Una Herramienta y Técnica para la Enseñanza de la Programación

Ricardo Pérez Calderón

Universidad Politécnica del Valle de México
Grupo de Tecnologías de la Información, División de Ingeniería en Informática
Tultitlan, Estado de México
rperez@upvm.edu.mx, Apsev1.0@hotmail.com

Abstract. La enseñanza de la programación se debe sustentar en estrategias didácticas usando la mayoría de los medios que existen en la tecnología de la información. La creación de estrategias o medios didácticos permiten usar una gran variedad de elementos tanto manuales cómo automáticos. El software en la enseñanza es un medio que facilita el logro de los objetivos, ayuda al profesor y al alumno en su proceso de enseñanza-aprendizaje, uno cómo el facilitador y el otro cómo generador de su propio conocimiento. Esta línea de investigación enfatiza los diversos enfoques de la educación que se encuentran en boga, el estilo de aprendizaje de los alumnos y los conceptos técnicos de programación, sustentadolo, en una técnica para el diseño de algoritmos y una herramienta automatizada en el desarrollo de programas. Todo lo anterior encaminado hacia la reducción del tiempo en aprender a programar

Keywords: Algoritmo, pseudocódigo, programación, paradigma y lenguajes.

1 Introducción

La enseñanza es una actividad básica del ser humano. Es la forma cómo nuestra cultura ha perpetuado y transmitido su conocimiento, que ya es tan basto y que sólo un ser humano no puede asimilarlo. El ser humano debe ahora especializarse en un tema y sobre este deberá aprender para dedicarse posteriormente en una actividad profesional. Una de estas actividades es la programación, cuya enseñanza o aprendizaje puede no resultar sencilla. A este respecto se presenta en este trabajo una propuesta sobre su enseñanza reforzada con el uso de una herramienta, es decir, el cómo desarrollar algoritmos a través de una metodología de diseño, para que estos puedan ser llevados en la creación de un programa en al menos en uno de los 4 grandes paradigmas de la programación.

Es importante mencionar que la programación al ser una actividad netamente profesional y compleja es necesario crear diversas estrategias didácticas para facilitarla, este trabajo minimiza la complejidad en el aprender a programar, y si algún profesionista egresado de una área no Informática deseara aprender a programar, lo podría hacer usando una este tipo de herramienta automatizada. El aprender algún

M.A Cruz-Chávez, J.C Zavala-Díaz (Eds): CICos 2008, ISBN:978-607-00-0165-9. pp. 229-239, 2008

conocimiento en los adultos se le llama andragogía y en un profesionista con su poca o mucha experiencia en el área de programación no es la excepción y él le daría aplicabilidad y sustentabilidad a esta herramienta.

1.1 Reforzamiento

Se ha detectado que los estudiantes de una carrera en informática o afines tienen problemas cuando empiezan a desarrollar sus programas; dificultades que pueden extenderse a lo largo de sus estudios. Muchas veces estos tropiezos son de mucha tensión para el alumno y no puede tener estabilidad académica a consecuencia de una fuerte carga estudiantil que se tiene cómo alumno universitario. Por lo tanto la reducción en el tiempo de aprendizaje es fundamental para ayudar al estudiante en el aprovechamiento de todas aquellas materias que requieren de una habilidad de programación, o en su defecto que necesiten más tiempo para desarrollar sus actividades académicas de cualquier índole. La enseñanza de la programación es una actividad compleja y difícil, sin embargo a través de sus diversas vertientes existe una infinita gama de tópicos para estudiarse y desarrollarse como líneas de investigación. Al respecto en este trabajo se plantea la problemática al momento de desarrollar algoritmos y proponer una solución estratégica didáctica. Por lo tanto el objetivo primordial que se busca es implementar un sistema informático para apoyar la enseñanza de lenguajes de programación, el desarrollo de algoritmos y la elaboración de programas. Partiendo de esta idea la hipótesis propuesta es: “La reducción del tiempo en aprender a programar, está relacionada positivamente con el uso de la metodología de diseño de algoritmos y de una herramienta automatizada en un sistema de información”

