# Sistema para la solución de problemas de cómputo basado en el Método de los Grafos Dicromáticos

Yosveni Escalona Escalona
yescalona@uci.cu
*Universidad de las Ciencias Informáticas*

Alejandro Romero Vega
aromero@mecanica.cujae.edu.cu
*Instituto Superior Politécnico José Antonio Echeverría*

Sergio A. Marrero Osorio
smarrero@mecanica.cujae.edu.cu
*Instituto Superior Politécnico José Antonio Echeverría*

Rafael Rodrígez Puente
rafaelrp@uci.cu
*Universidad de las Ciencias Informáticas*

## RESUMEN

El presente artículo propone una herramienta de resolución de problemas de cómputo basado en el Método de los Grafos Dicromáticos. Además, se describe la arquitectura diseñada, así como las herramientas y tecnologías empleadas para la implementación de la solución. La propuesta constituye un aporte al valor práctico del Método de los Grafos Dicromáticos tomando en cuenta la integración con otras herramientas existentes para la aplicación del dicho métodos en problemas ingenieriles.

**PALABRAS CLAVE:** Método de los grafos dicromáticos, arquitectura de software, resolución de problemas de cómputo.

## INTRODUCCIÓN

En la actualidad se le concede gran importancia a la solución de problemas de cómputo, debido a que esto contribuye al desarrollo científico y tecnológico. Dar solución a un problema equivale a obtener un algoritmo y ejecutarlo.

En determinadas situaciones del mundo real, la resolución de problemas de cómputo (RPC) a través de modelos matemáticos requiere un análisis relativamente simple. Sin embargo, existen problemas con un nivel de complejidad tal que se requieren métodos más eficientes para su solución, entre los cuales se puede mencionar el Método de los Grafos Dicromáticos (MGD) (Martínez-Escanaverino, García y Ortiz 1997; Martínez-Escanaverino 2000; Martínez-Escanaverino et al. 2001; Martínez-Escanaverino y Martínez 2005).

El MGD ha sido divulgado en eventos (Martínez-Escanaverino et al. 2001, Martínez-Escanaverino y Martínez 2005), publicaciones (Rivero 2000, Marrero-Osorio y Martínez-Escanaverino 2009, Marrero-Osorio 2009) y cursos sobre RPC, simulación, optimización y diseño racional en ingeniería mecánica. El mismo permite aprovechar las particularidades teóricas de los grafos dicromáticos durante el proceso de solución del problema, pasando por un proceso rigurosamente ordenado a partir de la caracterización del problema y su correcta formulación hasta la obtención del algoritmo.

La representación de modelos matemáticos en ingeniería puede generar grafos de gran complejidad, lo que dificulta el análisis de forma manual. A pesar de que existen herramientas de edición de grafos como la seleccionada en (Rivero 2000), que facilita en cierto grado la utilización del MGD, no se cuenta con un sistema informático que automatice todos los pasos de dicho método; lo que limita en cierta medida su valor práctico. Por ejemplo, en la tesis de Llamos (2000), enfocada al diseño óptimo de cajas reductoras, a pesar de manipular un modelo matemático con 444 vértices en total, no se llegan a procesar otros modelos matemáticos más completos que consideren otros aspectos de las máquinas, como los económicos, los medioambientales. La razón de ésta insuficiencia es precisamente el límite que impone la complejidad, al no contar con facilidades para enfrentarla.

El objetivo principal del presente artículo es dar a conocer la forma general que podría tener la primera versión de una herramienta informática que automatice todos los pasos del MGD. Se describe brevemente el MGD, se presenta un conjunto de definiciones necesarias sobre la propuesta de solución, se explica la arquitectura base del sistema, se desarrolla un caso de estudio y finalmente se ofrecen las conclusiones.

## MÉTODO DE LOS GRAFOS DICROMÁTICOS

En la Figura 1 se muestra el esquema de ejecución del MGD. Inicialmente se construye el grafo del modelo, cuyos vértices representan las variables con un color y las ecuaciones (relaciones) con otro y las aristas simbolizan cuáles variables se encuentran en cada ecuación.

