Universidad de las Ciencias Informáticas

Facultad 5

DESARROLLO DE UN EXPLORADOR CLIENTE *OBJECT LINKING AND EMBEDDING FOR CONTROL PROCESS OF DATA ACCESS (OPC DA)*

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autora: Annaliet Iglesias Machado

Tutor: Ing. Evián Suárez Rodríguez

Co-Tutor: Ing. Adrián Carlos Moreno Borges

La Habana, Julio del 2011

"Año 53 de la Revolución"

La conclusión es que sabemos muy poco y sin embargo es asombroso lo mucho que conocemos.

Y más asombroso todavía es que un conocimiento tan pequeño pueda dar tanto poder.

Bertrand Arthur William Russell

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Annaliet Iglesias Machado	Ing. Evián Suárez Rodríguez
Autora	Tutor

Datos de contacto

Generales del tutor:

Nombre y apellidos: Ing. Evián Suárez Rodríguez
Especialidad: Ingeniería en Ciencias Informáticas
Años de experiencias: 2 años
Correo electrónico: esrodriguez@uci.cu
Teléfono particular: 837 2308

Resumen

La presente investigación se desarrolla con el objetivo de proveer al Centro de Informática Industrial (CEDIN) de la Facultad 5, de un Explorador Cliente *Object Linking and Embedding for Control Process of Data Access* (OPC DA, por sus siglas en inglés), que pueda interactuar con Servidores OPC DA.

Para ello se realiza un análisis sobre el estándar OPC describiendo brevemente cada una de las especificaciones que brinda, y se enfatiza en la especificación asociada con el acceso a datos. Se describen las tecnologías, herramientas, lenguajes y la metodología de desarrollo de software a utilizar para la implementación del Explorador Cliente OPC DA.

Como resultado, el CEDIN cuenta con un Explorador Cliente OPC DA desarrollado sobre el framework Qt, preparado para crecer o adaptarse según las necesidades cambiantes de la empresa o instalación industrial. Este explorador le brinda al usuario la posibilidad de interactuar con los datos de las variables almacenadas en los Servidores OPC DA, que estén conectados al explorador. Además, es una herramienta de pruebas para la línea de trabajo OPC, pues sirve para evaluar las funcionalidades que se implementan en el Servidor OPC que se esté desarrollando.

Palabras clave: Cliente OPC DA, Estándar OPC DA, Framework Qt.

Índice de contenido

Índice de figuras

Índice de tablas

Introducción

Las nuevas tecnologías de la información y las comunicaciones, están teniendo un fuerte impacto en las diferentes ramas de la sociedad; las mismas constituyen un conjunto de técnicas y dispositivos avanzados que integran funcionalidades de gestión y control de datos que se han desarrollado durante los últimos años en dichas ramas, para lograr una mayor eficiencia en los servicios.

Actualmente existen sistemas informáticos en diferentes sectores de la sociedad, como son: la Cultura, el Deporte, la Ciencia, la Recreación, entre otros. Uno de los sectores donde se está llevando a cabo la informatización es el sector Industrial, ya que permite mejorar la productividad de la empresa, realizar operaciones de forma rápida y precisa, simplificar el mantenimiento de la instalación y controlar el proceso en tiempo real. La automatización industrial se ha convertido en un área de gran importancia y en pleno desarrollo. El principio básico de los procesos automatizados es que el hombre no intervenga en ellos, y si lo hace, deberá ser lo menos posible.

El mecanismo de control de procesos industriales Sistemas de Control, Supervisión y Adquisición de Datos (por sus siglas en inglés SCADA), surge con la necesidad de mantener una supervisión constante del funcionamiento de los procesos industriales y el incremento sustancial del papel que juega la automática dentro de estos procesos, con el objetivo de garantizar su seguridad.

En la actualidad se hace necesaria la integración tecnológica de los SCADA, con dispositivos y sistemas externos. Dada la gran variedad de productores de hardware y de software a nivel mundial, la integración con sistemas de supervisión y control se ha hecho posible mediante el estándar OPC, el cual proporciona un mecanismo para extraer datos de una fuente y comunicarlos con cualquier aplicación cliente.

El estándar OPC es el más usado en la intercomunicación entre diferentes sistemas de automatización. El desarrollo de productos basados en este estándar, garantiza una alta compatibilidad entre plataformas tecnológicas en el área de la automatización industrial, permitiendo a su vez, una independencia y generalización que liberará a las industrias de utilizar sólo la plataforma de programación propia del fabricante, y supondrá un abaratamiento en la inversión, debido a que ahora, el usuario tendrá mayores alternativas para escoger los equipos con la tecnología que considere más conveniente para sus procesos.

En el mundo existen varios desarrolladores de aplicaciones supervisoras, entre las que se encuentran: Matrikon OPC, SCADA/HMI Movicon de la firma italiana Progea [1] [2] los sistemas SCADA de la Nacional Instrument, LabViewy LookOut [3] [4]. Otras firmas que soportan OPC en sus sistemas supervisorios son OMRON [5] y WinCC de Siemens [6], las cuales han incorporado OPC en su desarrollo.

En la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI), fue creado el Centro de Informática Industrial (CEDIN), que se dedica al desarrollo de productos de software destinados al control de los procesos industriales. En este centro se encuentra el proyecto ¨SCADA Guardián del Alba¨ o ¨SCADA Galba¨ que está constituido por 7 módulos fundamentales: Visualización HMI (Interfaz Hombre Máquina), Capa de Conectividad, Base de Datos Histórica, Configuración, Seguridad, Núcleo de Procesamiento de Datos y Manejadores.

El módulo de los manejadores de dispositivos, asegura la comunicación del sistema de supervisión y control con los dispositivos de campo, se relaciona con el módulo núcleo de procesamiento de datos, encargado del tratamiento y análisis en tiempo real de información adquirida desde el campo.

Este sistema de adquisición tiene como función principal la comunicación directa con la interfaz genérica de *driver*, para la lectura y escritura de datos de dispositivos. La interfaz genérica propone una interfaz estándar de acceso a los dispositivos que oculte al sistema de supervisión y control los diferentes protocolos sin que impacte negativamente en el rendimiento de los drivers (*manejadores*) de los dispositivos de adquisición de datos, que son aplicaciones de software que permiten interactuar con los periféricos conectados a él.

En el CEDIN se necesitaba analizar las tramas, unidades de envió de datos del SCADA Galba y conjuntamente controlar el funcionamiento de los *Driver*. Con el surgimiento del Recolector Gráfico se resuelven estos problemas, y además, se garantiza la recolección de los puntos configurados de acuerdo a sus períodos de encuesta con la mayor concurrencia posible, muestra en despliegues tabulares los valores obtenidos de las variables, así como sus marcas de tiempo y calidad.

OPC está basado en tecnología COM/DCOM, pudiendo tener una implementación multiplataforma aunque para el Sistema Operativo (SO) Linux no es 100% segura. Dado este problema surge el *Gateway* OPC DA, que se conecta de manera asincrónica con el *Driver* OPC, éste funciona como pasarela entre el *Driver*

OPC y el Cliente OPC DA, con la finalidad de permitir que el primero, como aplicación remota, acceda a los datos de los Servidores OPC.

El *Gateway* OPC DA se conecta con el Cliente OPC DA y éste a su vez con el Servidor OPC DA que tiene la información de los dispositivos conectados a él. El Cliente OPC DA cuenta con las funcionalidades mínimas de lectura y escritura en tiempo real, de manera sincrónica, provocando que el hilo de la invocación quede bloqueado desde la petición hasta el retorno de la misma, es decir, no se puede ejecutar otra tarea hasta que se reciba una notificación que la instrucción anteriormente invocada ha sido ejecutada.

Según la **situación problémica** anteriormente descrita, el **problema a resolver** que se identifica para la presente investigación consiste en: ¿Cómo intercambiar información de manera asincrónica con Servidores OPC DA en procesos industriales?

La investigación se enmarca en el **objeto de estudio** asociado al proceso de intercambio de información de manera asincrónica en procesos industriales, que tiene como **objetivo general:** Implementar un cliente para intercambiar información de manera asincrónica con Servidores OPC DA en procesos industriales. Centrándose en el **campo de acción** intercambio de información de manera asincrónica con Servidores OPC DA en procesos industriales.

Para dar cumplimiento al objetivo anteriormente planteado se definen las siguientes **tareas de investigación:**

- Elaboración del marco teórico a partir del estado del arte existente sobre el tema a investigar.
- Valoración crítica de los diversos Clientes OPC DA existentes en el mercado, tomándolos como punto de partida para la especificación de requerimientos de la solución a implementar.
- Selección de las tecnologías, metodología, lenguajes y herramientas de desarrollo a utilizar para la implementación de la solución.
- Identificación de los requerimientos necesarios para la implementación del estándar OPC DA.

- Análisis y diseño de la arquitectura jerárquica de la interfaz visual del explorador Cliente OPC DA según los requerimientos especificados.
- Implementación de la arquitectura jerárquica de la interfaz visual del explorador Cliente OPC DA y someterla a pruebas de rendimiento.

Para guiar la investigación se proponen la siguiente **idea a defender:** Con el desarrollo del Explorador Cliente OPC DA, se obtendrá un producto capaz de intercambiar información con un Servidor OPC DA, de manera asincrónica mediante la especificación de acceso a datos, y probar las funcionalidades implementadas en el Servidor OPC DA.

Entre los **métodos de la investigación científica** utilizados se encuentran los siguientes:

Métodos teóricos:

- El método analítico – sintético, para descomponer el problema de investigación en el estudio por separado del estándar OPC y sus especificaciones, para luego sintetizarlos en la solución general del problema planteado.
- El método histórico – lógico, para determinar las tendencias actuales, en este caso se empleó para realizar el estudio de la trayectoria real de determinados clientes que servirán de guía para la construcción del Cliente OPC DA.
- El método modelación, para modelar cada uno de los artefactos generados en las fases que propone la metodología de desarrollo de software utilizada.

Métodos Empíricos:

- El método observación, para conocer y analizar las características específicas que posee un Cliente OPC DA, permitiendo proponer las mejoras que en la investigación se detallan.

El presente documento se encuentra estructurado en tres capítulos:

El capítulo 1: **"Fundamentación Teórica",** aborda detalladamente todo lo referente a la fundamentación teórica que sustenta este trabajo de diploma. Se describen los conceptos fundamentales, y se realiza una selección de las tecnologías, metodología, lenguajes y herramientas de desarrollo de software.

El capítulo 2: **"Diseño de la Propuesta de Solución",** expone el modelo conceptual, se definen todas las funcionalidades del sistema, así como la arquitectura que sustenta la solución propuesta.

El capítulo 3: **"Implementación y Pruebas",** refleja la modelación de cada uno de los artefactos obtenidos en la fase de implementación y se realiza una valoración crítica, tanto desde el enfoque cuantitativo como cualitativo, de los resultados obtenidos, a partir del desarrollo de un conjunto de pruebas experimentales realizadas al sistema propuesto como solución.

Como resultado final se contará con un Cliente OPC DA que permita el intercambio de información de manera asincrónica contenida en los Servidores OPC DA. Además, servirá como herramienta para probar el Servidor OPC DA que está en fase de desarrollo en el centro, siendo por lo tanto un utilitario más con el que contarán los productos OPC.

Capítulo 1: Fundamentación Teórica.

Introducción

En el presente capítulo se realiza el estudio sobre el marco teórico referente a los mecanismos estándares de comunicación, entre los que se destaca su definición, surgimiento y arquitectura que propone. Se caracterizan las tecnologías, metodología, lenguajes y herramientas de desarrollo necesarias para desarrollar el mecanismo de comunicación OPC DA.

1.1. Concepto de OPC

OPC, es un conjunto de estándares, desarrollados y mantenidos por la Fundación de OPC, basados en tecnología COM, que en la actualidad forma parte del conocido *The Microsoft.NET Framework*®.

OPC utiliza a .NET para proveer conectividad, *plug-and-play*, y la interoperatividad entre dispositivos de automatización, sistemas y software de firmas diferentes. La tecnología COM, provee la infraestructura de software necesaria que define cómo una aplicación que comparte datos bajo los sistemas operativos de Microsoft como Windows. Las especificaciones de OPC, definen un conjunto de objetos COM, métodos y propiedades que se comparten, en tiempo real, en los sistemas automatizados [7].

Dentro de las Ventajas que ofrece OPC se encuentran:

- Provee una estructura estándar multi-fabricante bajo los preceptos de *plug-and-play*. El hardware y software, de suministradores diferentes, pueden ser integrados con facilidad y transparencia.
- Remueve las barreras entre los fabricantes de los dispositivos y los desarrolladores de software.
- Incrementa la flexibilidad, reduce el tiempo de integración, desarrollo e instalación del sistema.
- Minimiza los costos de los sistemas automatizados.
- Presenta un modelo que elimina la necesidad de *drivers* redundantes.
- El costo de desarrollo y montaje es mucho menor que con las soluciones de un sólo fabricante.

1.1.1. Ventajas y aplicación del modelo de la tecnología de OPC

Antes de la creación de OPC, las aplicaciones de gestión de la información o supervisión se manipulaban por *drivers* específicos ofrecidos por el fabricante de los dispositivos, lo que proporcionaba que los suministradores de estos drivers ofrecieran una solución completa a los sistemas automatizados.

