UNIVERSIAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

MIGRACIÓN DE LA INTERFAZ HOMBRE – MÁQUINA (HMI) DE UN SISTEMA DE CONTROL DISTRIBUIDO (DCS) DISCONTINUADO A UNA PLATAFORMA DE SOFTWARE ACTUAL

INFORME DE SUFICIENCIA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

VALERIANO HIZO HARO

PROMOCIÓN

2009-II

LIMA-PERÚ

2013

MIGRACIÓN DE LA INTERFAZ HOMBRE – MÁQUINA (HMI) DE UN SISTEMA DE CONTROL DISTRIBUIDO (DCS) DISCONTINUADO A UNA PLATAFORMA DE SOFTWARE ACTUAL

Agradezco a:
Dios, por guiar mis pasos,
Mi familia, por su apoyo incondicional de toda la vida,
Mis profesores, compañeros de estudio y la UNI en general,
por todo lo vivido y aprendido en esta aventura académica
conocida como carrera universitaria.

SUMARIO

En la actualidad existe un gran número de empresas que se dedica al rubro de la automatización industrial utilizando sistemas SCADA, empleando productos de diversas marcas, los cuales ofrecen gran variedad de herramientas personalizadas según la marca, pero que al final cumplen fines similares: Adquirir información de campo, mostrarlos en una interfaz gráfica para el monitoreo respectivo, ejecutar los algoritmos de control y transmitir comandos a campo para controlar los diversos equipos de una determinada planta.

En el presente trabajo se describe la migración de la interfaz Hombre – Máquina (HMI) de un sistema de control distribuido (DCS) a un sistema SCADA cuya plataforma de software es actual. Las razones por las cuales se ha realizado esta migración son principalmente las siguientes: el sistema de control distribuido está actualmente descontinuado, es difícil la adquisición de soporte especializado, la adquisición de repuestos de hardware en caso de daño de los mismos es escaso, la necesidad de contar con un sistema que permita la integración de los subsistemas de la planta en una sola plataforma de software.

Dado este contexto, se plantea los objetivos y alcances del informe, seguidamente se describe el marco teórico de los sistemas SCADA, viendo en más detalle la plataforma en la que se trabaja: "*Wonderware System Platform*".

En el informe se utiliza la aplicación implementada en la plataforma del DCS Bailey de la planta concentradora de una empresa minera ubicada en el departamento de Ancash, a una altitud aproximada de 4200 msnm. Se hace el estudio del caso para la implementación de dicha aplicación y con ello se realizará la réplica de la misma en la nueva plataforma.

En el planteamiento de la solución al problema, se presenta las alternativas de solución, se escoge una de ellas y se elabora el esquema para implementar la réplica de la aplicación inicial. Finalmente se muestra los resultados obtenidos para cada punto y se muestra las conclusiones de la implementación realizada.

INDICE

INTRODUCCIÓN

El empleo de sistemas computacionales para el control y supervisión de procesos industriales ha ido en aumento en el mundo, estos sistemas son comúnmente conocidos como SCADA (Supervisory Control and Data Adquisition - Supervisión, Control y Adquisición de Datos).

El empleo de los sistemas SCADA tiene su justificación en las diversas funcionalidades que ofrece, consiguiendo un ahorro tanto en tiempo como dinero. Ello debido a que con el sistema SCADA se evita la necesidad de tener operadores en las diversas instalaciones de una planta, más aún cuando las instalaciones se encuentran a una gran distancia una de la otra, cientos o miles de Kilómetros (como en la industria de extracción de petróleo), en donde el gasto de hacer visitas rutinarias es apreciable.

El propósito de contar con un sistema SCADA es tener un control centralizado del proceso mediante una estación central, a partir de la cual un operador puede monitorear constantemente el proceso en estaciones remotas, controlarlo a través del envío de comandos al controlador local, como abrir o cerrar válvulas, arrancar una bomba, etc.

Si bien es cierto los sistemas SCADA ofrecen una gran variedad de herramientas, el uso inadecuado de las mismas puede conducir a resultados no deseados, esto es, que la visualización de la información de campo en las páginas gráficas de la estación de operación sea lenta, que los comando enviados a campo se demoren más de lo necesario o que inclusive no sean enviados, lo cual es un problema grave.

En el presente trabajo se hace el estudio de un Sistema de Control Distribuido (DCS) discontinuado (Bailey) con el objetivo de migrar su Interfaz Hombre Maquina (HMI) a una nueva plataforma de software, la cual es una plataforma vigente y en constante desarrollo. A través del HMI se realiza el control y supervisión del proceso industrial que el DCS Bailey administra. Cabe remarcar que uno de los objetivos de esta migración es contar con una sola plataforma de software para monitorear todos los subsistemas de la planta (independiente de la fuente de datos), esto es posible debido a que la nueva plataforma de software tiene características de comunicación y redes "optimizadas para SCADA" [1], y cuenta con una gran escalabilidad (desde 250 hasta más de 1 millón de conexiones I/O). Se debe tener en cuenta que uno de los requisitos que debe cumplir la nueva plataforma es contar con la funcionalidad de redundancia de servidores. Una vez

hecho el estudio y terminada la implementación se espera que la funcionalidad del nuevo sistema de HMI responda de tal manera que cumpla con las exigencias del usuario final, esto dicho en palabras del mismo, es que para tareas similares la respuesta del nuevo sistema debe desviarse lo menos posible del sistema precedente. También se debe verificar la funcionalidad de la redundancia.

Para este caso la aplicación que se analiza le pertenece a una empresa minera situada en el distrito de San Marcos, departamento de Ancash. Específicamente se verá el HMI que se usa en el área de la planta concentradora.

En el presente trabajo se usa información general que se puede encontrar en internet, información recopilada de acuerdo a la experiencia del autor, cierta información de informes de suficiencia relacionados y manuales del fabricante. Además se muestra información recopilada mediante la ejecución de un proyecto con la empresa Invensys Process Systems del Perú S.A.

La tecnología a usar es la plataforma de la marca Wonderware, conocida como "*Wonderware System Platform*", el cual es un sistema que tiene la escalabilidad, estabilidad y desempeño para manejar las instalaciones industriales más grandes.

El trabajo está organizado en cuatro capítulos principales

Capítulo I: "Planteamiento de ingeniería del problema".- Se establece la necesidad y el objetivo del trabajo a realizar, precisando los alcances.

Capítulo II: "Marco teórico conceptual".- Conformada principalmente por 3 partes: Sistemas Scada - la plataforma de Wonderware, Sistemas de Control Distribuido -el DCS Bailey, Protocolos de comunicaciones industriales – OPC Rovisys.

Capítulo III: "Metodología para la solución del problema". – Topología del sistema de control Bailey, estudio y evaluación de la solución a proponer, implementación de la solución.

Capítulo IV: "Análisis y Presentación de Resultados". Análisis descriptivo, análisis teórico, presupuesto y tiempo de ejecución.

CAPÍTULO I
PLANTEAMIENTO DE INGENIERÍA DEL PROBLEMA

En este capítulo se realiza el planteamiento del problema, para lo cual primero se describe el problema, luego se fijan los objetivos del trabajo, se analiza y se propone una solución de ingeniería. Una vez propuesta una solución se fijará los alcances del trabajo. Finalmente se hace una síntesis del presente informe.

1.1 Descripción del problema

- Discontinuidad del Sistema de Control Distribuido original, tanto de hardware como de software. La disponibilidad de partes y recursos es cada vez más difícil de conseguir hasta que en un caso extremo uno de los dos puntos mencionados (partes o recursos) desaparece. Esto es, el hardware que este sistema usa no soporta los nuevos sistemas operativos basados en Windows (como el Windows 7 o el Windows Server 2008), además no soporta comunicaciones con protocolos modernos.
- Difícil y costosa adquisición de soporte técnico especializado. Debido a su obsolescencia, este producto ya no es vendido y ello conlleva a que los ingenieros se capaciten en sistemas actuales y dejen de lado este sistema. Por ello cada vez son menos los que conocen del sistema para que puedan dar soporte técnico especializado.
- Escaso hardware usado para mantenimiento.
- La plataforma de software del Bailey es una plataforma con bajo nivel de integración, soporta solo unos pocos protocolos de comunicación, protocolos modernos no son soportados (como el Foundation Fieldbus o Profibus) salvo con Modbus TCP y RTU. [3] Debido a esta limitante, la plataforma del Bailey no es recomendable en cuanto a integración se refiere.

1.2 Objetivos del trabajo

- Plantear una arquitectura de red que permita implementar la migración de la interfaz hombre máquina (HMI) de un DCS descontinuado (DCS Bailey) a una plataforma de software actual ("*Wonderware System Platform*"). Esta arquitectura de red deberá soportar redundancia de servidores.
- Probar la funcionalidad del HMI implementado en la plataforma de software actual y hacer una comparación con el HMI del sistema original.
- Verificar la funcionalidad de la redundancia implementada.

1.3 Evaluación del problema

El avance rápido de la tecnología fuerza a las empresas desarrolladoras de sistemas a que actualicen e innoven sus productos tanto de hardware como de software y así mantener su competitividad y vigencia en el mercado. El no avanzar al paso que lo hace la tecnología o más aún serle indiferente significará una obsolescencia con la inevitable salida de circulación del mercado.

En el presente informe se tiene un sistema de control (Sistema de Control Distribuido Bailey de ABB) funcionando y operando una planta de tratamiento de minerales, sin embargo el desarrollo de este sistema se quedó estático a finales de los 90, con lo cual vino su posterior obsolescencia, por lo que hoy en día los softwares que este sistema usa están por salir de circulación del mercado (por ejemplo este sistema podía ejecutarse sobre Windows XP de 32 Bits, para el cual Microsoft dejará de dar su servicio de soporte extendido el 8 de abril de 2014), los nuevos sistemas operativos basados en Windows son el Windows 8 para el caso de Estaciones de trabajo y el Windows Server 2012 para el caso de servidores, los cuales no son soportados por el sistema Bailey, más por el contrario, en las plataformas de software actuales nuevas versiones de software son creadas para soportar las últimas versiones de sistemas operativos de Windows. Además, para el hardware que usa (como tarjetas de entrada - salida) actualmente ya no hay línea de producción por parte del fabricante. Esto unido al bajo nivel de integración que tiene el sistema Bailey, hacen que no sea posible integrar otros sistemas de control en su HMI. También, se tiene que como parte de su plan de expansión, el usuario final (empresa minera) tiene la necesidad de contar con una plataforma de software que centralice o integre todos los subsistemas de la planta. Para ello, el primer paso de su plan de integración es migrar el sistema Bailey, y que a su vez como primer paso para ello es migrar el HMI del Bailey a una nueva plataforma de software.

