Subsistema de comunicaciones para el SCADA SAINUX. Communications subsystem for SCADA SAINUX.

Yanelys del Rosario Lalcebo
Universidad de las Ciencias Informáticas. La Lisa, La Habana. Cuba. ydelrosario@uci.cu

Yadiel Martinez Gonzalez
Universidad de las Ciencias Informáticas. La Lisa, La Habana. Cuba. ymglez@uci.cu

Yolier Galán Tassé
Universidad de las Ciencias Informáticas. La Lisa, La Habana. Cuba. ytasse@uci.cu

Yenier Figueroa Machado
Universidad de las Ciencias Informáticas. La Lisa, La Habana. Cuba. yfigueroa@uci.cu

ABSTRACT

The communication between the various modules that make up SCADA is very cumbersome because it does not always have to be supported by the same platform, the same OS or the same programming language, so it is vital to have a means of communication to resolve such problems. Moreover, considering the Cuba´s limitations for paying software licenses, it is necessary to develop technologies based computing products free and at the same time to obtain a commercial version of the final product. This work is about the development of the communication subsystem for Industrial Automation System (SAINUX) based on OpenDDS Middleware technology, which it is used in systems that require real-time, robustness and fault tolerance capabilities; also it is based on the message model Publish/Subscribe. This technology is released by the Lesser General Public License (LGPL), which allows obtaining a final product without obligation to redistribute any part of the code, supporting this way, the development of the national economy that can provide marketing software.

Keywords: communications subsystem, middleware, Publish-Subscribe, SCADA

RESUMEN

Permitir la comunicación entre los distintos módulos que componen un SCADA resulta muy engorroso debido a que no siempre deben estar soportados por la misma plataforma, el mismo sistema operativo o el mismo lenguaje de programación, por lo que resulta de vital importancia contar con un medio de comunicación que resuelva este tipo de problemas. Además, a partir de las limitantes que tiene Cuba para el pago de licencias de software, se hace necesario el desarrollo de productos informáticos basados en tecnologías libres y que a la vez permitan obtener una versión comercial del producto final. Este trabajo consiste en el desarrollo del subsistema de comunicaciones para el Sistema de Automatización Industrial (SAINUX) basado en la tecnología Middleware OpenDDS, la cual es utilizada en sistemas que requieren de capacidades de tiempo real, robustez, tolerancia a fallas, y está basada en el modelo de mensajes Publicación/Suscripción. Esta tecnología está liberada por la Licencia Pública General Menor (LGPL) que permite obtener un producto final sin la obligación de redistribuir cualquier parte del código, apoyando así el progreso de la economía nacional que puede proporcionar el mercadeo de software.

Palabras claves: subsistema de comunicaciones, middleware, publicación-suscripción, SCADA.

1. INTRODUCCIÓN

El pago de las licencias de software actualmente, no representa una opción económicamente viable para Cuba debido al alto coste de las mismas y además de que es un país bloqueado. La necesidad de apoyar el progreso de la economía cubana, ha influenciado el desarrollo de software basado en tecnologías libres que a la vez permitan la comercialización.

Por lo general, las tecnologías libres que existen hoy en día están patentadas bajo la Licencia Pública General (GPL). Esta es una licencia que permite utilizar y modificar código libremente, pero obliga a distribuir todo el código desarrollado bajo sus mismos términos.

Otro tipo de licencia que existe y que es más abierta en cuanto a su uso, es la Licencia Pública General Menor (LGPL). Esta es una modificación de la GPL que permite a los desarrolladores construir tanto software libre como propietario, por lo que se pueden realizar versiones comerciales de un producto final.

Teniendo en cuenta los elementos antes expuestos y en favor de propiciar el avance de la economía nacional a través del mercadeo de software, es necesario que el desarrollo de productos informáticos se haga basado en tecnologías liberadas por la LGPL.

Un claro ejemplo de ello, es el producto Sistema de Automatización Industrial UX (SAINUX) desarrollado por el Centro de Informática Industrial (CEDIN) de la Universidad de las Ciencias Informáticas (UCI), cuyo objetivo es poder automatizar los procesos de las empresas que lo requieran en el país, así como poder ser un producto comercializable. Este está compuesto por varios módulos que tienen una función específica dentro del sistema, pero para lograr que funcione, los módulos deben estar comunicados y coordinados entre sí. Por esta razón el objetivo de la presente investigación es desarrollar un subsistema de comunicaciones basado en tecnologías Middleware existentes, que permita el intercambio de información entre los procesos distribuidos de SAINUX y que garantice la instalación del producto en el ámbito nacional, así como su comercialización con otros países.

