Facultad 5

Título: Propuesta de servidor OPC UA para sistemas de automatización y control.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Lorgio Vargas Vento

Tutor(es): Ing. Lauren San Juan Guía.

Ing. Leitniz Pérez Buján

La Habana

2015

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informática Industrial de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Lorgio Vargas Vento

_____

Ing. Lauren San Juan Guía.

_____

Ing. Leitniz Pérez Buján

_____

Datos de contacto

Ing. Leitniz Pérez Buján (lbujan@uci.cu**)**

Graduado con el título de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas, Cuba. Tiene 2 años de graduado y 1 año de experiencia en el tema de la investigación.

Ing. Lauren San Juan Guía (lsanjuan@uci.cu**)**

Graduado con el título de Ingeniero en la Universidad de las Ciencias Informáticas, Cuba. Tiene 2 años de graduada y 2 años de experiencia en el tema de la investigación.

Dedicatoria

Quiero dedicar esta tesis y por ser los que soportan mis sueños a:

A mi padre y a mi madre por estar siempre que los he necesitado y por darme la educación que me ha llevado a ser la persona que soy.

A mis abuelitos que siempre me han brindado un amor interminable e incondicional. En especial a mi abuelito Lorgio que en paz descanse.

A mi hermanita de la que siempre he estado orgulloso.

Agradecimientos

- ✓ *A mi mamá por ser la mejor madre del mundo.*
- ✓ *A mi papá por siempre exigirme ser mejor.*
- ✓ *A mis abuelitos Lorgio, Oslidio, Sarah y Matilde, por consentirme siempre.*
- ✓ *A mi hermanita por ayudarme siempre que la necesito.*
- ✓ *A Oliver a pesar de que todavía no puede decir mi nombre.*
- ✓ *A Normando y a Tania por ser como un padre y una madre para mí.*
- ✓ *A mis tíos y tías, Ana Elena, Osly, Jorge y Gladis por todo su apoyo.*
- ✓ *A Mis primas Lenia, Liana, Sabrina y Carolina por ser como hermanas para mí.*
- ✓ *A Tony por hacer feliz a mi hermana.*
- ✓ *A mi mejor amigo Felipe, por toda su paciencia y ayuda.*
- ✓ *A mis amigos Rioger, Cuba, Erduin, Eliani, Luis Miguel, Ernesto Alfredo, Esmaykel, Yerson, Ernesto Leyva, Kiki, Yanet, Lara, Koko y todos los que se me puedan quedar que porque no los mencione no son menos importantes.*
- ✓ *A mis tutores por su apoyo y ayuda para convertirme finalmente en un Ingeniero.*
- ✓ *A mis brigadas, la de la Facultad 6 que a pesar de estar solo un año con ella, fue con la que di mis primeros pasos en la universidad y la de la Facultad 5 que sin discusión es la mejor de toda la universidad.*
- ✓ *A la Revolución por darme la oportunidad de realizar mis sueños.*

Resumen

El estándar de conectividad de datos OPC, surge como una solución abierta y flexible para problemas de integración entre diferentes aplicaciones de automatización y control. Este estándar provee una manera común de acceder a datos de cualquier fuente, ya sea dispositivos, bases de datos o sistemas terciarios. El Centro de Informática Industrial de la Universidad de las Ciencias Informáticas, ha venido desarrollando un conjunto de soluciones OPC basadas en la especificación OPC de Acceso a Datos. Uno de los componentes desarrollados es el servidor OPC DA, incorporado al sistema de Supervisión, Control y Adquisición de Datos (SCADA) "Guardián del ALBA", para el intercambio de datos entre los procesos que este automatiza. Pero, este está sujeto a las limitaciones del estándar, tales como la dependencia del sistema operativo Windows al utilizar la tecnología COM/DCOM[1] implicando que la seguridad dependa de este sistema operativo y la falta de enlaces entre las diferentes especificaciones. Esta problemática lleva a que sea necesario proponer una solución estándar, que exista interoperabilidad integrada entre sus especificaciones y no dependa de la plataforma Microsoft.

Para dar solución a dicho problema se realiza el siguiente trabajo de diploma que tiene como objetivo principal desarrollar una propuesta de servidor OPC UA para los sistemas de automatización y control. La solución está compuesta por un sistema basado en las especificaciones de un servidor OPC UA con la capacidad de intercambiar datos con cualquier equipo que cumpla el estándar OPC UA. Obteniéndose como resultado una primera propuesta de un servidor OPC UA para la línea de Comunicaciones del Centro de Informática Industrial.

Palabras clave

Automatización, comunicación, estándar, protocolo.

[1] Tecnología propietaria de Microsoft para desarrollar componentes software distribuidos sobre varios ordenadores y que se comunican entre sí.

Abstract

The standard of data connectivity OPC, was created as an open and flexible solution for integration problems, between automation and control applications. This standard provides a common way to access to data of any source, either dispositive, data bases or third-party systems. The Industrial Informatics Center of the University of the Informatics Sciences has been developing a set of OPC solutions based in Data Access OPC specification. One of the components developed is OPC DA Server, incorporated to Supervision, Control and Data Acquisition System (SCADA) “Guardián del Alba”, for the interchange of data among the processes that this automates. But, it is subject to the limitations of the standard as the dependence of the Windows operating system using COM/DCOM technology, this also means that security depends on this operating system and the lack of links between the different specifications. For that reason, is necessary to propose a solution that be standard, provides interoperability between the specifications and do not depends of Microsoft platform.

In order to give a solution for this problem, the following research has as main objective to develop an OPC UA server for the automation and control systems, on a freeware platform. The solution is compose for a system based on the OPC UA server specifications with the capacity of exchanging data with any devices on OPC UA standard. As the main result, we obtaining the first OPC UA server for the Communication Line of the Industrial Informatics Center.

Keywords:

Automation, communication, protocol, standard.

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

La nueva era de la automatización industrial trajo para los usuarios los beneficios de sistemas de control integrados como una sola pieza, donde no tendrían que depender del software y hardware propietarios. Los avances, cada vez más acelerados, en tecnología de software prometen desaparecer las múltiples dificultades de integración impuestas por los sistemas propietarios, dando lugar a dispositivos y sistemas compatibles.

El estándar de conectividad de datos OPC (OLE[2] para el control de procesos), surge como una solución abierta y flexible que permite solucionar los problemas de integración entre diferentes aplicaciones de automatización y control. Este estándar provee una manera común de acceder a datos de cualquier fuente, ya sea dispositivos, bases de datos o sistemas terciarios. De esta manera, es posible intercambiar datos con cualquier equipo que cumpla el estándar OPC y se reducen los costes de desarrollar o comprar drives.

Pero, las primeras especificaciones de la Fundación OPC[3] - lo que conocemos hoy como OPC clásico - no están conectadas, no existe enlace entre elementos como la lectura de un valor de acceso a datos con su lectura de datos históricos o eventos basados en estas lecturas. Los estándares de OPC clásico, están basados en COM/DCOM de Microsoft, tecnología que presenta dificultades de configuración que afectan considerablemente el coste en tiempo y las facilidades de uso. La aplicación de esta tecnología implica que la seguridad dependa del sistema operativo Windows. Además, la presión y exigencia de usuarios finales y desarrolladores de software, que buscaban la portabilidad del estándar OPC sobre el sistema operativo Linux y otras plataformas sin tecnología Microsoft, se convirtió en un elemento de principal importancia para la Fundación.

Con estos requisitos y características se desarrolló una nueva definición del estándar

[2] **Object Linking and Embedding**: "incrustación y enlazado de objetos", es el nombre de un sistema de objetos distribuido y un protocolo desarrollado por Microsoft.

[3] Consorcio industrial que crea y mantiene los estándares de conectividad abierta de dispositivos y sistemas de automatización industrial, como los sistemas de control industrial y control del proceso en general.

OPC UA (Open Communication Unified Architecture, en inglés), diseñada para ser independiente de la plataforma de desarrollo, enfatizando en los servicios Web y el uso de SOA. Esta innovación permitió poder desarrollar sistemas de control distribuidos, integrados en diversas plataformas. OPC UA provee todos los datos en su espacio de direcciones unificado; datos actuales, datos históricos y eventos son relacionados unos con otros, atendiendo directamente el problema de interoperabilidad. Además, propone un modelo orientado a objetos que permite manejar datos estructurales y complejos.

El Centro de Informática Industrial (CEDIN) perteneciente a la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI), cuya misión es desarrollar productos y servicios informáticos de automatización industrial, ha venido desarrollando un conjunto de soluciones OPC basadas en la especificación OPC de Acceso a Datos (DA). Uno de los componentes desarrollados es el servidor OPC DA, incorporado al Sistema de Supervisión, Control y Adquisición de Datos (SCADA) "Guardián del ALBA", para el intercambio en tiempo real de datos entre los procesos que este automatiza, utilizándose como una plataforma que permite facilitar la comunicación de forma estándar. Pero está sujeto a las limitaciones del estándar mencionadas anteriormente además de otras, tales como:

- No está acorde a la política de software libre de la universidad, al basarse en una tecnología Microsoft.
- Como solución transitoria, utiliza la seguridad brindada por el sistema SCADA para complementar la seguridad de Windows, pero sigue sin tener un mecanismo autónomo de seguridad.
- Sólo soporta el Acceso a Datos, por tanto no tiene soporte a Acceso a Datos Históricos, ni a Alarmas y Eventos, que aportan una mayor supervisión de la información, ya que el acceso a datos solo maneja puntos[4].

Atendiendo a la **situación problémica** planteada en el presente Trabajo de Diploma, se

[4] Variables específicas del SCADA, pueden ser analógicas o digitales. El valor en el SCADA se relaciona con el valor en el dispositivo a partir de conversión de unidades de campo a unidades de ingeniería y viceversa, este proceso es denominado linealización.

propone como **problema científico**: ¿Cómo permitir el intercambio entre aplicaciones de automatización y control de forma estándar y con interoperabilidad integrada entre las especificaciones en los proyectos del CEDIN?

Enmarcan la investigación en el **objeto de estudio** los servidores OPC, especificando el **campo de acción** en los servidores OPC UA.

Con el objetivo de resolver el problema planteado, se ha trazado como **objetivo general** desarrollar un servidor OPC UA que permita el intercambio de información entre aplicaciones de automatización y control de forma estándar y con interoperabilidad integrada.

Por tanto, se plantea como **idea a defender** que el desarrollo de un servidor OPC UA permitiría el intercambio entre aplicaciones de automatización y control de forma estándar y con interoperabilidad integrada en los proyectos del CEDIN.

Con los siguientes **objetivos específicos**:

1. Analizar los principales conceptos asociados al estándar OPC y OPC UA para obtener una base teórica necesaria para el desarrollo de la solución.
2. Estudiar y seleccionar las tecnologías, metodología de software, herramientas y lenguaje de programación más adecuados, según las necesidades de la organización, especificaciones de la OPC y características del software.
3. Modelar y diseñar un servidor OPC UA según las especificaciones de la Fundación OPC.
4. Implementar sobre una plataforma libre el servidor OPC UA que permita el intercambio de información entre aplicaciones de automatización y control de forma segura, estándar y con interoperabilidad integrada.
5. Probar los componentes del sistema implementado para validar su correcto funcionamiento.
6. Documentar la propuesta de servidor OPC UA para que pueda servir de consulta durante la realización de posteriores versiones.

Conforme a este planteamiento se derivan las siguientes **tareas de la investigación** a desarrollar:

1. Análisis de los principales conceptos asociados al estándar OPC y OPC UA para obtener una base teórica necesaria para el desarrollo de la solución.
2. Realización de un estudio acerca de los proveedores de servidores OPC UA, SDK y soluciones existentes que permitan el intercambio de información entre aplicaciones de automatización, para determinar fortalezas e insuficiencias en las mismas y valorar la vialidad de su uso.
3. Selección y argumentación de tecnologías y metodología de software a utilizar en el desarrollo del servidor de prueba.
4. Levantamiento de los requerimientos del sistema y descripción de los procesos a automatizar.
5. Diseño del sistema teniendo en cuenta la arquitectura y patrones de desarrollo de software para el correcto funcionamiento del sistema.
6. Implementación de los componentes que conforman el servidor OPC UA de prueba.
7. Documentación del código fuente siguiendo los estándares definidos por la herramienta seleccionada.
8. Ejecución de pruebas y solución de no conformidades para garantizar la calidad del software aplicando las normas internacionales requeridas para el producto.

Para llevar a cabo las tareas de la investigación se emplearon los siguientes **métodos de Investigación Científica**:

Métodos Teóricos:

1) **Análisis histórico y lógico:** Con el objetivo de analizar la evolución de los estándares de la Fundación OPC, su situación actual, arquitectura y principales características.

2) **Análisis y síntesis:** Para realizar el análisis y estudio de las bibliografías

existentes sobre el tema y lograr obtener de manera sintetizada el contenido necesario y suficiente para la realización del presente trabajo.

3) **Método comparativo:** Este método resulta de gran utilidad para establecer las ventajas y desventajas de los estándares desarrollados por la Fundación OPC, así como argumentar la selección del modelo OPC UA.

4) **Modelación:** Para la creación de modelos (propuestas, alternativas y estrategias) que visualizan una reproducción simplificada de la realidad.

Métodos Empíricos:

1) **Criterios de especialista:** Con el fin de consultar a las personas que poseen dominio sobre el tema y que sus opiniones ayuden al perfeccionamiento de la investigación.

Al concluir el Trabajo de Diploma se espera obtener como **principal resultado** el análisis, diseño e implementación de un servidor OPC UA que permita el intercambio de información entre aplicaciones de automatización y control de forma estándar y con interoperabilidad integrada

El desarrollo del documento se encuentra compuesto por tres capítulos donde se refleja el trabajo investigativo, el diseño del sistema y la implementación de la solución propuesta, distribuido de la siguiente manera:

- **Capítulo 1:** Fundamentación Teórica, definición de las herramientas y tecnologías utilizadas.

  Este capítulo cuenta con los conceptos fundamentales para comprender la solución así como el respaldo teórico de la misma. Se explican las metodologías, tecnologías y lenguajes seleccionados para ser utilizados en el desarrollo de la solución, tomando como principio la selección de herramientas libres.

- **Capítulo 2:** Descripción de la solución propuesta

Se definen los requisitos con los que debe cumplir el sistema, así como la arquitectura sobre la cual se realiza el diseño e implementación de la solución. Se describe la modelación propuesta y el diseño del sistema. Se plantea el modelo de datos y se describe la forma en que son utilizados los diferentes patrones que enriquecen la arquitectura de la solución.