Lo que se propone en el presente informe es el uso de la plataforma de software de la marca Wonderware como sistema integrador, esto debido a la capacidad de integración con la que este sistema cuenta. Cabe remarcar que el objetivo principal de este informe es hacer la migración del HMI del sistema de control Bailey a la plataforma de Wonderware, lo cual en otras palabras es hacer la integración del sistema ya implementado en el DCS Bailey a la nueva plataforma.

1.4 Alcance del trabajo

El presente trabajo tiene el siguiente alcance:

- Hacer el estudio de los bloques de control de un DCS en particular para identificar sus principales parámetros.
- Presentar y evaluar las alternativas de implementación en la nueva plataforma.

- Implementar la solución elegida, esto es, elaborar y configurar la base de datos necesaria para cumplir con la funcionalidad adecuada.
- Implementar un total de 5 páginas gráficas (representaciones gráficas de sectores de la planta industrial) a través de los cuales se hará la supervisión y control de la planta.
- Probar el funcionamiento de la nueva plataforma y mostrar los resultados obtenidos.

1.5 Síntesis del trabajo

El presente trabajo consta principalmente en tres partes, la primera referente al marco teórico, la segunda referente a la solución de ingeniería que se propone y la tercera referente al análisis y presentación de resultados.

Marco teórico: Se da una descripción de la tecnología que se usa en el control industrial, se da una visión al software que se pretende reemplazar, se enuncia sus principales componentes. También se da una descripción de la plataforma de software con la que se trabaja para la implementación de la solución.

Solución del problema: En esta etapa se presentan y se evalúan las posibles soluciones. Se escoge una y se explica el método para su implementación, además se detalla los recursos a emplear (Hardware y Software).

Análisis y presentación de resultados: Se muestra los resultados obtenidos con la implementación realizada, se hace un análisis de dichos resultados y finalmente se dan las conclusiones y recomendaciones.

CAPITULO II
MARCO TEÓRICO CONCEPTUAL

2.1 Sistemas SCADA

SCADA de Supervisory Control and Data Adquisition (Supervisión, Control y Adquisición de Datos), es un sistema computacional que se compone de hardware y software, los cuales son configurados con la finalidad de adquirir datos de equipos que se encuentran en ubicaciones remotas para mostrarlos en una Interfaz Hombre Maquina (HMI – Human Machine Interface), a través de la cual un operador puede hacer una supervisión y control de una planta o proceso industrial.

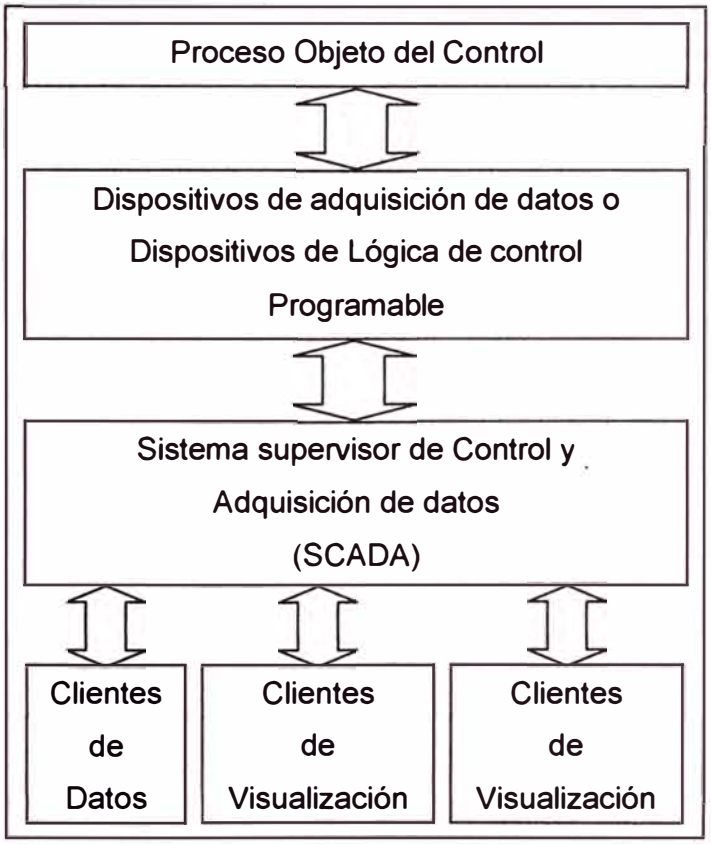

Figura 2.1 Esquema de un sistema SCADA

Los equipos en ubicaciones remotas son denominados Unidades de Terminal Remota (RTU – Remote Terminal Unit) los cuales tienen la tarea de hacer el control y adquisición de datos hacia y desde el campo. La estación central que tiene la Interfaz Hombre Maquina es denominada estación maestra o Unidad Terminal Maestra (MTU – Master

Terminal Unit) y es la encargada de mostrar la información al operador, incluyendo el control de lazo abierto, registro de datos históricos, generación de alarmas y eventos de proceso y de sistema, además de tener implementado un sistema de seguridad para la operación desde el HMI. Para que el control y la adquisición de datos puedan darse es necesaria la implementación de una infraestructura de comunicación, lo cual conforma el medio físico que conecta las muchas terminales remotas y las estaciones en el sistema.

Proceso objeto del control: Es el proceso que se desea supervisar. En consecuencia es el origen de los datos que se requiere colectar y distribuir.

Adquisición de datos: Son un conjunto de instrumentos de medición dotados de alguna interfaz de comunicación que permita su interconexión.

SCADA: Combinación de hardware y software que permite la colección y visualización de los datos proporcionados por los instrumentos.

Clientes: Conjunto de aplicaciones que utilizan los datos obtenidos por el sistema SCADA. [2]

2.2 La Plataforma Wonderware

"*Wonderware System Platform*" provee una única plataforma para todas las necesidades de la automatización industrial. En el centro del "*Wonderware System Platform*" está el "modelo de la planta", el cual es la representación lógica de los procesos y equipos físicos que son controlados y supervisados. El sistema cuenta con un historiador de alto rendimiento con almacenamiento de historia de producción, compresión eficiente de datos, autoconfiguración de almacenamiento histórico que ayuda eliminar un doble esfuerzo, y un servidor de gestión de información industrial vía Web (Industrial Web Information Server) que simplifica dramáticamente la organización y presentación de información de operaciones para su uso a través de todas las funciones en una organización, esto permite rápidamente evaluar problemas e identificar oportunidades de mejoras de proceso. [1]

"*Wonderware System Platform*" provee una plataforma de servicios de aplicación industrial común y estratégica encima de virtualmente cualquier sistema existente, y está construido sobre la base de estándares industriales, la tecnología ArchestrA real – time SOA (Service Oriented Architecture).

El "*Wonderware System Platform"* contiene un grupo básico de capacidades y servicios para soportar producción sostenible y mejoramiento en el desarrollo de las operaciones a través de un grupo de áreas de 6 capacidades: [4]

- Servicios de dominio industrial.
- Servicios de conectividad a dispositivos y software.
- Servicios de manejo de datos e información.

- Servicios de desarrollo de aplicaciones.
- Servicios de extensibilidad y manejo de sistema.

A continuación se describe los principales componentes del "*Wonderware System Platform*".

2.2.1 Servidor de Aplicaciones de Wonderware

El Servidor de Aplicaciones de Wonderware (Application Server), es formalmente conocido como Industrial Application Server, provee el marco para el desarrollo de aplicaciones para sistemas amplios, adquisición de datos en tiempo real, administración de alarmas y eventos, seguridad centralizada, manipulación de datos, descargas remotas e ingeniería colaborativa.

Kit de herramientas ArchestrA Object: Se usa para crear objetos de aplicación (ApplicationObjects) para su uso dentro del Wonderware Application Server.

El Wonderware Application Server provee un entorno de desarrollo integrado de ArchestrA – Archestra Integrated Developement Environment (IDE)

a) Concepto de Galaxia

Una Galaxia (Galaxy) representa el entorno de producción completo, incluyendo todas las computadoras y componentes que ejecutan la aplicación. Una Galaxia es una colección de Plataformas, AppEngines, instancias y atributos que se definen como parte de una aplicación especifica. La información sobre esta colección de objetos es almacenada en una base de datos de la Galaxia.