2. CONTENIDO

Para llevar a cabo el desarrollo del subsistema de comunicaciones de SAINUX se estudiaron una serie de elementos, como las tecnologías a utilizar y los procedimientos a seguir para su implementación. A continuación se exponen los mismos:

2.1 MIDDLEWARE EN SISTEMAS SCADA

Con el devenir de los Sistemas de Control y Adquisición de Datos (SCADA) distribuidos y la necesidad de gestionar la comunicación entre sus componentes surgen los denominados middleware:

Según el doctor Douglas C. Schmidt, el middleware es un software de infraestructura que reside entre las aplicaciones y el sistema operativo, redes, y hardware subyacentes, específicamente intentando brindar una plataforma más apropiada para el desarrollo y ejecución de los sistemas distribuidos. (Schmidt, 2005)

Una definición bastante completa, puede encontrarse en el libro “Documentación de ZeroC ICE”, donde se plantea: se puede entender un middleware como un *software* de conectividad que hace posible que aplicaciones distribuidas pueden ejecutarse sobre plataformas heterogéneas, es decir, sobre plataformas con distintos sistemas operativos, que usan distintos protocolos de red y que incluso, involucran distintos lenguajes de programación en la aplicación distribuida. (Fernández, 2006)

En la pirámide de automatización de un sistema SCADA (ver figura1), el middleware, es la capa de software que se encuentra por encima de los niveles físicos y de red y por debajo de las aplicaciones de usuario. (Aragón Cáceres y Llanes Jiménez, 2009)

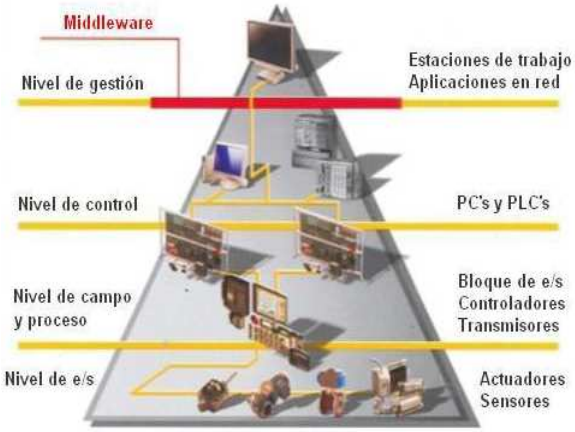

Figura. 1: Pirámide de automatización de un SCADA. (Aragón Cáceres y Llanes Jiménez, 2009)

El *middleware* está capacitado para proporcionar: (Guibert y Cárdenas, 2008)

- Independencia entre el cliente y el servidor
- Traducción de la información de una aplicación y el paso de dicha información a otra
- Control de las comunicaciones
- Gestión de dispositivos
- Procesamiento de datos
- Conectar la información con las aplicaciones de negocio

2.2 MIDDLEWARE *OPENDDS*

Un estudio de las distintas implementaciones de middleware que pueden ser adaptadas a las necesidades del SCADA SAINUX para permitir el intercambio de información entre los módulos que lo componen, determinó seleccionar *OpenDDS* teniendo en cuenta que cumple con los siguientes requisitos fundamentales para el desarrollo del subsistema de comunicaciones:

- Desarrollo en C++
- Implementación con herramientas de software libre y código abierto
- Comunicación distribuida
- Capacidad de tiempo real
- Manejo de un elevado número de variables
- Persistencia a las conexiones
- Tolerancia a fallos
- Redundancia
- Seguridad

OpenDDS es una implementación C++ multiplataforma de código abierto, del Estándar de Distribución de Datos (DDS) del Grupo de Gestión de Objetos (OMG), para sistemas cuyos requisitos incluyen: tiempo real, robustez, distribución de datos y tolerancia a fallos, utilizando un modelo de publicación/suscripción. En particular, se

puede utilizar *OpenDDS* para construir *software* propietario y no se está bajo ninguna obligación de redistribuir cualquier parte del código fuente que se construye. (OpenDDS.org, 2012)

OpenDDS se basa en la capa de abstracción del Entorno de Comunicación Adaptativa (ACE) para facilitar la portabilidad de la plataforma. Además aprovecha las capacidades de The ACE Orb (TAO), como su compilador de Lenguaje de Definición de Interfaces (IDL) y la base del Repositorio de Información del DCPS (*DCPSInfoRepo*).(OpenDDS.org, 2012)

Entre sus principales componentes se encuentran: (Vega, 2008)

DomainParticipant: Es el punto de entrada de las comunicaciones para un dominio concreto. Representa a la aplicación que emplea DDS dentro de un dominio específico y proporciona los componentes de DDS necesarios para la comunicación.