- **Capítulo 3:** Implementación y pruebas del sistema

  Se describe la implementación de la solución y las pruebas realizadas a las funcionalidades desarrolladas. Además, se detallan los estándares de codificación y documentación utilizados, destacándose que estos forman parte de los estándares aplicados por el CEDIN.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

El estudio del estado y evolución de la temática analizada representa un elemento clave dentro de cualquier investigación. Es necesario poseer una fuerte base teórica que sirva de respaldo a la investigación así como saber manejar sus conceptos fundamentales. El presente capítulo contiene la fundamentación teórica que da respaldo a la solución propuesta. Se abordan temas relativos a la comunicación OPC con sistemas externos como SCADA y aplicaciones, enfocándose en el uso de estos desde la tecnología OPC UA. Se realiza además un estudio de sistemas y aplicaciones ya existentes de servidores OPC, para obtener una retroalimentación y posibilidades de desarrollo común en esa área. Se analiza el ambiente de desarrollo propuesto caracterizando las herramientas y tecnologías que se definieron para ser utilizadas en la confección de la solución.

1.2 Sistemas de automatización y control

La automatización es un conjunto de técnicas que relaciona sistemas mecánicos, eléctricos y electrónicos, que se combinan para luego ser dirigidos o controlados por medio de un software especializado, que se encarga de poner en movimiento a este mecanismo complejo de una forma automática. (Clubensayos, 2012).

Los sistemas SCADA, utilizan el computador y las tecnologías de comunicación para automatizar el monitoreo y control de procesos industriales. Estos sistemas son partes integrales de la mayoría de los ambientes industriales complejos o geográficamente dispersos, ya que pueden recoger la información de una gran cantidad de fuentes rápidamente y la presentan a un operador en una forma amigable. (García, 2008)

Se define entonces que un sistema SCADA es la combinación de un conjunto de hardware y software que permite la colección, visualización, procesamiento, filtración, análisis y almacenamiento de los datos proporcionados por dispositivos de adquisición de forma tal que perdure toda esa información.

Un sistema SCADA se compone por módulos o bloques de software que permiten las actividades de adquisición, supervisión y control, uno de estos módulos es el de Comunicación que se encarga de la transferencia de información entre la planta y la arquitectura hardware[5] que soporta el SCADA y entre esta y el resto de los elementos informáticos de gestión (Domingo, 2004).

En los sistemas de automatización se encuentran múltiples elementos de control y monitorización, cada uno con su protocolo de comunicaciones específico (Modbus, BSAP, Ethernet-IP, entre otros) y con características propias. Para el acceso a cada elemento de control se necesita utilizar un programa distinto por cada protocolo. (OPC Foundation, 2010). Incluso los servidores OPC necesitan implementar o usar estos programas para poder llegar a los dispositivos de campo. Se puede tener programas genéricos que te permiten visualizar datos de múltiples dispositivos que hablan protocolos diferentes.

Esto trae consigo que los desarrolladores de sistemas de automatización, se vean obligado a utilizar el software y hardware de un determinado proveedor, en la mayoría de los casos. Las aplicaciones de software desarrolladas por un fabricante normalmente no son compatibles con el hardware de otro, aunque tengan el mismo protocolo, ya que no implementan interfaces comunes para poder lograr la compatibilidad entre ambos. Este problema se resolvió con el surgimiento de OPC.

1.3 OPC

OPC es el método de conectividad de datos basado en los estándares más populares del mundo. Es utilizado para comunicar dispositivos, controladores y aplicaciones sin caer en los problemas habituales de las conexiones basadas en protocolos propietarios, respondiendo a uno de los mayores retos de la industria de la automatización (Kominek, 2009).

[5] Se distinguen 2 capas, la capa cliente (responsable de la interacción hombre-máquina) y la capa servidor de datos (control de datos del proceso). En sistemas sencillos estas dos capas se encuentran en un único hardware.

OPC no es un protocolo, sino más bien un estándar para la conectividad de datos que se basa en una serie de especificaciones OPC gestionadas por la Fundación OPC. Cualquier software que sea compatible con estas especificaciones proporciona a usuarios e integradores conectividad abierta e independiente tanto del fabricante del dispositivo como del desarrollador de la aplicación cliente (Kominek, 2009).

El primer y más exitoso estándar OPC Clásico- OPC Data Access- fue diseñado como interfaz para los drivers[6] de comunicación, permitiendo de una manera estandarizada acceso a lectura y escritura de datos en dispositivos de automatización. Mayormente es utilizado en los sistemas SCADA y HMI (Human Machine Interface, en inglés) accediendo a datos desde diferentes tipos de dispositivos y hardware de automatización de diferentes proveedores, utilizando una interfaz de software definida por el hardware del proveedor. Después, surgieron estándares como el OPC Alarmas & Eventos y OPC Acceso a Datos Históricos, que también fueron diseñados para acceder a información ofrecida por los sistemas SCADA (UA ANSI C Server Professional, 2011).

1.3.1 OPC y el problema de la conectividad

Hoy en día la automatización se utiliza de modo generalizado en todas las principales industrias. Mientras distintas industrias utilizan distintos sistemas de control y aplicaciones, todas comparten un reto común, que crece rápidamente: cómo compartir información entre todos estos componentes y el resto de la empresa.

A continuación, se enumeran los factores que tradicionalmente causaban los mayores problemas al compartir información, junto a una breve explicación del impacto que ha tenido OPC sobre cada uno de ellos:

- **Protocolos propietarios**: Los fabricantes utilizaban frecuentemente protocolos que permitían a productos de una determinada gama comunicarse entre ellos, pero requerían protocolos personalizados para la comunicación con productos de

[6] Manejador o controlador de dispositivos, programa informático que permite al sistema operativo interactuar con un periférico.

otros fabricantes. Para empeorar las cosas, distintas gamas de productos del mismo fabricante frecuentemente no eran capaces de intercambiar información, necesitando conectores adicionales. OPC resuelve este problema haciendo innecesario que el cliente de Datos tenga que conocer cómo comunica el servidor de Datos o cómo organiza dichos datos (Gómez, 2013).

- **Drivers de comunicación propietarios:** Todas las conexiones punto a punto requerían un protocolo propietario para posibilitar la comunicación entre los extremos específicos. Por ejemplo, si un HMI necesitaba comunicarse con un PLC (Programmable Logic Controller, en inglés), se requería de un driver en el HMI, escrito específicamente para el protocolo utilizado por el PLC. Si el driver específico para establecer la comunicación entre los dos extremos no estaba previamente desarrollado y disponible, el intercambio de datos era muy difícil y caro de establecer.(Gómez, 2013) OPC elimina la necesidad de disponer de drivers de comunicación propietarios para la comunicación entre cada aplicación y la Fuente de Datos

- **Integración compleja:** El uso habitual de protocolos propietarios para cada aplicación significaba que, incluso con un pequeño número de aplicaciones, se requería el uso de muchos protocolos. Utilizar OPC simplifica enormemente la integración porque, una vez que se configura un servidor OPC para una Fuente de Datos en particular, todas las aplicaciones que utilizan OPC pueden empezar a compartir datos con esa Fuente de Datos (Kominek, 2009), eliminando la necesidad de drivers de comunicación propietarios

- **Carga de trabajo en el dispositivo:** Cada protocolo establece su propia conexión al controlador para el que está diseñado para comunicar. Dado el gran número de protocolos que se utilizan en una instalación típica, frecuentemente el controlador se veía bombardeado por múltiples peticiones de la misma información para cada aplicación que necesitaba comunicar con él. El tráfico y, por tanto, la carga de los controladores se reduce enormemente utilizando conectores OPC, porque un conector OPC específico para un controlador requiere una única conexión a la

Fuente de Datos mientras que puede comunicar simultáneamente con múltiples aplicaciones para enviarles los Datos obtenidos (Kominek, 2009).

- **Obsolescencia de infraestructuras:** A medida que los fabricantes lanzan nuevos productos, eventualmente dejan de dar soporte a los antiguos. OPC extiende la vida útil de sistemas antiguos porque, una vez que se ha configurado un servidor OPC para el sistema, permite que cualquier aplicación cliente que utilice OPC (la mayoría) pueda comunicar con el sistema antiguo, sin importar si la aplicación cliente soporta o no de forma nativa la comunicación con dicho sistema antiguo (Kominek, 2009).

- **Conectividad corporativa:** OPC hace posible de forma real que se puedan compartir datos provenientes de la automatización a lo largo de toda la red corporativa, permitiendo que aplicaciones validadas reciban datos con Fuentes de Datos de la red de automatización, eliminando la necesidad de instalar nuevos drivers de comunicación (Kominek, 2009).

1.3.2 Funcionamiento de OPC

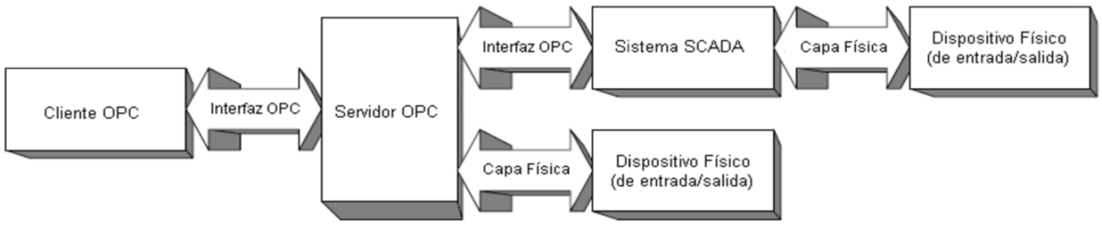
La "abstracción de dispositivo[7]" OPC se consigue utilizando dos componentes OPC especializados llamados cliente OPC y servidor OPC (Figura 1). Es importante resaltar que el hecho de que la Fuente de Datos y el Cliente de Datos puedan comunicarse entre sí mediante OPC no significa que sus respectivos protocolos nativos dejen de ser necesarios o hayan sido reemplazados por OPC. Al contrario, estos protocolos e interfaces nativas siguen existiendo, pero sólo se comunican con uno de los dos componentes del software OPC. Son los componentes OPC los que intercambian información entre sí, cerrando así el círculo. La información puede viajar de la aplicación al dispositivo sin que estos tengan que comunicarse directamente entre sí (Herver, 2012).

[7] OPC abstrae de los detalles de la implementación de Fuentes de Datos y Clientes de Datos, con lo que los datos se pueden intercambiar entre ellos sin que tengan que saber nada de sus respectivos protocolos de comunicación nativos y de la organización interna de sus datos.

OPC es un estándar dividido en varias especificaciones, las más comunes son:

- OPC DA (Data Access, Acceso a Datos).
- OPC HDA (Historical Data Access, Acceso a Datos Históricos).
- OPC A&E (Alarm and Events, Alarmas y Eventos).
- OPC DX (Data Exchange, Intercambio de datos).
- OPC XML-DA (Extensible Markup Language-Data Access, Lenguaje de Marcado Extensible para Acceso a Datos).
- OPC Security (Seguridad).

Figura 1: Relación Cliente – Servidor de OPC



1.4 OPC UA

Además del problema de la independencia de plataforma, las compañías miembros de OPC propusieron el requisito de exponer datos y sistemas complejos, removiendo las limitaciones del OPC Clásico. El OPC Arquitectura Unificada (OPC UA) nace entonces del deseo de crear un remplazo para todas las especificaciones basadas en COM existentes, sin perder ninguna característica o función. Adicionalmente, se debía cubrir todos los requerimientos de las interfaces de los sistemas de plataforma independiente con capacidades enriquecedoras y extensibles de modelado siendo además capaz de describir sistemas complejos. (Mahnke, 2009)

OPC UA es el nuevo estándar de la Fundación OPC proveyendo interoperabilidad en procesos automatizados y más. Definiendo servicios abstractos, OPC UA provee una

arquitectura orientada a servicios (SOA) para aplicaciones industriales. OPC UA integra las diferentes partes de la anterior especificación OPC en un espacio de direcciones unificada accesible desde un determinado conjunto de servicios (Leitner, 2010).

La Fundación OPC ha liberado un conjunto de estándares ampliamente aceptados en la industria que proveen interoperabilidad en la industria automatizada. OPC DA (DA) permite el acceso a datos, OPC HDA (HDA) permite el acceso a datos históricos y OPC A&E (AE) el acceso a alarmas y eventos.

1.4.1 Arquitectura de OPC UA

OPC UA especifica un conjunto abstracto de servicios y el mapeo de una tecnología concreta. OPC UA no especifica un API[8] sino solo los formatos de mensajes de datos a intercambiar por la red. Pilas de comunicación son utilizadas en los lados cliente y servidor para encriptar y desencriptar los mensajes de peticiones y respuestas. Diferentes pilas de comunicación pueden trabajar juntas mientras utilicen la misma tecnología de mapeo.

1.4.1.1 Visión General

Los componentes fundamentales de la Arquitectura unificada OPC son los mecanismos de transporte y el modelado de datos. (Figura 2).

La transportación define diferentes mecanismos optimizados para diferentes casos de uso. La primera versión de OPC UA define un protocolo binario TCP (*Transmission Control Protocol,* en inglés) optimizado para una comunicación de alto rendimiento de la intranet, así como un mapeo para aceptar estándares de Internet como Servicios Web, XML y HTTP.

Ambos transportes utilizan el mismo modelo base de seguridad para el intercambio de mensajes de los Servicios Web. El modelo de comunicación abstracto no depende de un

[8] **Interfaz de programación de aplicaciones** (Application Programming Interface, API) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

mapeo de protocolo específico y permite agregar nuevos protocolos en un futuro.

El modelado de datos define las reglas y bloques bases para exponer un modelo de información de OPC UA. Este define también los puntos de entrada en el espacio de direcciones y los tipos bases usados para construir la jerarquía de tipos. Esta base puede ser extendida por modelos de información construidos sobre los conceptos de modelado abstracto. En adición, define algunos conceptos mejorados como la descripción de las máquinas de estado utilizadas en diferentes modelos de información.

Figura 2: Cimientos de OPC UA

Los servicios UA son las interfaces entre los servidores como suministradores de los modelos de información y los clientes como consumidores de este modelo de información. Los servicios están definidos de una manera abstracta. Este concepto básico de OPC UA permite a un cliente acceder a la más mínima parte de los datos sin necesidad de entender el modelo completo expuesto por un sistema complejo. Los clientes OPC UA entendiendo modelos específicos pueden utilizar características mejoradas definidas por dominios y casos de uso especiales.