Ventajas del modelo basado en el estándar OPC

A continuación se mencionan algunas de las ventajas que ofrece el estándar OPC en los sistemas automatizados:

- Sólo se requiere que el fabricante ofrezca un *driver*, el Servidor OPC para dispositivo, no depende de las aplicaciones.
- Se disminuyen considerablemente las solicitudes a los dispositivos del proceso.
- Las aplicaciones de supervisión y gestión de la información no requieren conocer los *drivers* del fabricante, sólo el estándar OPC que es el mismo para todos los Servidores OPC.
- Los tiempos de desarrollo, montaje y programación son minimizados, según estadísticas de la firma Matrikon [15], en casi cinco veces con respecto al modelo del fabricante no estandarizado.
- Los costos se disminuyen en casi cinco veces según la firma Matrikon [15].
- Sólo se requieren de conocimientos básicos de OPC.

Aplicación del modelo basado en el estándar OPC

Como ejemplo ilustrativo de este modelo se presenta el caso de un sistema de control de un proceso que requiere de tres aplicaciones básicas de análisis de los datos, la Interfaz visual para interactuar con el proceso, HMI, almacenamiento de los datos históricos y un sistema de monitoreo de la maquinaria. Ver **Figura 4**.

Para completar la parte del proceso, tres dispositivos generan los datos: un PLC que controla la planta, un equipo de monitoreo de vibraciones y un motor de cálculo que ofrece resultados estadísticos del proceso.

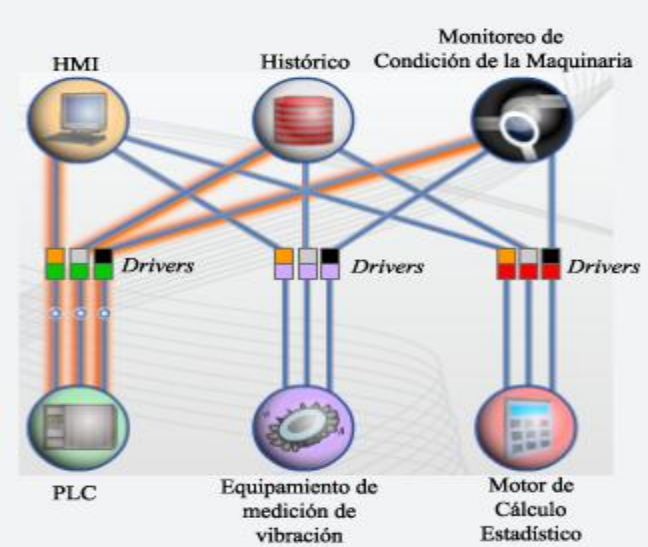


Figura 1: Modelo de sistema sin cumplir el estándar de OPC

En el modelo no estandarizado, cada aplicación requiere de un *driver* específico que se comunique con cada uno de los dispositivos del proceso, en este caso cada aplicación requiere de tres *drivers*, ver **Figura 4**, lo que hace que se requiera tres solicitudes a los dispositivos del proceso y complique a sobremanera el sistema. En esta figura se muestran resaltadas las líneas de intercambio de datos entre el PLC y las tres aplicaciones.

Si el ejemplo anterior se implementa bajo los preceptos de OPC, se observará los siguientes cambios.

- Se colocará como *driver* común, a cada dispositivo, un Servidor OPC, en este caso un servidor para manipular los intercambios con el PLC, otro con el equipo de monitoreo de vibraciones y otro con el dispositivo de cálculo especializado, ver **Figura 5**.

- Las tres aplicaciones accederán a los dispositivos por medio de los Servidores OPC correspondientes, minimizando la cantidad de solicitudes a los dispositivos. La **Figura 5**, resalta la interconexión entre las tres aplicaciones y el PLC, observándose que existe sólo una conexión

entre el Servidor OPC y el dispositivo PLC. Ésta minimiza las solicitudes a las fuentes de datos, elimina la complejidad del sistema y mejora su eficiencia.

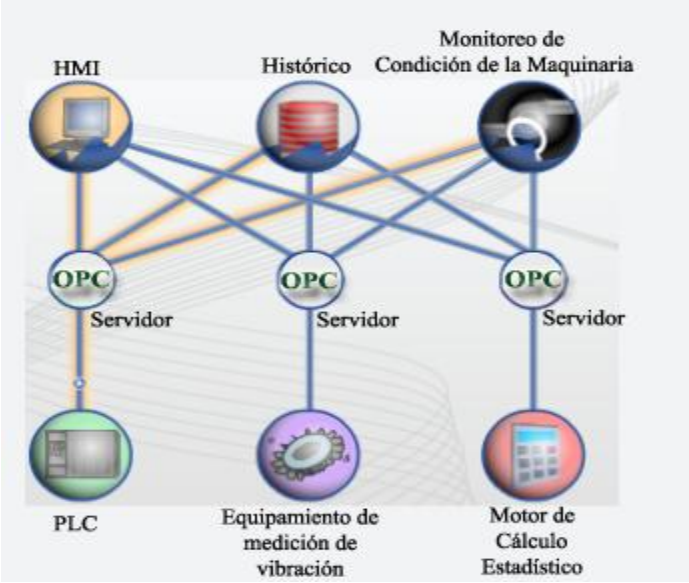


Figura 2: Modelo del estándar de OPC

1.2. Desarrollo histórico de OPC

La necesidad de implementar sistemas donde el proceso automatizado y los sistemas de gestión de la información se unan, para formar un todo único, ha sido uno de los principales problemas a resolver por parte de las firmas especializadas en automatización. En muchas ocasiones los fabricantes crean una barrera al desarrollo de sistemas de control de procesos, debido a la no-compatibilidad de sus dispositivos e interfaces de comunicación.

Esta necesidad crea un problema a los desarrolladores de sistemas automatizados que se limitan a utilizar el hardware y software de un determinado suministrador, al no poder comunicar los sistemas de una firma con los de otra. OPC brinda una solución para el intercambio de información entre los dispositivos y las aplicaciones de producción y gestión de la información, sin importar el fabricante y bajo los preceptos del *plug and play* [7].

En el mundo industrial, la integración de componentes desde diversos fabricantes ha resultado frecuentemente una tarea difícil, e incluso en la actualidad en ciertos casos lo sigue siendo. Las aplicaciones, como operaciones de control y monitorización o procesamiento de datos de gestión, requieren de un driver para cada componente de automatización cuyos datos necesiten ser leídos.

1.2.1. Surgimiento de OPC

En Mayo de 1995, *Microsoft*, unida a una fuerza constituida por cinco empresas, *Intellution*, *Opto-22, Fisher-Rosemount, Rockwell Software e Intuitiv Software* crearon un comité con el objetivo de definir completamente las interfaces basadas en el estándar OLE/COM y de esa forma crear el primer borrador de las especificaciones OPC, que fue completado al final de 1995. Con la colaboración de unas 90 compañías a lo largo del mundo, fueron comprobadas estas especificaciones, para finalmente dar el primer conjunto de especificaciones completado en Agosto de 1996. El objetivo del comité fue proporcionar un interfaz de programación de aplicaciones estándar para el intercambio de datos que puede simplificar el desarrollo de drivers de entrada/salida y mejorar el rendimiento de los sistemas de interfaz [8].

Seguidamente en Septiembre de 1996 se crea la Fundación de OPC [9], que es una organización global, independiente y sin fines de lucro, a la cual se le asignó la tarea de establecer varios comités técnicos para extender el espectro y la función de las especificaciones de OPC. El desarrollo y mantenimiento de estos estándares fueron asumidos por dicha fundación.

1.3. Desarrollo de OPC en el mundo

Como se había mencionado anteriormente, en el mundo existen varios desarrolladores de aplicaciones supervisoras que han incorporado en sus aplicaciones OPC, por las ventajas que ofrece este estándar.

Dentro de estos desarrolladores se encuentran:

MatriconOPC:

- **Matrikon OPC Explorer:** es un Cliente OPC diseñado para ayudar durante la instalación, prueba y configuración de Servidores OPC. Además se instala fácilmente y proporciona resultados de forma

rápida gracias a su interfaz de usuario intuitiva y flujo de trabajo optimizado. Tiene la desventaja de que no se puede conectar ningún servidor de manera remota [10].

- **Matrikon OPC for Crystal Reports:** está diseñado para proporcionar a los usuarios acceso instantáneo a sus datos almacenados en su OPC historiador en vez de tener que duplicar los datos en una base de datos relacional [11].

KEPServerEx:

- **KepWare OPC Quick Client:** es un Cliente OPC que soporta todas las operaciones disponibles en cualquier aplicación Cliente OPC. Con el uso del OPC Quick Client se puede acceder a todos los datos disponibles en su servidor de aplicaciones, incluido el sistema de diagnóstico, y el usuario define las etiquetas. El OPC Quick Client permite leer y escribir datos, y conocer el rendimiento del servidor de prueba. El OPC Quick Client proporciona información detallada acerca de los errores OPC devuelto por el servidor que sirve para diagnosticar un Cliente/Servidor OPC [12].

1.4. Desarrollo de la tecnología OPC

Microsoft ha evolucionado los modelos de intercambio de datos entre aplicaciones desde la aparición del *Dynamic Data Exchange* (DDE) y la creación de la tecnología *Object Linking and Embedding* (OLE), que manipula objetos vinculados e insertados [13].

La tecnología OLE, tuvo varias transformaciones hasta la creación del denominado *Component Object Model (*COM*),* método que utilizó Microsoft para intercambiar los datos por medio de objetos, propiedades y eventos, compartidos por medio de interfaces [14].

La tecnología COM, tuvo sus variaciones para lograr su portabilidad en red para convertirse en *Distributed Component Object Model* (DCOM), modelo que intercambia datos entre clientes y Servidores OPC COM en red.

El estándar OPC, fue creado basado en la tecnología OLE/COM, se orientó a interfaces para facilitar el intercambio de datos y la portabilidad de las versiones, por la facilidad que reporta COM en estos

aspectos. Las especificaciones de OPC fueron evolucionando a medida que estas tecnologías, OLE/COM de Microsoft, mejoraron.

Como las tecnologías de Microsoft (DDE, OLE, COM, COM+) no eran compatibles en plataformas que no sean de la compañía, surge la especificación de OPC XML, como medio de comunicación entre los Servidores OPC que se encuentren desarrollados en plataformas diferentes, lo que facilita su publicación por medio de protocolos como los usados en Internet.

El desarrollo de las especificaciones de OPC, no sólo estuvo regido por el desarrollo de las tecnologías de Microsoft, sino también por los sistemas automatizados y de gestión actuales, que trajeron consigo la creación de nuevas especificaciones, OPC *Data Access*, especializada en el intercambio de datos en tiempo real, OPC *Alarms&Events*, utilizada para el monitoreo de los eventos existentes en el proceso, *Historical Data Access*, para el almacenamiento de los datos históricos, OPC *Batch*, y OPC *Security,* para la transmisión por lotes y seguridad, respectivamente, unidas a dos nuevas especificaciones, OPC *Data eXchange* (DX) y OPC XML DA, para intercambio de datos entre Servidores OPC y compatibilidad entre plataformas diferentes.

1.5. Especificación OPC DA

La especificación OPC DA, es la más utilizada en los sistemas automatizados actuales por su versatilidad; los muestreos en tiempo real son la base de todos los controles de procesos industriales. Dado a que es la especificación que permite el intercambio de datos en tiempo real y siendo éste el centro de la investigación, a continuación se detallan aspectos importantes de la misma [7].

Desde su creación de 1996, los estándares de OPC han evolucionado en gran medida y esta especificación ha sido la más actualizada de todas. Existiendo versiones desde la 1.0 hasta la 3.0. La descripción del estándar OPC DA, tratada en este capítulo, se realizó basándose en la versión 2.5.

1.5.1. Descripción del estándar OPC-DA

Las especificaciones de Acceso a Datos de OPC, se especializan en la transmisión, planificación e intercambio de datos entre el servidor y sus clientes. Este estándar, define las especificaciones orientadas a interfaces COM, tanto para los clientes como para los Servidores OPC DA.

La **Figura 1** muestra un sistema de control y supervisión de flujo utilizando el estándar de OPC DA. El sistema está formado por una bomba que se encuentra controlada por un PLC y un flujómetro que realimenta al PLC.

Los clientes de OPC DA podrán acceder a los datos captados por medio del servidor de OPC DA que se actualiza a través del *driver* de comunicación con el PLC. En este caso existe un cliente en la misma PC del servidor, HMI y dos Clientes OPC en red con aplicaciones que requieren datos del proceso. Estas aplicaciones, pueden o no, ser del mismo fabricante, siempre y cuando cumplan con el estándar OPC DA.