Topic: Instanciación de un componente *TopicDescription*, que es la descripción básica de un dato que puede ser publicado y al que se pueden suscribir otros componentes.

Publisher: Objeto responsable de la diseminación de datos cuando estos deben ser publicados.

Datawriter: Objeto que permite a una aplicación instanciar un valor de un dato para que sea publicado por un *Topic*.

Subscriber: Objeto responsable de la recepción de los datos resultante de las suscripciones.

Datareader: Objeto que permite a la aplicación declarar los datos en los que está interesada recibir información (creando una suscripción usando un *Topic*) y accede a los datos recibidos usando un *Subscriber* asociado.

2.3 ARQUITECTURA DE OPENDDS

La arquitectura que implementa OpenDDS está separada en dos capas. La capa superior es la capa local de reconstrucción de datos (DLRL), la cual se encarga de serializar y deserializar los datos, y la capa inferior es la centrada en los datos de publicación y suscripción (DCPS). Esta última es responsable de manera eficiente de la difusión de datos desde los publicadores a los suscriptores interesados. Está implementada utilizando los conceptos de *publisher* y *datawriter* en el lado del emisor y *subscriber* y *datareader* en el lado del receptor. La capa DCPS consta de uno o más dominios de datos, cada uno de los cuales contiene un conjunto de participantes (editores y suscriptores) que se comunican a través de DDS. Cada entidad (es decir, el editor o suscriptor) pertenece a un dominio. Cada proceso tiene un participante de dominio para cada dominio de datos del cual es miembro. (OpenDDS.org, 2012)

La siguiente figura muestra gráficamente el funcionamiento de la capa DCPS, basado en la representación que se encuentra en el sitio de OpenDDS.org en la sección Arquitectura DDS:

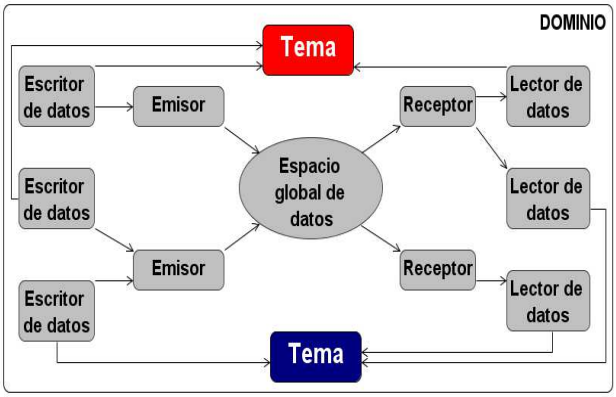

Figura. 2: Funcionamiento de la capa DCPS

2.4 MODELO DE MENSAJES PUBLICACIÓN/SUSCRIPCIÓN

El modelo de publicación/suscripción de OpenDDS se fundamenta en el intercambio asíncrono de mensajes. Aquí las entidades generadoras declaran una serie de *topics* o temas que están dispuestas a publicar y las consumidoras se suscriben a distintos temas de su interés. De este modo, cuando un productor publica un dato con una temática concreta, todos los suscriptores de esta lo reciben de forma transparente a la aplicación. El paradigma adopta una aproximación denominada *data centric*, ya que desacopla (en el tiempo y el espacio) la interacción entre los participantes (consumidores o generadores de información), y enfoca su esfuerzo en la distribución de los datos, con independencia de la localización y el instante de origen o destino de los mismos.

2.5 ESCENARIO DE LA INVESTIGACIÓN

SAINUX tuvo un producto antecesor que fue el SCADA UX, el cual se encuentra instalado en el acueducto de Santiago de Cuba para la supervisión y control de este proceso, garantizar la eficiencia y contribuir a la solución del problema de abasto de agua en dicha provincia. Con el objetivo de ahorrar tiempo y esfuerzos, SAINUX reutiliza varios módulos del SCADA UX que no contengan tecnologías ni herramientas privativas en su desarrollo; pero uno de los módulos fundamentales para el funcionamiento de un sistema distribuido es el de Comunicaciones. Este fue desarrollado en un inicio, utilizando la tecnología ICE del acrónimo *Internet Communication Engine*, una plataforma para desarrollo de aplicaciones de comunicación de alto rendimiento, que incluye varias capas de servicios y *plugin*. ICE es un software liberado bajo la Licencia Publica General (GPL), aunque también está disponible la adquisición de una licencia comercial, para quienes no desean utilizar la tecnología sujetos a los términos de GPL.

