Conciencia Tecnológica No. 42, Julio-Diciembre 2011

Traductor para Describir Sistemas de Información

Investigación

Ing. Emiro Muñoz Jerez
Universidad Industrial de Santander, Facultad de Ingenierías Físico Mecánicas, Escuela de Ingeniería de Sistemas e Informática
Cra 27 Calle 9, Bucaramanga – Santander – Colombia
Tel. (057) 6349042
emiro270273@gmail.com, jaimealb@uis.edu.co

Resumen

En la universidad y en las empresas, siempre se nos ha presentado un paradigma en el cual el mundo está compuesto por objetos, pero si se observa con detenimiento que la realidad no son simplemente objetos sino sistemas compuestos por objetos los cuales tienen algo en común que los relaciona. ¿Pero como describir sistemas?. En la actualidad no existe un traductor que permita describir sistemas sino solo estructuras u objetos. Para implementar estructuras están los lenguajes 3GL, Java, C o el mismo SQL. Para objetos está OQL. Aquí en este artículo lo que se pretende es especificar los detalles de implementación de un traductor que permita describir sistemas y que a su vez este traductor pueda ser conectado con un motor de Bases de Datos Orientado a Objetos o Relacional.

Palabras clave: MODRES, Traductor, Bases de Datos, Objetos, Sistemas.

Abstract

In the university and in the companies, always we have been appeared by a paradigm in which the world is composed by objects, but if is observed thoroughly that the reality they are not simply objects but systems composed by objects which have anything jointly that relates them. ¿But as systems describe?. At present there does not exist a translator who allows to describe systems but only you structure or objects. To implement structures there are languages like 3GL, Java or SQL, for objects OQL. The main purpose of this article, is to specify the implementation details of a translator that permits system descriptions and the connection to a database engine object-oriented or relational.

Key words: MODRES, Translator, Databases, Objects, Systems.

Introducción

Los lenguajes de programación constituyen una herramienta fundamental en la informática. Basta considerar que cualquier producto software es desarrollado utilizando uno o varios lenguajes de programación. De hecho, la elección de lenguajes de programación adecuados puede ser la clave del éxito de muchos proyectos informáticos.

En la actualidad existe una gran variedad de lenguajes de programación. La descripción de estos lenguajes requiere de una sintaxis[1] y de una semántica [2] . Mientras que para la descripción sintáctica, la notación BNF se utiliza de forma prácticamente universal. Ahora bien en la descripción semántica, no existe un formalismo comúnmente aceptado. En diversas ocasiones, la semántica es especificada a través del formalismo que impone el propio lenguaje de programación.

Los lenguaje de programación como Java, C#, C++, Visual Basic y los lenguajes de datos como SQL, Postgres SQL, MySQL e incluso el mismo Db4o, permiten definir estructuras y objetos. Pero en ellos no se puede definir sistemas, que es lo que se pretende implementar con el traductor mencionado en este artículo.

Para los lenguajes mencionados anteriormente, incluyendo el OQL del estándar ODMG [1], una clase es como una entidad, cada una con sus respectivos atributos y su propio método. Pero los modelos de sistemas van más allá enfocando sistemas (conjunto de clases interrelacionadas) compartiendo un método común, que es el comportamiento emergente del sistema, como lo presenta la figura 1.

[1] Identifica el formato de los programas que se pueden escribir.
[2] Hace referencia a su comportamiento como tal.

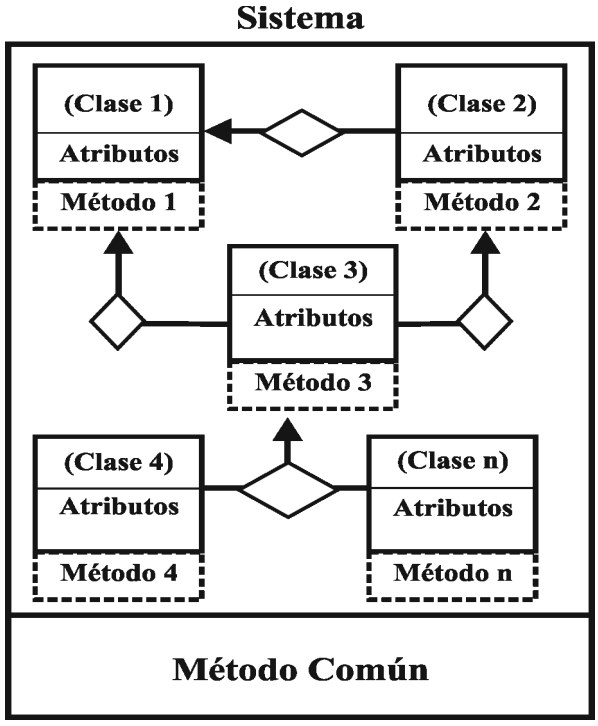

Figura 1. Esta figura muestra como se plantea la definición de un sistema según el modelo de sistemas propuesto por el profesor Albarracín en su tesis Doctoral [2].