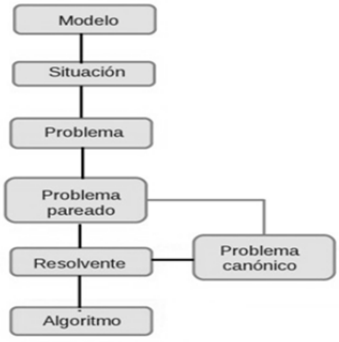
Una vez planteado un problema de cómputo determinado, se descartan de dicho grafo las variables de entrada (datos), quedando transformado así en el grafo de la situación. A partir de este, suprimiendo las componentes conexas (islas) que no contengan variables de salida, se obtiene el grafo del problema; el cual es sometido a un pareo que orienta hacia su correspondiente variable a una sola de las aristas conectadas con cada ecuación, de modo que el mismo sea un pareo máximo.

Así pasa a obtenerse una nueva representación llamada grafo del problema pareado. Posteriormente, se obtiene el grafo del resolvente, asignándole orientación, de variable a la ecuación, a las aristas que aún no la tienen. Para los problemas que no pueden ser caracterizados plenamente, a partir de este grafo del resolvente es necesario obtener un nuevo grafo llamado problema canónico (si el problema tiene un pareo perfecto ya se encuentra en su forma canónica) y de este se llega al resolvente definitivo.

A partir de dicho resolvente (que ya se encuentra en su forma canónica) se alcanza el grafo del algoritmo descartando los caminos que no conducen a ninguna de las variables de salida definidas al plantear el problema. A través de éste método, siguiendo los pasos indicados

en el grafo del algoritmo (figura 1), se puede dar solución al problema inicialmente planteado.

Figura 1: Esquema del MGD




Fuente: Martínez Escanaverino, curso de modelación, simulación y optimización.

## METODOLOGÍA

A continuación se enuncian algunas definiciones que se tuvieron en cuenta para la implementación de la solución. Además, se explica la metodología de desarrollo utilizada, la arquitectura base y las herramientas y tecnologías. Para el desarrollo del presente trabajo, se utilizó como metodología de desarrollo Rational Unified Process (2007). Esta metodología permite agrupar las actividades en grupos lógicos como son: modelo del negocio, requerimientos, análisis y diseño, implementación y prueba.

En el modelo del negocio se describieron los procesos del negocio, identificando las actividades que deberán ser automatizadas. Los requerimientos representan las funcionalidades a implementar lo cual coincide con el análisis y diseño realizado a partir de la elaboración de la arquitectura base. En la implementación se precisó la organización de las clases y objetos a partir de la implementación de los requerimientos previamente identificados.

La arquitectura de software es el conjunto de decisiones significativas acerca de la organización de un sistema; la selección de los elementos estructurales a partir de los cuales se compondrá el sistema y sus interfaces; junto con la descripción del comportamiento de dichas interfaces en las colaboraciones que se producen entre los elementos del sistema, la composición de esos elementos estructurales y de comportamiento para formar subsistemas de tamaño cada vez mayor y el estilo o patrón arquitectónico que guía esta organización: los elementos y sus interfaces, las colaboraciones y su composición (Bass, Clements y Kazman 2003; Garlan y Perry 1995; Rumbaugh, Jacobson y Booch 2004).

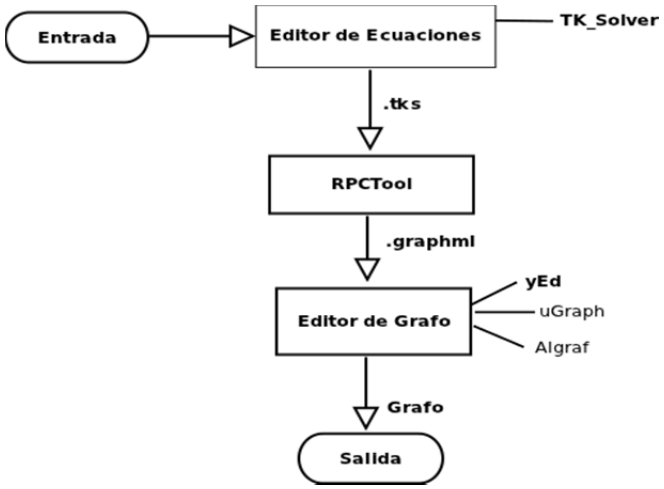
Para el desarrollo del sistema propuesto se utilizó el lenguaje de programación Java y el entorno de desarrollo Netbeans IDE, en su versión 7.1.