2 Preparación

Los lenguajes de programación han sufrido una transformación sustancial desde sus inicios. Si bien en las primeras computadoras era necesario cablearlas para su programación, en nuestros días no hay más que sólo el tener una PC para experimentar sobre su programación. Tras su evolución, hoy en día podemos identificar varias generaciones de lenguajes de programación. Lenguajes de 1ra. Generación, bajo esta clasificación se engloban todos aquellos códigos de programación conocidos cómo lenguajes máquina ó códigos máquina un ejemplo es, el lenguaje ensamblador, el cual es un elemento importante dentro de las computadoras. Lenguajes de 2da. Generación- Esta etapa fue Impulsada por los estudios del matemático húngaro John Vonn Neumann sobre el concepto del programa almacenado en memoria, aquí se desarrollaron los primeros lenguajes ensambladores simbólicos, junto con algunos otros que hoy se catalogan cómo de nivel medio. Lenguajes de 3ra. Generación.- Son todos aquellos lenguajes que parten de una gramática y poseen una sintaxis propia de un lenguaje libre de contexto. Aquí caen la mayoría de los lenguajes de programación que existen actualmente. Lenguajes de 4ta. Generación son aquellos que son orientados al usuario final que parten de un modelo declarativo, en el que lo que importa es la expresión de problema y la

solución esperada más no la forma en que ésta se obtenga, diseñados para ser usados por el usuario final. El diseño e implementación de los lenguajes de programación al igual que las computadoras también ha ido evolucionando de una manera continua y metódica desde que aparecieron en la década de los cincuenta, se solía desarrollar nuevos lenguajes como parte de proyectos de desarrollo de software especializados, un ejemplo de ello; es que cuando el departamento de defensa de los estados unidos realizó un estudio que formaba parte de sus esfuerzos para el desarrollo de un lenguaje llamado ADA en la década de los setenta, encontró que se estaba utilizando más de 500 lenguajes en diversos proyectos en ese mismo organismo, lo que llevo a tomar una serie de estándares para la unificación de sus proyectos en el uso de la herramientas. Existen cuatro grandes clasificaciones que catalogan a la mayoría de los lenguajes de programación, los cuales se mencionaran a continuación, cómo paradigmas de la programación.

Paradigma Imperativo. En el modelo de programación imperativa se utilizan las diferentes estructuras de control y estructuras de datos predefinidas por el propio lenguaje o las definidas por el propio desarrollador, en este contexto el usuario definen variables, constantes, funciones, procedimientos, entre otras. El programa se comporta como una maquina abstracta. En términos coloquiales se le indica al programa que instrucciones se deben de ejecutar primero y hasta que no termine esa instrucción o conjunto de instrucciones no se inicia la siguiente y así sucesivamente hasta el final del programa. Un ejemplo muy claro de este tipo programación es el lenguaje Pascal, que por cierto es una de los que se plantea en el proyecto y en la herramienta automatizada.

Paradigma Funcional. En la programación funcional todo gira con base en la definición de funciones y a la aplicación de esas funciones. Por ejemplo en la hoja electrónica de Excel, para hacer un promedio con el asistente, este manda a ejecutar la función, pero primero pide que se resuelvan todos lo argumentos y en el momento que se tengan los datos, entonces si se ejecuta la función. Un ejemplo muy claro de este tipo de programación es el lenguaje Scheme. En términos coloquiales se le indica al programa, ejecutar una función predefinida con o sin parámetros, y hasta que se tenga resueltos todos los parámetros, entonces hasta ese momento se ejecuta la función. También es otro de los lenguajes que se plantea en la investigación y en la herramienta propuesta.

Paradigma Basado en reglas. En la programación lógica un problema es expresado a través de los tópicos necesarios para ello, es decir un conjunto de relaciones, reglas y hechos que le rigen. Por ejemplo si se desea saber el como llegar a una estación X del metro partiendo de otra estación que no se encuentra en la misma línea, entonces se puede llegar a la estación solicitada a través de diversos caminos, es decir diferentes transbordos, a lo mejor unos se tardan más tiempo que los otros, pero se llega al resultado, a este concepto se le denomina Backtracking (regreso a más posibles respuestas). En el ejemplo anterior se observan reglas, hechos y relaciones. Un ejemplo muy claro de este tipo de programación es el lenguaje Prolog. Sin embargo a pesar de que sea a través de conceptos poco comunes en el ámbito de la programación, si se pueden generar infinidad de aplicaciones científicas, técnicas o

académicas, como por ejemplo bases de datos deductivas. También Prolog es otro de los lenguajes que se plantea en la investigación y en la herramienta propuesta.