1.4.1.2 Componentes cliente y servidor

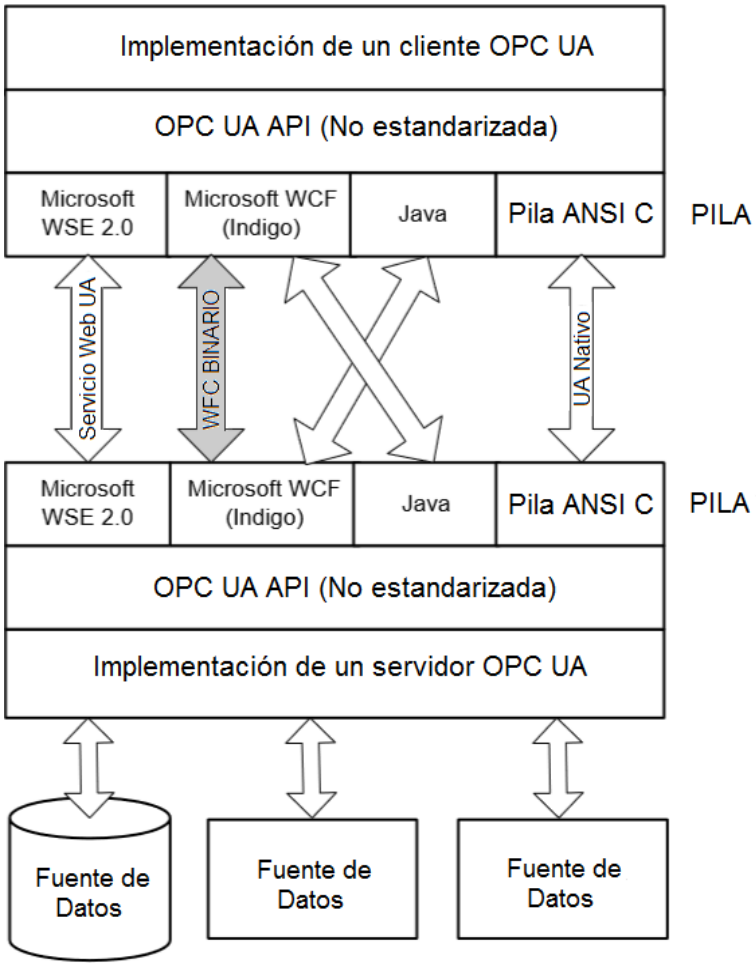
Un cliente OPC UA consiste en una implementación de un cliente utilizando una pila de comunicación OPC UA. La implementación del cliente accede a la pila de comunicación usando la API de OPC UA. API que no está estandarizada y puede cambiar debido a los diferentes lenguajes de programación y potencialmente, por diferentes pilas de comunicación. La pila del cliente se comunica con la pila de comunicación del servidor, la Fundación OPC estandarizó solamente esta comunicación, pero cualquiera puede incluso desarrollar su propia pila con su propia API. Muchas pilas de comunicación pueden existir para diferentes sistemas operativos, lenguajes de programación y mapeos. Por ejemplo, puede existir para el lenguaje de programación Java y otra para la nueva WFC (Windows Communication Foundation, en inglés) de Microsoft. La pila de comunicación del lado del cliente permite crear peticiones basadas en las definiciones del servicio (Figura 3.).

La pila de comunicación del lado del servidor envía los mensajes de petición a la implementación del servidor mediante la API de OPC UA. Desde que la API de OPC UA realiza la especificación del servicio abstracto, puede ser lo mismo desde el lado del cliente. La solución del servidor implementa la lógica necesaria para retornar el mensaje de respuesta apropiado. La implementación del servidor OPC UA obtiene sus datos de un sistema subyacente, por ejemplo, este puede ser una base de datos de configuración, un conjunto de dispositivos o algún servidor OPC.

Dentro de las especificaciones de OPC UA existe el concepto de servidores aglomerados. Un servidor aglomerado agrega uno o más servidores OPC UA y provee la información de esos servidores en su espacio de direcciones. Pero, el cliente no tiene que acceder a varios servidores, sino a solo uno. Este mecanismo permite una arquitectura flexible al encadenar varios servidores OPC UA para diferentes clientes con diferentes requerimientos. Por ejemplo, varios servidores OPC UA corriendo en dispositivos pequeños serán aglomerados por un servidor OPC UA. Varios clientes del sistema DCS (Distributed Control System, en inglés) pueden acceder a este servidor. Otro servidor OPC UA aglomera este servidor y provee parte de la información al sistema MES (Manufacturing Execution System, en inglés). El sistema MES puede trabajar como un

servidor aglomerado también, OPC UA soporta los servidores aglomerados ya que permite marcar el origen del dato.

Figura 3 : Arquitectura OPC UA



1.4.2 Especificaciones de OPC UA

Las especificaciones OPC UA están particionadas, particiones también requeridas por la estandarización IEC (International Electrotechnical Commission, en inglés) (OPC UA es conocida como el estándar IEC 62541); las centrales definiendo la base para OPC UA y las partes específicas de los tipos de accesos, detallando mayormente los modelos de información de OPC UA.

Para cubrir todas las características exitosas del OPC Clásico, modelos de información del dominio del proceso de información están definidos por OPC UA.

Espacio de direcciones

El conjunto de Objetos y la información relacionada que el servidor OPC UA hace disponible para los clientes es llamado Espacio de Direcciones. El Espacio de Direcciones OPC UA representa su contenido como un conjunto de Nodos conectados por Referencias. Las características primitivas de los Nodos están descritas por Atributos. Los Atributos son los únicos elementos de un servidor que tienen valores de datos. Los tipos de datos que definen los valores de los Atributos pueden ser simples o complejos. Los Nodos en el Espacio de Direcciones son tipificados según su uso y significado (OPC UA Part 1, 2012).

Alarmas & Condiciones (AC)

Las condiciones se usan para representar al estado de un sistema o uno de sus componentes. Algunos ejemplos comunes son:

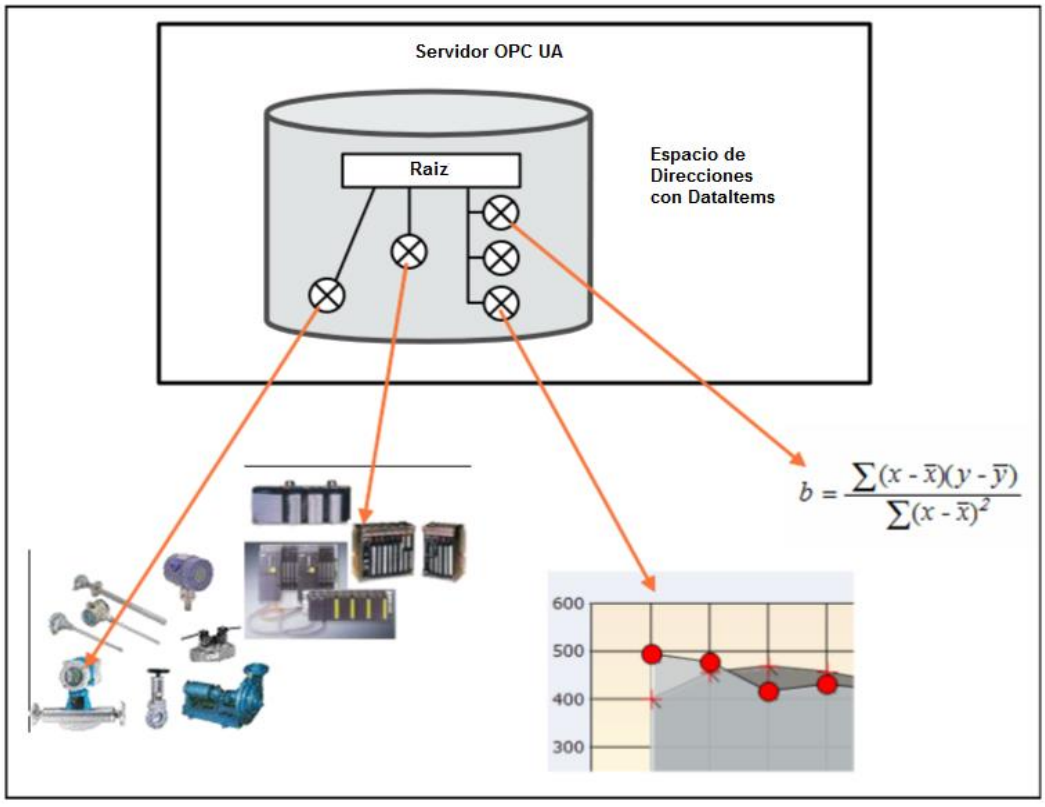
- Una temperatura excediendo un límite configurado.
- Un dispositivo necesitando mantenimiento.
- Un proceso que requiere a un usuario para confirmar algún paso antes de proceder.

Las alarmas son especializaciones de las condiciones, que le añaden los conceptos de estado activo, estado engavetado y estado suprimido a una Condición. Una Alarma en el estado activo indica que la situación que la Condición está representando actualmente existe. Cuando una Alarma está inactiva está representando una situación que ha regresado a un estado normal (OPC UA Part 9, 2010).

Acceso a Datos (DA)

El acceso a datos se ocupa de la representación y el uso de datos de automatización en servidores OPC UA. Los datos de automatización pueden estar ubicados dentro del servidor OPC UA o en tarjetas I/O directamente conectadas al servidor UA. También puede situarse en subservidores o en otros dispositivos como controladores o módulos de entrada/salida, conectados por enlaces seriales a través de buses de campo u otro enlace de comunicación. Los enlaces para las instancias de datos de automatización son llamados DataItems. Las categorías de datos de automatización son especificadas completamente por el proveedor. La Figura 4 ilustra cómo el Espacio de Direcciones de un servidor UA podría consistir en un alcance amplio de diferentes DataItems (OPC UA Part 8, 2006).

Figura 4 : Relación de los DataItems de OPC con los datos de automatización.



Programas (Prog.)

Las facilidades integradas de automatización manejan sus operaciones a través del cambio de datos y la invocación coordinada de funciones del sistema. OPC UA define Métodos y Programas como una forma interoperable para anunciar, descubrir y pedir estas funciones. Proveen un mecanismo que normaliza la descripción semántica, invocación y el resultado de estas funciones. Los métodos representan funciones básicas en el servidor que puede ser invocado por un cliente. Los programas por contraste, son modelos más complejos en el sistema (OPC UA Part 10, 2007).

Acceso Histórico (HA)

Un servidor OPC UA manteniendo Acceso Histórico provee a uno o más clientes OPC UA acceso transparente a datos históricos o eventos históricos. Los datos históricos o eventos pueden ser hallados en una colección de datos privativos, base de datos o un buffer de corto plazo en memoria. Un servidor OPC UA manteniendo Access Histórico puede o no proveer eventos y datos históricos para una cierta cantidad o todas las Variables, Objetos, Propiedades o Vistas dentro del Espacio de Direcciones. (OPC UA Part 11, 2009).

1.4.3 Ventajas de OPC UA

OPC UA ofrece ventajas sobre su predecesor, algunas de estas son:

Acceso a datos unificado: Mientras que las anteriores especificaciones de la Fundación OPC servían a los propósitos por los que fueron creados, ellos no estaban conectados. Por lo tanto, no existía una conexión entre una valor actual leído con DA con el histórico leído con HDA o eventos levantados basados en este valor. OPC UA provee todos los datos en su espacio de direcciones unificado. De esta manera, los datos actuales, los datos históricos y los eventos están relacionados entre sí.

Requerimientos adicionales: OPC UA soporta un conjunto de nuevos rasgos, como acceso a eventos históricos, jerarquía múltiple y provee métodos y programas (a menudo llamados comandos). Un gran logro de OPC UA es un modelo de datos mayor por encima de una simple información de tipo de datos; en donde la anterior especificación solo

proveía una jerarquía simple con objetos conteniendo datos, OPC UA provee un extenso meta modelo donde esos objetos son tipificados. En adición para proveer un objeto de tipo de dato Float y alguna meta información como el diseño de la unidad, OPC UA permite tipificar el ítem, así puede por ejemplo, ser identificado como un sensor de calor. Equipado con un poderoso y extensible modelado de tipos OPC UA permite adicionar y eliminar objetos y referencias entre ellos.

Migración tecnológica: Los antiguos estándares de la Fundación OPC estaban basados en tecnología COM/DCOM de Microsoft. Microsoft actualmente dejó a un lado COM/DCOM en favor de Servicios Web y SOA con capacidad multiplataforma. Los proveedores demandan especificaciones independientes de plataforma que permitan correr aplicaciones OPC en sistemas que no sean de Microsoft. OPC UA soporta esto mediante la especificación de un sistema abstracto de servicios y mapas de diferentes tecnologías tales como Servicios Web.

1.5 Desarrollo de OPC en el mundo

KepServerEx: Es uno de los servidores OPC desarrollado por la compañía Kepware, empresa que tiene prestigio a nivel mundial en software de comunicaciones para la automatización y ofrece una experiencia única, tanto en OPC como en dispositivos de comunicaciones embebidos. Es una solución flexible y escalable para conectar, administrar, monitorear y controlar diversos dispositivos de automatización y aplicaciones de software (Kepware Technologies, 2015)**.**

OPC DataHub: Es parte de la familia de productos Cogent DataHub[9]. Es una herramienta de integración de datos OPC que proporciona OPC puente, túnel OPC y agregación de servidores y clientes OPC, así como OPC a la conectividad de base de datos SQL, todo en un solo paquete. El OPC DataHub envía y recibe datos desde cualquier servidor OPC DA y se conecta en tiempo real a Excel y aplicaciones personalizadas construidas en C ++, .Net y Java a través de TCP. También se puede enviar datos OPC a través de

[9] Es una herramienta de software que permite conectar, integrar, guardar y mostrar datos en tiempo real.

mensajes de correo electrónico y SMS y ofrece conectividad a las aplicaciones nativas de Linux (OPC DataHub®, 2015)

MatrikonOPC es el mayor proveedor mundial de productos de conectividad OPC, con una colección de más de 500 interfaces como servidores OPC, clientes OPC y OPC HDA. Con más de 30 000 usuarios y más de 100 000 instalaciones en todo el mundo, MatrikonOPC ofrece conectividad a todos los sistemas de control más importantes y de aplicación en el mercado. (Microsoft, 2010)

Fundación OPC cuenta con más de 300 miembros de todo el mundo, incluyendo casi todos los principales proveedores del orbe en los sistemas de control, instrumentación y sistemas de control de procesos, los cuales tributan al mejoramiento y desarrollo de las especificaciones, siendo esto una garantía de valor agregado, tiene 15 000 productos y son miembros 450 empresas (OPC Foundation, 2010).