Fundamentos Teórico

Traductor

Un traductor se define como un programa que traduce o convierte un texto o programa escrito en un lenguaje fuente a un texto o programa equivalente, escrito en un lenguaje destino, produciendo, si cabe, mensajes de error [3]. Los traductores engloban tanto a los compiladores como a los intérpretes. La diferencia radica en que un compilador se ejecuta en modo desarrollo y los intérpretes lo hacen en modo ejecución. Es decir el compilador ejecuta todo el bloque de instrucciones, mientras que el intérprete los hace instrucción por instrucción generando resultados de manera inmediata. La figura 2 muestra el esquema básico de un traductor (Compilador/Intérprete).

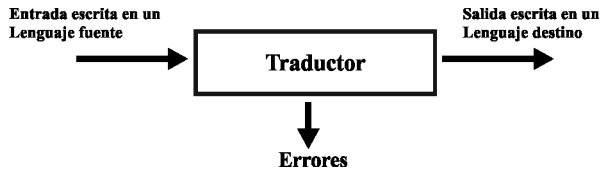

Figura 2. Esquema básico de un Traductor [3].

Bases de Datos Orientadas a Objetos

Una base de datos orientada a objetos (BDOO), a diferencia las bases de datos convencionales incorpora dentro de su modelo de datos la funcionalidad pertinente, o sea que tienen la capacidad de almacenar y recuperar objetos en los que se almacena su estado y su comportamiento. Otra propiedad de las BDOO es la potencia que proporcionan al programador al permitirle especificar tanto la estructura de objetos complejos, como las operaciones que se pueden aplicar sobre dichos objetos. Si bien es cierto que hoy en día se habla mucho de este modelo, todavía es muy reciente y es propio de los sistemas informáticos orientados a objetos [4]. Este modelo incorpora todos los conceptos importantes del paradigma de objetos:

Encapsulación. Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.

Herencia. Propiedad a través de la cual los objetos heredan dentro una jerarquía de clases.

Polimorfismo. Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

Modelo Propuesto por ODMG

El ODMG [1], es un consorcio industrial de vendedores de SGBDOO que después de su creación se afilió al OMG. El ODMG no es una organización de estándares acreditada en la forma en que lo es ISO o ANSI pero tiene mucha influencia en lo que a estándares sobre SGBDO se refiere. En 1993 publicó su primer conjunto de estándares sobre el tema: el ODMG-93, que en 1997 ha evolucionado hacia el ODMG 2.0. En enero de 2000 se ha publicado el ODMG 3.0. El estándar ODMG define el Modelo de Objetos que debe ser soportado por el SGBDO. ODMG se basó en el Modelo de Objetos del OMG que sigue una arquitectura de "núcleo + componentes".

Por otro lado, el lenguaje de BD es especificado mediante un Lenguaje de Definición de Objetos (ODL) que corresponde con el DDL de los SGBD tradicionales, un Lenguaje de Manipulación de Objetos (OML) y un Lenguaje de Consulta (OQL) [1].

El Traductor, planteado en este artículo en lo posible deberá ajustarse al modelo ODMG mencionado anteriormente, pero esto no quiere decir que se tenga que diseñar siguiendo todas las reglas o normas del estándar, por que lo que aquí se pretende es que este traductor rompa con el paradigma de objetos tradicional.

Metodología

La metodología según Pressman [5], se refiere a las estrategias planteadas para el desarrollo del proyecto. Abarca entonces todo el conjunto de actividades que se incluirán en el cronograma.

Para el buen desarrollo de este proyecto, se ha seleccionado la metodología en cascada, debido a que esta metodología o modelo es la más apropiada para este tipo de proyecto de investigación. La elección de este modelo se debe a las siguientes características:

El desarrollo de software es cíclico.

Este modelo tiene una secuencia ordenada.

Provee de un gran control sobre las fechas de entrega.

Establece criterios de entrada y salida en cada fase claramente definidos, de acuerdo con Pressman [5].