Paradigma Orientado a Objetos. La programación Orientada a objetos (POO) parte de la concepción de conceptos más sofisticados en comparación a los paradigmas anteriores, de entidades participantes en un problema cómo objetos, metodos, mensajes y las relaciones que existen (herencia, polimorfismo, encapsulamiento, entre otras cosas). Un ejemplo muy claro de este tipo de programación pueden ser muchos lenguajes, pero el que llama mucho la atención por su concepción es el lenguaje Java, el cual cambia de forma radical con todos los paradigmas de programación y que vino a revolucionar el contexto del desarrollo de sistemas tanto de dispositivos fijos (PC) como dispositivos móviles (PDA, Celular, entre otros).

Estos son los 4 grandes paradigmas de la programación, en ellos se va a sustentar el proyecto y la herramienta automatizada que se desarrolló para soportar esta línea de investigación. De estos grandes paradigmas existen lenguajes muy representativos cómo los que se ha mencionado, que forman parte de nuestro tema de estudio.

Herramientas de enseñanza. La educación exige la necesidad de crear diversos tipos de estrategias y recursos para que realmente ofrezca posibilidades de desarrollo a todos los alumnos [01] y no solo a unos cuantos y una vez que los alumnos, sean más competentes sus entornos se volverán más exigentes de igual manera. El software educativo es una herramienta capaz de transformar el proceso de enseñanza-aprendizaje. Se hará una breve descripcion de los métodos didácticos que se han formulado y que tienen que ver con la mejora a la estrategia didáctica de la programación.

Metodologías de la enseñanza de la programación.

La Universidad de Sevilla, a través de su departamento de lenguajes y sistemas informáticos, cuenta con varias asignaturas que comprenden su metodología didáctica en la enseñanza de la programación. El departamento de la universidad considera que uno de los aspectos más importantes es la manera de motivar al alumno, además la manera de como que se le haga llegar de todo tipo de información tanto de los profesores como de los propios alumnos, y de materias anteriores para ayudarle y que se le facilite la programación [02] .

Gerald Jay Sussman y Jack Wisdom, comenta que se reconoce que un estudiante puede saber la teoría y que también puede tener problemas para la aplicación de ésta. Cuando el estudiante no tiene un procedimiento formal para aprender la técnica de resolución a un problema le costará trabajo aplicar los conceptos que ya conoce. El autor comenta que expresar una metodología de enseñanza como un lenguaje de programación obliga a que ésta no sea ambigua y sea altamente efectiva. La tarea de formular un método como un programa y depurar el programa es un ejercicio poderoso en el aprendizaje. Por otro lado el mismo autor comenta que a través de la programación él puede enseñar mecánica, este estudio lo ha llevado acabo en el MIT donde enseña [12]

En el trabajo de Mario Oviedo Galdeano se analizan los problemas más comunes en la enseñanza de la programación, desde la visión del autor, y que se consideran más importantes para el logro del objetivo de la asignatura, y que de alguna manera son los mismos que para este estudio se han detectado con el paso del tiempo y de la experiencia. El autor comenta que la programación, al ser una actividad mental compleja y creativa, requiere de 4 características: inteligencia, conocimiento, habilidades y disciplina, las cuales se adquieren con el paso del tiempo. También el autor sugiere una estrategia para tal efecto y la divide en lenguajes de programación y herramientas de desarrollo, que con ayuda de conceptos de técnicas de programación, lo llevaría a la enseñanza de la programación en sí misma. Puede considerarse hacer un examen diagnostico en el cual se observara el nivel de madurez del grupo y así poder generar la estrategia didáctica más efectiva para el logro del objetivo. [03]

Se ha comprobado que el uso del método global, es decir las letras solo se pueden comprender en el contexto de sus palabras y las palabras solo se pueden comprender en el contexto de sus frases; para el aprendizaje de un lenguaje de programación, ahorra tiempo y esfuerzos. Así se creó un ambiente de aprendizaje con un editor interactivo de algoritmos, un constructor automático de trazas y un traductor de para programas en lenguaje Pascal, en este trabajo se presentan los resultados obtenidos en una experiencia de campo diseñada para comprobar la efectividad de la aplicación [01]

Existe un trabajo de simulación sobre pedagogía que sin duda es parecido al aquí presentado; es de Arnoldo Oronico, que tiene como titulo “Una robótica pedagógica”, en el, lo que se busca es facilitar la manera en que se aborda el tema de la robótica a través de un software de simulación que genere el aprendizaje significativo y con ello lo lleve estrechamente a la realidad. [05]