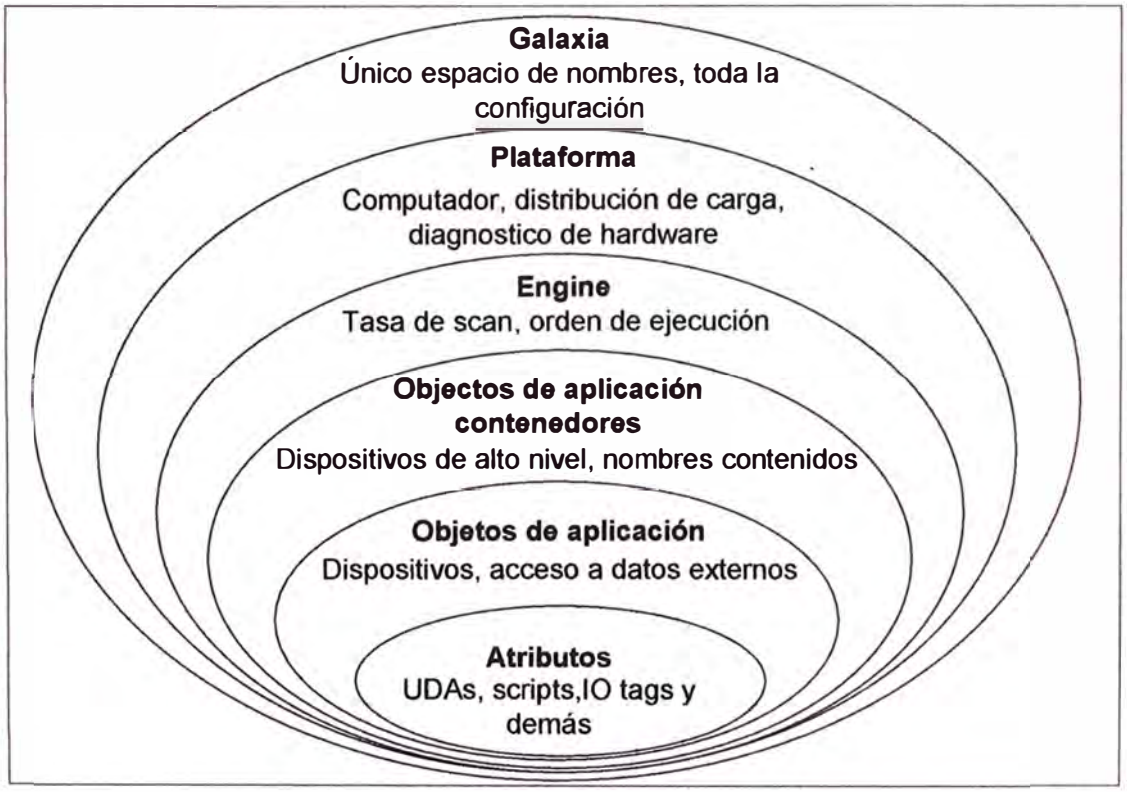

Figura 2.2 Jerarquía de elementos de una Galaxia

La base de datos de una galaxia reside en una única computadora de la red. La base de datos de la galaxia puede residir en cualquier computadora en la red, con los software SQL Server, Bootstrap (servicio del "*Wonderware System Platform*") y el Repositorio de Galaxia instalados. No se puede almacenar partes de la base de datos de una galaxia en varias computadoras.

Un Repositorio de Galaxia (GR) es el nombre de la única computadora donde la base de datos de la galaxia está localizada. Se puede descargar componentes de una galaxia como WinPlatforms y AppEngines, en múltiple computadoras para compartir la carga de trabajo mientras la aplicación se está ejecutando.

El espacio de nombres de una Galaxia es el juego de identificadores únicos de objetos y atributos. El espacio de nombres y los valores de cada uno de sus identificadores definen una aplicación del Wonderware Application Server.

Las galaxias también incluyen seguridad, el cual esta deshabilitado por defecto. Usando la seguridad permite limitar lo que los usuarios pueden hacer.

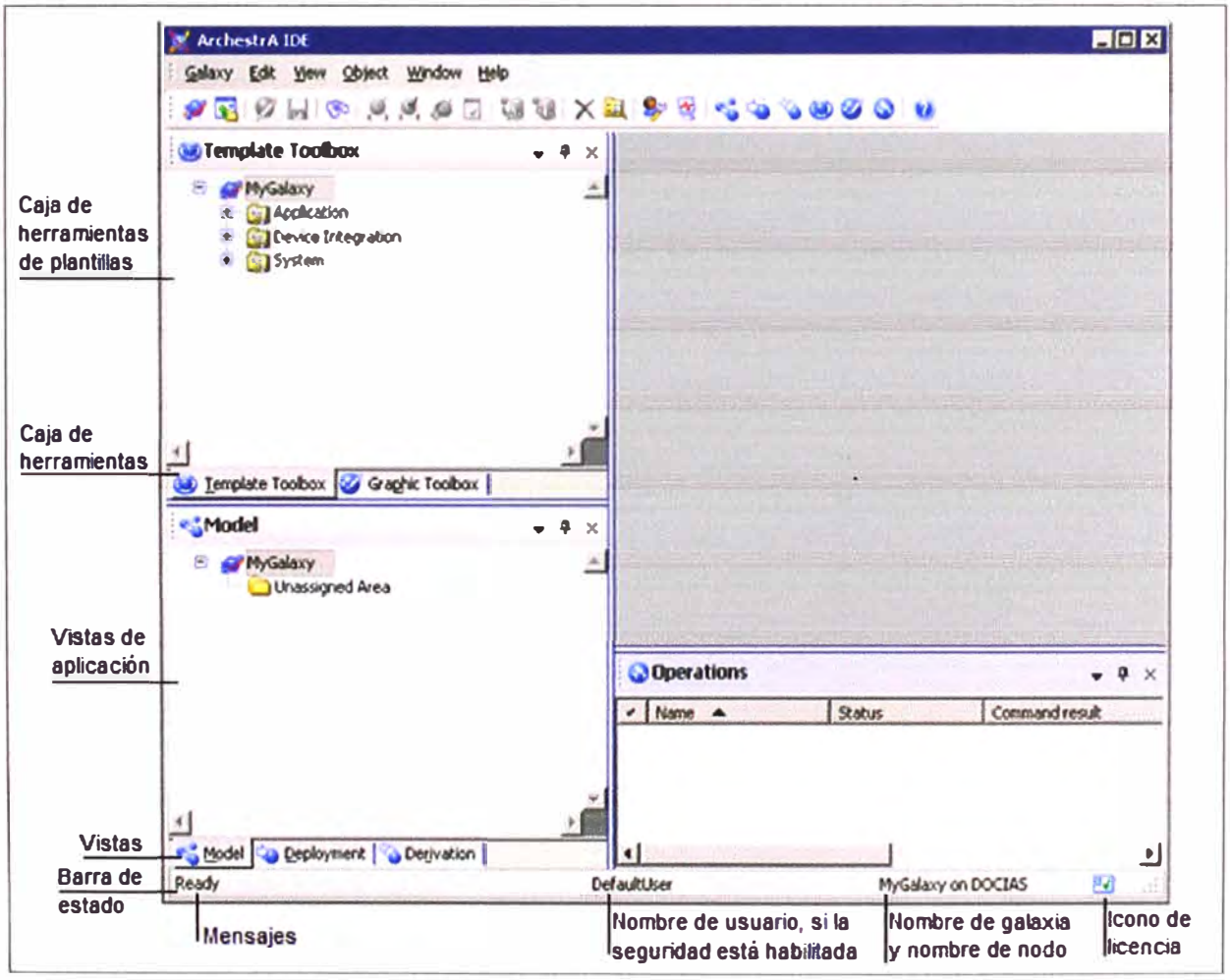

Figura 2.3 Vista del Entorno de Desarrollo Integrado – ArchestrA IDE

b) Objetos de Automatización

Los Objetos de Automatización o Automation Objects permiten la encapsulación de todos

los elementos de configuración para cada pieza del sistema, como la definición de I/Os, lógica (scripting), configuración histórica, configuración de alarmas y eventos, gráficos y control de acceso y seguridad. Esta aproximación contenida en sí misma reduce dramáticamente el tiempo de ingeniería asociada con la creación inicial y mantenimiento de objetos. Manteniendo toda la configuración del objeto relacionada estrechamente y contenida dentro del objeto mismo, no hay necesidad de usar múltiples editores para asegurarse que el alarmado, definiciones de I/Os, scripting, historización y seguridad sean consistentes para un objeto.

Se dispone de objetos Template (plantillas) y objetos Instance (instancias):

Objetos Plantillas: Estos son definiciones de alto nivel de los objetos: equipos, dispositivos o simplemente partes del sistema de la Galaxia.

Objetos Instancias: estos son los objetos en tiempo de ejecución y representan los items específicos del entorno, como procesos, válvulas, tanques y así.

También se dispone de objetos de dominio (Domain objects) y objetos de sistema (System objects)

Objetos de Dominio:

Objetos de aplicación (Application objects): representan el equipo físico o construcciones lógicas en la Galaxia.

Objetos de integración de dispositivos (Device Integration objects): representan la comunicación con dispositivos externos.

Objetos de sistema (System objects): representan partes de una Galaxia y no el dominio el cual están monitoreando y/o controlando.

Atributos y Referencias de Atributos

Cada pieza de e información disponible dentro de un objeto es llamado un atributo. La interacción con objetos, en configuración o en tiempo de ejecución es hecha a través de los atributos del objeto específico.

Las referencias de los atributos se refieren a la información dentro del atributo de un objeto. Consiste en una cadena o string de referencia del objeto más una cadena o string de referencia del atributo, separado por un punto ("."). Por ejemplo:

ObjectName.AttributeName. [4]

Categorías de Objetos

Dentro del Template Toolbox hay tres principales categorías de objetos. Estos objetos son:

- Application objects
- AnalogDevice
- Boolean

- DiscreteDevice
- Double
- FieldReference
- Float
- Integer
- Sequencer
- SQLData
- String
- Switch
- UserDefined

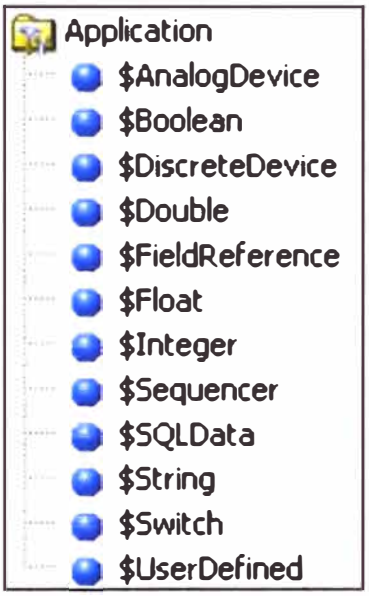

Figura 2.4 Visualización de los Application objects en el AchestrA IDE

- Device Integration objects
- DDESuiteLinkClient
- InTouchProxy
- OPCClient
- RedundantDIObject

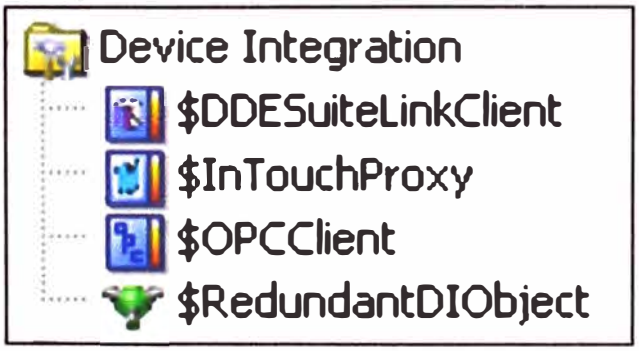

Figura 2.5 Visualización de los Device Integration objects en el AchestrA IDE

- System objects
- AppEngine
- Area
- InTouchViewApp
- ViewEngine
- WinPlatform

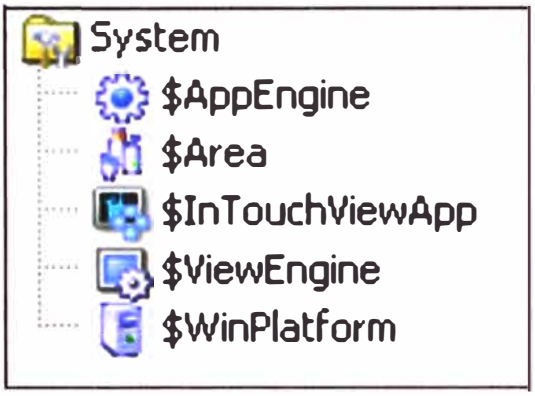

Figura 2.6 Visualización de los System objects en el AchestrA IDE

2.2.2 Software para la Interfaz Hombre-Máquina

Wonderware InTouch software es una interfaz hombre-máquina (HMI) para visualización y control de procesos. Toma el manejo de las operaciones, control y optimización a un nivel completamente nuevo. La reputación del HMI InTouch está por encima de todos los demás.