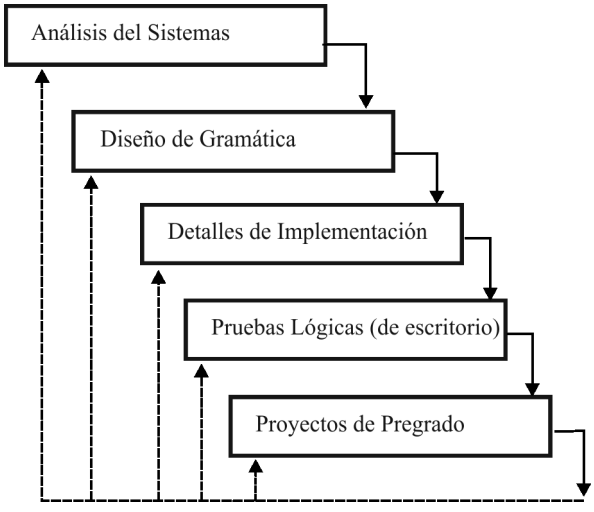

Figura 3. En la figura se muestran las etapas con las cuales se desarrolló el proyecto, para ello se utilizó el modelo en cascada.

Análisis de sistemas

En esta primera etapa del proyecto se ejecutará un análisis detallado de la organización en las empresas y del Modelo de Datos Reorientado a Sistemas MODRES [2], su estructura y su funcionalidad. Con el objetivo de entender como en este modelo los sistemas, clases y sus métodos son declarados de una manera diferente, a la que se está acostumbrado en un lenguaje o motor de base de datos tradicional.

Diseño de gramática

La creación de un modelo de datos, implica la utilización de un lenguaje que permita describir dicho modelo. Pero como dicho lenguaje no existe todavía por causa de la mayor granularidad del MODRES [2], respecto a los anteriores modelos, y también por causa del mejor acercamiento hacia un paradigma de sistemas en la concepción del MODRES, es necesario crearlo, siendo dicho traductor planteado por el profesor Albarracín [2]. El traductor como tal requiere de los componentes de definición, manipulación y consulta de objetos (ODL Y OML). Similar a su antepasado SQL del modelo relacional compuesto también de un DDL y un DML. Por tal motivo, se hace necesario diseñar una Gramática libre de contexto para este modelo, que sea capaz de generar el ODL y OML propio del traductor.

Para el diseño de esta gramática, se utilizará el sistema formal "BNF" o forma de Backus-Naur [10]. Teniendo en cuenta la sintaxis SQL, el estándar ODMG y la sintaxis empleada por los lenguaje de programación orientados a objetos, como es el caso de los lenguajes JAVA y C#.

Pero si bien es cierto que el primer paso para construir el traductor, es definir la gramática, se hace necesario también describir la estructura del lenguaje que implementa los componentes ODL y OML.

La estructura que tendrá el traductor en cuestión deberá contener los siguientes ítems de acuerdo con lo citado en la tesis de pregrado [6].

Valores literales. Ellos son valores de variables, como números, fechas, horas y los valores tipo NULL.

Caracteres especiales. Estos son aquellos caracteres que todo lenguaje de programación permite al definir una instrucción, siendo los mas comunes los siguientes: (, *,), ‘, <, +, /, {, &, >, -, (,}, =).

Palabras reservadas. Son aquellas que son únicas y propias del lenguaje, no siendo posible definir una variable, constante, u objeto con el nombre de estas palabras reservadas.

Comentarios. Estos son aceptados desde una secuencia ‘/*’ hasta la máxima secuencia ‘*/’.

Sintaxis generales. Solo aceptan un comando de entrada por vez y siempre deberá terminar en ";".

Sintaxis de sentencias ODL. Estas son las que permiten la definición de jerarquías de sistemas y sus elementos estructurales, las clases.

Sintaxis de sentencias OML. Estas son las que permiten definir y utilizar los métodos de los sistemas así como consultar sus elementos estructurales y sus elementos funcionales.

Detalles de implementación

Aquí en esta etapa se especificará los detalles de implementación del traductor. Detalles tales como:

Validación de Supersistemas, Subsistemas,

Sistemas, Clases y Métodos.
Resolver la herencia.
Descomponer los Supersistemas, Subsistemas y Sistemas.
Reagrupar las clases y sus métodos.
Conexión y entrega de Clases a un motor de Bases de datos orientado a objetos o relacional.
Diseño de Interfaz de usuario.
Proceso de Pruebas.

