

Proyecto ARGOS: Un SCADA en Software Libre

Alejandro Piña, Rafael Aiquel

Resumen - Un software de SCADA es un conjunto de herramientas informáticas de uso extendido en la automatización de procesos industriales. El propósito del proyecto Argos es ofrecer un SCADA desarrollado bajo la filosofía de proyectos de software libre. Con esto se busca promover el estudio del núcleo de un sistema SCADA y permitir el desarrollo del mismo hasta convertirse en una alternativa libre para la automatización industrial. El SCADA está implementado en plataformas con GNU/Linux y los componentes de software han sido desarrollados usando el lenguaje de programación C/C++, se utiliza XML como formato para los archivos de configuración, además de varias utilidades y librerías de software libre.

Palabras claves – Linux, Scada, Software Libre

I. INTRODUCCIÓN

Entre los elementos fundamentales de un SCADA (Control Supervisor y Adquisición de Datos, por sus siglas en inglés) se encuentran los componentes de software que realizan las tareas necesarias para monitoreo y control de procesos desde un centro de computo [1]. Dichos procesos pueden ser de distinta índole, pero comparten características comunes en estado transitorio, como lo son: la recolección de información desde diversos dispositivos, la toma de decisión de acuerdo a algún algoritmo de control y la asignación de órdenes a elementos con capacidad de ejecutar una acción.

El software de un SCADA le proporciona a los usuarios un conjunto de herramientas informáticas con las cuales se pueda diseñar, desarrollar, implementar y mantener sistemas para la supervisión, control y adquisición de datos, permitiendo de esta manera automatizar procesos industriales, integrar los distintos niveles de información, además de brindar la posibilidad de crear interfaces gráficas entre los operadores y las máquinas [2].

A.P. y R.A. están con la Industria Venezolana de Aluminio VENTALUM, Gerencia de Investigación y Desarrollo, Av. Fuerzas Armadas, Zona Industrial Matanzas, Puerto Ordaz 8050, Venezuela, E-mail: alejandro.pina@venalum.com.ve, rafael.aiquel@venalum.com.ve

A.P. y R.A. están con la Universidad Católica Andrés Bello, Nucleo Guayana, Escuela de Ingeniería Informática, Puerto Ordaz 8050, Venezuela, E-mail: apina@ucab.edu.ve, raiquelg@ucab.edu.ve

El software libre permite que el usuario pueda usar, probar

y modificar el software sin restricciones. La filosofía de desarrollo del software libre permite que, gracias al trabajo cooperativo, un proyecto evolucione. Las libertades del software libre garantizan que un proyecto no pueda ser cancelado unilateralmente por las razones que fuesen, es decir, siempre y cuando existan interesados en continuar con el proyecto, éste seguirá desarrollándose [3]. En contraste, un proyecto de software propietario, si los promotores del mismo decidieran abandonarlo, su desarrollo no continuará.

II. SCADA EN CÓDIGO ABIERTO

A. Arquitectura

El proyecto Argos está siendo desarrollado para ser un SCADA con código abierto, asimismo, las herramientas que Argos proporciona actualmente sientan una base (con funcionalidades básicas) para implementar sistemas de supervisión en procesos automatizados.

La arquitectura clásica mostrada en la Figura 1, muestra como interactúan los componentes del software para formar un SCADA. Estos están distribuidos en la totalidad de la red de supervisión, aunque pueden existir aplicaciones en las que todos los componentes de software se ejecuten dentro de la misma estación de trabajo.

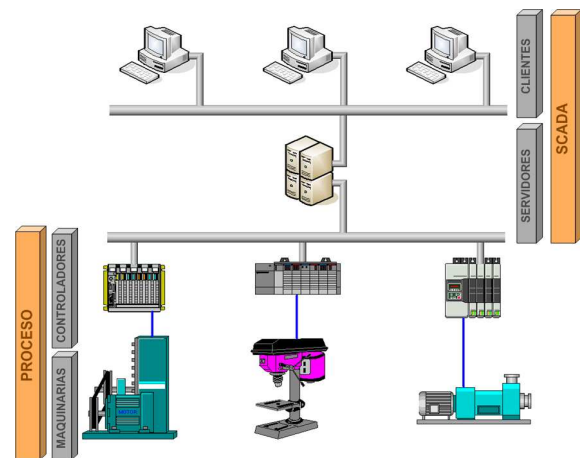


FIG 1. Arquitectura de un proceso automatizado

Argos se ha diseñado tomando en cuenta experiencias obtenidas durante el desarrollo e implementación del sistema supervisor de celdas de VENALUM [4], creando así una arquitectura (Figura 2) que permite adaptarse a los distintos esquemas de automatización moderna, en donde cada componente de software cuenta con estructuras de datos de alto rendimiento que operan de manera distribuida ya sea en una plataforma de red o en un mismo PC.