Una aplicación de software de Interfaz Hombre Maquina (HMI) muestra una representación gráfica del entorno de manufactura. Las herramientas, materiales, y procesos usados para crear un producto aparecen como elementos visuales en las ventanas de aplicación HMI. Los operadores de planta interactúan con una interfaz de aplicación gráfica para monitorear y administrar el proceso de manufactura.

Como su nombre sugiere, se usa el Application Manager para crear y administrar aplicaciones InTouch. La aplicación de entorno de desarrollo, llamado WindowMaker, incluye un juego de gráficos y otras herramientas de desarrollo para crear aplicaciones. Las aplicaciones se ejecutan usando WindowViewer. [5]

2.2.3 Software de Historización de Wonderware

El Historian es una base de datos relacional de tiempo real que almacena datos de plantas industriales. Historian adquiere y almacena datos de procesos industriales en resolución completa o a una resolución especificada y provee datos en tiempo real o histórica, junto con datos de configuración, eventos, resúmenes, y otros datos asociada a la producción, a las aplicaciones cliente en un computador de escritorio. El Historian combina el poder y la flexibilidad de Microsoft SQL Server con la alta velocidad de

adquisición y eficaz compresión de datos características de un sistema de tiempo real.

a) Dato de proceso

El dato de proceso es cualquier información relevante para ejecutar un proceso satisfactoriamente. La siguiente información es considerada a ser dato de proceso:

Datos de tiempo real - ¿Cuál es el valor actual de este Tag?

Datos históricos - ¿Cuál es el valor de este Tag cada segundo del último Lunes?

Datos resumen - ¿Cuál es la media de cada uno de estos cinco Tags?

Datos de eventos - ¿Cuándo hubo una parada de emergencia automática en la caldera?

Datos de configuración - ¿Cuántos servidores I/O (de entrada/salida) se está usando y cuáles son sus tipos?

Para mejorar el desempeño y calidad mientras se reduce los costos, los datos de proceso deben estar disponibles para análisis. La información del proceso es típicamente analizado para determinar:

- Análisis de proceso, diagnostico y optimización.
- Mantenimiento predictivo y preventivo al equipamiento.
- Calidad de producto y proceso.
- Salud y seguridad; impacto medioambiental.
- Reporte de producción.
- Análisis de fallas.

b) Wonderware Historian - base de datos relacional de tiempo real

Como una base de datos de tiempo real, el Wonderware Historian es una extensión al Microsoft SQL Server, proporcionando más de un orden de magnitud en el incremento de la velocidad de adquisición, una correspondiente reducción en el volumen de almacenamiento, y extensiones elegantes al lenguaje de consultas estructuradas (SQL) para consultar datos en series de tiempo.

Captura de datos en alta velocidad: La amplia gama de servidores I/O y servidores de adquisición de datos (Data Access Server - DAServer) de Wonderware son usados para conectarse a más de 500 dispositivos de control y adquisición de datos. El Historian adquiere y almacena datos de proceso muchas veces más rápido que un sistema de manejo de base de datos relacionales (Relational Data Base Management System - RDBMS).

Reducido espacio de almacenamiento: El Wonderware Historian almacena datos en una fracción del espacio requerido por una base de datos relacional normal.

Extensiones en dominio de tiempo al SQL: Microsoft SQL Server soporta sus propias extensiones al lenguaje SQL, llamado Transact-SQL. El Wonderware Historian extiende aun más el Transact-SQL, permitiendo el control de resolución y proveyendo la base para

funciones relacionadas en el tiempo como tasas de cambio y cálculo de procesos en el servidor.

c) Subsistemas del Wonderware Historian

El Wonderware Historian está hecho de subsistemas especializados, los cuales trabajan juntos para administrar la información como sea adquirida o generada, almacenada y requerida, como sigue:

- Subsistema de configuración
- Subsistema de adquisición de datos
- Subsistema de almacenamiento de datos
- Subsistema de requerimiento de datos
- Subsistema de eventos
- Subsistema de replicación

2.2.4 Cliente de datos históricos de Wonderware

El software Wonderware Historian Client provee un número de herramientas cliente para direccionar una representación específica de información y requerimientos de análisis. Estas herramientas remueven el requerimiento de familiarizarse con el SQL y provee interfaces intuitivas para acceder, analizar y graficar información actual e histórica, ambas adquiridas en series de tiempo.

Un operador, un ingeniero de proceso o un gerente, a través del Wonderware Historian Client puede organizar, explorar, analizar, presentar, y diseminar la información de proceso en una amplia gama de formatos. Todo esto puede ser hecho desde una computadora de escritorio.

El Wonderware Historian Client se integra estrechamente con las más populares herramientas de Microsoft office. Con el Wonderware Historian Client se puede:

- Explorar la información gráficamente para encontrar información importante.
- Analizar la información para producir información relevante.
- Desarrollar y ejecutar consultas ad hoc contra cualquier información almacenada en la base de datos del Wonderware Historian.
- Visualizar el estado actual del proceso.
- Producir importantes reportes automáticos.

a) Aplicaciones de escritorio

El Wonderware Historian Client incluye las siguientes aplicaciones independientes:

Wonderware Historian Client Trend: Permite hacer tendencias sobre el tiempo de información histórica y de tiempo real. Tiene poderosas herramientas que hacen que la información sea comparada con otras de diferentes periodos. Alarmas y traspaso de límites son fácilmente visibles. Es posible adicionar y ver anotaciones en las tendencias.

Wonderware Historian Client Query: Esta herramienta de apuntar y hacer click (point and click) permite la creación y ejecución de consultas complejas con cualquier Wonderware Historian. No requiere conocimiento de la estructura de la base de datos o SQL.

2.2.5 Componentes de la topología para el System Platform

Una Galaxia abarca el todo el sistema de control y supervisión, el cual está representado por un único espacio de nombres lógico y una colección de WinPlatforms (WinPlatforms son objetos que representan los nodos del sistema, pueden ser servidores o estaciones de trabajo), AppEngines (AppEngines son objetos que contienen y donde se ejecutan los objetos de aplicación) y objetos de aplicación. La galaxia define el espacio de nombres en el cual todos los componentes y objetos residen.

Debido a su naturaleza distribuida y servicios comunes, el Industrial Application Server no requiere computadores costosos de tipo servidores o fault-tolerant para habilitar una aplicación industrial robusta. Fault-tolerant significa tolerancia a fallas, y es una característica que define un nivel de redundancia de servidores.

El Industrial Application Server distribuye objetos a través de un entorno distribuido (en red) permitiendo a una única aplicación ser separada en un número diferente de objetos componentes, cada uno de los cuales puede ejecutarse en una computadora diferente.

Los componentes principales de la topología son:

- Repositorio de Galaxia (Configuración de la Base de Datos)
- Nodo Servidor de AutomationObjects (AOS)
- Nodo de visualización
- Nodo Servidor de I/Os
- Nodo Estación de Ingeniería
- Nodo Historian
- Portal Information Server (Nombre anterior: SuiteVoyager Portal)

Repositorio de Galaxia (Configuración de la Base de Datos): El Repositorio de Galaxia (también llamado Configuración de la Base de Datos o "GR") puede ser instalado en un nodo dedicado o en la Estación de Ingeniería.

El Repositorio de Galaxia administra la configuración de la información asociada con una o más Galaxias. Esta información es almacenada en bases de datos individuales, una para cada Galaxia en el sistema. El Microsoft SQL Server 2008 SP1 (Standard Edition) es la base de datos relacional usada para almacenar la información en la aplicación a mostrar en el presente informe.

