mejor manera el proceso de construcción del árbol sintáctico en un compilador.

4.4.3. Consideraciones por tomar en cuenta para el prototipo

Estas consideraciones serán tomadas en cuenta para que la aplicación de esta técnica de visualización en el prototipo funcional sea óptima y aporte de gran forma a este proyecto de titulación:

- La redacción del contenido del símbolo de operación debe ser realizada con frases breves y sencillas.
- Evitar usar siglas.
- Debe realizarse de forma limpia y ordenada.

Estas consideraciones fueron tomadas en cuenta antes de iniciar con el prototipo funcional por los autores del trabajo de titulación basándose en los resultados del Test de Kolb para así aprovechar las características de los estilos de aprendizaje predominantes en los estudiantes de la Facultad de Ingeniería en Sistemas de la PUCE.

5. CAPÍTULO V: MODELO VISUAL

5.1. INTRODUCCIÓN

Para determinar las características que debe poseer una herramienta que aporte a la comprensión y aprendizaje de la construcción del árbol sintáctico en la materia de compiladores previamente se debe crear un modelo visual que establezca las mismas. Para este propósito, se utilizará la información del Capítulo I y los resultados obtenidos en el Capítulo IV, con el fin de obtener un modelo visual que abarque todos los aspectos necesarios que deba contener la herramienta centrándonos especialmente en las visualizaciones.

5.2. CARACTERÍSTICAS PARA EL SOFTWARE

Con ayuda de los capítulos anteriores especialmente el Capítulo 2, ayudo a identificar las características que se cree que son las más útiles e importantes para que el software ayude al aprendizaje del Árbol Sintáctico Ascendente y Descendente dentro de la materia de Compiladores. Se espera que el software cumpla con las características que se muestran a continuación.

- Al iniciar el programa, deberá contar con una introducción para el uso de la herramienta. Además, se podrá visualizar varios conceptos sobre los árboles sintácticos que serán dados de manera textual y por voz que serán útiles para el usuario. Dentro de la introducción también se mostrarán videos para mayor comprensión de los conceptos que se utilizarán dentro del software. Esto se puede ver en la figura 5.1.