IndustrialIT SCADA Vantage de ABB es un sistema SCADA que se usa normalmente en el sector del petróleo y el gas. La información proporcionada incluye casos y tipos, datos actuales, alarmas y eventos e histórico. Incluye:

- Alto rendimiento
- Facilidad de apoyo y mantenimiento
- Alta disponibilidad
- Bajo costo total de propiedad
- El uso de estándares de la industria (Microsoft, ODBC, SQL, VB, OPC, entre otros)
- Ejecutar en hardware compatible con Windows
- Utilizable en toda la empresa como una interfaz hombre-máquina (HMI) - desde pequeñas pantallas locales a las salas de control centralizados y escritorios corporativos (SCADA VANTAGE, 2003);

IndustrialIT eXtended PAT (Tecnología analítica de procesos) de ABB se presentó en 2007 y fomenta la integración de las medidas analíticas en el proceso de fabricación. Utiliza OPC UA para ofrecer una conectividad normalizada a los analizadores de

procesos. Con su potente capacidad de integración y funcionalidad, xPAT ayuda a los clientes del sector de la salud a implantar la calidad a lo largo de todo el ciclo de vida del producto farmacéutico, desde el descubrimiento de medicamentos hasta la producción pasando por el desarrollo. (Control Engineering, 2010)

Ya se han lanzado los primeros productos de otros proveedores, incluso antes de que se presentara la especificación. Entre ellos se encuentra el sistema HMI/SCADA de ICONICS, el GÉNESIS 64, que también utiliza la OPC UA para las comunicaciones internas, el TwinCat de Beckhoff que funciona en controladores, así como el SIMATIC NET de Siemens. Largas listas de competidores han prometido distribuir sus primeros productos de OPC UA, como Emerson, Honeywell, Wonderware y Yokogawa. (Arquitectura OPC UA, 2009) A continuación se muestra un listado publicado por Thomas J. Burke, Presidente de la Fundación OPC y su Director Ejecutivo:

Figura 5: Listado de distribuidores de OPC

ABB	Invensys/Foxboro	SISCO
Absynt Technologies Ltd	Invensys/Wonderware	**SMAR**
ascolab GmbH	**Kepware**	**Softing AG**
Beckhoff	**Matrikon**	**Software Toolbox**
CAS	Metso Automation	SRI International
Cognex	**Microsoft**	Tampere University of Technology
Cyberlogic	OPC-F	Technosoftware AG
Helsinki University of Technology	**OSIsoft, Inc.**	VTT
Honeywell	Prosys PMS Ltd	Wapice Ltd
Iconics	**Rockwell**	**Yokogawa Electric Asia**
InduSoft LLC	SAP	
Ing.-Buero Allmendinger	**Siemens**	

1.6 SDK UA

La SDK UA es una biblioteca C++ (dividida en SDK Servidor y SDK Cliente) que soporta

el desarrollo de servidores y clientes OPC UA portables. Ambas SDK utilizan la misma biblioteca base UA, que maneja toda la encapsulación C++ de los tipos C ANSI[10] sin procesar que están definidos en la pila de comunicación OPC UA de la Fundación OPC. (SDK OPC UA, 2010)

La SDK está disponible como versión binaria tanto para Windows como para Linux. También está disponible como código fuente. El código SDK es la mejor solución si se quiere desarrollar una aplicación para un nuevo sistema operativo, compilar las bibliotecas para una versión optimizada en un CPU específico que proporciona un mejor funcionamiento o si se quiere un control completo sobre la aplicación (SDK OPC UA, 2010).

Al compilar las bibliotecas se puede elegir construir bibliotecas estáticas o dinámicas. Usando bibliotecas estáticas se puede crear un solo binario que incluya todas las dependencias. Con bibliotecas dinámicas se tiene más flexibilidad al mejorar partes específicas del software.

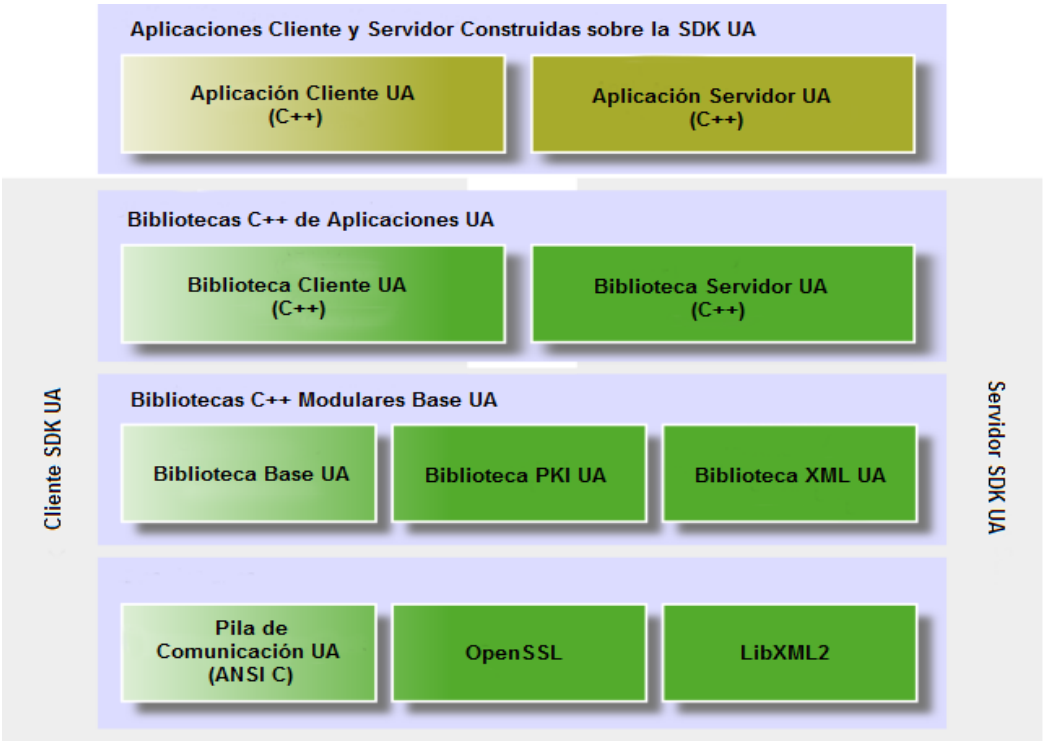
La portabilidad de la SDK UA está garantizada por el concepto de las capas de portabilidad. El resto del código es genérico e independiente del sistema operativo y puede ser reutilizado sin ningún cambio. La pila de comunicación UA tiene una capa de portabilidad que encapsula las llamadas más importantes del sistema operativo. La Biblioteca Base UA está construida sobre esta pila y no necesita ninguna otra capa de portabilidad. Lo mismo se aplica para las capas de software construidas sobre la Biblioteca Base.

Los módulos adicionales, Biblioteca PKI UA y Biblioteca XML UA, son capas de abstracción adicionales que encapsulan componentes especiales de software. Utilizan los proyectos de código abierto OpenSSL y LibXML2 para implementar estas funcionalidades, porque están actualmente portables para muchos sistemas operativos y

[10] Estándar publicado por el Instituto Nacional Estadounidense de Estándares (*ANSI*), para el lenguaje de programación *C*.

no tiene licencias restrictivas como la GPL (General Public License, en inglés).

Figura 6: Visión general de la arquitectura del SDK UA



Debido a las facilidades y funcionalidades que ofrece esta SDK en el desarrollo de un servidor OPC UA, fue escogido como plataforma para el desarrollo de la solución propuesta.

1.6.1 Seguridad

OPC UA requiere una autenticación bidireccional de las aplicaciones cliente y servidor con certificados de instancias de aplicaciones X509[11] durante el establecimiento de una conexión de comunicación segura. Se requieren certificados de instancia de aplicación para identificar de forma única cada instalación o instancia de una aplicación. El servidor

[11] X.509 es un estándar UIT-T para infraestructuras de claves públicas (en inglés, Public Key Infrastructure o PKI). X.509 especifica, entre otras cosas, formatos estándar para certificados de claves públicas y un algoritmo de validación de la ruta de certificación.

proporciona su certificado instancia de la aplicación a través del conjunto de servicios de descubrimiento. El cliente proporciona su certificado instancia de la aplicación en el Servicio OpenSecureChannel.

El certificado de instancia de aplicación del servidor puede ser creado como certificado autofirmado durante la instalación o el inicio del servidor OPC. La biblioteca PKI proporciona funcionalidad para crear y gestionar certificados. Como segunda opción el certificado instancia de aplicación del servidor puede ser proporcionado por el administrador del servidor OPC. Para esta opción, la herramienta de configuración del servidor debe proporcionar una forma de reemplazar el certificado generado o el cambio del nombre del certificado utilizado.

1.7 Metodología, lenguajes y herramientas utilizados.

Basándose en el estudio realizado de las diferentes metodologías, lenguajes, herramientas y tecnologías existentes y su desarrollo actual, se decide utilizar las listadas en este capítulo por su peculiaridad de pertenecer a la categoría de software libre. Además, con el uso de las mismas se está siendo consecuente con las normas utilizadas por el Centro de Informática Industrial.

1.7.1 Lenguaje de modelado: UML

El lenguaje de modelado es la notación (principalmente gráfica) utilizada para expresar un diseño, el proceso indica los pasos que se deben seguir para llegar a este. (González, 2009). El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

Este constituye un lenguaje para especificar y no para describir métodos o procesos. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software pero no especifica en sí mismo qué metodología o proceso usar. (Proyecto Zero, 2010)

1.7.2 Lenguaje de programación: C++

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. (Florentino, 2007)

C++ es un lenguaje de programación diseñado con la intención de extender al exitoso lenguaje de programación C con mecanismos que permitiesen la manipulación de objetos. Este lenguaje tiene facilidades para la programación genérica, estructurada y orientada a objetos, por lo que es un lenguaje multiparadigma. C++ permite trabajar tanto a alto como a bajo nivel. (Calvo, 2006)

Se decide utilizar C++ como lenguaje de programación para el desarrollo del sistema debido a que:

- Es un lenguaje orientado a objetos.
- Es muy potente y práctico, debido a su robustez, para la creación de sistemas complejos.
- Es independiente de la plataforma.
- Es muy utilizado en el mundo por lo que existe abundante información sobre su uso.
- Existen muchos algoritmos ya desarrollados en C++, de manera que pueden tomarse y amoldarse a la solución deseada.

1.7.3 Metodología de desarrollo: AUP UCI

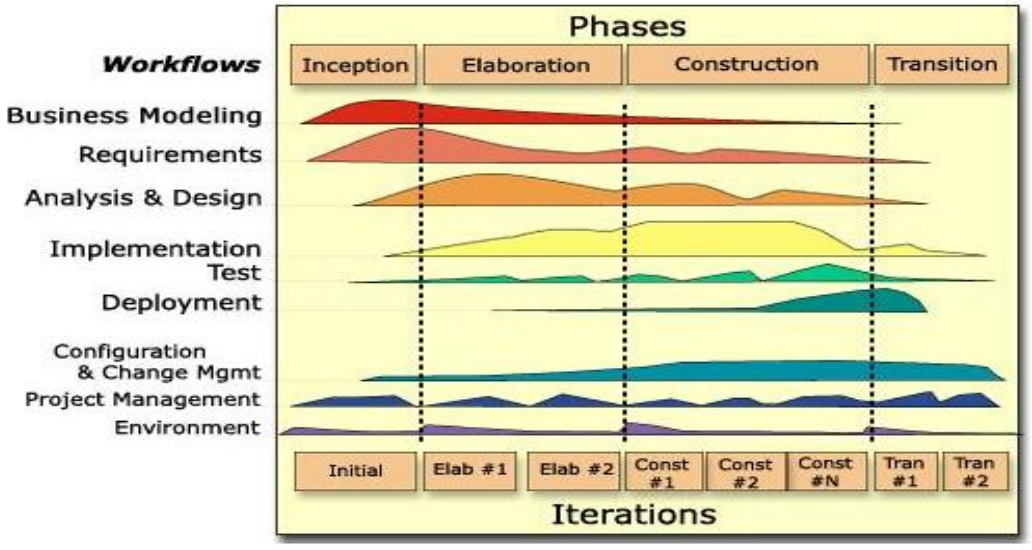
Una metodología de desarrollo es una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia y la eficiencia en el proceso de generación de software. (Blanco, 2008)

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP[12]) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP por sus siglas en inglés). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.

El AUP UCI aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas (Test Driven Development - TDD).
- Modelado Ágil.
- Gestión de Cambios Ágil.
- Refactorización de Base de Datos para mejorar la productividad.

Figura 7: Metodología de desarrollo de software: AUP



El proceso unificado (Unified Process o UP) es un marco de desarrollo de software iterativo e incremental. A menudo es considerado como un proceso altamente ceremonioso porque especifica muchas actividades y artefactos involucrados en el

[12] Versión simplificada de RUP, que favorece el desarrollo ágil de proyectos.

desarrollo de un proyecto de software. Dado que es un marco de procesos, puede ser adaptado y la más conocida es RUP (Rational Unified Process) de IBM. (Ervin, 2009)

El proceso AUP establece un Modelo más simple que el que aparece en RUP por lo que reúne en una única disciplina las disciplinas de Modelado de Negocio, Requisitos y Análisis y Diseño. El resto de disciplinas (Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno) coinciden con las restantes de RUP como se muestra en la figura 7. (Charles, 2013)

1.7.4 Herramienta CASE: Visual Paradigm

Una herramienta CASE es un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. (Bello, 2008) Las herramientas CASE abarcan todos los pasos del desarrollo del software y también aquellas actividades generales que se aplican a lo largo del proceso. Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

Se decide utilizar Visual Paradigm como herramienta CASE debido a que:

- Posee capacidades de ingeniería directa e inversa.
- Puede integrarse a los principales entornos de desarrollo.
- Posee disponibilidad de múltiples versiones para cada necesidad.

1.7.5 Entorno de desarrollo: QtCreator

Un entorno de desarrollo integrado (Integrated Development Environment, IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de éstos. Los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado (por ejemplo C, C++, C#, Java, Python y Visual

Basic, entre otros). (Universidad de Oviedo, 2009)

Se decide utilizar QtCreator debido a que es un IDE gratuito, libre, potente, tiene gran soporte en la comunidad de software libre y permite instalar complementos o plugins[13] para lograr un entorno de desarrollo ampliado con posibilidades variadas de uso.

1.7.6 Base de datos: PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (PostgreSQL, 2009)

Se decide la utilización de PostgreSql ya que provee soporte profesional tanto de la comunidad como de empresas especializadas, requerimientos de administración y mantenimiento relativamente bajos con respecto el resto de bases de datos comerciales, fiabilidad y estabilidad legendarias y un rendimiento excelente.

1.7.7 Generador de documentación: Doxygen

El paquete Doxygen permite generar la documentación correspondiente a una aplicación. Puede generar archivos en HTML[14] y un manual de referencia a partir de un grupo de ficheros fuente documentados. Se decide utilizar Doxygen ya que contiene un sistema de documentación para C++ que es el lenguaje de programación que se utiliza en el sistema. Además, brinda la facilidad de extraer la documentación directamente de las fuentes, lo que hace mucho más fácil mantener la consistencia con el código.

[13] Aplicación que se relaciona con otra para aportarle una función nueva y específica.
[14] Siglas en inglés: HyperText Markup Language (Lenguaje de Marcado de Hipertexto), empleado para elaborar páginas web.

Capítulo 2: Descripción de la solución

2.1 Introducción

El presente capítulo contiene la descripción de la solución propuesta, descripción que se compone de la definición de la solución, el modelo de dominio, la definición de los requisitos de la aplicación, tanto funcionales como no funcionales y las especificaciones de los requisitos funcionales utilizando historias de usuario. Se define la arquitectura como organización fundamental de la solución, en donde se recogen los patrones arquitectónicos a utilizar. Además se realiza un análisis y diseño del sistema compuesto por el diagrama de clases, patrones de diseño y clases persistentes. Tomando en referencia los conceptos y términos descritos en el capítulo anterior, regido por la metodología seleccionada.