Se accede al Repositorio de Galaxia cuando los objetos en la base de datos son visualizados, creados, modificados, borrados o descargados

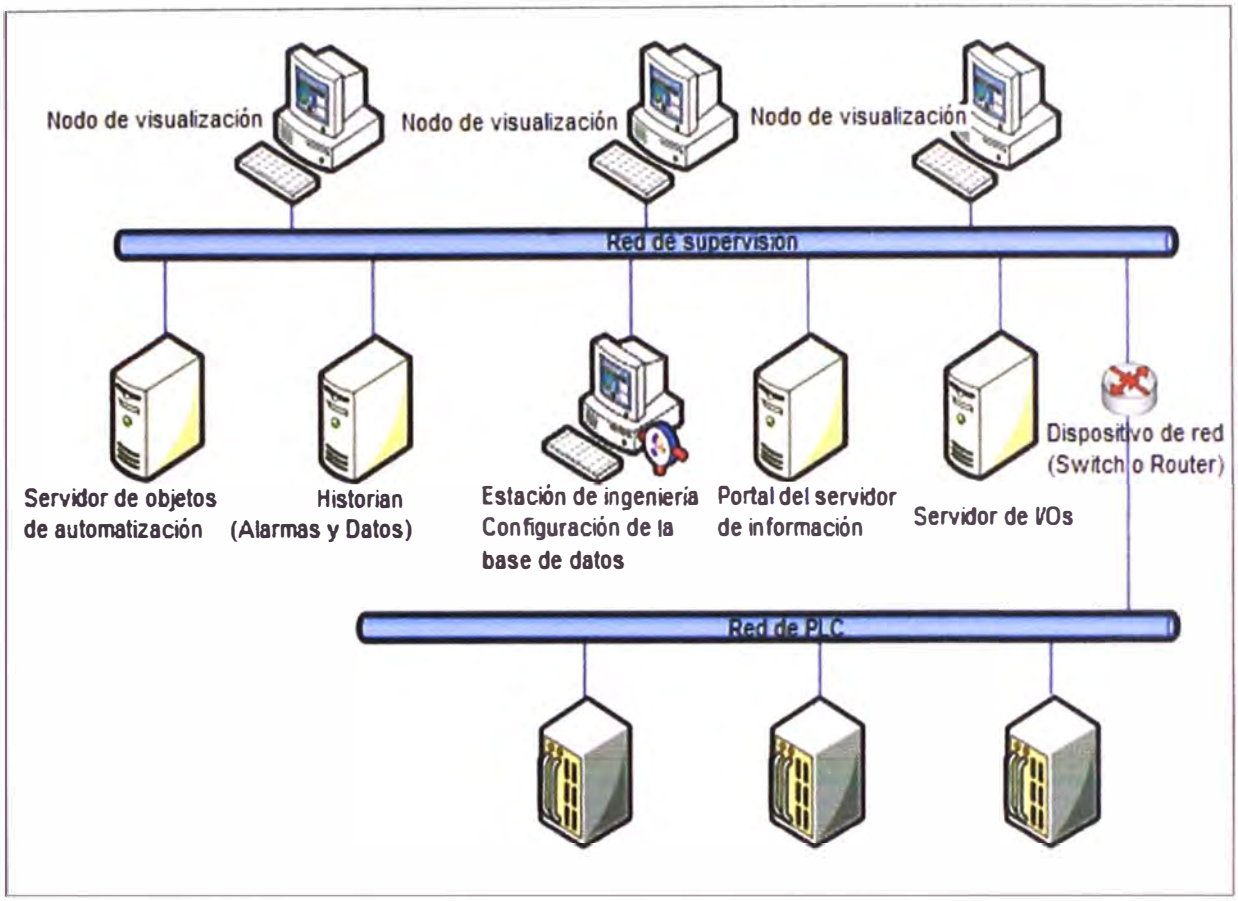

Figura 2.7 Componentes de la topología del Wonderware System Platform.

Nodo Servidor de Objetos de Automatización (AOS): Provee el recurso de procesamiento para AppEngines, Areas, Objetos de Aplicación y Objetos de DeviceIntegration (DI). El nodo Servidor de AutomationObjetcts requiere una plataforma para ser descargada en ella.

Nodo de Visualización: Un nodo de visualización es un computador ejecutando el Software InTouch encima de una Plataforma (Platform). La Plataforma provee comunicación con cualquier otro componente de Galaxia vía el protocolo Message Exchange (MX).

Nodo de Servidor de I/Os: Un nodo de Servidor de I/Os funciona como una fuente de datos para el sistema Industrial Applicatior Server. Los Servidores de I/O se comunican con dispositivos externos. Los protocolos de comunicación soportados son: Dynamic Data Exchange (DDE), SuiteLink™ and OPC.Los DIObjects en el lado del Industrial Application Server administran la comunicación entre los AutomationObjects y los Servidores de I/Os. Los DIObjects requieren de una Plataforma y un AppEngine. Estos objetos pueden residir en bien en el nodo Servidor de I/Os o en cualquier nodo Servidor de AutomationObjects en la Galaxia.

Nodo de Estación de Ingeniería: Todas las topologías recomendadas incluyen un nodo de Estación de Ingeniería dedicado para propósitos de mantenimiento. El nodo de

Estación de Ingeniería contiene el System Platform Developmente Package (paquete desarrollador del System Platform) el cual consiste de componentes tales como el Integrated Development Environment (IDE) y el InTouch WindowMaker. El Repositorio de Galaxia también puede residir en este nodo. Así, se puede implementar cualquier cambio requerido en el sistema desde el nodo de Estación de Ingeniería usando el IDE, el cual accederá a la configuración de la base de batos local.

Nodo de Historian: El nodo de Historian es usado para ejecutar el software IndustrialSQL Server Historian. El IndustrialSQL Server Historian almacena toda la información de proceso histórica y la provee en tiempo real a las aplicaciones clientes del "*Wonderware System Platform*" como los softwares Historian Client e Information Server. El nodo Historian no requiere una Plataforma. El servidor de Automation Objects inserta la información (configurada para historización) al nodo de Historian usando el Manual Data Adquisition Service (MDAS) empaquetado con el Industrial Application Server y el IndustrialSQL Server Historian.

Portal del Servidor de Información (Information Server): Una máquina servidor con un portal del software Wonderware Information Server puede ser incorporado en cualquier Galaxia.

Se requiere que una plataforma sea descargada para el Portal Information Server, así este software pueda acceder a la Galaxia.

2.2.6 Ventajas del Wonderware System Platform sobre otras plataformas

Las arquitecturas de software basadas en objetos han estado presentes en el mundo computacional comercial por muchos años. Ahora (2012 como año de referencia) estas arquitecturas están siendo aplicadas en control de procesos y aplicaciones SCADA para entregar beneficios significativos en operación y costos. [6]

Sistemas basados en tags

Desde el comienzo de sistemas supervisados y HMI basados en PC, procesamiento de acceso a información, desarrollo de scripts, alarmas y análisis de información han sido basados en el concepto de tags. Estos sistemas usan un lista "plana" de tags con jerarquías, interdependencias o relaciones incorporadas.

Cambios globales y grandes a la base de datos de un sistema de tags son hechos a menudo externamente a la aplicación, frecuentemente vía un archivo de texto o a través de herramientas como Microsoft Excel. Una vez hechos los cambios estos son importados a la base de datos de la aplicación.

El mantenimiento a aplicaciones basadas en tags típicamente involucra un análisis y actualización tag por tag que puede consumir una cantidad significante de trabajo. Puesto que los cambios en el sistema demoran y frecuentemente involucran mano de obra

externa, las mejoras a un sistema basado en tags son limitadas.

Sistemas basados en objetos

El concepto de desarrollo orientado a objeto fue originado en el mundo de la tecnología de la información (TI). El objetivo era proveer herramientas que pudieran liberar al desarrollador de tareas de programa repetitivas y tediosas, mientras que al mismo tiempo se maximice la reutilización de código a través del desarrollo de objetos de software comunes.

Estas herramientas no encajan exactamente en el ambiente industrial. Por una cosa, los integradores de sistemas y los ingenieros de producción no son típicamente programadores de computador. Además, hay algunas diferencias de arquitectura claves entre TI y aplicaciones de automatización y producción. Los dos entornos son lo suficiente diferentes para dictar que las herramientas basadas en objetos son deliberadamente construidas para el entorno industrial. El ArchestrA System Platform usa una arquitectura basada en objetos la cual es llamada ArchestrA. Está diseñada específicamente para clientes industriales quienes desarrollan, administran y dan mantenimiento a sistemas de supervisión.

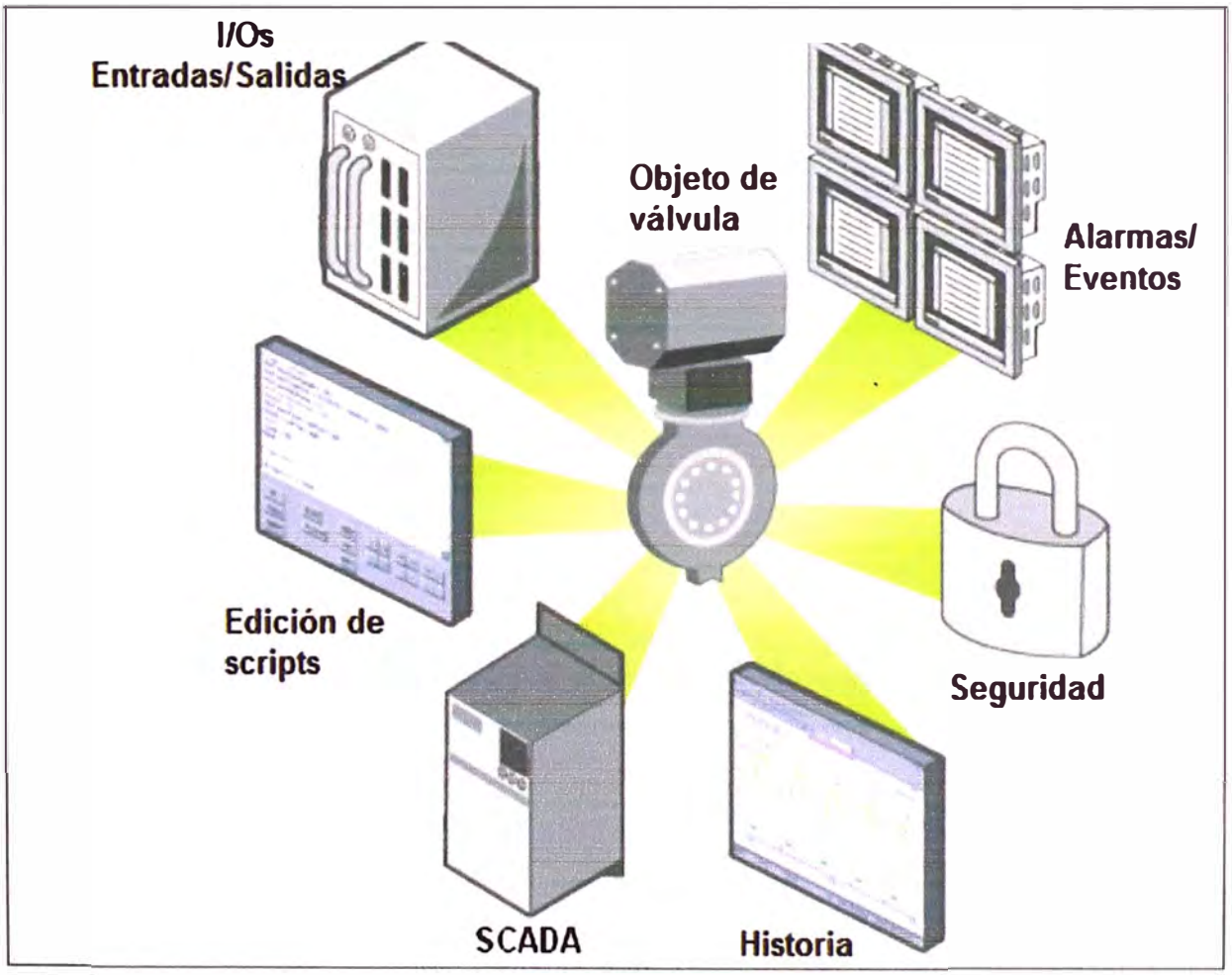

Figura 2.8 Sistema basado en objetos

La siguiente tabla contrasta las arquitecturas basadas en objetos y basadas en tags.