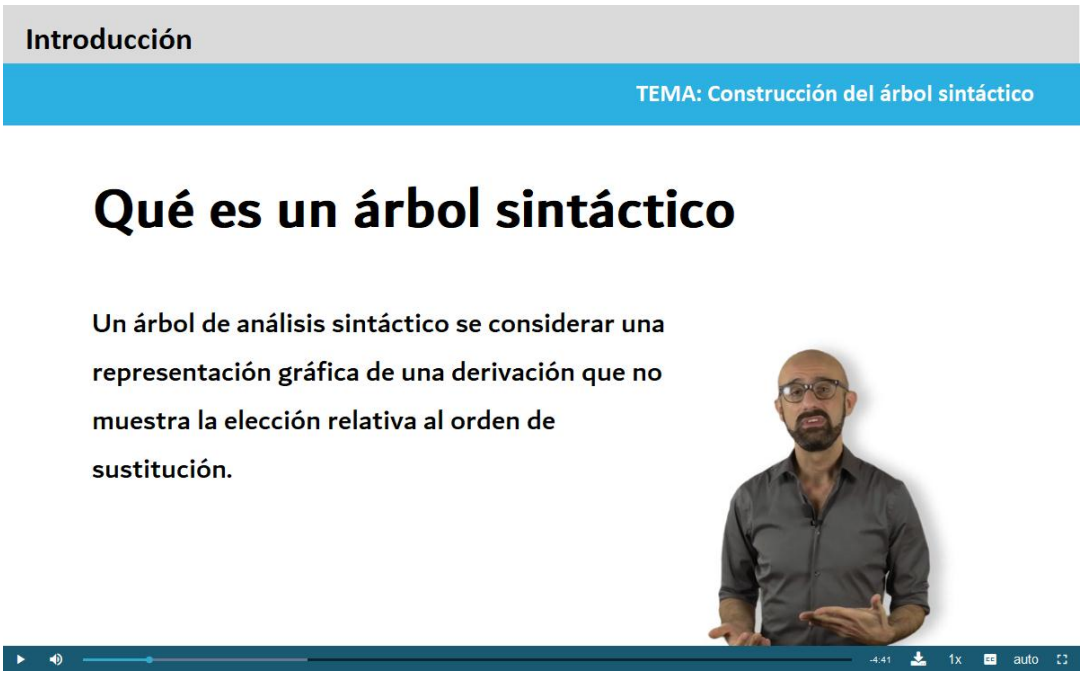


Figura 5.1. Introducción de software a manera de video

- Se utilizará lenguaje EBNF con el fin de que los usuarios puedan utilizar la herramienta con más facilidad como se muestra en figura 5.2.

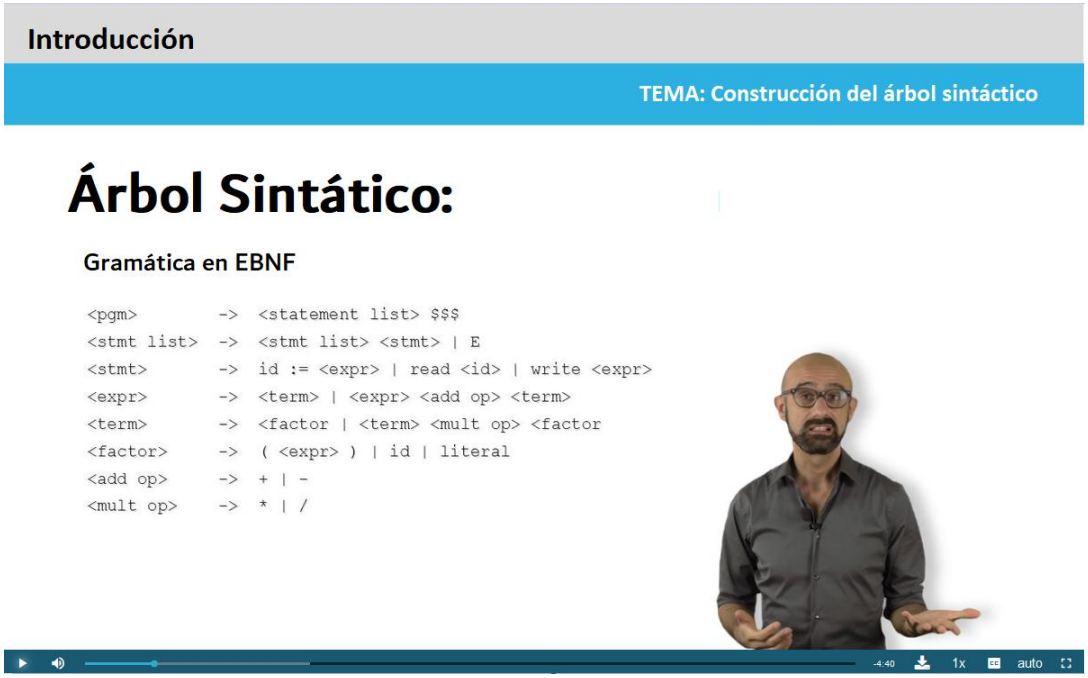


Figura 5.2. Visualización de un lenguaje EBNF

- Contará con ejercicios de ejemplo para practicar la creación del árbol sintáctico como se visualiza en la figura 5.3.