## RESULTADOS

Se desarrolló la primera versión de un sistema que automatiza los procesos del MGD, tomando en cuenta la arquitectura definida para el mismo así como la integración con otras herramientas.

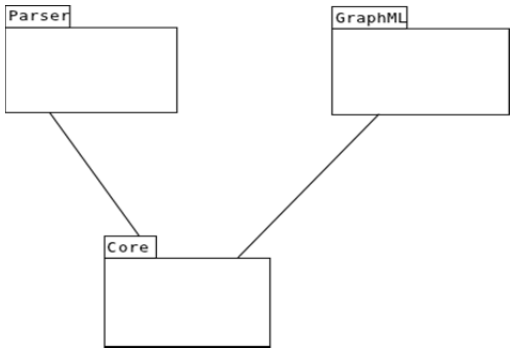
La arquitectura base del sistema se obtuvo a partir del flujo de información definido en el MGD (Martínez-Escanaverino 2000), tomando en cuenta las entradas y salidas de cada una de sus etapas. La propuesta de solución la componen tres herramientas: El editor de ecuaciones, en este caso, TK Solver, la herramienta de resolución de problemas (RPCTool) y un editor de grafos propuesto por yWork . La Figura 2 representa la integración y el flujo de información entre cada una de estas herramientas.

Figura 2: Flujo de información entre las aplicaciones




Para dar solución al problema, como se observa en la figura 2, se parte de la obtención del modelo matemático utilizando el editor de ecuaciones, lo que constituye la salida para esta etapa. Luego, a partir de este modelo y a través del RPCTool se obtienen los ficheros graphml con la información de cada uno de los grafos que define el MGD. Estos grafos son visualizados utilizando el editor de grafo, en este caso, yEd. A continuación se muestran los tres módulos que compone al RPCTool (figura 3).

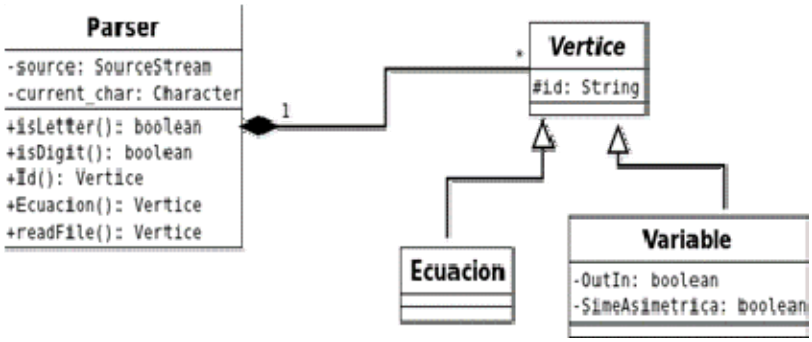
Figura 3: módulos del RPCTool.




El módulo *Parser* contiene la estructura de las clases del sistema que permiten la lectura y procesamiento del fichero de texto, estos contienen el modelo matemático descrito para el

problema de cómputo determinado. Este fichero es el obtenido a partir del uso del editor de ecuaciones. La Figura 4 representa el diagrama de clases para este módulo.