TABLA N° 2.1: Contrastación de arquitectura basadas en objetos y basadas en tags

	Arquitectura basada en objetos		**Arquitectura basada en tags**	
	Desarrollo	**T. de ejecución**	**Desarrollo**	**T. de ejecución**
Estructura de la aplicación	Jerárquico - Los objetos son creados usando una metodología de flujo de trabajo orientado a objetos	Jerárquico - Los componentes representan dispositivos físicos y pueden coordinar con componentes en computadores diferentes	Jerárquico - Contenido gráfico a veces creado usando orientación a objetos	Plano - Instancias compactas de software se ejecutan en una o múltiples máquinas como "aplicaciones" separadas
Desarrollo de gráficos	Se realiza al último	N/A	Se realiza al inicio	N/A
Creación de scripts	Desarrollado en objetos plantillas y luego descargadas a aplicaciones específicas en tiempo de ejecución	N/A	Desarrollado separadamente, ligado a una interfaz gráfica	N/A
Estándares	Aplicado estrictamente	N/A	No aplicado estrictamente	N/A
Cambios en la aplicación	Propagado desde los objetos plantillas	Los objetos pueden ser distribuidos, intercambiados o mejorados	Basados en gráficos o cambiados usando herramientas como Excel	Requiere re-compilación de la aplicación
Representación de la información	Construcciones lógicas tales como dispositivos físicos (por ejemplo: válvulas o bombas) o dispositivos lógicos (por ejemplo: lazos PID o cálculos) son representados como objetos	N/A	Dispositivos gráficos son representados como objetos o tags	N/A

Ahorros en el ciclo de vida del sistema

Las arquitecturas basadas en objetos pueden proveer un ahorro significativo durante todo su ciclo de vida. Estos ahorros pueden ser categorizados en cuatro áreas básicas: Ahorros en desarrollo inicial relacionado a la generación de la aplicación, Ahorros en desarrollo inicial relacionado a los cambios de aplicación, Ahorros en mantenimiento a

través del ciclo de vida del sistema, Ahorros a través de todas las localidades. Estas áreas básicas se muestran en la siguiente tabla:

TABLA N° 2.2: Ahorros en tiempo de sistemas basados en objetos

Área de ahorro	Explicación
Ahorros en desarrollo inicial relacionado a la generación de la aplicación	Resultan del tiempo ahorrado cuando los usuarios desarrollan las aplicaciones definiendo objetos plantillas una vez y luego usando estas plantillas muchas veces.
Ahorros en desarrollo inicial relacionado a los cambios de aplicación	Estos representan los ahorros en el desarrollo ganados a través de la habilidad de propagar los cambios desde los objetos plantillas a todas las instancias en tiempo de ejecución derivadas de esas plantillas. Cuando se piden múltiples cambios en la aplicación durante el desarrollo, los ahorros en tiempo pueden ayudar en gran medida.
Ahorros en mantenimiento a través del ciclo de vida del sistema	Usando un sistema distribuido reduce significativamente los costos de mantenimiento a través de la habilidad de supervisar remotamente, cambiar y descargar software a todos los computadores HMI en la red. Esto es especialmente importante para redes distribuidas geográficamente por que los usuarios pueden ahorrar tanto tiempo como dinero eliminando la necesidad de viajar a cada localidad por mantenimiento o actualización.
Ahorros a través de todas las localidades	Estos ahorros resultan de la reutilización de plantillas y aplicaciones creadas para un proyecto u otros nuevos. Las compañías usan esto para llevar estándares en sus proyectos. Esto es particularmente beneficioso para integradores de sistemas.

A continuación un ejemplo simple para ilustrar como el desarrollo basado en objetos puede disminuir los costos.

Para comenzar se asume que se está desarrollando una aplicación de supervisión que tiene, entre otras cosas, 27 válvulas de doble efecto, teniendo cada una seis parámetros de proceso (I/O) que serán monitoreadas. Estos son puntos I/O en el PLC que medirán el funcionamiento de esta válvula.

En un sistema tradicional basado en tags, serán creados 162 tags (27 válvulas x 6 valores de parámetros (I/O) por válvula). En un sistema SCADA basado en objetos, se crea un objeto plantilla común para la válvula, y se instancian o replican objetos de ese objeto plantilla, que representan cada válvula individual. Ahora, se asume que toma 0.4 horas por tag para desarrollar la aplicación en un sistema SCADA tradicional basado en tags. Esto no incluye el desarrollo de lógica de lógica de control en el PLC o en gráficos

de proceso. Se hace la estimación de que toma dos horas desarrollar el objeto plantilla de la válvula y un adicional del 20% más (o 0.4 horas) por objeto instancia para personalizar cada válvula individual en la aplicación.

- Tipo de dispositivo: Válvulas de doble vía
- Número de instancias: 27
- I/O por instancias: 6

Cabe recordar que un objeto plantilla encapsula la creación de scripts, seguridad, alarmado, eventos, configuración para historia y comunicaciones a dispositivos. En un sistema basado en tags, todo esto necesita ser programado usando tags adicionales de memoria. Ahora, se presenta una comparación del tiempo total del desarrollo de la aplicación usando la aproximación para cada tipo de desarrollo.

TABLA N° 2.3: Esfuerzo en el desarrollo inicial

HMI basado en tags	SCADA basado en objetos	Ahorros
162 tags x 0.4 hrs. por tag=64.8 horas	(2 hrs. x 1 objeto plantilla) + (27 instancias de válvula x 0.4 hrs. por instancia)=12.8 hrs.	52 horas o el 80%

Esto representa un ahorro de tiempo impresionante, aún si se estima la mitad de este número, se ahorra un 40% en costo de desarrollo.

Ahora, para el caso en que se requiera modificaciones que afectan al 10% de la aplicación. Usando un desarrollo basado en tags, es razonable pensar en que se necesitará el 10% del tiempo empleado en el desarrollo original para hacer los cambios. Sin embargo, usando desarrollo basado en objetos como el Wonderware ArchestrA System Platform, el 10% del cambio solo necesita ser aplicado al objeto plantilla, esto dada la relación padre - hijo entre objetos y componentes. En esta situación, los ahorros adicionales pueden ser calculados de la siguiente manera:

TABLA N° 2.4: Esfuerzo de cambios en la aplicación

HMI basado en tags	SCADA basado en objetos	Ahorros
64.8 hrs. x 10% de cambio=6.48 hrs	2 hrs por objeto plantilla x 10% de cambio=0.2 hrs	6.28 horas o el 96%

Plataforma de Wonderware vs otras marcas

La siguiente es una tabla comparativa entre los sistemas SCADAS de tres marcas conocidas en el medio de la automatización industrial.

TABLA N° 2.5: Tabla comparativa entre 3 marcas de sistemas SCADA

Marca	Producto	Redundancia	Orientado a objetos	Orientado a tags	Abierto
Wonderware	WSP	Si	Si	No	Si
Rockwell	Factory Talk View	Si	No	Si	No (solo opc)
Siemens	WinCC	Si	No	Si	No (solo opc)

La principal ventaja que tiene el WSP sobre el Factory Talk View y el WinCC es su característica de ser orientado a objetos, cuyos beneficios se evidencian en los ejemplos mostrados en "*Ahorros en el ciclo de vida del sistema*".

2.3 Sistemas de control Distribuido

Los DCS (Distributed Control System) o Sistemas de Control Distribuido, son parte de los sistemas de manufactura o fabricación de una planta. Se usan en aplicaciones industriales para monitorear y controlar equipos distribuidos con intervención humana remota. Los DCS cumplen sus funciones a través de unos módulos de control automáticos e independientes distribuidos en una planta o proceso.

La filosofía de funcionamiento de esta arquitectura es evitar que el control de toda la planta este centralizado en una sola unidad, que es lo que se busca con el sistema SCADA. De esta forma si una unidad de control falla, el resto de unidades podría seguir funcionando independientemente. Todas las unidades de control están conectadas a través de redes de comunicación y monitoreo. [2]

Características de un DCS

Flexibilidad y capacidad de expansión: Capacidad de elegir (etapa inicial) o aumentar (etapas posteriores) el numero de variables de entrada/salida y del numero de controladores debido a una amplia gama de aplicaciones expansibles y clientes específicos.

Operaciones de mantenimiento*:* Las configuraciones de control e interfaces de operador deben ser fáciles de mantener y modificar no solo por ingenieros profesionales.

Apertura: Las variables y parámetros de control son leídos y escritos desde otras funciones de control.

Operatividad: Funciones avanzadas de control se deben mostrar en las mismas ventanas de operación y deben ser leídas por los operadores sin dar ninguna confusión.

Portabilidad: Parte del algoritmo de control no depende del entorno de hardware y debe poder adaptarse a distintas tecnologías informáticas.

Rentabilidad: Las ventajas de los algoritmos de control deben quedar claro. No solo acerca de la controlabilidad, sino también acerca de las inversiones realizadas, antes y después de la implementación del DCS.

Robustes/Redundancia: La redundancia en los sistemas de control apunta a disponer elementos/componentes adicionales que garantizan la operación de las funciones que cumplen dentro del sistema de control frente a fallas del mismo. [7]

Componentes funcionales de un DCS

En general el camino que recorre la información desde los dispositivos fuente (como instrumentos de medición en campo) hasta la interfaz de operación o HMI es como se

muestra en la siguiente figura y luego se presenta una lista en términos generales.