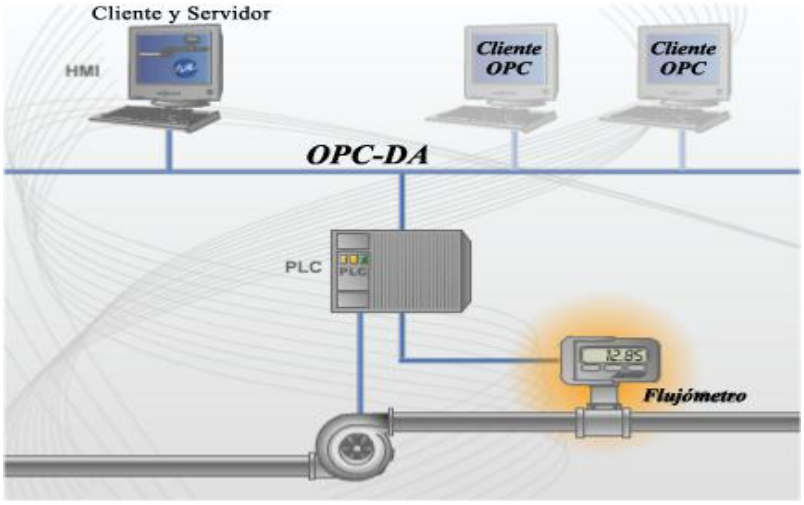


Figura 3: Ejemplo de sistema de control de flujo utilizando el estándar OPC DA, en aplicaciones de supervisión con varios clientes

Grupo de OPC

Cada grupo de OPC DA, tiene un nombre único. El Cliente OPC puede especificar si un grupo determinado está en estado inactivo. Los grupos de OPC DA pueden ser creados dinámicamente acorde a las aplicaciones que lo requieran (clientes). Un cliente puede agrupar varios *ítems* en un grupo para organizar los accesos al Servidor OPC. Por ejemplo un sistema supervisor que requiera muestrear varias variables con el mismo tiempo de muestreo, puede organizarlas en un mismo grupo, así también pueden realizarse grupos lógicos que representen zonas del proceso [7].

Un grupo presenta las siguientes propiedades:

- ***Name:*** identificador único, en el mismo deben respetarse las mayúsculas y minúsculas.
- ***Active:*** define si un grupo está activo, en caso contrario, todos los *ítems* que contiene no serán actualizados.
- ***Ítems:*** esta propiedad define la cantidad de *ítems* incluidos en el grupo.
- ***Período:*** esta propiedad, define el tiempo de actualización del dato en milisegundos (ms), desde el servidor. Todos los *ítems* que están incluidos en el grupo serán refrescados por el servidor en el tiempo descrito, en caso contrario se genera una excepción.
- ***Por ciento de banda muerta:*** el rango de la zona muerta es desde 0 hasta 100. La zona muerta es aplicada sólo a valores del tipo analógico, esta propiedad se utiliza para generar excepciones en caso de que se superen los límites permitidos. Esta propiedad puede colocarse cuando se crea el nuevo grupo, de esta forma todos los ítems internos presentarán el mismo valor. La principal aplicación de esta propiedad es hacer un filtrado en caso de que los valores estén defectuosos por causa del ruido.

***Ítem* de OPC**

A diferencia de los grupos y Servidores OPC, los *ítems* no implementan ninguna interfaz de OPC y por lo tanto no es un objeto COM. Es un objeto interno que el Servidor OPC almacena. Contiene la información que debe ser compartida al cliente, por ejemplo, tipo de datos, valor, si está activo, si está protegido contra escritura, etc. Desde la perspectiva de un Cliente OPC, un *ítem* representa la conexión lógica a la fuente de datos que comparte dicho objeto [7].

Usando el identificador del *ítem*, ItemID, el cliente puede realizar las solicitudes o escrituras de los datos compartidos, teóricamente no existe límite para la cantidad de *ítems* que publica un servidor, incluso en un sólo grupo del servidor.

Un *ítem* provee un punto de conexión entre el objeto físico o lógico que ofrece el valor, por ejemplo una consigna de control. Las propiedades de un *ítem* son:

- ***Id:*** identificador único, en el mismo deben respetarse las mayúsculas y minúsculas.
- ***Valor:*** valor actual del *ítem*, entregado por el Servidor OPC DA.
- ***TimeStamp:*** tiempo en que fue muestreado el dato.
- ***Calidad:*** calidad del valor leído, puede ser buena o mala en caso de errores en el servidor.
- ***Tipo de Dato:*** tipo de dato en que se entrega el valor del *ítem* al cliente.
- ***Acceso:*** si el *ítem* presenta permiso de lectura y/o escritura.

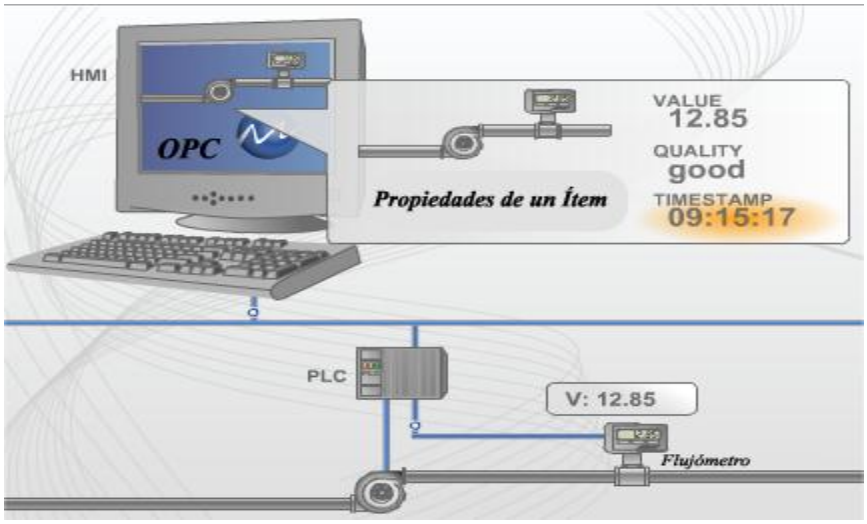


Figura 4: Propiedades de un ítem de OPC DA, en un sistema de control de flujo

Identificador del *ítem*, Id

El identificador del *ítem*, es usado por los clientes para establecer la conexión al Servidor OPC, no es más que una cadena compuesta por el camino de todos los niveles que presenta el *ítem*. Ejemplo de ello es *ServidorOPC1.Grupo2.Item3*

Interfaces para aplicaciones del Cliente OPC DA

Las aplicaciones Clientes OPC DA no requieren implementar todas las interfaces, todo lo que requieran del servidor deben solicitarlo a la interfaz correspondiente. Sin embargo, requieren implementar una serie de interfaces que son la base del mecanismo de suscripción y notificación de eventos que sucedan en el servidor y que son de gran importancia para los clientes.

Un ejemplo de lo anterior son las actualizaciones de los valores de los *ítems*, estas operaciones deben ser notificadas por el servidor hacia todos los clientes interesados en el *ítem* modificado.

Las aplicaciones Clientes OPC DA, básicamente deben implementar dos interfaces de eventos. La IOPCShutDown, especializada en las notificaciones de desconexión del servidor y la interfaz IOPCDataCallback, especializada en las notificaciones de operaciones de lectura y escritura de *ítems*.

A continuación se describen los métodos de dichas interfaces:

IOPCDataCallback: Una aplicación cliente debe soportar esta interfaz para que el Servidor OPC DA notifique los cambios que existen en los *ítems* que el cliente esté interesado. El mecanismo para soportar estas notificaciones está basado en los puntos de conexión que existen en el servidor que se obtienen a través de la interfaz IConnectionPointContainer por medio de los métodos FindConnectionPoint o EnumConnectionPoints. El servidor obtendrá el puntero a la interfaz IOPCDataCallback del cliente que a su vez habrá implementado los métodos que se invocarán cuando ocurra el evento. Los eventos que debe implementar el cliente son:

- **OnDataChange**: se ocupa de las notificaciones en los grupos de los cambio de datos y refrescamientos.
- **OnReadComplete**: se ocupa de las notificaciones en los grupos cuando se efectúan lecturas asincrónicas.
- **OnWriteComplete:** se ocupa de las notificaciones en los grupos cuando se completan escrituras usando la interfaz IAsyncIO2.

- **OnCancelComplete**: se ocupa de las notificaciones en los grupos cuando se completan cancelaciones asincrónicas.

IOPCShutdown: una aplicación cliente debe soportar esta interfaz para que el Servidor OPC DA notifique las desconexiones del servidor. El mecanismo para soportar estas notificaciones está basado en los puntos de conexión que existen en el servidor que se obtienen a través de la interfaz IConnectionPointContainer por medio de los métodos FindConnectionPoint o EnumConnectionPoints. El servidor obtendrá el puntero a la interfaz IOPCShutDown del cliente que a su vez habrá implementado un método que se invocará cuando ocurra el evento. Un cliente, que posea conexiones a varios Servidores OPC, debe poseer una instancia de los métodos separados para cada servidor.

Los clientes deben implementar el siguiente método:

- **ShutdownRequest**: es proporcionado para que el servidor le notifique una desconexión. El cliente debe deshabilitar todas las conexiones, quitar todos los grupos y liberar todas las interfaces.

1.6. Arquitectura que propone OPC

La arquitectura OPC es un modelo Cliente/Servidor, donde el Explorador Cliente OPC DA se conecta con uno o varios Servidores OPC mediante la especificación OPC DA, y estos se comunican con los dispositivos mediante mensaje propietario. El servidor almacena los datos de los dispositivos conectados a él, y el cliente (Explorador Cliente OPC DA) es el encargado de visualizar estos datos a través de un explorador Cliente OPC DA. Ver **Figura 3**.

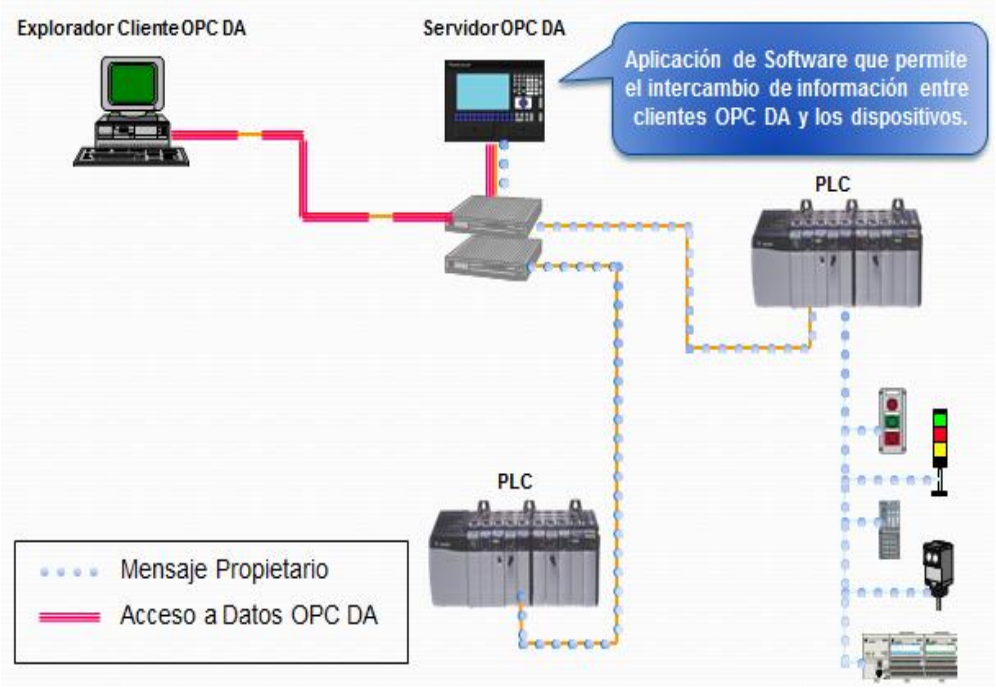


Figura 5: Arquitectura de OPC

1.7. Tecnologías, metodología, lenguajes y herramientas de desarrollo

La selección de las tecnologías, metodología, lenguajes y herramientas de desarrollo se basó en la definición realizada por el proyecto SCADA Galba en el documento de la Arquitectura de software v2.0, donde se especifica como metodología de desarrollo de software el proceso unificado de desarrollo de software *(por sus siglas en inglés* RUP, *Rational Unified Process)*, como IDE de desarrollo (*Integrated Development Environment*) Microsoft Visual C++ 2005, framework Qt como biblioteca multipropósito, para la implementación de la aplicación el lenguaje de programación C++, y para modelar cada uno de los artefactos que se generen, la herramienta case Visual Paradigm para el lenguaje unificado de modelado (por sus siglas en inglés UML, *Unified Modeling Language)*.

A continuación se describen brevemente algunas características de las tecnologías, metodología, lenguajes y herramientas de desarrollo seleccionadas:

1.7.1. RUP

Es una metodología que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Cuenta con cuatro fases (Inicio, elaboración, construcción, transición). En cada fase se ejecutarán una o varias iteraciones (que varían según el proyecto). Propone 9 flujos de trabajo (modelado del negocio, requisitos, análisis y diseño, implementación, prueba, despliegue, gestión de cambios y configuración, gestión de proyectos, gestión del entorno.)