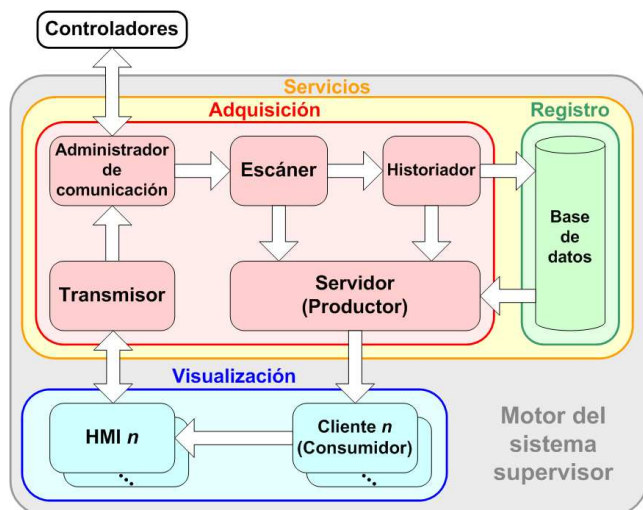


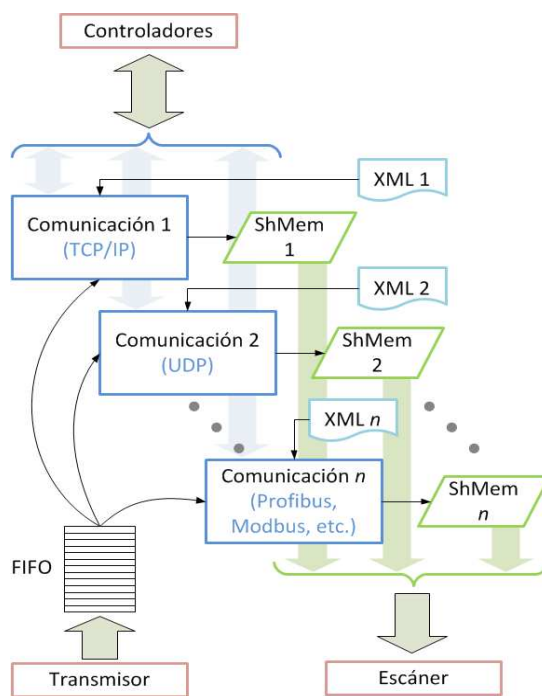
FIG 2. Organización de Argos

Entre las principales herramientas que se proporcionan en el SCADA de código abierto Argos, se encuentran los siguientes:

- **Comunicación**, estos procesos se encargan de establecer la comunicación con los equipos controladores de campo y pueden ejecutarse en uno o varios nodos. La importancia de estos servicios radica en la capacidad de manejar una comunicación efectiva con la gran diversidad de instrumentos presentes en un proceso automatizado[1], la realidad nos demuestra que estos equipos en su mayoría poseen distintos protocolos de comunicación que en algunos casos pueden ser cerrados o propietarios, mientras que otros pueden ser abiertos y estandarizados. En Argos se vienen desarrollando servicios con la capacidad de entender diversos protocolos y de esta manera organizar y centralizar las comunicaciones industriales de una planta. En la Figura 3 se observa la interacción de múltiples administradores de comunicación donde cada uno puede entender un protocolo distinto y manejar uno o más dispositivos de campo para luego a través de mecanismos de POSIX IPC

(Interprocess Communications) poder entregarle los datos a las demás herramientas del SCADA.

- **Escáner**, convierte los registros de todos los controladores en unidades de ingeniería, que posteriormente serán mostrados a los usuarios finales. Este servicio es uno de los que requiere mayor procesamiento ya que se encarga de recolectar los datos de cada una de las memorias de intercambio que están siendo actualizadas por los administradores de comunicaciones, para luego crear y modificar constantemente la base de datos de tiempo real en la cual se encontrarán las variables del proceso y las alarmas del sistema (Figura 4). Al igual que los demás servicios del SCADA Argos, el escáner se configura a través del procesamiento de ficheros en formato XML (Extensible Markup Language), los cuales tienen la información necesaria que permite el flujo de datos entre los múltiples servicios usando IPC.



TCP = Transmission Control Protocol

IP = Internet Protocol

UDP = User Datagram Protocol

FIFO = First In, First Out

Profibus, Modbus = Protocolos de Comunicación Industrial

FIG 3. Administradores de Comunicación

- *Historiador*, es un proceso configurado para almacenar información de manera permanente, principalmente usado para contar con gráficos de tendencias e históricos de alarmas y eventos (Figura 5). Este servicio toma la información a una frecuencia configurable desde la base de datos de tiempo real, con esto se encarga de mantener un buffer (Figura 6) para cada una de las variables que se están registrando, posteriormente hace el volcado de los datos en un medio de almacenamiento permanente (Figura 7), esto a través de un servicio dedicado a guardar la información utilizando para ello el motor de base de datos PostgreSQL.

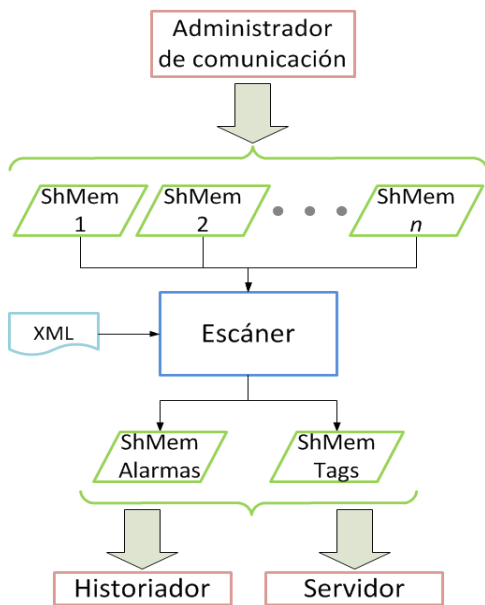


FIG 4. Escáner del SCADA

- Los procesos *Servidores* y *Cientes* se encargan de la transferencia de información, ya sea entre nodos localizados remotamente, o que se estén ejecutando en el mismo servidor (Figura 8), para que pueda ser presentada al operador. Actualmente Argos se basa en la filosofía de productor-consumidor para disminuir el tráfico en la red debido a los grandes volúmenes de información que se manejan en este tipo de sistemas, haciendo uso de grupos multicast los procesos productores ponen a disposición los datos en la red, para que así todos los consumidores que estén ligados al grupo puedan obtener la información que necesitan y que haya sido configurada a través de los ficheros en XML.