Mediante el aprendizaje en grupo lo que hace es que los alumnos al mismo tiempo diseñen un programa y compartan responsabilidades, fracasos, frustraciones, y éxitos. Esta técnica, derivada de la falta de computadoras en las escuelas, es popular en el ámbito empresarial, lo que llevo a mejoras significativas tanto en calidad como en cantidad en el aprendizaje de la programación. Con el desarrollo de este concepto y con el uso de la Internet se está depurando el concepto de tal manera que en forma remota no solo es posible la participación de dos alumnos sino los que fuesen necesarios para la colaboración de un proyecto. El método se llama DOMOSIN-TPC [04]

La complejidad de los programas que se desarrollan actualmente produce la necesidad de iniciar a los alumnos en un camino que los conduzca a utilizar efectivas técnicas de programación. Es importante para ello poner énfasis en el diseño previo y el uso de la metodología correcta. Se ha comprobado, que una estrategia trascendental es comenzar a enseñar programación utilizando los algoritmos como recursos esquemáticos para plasmar el modelo de la resolución de un problema. Esto genera una primera etapa de la programación que resulta un tanto dificil y tediosa para los alumnos que están necesitados de aplicar los conceptos en una computadora. El hecho de esquematizar la problematica hasta dejarse bien cuando se trabaja de manera

teorica y comprobar la corrección de la misma en una u otra parte presenta inconvenientes importantes [01].

Estos son los trabajos que se han realizado para el desarrollo y mejoramiento de la enseñanza de la programación. El presente trabajo reforzara y tomara experiencias de ellos para ratificar o desechar la hipótesis planteada en la investigación.

3 Aplicacion.

Existen un sin fin de maneras de escribir algoritmos; se pueden denotar cómo una metodología, para que a su vez con un proceso considerable puedan llegar a ser implementados en un lenguaje de programación. Se han implementado cientos de lenguajes desde los años 50s. Sin embargo ninguna persona que se considere especialista en programar domina más de 3 lenguajes; la mayoría de la empresas, siempre usan por lo regular un solo ambiente de desarrollo para su aplicaciones, no tienen una mezcla de ellos, de hecho casi siempre se especializan en uno. ¿Entonces cómo pregunta W. Pratt Terrance [06], porque estudiar una diversidad de lenguajes que es poco probable usar en el ámbito profesional? La respuesta a esta pregunta esta centrada en los diversos paradigmas de la programación que son un referente para que los estudiantes tengan una amplia perspectiva de cada uno de ellos y que además puedan pensar en crear programas en los diferentes ambientes de la programación. Entonces se les enseña la mejor manera de atacar un nuevo lenguaje a pesar de que no lo dominen, también en el entendido de que la mayoría de los lenguajes se soportan sobre una gama de conocimientos básicos (tipos de datos, estructuras de control, funciones, procedimiento, entre otros). Tambien por otro lado como comenta W. Pratt existen una serie de razones para fundamentar esta pregunta en la cual no solo se distingue a un lenguaje por las siguientes características, si no que además deberíamos de incluir el costo, que muy atinadamente comenta Eduardo René Rodríguez Ávila: a) Mejorar la habilidad para desarrollar procesos eficaces, b) mejorar el uso del lenguaje, c) acrecentar el propio vocabulario con instrucciones, d) la sintaxis o estructuras de control sobre programación, e) hacer posible una mejor elección del lenguaje de programación. Tomando en cuenta estos elementos se ha diseñado una herramienta, qué se ve y se usa como si fuera un lenguaje de programación; que es totalmente de apoyo para la docencia y que a su vez es muy útil para el alumno en su proceso aprendizaje, esto no lleva a reafirmar los puntos anteriores de Terrence W. Pratt, aunado a ello Eduardo René Rodríguez Ávila[07], nos da una explicacion para sustentar un correcto y completo algoritmo, desde sus caracteristicas, planteamiento para el desarrollo de ellos, hasta la manera de entender la problematica a resolver, sin necesidad de usar un lenguaje de programación.

3.1 Ciclo de vida de la herramienta.

1. Análisis. El desarrollo y metodología de la aplicacion se ha realizado con un levantamiento de informacion en base a diversos topicos que acontinuacion se comentan: la experiencia de los docentes en la industria, así como en la enseñanza de diversas asignaturas relacionadas con la programación, platicas informales con los alumnos en varias generaciones en la carrera de Informática, en donde se comenta la dificultad del área, la complejidad de las materias de la academia de programación y su respectiva seriación, malas bases en el desarrollo de programas, el desconocimiento de las estructuras de control, el desconocimiento de las diversas estructuras de datos, el no tener una metodología propia para el desarrollo de programas, el buen o mal desarrollo de la lógica de programación y sus elementos, el desconocimiento de varias metodologías para el desarrollo de algoritmos, el contar o no con computadora en su casa, el contar o no con el lenguaje programación instalado en su casa y por ultimo el tiempo para practicar por parte de los alumnos es casi nulo. Muchas de estas características se subsanan con la herramienta propuesta, con ello se desarrolló la respectiva modelación a través de diagramas de flujo.