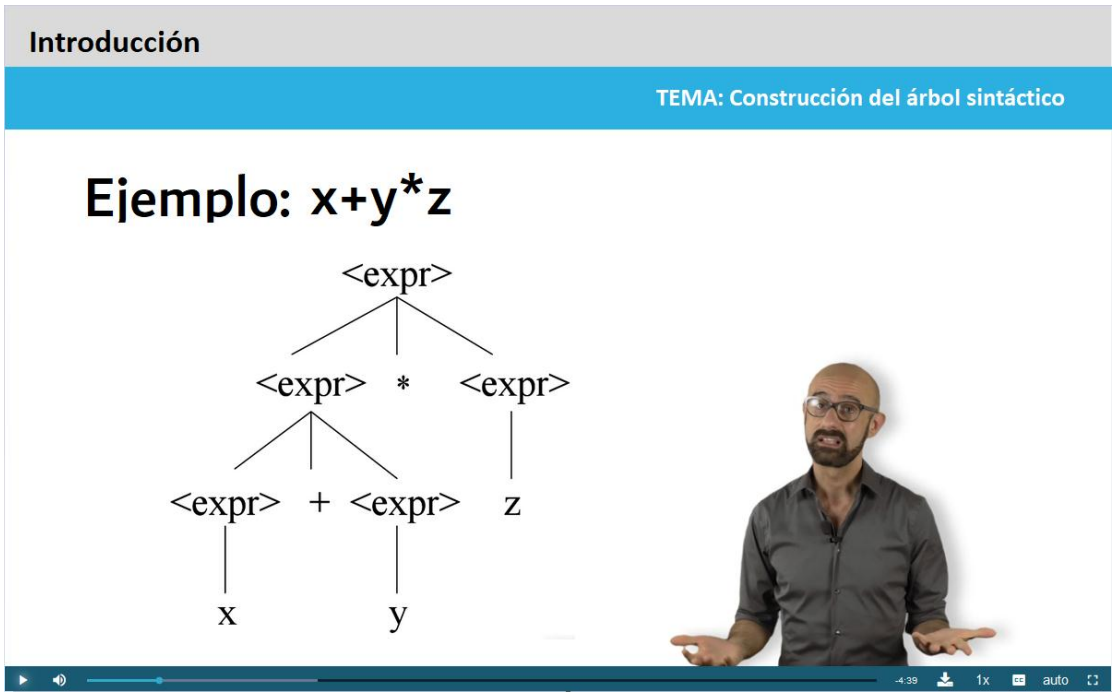


Figura 5.3. Ejercicios de ejemplo explicados

- El prototipo contará con una barra de menú con Archivo y Ayuda. En el Archivo tendrá un submenú con varias opciones que permitirán realizar el trabajo de la construcción más fácilmente, estas opciones se pueden observar en la figura 5.4. siguiente.

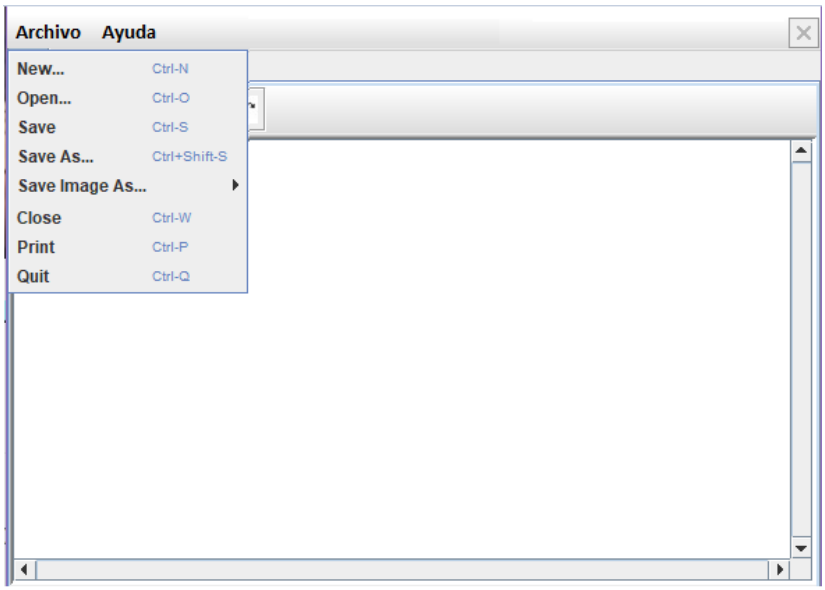


Figura 5.4. Barra de menú del software

- El programa facilitará un espacio para insertar los nodos necesarios para que el usuario pueda crear el árbol sintáctico de una manera fácil e intuitiva, lo cual se hará a través de una interacción mediante clics para crear nuevos nodos y renombrarlos. Los nodos que creen el árbol se conectarán mediante flechas que se arrastraran desde el nodo origen hasta el nodo final como se puede observar en la figura 5.5.a.