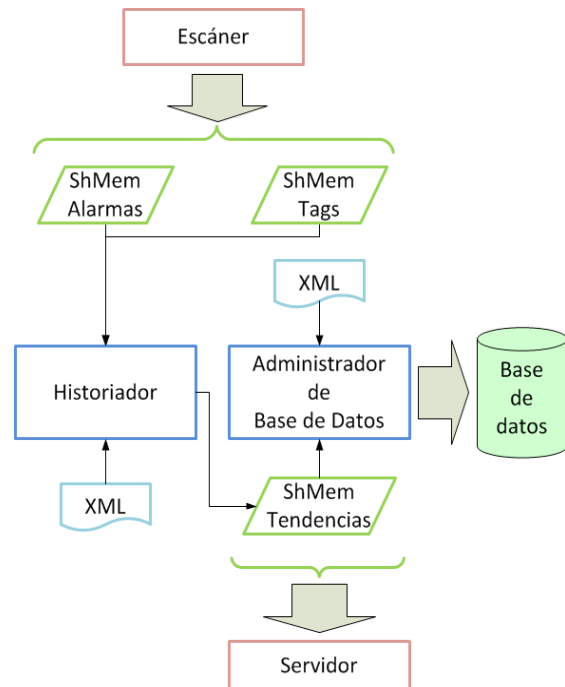


FIG 5. Registro de Alarmas y Tendencias

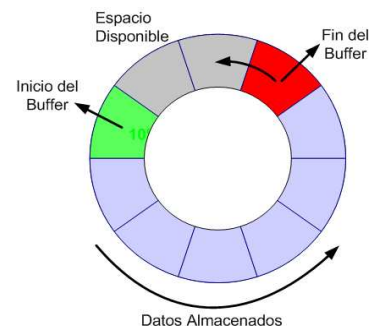


FIG 6. Buffer Circular de Variables del Proceso

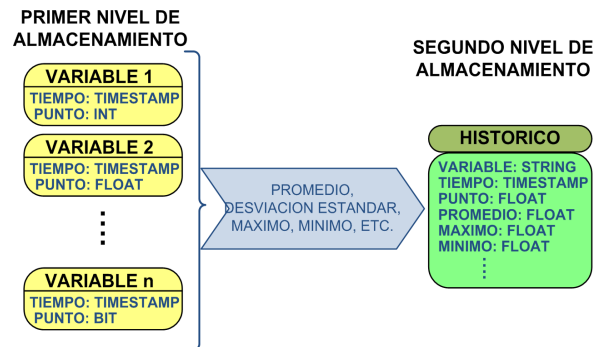


FIG 7. Modelo de Almacenamiento

- *Transmisor*, su única función será escribir registros en el controlador. Este proceso escucha las peticiones a través de conexiones TCP con los servicios HMI, los cuales canalizan los requerimientos de escritura de los usuarios del sistema SCADA. Cuando el transmisor recibe la petición coloca un mensaje en la estructura de tipo FIFO (POSIX) correspondiente a cada administrador de comunicación, que a su vez será el encargado de la operación de escritura en el instrumento de campo.

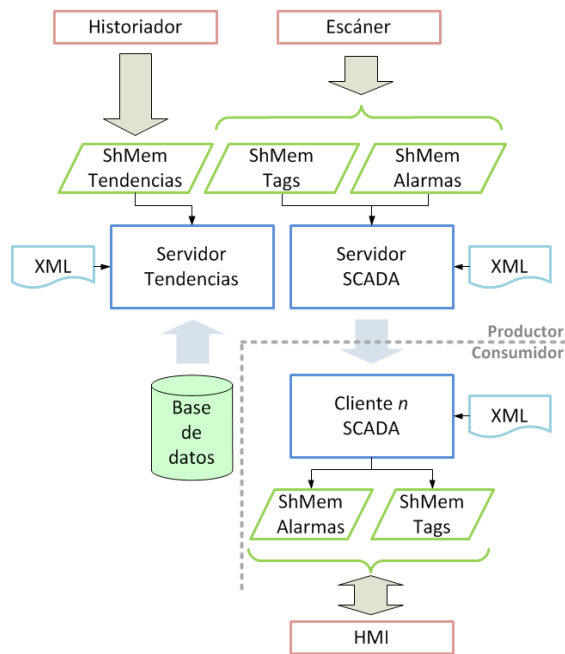


FIG 8. Relación Productor-Consumidor

- Por último, los procesos *HMI* serán los encargados de desplegar la información adquirida a los usuarios finales, mediante distintos recursos gráficos, tal y como se muestra en la Figura 9.

Para que el sistema supervisor a implementar pueda contar con todas las variables del proceso, se ha diseñado una base de datos con un mínimo tiempo de acceso. Esta estructura de almacenamiento debe permitir búsquedas de datos de forma rápida y precisa, por tal razón se decidió utilizar una estructura de datos de tipo Tabla Hash, la cual posee las características idóneas para satisfacer los requerimientos del sistema.

Una tabla *hash* (Figura 10) es un conjunto arbitrario de elementos agrupados sin ningún tipo de ordenamiento

físico aparente. Esto disminuye los tiempos de inserción de elementos (en nuestro caso variables del proceso) debido al ahorro en los procesos de ordenamiento, que normalmente son efectuados en otras estructuras de datos para optimizar los tiempos de búsqueda. El método de búsqueda de hashing no es basado en comparaciones, en lugar de navegar por las estructuras cotejando palabras claves con las claves en los elementos, se intentan ubicar elementos en una tabla directamente haciendo operaciones aritméticas para transformar claves en direcciones sobre la tabla a través de las funciones hash accediendo normalmente a las variables del proceso almacenadas en la tabla con sólo una operación. Como resultado se obtiene una alta disponibilidad de datos, lo que permite que se pueda mostrar lo que está sucediendo en el proceso en tiempo real, a través de las interfaces de visualización.