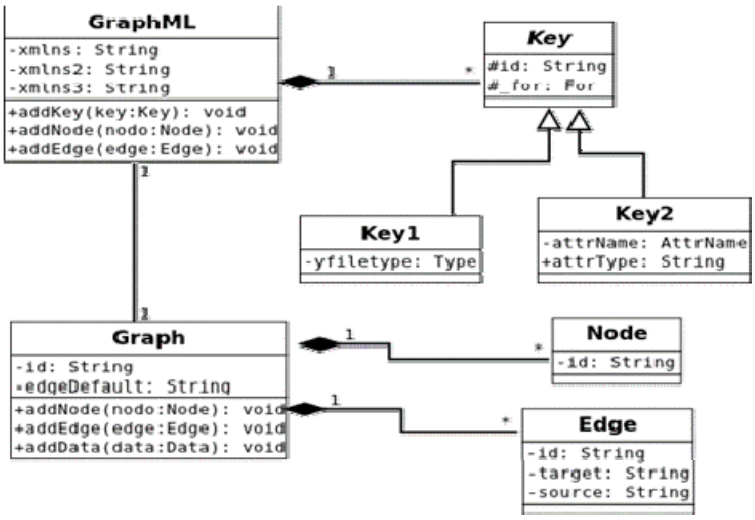
Figura 4: Diagrama de clases del módulo Parser




La clase Vértice representa la generalización de los elementos del modelo matemático cuya equivalencia puede ser una ecuación (relación dentro del modelo matemático) o una variable involucrada en cada relación. El atributo id es el nombre o valor que toma cada uno de los vértices. La Ecuación es una especificación de un vértice, representa las relaciones en el modelo matemático. La clase Variable es la especificación de un vértice, cuya representación equivale a las variables involucradas en cada relación del modelo matemático. Además del valor que toma mediante el id descrito en la clase Vértice, también posee propiedades de entrada o salida y su simetría como se describe en el MGD. Por último, la clase Parser constituye la principal del módulo, de esta depende la lectura del fichero y el procesamiento; logrando como salida el conjunto de vértices identificados para el modelo.

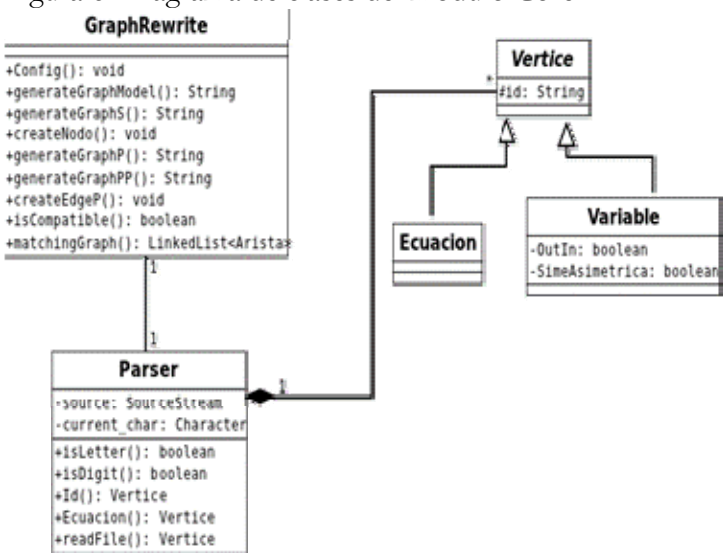
Por otra parte, el módulo GraphML contiene la estructura de clases que componen el fichero graphml, constituyendo la salida para el mismo. Esta información es utilizada por el editor de grafos para la representación de los grafos dicromáticos. La figura 5 representa el diagrama de clases para este módulo.

Figura 5: Diagrama de clases del módulo GraphML



La clase GraphML representa jerárquicamente el nivel principal dentro del fichero graphml. Contiene en su estructura el grafo. La clase Graph representa el grafo del problema propuesto, o sea, el modelo matemático que se desea resolver. Contiene los nodos y las relaciones entre cada uno ellos, denominadas aristas. Finalmente, el módulo Core representa el núcleo de la aplicación permitiendo la creación del grafo dicromático a partir de la integración con el editor de ecuaciones y el editor de grafos, como se muestra en la figura 2. La figura 6 representa el diagrama de clases.