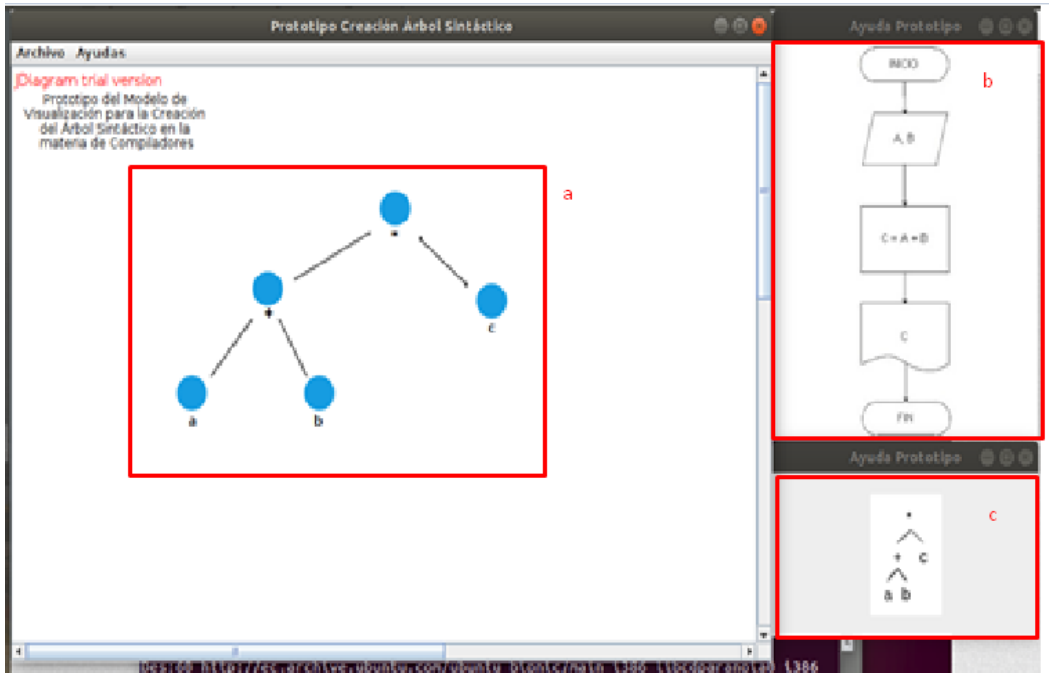


Figura 5.5. Construcción del árbol sintáctico en el prototipo

- En el software se mostrará la gramática a manera de flujograma en la parte derecha de la pantalla con la cual se construirá el árbol sintáctico ascendente y descendente. Esto se puede visualizar en la figura 5.5.b.
- Tendrá las correcciones correspondientes de los errores que se produjeron al momento de realizar el árbol sintáctico como se puede ver en la figura 5.5.c., también se obtendrá la correspondiente calificación del trabajo realizado y, además, contará con el resultado correcto para que haya una mayor comprensión y entendimiento del proceso realizado.

5.3. CARÁCTERÍSTICAS VISUALES

Dentro de las características visuales observadas durante el proceso de revisión a distintas herramientas en el Capítulo I; se pueden adoptar algunas, ya que si se las llega a integrar en una sola herramienta se puede lograr las visualizaciones que aporten al usuario en su proceso de comprensión y aprendizaje del árbol sintáctico.

Como *primera característica* se debe destacar a la sincronía de la visualización en la creación del árbol sintáctico; esto quiere decir que la creación de la construcción del árbol debe ser en base a la especificación, ya que esto da una idea clara de cómo interactúan las reglas léxicas y reglas semánticas en el proceso de construcción del árbol, esta característica se puedo observar en EvDebugger, Jaccie y en la herramienta online mshang.ca/syntree. Por lo tanto, esta característica sería adoptada de estas herramientas.