Analizador léxico

Un analizador léxico es aquel que toma las entradas digitadas y las divide en componentes léxicos, que son secuencias de caracteres que tienen un significado atómico de acuerdo con lo escrito en el libro [7]. Una vez hecha esta clasificación ellas son enviadas al analizador sintáctico como TOKENS. La figura 4, muestra como se lleva a cabo este proceso

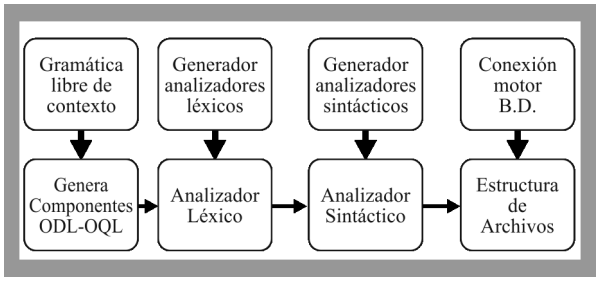

Figura 4. En la figura se muestra cómo es el diseño básico de un Traductor [7].

Analizador Sintáctico

Un analizador sintáctico es un programa que recibe como entrada los elementos individuales o componentes léxicos (tokens) del programa fuente y determina la estructura de las sentencias o instrucciones que lo conforman según el libro [7].

Es por esto que las principales funciones del analizador son detectar e informar los errores sintácticos y la generación de un árbol sintáctico por cada una de las expresiones digitadas por el usuario. La figura 5 muestra el esquema general del análisis sintáctico.

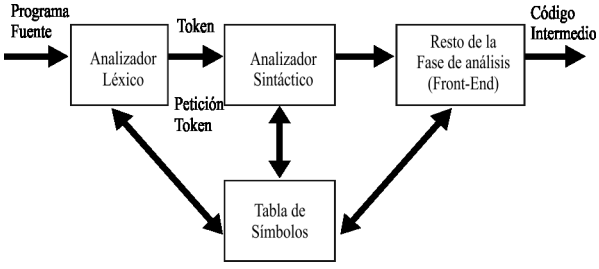

Figura 5. Esta figura presenta el esquema general del análisis sintáctico de cualquier traductor [7].

Pruebas lógicas (De escritorio)

A medida que se desarrolle cada uno de los componentes del traductor, se aplicarán las pruebas respectivas según las necesidades del MODRES. El procedimiento a seguir para realizar este tipo de pruebas es:

Establecer previamente su ejecución.
Realizar evaluación de cada componente (Analizadores léxico y sintáctico).
Observar y registrar los resultados.

Proyectos de pregrado

En esta última etapa de desarrollo del proyecto, se busca que los educandos de la escuela de ingeniería de sistemas, participen de la implementación de cada una de las etapas de desarrollo del traductor, la cual permitirá, mostrar un producto software que valide y verifique cada uno de los objetivos propuestos en el proyecto.

Descripción de resultados

La traducción de un sistema descrito en este lenguaje y su posterior conversión de sus clases a un motor de bases de datos orientado a objetos o relacional, tendrá en cuenta los siguientes componentes que se muestran en la figura 6.

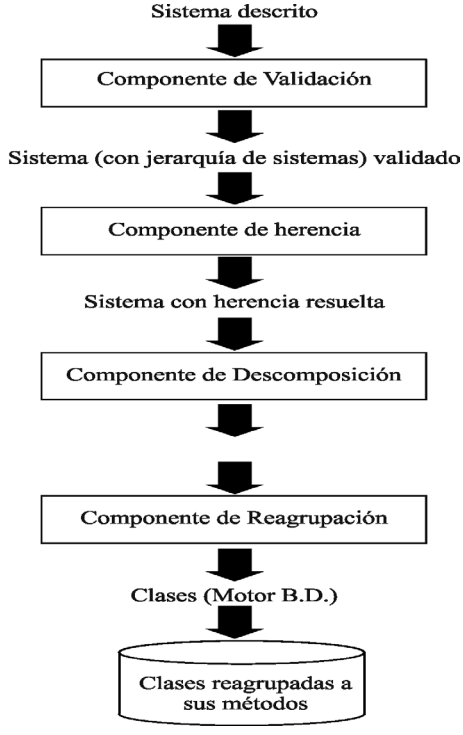

Figura 6. En esta figura se observan cada una de las etapas que deberá ejecutar el traductor con el objeto de traducir sistemas, en objetos o al modelo relacional.