2.2 Definición de la solución

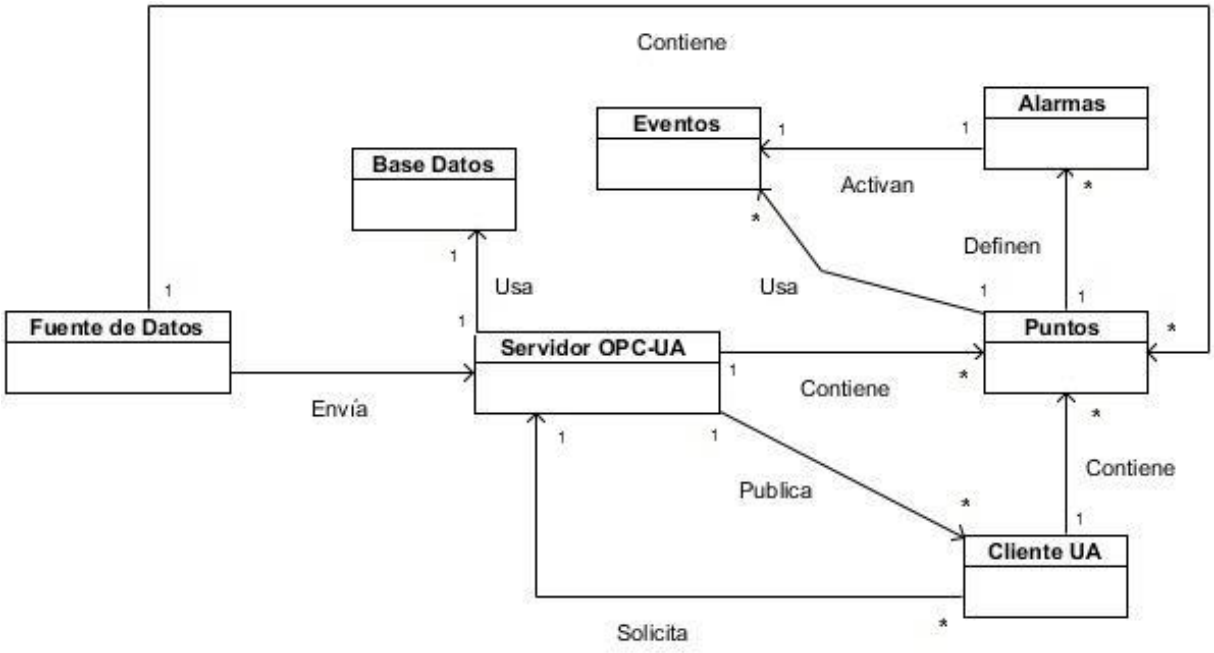
La propuesta de servidor OPC-UA para sistemas de automatización y control es diseñada como una arquitectura para la comunicación entre una fuente de datos y clientes OPC UA utilizando como base la SDK OPC UA que permita la publicación de puntos, alarmas y datos históricos a partir de los puntos obtenidos de la fuente de datos.

2.3 Modelo del dominio

Un modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema.

Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. Este tipo de modelo tiene como objetivo principal ayudar a comprender los conceptos con los que deberá trabajar la aplicación. Estos conceptos se representan en el siguiente diagrama (Figura 8):

Figura 8: Modelo del dominio.



2.3.1 Glosario de términos del dominio

Servidor OPC-UA: Provee todos los datos en su espacio de direcciones unificado a clientes UA; datos actuales, datos históricos y eventos son relacionados unos con otros.

Fuente de Datos: Sistema subyacente que provee los datos al servidor, este puede ser una base de datos de configuración, un conjunto de dispositivos o algún servidor OPC.

Cliente UA: Implementa funcionalidades que permiten enviar mensajes de petición al servidor OPC-UA basadas en las definiciones del servicio.

Base Datos: Guarda los puntos y eventos históricos asociados a estos, así como la configuración del servidor.

Puntos: Variables específicas que se refieren a los datos que el servidor obtiene de la fuente de datos y pública al cliente, estos datos son contenidos en el espacio de direcciones del servidor.

Eventos: Son usados para describir una ocurrencia de cierto significado en el Sistema**.**

Alarmas: Activa un evento, dependiendo de una condición previa declarada de acuerdo al estado de un punto

2.4 Requisitos del sistema

Los requisitos con que debe cumplir un sistema se clasifican en funcionales y no funcionales. Los funcionales son capacidades o condiciones que el sistema debe cumplir, mientras que los no funcionales son propiedades o cualidades que hacen al producto atractivo, usable, rápido y confiable.

2.4.1 Requisitos funcionales

La propuesta de servidor OPC-UA para sistemas de automatización y control debe brindar funcionalidades que permitan la comunicación entre una fuente de datos y un cliente OPC UA. Los requisitos funcionales con que el sistema debe cumplir son:

RF 1. Configurar parámetros generales del servidor: configurar los parámetros generales del servidor.

RF 2. Configurar puntos: configurar los parámetros concernientes a los puntos.

RF 3. Configurar alarmas: configurar los parámetros concernientes a las alarmas.

RF 4. Iniciar servidor: iniciar un servidor OPC UA utilizando la configuración predefinida.

RF 5. Detener servidor: opción de interrupción del servidor OPC UA.

RF 6. Conectar con fuente de datos: conexión con una fuente de datos a través de conexión TCP-IP.

RF 7. Conectar con base de datos interna: conexión a una base de datos hospedada en un servidor PostgreSQL.

RF 8. Guardar la configuración: guardar la configuración del servidor para su reutilización.

RF 9. Obtener puntos: obtener puntos de la fuente de datos.

RF 10. Publicar puntos: publicar a clientes OPC UA la información de los puntos que se obtengan de la fuente de datos.

RF 11. Publicar alarmas: publicar a clientes OPC UA la información de las alarmas generadas a partir de los puntos que se obtengan de la fuente de datos.

RF 12. Guardar datos históricos: guardar en base de datos la información correspondiente a los puntos históricos.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales con que el sistema debe cumplir son:

- **Requisitos de rendimiento y disponibilidad del sistema.**

  RNF 1. El sistema debe garantizar su disponibilidad en todo momento, a excepción del límite de tiempo de ejecución del servidor impuesto por la versión de prueba del SDK OPC UA, utilizado para el desarrollo de la solución.

  RNF 2. El sistema debe garantizar la rapidez de las comunicaciones para la validez de la información en tiempo real.

- **Restricciones en el diseño y la implementación.**

  RNF 3. El sistema debe contar con un conjunto de patrones que permita la reutilización del código así como facilidad en la actualización del mismo.

- **Requisitos de confiabilidad.**

  RNF 4. Se debe garantizar que el sistema no afecte la fiabilidad e integridad de la información que se transmite.

  RNF 5. Se debe garantizar que el sistema no afecte la disponibilidad de las fuentes de datos.

  RNF 6. Las actividades de mantenimiento, supervisión y edición del sistema no deben interferir en el correcto desempeño del resto del sistema, ni afectar la ejecución de la aplicación.

- **Requisitos de apariencia o interfaz externa**

  RNF 7. El sistema debe proveer una interfaz visual amigable e intuitiva.

- **Requisitos de soporte.**

  RNF 8. Se debe ofrecer servicios de mantenimiento y actualización.

- **Requisitos de portabilidad.**

  RNF 9. El sistema debe funcionar en el sistema operativo GNU/Linux.

- **Requisitos de documentación de usuario.**

  RNF 10. Se debe garantizar que el sistema cuente con una correcta documentación.

- **Requisitos asociados al licenciamiento.**

  RNF 11. Se debe garantizar que el sistema se desarrolle bajo los principios del software libre y por tanto cualquier componente de software que se utilice también debe cumplir con esta premisa.

2.5 Historias de Usuario

Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos, son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos, también permiten responder rápidamente a los requisitos cambiantes (qVision, 2014).

Características:

- **Independientes unas de otras:** cada una tiene un resultado, si se relacionan.

- **Negociables**: La discusión con los usuarios debe permitir esclarecer su alcance y éste debe dejarse explícito bajo la forma de pruebas de validación.

- **Valoradas por los clientes o usuarios:** Cada historia debe ser importante para los usuarios más que para el desarrollador.

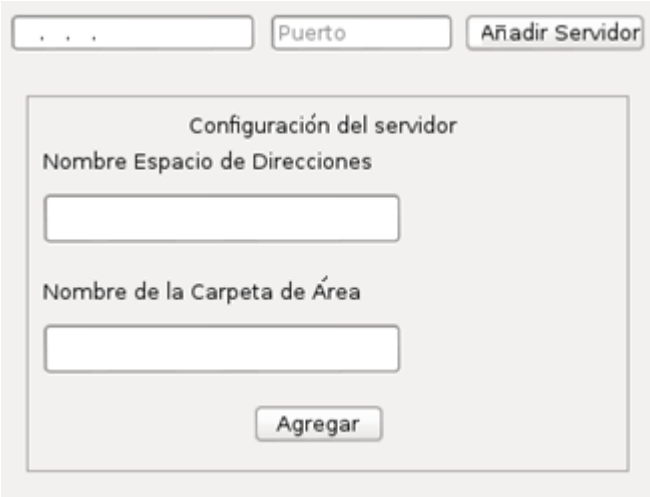
- **Estimables:** Un resultado de la discusión de una historia de usuario es la estimación del tiempo que tomará completarla.

- **Short (cortas):** Generalmente se recomienda la consolidación de historias muy cortas en una sola historia.

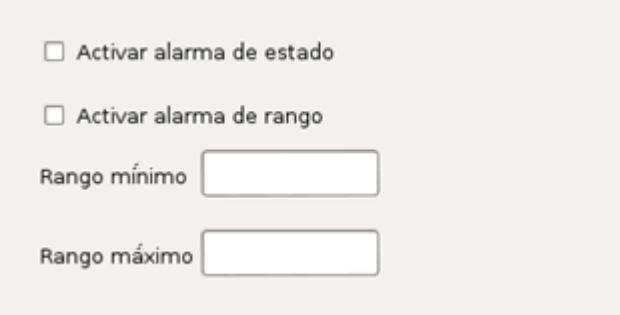
- **Testeable:** Las historias de usuario cubren requerimientos funcionales, por lo que generalmente son verificables.

2.5.1 Descripción de requisitos de software

<table>
<tr><th colspan="3">HISTORIA DE USUARIO</th></tr>
<tr><td>Número: HU1</td><td colspan="2">Nombre del requisito: Configurar parámetros generales del servidor.</td></tr>
<tr><td colspan="2">Programador: Lorgio Vargas Vento.</td><td>Iteración Asignada: 1</td></tr>
<tr><td colspan="2">Prioridad: Alta</td><td>Tiempo Estimado: 15 días.</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo:
Rotura de alguna estación de trabajo perteneciente al proyecto.
Problemas eléctricos.</td><td>Tiempo Real: 2 semanas.</td></tr>
<tr><td colspan="3">Descripción: Se inserta la configuración general del servidor OPC UA, los valores a insertar son la dirección IP y puerto de la fuente de datos, el nombre del Espacio de Direcciones y el nombre de la Carpeta de área, además debe mostrar las configuraciones realizadas hasta el momento.</td></tr>
<tr><td colspan="3">Observaciones: NA</td></tr>
<tr><td colspan="3">Prototipo de interfaz:</td></tr>
</table>



HISTORIA DE USUARIO	
Número: HU2	**Nombre del requisito:** Configurar puntos.
Programador: Lorgio Vargas Vento.	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 15 días.
Riesgo en Desarrollo: • Rotura de alguna estación de trabajo perteneciente al proyecto. • Problemas eléctricos.	**Tiempo Real:** 2 semanas.
Descripción: Debe permitir la configuración de los puntos a obtener de la fuente de datos. De estos se configurará el nombre del punto, el tipo (analógico o digital), su rango y si se guardarán sus datos históricos o no.	
Observaciones: NA	
Prototipo de interfaz: 	

<table>
<tr><th colspan="3">HISTORIA DE USUARIO</th></tr>
<tr><td>Número: HU3</td><td colspan="2">Nombre del requisito: Configurar alarmas.</td></tr>
<tr><td colspan="2">Programador: Lorgio Vargas Vento.</td><td>Iteración Asignada: 2</td></tr>
<tr><td colspan="2">Prioridad: Alta</td><td>Tiempo Estimado: 15 días.</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo:
Rotura de alguna estación de trabajo perteneciente al proyecto.
Problemas eléctricos.</td><td>Tiempo Real: 2 semanas.</td></tr>
<tr><td colspan="3">Descripción: Debe permitir la configuración de las alarmas correspondientes a los puntos a obtener de la fuente de datos. De esta se configurará si se activará o no la alarma de estado de la variable y la variable de rango, de activarse esta última se especificarán sus rangos: mínimo y máximo.</td></tr>
<tr><td colspan="3">Observaciones: NA</td></tr>
<tr><td colspan="3">Prototipo de interfaz:
 
</td></tr>
</table>

En los Anexos de esta investigación pueden consultarse el resto de las Historias de Usuario.

2.6 Definición de la Arquitectura

La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución.

Los patrones arquitectónicos son patrones de software que ofrecen soluciones a problemas de arquitectura en ingeniería de software. Dan una descripción de los

elementos y el tipo de relación que tienen, junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones (González, 2015).

2.6.1 Arquitectura Cliente-Servidor

La tecnología Cliente-Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requisitos a uno o más servidores centrales. Este modelo provee usabilidad, interoperabilidad, flexibilidad y escalabilidad en las comunicaciones (Oposiciones TIC, 2011).

Figura 9: Esquema de comunicación Cliente-Servidor.

En esta arquitectura el cliente envía un mensaje al servidor solicitando un servicio, en este caso el servidor OPC UA es la fuente de datos (como un dispositivo hardware a nivel de planta) y cualquier aplicación basada en OPC UA debe poder acceder a dicho servidor para leer/escribir cualquier variable que ofrezca el servidor. La figura (Figura 9) muestra el esquema de comunicación Cliente-Servidor empleado en la Propuesta de Servidor OPC UA para Sistemas de Automatización y Control.

2.6.2 Patrón arquitectónico Modelo-Vista-Controlador (MVC)

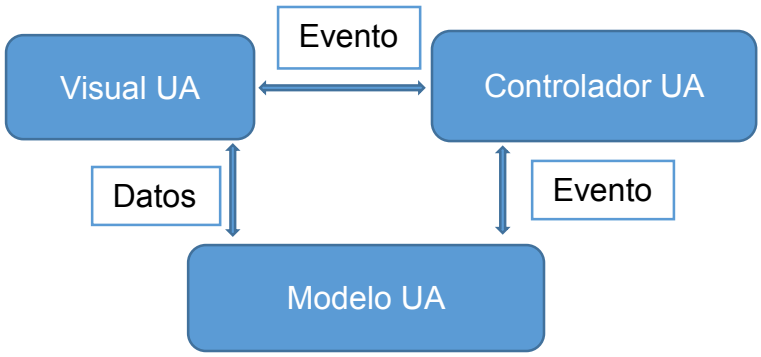
Patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento. (Patrones de Diseño, 2013)

El Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

El Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

La Vista: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

Figura 10: Patrón arquitectónico modelo-vista-controlador del servidor OPC UA.