El tipo de visualización debe ser *dinámica*; es decir, que la creación del árbol sintáctico tenga animaciones que creen una interacción con el usuario como se observa en la figura 5.6. b, con el fin de que la herramienta pueda suplir necesidades después de haber ejecutado el lenguaje. Esto se pudo apreciar de gran manera en la herramienta Jaccie que cuenta con una interacción paso a paso, como se observa en la figura 5.6.a. Esto está implementado para cada etapa del compilador; lo que ayuda a revisar cada etapa que forma parte del proceso de compilación, antes de conseguir el resultado final. También esta opción ayuda a revisar las diferentes variables que se involucran en el proceso e inclusive muestra las reglas semánticas para la creación del árbol, lo cual ayuda a llegar al fin de este trabajo que es el aprendizaje más visual.

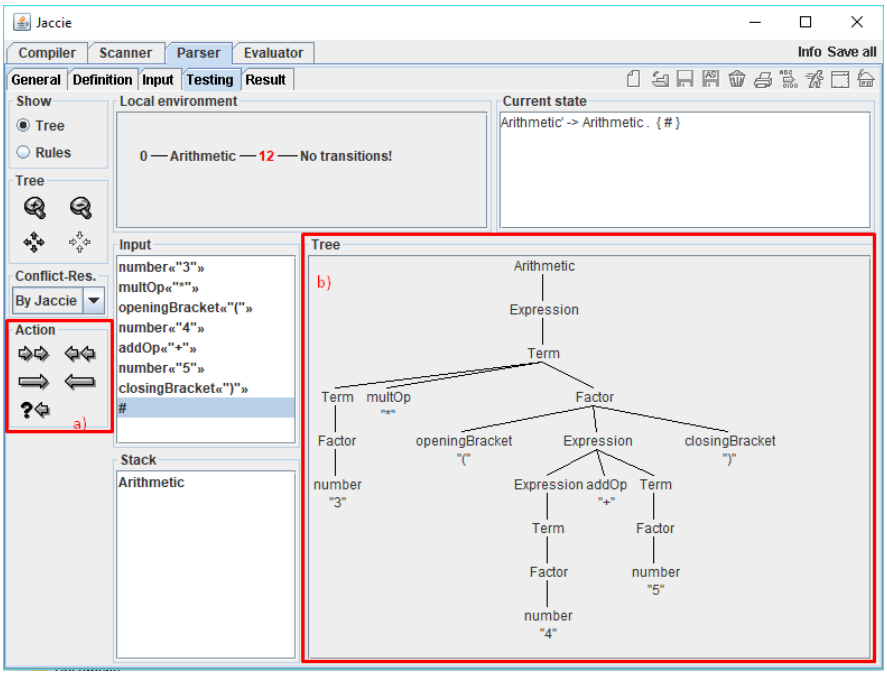


Figura 5.6. Visualización dinámica presentada por Jaccie

La *tercera característica* que se incluye en este modelo visual es la interacción directa del usuario en la construcción del árbol sintáctico a manera de práctica con la creación de una visualización gráfica que cree los nodos necesarios para que el usuario pueda unirlos, como muestra la figura 5.7., con el fin de probar sus conocimientos y tener un concepto más claro sobre la creación del árbol sintáctico.