El uso de la distribución libre de ICE en la implementación del módulo de comunicación del SCADA UX, imposibilita la utilización del mismo en la implementación de SAINUX. Esto se debe a los términos de su licencia GPL, impidiendo cualquier apoyo al progreso de la economía nacional que pudiera proporcionar el mercadeo del software. También es posible adquirir una licencia comercial de la tecnología, pero esta no constituye una opción económicamente viable, debido al alto coste de la misma. Por tales motivos no es factible utilizar el módulo de comunicaciones del SCADA UX y se hace necesario desarrollar un nuevo subsistema de comunicaciones entre los diferentes módulos de SAINUX, basado en otra tecnología, que permita gestionar el intercambio de datos entre los mismos.

3. PROPUESTA DE SOLUCIÓN

En el proceso de desarrollo de *software* intervienen varios factores que determinan la calidad del producto final, ellos son: la metodología, los lenguajes y las herramientas a utilizar durante el desarrollo del subsistema de comunicación, los cuales, en el presente trabajo, no son objeto de selección, pues ya han sido analizados y valorados por analistas, arquitecto y jefe del proyecto de SAINUX. Como metodología de desarrollo se empleará *Rational Unified Process* (RUP). RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminarse cada una de las iteraciones. Es más apropiada para proyectos de gran envergadura, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas, por lo que la hace ideal para el sistema SAINUX. El lenguaje de modelado a utilizar será *Unified Modeling Language* (UML), es desde finales de 1997 un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de *software* (Grady, 2007). Como herramienta CASE, *Visual Paradigm* la cual propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación, ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del *software* a través de la representación de todo tipo de diagramas (EcuRed, 2012). El lenguaje de programación y entorno de desarrollo del cual se hará uso son C++ y eclipse respectivamente.

3.1 FUNCIONALIDADES DEL SUBSISTEMA DE COMUNICACIONES

Teniendo en cuenta la necesidad de crear un subsistema de comunicaciones que permita el intercambio de información entre todos los módulos de SAINUX, a continuación se enuncian las funcionalidades determinadas para desarrollar dicho subsistema.

1- El sistema debe permitir el envío y recepción de datos complejos de tipo:

- **Puntos:** El flujo principal de información en los sistemas SCADA lo constituyen las variables (puntos). Estas variables pueden representar innumerables indicadores como son: presión, temperatura, flujo, potencia, peso, intensidad de corriente, voltaje, potencial hidrógeno, densidad, carga, resistencia o capacitancia entre otros. (Pérez, 2009)
- **Alarmas:** Las alarmas se basan en la vigilancia de los parámetros de las variables del sistema. Son los sucesos no deseables, porque su aparición puede dar lugar a problemas de funcionamiento. (Pérez, 2009)
- **Eventos:** El resto de las situaciones normales, tales como puesta en marcha, paro, cambios de consignas de funcionamiento, consultas de datos, entre otras, serán los denominados eventos del sistema o sucesos. (Pérez, 2009)
- **Comandos:** Acción que se ejecuta en sistemas SCADA que modifican los valores recolectados de campo. Estos pueden ser desde acciones para modificar el valor de una variable hasta el reconocimiento de una alarma.
- **Estado de la comunicación:** Es la información relacionada con los estados de los dispositivos, canales y sub – canales.
- **Bitácoras:** Hechos ocurridos durante la utilización del sistema que puedan ser aprovechados como experiencias en el futuro.

3.2 DISEÑO DE LA SOLUCIÓN

Para dar solución a las funcionalidades propuestas, los autores diseñaron el siguiente diagrama, que muestra los distintos paquetes que componen el modelo de Publicación/Suscripción en el subsistema de comunicaciones de SAINUX, cuyos propósitos se explican a continuación.