2. Desarrollo. La aplicacion se programó en un lenguaje con ambiente visual, usando un motor de base de datos básico, un dns de conexion ODBC y un respectivo proceso de instalación (setup).

3. Pruebas. Esta fase se llevo a cabo con algoritmos que se dan en lógica de programación y algunos de la referencia bibliografica del Dr. Cairo [13], cómo por ejemplo: un número par, un número múltiplo de otro, identificación de una persona, una tabla de multiplicar, Serie de fibonacci, un número perfecto, entre otros. Esto se llevo en varias etapas con dos grupos de trabajo (experimental y control), a ambos se les dio teoría y una metodología para hacer algoritmos y hasta la tercera fase, se separaron, al grupo de control se les exhortó a que realizaran los algoritmos en los ambientes de programación reales y al grupo experimental en la herramienta propuesta.

Tiempos en que se tardo el grupo de control en realizar el primer algoritmo. **Table 1.**

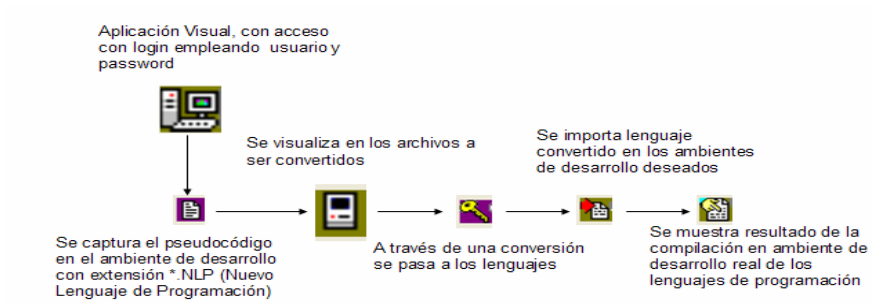
No	Descripción del programa	Tiempo en minutos
1	Suma de los primeros 10 dígitos positivos	30
2	Desarrollar una tabla de multiplicar	20
3	Determinar si una persona es joven o adulta	10
4	Obtener el Factorial de un numero positivo	15

Tiempos en que se tardo el grupo de experimental en realizar el primer algoritmo. **Table 2.**

No	Descripción del programa	Tiempo en minutos
1	Suma de los primeros 10 dígitos positivos	50
2	Desarrollar una tabla de multiplicar	50

3	Determinar si una persona es joven o adulta	30
4	Obtener el Factorial de un numero positivo	35

Fig. 1.



Descripción del proceso de conversión de pseudocódigo a los lenguajes de programación

3.2 Código del programa

Ejemplo de un algoritmo en pseudocódigo tomado del libro Metodologia de programacion del Dr. Osvaldo Cairo [11].

```
Programa NúmeroPerfecto
 define
  Valor TipoDeDato Entero
  Contador TipoDeDato Entero
  Indice TipoDeDato Entero
  Resultado TipoDeDato Entero
  Operacion TipoDeDato Entero
  I TipoDeDato Entero
  j TipoDedato Entero
  Suma TipoDeDato Entero
  Valida TipodeDato Entero
Inicioprograma
 lee (Valor)
 I = 1
 Mientras I menoroiguala Valor haz
  Inicioproceso
   suma = 0
   j = 1
   Operacion = i div 2
    Mientras j menoroiguala operación haz
     Inicioproceso
       Resultado = i residuo j
       Si resultado iguala 0 entonces
         Suma = suma mas j
       j = j mas 1
     Finproceso
```

```
    Si suma iguala I entonces
      Imprime ('El Número Perfecto Es ==> :'  I);
    Si suma diferentea i entonces
      Imprime ('El Número No Perfecto Es ==> :'  I);
    i = i mas 1
   Finproceso
 Lee (valor)
 finprograma
```