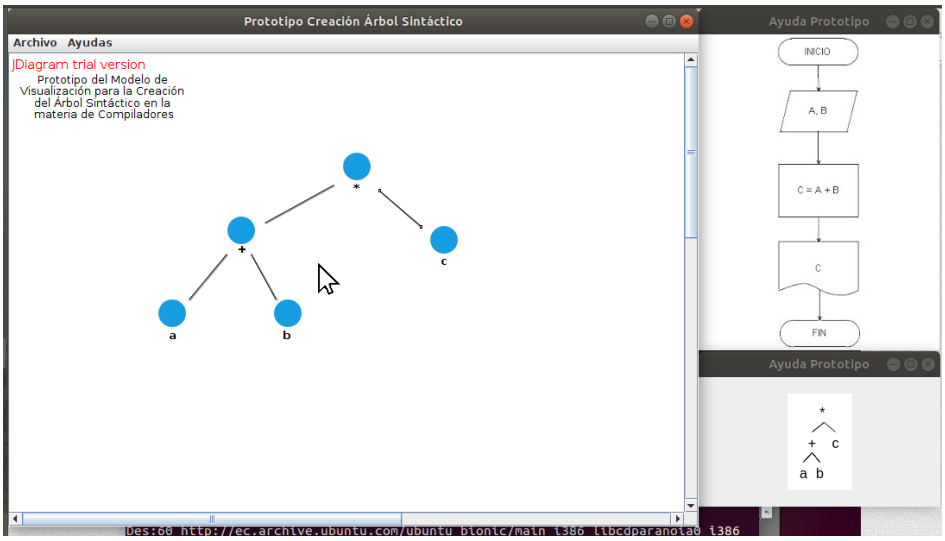


Figura 5.7. Ejemplo de visualización interactiva para creación del árbol sintáctico

La anterior característica gráfica se ve complementada con la técnica de visualización seleccionada en el Capítulo 4, las visualizaciones de algoritmos, con ayuda de los flujogramas. Esta técnica de visualización va a dar la apertura de generar instrucciones gráficas para la construcción del árbol sintáctico de una manera visual y mucho más gráfica, aportando a la abstracción de los usuarios para fortalecer conceptos y dar la opción de una nueva manera de interpretar la teoría con una visualización a la que estén familiarizados ya que es usada para comprender algoritmos de programación.

En conclusión; de la revisión de las herramientas realizada en el Capítulo I, aportó dos características importantes para las visualizaciones necesarias para una herramienta que brinde soporte para el aprendizaje y comprensión de la construcción del árbol sintáctico en la materia de compiladores, mientras que la investigación de técnicas de visualización más los resultados del test de aprendizaje de Kolb aportaron para las otras dos características que en sumadas a las

ya existentes harían que la herramienta sea de mucha utilidad para los usuarios al momento del aprendizaje de esta parte de la materia de compiladores.

5.4. TIPO DE USUARIO

5.4.1. Características del usuario

La herramienta está orientada a estudiantes de Ingeniería en Sistemas y Computación o carreras relacionadas; donde exista la necesidad de comprender la construcción del árbol sintáctico en un compilador, también puede ser utilizada por investigadores como punto de partida para nuevos modelos visuales en diferentes áreas de la Informática o de otros campos.

5.4.2. Requerimientos que Debe Cumplir el Usuario

El usuario debe tener conocimiento previo sobre diagramas de flujo para poder entender las instrucciones que provee la herramienta y sobre todo un conocimiento del proceso de compilación de Análisis Léxico y Análisis Sintáctico.

Para estudiantes de Ingeniería en Sistemas y Computación de la PUCE; se recomienda que al menos tengan aprobados los cinco primeros semestres de la carrera, ya que en esos semestres se adquiere el conocimiento necesario para comprender lo antes explicado.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Luego de realizar la investigación de las herramientas de visualización de la construcción del árbol sintáctico y de las técnicas de visualización para el modelo visual del prototipo funcional, se llegó a las siguientes conclusiones:

Las herramientas revisadas muestran que cumplen con el objetivo de construir el árbol sintáctico ascendente y descendente desde su gramática; pero no permiten el aprendizaje interactivo y visual, siendo una falencia para que se apliquen en el entorno educativo. A través del modelo visual propuesto y del prototipo funcional; como ejemplo, se plantea una nueva idea para que el aprendizaje visual haga que los estudiantes que necesiten comprender y practicar el proceso de creación del árbol sintáctico tengan acceso a una herramienta de fácil uso y con opciones didácticas en este tema.