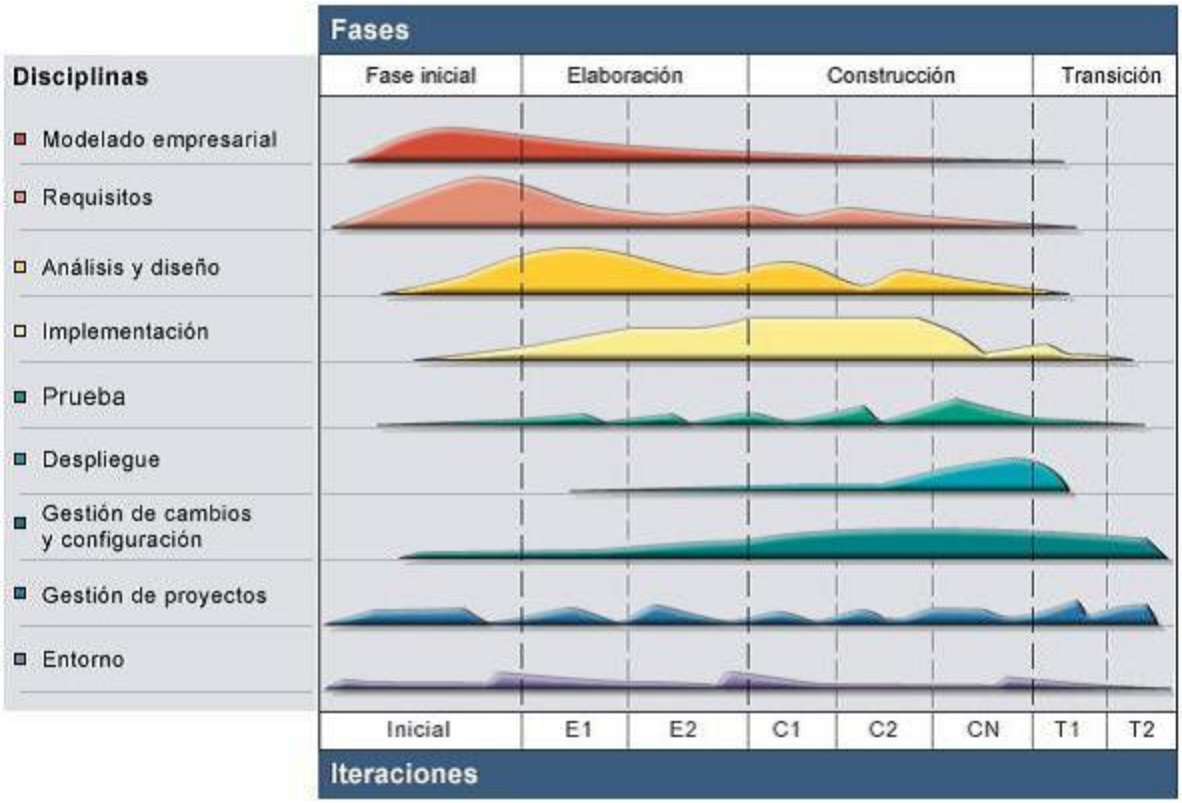


Figura 6: Vista general de RUP

El ciclo de vida de RUP se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por casos de uso.

1.7.2. Microsoft Visual C++ 2005

Microsoft Visual C++ 2005 proporciona un entorno de desarrollo eficaz y flexible para crear aplicaciones basadas en Microsoft Windows y en Microsoft .NET. Se puede utilizar como un sistema de desarrollo integrado o como un conjunto de herramientas individuales.

C++ es el lenguaje de alto nivel más popular del mundo, y Visual C++ ofrece a los desarrolladores una herramienta universal con la que se puede generar software. [16]

1.7.3. Framework Qt

Qt es una biblioteca multipropósito que permite el desarrollo de interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y Servidores OPC. Se puede integrar con distintos IDE existentes, como es el caso de Eclipse, Microsoft Visual Studio, Code::Blocks, entre otros. Utiliza el lenguaje de programación C++ de forma nativa, además existen interfaces para C, como: Python (PyQt), Java (QtJambi), Perl (PerlQt), Gambas (Gb.Qt), Ruby (QtRuby), PHP (PHP-Qt), Mono (Qtoto), entre otros. Está disponible para sistemas operativos como Microsoft Windows, Linux, Mac OS X, bajo licencias GPL v1, GPL v2, GPL v3 y LGPL.

1.7.4. Visual Paradigm para UML

Visual Paradigm es una herramienta que utiliza UML como lenguaje de modelado. Permite la generación de código e ingeniería tanto directa como inversa, utiliza el lenguaje de modelado UML para visualizar y diseñar los elementos de software. Posibilita el modelado de caso de usos incluyendo todas las funciones del diagrama de casos de uso, la generación de un diagrama de actividad, diagramas de clases y además, genera la documentación de proyectos automáticamente en varios formatos como PDF, HTML y formatos de Microsoft Word y puede integrarse con el control de versiones.

Esta herramienta soporta el ciclo de vida completo de desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegues.

1.7.5. UML

Es un lenguaje estándar de modelado para software, que permite la visualización, especificación, construcción y documentación de los artefactos de sistemas donde se manejan conceptos orientados a objetos. UML ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocios, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables [17].

1.7.6. C++

En la actualidad, C++ es un lenguaje versátil, potente y general. Mantiene las ventajas de C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original [18].

Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada que es una teoría de programación que consiste en construir programas de fácil comprensión, la programación genérica que está mucho más centrada en los algoritmos que en los datos y los algoritmos son parametrizados al máximo y expresados de la forma más independiente posible de detalles concretos, por último la programación orientada a objetos que se basa en la idea de un objeto, es una combinación de variables locales y procedimientos llamados métodos, que juntos, conforman una entidad de programación.

Conclusiones parciales

En este capítulo, se realizó un análisis de la documentación referente al estándar OPC para el establecimiento de los conceptos y definiciones asociados al mismo. Se efectuó una investigación sobre los Clientes OPC existentes en el mercado en busca de características y funcionalidades, para ello se tomaron como referencia MatrikonOPC Explorer y KepWare OPC Quick Client para el desarrollo de la solución. Se detallaron las especificaciones que brinda la Fundación OPC para la automatización industrial, seleccionando la especificación de Acceso a Datos (OPC DA) pues es la que se especializada en intercambio de datos en tiempo real.

Capítulo 2: Diseño de la Propuesta de Solución.

Introducción

En este capítulo se realiza una descripción del diseño de la propuesta de solución. Se exponen las principales características de las herramientas para el apoyo a la creación de Clientes OPC. Se desarrolla un modelo de dominio donde se expone un marco conceptual y se establecen las relaciones entre los conceptos que lo conforman. Se especifican los requerimientos funcionales y no funcionales, agrupándose los primeros en casos de uso, con el fin de estructurar el diagrama de casos de uso del sistema. En el diseño se modela y adquiere una forma del sistema para que soporte los requerimientos funcionales o no funcionales, contribuyendo a obtener una arquitectura sólida y estable para la futura implementación de la aplicación. Entre los artefactos que serán mostrados en el presente capítulo se encuentran: modelo de diseño, especificándose la estructura y la definición de los elementos que posee; diagramas de clases y descripción de las clases del diseño.

2.1. Modelo de dominio

Debido a la relativa simplicidad del entorno donde está enmarcado el sistema y el conocimiento que se posee acerca de su funcionamiento, no es necesario realizar un modelo de negocio para comprender la problemática que ha de resolverse, siendo suficiente un modelo de dominio.

El mismo es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real, no de componentes de software.

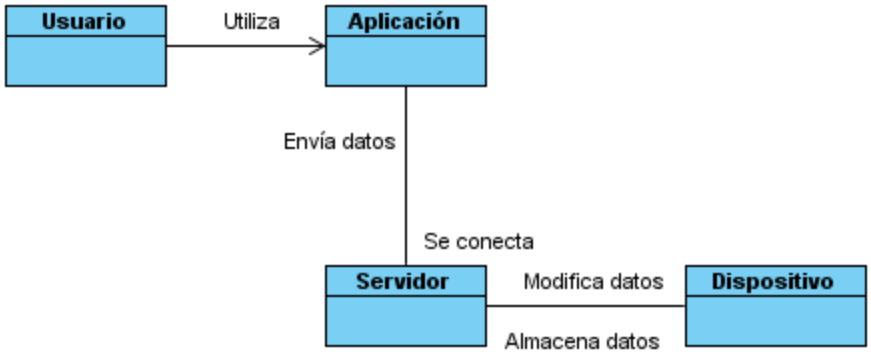


Figura 7: Diagrama de modelo del dominio

A continuación se representa y se describen los conceptos fundamentales del dominio del sistema.

- **Usuario:** persona capacitada para el manejo y funcionamiento del Explorador Cliente OPC DA.
- **Aplicación:** es una aplicación diseñada para visualizar y modificar los datos de los dispositivos almacenados en los Servidores OPC DA, esta aplicación también tiene la función de probar el o los Servidores OPC DA conectados a ella.
- **Dispositivo:** contienen los datos de los procesos de medición y monitoreo, puede ser un dispositivo de campo u otra fuente de datos.

2.1.1. Descripción del sistema propuesto

Después de haber realizado una investigación para resolver la situación problémica antes mencionada, se propone el desarrollo de una interfaz visual Cliente OPC DA, a la cual se le implementan las funcionalidades del estándar OPC descritas en los requerimientos funcionales del sistema.

Al realizar el estudio de las interfaces visuales de algunos de los Clientes OPC se tuvo una idea de cómo posicionar la información. La interfaz debe poseer una barra de herramienta donde aparezcan las funcionalidades más usadas por el usuario para un mejor acceso, además, la parte derecha de la aplicación contará con un área donde aparecerán enumerados los Servidores OPC y los grupos

pertenecientes a cada servidor, en la parte izquierda aparecerá un área en la que el usuario podrá conocer la información de los Servidores OPC conectados y los grupos creados, y en el área del centro es donde aparecerán los *ítems* añadidos por el usuario. Los elementos incluidos que deban resaltar tendrán un tamaño razonable para que no resulte difícil seleccionar alguno de ellos, y los íconos que se encuentren en la barra de herramientas facilitarán el acceso a las funcionalidades previstas.

2.2. Especificación de los requerimientos de software

La especificación de los requerimientos de software constituye un elemento de vital importancia para la elaboración de un software de calidad superior. Es necesario hacer énfasis en la precisión con que se debe realizar esta tarea pues tiene un papel primordial en la implementación de la interfaz, y se enfoca en un área fundamental: la definición de lo que se desea producir.

2.2.1. Requerimientos funcionales

Los requerimientos funcionales son condiciones o capacidades que el sistema debe cumplir. Estos definen el comportamiento interno de un software: cálculos, manipulación de datos y otras funcionalidades específicas. Una vez examinados los conceptos fundamentales descritos con mayor precisión en el modelo de dominio, se obtuvo el siguiente levantamiento de requerimientos funcionales.

Requerimientos funcionales:

- RF 1: Enumerar Servidores OPC locales.
- RF 2: Añadir Servidores OPC locales.
- RF 3: Añadir PC remota.
- RF 4: Añadir Servidores OPC remotos.
- RF 5: Eliminar dirección.
- RF 6: Conectar Servidores OPC.
- RF 7: Desconectar Servidores OPC.

- RF 8: Eliminar Servidores OPC.
- RF 9: Mostrar las propiedades de Servidores OPC.
- RF 10: Añadir de grupos.
- RF 11: Mostrar las propiedades de los grupos.
- RF 12: Activar grupos.
- RF 13: Desactivar grupos.
- RF 14: Eliminar grupos.
- RF 15: Adicionar *ítems*.
- RF 16: Mostrar las propiedades de los *ítems*.
- RF 17: Leer *ítems*.
- RF 18: Escribir *ítems*.
- RF 19: Eliminar *ítems*.
- RF 20: Cargar configuración.
- RF 21: Salvar configuración.

A continuación, los requerimientos funcionales serán agrupados en casos de uso del sistema. En este sentido los casos de uso identificados son los siguientes:

CUS 1: Administrar Servidores OPC locales

- RF 1: Enumerar Servidores OPC locales.
- RF 2: Añadir Servidores OPC locales.

CUS 2: Administrar Servidores OPC remotos

- RF 3: Añadir PC remota.
- RF 4: Añadir Servidores OPC remotos.
- RF 5: Eliminar dirección.

CUS 3: Operar Servidores OPC

- RF 6: Conectar Servidores OPC.
- RF 7: Desconectar Servidores OPC.
- RF 8: Eliminar Servidores OPC.
- RF 9: Mostrar las propiedades de Servidores OPC.

CUS 4: Gestionar grupo

- RF 10: Añadir de grupos.
- RF 11: Mostrar las propiedades de los grupos.
- RF 12: Activar grupos.
- RF 13: Desactivar grupos.
- RF 14: Eliminar grupos.

CUS 5: Operar *ítem*

- RF 15: Adicionar *ítems*.
- RF 16: Mostrar las propiedades de los *ítems*.
- RF 17: Leer *ítems*.
- RF 18: Escribir *ítems*.

- RF 19: Eliminar *ítems*.

CUS 6: Cargar configuración

CUS 7: Salvar configuración

2.2.2. Requerimientos no funcionales

Los requerimientos no funcionales son cualidades o propiedades que el producto debe tener, es decir, restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad del mantenimiento, extensibilidad y fiabilidad, que junto a las funcionalidades esperadas del software, ayudarán a marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. Éstos no describen lo que el software hará, sino cómo lo hará. Es preciso tener en cuenta que la definición de los requerimientos cobra un valor adicional porque los errores, debidos a requerimientos no funcionales, son difíciles y caros de resolver.