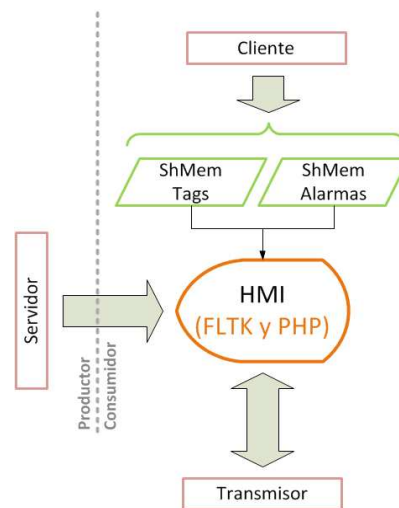


FIG 9. Interfaz para Visualización

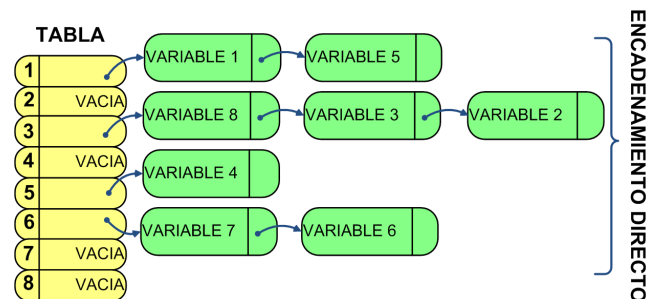


FIG 10. Estructura de Base de Datos de Tiempo Real

Cuando existen variables que deben ser almacenadas por largos períodos de tiempo, se hace uso del motor de base de datos. Sin embargo, debido a la alta frecuencia de adquisición de datos en algunos casos se hace imposible guardar directamente cada valor de variable de proceso, por ende, Argos provee algunas herramientas que usan un buffer circular (Figura 6) para realizar operaciones estadísticas con los datos, para luego ser almacenados siguiendo el modelo mostrado en la Figura 7.

Existen básicamente dos grupos de tablas, en el primero se almacenan las variables de acuerdo a una frecuencia de muestreo preestablecida (se pueden establecer diferentes frecuencias de muestreos por grupos de datos según lo requiera el proceso), mientras que el otro grupo contiene la información histórica de todas las variables almacenadas. El proceso de almacenamiento es llevado cabo de la siguiente manera: en cada instante de tiempo y de acuerdo a su frecuencia de muestreo se van almacenando los valores en las tablas del primer grupo. Después de un tiempo preestablecido, se realiza una transacción donde se transfieren los datos almacenados hacia las tablas del segundo grupo, que contienen la información procesada y permanente. Es importante acotar que los datos antes de ser transferidos son sometidos a un proceso de análisis estadístico para evitar almacenar grandes volúmenes de datos sin que estos tengan información relevante.

El diagrama de despliegue [5] de la Figura 11 muestra una de las múltiples disposiciones de los nodos propuesta por Argos, que incluyen elementos de hardware, así como también los componentes de software, evidenciando como interactúan.

B. Bibliotecas

El Proyecto Argos hace uso pleno de otros proyectos de software libre [6], como lo son: las bibliotecas muParser, tinyXML, FLTK y su ambiente de desarrollo Fluid, así como el software de base de datos PostgreSQL con sus API (Application Programming Interface).

Esto ha facilitado el desarrollo del proyecto, ya que, son componentes probados que reducen el trabajo requerido y apalancan las funcionalidades del sistema.

MuParser: es una biblioteca para el análisis de expresiones matemáticas. Es usado en Argos para calcular variables de procesos a partir de otras usando operaciones matemáticas.

TinyXML: se usa para el manejo de los archivos de configuración con formato XML.

FLTK: las interfaces gráficas están basadas en esta biblioteca. Se eligió por su ligereza para garantizar el rendimiento en computadores de recursos limitados, como sistemas embebidos. Se está implementando el uso de SVG para facilitar el desarrollo de las interfaces gráficas.

PostgreSQL: Es una de las bases de datos de software libre más utilizadas, es un proyecto probado y con una comunidad extensa. Se usa la API para integrar la base de datos al sistema.

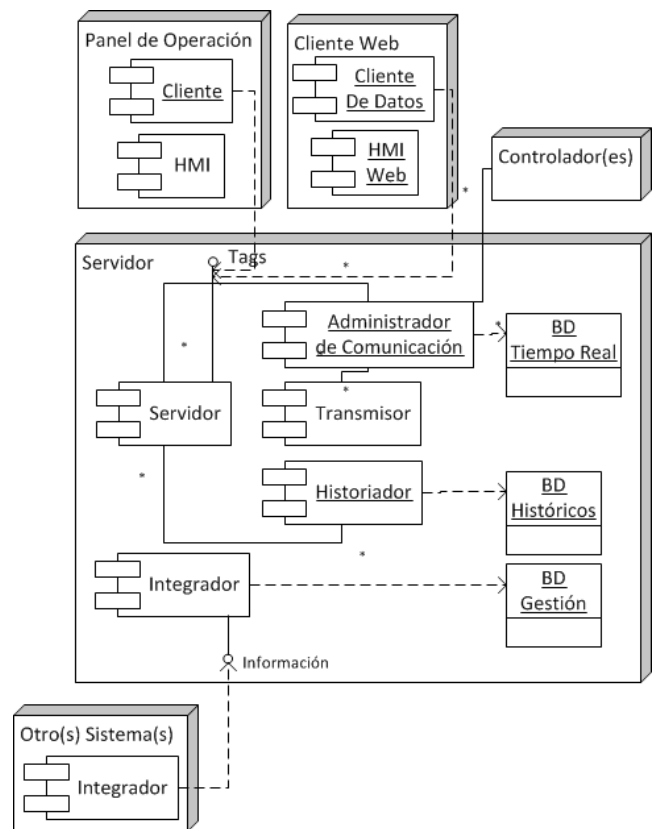


FIG 11. Diagrama de Despliegue de los Componentes

III. RESULTADOS

Todas las herramientas que proporciona Argos han sido desarrolladas y probadas en estaciones de trabajo de Laboratorio con las siguientes características: x86 y x86_64 con procesadores Pentium IV, AMD 64 y 1Gb de memoria RAM.