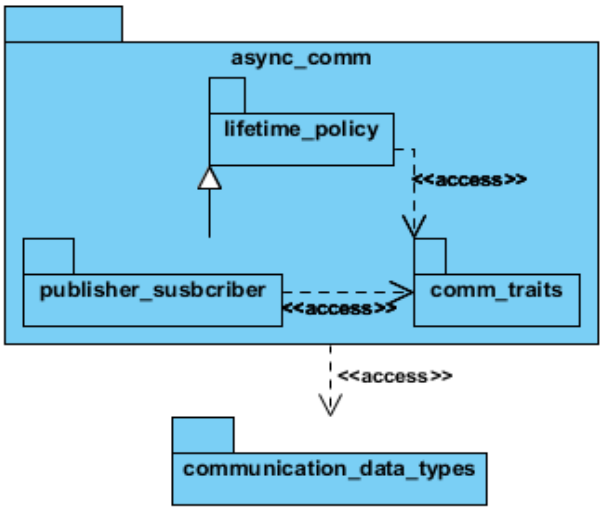

Figura. 3: Diagrama de paquetes del subsistema de comunicaciones de SAINUX

Publisher_subscriber:

Este paquete engloba la especificación del patrón de diseño *Facade* (fachada), el cual proporciona las interfaces *publisher* y *subscriber* que representan a todo el subsistema facilitando su uso y desacoplando a los clientes en tiempo y espacio.

Las clases que lo componen son:

- ***basic_communication:*** Engloba las características que son comunes a los publicadores y suscriptores, y actúa específicamente como uno de ellos, en dependencia del parámetro plantilla que se le pase (*publisher* o *subscriber*).
- ***publisher:*** Es la clase encargada de diseminar los datos en el dominio.
- ***write_actions:*** Esta clase se encarga de publicar un tipo de dato específico.
- ***subscriber:*** Se encarga de gestionar las suscripciones relativas a los *topics* sobre los que se ha declarado interés de recibir datos.

Lifetime_policy:

Este paquete implementa el patrón de diseño *Policy* (políticas), el cual pone su énfasis en el comportamiento de creación y destrucción de algunas entidades, tales como: *Topic*, *DomainParticipant* y *DomainParticipanFactory* en la clase *lifetime_policy*, la cual actúa en dependencia del parámetro plantilla que recibe, que puede ser: *publisher* o *subscriber.*

Las clases que lo componen son:

- ***lifetime_policy:*** Inicializa y destruye los participantes de dominio y los *topics*.
- ***ITopic*:** Es una interfaz que posee las funcionalidades de los *topics*.
- ***Topic:*** Esta es la clase genérica encargada de la creación de *topics*, asociándolos a tipos de datos.

Comm_traits:

Este paquete contiene la especificación del patrón de diseño *Traits* (rasgos o características), que es una técnica que sale naturalmente del uso de plantillas cuando tienes que manejar varios tipos/clases distintas. Consiste en definir clases plantillas intermediarias que contienen aparte, las características de las clases que se manipulan en el patrón. En este caso se utiliza para definir las características de los tipos de datos: puntos, alarmas, eventos, comandos, estado de la comunicación y bitácoras, así como para el modo de comunicación que se establecerá, ya sea de publicador o de suscriptor.

Las clases que lo componen son:

- ***communication_traits:*** Es una clase genérica que sirve de base para las especializaciones del modo de comunicación (publicador o suscriptor).
- ***metadata_traits:*** Es una clase genérica que sirve de base para las especializaciones de los tipos de datos (puntos, alarmas, eventos, comandos, estado de la comunicación y bitácoras).

Communication_data_types:

En este paquete se definen los tipos de datos que se manejan en el SCADA SAINUX, como son alarmas, eventos, puntos, comandos, estado de la comunicación y bitácoras utilizando el lenguaje de definición de interfaces “IDL”. En este los ficheros creados tendrán la extensión .idl, y al usar el compilador IDL se generan los stub y skeleton que definen cada tipo de dato como tal, que son los que cada módulo va a utilizar para el manejo de los mismos.

3.3 DESPLIEGUE DEL SUBSISTEMA DE COMUNICACIONES

El siguiente gráfico fue realizado por los autores, para mostrar las relaciones físicas entre los distintos nodos que componen el sistema y el reparto de los componentes sobre dichos nodos. El subsistema de comunicaciones de SAINUX tiene la siguiente disposición física:

Los clientes pueden ser cualquiera de los módulos de SAINUX que pueden publicar o suscribirse a datos y además existe un servidor que es donde se encuentra el *DCPSInfoRepo* que actúa como el mecanismo de descubrimiento entre los publicadores y suscriptores.