Requerimientos no funcionales de software

El cliente debe tener instalado el Sistema Operativo Windows, cualquiera de sus versiones.

Requerimientos no funcionales de hardware

Requerimientos mínimos:

- Ordenador Pentium.
- Memoria RAM de 128 MB.
- Monitor VGA.
- Teclado y mouse.
- Microprocesador a 1.6 GHZ.
- Disco duro de 20 GB.
- La PC de trabajo debe estar conectada a una Red de Área Local (LAN).

Requerimientos no funcionales de restricciones en el diseño e implementación

- Uso del framework Qt para el desarrollo de la interfaz visual.
- Emplear el paradigma programación orientado a objetos.
- C++ como lenguaje de programación.

Requerimientos no funcionales de usabilidad

Las operaciones deben ser lo más intuitivas posibles a la vista del cliente.

Requerimientos no funcionales de portabilidad

Permitir que el sistema se ejecute sobre el Sistema Operativo Windows 2000 o superior.

Requerimientos no funcionales de apariencia o interfaz externa

La interfaz debe ser sencilla y amigable. Diseño sencillo e intuitivo, permitiendo que no sea necesario mucho entrenamiento para utilizar el cliente.

2.3. Modelo de casos de uso del sistema

El modelo de casos de uso del sistema es un artefacto narrativo de ingeniería de software que describe, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del cliente, permitiendo de esta forma el establecimiento de un acuerdo entre clientes y desarrolladores sobre las condiciones y requerimientos que debe cumplir el sistema.

Este modelo está formado por actores, casos de uso y las relaciones que se establecen entre ellos, es decir, representa gráficamente a los procesos y su interacción con los actores y constituye una entrada de gran valor para las siguientes fases de construcción de un software.

Definición de los actores

Un actor del sistema es una persona o la abstracción de un software; son terceros fuera del sistema: puede intercambiar información con él, representar el rol que juegan una o varias personas, puede ser un recipiente pasivo de información, un equipo o un sistema automatizado. En el sistema en desarrollo se define el siguiente actor del sistema:

Actores del Sistema	Justificación
Usuario	Representa al usuario que va a hacer uso de la aplicación, teniendo la posibilidad de interactuar con las funcionalidades que le brinda la misma.

Tabla 1: Actor del sistema

Diagrama de casos de uso

En el siguiente diagrama se representan las relaciones de los actores y casos de uso del sistema:

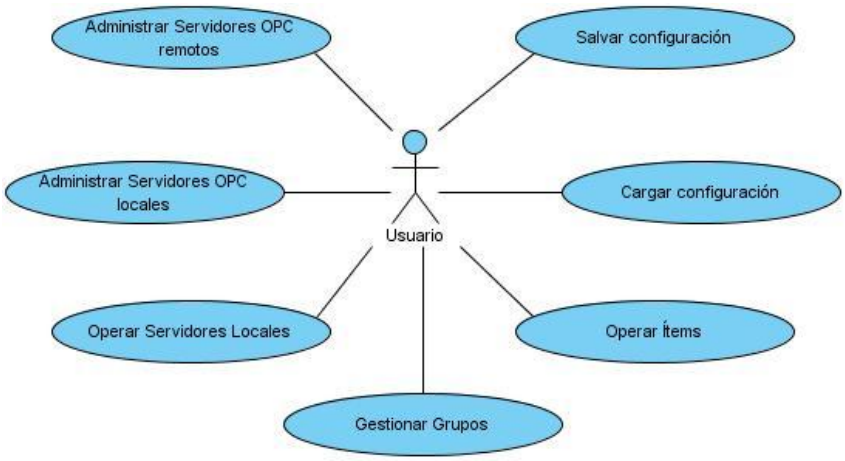


Figura 8: Diagrama de casos de uso del sistema

Descripción de los casos de uso del sistema

A continuación se describen brevemente los casos de uso identificados para ver la descripción detallada dirigirse a los anexos.

CUS Administrar Servidores OPC locales: El caso de uso se inicia cuando el usuario desea administrar los Servidores OPC locales, como resultado la aplicación muestra la lista de los Servidores OPC o añade un servidor, además se observan las propiedades de los servidores seleccionados automáticamente.

CUS Administrar Servidores OPC remotos: Este caso de uso se inicia cuando el usuario desea administrar los Servidores OPC remotos, como resultado la aplicación muestra la lista de los servidores o añade un servidor, además se observan las propiedades de los seleccionados automáticamente.

CUS Operar Servidores OPC: El caso de uso se inicia cuando el usuario desea operar los Servidores OPC, como resultado se puede conectar o desconectar los Servidores OPC e inmediatamente se pueden observar las propiedades del servidor seleccionado en la parte derecha de la aplicación, también se pueden eliminar los Servidores OPC deseados.

CUS Gestionar Grupos: El caso de uso se inicia cuando el usuario desea gestionar los grupos, como resultado se muestran los grupos añadidos, se puede activar o desactivar los grupos, e inmediatamente se puede observar las propiedades del grupo seleccionado en la parte derecha de la aplicación, también se eliminan los grupos deseados.

CUS Operar Ítems: El caso de uso se inicia cuando el usuario desea gestionar los ítems, como resultado se muestran los ítems añadidos, se puede eliminar los ítems e inmediatamente se puede observar las propiedades de los ítems añadidos en la parte del medio de la aplicación.

CUS Salvar: El caso de uso se inicia cuando el usuario necesita salvar la configuración guardada.

CUS Cargar: El caso de uso se inicia cuando el usuario necesita cargar la configuración guardada.

2.4. Modelo de diseño

Un modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requerimientos funcionales y no funcionales junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema. Este artefacto constituye la entrada fundamental utilizada para el correcto desarrollo de las entradas de implementación.

Descomposición en paquetes

Los paquetes del diseño constituyen una herramienta organizacional, utilizados en la solución para dividir el modelo de diseño en piezas más pequeñas y puedan ser de mejor entendimiento para el desarrollo de la arquitectura que se propone.

La estructura del Explorador Cliente OPC DA está estructurada por paquetes. A continuación se ilustra la relación existente entre dichos paquetes:

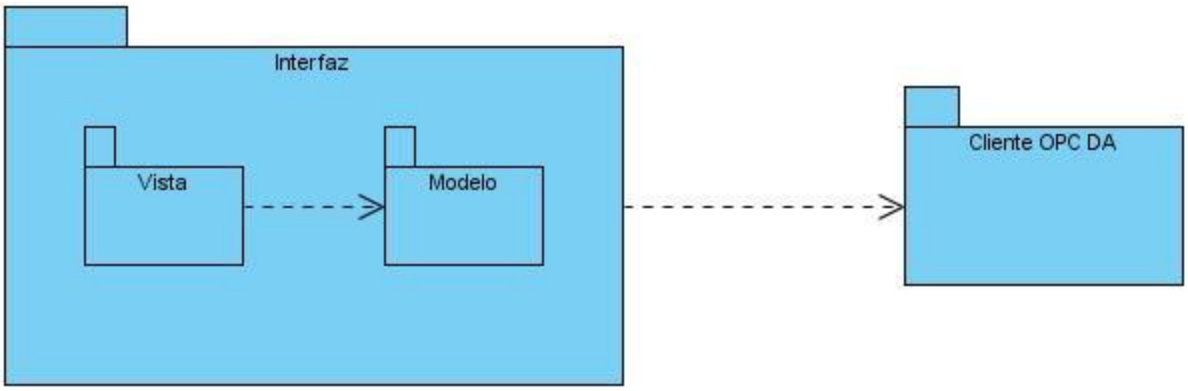


Figura 9: Diagrama de paquetes

Descripción y diseño de los paquetes

A continuación se detallan los paquetes definidos en la **Figura 9** que forman parte de la propuesta de solución:

- **Interfaz:** paquete donde se encuentra el entorno de desarrollo sobre el cual debe estar diseñada la interfaz gráfica de la aplicación. Este paquete se divide en dos partes:
  - **Vista:** agrupa las clases que implementan la lógica de dicho paquete encargadas de mostrar la interfaz gráfica con la que va a interactuar el usuario.
  - **Modelo:** contiene las clases que implementan la lógica que rige al Explorador Cliente OPC DA.

- **Cliente OPC DA:** paquete encargado de recolectar la información de los diferentes dispositivos para luego ser mostrada por el paquete interfaz.

Dentro del paquete Interfaz se emplea el Modelo/Vista que propone Qt, el cual divide las funcionalidades mediante la vista y el modelo, donde los cambios realizados en el último se actualizan automáticamente en la vista, sin necesidad de hacer cambios en ella, ofreciendo a los desarrolladores una mayor flexibilidad para personalizar la presentación de las partidas, y proporciona una interfaz de modelo estándar para permitir una amplia gama de fuentes de datos [18].

Seguidamente se ilustran los diagramas de clases por paquetes:

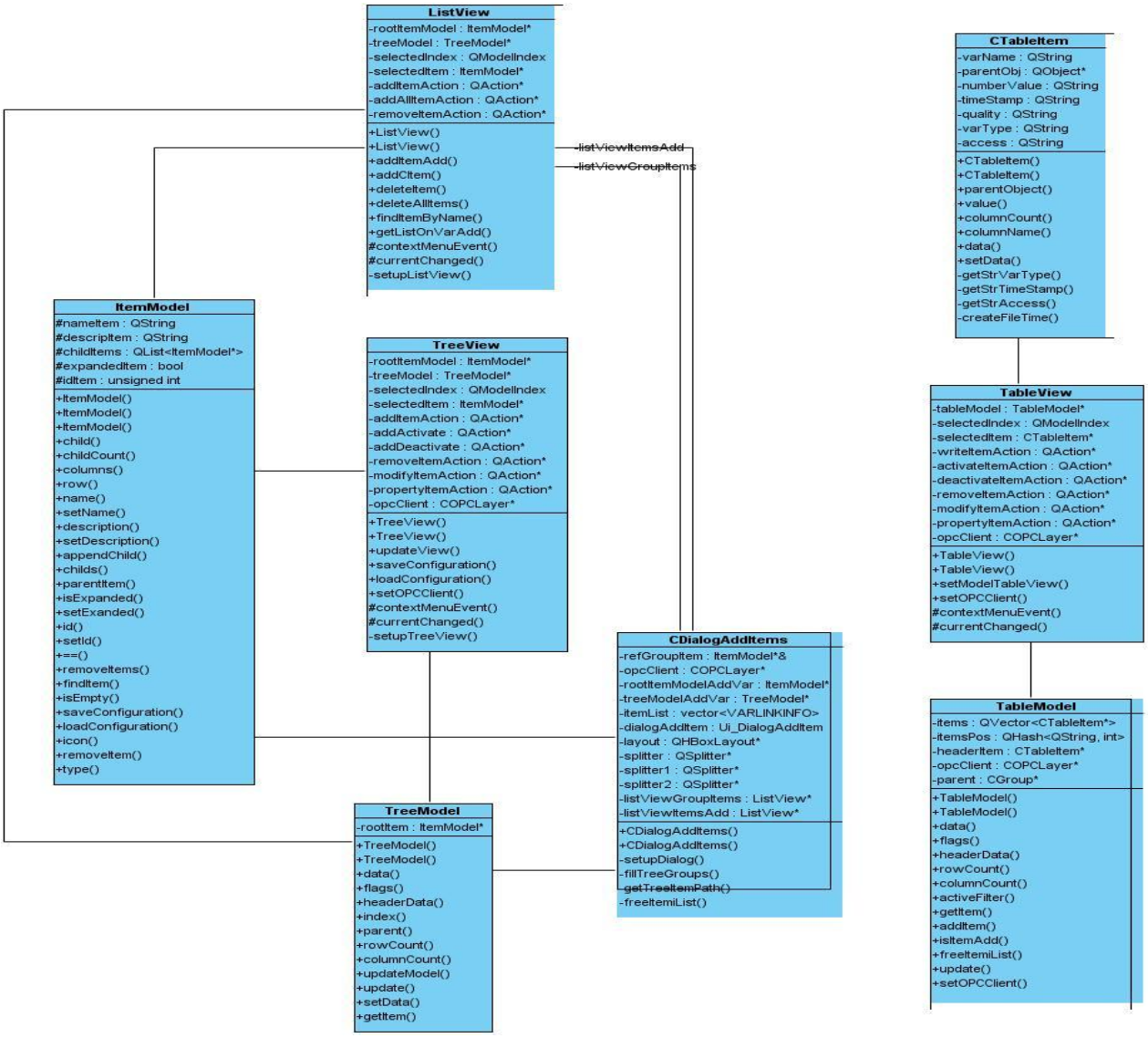


Figura 10: Diagrama de clase del paquete vista

En la **Figura 10** se ilustra el diagrama de clases correspondiente al paquete vista (TreeView, TableView, ListView y CDialogAddItems), igualmente aparecen las clases del paquete modelo que se relacionan con él.