Actualmente el software está disponible como un proyecto de código abierto [6], bajo licencia GPL, en el repositorio SourceForge.NET, su estado es ALPHA y en continuo desarrollo para generar los candidatos a lanzamiento (RC)

y cuenta con los siguientes avances:

- Procesos de adquisición y administración de variables, eventos y alarmas.
- Procesos de envío y recepción de datos con los clientes HMI.
- Procesos para el almacenamiento en base de datos de las variables, eventos y alarmas.
- Capacidad para configuración a través de archivos en formato XML para todos los procesos anteriormente mencionados.
- Objetos gráficos diseñados para ser incorporados en aplicaciones de Ventanas usados para desplegar la información de las variables del proceso (como clientes HMI).

Para desarrollos futuros orientados a cubrir la mayoría de los requerimientos del sector industrial, en lo que a sistemas de control supervisor se refiere, se consideran los siguientes requerimientos:

- Desarrollo de manejadores para la adquisición de datos de la gran gama de dispositivos de instrumentación y control que actualmente se encuentran en el mercado e instalados en la mayoría de las plantas industriales.
- Simuladores para poder diseñar y desarrollar todo un sistema SCADA sin la necesidad de estar conectado a hardware industrial.
- Diseño y desarrollo de un entorno integrado para la configuración e implementación de todas las herramientas en un sistema SCADA.
- Diseño y desarrollo de un entorno que permita la creación de interfaces Web dinámicas compatibles con navegadores Web estándar.
- Procesos que administren la redundancia del sistema SCADA, asegurando de ésta manera la

robustez del control supervisor sobre un proceso productivo.

IV. CONCLUSIONES

El proyecto Argos ha sido desarrollado para ser una alternativa en el área de automatización industrial basada en herramientas de software libre. De esta manera pueda ser utilizado, estudiado y modificado por las distintas comunidades de conocimiento libre y desarrolladores de software, siguiendo la filosofía de los proyectos de código abierto.

Asimismo, el desarrollo comunitario permitirá proporcionar una arquitectura de software robusta y tecnológicamente avanzada, tal y como lo requiere el demandante sector industrial para lograr optimizar procesos productivos.

El modelo de desarrollo propuesto para el proyecto Argos permitirá la evolución del software de SCADA en un ambiente de colaboración, teniendo como punto de partida el desarrollado realizado por un equipo de investigación y desarrollo del centro de innovación tecnológica del aluminio, CINTAL, usando en su totalidad herramientas libres.

REFERENCIAS

- [1] A. Rodríguez. “**Comunicaciones Industriales**”. Marcombo, Primera Edición, 2008, pp. 256-270.
- [2] J. Hernández., L. León. “**Implementación de Objetos Gráficos para el Desarrollo de Despliegues Operacionales en el Sector Comercio y Suministro del SCADA Nacional de PDVSA**”. Tesis de Grado, Escuela de Ingeniería de Sistemas, Universidad de los Andes, Mérida, Venezuela, 2008.
- [3] J. M. Hernández. “**Software Libre: Técnicamente Viable, Económicamente Sostenible y Socialmente Justo**”. Infonomia, primera edición, 2005, pp. 43-63.
- [4] Abaffy, C., Lárez, J., Aiquel, R., González, J. “**CVG Venalum Potline Supervisory System**”. Artículo Técnico, TMS 2006, pp. 301-305, 2006.
- [5] I. Besembel, J. Montilva. “**Modelado de Sistemas usando UML 2.0**”. Centro de Excelencia en Ingeniería del Software (CEISOFT), 2006.
- [6] CINTAL, “**Argos el SCADA en Software Libre**”. <http://www.cintal.com.ve/argos/>, 2009.

BUSCONEST: Una Aplicación para Búsquedas de Documentos Académicos Electrónicos

Rivas Sergio, Zambrano Jossie, Villapol María E.

Resumen—Esta investigación está fundamentada en la localización de documentos electrónicos y el funcionamiento de los buscadores más comunes. En este trabajo se desarrolló un motor de búsqueda Web, BUSCONEST, sobre documentos digitales en formato PDF. BUSCONEST fue diseñado para buscar en los trabajos producidos por los estudiantes que egresan de la Facultad de Ciencias de la Universidad Central de Venezuela. El motor de búsqueda toma en cuenta, para el posicionamiento de los resultados, los metadatos del documento, las palabras clave y frecuentes dentro su contenido y la importancia académica del trabajo según sus autores. Además usa estructuras auxiliares para optimizar el espacio de almacenamiento y tiempo de ejecución, aportando soluciones como la implementación del algoritmo de ordenamiento basado en el modelo de balance lineal simple, manejo de persistencia y ejecución de las consultas.

Palabras claves—Buscador Web, Algoritmo de Ordenamiento, Balance Lineal Simple, Posicionamiento.

I. INTRODUCCIÓN

La creciente cantidad de información existente en la Internet y la necesidad de los usuarios de acceder dicha información han incrementado la popularidad de los buscadores en la Internet (tales como: google y yahoo), estos suelen ser lo más generales posibles para soportar un amplio rango de búsquedas. En ambientes donde se desea recuperar información de un determinado tipo y con un restringido conjunto de formatos bien específicos, estos buscadores podrían no ser tan precisos en la recuperación de la información.

En la Facultad de Ciencias de la Universidad Central de Venezuela (UCV) [1], se genera una gran cantidad de documentos académicos que son producto de los proyectos finales de investigación de los estudiantes de las carreras allí impartidas. Actualmente, es necesario entregar una copia electrónica en formato PDF y una física. Por lo cual, se presenta la problemática de cómo los usuarios podrían recuperar la información digital. Los buscadores tradicionales, debido a la búsqueda por frecuencia del texto, tienen problemas en dos aspectos fundamentales: las palabras según

el contexto pueden significar distintas cosas, y los resultados se ordenan según la frecuencia del texto buscado en el documento encontrado.