Figura 6: Diagrama de clases del módulo Core




La clase GraphRewrite es la encargada de recibir el conjunto de vértices identificados por el Parser a partir de los cuales realiza el proceso de transformaciones que describe el MGD. Además, crea y devuelve los grafos asociados al modelo.

## CASO DE ESTUDIO

Para demostrar la aplicabilidad y valor práctico de la propuesta descrita en el presente trabajo, a continuación se describe un caso de estudio. El mismo aborda el problema de la escalera de mano enunciado en Martínez-Escanaverino, García y Ortiz (1997).

Se desea determinar el coeficiente de fricción μ2 entre la escalera y la pared. La G representa el peso de la escalera, N1 y N2 las fuerzas normales, f1 y f2 las fuerzas de fricción ejercidas por la pared y el piso sobre la escalera. Se conoce el peso de la escalera N2, el coeficiente de fricción μ1 y el ángulo máximo de inclinación α respecto a la vertical entre la escalera y la pared. A continuación se representa el modelo matemático para el presente problema.

1) $N_2 - f_1 = 0$
2) $N_1 + f_2 - G = 0$
3) $\left(N_1 - \frac{1}{2}G\right)\tan \propto -f_1 \ = 0$
4) $f_1 - \mu_1 N_1 = 0$
5) $f_2 - \mu_2 N_2 = 0$

De acuerdo a la descripción del problema anterior se tienen como variables de entrada N2, α, μ1 y como variable de salida a μ2. Determinadas por: E = $\{N_2, \alpha, \mu_1\}$ y S = $\{\mu_2\}$.

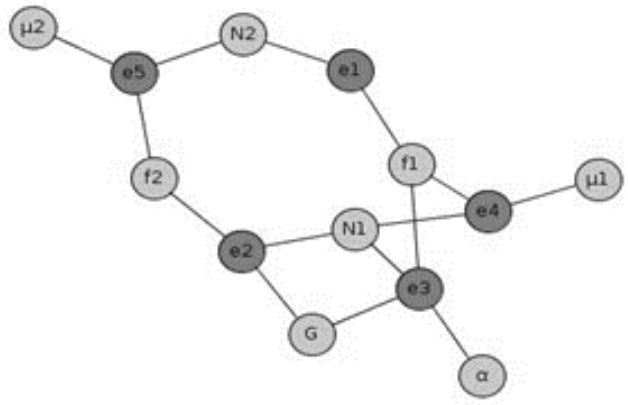
***Pasos para la solución del modelo matemático utilizando el sistema***

A continuación se ofrecen una serie de pasos lógicos para resolver el problema a partir de un modelo matemático según como describe el MGD.

- Definir el modelo matemático utilizando el editor de ecuaciones.
- Definir conjuntos de variables de entradas y salidas y las simetrías presentes en las relaciones.
- Generar Grafo del Modelo
- Generar Grafo de la Situación
- Generar Grafo del Problema
- Generar Grafo del Problema Pareado
- Generar Grafo Resolvente
- Generar Grafo del Algoritmo

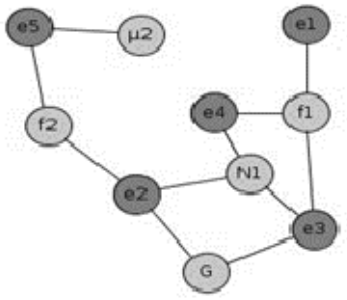
A continuación se muestra la resolución del problema de la escalera de manos a través de la utilización del sistema implementado. Una vez que se defina el modelo matemático en el editor de ecuaciones se carga el archivo o fichero; el usuario tiene la posibilidad de generar paso a paso cada una de las etapas del MGD u obtener el grafo del algoritmo. Para una mejor comprensión de esta investigación se propone ir generando cada una de las etapas por separado. En la figura 7 se muestra el grafo del modelo que se genera a partir del conjunto de relaciones matemáticas que describen lo que ocurre físicamente en el caso analizado.