Como resultado de la encuesta llevada a cabo; los alumnos de la Facultad de Ingeniería en Sistemas y Computación de la PUCE, tienen como estilo de aprendizaje de Kolb al convergente. Por esto; se concluye que los estudiantes de la facultad podrían obtener mayor cantidad de conocimientos en cada clase, si es que estas fueran mucho más visuales y si utilizan herramientas con las características descritas en este trabajo de titulación.

Las técnicas de visualización pueden ser explotadas en el tema; ya que prestan cualidades únicas, que pueden ser aplicadas para herramientas de software que faciliten la enseñanza y aprendizaje de conceptos de la informática y computación.

Se puede concluir que el aprendizaje realizado con herramientas visuales e interactivas; como las propuestas por el prototipo de este trabajo de titulación, ayudan al proceso de enseñanza y refuerzan los conceptos impartidos en clase, gracias a la práctica que proporcionan.

El modelo de visualización propuesto tiene como una característica el uso de flujogramas, que al realizar el prototipo para la construcción del árbol sintáctico ayuda a concluir que ciertos conceptos pueden ser rediseñados utilizando técnicas de visualización con el fin de que sean más didácticos y simples de comprender.

RECOMENDACIONES

Al finalizar el trabajo de titulación con el tema de "Realizar un prototipo funcional aplicando técnicas de visualización para la construcción del árbol sintáctico ascendente y descendente en la asignatura de Compiladores" se han encontrado las recomendaciones que se detallan a continuación:

Se recomienda a los estudiantes de la facultad continuar con la implementación de una herramienta de software que utilice el modelo de visualización propuesto en este trabajo de titulación ya que este puede ser el inicio de un proyecto a gran escala en toda la PUCE.

Una de las características del modelo de visualización propuesto es la abstracción de conceptos, por lo que se recomienda utilizar herramientas de software simples que fomenten la práctica de

los estudiantes para lograr que los conceptos impartidos en clase sean comprendidos de mejor manera.

BIBLIOGRAFÍA

Fuentes, J. R. L. (2015). *Desarrollo de Software Ágil: Extremme Programming y Scrum. 2ª Edición*. CreateSpace Independent Publishing Platform. Recuperado a partir de https://books.google.es/books?id=TxRpCwAAQBAJ

José Luis Abreu. (2014). El Método de la Investigación.

Kolb, D. A. (1984). *Experiential learning: experience as the source of learning and development*. Englewood Cliffs, N.J: Prentice-Hall.

Mössenböck, H. (s. f.). The Compiler Generator Coco/R, 42.

Nicolalde Rodríguez, D. A., & Urquiza Fuentes, J. (2017). Revisión de herramientas de visualización de la Traducción Dirigida por Sintaxis. *Informática Educativa Comunicaciones*, (26).

Resler, D., & Deaver, D. (1998). VCOCO: a visualisation tool for teaching compilers, 199-202.

Rodríguez-Cerezo, D., Enriquez, P., & Sierra, J. L. (2014). Atribute grammars made easier: EvDebugger. *Atribute grammars made easier: EvDebugger*, 23–28.

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, xiii–xxiii.

Berón, M. M., Riesco, D., & Montejano, G. (2010). Estrategias para Facilitar la Comprensión de Programas, 5.

Castro, S. M. S., & Estévez, P. F. E. (2003). Un Modelo Unificado de Visualización, 12.

Fernández, L., & Velázquez, J. Á. (s. f.). Estudio sobre la Visualización de las Técnicas de Diseño de Algoritmos, 10.

Geilfus, F. (2005). *80 herramientas para el desarrollo participativo: diagnóstico, planificación, monitoreo y evaluación*. Costa Rica: IICA.

Moroni, N., & Señas, P. (2002). La Visualización de Algoritmos como Recurso para la Enseñanza de la Programación, 5.

Moroni, N., & Señas, P. (s. f.-a). SVED: SISTEMA DE VISUALIZACIÓN DE ALGORITMOS, 10.

Moroni, N., & Señas, P. (s. f.-b). SVED: SISTEMA DE VISUALIZACIÓN DE ALGORITMOS, 10.

Pontis, S. (s. f.). La historia de la esquemática en la visualización de datos, 12.