Por lo antes expuesto, se ha desarrollado un motor de búsqueda, BUSCONEST, el cual se diferencia de los tradicionales porque ordena los documentos que cumplen el criterio de búsqueda de acuerdo a su importancia en el contexto académico.

Varios son los trabajos de investigación sobre el diseño de motores de búsqueda de información, en particular, aquella localizada en la Web [2]. Existen igualmente diversos trabajos enfocados en el ordenamiento de la información recuperada en la Web [4][5]. A diferencia de estos, en este trabajo se presenta un motor de búsqueda que funciona sobre una base de datos de documentos mucho más restringida y que ordena los documentos que cumplen el criterio de búsqueda de acuerdo a su relevancia en el contexto académico.

Con la finalidad de alcanzar los objetivos planteados, este artículo se divide en las siguientes secciones. La sección dos presenta una breve reseña sobre los motores de búsqueda. BUSCONEST es descrito en la sección tres. La validación del motor de búsqueda en contra de algunos casos de pruebas es entonces presentada en la sección cuatro. Finalmente, la sección cinco concluye este artículo incluyendo los trabajos futuros que se desprenden del mismo.

II. MOTORES DE BÚSQUEDA

Existen dos tipos de buscadores generales usados en la Internet: *de índices temáticos* y *motores de búsqueda* [6]. El motor de búsqueda, que es el usado en este trabajo, es el programa que dado un criterio consigue los resultados de la información relacionada con este. Sus funciones son: ubicar la información, ordenarla, y mostrar los resultados.

A continuación se describen las partes resaltantes que intervienen en un motor de búsqueda.

A. Arañas o Robots

La araña es un programa que se ejecuta constantemente en un segundo plano para garantizar la indexación de las páginas Web. La misma va guardando información relevante (útil para ser criterio de búsqueda o puntaje) acerca de cada página visitada, y así va formando una base de datos en expansión.

B. Base de Datos e Indexación

Una vez recolectada la información es necesaria ordenarla e indexarla para poder recuperar rápidamente una entrada de acuerdo a un criterio. Para la búsqueda, la información se

Artículo recibido el 22 de Enero del 2010. Este artículo fue financiado por la Universidad Central de Venezuela.

R.S., Z. J., y M.E.V. están con la Universidad Central de Venezuela, Avenida los Ilustres, Facultad de Ciencias, Escuela de Computación, Los Chaguaramos, Caracas, Distrito Capital, Venezuela, Tlf. +58-212-6051264 / 1067, Fax: +58-212-6051676, E-mail: sergio.rivas@ciens.ucv.ve, jossie.zambrano@ciens.ucv.ve, maria.villapol@ciens.ucv.ve.

segmenta y clasifica según uno o más atributos. Este tipo de sistemas requiere grandes cantidades de almacenamiento, en muchos casos guardados de forma distribuida.

C. Algoritmos de Búsqueda

Los algoritmos de búsqueda buscan la información tomando en cuenta que se deben mostrar los resultados más relevantes. Esta relevancia se determina según un puntaje (*ranking*) de cada resultado, donde se calculan una o más medidas de cercanía con el criterio para poder ordenarlos de la forma más apropiada.

III. BUSCONEST

BUSCONEST es un motor de búsqueda sobre un repositorio de publicaciones digitales de carácter académico de la Facultad de Ciencias de la Universidad Central de Venezuela. Las características de este motor buscador se describen a continuación:

- Ordena los documentos que cumplen el criterio de acuerdo a su importancia en el contexto académico.
- Busca tanto por las palabras claves, título y otros metadatos del documento como en su contenido en sí, lo que plantea selección e indexación del texto de cada documento agregado al repositorio.
- Utiliza algoritmos y estructuras auxiliares o redundantes, tales como tablas inversas de palabras, para mantener el equilibrio entre la eficiencia en tiempo y en espacio.

El sistema tiene prácticamente los mismos componentes de un motor de búsqueda de Internet salvo por el componente araña o robot. La ausencia de la araña se debe a que BUSCONEST no se desempeña sobre un medio tan extenso e indeterminado como Internet, sino que actúa sobre un repositorio digital bien establecido y definido. A continuación se explica el funcionamiento de los módulos del sistema.

A. Procesamiento de la información

Este módulo se encarga de almacenar y procesar el documento en el sistema. Una vez adquirido el archivo y algunos datos del estudiante, se solicita al Sistema de Gestión Académica de Control de Estudios (CONEST), a través de Servicios Web, toda la información asociada. Luego se almacena la metadata del documento académico en formato XML, posteriormente el archivo binario en formato PDF es sometido a la extracción de texto plano, donde es procesado para la obtención de todas las palabras distintas con sus respectivas frecuencias en el contenido. Finalmente se termina de obtener y calcular toda la metadata en la que también se encuentran los pesos de relevancia. En este módulo se llevan a cabo las tareas que hacen las arañas en los motores de búsqueda.

B. Recuperación de documentos

Después de que el usuario ha establecido sus criterios de búsqueda, se procesa el campo de texto de comandos y se realiza la búsqueda en el manejador de base de datos, para finalmente obtener todos los documentos deseados.

C. Ordenamiento

En este módulo se aplica un algoritmo matemático de ordenamiento valiéndose de los campos que contienen los pesos numéricos que serán tomados en cuenta tales como: frecuencia de palabras en el título, resumen, contenido, y el peso académico que contiene el rendimiento de los autores incluyendo la eficiencia, promedio de notas, nota del trabajo y si el trabajo obtuvo alguna distinción honorífica. El ordenamiento se basa en el modelo matemático de *balance lineal simple* [7].