Figura 7: grafo del modelo



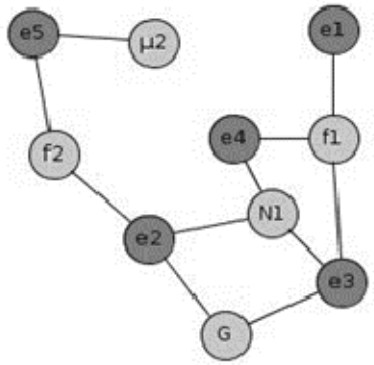

Los nodos de color amarillo representan las variables del modelo matemático y los de color verde las relaciones o ecuaciones. Las variables que están involucradas en una o varias ecuaciones se visualizan mediante una arista o relación. Teniendo en cuenta lo descrito por el MGD y la declaración de variables de entrada E = {N2, α, μ1} se obtiene el siguiente grafo de la situación como muestra la figura 8.

Figura 8: grafo de la situación



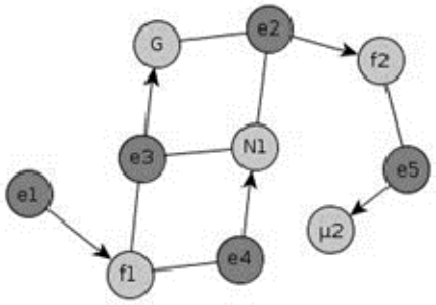

Después de obtener el grafo de la situación es posible generar el grafo del problema descartando las componentes no conexas (denominadas islas) que no contengan variables de salidas, según como lo describe el MGD. Para este ejemplo al aplicar las transformaciones y pasar de la situación al problema no existen componentes no conexas por tanto, el grafo del problema es igual al de la situación como se muestra en la figura 9.

Figura 9: grafo del problema



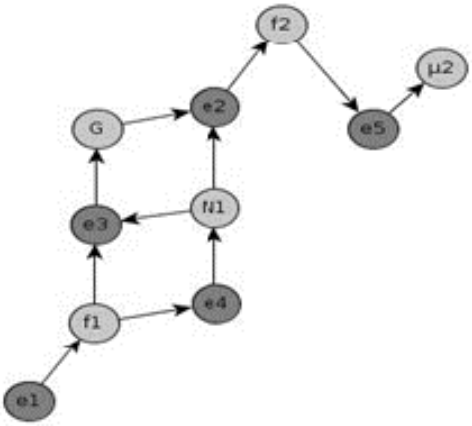

A partir del grafo del problema es posible generar el grafo del problema pareado, comprobando que el mismo sea compatible y realizable a partir de la existencia o no de deficiencias y variables libres como plantea el MGD. El grafo de la Figura 10 es el resultado de este proceso.

Figura 10: grafo del problema pareado



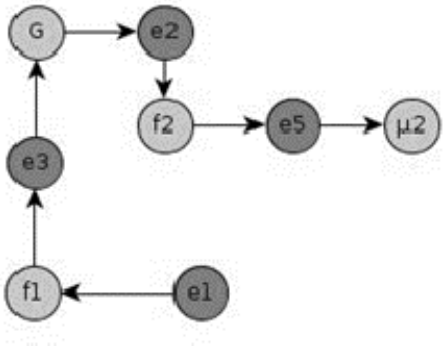

Obtener el grafo resolvente consiste en generar un grafo en el cual se orientan las aristas que permanecían sin orientación hasta el momento. La figura 11 muestra el resultado.

Figura 11: grafo resolvente




Finalmente, a partir del grafo resolvente es posible obtener el grafo del algoritmo. Como se aprecia en la figura 12 se obtuvo un árbol, lo que equivale a decir que la solución es cerrada, en el sentido de que es definitiva, concluyente. El grafo muestra la secuencia lógica para resolver el problema, pudiendo estar asociado a un problema de simulación o de optimización.

Figura 12: grafo del algoritmo




## DISCUSIÓN

La herramienta desarrollada aumenta el valor práctico del MGD, brindando la posibilidad de que los problemas sean resueltos sin la necesidad de procesar la información de forma manual y disminuyendo la complejidad de los procesos que impone el MGD.