2.7 Análisis y diseño del sistema

Esta disciplina explica cómo transformar los requisitos en los productos de trabajo que especifiquen el diseño del software que se va a desarrollar.

Los principales propósitos de esta disciplina son:

- Transformar los requisitos en un diseño de lo que será el sistema.
- Evolucionar una arquitectura robusta para el sistema.
- Adaptar el diseño para que se ajuste al entorno de implementación, con un diseño pensado para el rendimiento. (Jacobson, 2000)

El análisis consiste en obtener una visión del sistema a partir de los requisitos funcionales. Por otro lado, el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, para determinar de qué forma deberá cumplir el sistema sus objetivos.

2.7.1 Clases persistentes

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases persistentes se definen para conocer la información real representada en las tablas de la base de datos. (Mesa & Castillo, 2008)

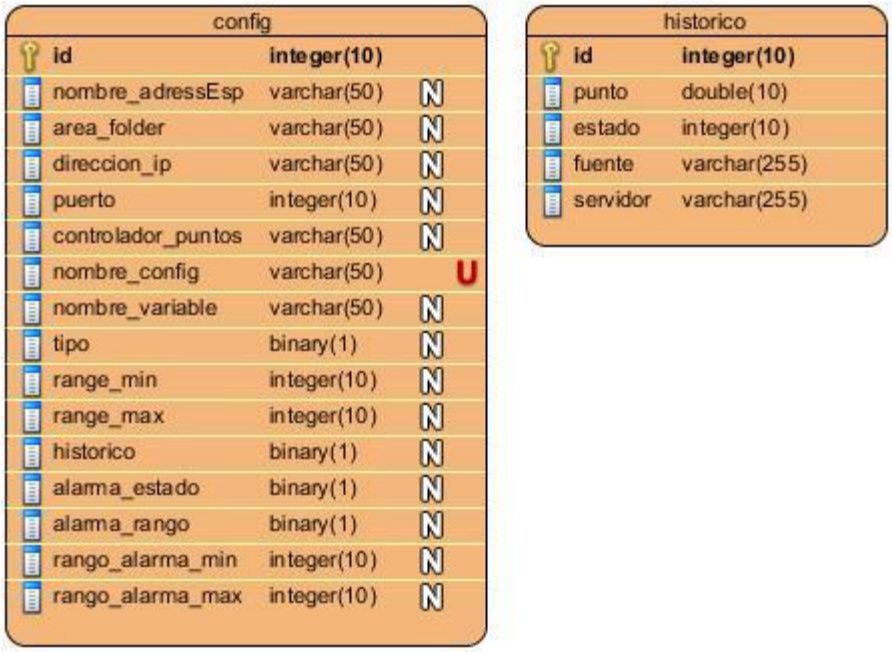
El sistema debe alamacenar los datos históricos de los puntos asociadas obtenidos de la

fuente de datos, también debe almacenar las configuraciones del servidor generadas por el usuario.

La clase dbcontrollers es la encargada de realizar las operaciones relacionadas con el almacenamiento de las configuraciones, puntos y alarmas históricos. Esta clase gestiona el uso de la base de datos PostgreSQL, pudiendo de esta manera agregar, eliminar y cargar las configuraciones del servidor y agregar los puntos históricos.

La base de datos se compone por dos tablas, “config” (encargada de guardar la configuración) e “historico” (encargada de guardar los datos históricos).

Figura 11: Clases persistentes.



Descripción de las tablas:

- config
  - id: llave primaria.
  - nombre_adressEsp: nombre del Espacio de Direcciones.
  - area_folder: nombre de la carpeta de área.
  - dirección_ip: dirección IP de la fuente de datos.

  - puerto: puerto de la fuente de datos.
  - controlador_puntos: nombre del controlador de puntos.
  - nombre_config: nombre de la configuración.
  - nombre_variable: nombre de la variable.
  - tipo: tipo de la variable (true: analógico, false: digital).
  - range_min: rango mínimo de la variable.
  - range_max: rango máximo de la variable.
  - historico: si se guardará los datos históricos de la variable.
  - alarma_estado: si se activará la alarma de estado de la variable.
  - alarma_rango: si se activa la alarma de rango de la variable.
  - rango_alarma_min: rango mínimo de la alarma de rango.
  - rango_alarma_max: rango máximo de la alarma de rango.
- historico
  - id: llave primaria.
  - punto: valor del punto.
  - estado: el estado del punto.
  - fuente: timestamp del momento que se obtuvo el punto de la fuente de datos.
  - servidor: timestamp del momento que el servidor publicó el punto.

2.7.2 Diagramas de clases

El diagrama de clases captura la estructura lógica del sistema y las clases que constituyen el modelo. Es un modelo estático, describiendo lo que existe y qué atributos y comportamiento tiene. Los diagramas de clases son los más útiles para ilustrar las relaciones entre las clases e interfaces (Architect G. d, 2007).

Las siguientes figuras (Figuras 12, 13) muestran las clases que componen la solución y las relaciones entre ellas siguiendo la estructura definida por la arquitectura utilizada.

Figura 12: Diagrama de clases del paquete controladorUA.

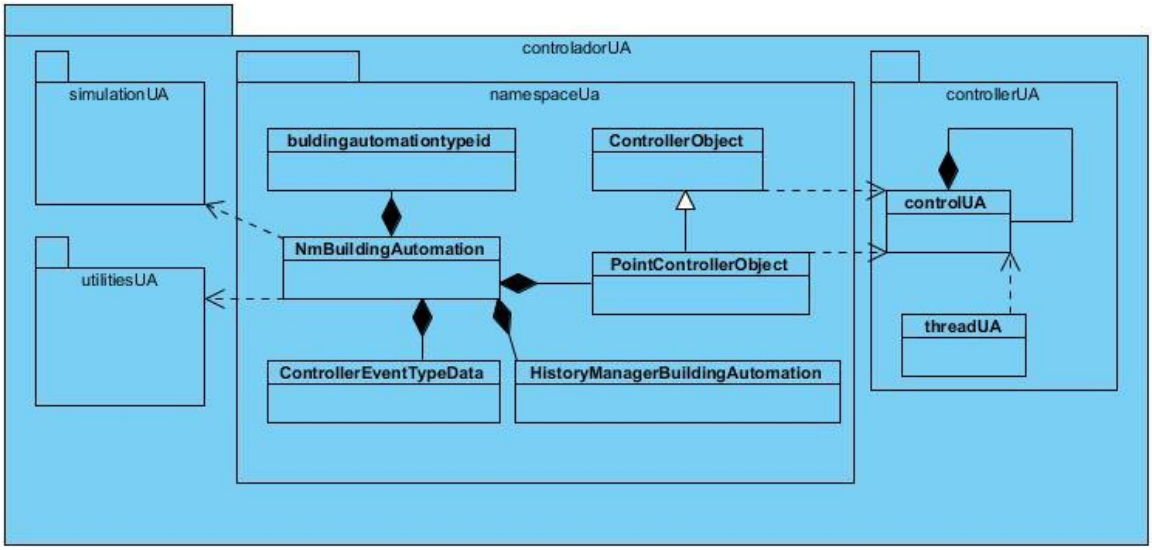
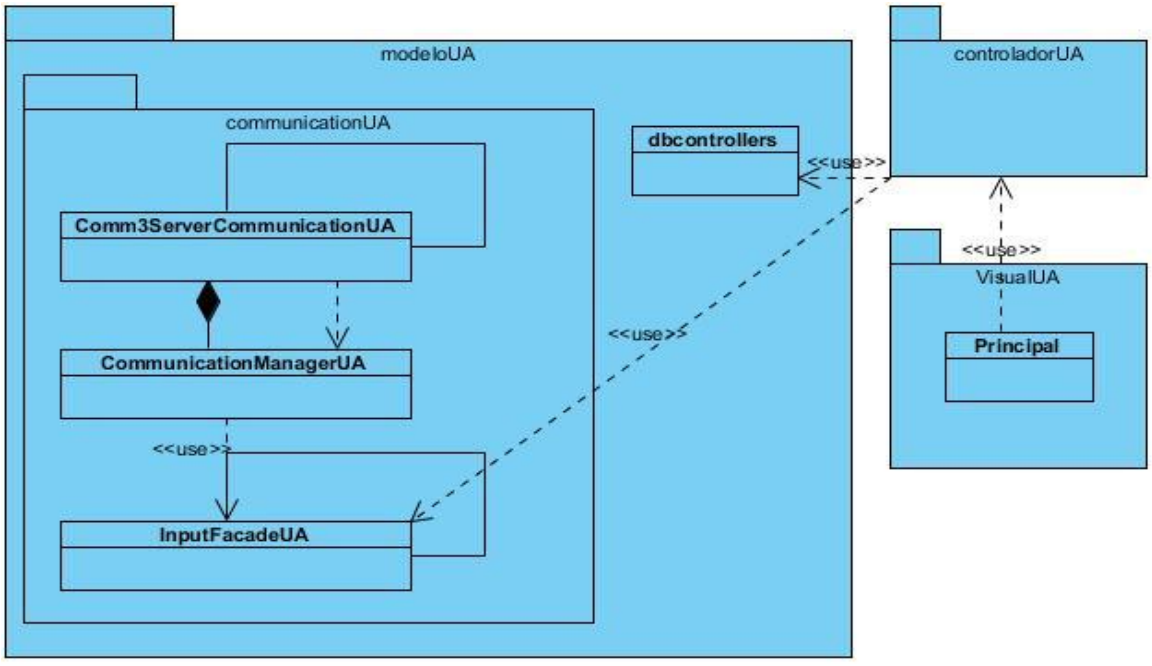


Figura 13: Diagrama de clases de los paquetes modeloUA y visualUA



2.7.2.1 Descripción de las principales clases

En los Anexos de esta investigación pueden consultarse las descripciones de las clases más importantes para el desarrollo de la Propuesta de servidor OPC-UA para Sistemas de Automatización y Control, donde se detallan cada una de las funcionalidades que las componen.

2.7.3 Patrones de diseño

Los patrones de diseño tienen diferentes clasificaciones en dependencia de su funcionalidad.

- **Patrones Creacionales:** Muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Estas decisiones, normalmente son resueltas dinámicamente, decidiendo qué clases instanciar o sobre qué objetos u objeto delegar responsabilidades.
- **Patrones estructurales:** Describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.
- **Patrones de comportamiento:** Se utilizan para organizar, manejar y combinar comportamientos (Grupo UML, 2006)

En el desarrollo de la aplicación, como patrón creacional se utilizó el Singleton o instancia única para restringir la instanciación del acceso a la base de datos y para la utilización de la clase controladora.

2.7.4 Patrón Singleton

El patrón de diseño Singleton se encarga de restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. Varias funcionalidades distintas pudieran necesitar el acceso a la base de datos, con el uso de este patrón aseguramos de que se acceda a sólo una instancia de esta (Carrascal, 2013).

En el sistema es utilizado en la clase controladora controlUA, esta clase contiene toda la información de la configuración del servidor, por lo que es necesario que exista una sola instancia de esta. También es utilizada en la clase dbcontrollers, clase dedicada al control e interacción con la base de datos del sistema, es necesario que exista sólo una instancia de esta clase, ya que es la encargada de la conexión con el servidor de base de datos y esta conexión debe ser única.

Capítulo 3: Implementación y pruebas del sistema.

3.1 Introducción

Las actividades de implementación y pruebas abarcan parte de las acciones más importantes para el desarrollo de un producto de software. En primer lugar, la implementación constituye el proceso de formación de la solución, que incluye el establecimiento de los estándares de codificación, el tratamiento de los errores, la elaboración de los componentes y sus relaciones, hasta el diseño del futuro despliegue del sistema. Por otra parte, las pruebas son las encargadas de obtener los errores cometidos durante el desarrollo del producto. El presente capítulo refleja el proceso y desarrollo de estas etapas en la solución propuesta.

3.2 Estándar de codificación

Los estilos de codificación y documentación explicados en este capítulo son aplicados en el Centro de Informática Industrial, esto justifica el empleo de los mismos en la propuesta de servidor OPC UA para sistemas de automatización y control. Los estilos utilizados evidencian la intención de lograr una aplicación que pueda ser comprendida sin necesidad de tener un amplio conocimiento respecto al tema.

3.2.1 Nombres

- Los nombres de las clases son sustantivos singulares.
- Los nombres de clases y objetos deben reflejar qué hacen y no cómo lo hacen.
- Escoger nombres lo suficientemente largos que expresen correctamente el sentido de lo que se quiere, pero evitando manejar nombres que dificulten la labor de implementación.
- Evitar nombres que permitan una interpretación subjetiva (evitar ambigüedad y asegurar abstracción).
- Evitar redundancia no repitiendo nombres de clases en sus elementos.
- Dado que los nombres generalmente son el producto de concatenar varias

palabras, se debe emplear la primera palabra en minúscula, mayúscula para denotar la letra de inicio de cada una de las palabras restantes por las que esté formado y minúscula para las letras intermedias en el caso de los nombres de métodos y funciones. Para el caso de los nombres de variables y atributos debe aplicarse la misma convención.

- Las variables booleanas deben contener la palabra *is* en su nombre.
- Los nombres de constantes deben contener solo letras mayúsculas.
- Minimizar el uso de abreviaturas. En caso de ser requeridas, se debe ser consistente en su uso y cada abreviatura debe significar solo una cosa.
- Los nombres de los métodos son frases que incluyen verbos.
- Los nombres de los atributos y parámetros son frases con sustantivos.

3.2.2 Codificación

- Se establece un tamaño de indentación[15] estándar de tres espacios, sin tabulaciones.
- Alinear secciones del código.
- Alinear verticalmente llaves de apertura y cierre.
- Usar espacios antes y después de los operadores que el lenguaje de programación permita.
- 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.
- Emplear constantes en sustitución de números o cadenas de caracteres literales.
- Minimizar el alcance de las variables para evitar confusión y facilitar el mantenimiento.
- Emplear cada variable y rutina solo para un propósito.

[15]Espacio o sangría que se pone a la derecha de cada línea de código.

- Evitar el uso de variables públicas, sustituirlas por variables privadas y métodos que provean el valor de tal variable, para mantener el encapsulamiento.
- Emplear las letras *i, j, k, l, m, p, q, r* para contadores en ciclos.
- Mantener la modularidad del código bajo el criterio de la lógica que encierra, no exagerar la modularidad.
- Emplear correctamente los tipos de ciclos: si es al menos una vez usar *do-while*, si es ninguna o más veces usar *while-do*, y si se conoce el número exacto de ciclos usar *for*.
- Inicializar todas las variables.
- Emplear líneas en blanco para separar pasos lógicos (Ej.: declaraciones, lazos).