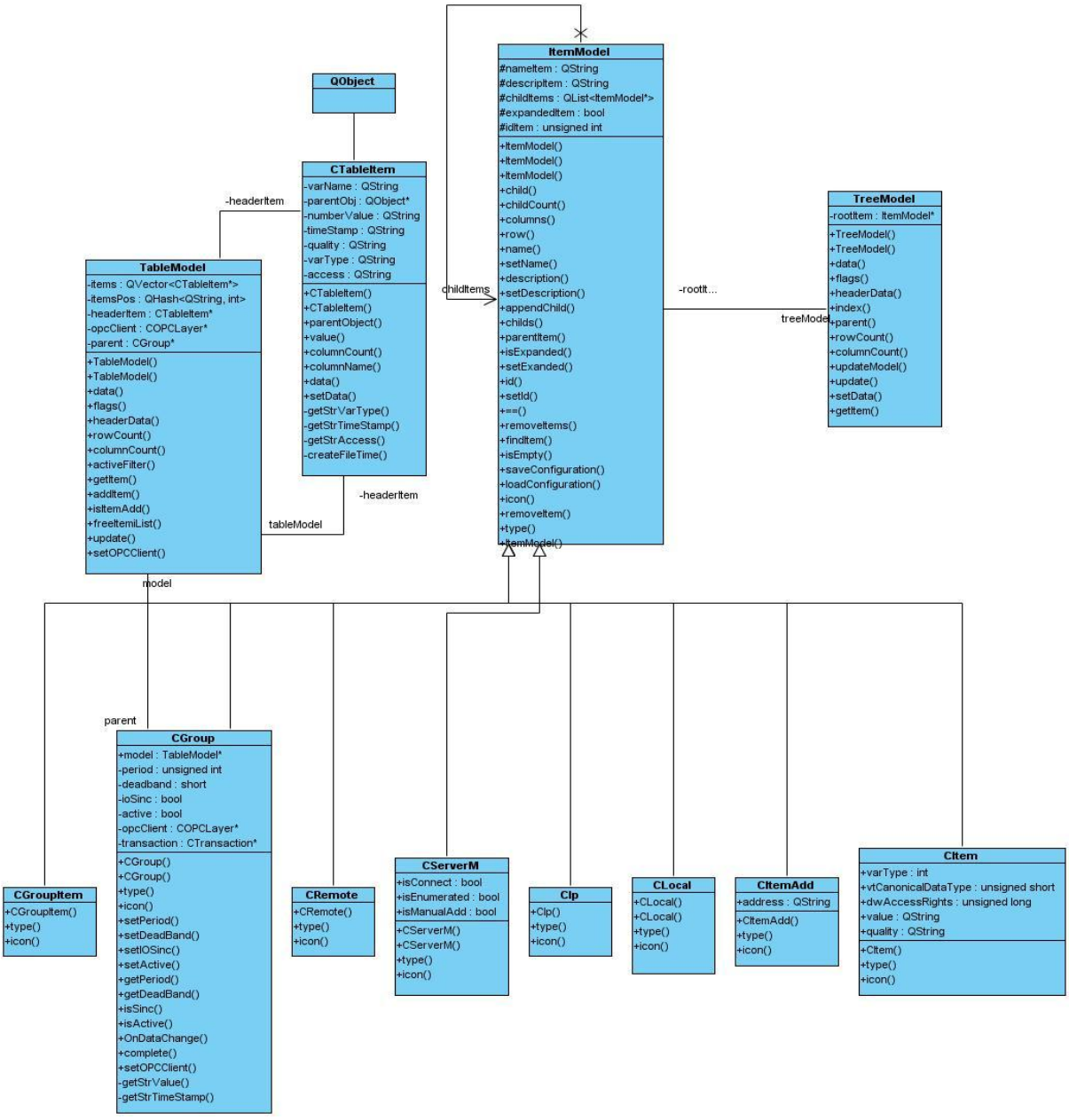


Figura 11: Diagrama de clase del paquete modelo

Breve descripción de las clases ilustradas en las Figuras 10 y 11:

- **TreeView:** se encarga de visualizar el árbol de configuración, es decir, la aplicación se va configurando a partir de los nodos hijos.
- **TableView:** clase encargada de visualizar diversos valores que contiene un *ítem*.
- **ListView:** ofrece la visualización de las listas de variables a adicionar en un grupo.
- **CDialogAddítems:** representa la ventana de diálogo para adicionar una variable.
- **TreeModel:** define el modelo por el cual se representan los elementos configurados en forma de árbol.
- **TableModel:** representa al modelo de datos de la tabla donde se visualizan los valores de los *ítems*.
- **ÍtemModel:** elemento para el modelo de árbol.
- **TableÍtem:** permite tipificar los campos de las columnas de captura y su índice correspondiente.
- **Cserver:** elemento para el modelo del árbol, que representa un Servidor OPC.
- **Cremote:** elemento para el modelo del árbol, que representa los Servidores OPC remotos.
- **Clocal:** elemento para el modelo del árbol, que representa los Servidores OPC locales.
- **Cip:** elemento para el modelo del árbol, que representa una dirección IP.
- **CGroupItem:** elemento para el modelo del árbol, que se utiliza al adicionar un grupo, representa un grupo OPC.
- **Cgroup:** elemento para el modelo del árbol, que representa un grupo OPC.
- **CItemAdd:** elemento para el modelo del árbol, que se utiliza al adicionar un *ítem*, representa un ítem OPC.

- **CItem:** elemento para el modelo del árbol, que representa un *ítem* o tag de un Servidor OPC.

Conclusiones parciales

El desarrollo de este capítulo ha permitido diseñar el sistema que se desea construir, así como las restricciones que deben existir para satisfacer las necesidades de los clientes. Se especificaron todos los requerimientos funcionales y no funcionales del sistema, los actores que intervienen en los casos de uso del sistema, éstos fueron descritos de forma detallada reflejando las funcionalidades recogidas previamente en los requerimientos.

Capítulo 3: Implementación y Pruebas.

Introducción

El presente capítulo cubre dos flujos de trabajo: implementación y pruebas. El resultado principal de la implementación es la obtención de componentes, dentro de los que se incluyen ficheros, ejecutables y sus dependencias. Además, este flujo especifica cómo van a estar desplegadas físicamente las distintas partes del sistema y mediante qué protocolos se comunicarán. También se conocerán los distintos tipos de pruebas, específicamente las que se le realizarán al Explorar Cliente OPC DA.

3.1. Implementación

La implementación es el centro de las iteraciones durante la fase de construcción, aunque también se lleva a cabo la implementación durante la fase de elaboración (para crear la línea base de la arquitectura) y durante la fase de transición (para tratar los defectos tardíos como los encontrados en su distribución).

Estándar de codificación

El estándar de codificación, utilizado para el desarrollo de este producto de software, permite establecer una serie de reglas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. El Explorador Cliente OPC DA está desarrollado completamente en inglés ya que es el idioma más utilizado internacionalmente para la construcción de componentes de software.

A continuación se describen algunos de los elementos fundamentales que componen el estándar utilizado:

- **Nombres**
  - Los nombres de las clases son sustantivos singulares.
  - Los nombres deben reflejar el qué y no el cómo.
  - Los nombres no deben revelar detalles de implantación.

- Escoger nombres lo suficientemente largos para ser expresivos, pero evitando manejar nombres que dificulten la labor de implantación.
- Evitar nombres que permitan una interpretación subjetiva (evitar ambigüedad y asegurar abstracción).
- Evitar redundancia no repitiendo nombre de clases en sus elementos.
- El nombre de los métodos es la concatenación de dos palabras, se usa minúscula para el nombre que representa la funcionalidad y mayúscula para el que representa el nombre del método.
- Minimizar el uso de abreviaciones, en caso de ser requeridas, debe ser consistente en su uso y cada abreviación debe significar sólo una cosa. En general, agregar a la documentación las abreviaturas.
- Los nombres de los atributos y parámetros son frases con sustantivos.
- Evitar la reutilización de nombres para distintos propósitos.

➢ **Manejo de errores**

- Se pueden manejar los errores mediante mecanismos de excepciones o mediante valores de retorno, aunque deben ser uniforme dentro de un mismo objeto.
- Es buena práctica emplear herramientas para identificar errores en la codificación en el momento preciso.

➢ **Codificación**

- Alinear secciones del código.
- Alinear verticalmente llaves de apertura y cierre.

- Emplear líneas en blanco para organizar el código, permitiendo crear párrafos de código para una mejor lectura.
- Evitar colocar más de una sentencia por línea.
- Minimizar el alcance de las variables para evitar confusión y facilitar el mantenimiento.
- Evitar el uso de variables públicas, sustituirlas por variables privadas y métodos que provean el valor de tal variable, para mantener el encapsulamiento.
- Emplear i, j, k, l, p, q, r para contadores en ciclos.
- Emplear al máximo operadores del tipo: +=, *=, /=, -=, ++, --, etc.
- Inicializar todas las variables.
- Emplear líneas en blanco para separar pasos lógicos (declaraciones, lazos, etc.).

Estándar de documentación

El estándar de documentación tiene como prioridad homogenizar toda la documentación entregada, para así poder lograr una fácil identificación y orden a la hora de revisar. Mantiene una integridad entre los grupos de trabajo y un orden a la hora de documentar las tareas o actividades a desarrollar.

- **Estilo de bloques de documentación JavaDoc**

Se adopta el estilo de bloques de documentación JavaDoc, el cual consiste de un bloque de comentario de estilo C, este bloque de comentario comienza con dos asteriscos de esta manera:

```
/**
 * Texto
 */
```

En este caso los asteriscos que se encuentran en la línea de la mitad son opcionales, por lo que también es válido que el bloque sea de esta manera:

```
/**
 Texto
 */
```

- **Descripción Breve con comando \brief ó @brief**

Para hacer una descripción breve se adopta el uso del comando \brief ó @brief en el bloque de comentarios ya descrito.

La acción de este comando termina al final de un párrafo, de tal manera que la descripción detallada sigue después de una línea vacía.

Aquí tenemos un ejemplo:

```
/**
 * @brief descripción breve.
 * Continuación de la descripción breve.
 * La descripción detallada comienza aquí, nótese
 * que se debe dejar una línea en blanco para lograr
 * tener las dos descripciones (breve y detallada)
 */
```

A la hora de hacer una descripción de los argumentos de métodos y funciones, se hará en línea, es decir, luego de la declaración de cada uno de los métodos se da una breve descripción de cada uno de ellos, en el siguiente ejemplo se muestra el formato a utilizar:

int multFunction (int c , /**<descripción del operando */ o

/**

* @brief Breve descripción del método

* @author Nombre del autor

* @date fecha

Comandos @author y @date

Es importante que se especifique el nombre del autor y la fecha de creación de cualquier estructura en un código, para ello se utilizan los comandos @author y @date para el nombre del autor y la fecha respectivamente, en el siguiente ejemplo se puede ver la acción de estos comandos:

/**@brief Descripción breve

* Aquí comienza la descripción detallada

* @author Edgar Acosta dragoicaru@hotmail.com

* @date 08-08-2007

*/

3.2. Diagrama de componentes

El diagrama de componente describe cómo los elementos del modelo de diseño serán implementados en términos de componentes, describiendo además, su organización de acuerdo con los mecanismos de

estructuración y modularización disponibles en el entorno de implementación y el lenguaje de programación utilizado, así como la dependencia que se establece entre dichos elementos.

Algunos estereotipos estándar de componentes son:

- **<<executable>>:** es un programa que puede ser ejecutado en un nodo.
- **<<file>>:** es un fichero que contiene código fuente a datos.
- **<<library>>:** es una biblioteca estática o dinámica.
- **<<table>>:** es una tabla de la base de datos.
- **<<document>>:** es un documento.