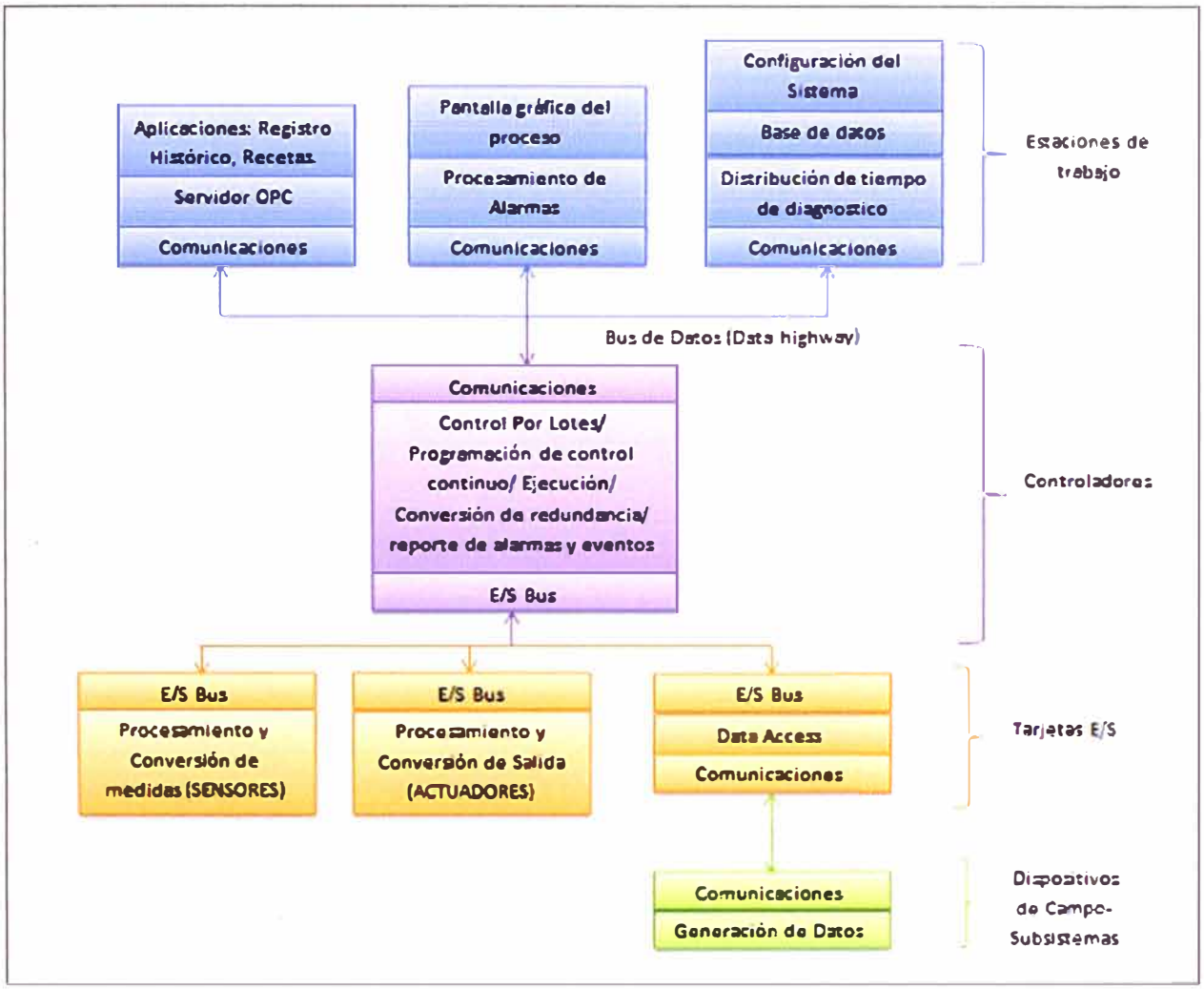

Figura 2.9 Componentes funcionales de un DCS

- Instrumentos o equipos de campo
- Línea de transmisión o cableado
- Unidad de montaje de terminales (en gabinete eléctrico de control)
- Modulo de bus de campo
- Procesador
- Red de comunicaciones (LAN industrial)
- Estación de operación (visualización de la información en el HMI)

Como se dijo anteriormente, el camino descrito está en términos generales, puede que para algunos DCS o por el uso de un protocolo de comunicación en particular se agregue algunos elementos extras. En cuanto a la implementación de la red de comunicaciones (LAN industrial) se usa la topología estrella con un arreglo de dos switchs de manera redundante, los cuales conforman al nodo concentrador, esto para el caso de sistemas pequeños (de aproximadamente solo 4 nodos). Para el caso de sistemas más complejos la topología estrella pasa a ser topología de árbol.

Seguidamente se muestran ejemplos de la estructura física de los DCS de algunas marcas reconocidas en el medio de la automatización y control.