3.3 Estándar de documentación.

Se utilizó la herramienta Doxygen para generar la documentación del código de las distintas clases utilizadas para la implementación de las bibliotecas, los formatos brindados por esta herramienta se explican a continuación.

3.3.1 Estilo de bloques de documentación.

Se adopta el estilo de bloques de documentación JavaDoc, el cual consiste en un bloque de comentario de estilo C. Este bloque de comentario comienza con dos asteriscos, los asteriscos que se encuentran en la línea de la mitad son opcionales. Ejemplo:

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

3.3.2 Descripción breve con el comando \brief o @brief.

Para hacer una descripción breve se adopta el uso del comando \brief o @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, tal como lo muestra el 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)
 */
```

Nota: Si no se utiliza el comando @brief, Doxygen toma la descripción hecha en el bloque como una descripción detallada y no como una descripción breve.

3.3.3 Descripción de argumentos y métodos.

La descripción de argumentos de métodos y funciones se encuentra en línea, es decir, luego de la declaración de cada uno de los argumentos se da una breve descripción de cada uno de ellos. En el siguiente ejemplo se muestra el formato a utilizar:

```
int multFunction (  int c , /**<Primer operando de la operación */
                    int d , /**<Segundo operando de la operación */
                    int e /**<Tercer operando de la operación */ );
```

3.3.4 Documentación de tipos de datos.

Para describir la función y los elementos que componen los tipos de datos definidos se pueden utilizar los formatos especificados en los apartados anteriores. A continuación se

muestra un ejemplo:

```
/**
* @brief Enumerado de los tipos de datos admitidos
*
* Descripción más detallada de la función de este tipo de dato.
*/ enum DataTypes{       INTEGER, /**<Puede ser un valor de tipo entero */
                        DOUBLE, /**<Puede ser un valor de tipo double */
                        STRING /**<Puede ser un valor de tipo string */ };
```

Observamos que se genera una descripción breve seguida de una descripción más detallada para la función, luego se explica brevemente el valor de retorno de la misma y la descripción de los parámetros usando el formato de documentación en línea.

3.3.5 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 Lorgio Vargas Vento
* @date 13-04-2015
*/
```

3.3.6 Comando @see.

Existe otro comando útil que permite hacer referencias a otras clases o métodos cuando esto sea necesario, es importante usar este comando en la documentación a la hora de implantar los métodos o funciones propias de alguna clase, este comando es @see y su uso se muestra en el siguiente ejemplo:

```
/**
 * Constructor de la clase
 * @see Test::Test()
 */
Test1();
```

Esto genera en la documentación para el constructor de la clase de prueba Test1 una referencia hacia la documentación existente para el constructor de la clase de prueba Test.

Nota: Si se quiere hacer una referencia desde cualquier estructura hacia la documentación completa de una clase, se debe utilizar el comando @see seguido del nombre de la clase de esta forma:

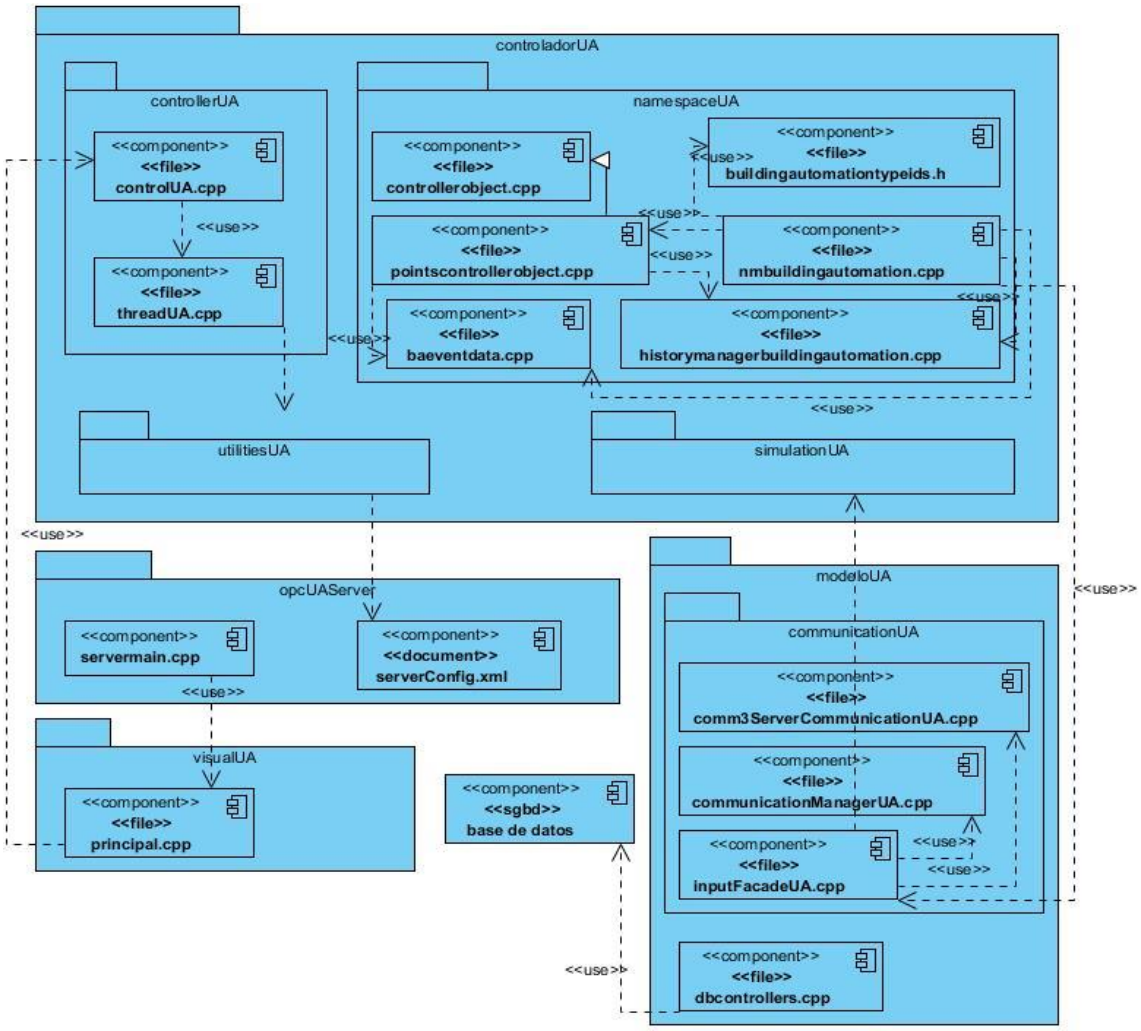
```
 /**
  * Constructor de la clase
  * @see Clase
  */
 Test1();
```

3.4 Implementación.

3.4.1 Diagrama de componentes.

Un diagrama de componentes ilustra los fragmentos de software, controladores embebidos, etc. que conforman un sistema. Este tipo de diagrama tiene un nivel de abstracción más elevado que un diagrama de clases. Usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. (Guía de Usuario de Enterprise Architect).

Figura 14: Diagrama de componentes



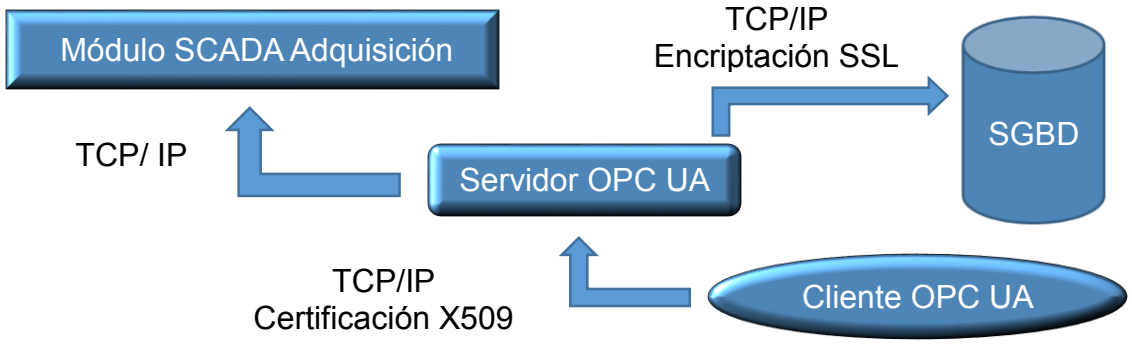
Un componente es una pieza de software que describe un conjunto de servicios que son usados sólo por medio de interfaces bien definidas. El diagrama (Figura 14) describe las interacciones existentes entre los principales componentes que integran la solución, así como sus dependencias más significativas.

3.4.2 Diagrama de despliegue.

Es un tipo de diagrama del Lenguaje Unificado de Modelado (UML) que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes (Diagramas de Despliegue, 2012).

El diagrama de despliegue del sistema muestra como la solución se comunica mediante TCP/IP al módulo de Adquisición del SCADA que actúa como fuente de datos. Mediante una conexión TCP/IP con encriptación SSL se comunicará con el Sistema de Gestión de Base de Datos (SGBD). Los cliente podrán comunicarse al servidor OPC UA mediante una conexión TCP/IP bajo un certificado del estándar X509.

Figura 15: Diagrama de despliegue.



3.5 Pruebas.

La realización de pruebas es una actividad en la cual un sistema o componente es ejecutado bajo ciertas condiciones o requisitos específicos, cuyos resultados son observados, registrados y evaluados. Estas técnicas ayudan a definir un conjunto de casos de prueba aplicando un cierto criterio, estos son determinados por los valores a

asignar a las entradas en su ejecución, realizando de esta forma un conjunto de pruebas unitarias a cada uno de los componentes del sistema atendiendo a las funcionalidades a cumplir por este. (Pruebas de Software, 2014)

Las pruebas unitarias en su enfoque funcional o de caja negra permiten comprobar el correcto funcionamiento de cada uno de los componentes del sistema, analizando las entradas y salidas y verificando que el resultado es el esperado. Se consideran exclusivamente las entradas y salidas del sistema sin tener en cuenta la estructura interna del mismo. Este método permite validar el software, entendiendo como validación el proceso que determina si el software satisface los requisitos anteriormente definidos.

3.5.1 Descripción de los casos de prueba.

A continuación se detallan los casos de prueba (CP) elaborados por cada una de los requisitos funcionales para comprobar el correcto funcionamiento del sistema bajo las diferentes condiciones a las que puede someterse.

Tabla 1: CP 1. Configuración del servidor.

Escenario	**Parámetros**		**Respuesta del Sistema**	**Resultados de la Prueba**
	IP	Puerto		
CP 1. Configurar el IP y puerto de la fuente de datos	10.8.8.168	65510	Reconocimiento de la fuente de datos por el IP y puerto especificado. Actualización del visual.	Prueba exitosa

Tabla 2: CP 2. Configuración general del servidor.

Escenario	Parámetros	Respuesta del Sistema	Resultados de la Prueba
	Nombre		
CP 2.1. Configurar el nombre de espacio de direcciones	ServidorOPC	Actualización del visual. Cliente actualizado.	Prueba exitosa
CP 2.2. Configurar el nombre de la carpeta de área de variables.	PuntoSCADA	Actualización del visual. Cliente actualizado.	Prueba exitosa
CP 2.3. Configurar el nombre del controlador de variables	ControladorPuntos	Actualización del visual. Cliente actualizado.	Prueba exitosa

Tabla 3: CP 3. Configuración de puntos.

Escenario	Parámetros					Respuesta del Sistema	Resultados de la Prueba
	nombre	rangoMin	rangoMax	tipo	historico		
CP 3.1. Configurar nombre de la variable	Punto	---	---	---	---	Actualización del visual. Cliente actualizado.	Prueba exitosa
CP 3.2. Configurar rango mínimo y	---	0	1000	---	---		Prueba exitosa

máximo de la variable.							
CP 3.3. Configurar el tipo de la variable (analógico o digital)	---	---	---	1	---		Prueba exitosa
CP 3.4. Especificar si se guardarán sus datos históricos o no.	---	---	---	---	true		Prueba fallida

Tabla 4: CP 4. Configuración de alarmas.

Escenario	**Parámetros**			**Respuesta del Sistema**	**Resultados de la Prueba**
	alarma	rango_min	rango_max		
CP 4.1. Configurar si se activará la alarma de estado	true			Actualización del visual. Cliente actualizado.	Prueba exitosa
CP 4.2. Configurar si se activará la alarma de rango, e insertar su rango mínimo y rango máximo	true	0	1000		Prueba exitosa

Tabla 5: CP 5. Iniciar servidor.

Escenario	Respuesta del Sistema	Resultados de la Prueba
CP 5. Iniciar servidor OPC UA con los datos de configuración predefinidos	Inicio correcto de servidor OPC UA.	Prueba exitosa

Tabla 6: CP 6. Detener servidor.

Escenario	Respuesta del Sistema	Resultados de la Prueba
CP 6. Detener servidor OPC UA	Servidor detenido exitosamente.	Prueba exitosa

Tabla 7: CP 7. Conectar con fuente de datos.

Escenario	Respuesta del Sistema	Resultados de la Prueba
CP 7. Conectar a un SCADA GALBA.	Conexión exitosa al SCADA.	Prueba exitosa

Tabla 8: CP 8. Conectar con base de datos.

Escenario	Respuesta del Sistema	Resultados de la Prueba
CP 8. Conectar con un servidor de base de datos PostgresSql.	Conexión exitosa a la base de datos.	Prueba exitosa

Tabla 9: CP 9. Guardar la configuración.

Escenario	Respuesta del Sistema	Resultados de la Prueba
CP 9. Guardar la configuración del servidor introducida en la base de datos de PostgresSql	Configuración guardada correctamente.	Prueba exitosa

Tabla 10: CP 10. Obtener puntos.

Escenario	Respuesta del Sistema	Resultados de la Prueba
CP 10. Obtener puntos de la fuente de datos SCADA	Puntos obtenidos correctamente.	Prueba exitosa

Tabla 11: CP 11. Publicar puntos.

Escenario	Respuesta del Sistema	Resultados de la Prueba
CP 11. Publicar al cliente los puntos obtenidos del SCADA.	Visualización de puntos en la vista de acceso a datos del cliente.	Prueba exitosa

Tabla 12: CP 12. Publicar alarmas.

Escenario	Respuesta del Sistema	Resultados de la Prueba

CP 12. Publicar los eventos generados a partir de los puntos obtenidos	Visualización de puntos en la vista de eventos en el cliente.	Prueba exitosa

Tabla 13: CP 13. Guardar datos históricos.