A continuación serán expuestos los diagramas de componentes:

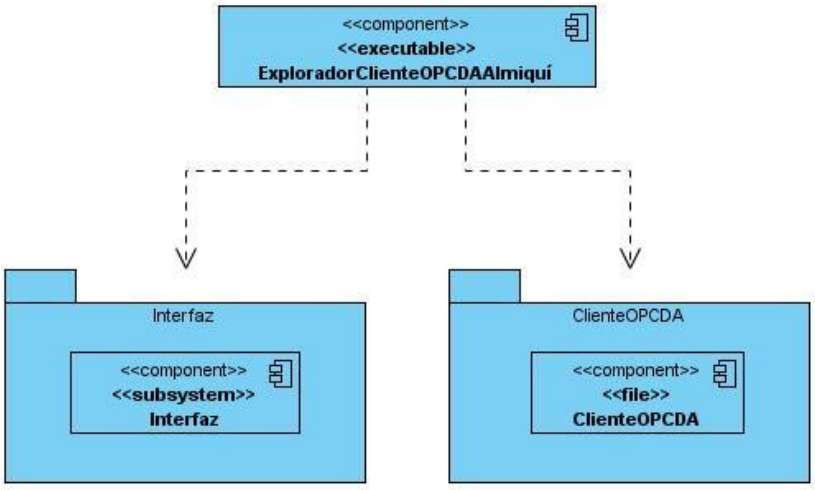


Figura 12: Diagrama de componentes

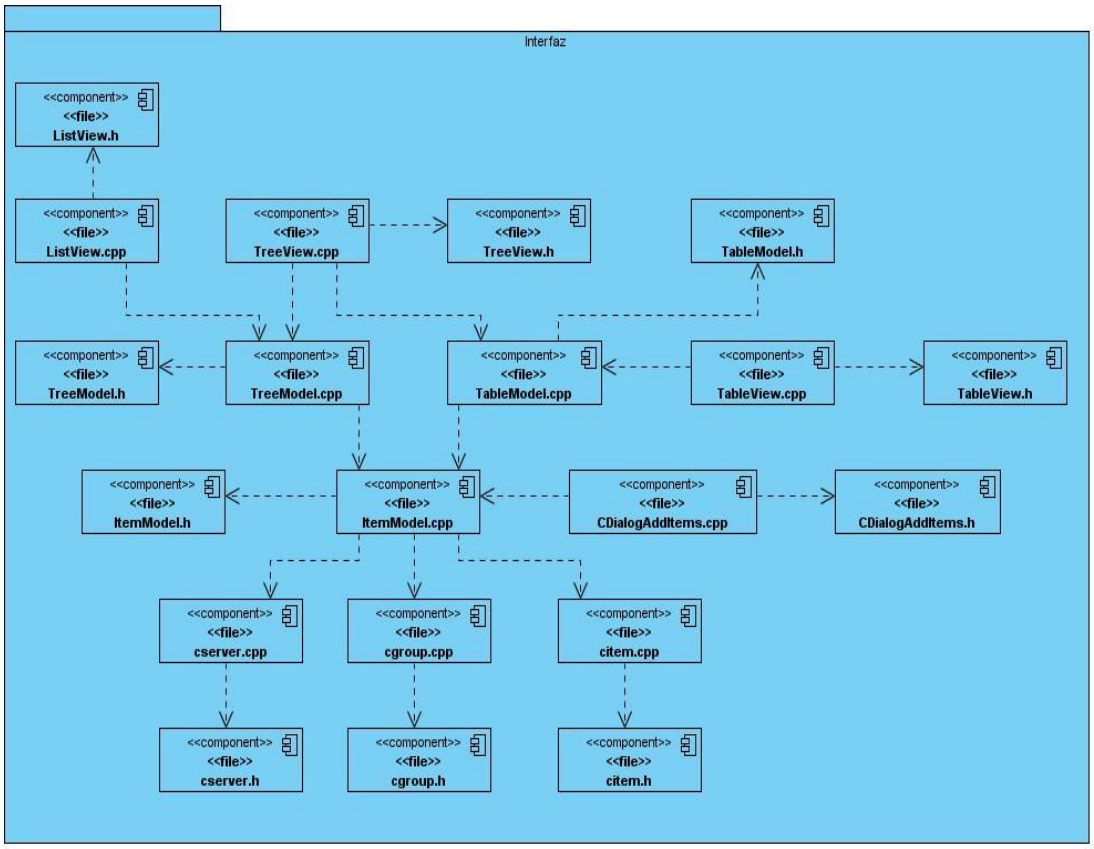


Figura 13: Diagrama del componente Interfaz

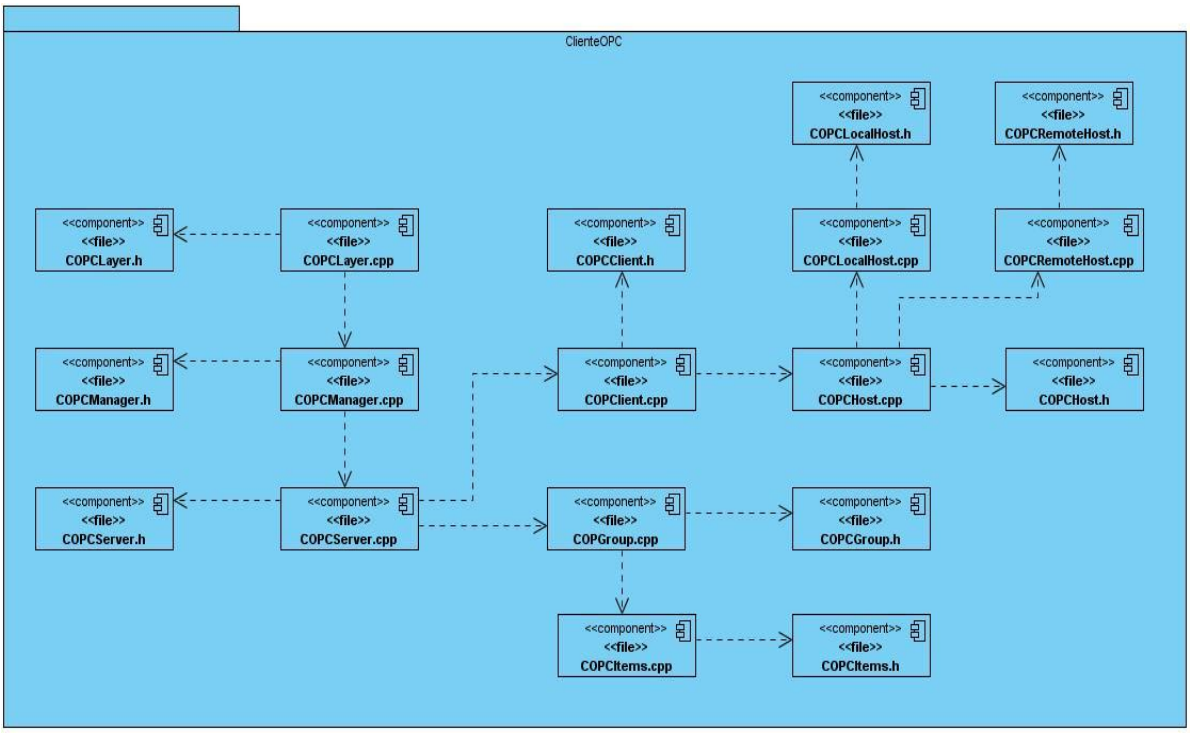


Figura 14: Diagrama del componente Cliente OPC DA

El componente Explorador Cliente OPC DA es el ejecutable que se utilizará para una mejor distribución de la aplicación, el cual necesita de cada uno de los archivos, en los cuales se encuentra toda la lógica de las funcionalidades implementadas en el desarrollo de la aplicación, dividida en los paquetes anteriormente especificados.

3.3. Diagrama de despliegue

El diagrama de despliegue muestra la configuración de los nodos (procesadores y dispositivos) que participan en la ejecución de los componentes que residen en los mismos. Mostrando mediante los protocolos de comunicación, cómo colabora un nodo con otro.

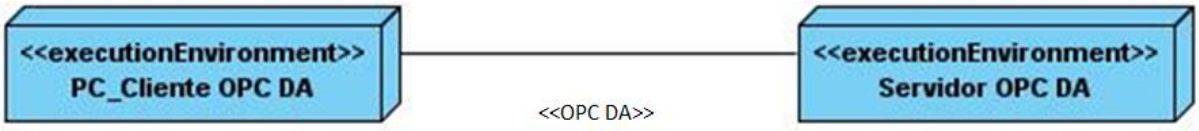


Figura 15: Diagrama de despliegue

A continuación se brinda una breve descripción de los nodos presentes en la figura anterior:

- **PC_Cliente OPC DA**: representa el Cliente OPC DA desarrollado e instalado en una computadora que se utilizará para interactuar datos en tiempo real con el Servidor OPC DA.
- **Servidor OPC DA**: representa la estación donde estará instalado el Servidor OPC DA, éste se encarga de almacenar los datos de los diferentes dispositivos conectados a él.
- **<<OPC DA>>:** representa la especificación del estándar de comunicación que utiliza la tecnología DCOM.

3.4. Prueba de software

Bajo el nombre de pruebas de software se agrupan un conjunto de prácticas correctivas (frente a las prácticas preventivas que se aplican durante el proceso de construcción de software) cuyo objetivo es determinar la calidad de los sistemas de software. La prueba es el proceso de ejecución de un programa con la intención de descubrir un error [20].

Según lo anteriormente planteado las pruebas de software son la actividad más común de control de calidad realizada en los proyectos de desarrollo o mantenimiento de aplicaciones y sistemas. El éxito de una prueba está dado por la cantidad de errores que le sean detectados al software. **3.4.1. Objetivos de las pruebas**

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Las de integración son necesarias para cada construcción dentro de la iteración, mientras que las de sistemas sólo son necesarias al final de la iteración.

- Diseñar e implementar las pruebas cuando los casos de pruebas que especifican qué probar, creando los procedimientos de pruebas que especifican cómo realizar las pruebas y creando si es posible, componentes de pruebas ejecutables para automatizar las pruebas.
- Realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados.

Según Pressman: “Las pruebas deberían empezar por lo pequeño y progresar hacia lo grande” [21].

3.4.2. Tipos de pruebas de software

Las pruebas de unidad que aíslan cada parte del programa y muestran que las partes individuales son correctas, las cuales no descubrirán todos los errores del código, ni errores de integración, ni de rendimiento, ni otros problemas que afectan a todo el sistema en su conjunto. Están divididas en dos grupos: pruebas de caja blanca y pruebas de caja negra.

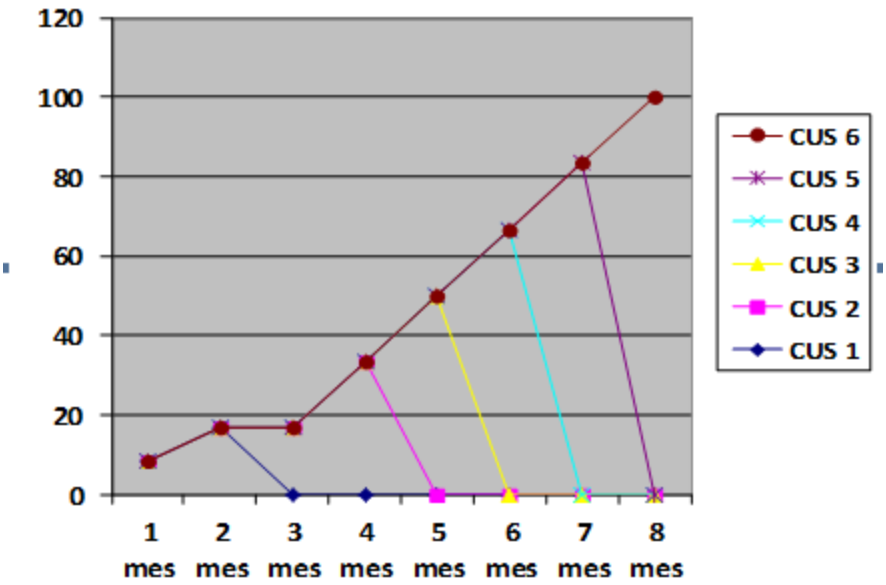
Para realizarle las pruebas al Explorador Cliente OPC DA se utilizaron las pruebas de caja negra pues son las indicadas para los módulos que van a ser interfaz de usuario, se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. A estas pruebas se le denominan pruebas funcionales, pues el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de la estructura del código del sistema.

Al Explorador Cliente OPC DA se le realizaron los casos de pruebas para los cuales se diseñaron los mismo, ver **Anexo 1**. Los resultados que arrojaron los casos de pruebas fueron satisfactorios demostrando que funcionalmente la aplicación cumple con los objetivos propuestos.

Para demostrar el avance que se tuvo en la realización del sistema se ilustra mediante la **Tabla 2** el cumplimento de las funcionalidades durante los meses de desarrollo comprendidos entre octubre – junio, además se muestra el por ciento de avance en dicho período.

Tiempo / meses	Funcionalidades realizadas	% de avance
2	CUS 1	14.29
1	CUS 2	28.57
1	CUS 3	42.86
1	CUS 4	57.14
1	CUS 5	71.43
1	CUS 6	85.71
1	CUS 7	100

Tabla 2: Avance de las funcionalidades de la aplicación



Gráfica 1: Avance de las funcionalidades de la aplicación

Conclusiones parciales

En este capítulo se realizaron los diagramas de componentes correspondientes al flujo de trabajo Implementación, estos modelos especifican la estructura de los componentes implementados para la aplicación desarrollada. Además, se le realizaron pruebas de caja negra a la interfaz, las cuales arrojaron resultados satisfactorios.

Conclusiones

- Se implementó un Explorador Cliente OPC DA para el Sistema Operativo Windows, brindando la posibilidad que cualquier Servidor OPC DA se conecte al explorador para que sus datos sean visualizados a través de éste, permitiendo al usuario interactuar con dichos datos.

- El Explorador Cliente OPC DA sirve como herramienta de prueba al Servidor OPC que se está desarrollando en el CEDIN.

Recomendaciones

Se recomienda:

- Confeccionar la ayuda y el manual de usuario del sistema.
- Realizar la funcionalidad enumerar Servidores OPC DA remotos.

Bibliografía referenciada

[1] **Progea Corporation,** Progea Corp Web Master Page, [En Línea] [Citado el: 2 de octubre del 2011], URL: www.progea.com.

[2] **Progea Group,** Movicon SCADA Web Page, [En Línea] [Citado el: 3 de octubre del 2011], URL: www.movicon.com.

[3] **National Instruments,** LabView HMI/SCADA Web Page, [En Línea] [Citado el: 3 de octubre del 2011], URL: http://www.ni.com/labview/.