A partir de esto, se facilita la obtención de la solución a problemas más complejos que involucren grandes modelos matemáticos. Además, constituye un apoyo a los cursos de pregrado y posgrado que sobre el MGD reciben los estudiantes e ingenieros, permitiendo que los resultados que se obtienen mediante la resolución de problemas puedan ser comprobados y que los problemas resueltos tengan un mayor alcance.

## CONCLUSIONES

A partir del desarrollo de este trabajo, se arribó a las siguientes conclusiones:

- ✓ El resultado obtenido aumenta el valor práctico del MGD.
- ✓ La herramienta desarrollada disminuye la complejidad de aplicación del MGD contribuyendo al mayor uso del mismo.
- ✓ La solución propuesta brinda la posibilidad de resolver problemas complejos con menor esfuerzo.
- ✓ La herramienta se puede utilizar como apoyo en cursos sobre RPC, simulación, optimización e investigaciones doctorales.

## REFERENCIAS

Martínez-Escanaverino, J. (2000) Dichromatic Graphs: A Tool for the Algorithmic Education of Mechanical Engineers in ASME Design Engineering Technical Conferences & computers and Information in Engineering, DETC 2000. Baltimore, Maryland: Amer Society of Mechanical.

Martínez-Escanaverino, J., Llamos Soríz, J.A., García Toll, A. y Ortiz Cárdenas, T. (2001) Rational design automation by dichromatic graphs en ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering, DETC'01.

Martínez-Escanaverino, J. y Martínez Forte, L. (2005) A problem solving rationale for conceptual design in engineering. in Proceedings of the 6th International Conference on Computer-Aided Industrial Design & Conceptual Design - CAID&CD 2005.

Martínez-Escanaverino, J., García Toll, A. y Ortiz Cárdenas, T. (1997) Algorítmica del diseño mecánico. *Ingeniería Mecánica*, vol 0, 31-37.

Rivero Llerena, G. (2000) Descifrado geométrico de transmisiones de engranaje por tornillo sinfín cilíndrico. *Ingeniería mecánica*, Vol. 3 (1).

Marrero-Osorio, S.A.y Martínez-Escanaverino, J. (2009) Diseño paramétrico de pinzas de fricción. Ingeniería Mecánica, vol.12 (1): p. 37-48.

Marrero-Osorio, S.A. (2009) Diseño Paramétrico Basado en Modelos Matemáticos. Caso de estudio: Máquinas para la Construcción Sostenible de Viviendas. Tesis Doctoral, Tutor: Martínez-Escanaverino, J. Departamento de Mecánica Aplicada, Facultad de Ingeniería Mecánica, Instituto Superior José Antonio Echeverría.

Llamos Soríz, J.A. (2000) Diseño óptimo de cajas reductoras para molinos de la caña de azúcar. Tesis Doctoral, Tutor: Martínez-Escanaverino, J., Departamento de Mecánica Aplicada, Instituto Superior José Antonio Echeverría.

Rational Unified Process (2007) Network Dictionary, p. 402.

Bass, L., Clements, P., y Kazman, R. (2003) Software Archictecture in Practice. Second Edition ed., SEI Series in Software Engineering. Pearson Educación.

Garlan, D., Perry, D. (1995) Introduction to the Special Issue on Software Architecture. IEEE Transactions on Software Eng. Vol. 21:269-274.

Rumbaugh, J., Jacobson, I., Booch, G. (2004) The Unified Modeling Language Reference Manual. Addison Wesley. 2 edition ed. The Addison-Wesley Object Technology.

Java, O. Documentation Java. 2012 [Consultado el 2012; Available from: http://www.oracle.com/es/technologies/java/index.html.

IDE, N. Documentation, Training & Support 2012 [Consultado el 2012; Available from: http://netbeans.org/.

Universal Techical Systems, I. TK Solver 5.0 Premiun 2012 [Consultado el 2012; Available from: http://www.uts.com/ItemSummary.asp?ItemID=0100-50-0010-00.

yWork, t.d.c. yEd Graph Editor. 2012 [Consultado el 2012; Available from: http://www.yworks.com/en/products_yed_about.html.