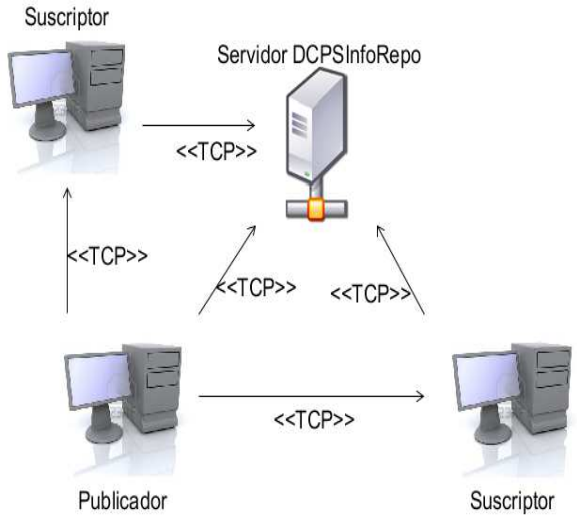

Figura. 4: Despliegue del subsistema de comunicaciones

4. CONTRIBUCIÓN Y APORTE SOCIAL:

El desarrollo del subsistema para la comunicación entre los diferentes módulos de SAINUX, utilizando una tecnología que permitirá el uso de dicho SCADA tanto en empresas cubanas como extranjeras, representa un aporte económico para el país desde el punto de vista de la sustitución de importaciones. La elaboración de un SCADA, su configuración y despliegue está calculado en miles de USD. Además, como parte de la solución del sistema, permitirá que SAINUX pueda ser desplegado en posibles clientes como Aguas de Santiago, Instituto de Meteorología Nacional, Supervisión energética UCI, Centro de Inmunología Molecular y Sismología.

5. CONCLUSIONES

Luego de finalizada la presente investigación para el desarrollo del subsistema de comunicaciones del SCADA SAINUX, se concluye que:

- Con el uso de *OpenDDS*, el subsistema de comunicaciones permite el envío asíncrono de mensajes, desacoplando en tiempo y espacio a los participantes (publicadores y suscriptores).
- El subsistema de comunicaciones desarrollado permite el envío y recepción de datos complejos, como son: puntos, alarmas, eventos, comandos, bitácoras y estado de la comunicación.
- El desarrollo del subsistema de comunicaciones, contribuye a que SAINUX pueda utilizarse en cualquier empresa de Cuba, que requiera automatización para sus procesos, así como la comercialización del mismo en el exterior.

REFERENCIAS

1. **Schmidt, Douglas C.: "**Middleware for Distributed Systems", 2005. Disponible en: http://repositorio.bib.upct.es/dspace/bitstream/10317/570/1/pfc2683.pdf
2. **Fernández, D.V.:** Documentación de ZeroC ICE, 2006; Disponible en: http://www.esi.uclm.es/www/dvallejo/docs/docICE.pdf
3. **Aragón Cáceres, José y Llanes Jiménez, A. Beatriz.** Servicio de Integración con Terceros para el acceso a variables del sistema SCADA Guardián del ALBA. Descripción general de un SCADA. Comunicaciones. Universidad de las Ciencias Informáticas, Ciudad de La Habana, 2009. Disponible en: http://bibliodoc.uci.cu/TD/TD_2554_09.pdf
4. **Guibert Nápoles, Rosalbis y Cárdenas del Valle, Yusniel:** "Mecanismos de seguridad para el middleware del SCADA Guardián del ALBA", Tesis de grado, Universidad de las Ciencias Informáticas, Ciudad de La Habana, 2008.
5. **Vega, J.M.L.** Plataforma de Trabajo Colaborativo sobre Middleware DDS. 2008; Disponible en: http://dtstc.ugr.es/tl/pdf/pf/PFC_jmlvega_Final_rev04.pdf
6. **Inc, O.C. OpenDDS.** 2012. Disponible en: http://www.OpenDDS.org/
7. **OpenDDS. Articles.** Introduction to OpenDDS. DDS Architecture. 2012. Disponible en: http://www.OpenDDS.org/Article-Intro.html
8. **Grady, J.B.** El Lenguaje Unificado de Modelado. 2007
9. **Portal EcuRed.** Visual Paradigm. 2012. Disponible en: http://www.ecured.cu/index.php/Visual_Paradigm
10. **Pérez Javier, Maikel:** "Selección de tecnologías para el desarrollo del Subsistema de Comunicación con Terceros", Proyecto SCADA Guardián del ALBA, Universidad de las Ciencias Informáticas, Ciudad de La Habana, 2009.

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.