[4] **National Instruments,** OPC National Instrument Resources, [En Línea] [Citado el: 3 de octubre del 2011], URL: http://www.ni.com/opc/.

[5] **OMRON Corporation,** OMRON Web Master Page, [En Línea] [Citado el: 3 de octubre del 2011], URL: www.omron.com.

[6] **Wincc SIEMENS Group,** WINCC Web Page, [En Línea] [Citado el: 2 de octubre del 2011], URL: www.siemens.com\wincc.

[7] **Ing. Yaneisy Hernández Hernández**. Desarrollo de herramientas para Servidores OPC de OPC. Universidad Central "Marta Abreu" de las Villas. Santa Clara, Enero 2004.

[8] **Kamen Edward W.,** Introduction to Industrial Controls and Manufacturing. Junio 15, 1999.

[9] **OPC Foundation**, OPC Foundation Web Master Page, [En Línea] [Citado el: 8 de octubre del 2011], URL: www.opcfoundation.org.

[10] **Matrikon Inc**, OPC Matrikon Web Page, [En Línea] [Citado el: 14 de octubre del 2011], URL: http://www.matrikonopc.com/products/opc-desktop-tools/opc-hda-explorer.aspx, 2003.

[11] **Matrikon Inc**, OPC Matrikon Web Page, [En Línea] [Citado el: 14 de octubre del 2011], URL: http://www.matrikonopc.com/products/opc-archiving/opc-client-for-crystal-reports.aspx, 2003

[12] **Kepware Technologies Inc,** Kepware Technologies Web Page, [En Línea] [Citado el: 14 de octubre del 2011], URL: http://www.kepware.com/Products/feature-quickclient.asp.

[13] **Brockschmidt K.,** Inside OLE, Second Edition, Microsoft Press, Redmond, WA, 1995.

[14] **Microsoft Corp,** Microsoft COM Specification, version 0.9, Octubre 24, 1995.

[15] **Matrikon Inc,** OPC Matrikon Web Page, [En Línea] [Citado el: 13 de octubre del 2011], URL: www.matrikon.com/opc.

[16] **Systems, Zator.** Programación C++. [En línea] [Citado el: 20 de noviembre del 2011], http://www.zator.com/Cpp/E1.htm.

[17] **James Rumbaugh, Ivar Jacobson, Grady Booch.** El Lenguaje Unificado de Modelado. Addison Wesley, 2000.

[18] Introducción a C++ y a la resolución de problemas. Introducción a C++ y a la resolución de problemas. [En línea] Febrero de 2008. [Citado el: 12 de Febrero de 2008.] http://www.usabilidadweb.com.ar/.

[19] **Nokia Corporation.** Nokia Corporation Web Page. [En línea] Febrero de 2008. [Citado el: 6 de mayo del 2011.], URL: http://doc.qt.nokia.com/latest/model-view-programming.html.

[20] **Mtra. María de Lourdes Santiago Zaragoza**. Desarrollando aplicaciones informáticas con el Proceso de Desarrollo Unificado RUP). [En Línea] [Citado: 22 de noviembre del 2011], URL: http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm.

[21] **Pressman, Roger S.** Ingeniería de Software. Un enfoque práctico. Parte 1. La Habana: Felix Varela, 2005.

Bibliografía consultada

1. **Gómez Karel, González Leonardo, Arencibia Annia**. Centro de Control para el Sistema de Información para la Salud. Trabajo de Diploma para optar por el título de Ingeniero Informático. Universidad de las Ciencias Informáticas. La Habana, Cuba, junio de 2007.

2. **Gregory, B,** Applying COM+, Octubre, 2000.

3. **Ing. Adrian Carlos Moreno Borges, Ing. Yordanis Bridon Danger.** Desarrollo de una Interfaz Asíncrona utilizando el Protocolo Ethernet/IP. Universidad de las Ciencias Informáticas. Cuidad de la Habana.

4. **Ing. Antonio Cedeño Pozo, Ing. Luis Enrique García Hernández, Yosep Yasmany Pérez Pérez.** Framework para el desarrollo de manejadores de dispositivos para sistemas SCADA. Universidad de las Ciencias Informáticas. Cuidad de la Habana. Septiembre 2009.

5. **Ing. Catherine Aspée Barrios, Ing. Luis Enrique García Hernández, Yosep Yasmany Perez Perez.** Sistema SCADA GALBA Manual de Usuario Recolector Gráfico. Universidad de las Ciencias Informáticas. Ciudad de La Habana. Enero 2010.

6. **Ing. Catherine Aspée Barrios, Ing. Luis Enrique García Hernández.** Sistema SCADA GALBA Documento Entrega Formal del Recolector Gráfico. Universidad de las Ciencias Informáticas. Ciudad de La Habana. Enero 2010.

7. **Ing. Evian Suárez Rodríguez, Ing. Luis Enrique García Hernández.** Desarrollo SCADA Nacional - Guía de Implementación de la “Pasarela OPC DA”. Universidad de las Ciencias Informáticas. Ciudad de La Habana. Febrero 2010.

8. **Ing. Evian Suárez Rodríguez, Ing. Luis Enrique García Hernández.** Desarrollo SCADA Nacional - Guía de Implementación de la Manejador OPC. Universidad de las Ciencias Informáticas. Ciudad de La Habana. Febrero 2010.

9. **Ing. Evián Suárez Rodríguez, Ing. Yolier Galan Tasse.** Desarrollo SCADA Nacional - Guía de configuración de DCOM para OPC. Ciudad de La Habana. Octubre 2008.

10. **Ing. Luis Enrique García Hernández, Ing. Rubén Gómez Johnson, Ing. Adrián Carlos Moreno Borges.** Manejadores GE Fanuc y BSAP Serial para el proyecto SCADA PDVSA. Universidad de las Ciencias Informáticas. Cuidad de la Habana. Enero 2009.

11. **Ing. Luis Enrique García Hernández, Yosep Yasmany Perez Perez.** Sistema SCADA GALBA Diseño del Producto Recolector Gráfico. Universidad de las Ciencias Informáticas. Ciudad de La Habana. Junio 2009.

12. **Ing. Yunier Velázquez Batista, Ing. Irina Elena Argota Vega, Ing. Evian Suárez Rodríguez, Ing. Luis Enrique García Hernández.** Desarrollo SCADA Nacional - Documentación Arquitectura de Software Gateway OPC DA**.** Universidad de las Ciencias Informáticas. Ciudad de La Habana. Febrero 2010.

13. **Ing. Yunier Velázquez Batista.** Desarrollo SCADA Nacional – Manual de Usuario "Gateway OPC". Universidad de las Ciencias Informáticas. Ciudad de La Habana. Mayo 2008.

14. **James Rumbaugh, Ivar Jacobson, Grady Booch**.El Proceso Unificado de Desarrollo de Software. La Habana. Cuba. Editorial Félix Varela. 2004. Pág. 4, 5, 6 y 7.

15. **Kruchten, P.** 2000. The Rational Unified Process: An Introduction. 2000.

16. **M.Sc. René González Rodríguez, M.Sc. Moisés Herrera.** Documentación de diseño de la integración del SCADA al estándar OPC DA. Universidad de las Ciencias Informáticas. Cuidad de la Habana. Diciembre 2006.

17. **Modelo de Dominio.** [En Línea] [Citado el: 25 de noviembre del 2011], URL: http://iie.fing.edu.uy/ense/asign/desasoft/practico/hoja8/ejemplos_clase2.pdf.

18. **National Instruments,** LabView HMI/SCADA Web Page, [En Línea] [Citado el: 3 de octubre del 2011], URL: http://www.ni.com/labview/.

19. **National Instruments,** OPC National Instrument Resources, [En Línea] [Citado el: 3 de octubre del 2011], URL: http://www.ni.com/opc/.

20. **Neylith Quintero, Catherine Aspée Barrios, Ing. Evián Suarez Rodriguez, Ing. Luis Enrique García Hernández.** Desarrollo SCADA Nacional - Documentación Entrega Formal del Módulo: Gateway OPC**.** Universidad de las Ciencias Informáticas. Ciudad de La Habana. Febrero 2008.

21. **OPC Foundation,** OPC Batch Automation Specification, Version 1.0, Julio 19, 2001.

22. **OPC Foundation,** OPC Batch Specification, Version 2.0, Julio 19, 2001.

23. **OPC Foundation,** OPC Data Exchange Specification, Version 1.0, Marzo 5, 2003.

24. **OPC Foundation**, OPC Foundation Web Master Page, [En Línea] [Citado el: 8 de octubre del 2011], URL: www.opcfoundation.org.

25. **OPC Foundation**, OPC Historical Data Access Automation Specification, Version 1.0, Enero 26, 2001.

26. **OPC Foundation**, OPC Historical Data Access Specification, Version 1.10, Enero 26, 2001.

27. **OPC Foundation,** OPC Security Specification, Version 1.0, Octubre 17, 2000.

28. **Pressman, Roger S.** Ingeniería de Software. Un enfoque práctico. Parte 1. La Habana: Felix Varela, 2005.

29. **Progea Corporation,** Progea Corp Web Master Page, [En Línea] [Citado el: 2 de octubre del 2011], URL: www.progea.com.

30. **Progea Group,** Movicon SCADA Web Page, [En Línea] [Citado el: 3 de octubre del 2011], URL: www.movicon.com.

31. **Rafael Trujillo Codorníu, Antonio Cedeño Pozo, Luis E. García Hernández.** Interfaz Genérica para los Manejadores de Dispositivos en el Proyecto SCADA PDVSA.

32. **René Iván Rodríguez Infante, Wilsón Alfonso González**. Automatización de las pruebas de interfaz de usuario para componente HMI GTK del SCADA. Universidad de las Ciencias Informáticas. Ciudad de La Habana, Junio 2010.

33. **Sergio M Sastre Fernández.** Fundamentos del diseño y la programación orientada a objetos.

34. **SIEMENS Corporation**, SIEMENS Web Master Page, [En Línea] [Citado el: 6 de octubre del 2011], URL: www.siemens.com.

35. **Wikipedia**, Wikipedia Web Page, [En línea] [Citado el: 5 de febrero], http://es.wikipedia.org/wiki/Trama_de_red.

36. **Wincc SIEMENS Group,** WINCC Web Page, [En Línea] [Citado el: 2 de octubre del 2011], URL: www.siemens.com\wincc.

37. **Yosep Yasmany Perez Perez.** Sistema SCADA GALBA Patrones de Diseño del Recolector Gráfico. Universidad de las Ciencias Informáticas. Ciudad de La Habana. Junio 2009.

Glosario de términos

CASE: (*Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora)*, herramientas de ingeniería de software asistida por computadora.

Cliente OPC: Aplicación que solo utiliza los datos brindados por el Servidor OPC.

COM: (*Component Object Model)*, filosofía de programación de objetos orientados a Interfaces para compartir entre aplicaciones.

COM+: (*Component Object Model Plus),* mecanismo que evolucionó a COM, para lograr una transferencia de datos y manejos de eventos más eficientes, fue instalado en los sistemas operativos de Microsoft a partir del año 2000.

PLC: (*Controlador Lógico Programable),* dispositivos electrónicos usados en automatización industrial para realizar estrategias de control básicas. Por su robustez y características sencillas de control, están cercanos al proceso, permitiendo ejecutar las tareas básicas del control, aun cuando no tenga conexión a las capas superiores del control.

DCOM: (*Distributed Component Object Model)*, es una tecnología propietaria de Microsoft para desarrollar componentes software distribuidos sobre varios ordenadores y que se comunican entre sí.

Dispositivo: Se denomina dispositivo a un elemento de hardware que alberga información de proceso o estado de sí mismo, que forma parte de un sistema automatizado. Comúnmente los dispositivos son Controladores Lógicos Programables, Computadoras Industriales, Sistemas de Control Distribuidos, sensores o actuadores inteligentes con capacidad de comunicación.

Framework: En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

GPL: (*General Public License,* Licencia Pública General), es una licencia creada por la Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso de software.

IDE: (*Integrated Development Environment,* Entorno de Desarrollo Integrado), es un programa compuesto por un conjunto de herramientas para un programador.

Interfaz de Usuario: Medio que el usuario utiliza para comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo, normalmente suelen ser fáciles de entender y fáciles de accionar.

Multiplataforma: Término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

OLE: (*Object Linking and Embedding),* estándar desarrollado por Microsoft, que permite crear objetos en una aplicación y luego insertarlos en otra, compartiendo información entre ellas.

Protocolos de comunicación: son como reglas de comunicación que permiten el flujo de información entre computadoras o dispositivos diferentes que manejan lenguajes distintos.

Plug-and-play: se refiere a la capacidad de un sistema informático de configurar automáticamente los dispositivos al conectarlos. Permite poder enchufar un dispositivo y utilizarlo inmediatamente, sin preocuparte de la configuración.

Qt: Biblioteca multiplataforma para desarrollar interfaces gráficas de usuario.

Tag: Conjunto de caracteres que se añade a un elemento de los datos para identificarlo (Oxford English Dictionary).

XML: *(eXtensible Markup Language)*, lenguaje para el intercambio de datos complejos en Internet por ejemplo bases de datos.