D. Modelo de búsqueda y ordenamiento

Para hacer un ordenamiento se necesitan valores numéricos. Cada documento debe ser representado por un valor y para ello se toma en cuenta más de un criterio. Unos criterios son fijos para el documento en todo momento (por ejemplo el peso académico), y otros criterios son variables en el momento de la consulta (por ejemplo la presencia de palabras de una consulta específica).

El modelo de balance lineal simple puede explicarse a través de los siguientes pasos. Se llevan todos los criterios numéricos (*cr*) a un mismo rango de valores acotados, para ello se toma en cuenta el mínimo (*cota_{min}*) y el máximo (*cota_{max}*) valor para cada tipo de criterio y el valor mínimo (*cr_{min}*) y valor máximo (*cr_{max}*) del criterio numérico como se muestra en la ecuación (1).

$$crn = cot a_{min} + \frac{(cot a_{max} - cot a_{min})(cr - cr_{min})}{cr_{max} - cr_{min}} \quad (1)$$

En el caso de los criterios en BUSCONEST el rango cada criterio será [0,1] y asumiendo que el mínimo para todo valor es 0, la ecuación (1) se simplifica como se muestra en (2).

$$crn = \frac{cr}{cr_{max}} \quad (2)$$

Luego se establece el peso o importancia para cada criterio determinándolo a través de un vector de coeficientes *imp* (ver (3)) cuyos valores representan la importancia numérica en cada caso.

$$\vec{importancia} = (imp_1, imp_2, imp_3, \dots) \quad (3)$$

A cada documento se le asocia el vector de todos los criterios normalizados (ver (4)):

$$\vec{X} = (crn_1, crn_2, crn_3, \dots) \quad (4)$$

El modelo de balance lineal simple propone un vector que representa al documento óptimo, es decir, el que tiene máximo valor en todos los criterios en este caso el vector del documento óptimo se muestra en (5).

$$\overrightarrow{optimo} = (1,1,1,...) \quad (5)$$

El mejor documento es el que este lo más cerca del documento óptimo. La ecuación de la distancia euclidiana se muestra en (6).

$$\sqrt{\sum (1 - crn_i)^2} = ((1 - crn_1) + (1 - crn_2) + \dots)^{1/2} \quad (6)$$

Como el documento tiene diferentes pesos de importancia para los criterios se debe aplicar una modificación a la ecuación de la distancia (6) para obtener el valor de ordenamiento definitivo ponderado por el peso de cada criterio como lo muestra en (7).

$$\sqrt{\sum imp_i (1 - crni)^2} \quad (7)$$

En este sistema se hicieron algunas variaciones a la fórmula original evitando las potencias de ordenamiento para optimizarla y reducir los cálculos, ya que se necesita una respuesta muy rápida y es el manejador de base de datos quien se encarga de ejecutarla.

De la fórmula de la distancia lo que interesa es su valor numérico para establecer el orden, quiere decir que es válido alterarla si nos sigue dando ese mismo orden. Otra variación es asumir que el valor mínimo para cualquier peso numérico siempre será cero, así se ahorra el cálculo del rango mínimo en las normalizaciones.

E. Criterios utilizados

El criterio más importante es el *peso académico*, peso_academico, del documento que toma en cuenta la eficiencia, e, el promedio ponderado, prom_pond, y la calificación (nota obtenida en el trabajo especial de grado) dado por la ecuación (8):

$$peso_academico = (e)(prom_pond)(calificacion) \quad (8)$$

La eficiencia corresponde al valor del resultado de las unidades de créditos aprobadas entre las unidades inscritas por el alumno, a lo largo de sus estudios. El promedio ponderado se refiere al promedio de notas de las materias inscritas en todo el historial académico pesado por las unidades de crédito de cada una de ellas. Y la calificación es la nota obtenida en el documento digital que se encuentra en el repositorio (en este caso trabajos especiales de grado).

El otro criterio es el *peso numérico*, peso_numerico, calculado en base a las palabras presentes en el documento ya sea en el contenido, título, resumen y palabras claves. Este criterio espre calculado parcialmente durante la carga del documento; el valor numérico definitivo depende de la consulta que se haga. Por cada palabra presente en el documento, se tiene información de la frecuencia con que aparece: en el contenido f_c , en el título f_t , en el resumen f_r , y en las palabras claves f_{pc} .

El peso numérico de las palabras viene dado por la expresión (9):

$$peso_numerico = \ln(1 + f_c + 4f_t + 4f_r + 4f_{pc}) \quad (9)$$

La expresión (9) otorga más importancia a las palabras si están en otros lugares aparte del contenido. Estos pesos fueron refinados en las pruebas según criterios empíricos que evaluaban la relevancia. Además cuando se buscan dos o más palabras se le da más importancia al documento que tiene las frecuencias lo más parecidas posible para cada palabra.

IV. PRUEBAS Y RESULTADOS

La aplicación desarrollada se implementó bajo el Framework Ruby on Rails [8] en su versión 1.2.5 con sistema manejador de base de datos MySQL 5, usando como servidor de aplicaciones Mongrel y Apache 2 como servidor Web.

Se realizaron varias pruebas de la aplicación con una muestra de 327 documentos procesados los cuales ocupan cerca de 900Mb. En la pruebas se buscó probar los aspectos relacionados con la búsqueda por contenido, el puntaje de los documentos y su relación con el peso académico (*pa*) al mismo tiempo que se consideraban los premios académicos de los autores (*prem*), además de medir el rendimiento, tiempo de respuesta razonable y precisión. A continuación se muestran los resultados de algunas de las pruebas realizadas.

Al colocar la palabra *superficie* como criterio de búsqueda se obtienen 186 documentos de las 6 licenciaturas Biología, Matemática, Computación, Física, Geoquímica y Química. Debido a las limitaciones del espacio, en la Tabla 1, solo se muestran 5 de los resultados (*res*). Como se puede observar la palabra buscada no existe en ningún título ($f_t = 0$) y el primer resultado (*res 1*) corresponde a un estudiante con premio magna cum laude y méritos académicos.