A continuación se hace una breve explicación de las etapas plasmadas en la figura 6.

Componente de validación

En este componente el traductor reconoce y valida por medio de su gramática la definición del sistema, el cual está compuesto de clases y de un método en común a esas clases. Dicha validación consistirá en reconocer la definición de los atributos junto con sus tipos de dato. También deberá reconocer el cumplimiento de las reglas de integridad referentes a claves primarias y foráneas, y demás restricciones inherentes al MODRES. Y de la misma manera debe validar la pertenencia del método a ese sistema.

Componente de herencia

En este segundo componente el traductor resuelve la herencia propia de la jerarquía de sistemas (Subsistemas, Sistemas, Supersistema).

Componente de descomposición

Aquí separa las clases del método común, es decir, las clases quedan como estructuras separadas de su método. Esta separación se hace para construir una entrada apropiada a los motores de Bases de Datos Orientados a Objetos de la actualidad, incapaces de reconocer sistemas al estilo del MODRES.

Componente de Reagrupación

Este componente reconstruye cada una de las clases del sistema con su respectiva porción de método. Así, cada clase quedará con su propio método según el estándar ODMG, o sea con su propia funcionalidad. Tales clases no pueden ser independientes entre si puesto que deben ejecutarse todas en simultaneidad. Para implementar esta simultaneidad el motor de base de datos orientada a objetos o relacional, dispone de un conjunto de sentencias que permiten definir clases con sus respectivos atributos y su método. Aunque en realidad estas clases se definen primero en el traductor y luego se pasan a motor de base de datos mediante un archivo plano que contiene dicha sentencias.

Componente de entrega de clases al motor de bases de datos

En este componente el traductor entrega al motor de bases de datos cada una de las clases extraídas del sistema. Así, la implementación de las bases de datos orientadas a sistemas (MODRES) queda al nivel de las bases de datos orientadas a objetos o relacional.

Diseño interfaz gráfica usuario

La interfaz para el traductor habrá de seguir un estilo y un diseño tradicional. En la construcción de dicha interfaz se utilizará el entorno de desarrollo de Visual C#, como lo muestra la figura 7.

Figura 7. En esta figura se presenta la interfaz gráfica del traductor.

Pruebas del traductor

El procedimiento a seguir para realizar las pruebas es el siguiente:

Ejecutar el Intérprete.
Examinar el desempeño de los Analizadores (Léxico y Sintáctico).
Observar y registrar los resultados obtenidos.

Después de las pruebas debe ser establecida la conexión del traductor con Db4o para la entrega de las clases.

Proceso de prueba del Analizador Léxico

El examen del desempeño del Analizador léxico abarca los siguientes ítems:

El primero consiste en ingresar cadenas de caracteres válidos dentro de la gramática.
En el segundo ítem son ingresadas cadenas de caracteres no permitidos dentro de la gramática.
Por último son ingresadas cadenas de caracteres validos e inválidos.

Proceso de prueba del Analizador Sintáctico

Para desarrollar la prueba del Analizador sintáctico es necesario crear la siguiente clase en C#:
public class static void Main (String [] args)

```
{
   /* Inicia el analizador * /
 try
  {
    {
      parser p = new parser
      (new Lexer
      (new FileReader (args [0])));
      Object result = p.parse
().value;
    }
    catch (exception e)
    {
     e.printStackTrace();
    }
  }
}
```

Esta clase recibe como parámetro un archivo plano con las cadenas digitadas por el usuario con el cual realiza la conexión entre el analizador léxico y sintáctico.

Para el desarrollo de estas pruebas se deberá considerar la sintaxis utilizada en la jerarquía de sistemas, o sea en la descripción de los Supersistemas, Subsistemas y Sistemas propios de la gramática.

Si los resultados generados por el traductor son acordes con lo esperado se procederá a la conexión y entrega de los Supersistemas, Subsistemas y Sistemas al motor de base de datos, a través del archivo plano mencionado.

Proceso de prueba de conexión con el motor de base datos

La conexión con el motor de base de datos es realizada por el traductor después de verificar la correcta definición de los Supersistemas, Sistemas, Subsistemas, así como la adecuada manipulación de datos mediante instrucciones propias del traductor.