Escenario	Respuesta del Sistema	Resultados de la Prueba
CP 13. Guardar los datos históricos de los puntos obtenidos de la fuente de datos en la base de datos.	Puntos guardados correctamente en la base de datos. Visualización de los puntos históricos en la vista de tendencia histórica del cliente.	Prueba fallida

3.5.1.1 Resultados de los casos de prueba.

De los resultados obtenidos a partir de la aplicación de los diferentes casos de prueba efectuados a las funcionalidades del sistema se deriva la solidez del mismo. De un total de 19 pruebas realizadas al software, 17 resultaron satisfactorias y sólo 2 revelaron vulnerabilidades. Los errores detectados fueron solucionados, lo cual demuestra que el proceso de validar la primera versión funcional del sistema concluye de manera exitosa.

Conclusiones generales

Luego de terminado el proceso de investigación, desarrollo y validación de la aplicación realizada, se arriba a las siguientes conclusiones:

1- Se obtuvo la primera versión de un servidor OPC UA que resuelve los problemas actuales de interoperabilidad e independencia de plataforma que presenta el OPC Clásico.

2- Para el desarrollo de la aplicación se seleccionaron herramientas y tecnologías con las cuales es posible guiar el desarrollo del software propuesto.

3- Las funcionalidades desarrolladas para la propuesta de servidor OPC UA permite la obtención de puntos de una fuente de datos, la publicación de estos y las alarmas que generan a un cliente OPC UA, además de la creación de una base datos histórica de estos puntos publicados.

4- Las pruebas realizadas corroboraron la solidez de la implementación, demostrando que las funcionalidades desarrolladas cumplen con los requerimientos de software definidos.

Recomendaciones

- Adquisición de la versión completa de la SDK UA, esto permitirá eliminar las limitaciones que tiene la actual SDK de prueba utilizada en este trabajo de diploma.
- Inserción del estándar OPC UA entre los protocolos soportados por la línea de adquisición del Departamento de Construcción de Componentes.
- Implementación de la obtención de las alarmas y datos históricos directamente de la fuente de datos.

Referencias bibliográficas

1. **[Architect, G. d, 2007].** Architect, G. d. (2007) Guía de usuario de Enterprise Architect. [En línea] 6 de Abril de 2011. Obtenido de: http://www.sparxsystems.com.ar/index.php.
2. **[Arquitectura OPC UA, 2009]** MAHNKE, WOLFGANG, LEITNER, STEFAN-HELMUT. Marzo de 2009. Arquitectura OPC unificada. La norma futura para la modelización de las comunicaciones y la información en la automatización. Revista ABB 3/2009.
3. **[Bello, 2008].** BELLO LOBATO, VÍCTOR. Ingeniería de software I. [Citado el: 5 de junio de 2008]. [En línea] 19 de Enero de 2015. Obtenido de: http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.
4. **[Blanco, 2008].** BLANCO CUARESMA, SERGI. Marble Station. Metodologías de Desarrollo. [Citado el: 14 de Febrero de 2008]. [En línea] 19 de Enero de 2015. Obtenido de: http://www.marblestation.com/?p=644.
5. **[Calvo, 2006].** CALVO, LUIS. Lenguaje de Programación. [Citado el: 22 de Abril de 2006]. [En línea] 19 de Enero de 2015. Obtenido de: http://www.scribd.com/doc/Lenguaje-de-Programacion.
6. **[Charles Edeki, 2013]** EDEKI, CHARLES. Agile Unified Process. *International Journal of Computer Science and Mobile Applications* 2013, vol. 1, nº 3, ISSN 2321-8363.
7. **[Clubensayos, 2012]** Automatización - Composiciones de Colegio. Tecnología Ensayos. En línea [26 de jun. de 2012] [Citado el: 19 de Diciembre 2014] Disponible en: *www.clubensayos.com/Tecnología/Automatización/224556.html.*
8. **[Carrascal, 2013].** CARRASCAL HERNÁNDEZ, JAVIER (2013) Proyecto final de máster. Máster en Tecnologías de la Información Ingeniería de software y sistemas de información. Desarrollo de una herramienta de validación/importación para

Magento.

9. **[Control Engineering, 2010]** Process analysis: ABB Industrial IT Extended PAT.[En linea] 31 de mar. de 2010. [Citado el: 18 de enero 2015] Disponible en *http://www.controleng.com/single-article/process-analysis-abb-industrial-it-extended-pat/b8f9089bc0ebb84b49d60e1d956654b8.html*.

10. **[Diagramas de Despliegue, 2012].** Diagramas de Despliegue. [Citado el: 14 de mayo 2015] Obtenido de: http://www.diadspg.blogspot.com/.

11. **[Domingo, 2004]** DOMINGO PEÑA, JOAN, GRAU SALDES, ANTONI, MARTÍNEZ GARCÍA, HERMINIO, GÁMIZ CARO, JUAN. 2004. Diseño y aplicaciones con autómatas programables. Editorial UOC.

12. **[Ervin , 2009]** ERVIN FLORES, J. L. C. *Metodologías Agiles Proceso Unificado Ágil (AUP)* [Página web: Sitio oficial]. La Paz, Bolivia: Universidad Unión Bolivariana carrera de ingeniería de sistemas, Última actualización: 08 de julio de 2009. [Consultado el: noviembre de 2014]. Disponible en: http://ingenieriadesoftware.mex.tl/63758_AUP.html.

13. **[Florentino, 2007].** FLORENTINO, DEIDRY. (2007). Historia del computador y evolución, arquitectura, lenguajes de programación y algoritmos. [En línea] 18 de Enero de 2015. Obtenido de: http://nectacellcomputer.jimdo.com/historia-del-computador-arquitectura-lenguajes-de-programaci%C3%B3n-y-algoritmos/.

14. **[García, 2008].** GARCÍA GIMÉNEZ, JAVIER. Calibración, Control y Diseño SCADA de un robot paralelo neumático con el autómata S7-300. SCADA, una primera idea. 2008 [citado 16/01/2015]; Disponible en: http://repositorio.bib.upct.es/dspace/bitstream/10317/570/1/pfc2683.pdf.

15. **[Gómez, 2013]** GÓMEZ, OSVALDO. SERVIDORES OPC. [Citado el: 14 de Enero de 2015] [En línea] 16 de ago. de 2013; Disponible en: *https://prezi.com/dfx957sgsyat/untitled-prezi/.*

16. **[González, 2009].** GONZÁLEZ CORNEJO, J. E. ¿Qué es UML? [Citado el: 16 de Octubre de 2009]. [En línea] 27 de Noviembre de 2014. Obtenido de:

http://www.docirs.cl/uml.htm.

17. **[Grupo UML, 2006].** Grupo UML: junio 2006 [En línea] 13 de Jun. de 2006. [Citado el: 17 de marzo 2015] Obtenido de: http://umlpanamericana.blogspot.com/2006_06_01_archive.html.

18. **[Herver, 2012]** HERVER SEGURA, VICTOR HUGO, JIMENEZ MENDEZ, JAIME, TRINIDAD CORTEZ, CARLOS ANTONIO. Implementación de una maqueta prototipo utilizando un plc micrologix® 1400, labview® y tecnologías inalámbricas. Escuela superior de ingeniería mecánica y eléctrica. México D.F.

19. **[Jacobson, 2000]** JACOBSON, IVAR; RUMBAUGH, JAMES; BOOCH, GRADY, "El proceso unificado de desarrollo".2000. Addison Wesley. capítulo 9.

20. **[Kepware Technologies, 2015]** KEPServerEX Communications for Automation. Kepware Technologies. [En línea: 2015] [Citado el : 15 de enero 2015] Disponible en *http://www.kepware.com/kepserverex/***.**

21. **[Kominek, 2009]** KOMINEK, DAREK (2009) OPC: ¿De qué se trata y cómo funciona? "Guía para entender la Tecnología OPC". P. Eng. Alberta, Canadá.

22. **[Lcda. Vanessa González, 2015]** LCDA. GONZÁLEZ, VANESSA. Arquitectura del Software 08IST-D01 UNEFA-Apure [Citado el: 15 de marzo 2015] Disponible en: http://arquitecturadelsoftware08.blogspot.com/2014/06/unidad-4-patrones-arquitectonicos.html

23. **[Leitner, 2010]** LEITNER, STEFAN-HELMUT; MAHNKE, WOLFGANG (2010) OPC UA – Service-oriented Architecture for Industrial Applications. ABB Corporate Research Center Wallstadter Str. 59, 68526 Ladenburg, Germany.

24. **[Mahnke, 2009]** MAHNKE, WOLFGANG; LEITNER, STEFAN-HELMUT; DAMM, MATTHIAS (2009) OPC Unified Architecture. Springer-Verlag Berlin Heidelberg. ISBN 978-3-540-68898-3.

25. **[Mesa Valiente & Castillo Pérez, 2008].** R. MESA VALIENTE, I. CASTILLO PÉREZ. (2008). LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Desarrollo de la Base de Datos del Módulo Sección de Mejoramiento de la Calidad.

Trabajo de Diploma. Universidad de las Ciencias Informáticas, Cuba.

26. **[Microsoft, 2010]** Microsoft. Visual C++. MSDN. [En línea] Microsoft, 2010. [Citado el: 1 de Mayo de 2010.] http://msdn.microsoft.com/es-es/library/60k1461a(v=VS.80).aspx.

27. **[OPC Foundation, 2010]** Foundation, OPC. (2010).The OPC Foundation. [En línea] 10 de Enero de 2015 [Citado el: 4 de Junio de 2010.] Obtenido de: http://opcfoundation.org/Default.aspx/01_about/01_history.asp?MID=AboutOPC.

28. **[OPC UA Part 1, 2012]** OPC Foundation. 10 de Julio de 2012. OPC Unified Architecture Specification Part 1: Overview and Concepts. Industry Standard Specification.

29. **[OPC UA Part 8, 2006]** OPC Foundation. 25 de Septiembre 2006. OPC Unified Architecture Specification Part 8: Data Access. Industry Standard Specification.

30. **[OPC UA Part 9, 2010]** OPC Foundation. 9 de Marzo 2010. OPC Unified Architecture Specification Part 9: Alarms & Conditions. Industry Standard Specification.

31. **[OPC UA Part 10, 2007]** OPC Foundation. 29 de Enero 2007. OPC Unified Architecture Specification Part 9: Programs. Industry Standard Specification.

32. **[OPC UA Part 11, 2009]** OPC Foundation. 1 de Diciembre 2009. OPC Unified Architecture Specification Part 9: Historical Access. Industry Standard Specification.

33. **[OPC DataHub®, 2015]** OPC DataHub® - OPC Foundation, [Citado el: 14 de enero 2015] Disponible en *https://opcfoundation.org/products/view/90.*

34. **[Oposiciones TIC, 2011]** Arquitectura cliente servidor - Oposiciones TIC [En línea] 08 de jun. de 2011 [Citado el: 17 de marzo 2015] Disponible en: http://oposicionestic.blogspot.com/2011/06/arquitectura-cliente-servidor.html.

35. **[Patrones de Diseño, 2013]** PATRONES DE DISEÑO: MODELO VISTA CONTROLADOR. [En línea] 07 de jul. de 2013. [Citado el: 11 de marzo 2015] Disponible en: http://thelozu.blogspot.com/2013/07/modelo-vista-controlador.html

36. **[PostgreSQL, 2009]** MARTINEZ, RAFAEL. 2009. PostgresSQL. [En línea] 2009

[Citado el: 15 de Diciembre 2014] Disponible en *http://www.postgresql.org.es/sobre_postgresql.*

37. **[Proyecto Zero, 2010].** FUENTES SÁNCHEZ, JOSÉ ALBERTO; SANTOS OLIVA, MARÍA DE LOS ÁNGELES. Proyecto Zero. Modelado UML. [En línea] 15 de Enero de 2015. Obtenido de: https://forja.rediris.es/docman/view.php/282/444/uml20.pdf.

38. **[Pruebas de Software, 2014].** Pruebas de Software [En línea] 1 jul. 2014. [Citado el: 14 de mayo 2015] Obtenido de: http://es.slideshare.net/miluskaazabachegonzales/pruebas-de-software-36522714.

39. **[qVision, 2014]** Generación y Verificación de Historia de Usuario (SCRUM) [Citado el: 13 de marzo 2015] Disponible en: http://www.qvision.us/es/servicios/gestion-de-requerimientos-de-software/generacion-y-verificacion-de-historia-de-usuario-scrum.

40. **[SCADA VANTAGE, 2003]** ABB. 2003. SCADA Vantage Indistrial Solutions from ABB. Product Guide.

41. **[SDK OPC UA, 2010]** Unified Automation, 2010. C++ Based OPC UA Server SDK. [Citado el: 18 de enero 2015] Disponible en: *http://www.unified-automation.com/products/server-sdk/c-ua-server-sdk.html.*

42. **[UA ANSI C Server Professional, 2011]** UA ANSI C Server Professional, 2011. Motivation for OPC UA. Unified Automation. [Citado el: 17 de Diciembre 2014] [En línea: 2011]; Disponible en: *http://documentation.unifiedautomation.com/uasdkc/1.4.0/html/L2OpcUaMotivation.html.*

43. **[Universidad de Oviedo, 2009].** Sitio web de la E.U. de Ingeniería Técnica Informática de Oviedo. (2009). Entornos de Desarrollo Integrado. [En línea] 01 de Diciembre de 2014. Obtenido de: http://petra.euitio.uniovi.es/Entornos%20de%20Desarrollo%20Integrado.pdf.

Glosario de Términos

A

- **ALBA:** La Alternativa Bolivariana para los Pueblos de Nuestra América o ALBA es una propuesta de integración enfocada para los países de América Latina y el Caribe que pone énfasis en la lucha contra la pobreza y la exclusión social con base en doctrinas de izquierda.

- **Aplicación:** Terminología técnica para referirse a un programa de software.

- **Arquitectura:** Indica la estructura, funcionamiento e interacción entre las partes del software.

C

- **CEDIN:** Siglas que identifican el Centro de Informática Industrial de la facultad 5 de la Universidad de las Ciencias Informáticas.

- **Comunicación:** Proceso mediante el cual se puede transmitir información de una entidad a otra.

F

- **Fiabilidad:** Referido al comportamiento de un sistema o dispositivo, se define como la "probabilidad de que el dispositivo desarrolle una determinada función, bajo ciertas condiciones y durante un período de tiempo determinado".

I

- **IDE:** Siglas en inglés que identifican un Entorno de Desarrollo Integrado (Integrated Development Environment). Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

M

- **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.

S

- **SCADA:** Supervisión Control y Adquisición de Datos (SCADA): Aplicación de software especialmente diseñada para funcionar sobre computadores en el control de producción, proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática desde la pantalla del operador, brindando toda la información asociada al proceso.

- **Seguridad:** Proviene de la palabra securitas del latín. Cotidianamente se puede referir a la seguridad como la ausencia de riesgo o también a la confianza en algo o alguien.

- **Sistema:** Compuesto cuyos componentes se relacionan con al menos algún otro componente, puede ser material o conceptual.

- **Software:** Conjunto de programas, documentos, procesamientos y rutinas asociadas con la operación de un sistema de computadoras, es decir, la parte intangible o lógica de una computadora.

- **Software Libre:** Se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.