García, D. A., Arroyo, D. H., Escaja, C. O., & Flores, C. P. (2006). Intérprete de visualizaciones, 95.

Datos.gob.es. (2016). Visualización de datos, 51.

ANEXOS

Anexo 1: Test de Aprendizaje de Kolb

Test de Estilos de Aprendizaje de Kolb

Autor: Profesor David Kolb

1. Cuando Aprendo: [Prefiero valerme de mis sensaciones y sentimientos]	1. Cuando Aprendo: [Prefiero mirar y atender]	1. Cuando Aprendo: [Prefiero pensar en las ideas]	1. Cuando Aprendo: [Prefiero hacer cosas]
2. Aprendo mejor cuando: [Confío en mis corazonadas y sentimientos]	2. Aprendo mejor cuando: [Atiendo y observo cuidadosamente]	2. Aprendo mejor cuando: [Confío en mis sentimientos lógicos]	2. Aprendo mejor cuando: [Trabajo duramente para que las cosas queden realizadas]
3. Cuando estoy aprendiendo: [Tengo	3. Cuando estoy aprendiendo: [Soy	3. Cuando estoy aprendiendo: [Busco	3. Cuando estoy aprendiendo: [Me

sentimientos y reacciones fuertes]	reservado y tranquilo]	razonar sobre las cosas que están sucediendo]	siento responsable de las cosas]
4. Aprendo a través de: [Sentimientos]	4. Aprendo a través de: [Observaciones]	4. Aprendo a través de: [Razonamientos]	4. Aprendo a través de: [Acciones]
5. Cuando aprendo: [Estoy abierto a nuevas experiencias]	5. Cuando aprendo: [Tomo en cuenta todos los aspectos relacionados]	5. Cuando aprendo: [Prefiero analizar las cosas dividiéndolas en sus partes componentes]	5. Cuando aprendo: [Prefiero hacer las cosas directamente]
6. Cuando estoy aprendiendo: [Soy una persona intuitiva]	6. Cuando estoy aprendiendo: [Soy una persona observadora]	6. Cuando estoy aprendiendo: [Soy una persona lógica]	6. Cuando estoy aprendiendo: [Soy una persona activa]
7. Aprendo mejor a través de: [Las relaciones con mis compañeros]	7. Aprendo mejor a través de: [La observación]	7. Aprendo mejor a través de: [Teorías racionales]	7. Aprendo mejor a través de: [La práctica de los temas tratados]

8. Cuando aprendo: [Me siento involucrado en los temas tratados]	8. Cuando aprendo: [Me tomo mi tiempo antes de actuar]	8. Cuando aprendo: [Prefiero las teorías y las ideas]	8. Cuando aprendo: [Prefiero ver los resultados a través de mi propio trabajo]
9. Aprendo mejor cuando: [Me baso en mis intuiciones y sentimientos]	9. Aprendo mejor cuando: [Me baso en observaciones personales]	9. Aprendo mejor cuando: [Tomo en cuanta mis propias ideas sobre el tema]	9. Aprendo mejor cuando: [Pruebo personalmente la tarea]
10. Cuando estoy aprendiendo: [Soy una persona abierta]	10. Cuando estoy aprendiendo: [Soy una persona reservada]	10. Cuando estoy aprendiendo: [Soy una persona racional]	10. Cuando estoy aprendiendo: [Soy un persona responsable]
11. Cuando aprendo: [Me involucro]	11. Cuando aprendo: [Prefiero Observar]	11. Cuando aprendo: [Prefiero evaluar las cosas]	11. Cuando aprendo: [Prefiero asumir una actitud activa]

12. Aprendo mejor cuando: [Soy receptivo y de mente abierta]	12. Aprendo mejor cuando: [Soy cuidadoso]	12. Aprendo mejor cuando: [Analizo las ideas]	12. Aprendo mejor cuando: [Soy práctico]

	EC	OR	CA	EA
Total				

Nota: Se responde a cada pregunta del 0 al 3 a criterio del evaluado, siendo el 0 la menor calificación y 3 la más alta.