Conclusiones

El traductor ofrece una alternativa más evolucionada que los lenguajes de datos orientados a objetos, puesto que permitirá describir sistemas en términos de clases. Además permitirá implementar en el computador el Modelo de Datos Reorientado a Sistemas (MODRES), con la cual queda demostrado que si es posible describir sistemas, subsistemas y clases y no como todavía hoy en día se está trabajando, con el modelo relacional u objeto. Con lo anterior se podrá resolver el problema que presentan los lenguajes como Java, Visual C++, C#, Visual Basic y otros, que solo alcanzan a describir estructuras. También superará a los lenguajes de objetos como OQL que solo llegan hasta describir clases. Con el traductor se da un paso más allá describiendo jerarquías de sistemas. Por otro lado el traductor coadyuvará a resolver el problema de la desintegración de la organización.

También es bueno afirmar que a través de esta investigación queda la posibilidad abierta, de que en un futuro no lejano, dicho traductor, pudiera convertirse en un motor de base de datos que permita describir sistemas (SGBDOS).

Referencias

[1] R.G.G. Cattell. (2000). *The. Object Data Standard:* ODMG 3.0.San Francisco, California – Estados Unidos. Morgan Kaufmann Publisher. 280 p. ISBN: 1-55860-647-4.

[2] Albarracín Ferreira, Jaime (2006). *Integración de datos y procesos en las organizaciones mediante un Modelo de Datos Reorientado a Objetos.* Tesis Doctoral (Doctor en Informática). Oviedo (España): Universidad de Oviedo. 192 p.

[3] Gálvez Rojas Sergio y MORA MATA Miguel Ángel. (2005). *Java a Tope: Compiladores (Traductores y Compiladores). Dpto. de Lenguajes y Ciencias de la Computación E.T.S. de Ingeniería Informática,* Universidad de Málaga. Málaga - España. 292 p. ISBN: 84-689-1037-6.

[4] Merche Marquez (2002). Bases de datos orientadasaobjetos. http://www3.uji.es/~mmarques/e16/teoria/cap2.pdf [Consulta: 28/05/2009].

[5] Pressman S. Roger. (2002). *Ingeniería del Software, un enfoque práctico.* 5 ed. Madrid - España. MacGraw-Hill. 601 p. ISBN: 84-481- 3214-9.

[6] Contreras Herazo Francisco, García Payares Arturo Segundo y Cala Uribe Javier Norberto. 2008. *Construcción de los componentes ODL y OQL para un sistema generador de Bases de Datos Reorientada a Objetos (SGBDRO).* Tesis de Pregrado (Ingeniero de Sistemas). Bucaramanga (Colombia). Universidad Industrial de Santander. 145 p.

[7] Alfonseca Moreno Manuel, Echeandia Marina de la cruz, Ortega De La Puente Alfonso y Pulido Cañabate Estrella. (2006). *Compiladores e Intérpretes.* Madrid – España. Prentice – Hall. 361 p. ISBN: 84-205-5031-0.

[8] Aho Alfred V. (2008). *Compiladores, Principios, técnicas y herramientas.* 2 ed. México – México. Pearson Addison Wesley. 1009 p. ISBN: 978-970-26-1133-2.

[9] Booch Grady. (1996). *Análisis y Diseño Orientado a Objetos.* 2 ed. México – México. Pearson Addison Wesley Longman. 638 p. ISBN: 968-444-352-8.

[10] Garcia Perez Fernando, Chamorro Atance Félix y Molina Lopez José Manuel. (2000). *Informática de Gestión y Sistemas de Información.* 1 edición. Madrid – España. Mc Graw Hill. 239 p. ISBN: 84-481-2767-6.

[11] Grune Dick, E. Bal Henry, Jacobs J.H. Ceriel y G. Langendoen Koen. (2007). *Diseño de Compiladores Modernos.* Madrid – España. Mc Graw Hill. 682 p. ISBN: 978-84-481-5656-5.

[12] Joyanes Aguilar Luís. (1996). *Programación Orientada a Objetos.* Madrid – España. Mc Graw Hill. 658 p. ISBN: 84-481-0590-7.

[13] Lauden Kenneth C. (2004). *Construcción de Compiladores, Principios y Práctica.* México – México. THOMSON PARANINFO. 592 p. ISBN: 9789706862990.

[14] Ruiz Catálan Jacinto. (2010). *Compiladores Teoría e Implementación.* Madrid – España. Alfaomega Grupo Editor. 448 p. ISBN: 978-607-7854-68-5.

[15] Sudkamp Thomas A. (1988). *Languages and Machines.* Canadá. Addison - Wesley. 442 p. ISBN: 0-201-15768-3.