Ejemplo de un programa ya convertido con la herramienta propuesta en este caso es al lenguaje Pascal, partiendo del pseudocódigo, de igual manera se obtiene el código convertido para los otros lenguajes propuestos en la herramienta, listo para su compilación y corrida.

```
program númeroperfecto
    var
     valor : integer;
     contador : integer;
     indice : integer;
     resultado : integer;
     operación : integer;
     i : integer;
     j : integer;
     suma : integer;
     valida : integer;
   begin
    readln (valor);
    i := 1;
    while i <=  valor  do
     begin
      suma := 0;
      j := 1;
      operación := i div 2;
       while j <= operación do
        begin
          resultado := i mod j;
          if resultado = 0 then
            suma := suma + j;
          j := j + 1;
        end;
      If suma = i then
        writeln (el número  perfecto es ==>: i);
      if suma  <> i then
        writeln (el número not  perfecto es ==>: i);
      i := i + 1;
     end;
   end.
```

Parte del código que hace la conversion del pseudocodigo al lenguaje Pascal, existe tambien código para hacer las conversions a los otros lenguajes propuestos en la herramienta.

```
   For i:= 0 to Lista.items.count -1 do
    begin
      s          := lista.Items.Strings[i];
      arreglo[j] := s;
```

```
      j          := j + 1;
    end;
  k:=1;
  l:=1;
  while l <= j - 1 do
  begin
   cadena := arreglo[l];
   lon    := length(cadena);
   i:=1;
   token := '';
   FINaliza := 0;
   while i <= lon do
   begin
   if (copy(cadena,i,1) <> #32) AND (copy(cadena,i,1) >=
         #40) AND (copy(cadena,i,1) <= #126) THEN
   token := concat(token,copy(cadena,i,1));
   if (copy(cadena,i,1) <> #32) AND (copy(cadena,i,1) >=
   #40) AND (copy(cadena,i,1) <= #126) and (i = lon) then
    begin
     busca;
     palabras[k] := token;
     inserta;
     k := k + 1;
     token := '';
    end;
    if (copy(cadena,i,1) = #32) then
     begin
      busca;
      palabras[k] := token;
      inserta;
      k := k + 1;
      token := '';
     end;
```

Conclusiones.

En el grupo de control, los tiempos de desarrollo de los algoritmos en los restantes lenguajes casi eran los mismos que para el primero, por lo tanto el tiempo para hacer la conversión en los 6 lenguajes realmente se multiplicaba por 6 para algunos alumnos o más en algunos otros casos. Para el grupo experimental el tiempo para la conversión del primer lenguaje se elevo, en comparación con el grupo de control, pero en los siguientes 5 lenguajes se redujo drásticamente por las conversiones de la herramienta, ello nos lleva a corroborar la hipótesis planteada, entre más se practique más se aprende. Existe avance considerable con o sin la herramienta. Por último para este segundo grupo (experimental) empiezan a descubrir la sintaxis más rápidamente de los 6 lenguajes por un lado y por otro lado sí no se tiene experiencia previa en el desarrollo de la programación aprenden lo relacionado a ella, a través de la herramienta propuesta.

References

1. Norma Moroni, Perla Señas JEITICS 2005 Primeras Jornadas de Educación en Informática y TICS en Argentina.
2. www.lsi.us.es/docencia/pagina_asignatura.php?id=28&cur=2004#pro
   Structure and interpretation of classical mechanics, Gerald Jay Sussman, Jack Wisdom
3. Mario Oviedo Galdeano, academias de computación de la UPIICSA. La enseñanza de la programación.
4. Aprendizaje en grupo de la programación mediante técnicas de colaboración distribuida en tiempo real, Crescencio Bravo, Miguel A. Redondo, Manuel Ortega, Universidad Castilla-La mancha en España.
5. Arnoldo Oronico, Una robótica pedagógica, aodorico@gmail.com, Marco Teórico sobre Robótica.
6. Terrance W. Pratt, Marvin V. Zelkowitz, Lenguajes de Programación, Tercera edición, Pearson Educación.
7. M. en C. Eduardo René Rodríguez Ávila. "El Correcto y Completo Desarrollo de un Algoritmo", Sección de Estudios de Posgrado e Investigación del IPN en la UPIICSA.
8. Gramática obtenida de pascal, a través de Borland en Delphi
9. Gramática obtenida de Scheme, a través de Dr. Scheme, versión 301,http://www.plt-scheme.org.
10. Gramática obtenida de Prolog, a través de SWI-Prolog, versión 3.0,http://www.swi-prolog.org/download.html.
11. Osvaldo Cairó, Metodología de la programación, tercera edición, Editorial Alfaomega.