TABLA 1
PRUEBA CON LA PALABRA: SUPERFICIE.

Res\Criterio	f_c	f_t	f_{pa}	f_r	f_a	pa	prem
res 1	37	0	0	0	0	360	si
res 2	63	0	0	3	0	303	no
res 3	74	0	0	2	0	282	no
res 185	6	0	0	0	0	122	no
res 102	3	0	0	0	0	102	no

Al colocar la palabra *ronald*, la cual es un nombre, como criterio de búsqueda se obtienen 23 documentos (Véase los primeros 7 resultados en la Tabla 2). La palabra buscada no existe en ningún título, sin embargo, se observa que existe en los autores (f_a).

Al colocar la palabra *hierro* como criterio de búsqueda se obtienen 118 documentos, algunos de estos resultados se muestran en la Tabla 3. Como se puede observar la palabra buscada existe en títulos, palabras clave y resúmenes.

TABLA 2
 PRUEBA CON LA PALABRA: RONALD.

res/criterio	f_c	f_t	f_{pa}	f_r	f_a	pa	prem
res 1	2	0	0	0	1	344	no
res 2	3	0	0	0	1	315	no
res 3	1	0	0	0	1	315	no
res 4	2	0	0	0	0	284	no
res 5	1	0	0	0	0	305	no
res 6	2	0	0	0	0	279	no
res 7	1	0	0	0	0	301	no

TABLA 3
 PRUEBA CON LA PALABRA: HIERRO.

res/criterio	f_c	f_t	f_{pc}	f_r	f_a	pa	prem
res 1	41	1	0	1	0	305	no
res 2	27	0	0	2	0	330	no
res 5	106	0	1	3	0	219	no
res 46	7	0	0	0	0	229	no

La Tabla 4 muestra los resultados obtenidos al realizar pruebas con distintas palabras, que muestran la búsqueda por contenido. En la tabla se muestra la cantidad de resultados, valores máximos y mínimos para la frecuencia en contenido (f_c) y peso académico (pa), así como también donde se consiguieron las palabras (*Frecuencia en*) y si existe algún resultado con premio (*Existe premio*).

TABLA 4
 PRUEBA CON DISTINTAS PALABRAS.

Palabra \ Criterio	Can t	Ma x f_c	Minf c	Ma x pa	Minp a	Frecuenci a en	Existe premi o
superficie	186	108	0	367	102	contenido, palabras clave, resumen	si
cubo	16	39	1	360	130	contenido	si
polímero	34	113	1	327	122	contenido, resumen	no
compilador	18	56	1	323	83	título, palabras clave, resumen, contenido,	no
tensión	46	62	1	331	121	contenido	no
paralelo	65	10	1	360	83	contenido	si
tcp	24	61	1	317	83	contenido, resumen	no
carbono	151	88	1	367	97	contenido, Palabras clave, resumen	Si
hidrocarburo	27	5	1	367	121	contenido	Si
insecto	10	18	1	315	157	contenido, resumen	no
hierro	118	126	1	367	89	título, contenido, resumen	si
verdad	35	3	1	367	132	contenido	si
cielo	26	2	1	367	132	contenido	si
ronald	23	3	1	344	142	autores, contenido	no

Los resultados de la pruebas anteriores confirman la búsqueda por contenido, mostrando en los primeros lugares aquellos con mayor relevancia académica. Además, se evidencia la frecuencia en títulos, autores y su influencia en la posición.

V. CONCLUSIONES

Este artículo muestra la experiencia en la implementación de un motor de búsqueda de documentos académicos en formato PDF, que aprovecha las ventajas de la información académica otorgando relevancia a través del posicionamiento, con el fin de mostrar resultados más confiables y precisos.

Hasta el momento, BUSCONEST, ha sido probado usando una serie de criterios. Los resultados de dichas pruebas revelan que el sistema se comporta de acuerdo a lo esperado, ordenando las salidas de dichas búsquedas de acuerdo a su relevancia académica.

Se recomienda tomar en cuenta para futuros trabajos otros criterios para el ordenamiento y que estos sean configurables por el usuario que realiza la búsqueda, de igual forma es importante minimizar los tiempos de respuesta de la aplicación al realizar las mismas.

REFERENCIAS BIBLIOGRÁFICAS

- [1] Facultad de Ciencias, Universidad Central de Venezuela, Caracas, Venezuela. Disponible en <http://www.ciens.ucv.ve> (enero 2010).
- [2] Susumu Akamine, Yoshikiyo Kato, Daisuke Kawahara, Keiji Shinzato, Kentaro Inui, Sadao Kurohashi, and Yutaka Kidawara. "Development of a Large-scale Web Crawler and Search Engine Infrastructure". Proceedings of the 3rd International Universal Communication Symposium, Vol 398, Tokyo, Japan 2009, pp 126-131.
- [3] Andrei Broder. "A Taxonomy of Web Search". ACM SIGIR Forum, Vol.36, Issue 2 (Fall 2002), pp 3-10.
- [4] Anja Theobald, and Gerhard Weikum. "The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking". LNCS, Vol. 2287, Springer, 2002., pp 311-240.
- [5] Xiaolan Zhu, and Susan Gauch. "Incorporating Quality Metrics in Centralized/Distributed Information Retrieval on the World Wide Web". Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, Athens, Greece 2000, pp 288-295.
- [6] Michel W. Berry, Murray Browne. "Understanding Search Engines". Mathematical Modeling and Text Retrieval SIAM, 2005.
- [7] Sánchez E, J. I. "Elaboración de un Prototipo de Buscador de Documentos Académicos de la Facultad de Ciencias". Universidad Central de Venezuela., 2008.
- [8] David A. Black. "Ruby for Rails: Ruby Techniques for Rails Developers", 2006.