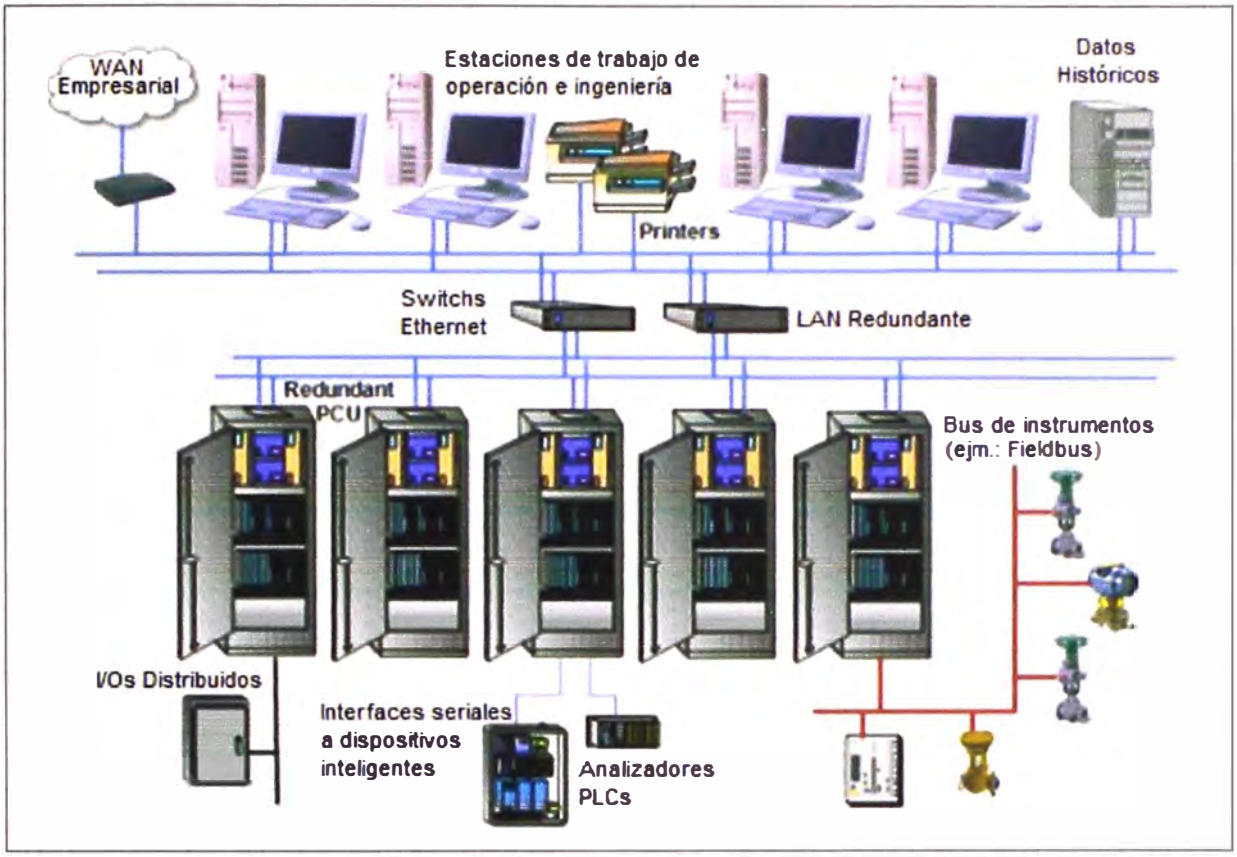

Figura 2.10 Estructura física general de un DCS

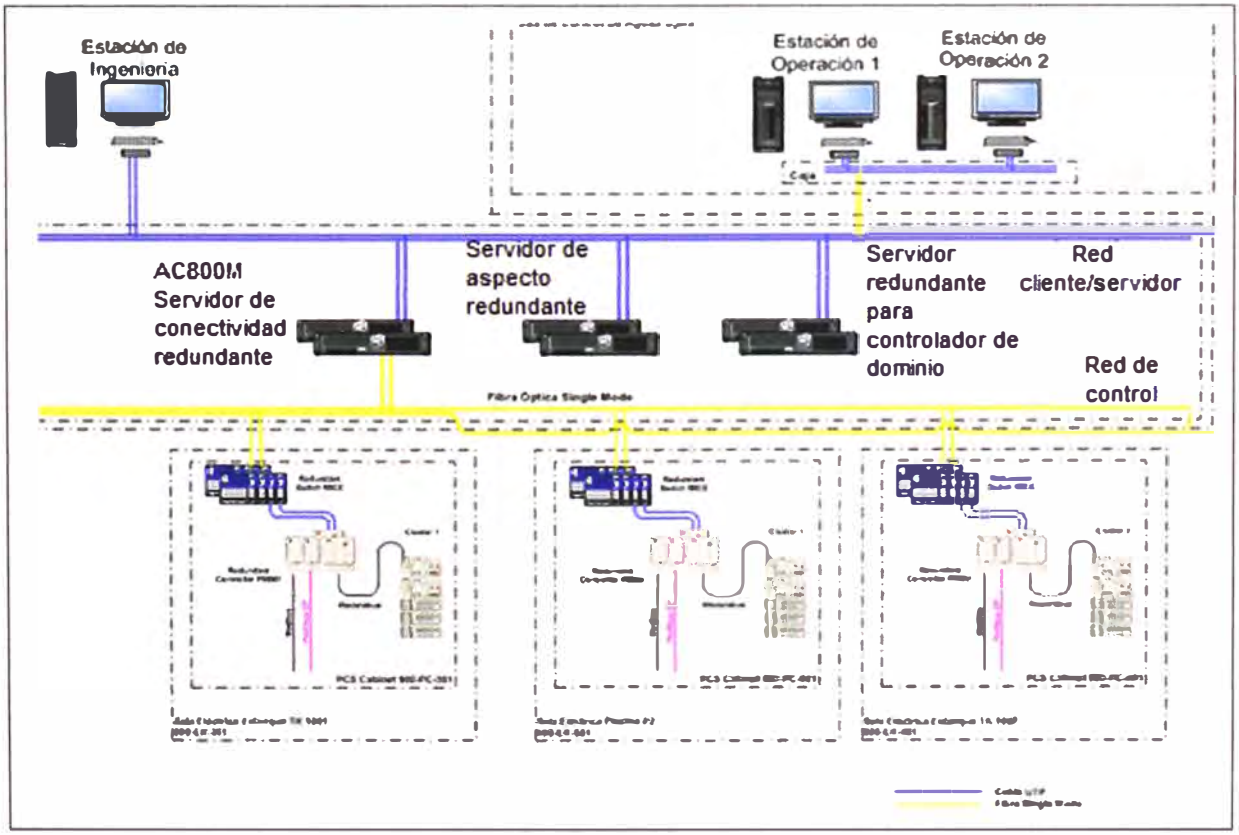

Figura 2.11 ABB, Industrial IT System 800 XA 5.0

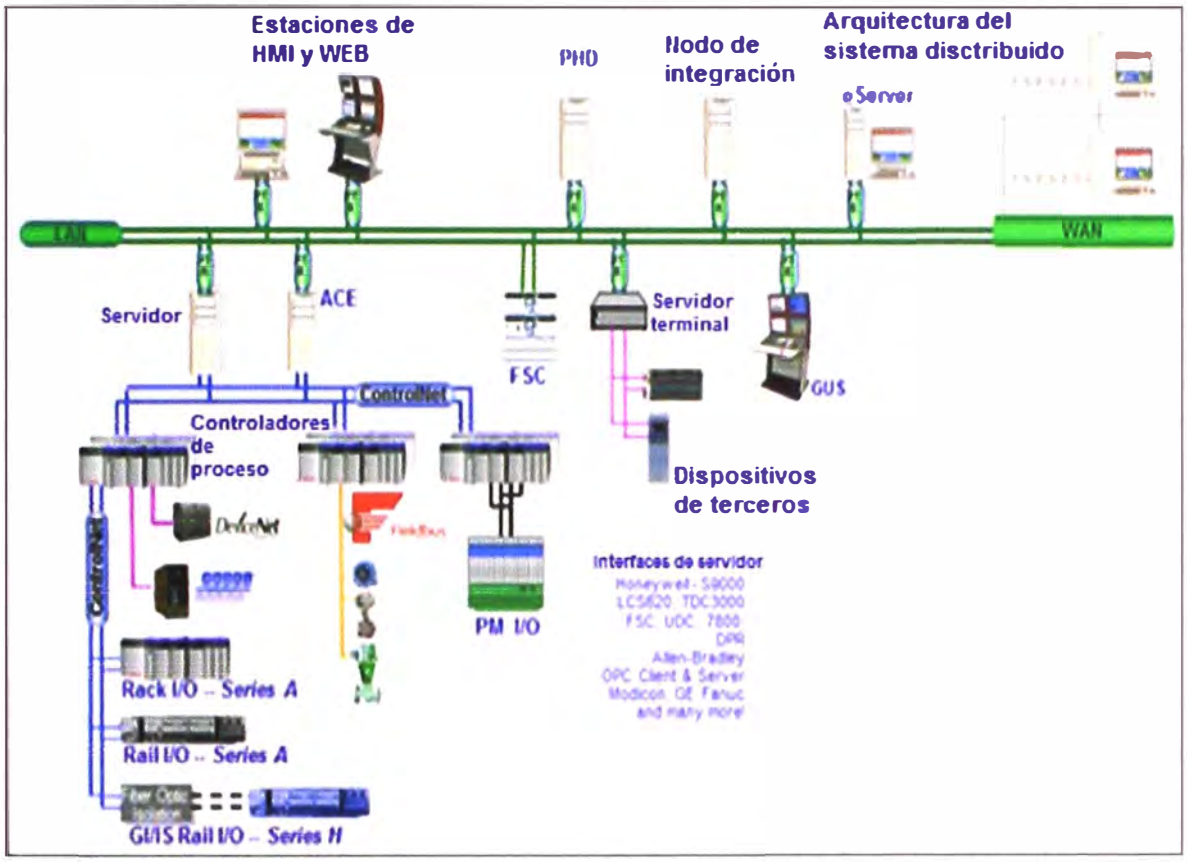

Figura 2.12 Honywell, Experion

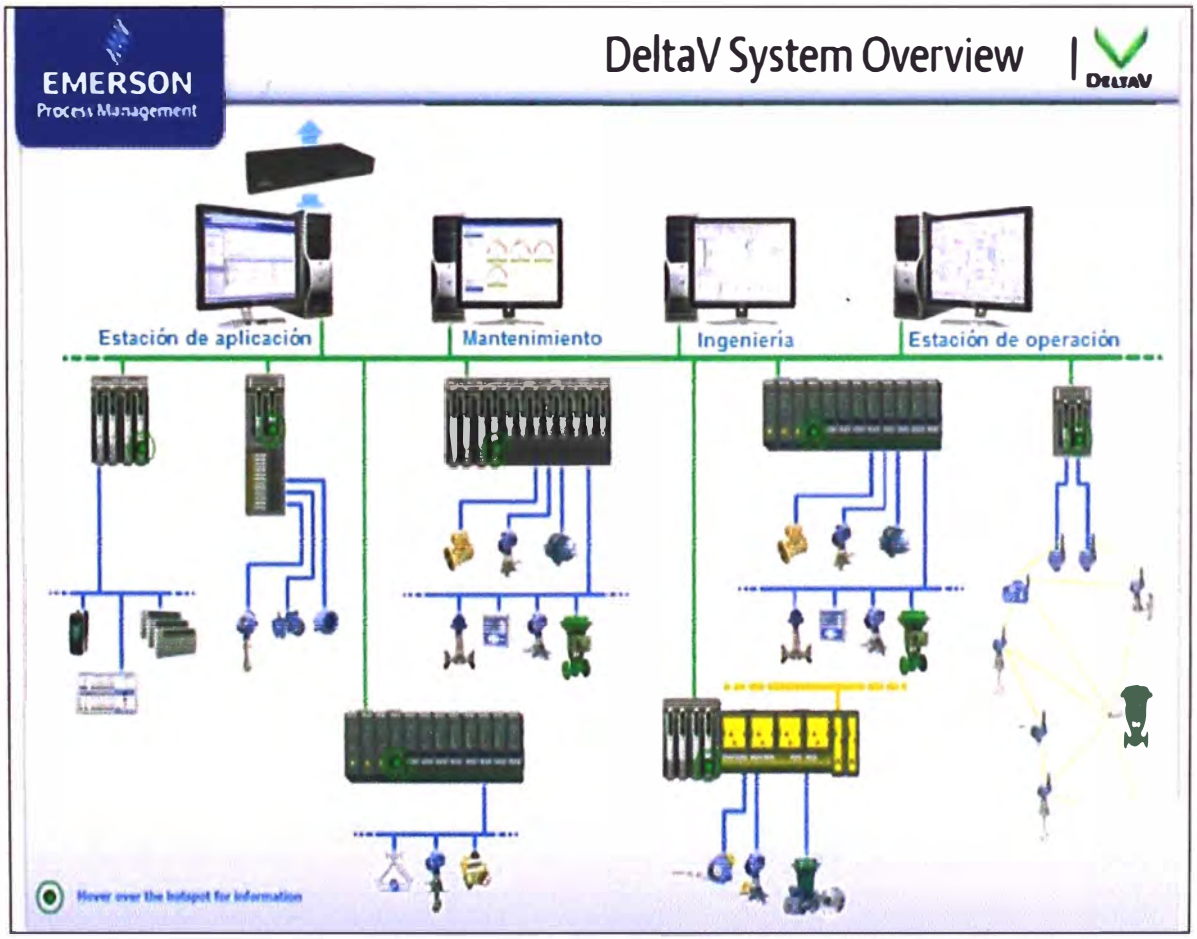

Figura 2.13 Emerson, Delta V

2.4. DCS Bailey

El DCS Bailey es un sistema que se caracteriza por trabajar con una topología de red en anillo la cual es llamada InfiNet.

2.4.1 Características del sistema Bailey

- Los elementos que conforman el DCS son principalmente la Unidad de Control de Procesos (PCU por sus siglas en inglés "Process Control Unit"), las estaciones de interfaz de operador y la red de comunicación (InfiNet).
- La red InfiNet soporta hasta 250 loops, con un máximo de 250 nodos por loop.
- Cada PCU tiene la capacidad de contar con dos controladores Bailey en redundancia.
- La configuración del sistema se hace a través de una computadora. Esto es, la configuración de la red InfiNet, de los controladores y los módulos de la red.
- El mantenimiento, monitoreo y diagnóstico de los elementos del sistema así como las acciones de control y supervisión de los equipos y elementos de campo se hace mediante una computadora conectada a la red InfiNet a través de una interfaz de red.
- La información de las variables de control es compartida entre módulos de diferentes nodos. [8]

2.4.2 Características de la red de comunicación

La red de comunicación del sistema Bailey, InfiNet, es del tipo propietario y presenta las siguientes características:

- El lazo de comunicaciones es serial y tiene la capacidad de ser redundante (anillo doble para redundancia).
- La velocidad de comunicación de 10 MBaud.
- Cada nodo conectado a la red puede operar independientemente, esto es, si un nodo deja de funcionar, el resto de nodos no se ven afectados.
- La red InfiNet proporciona una amplia cobertura en la red de la planta (la distancia máxima de nodo a nodo con cable coaxial es de 2 Km).
- No requiere de dispositivos que manejen el tráfico de datos ya que cada nodo es su propio maestro.
- La red InfiNet provee un tiempo de sincronización a través de un sistema de control basado en un token "Ring". [8]

2.4.3 Componentes del DCS Bailey

a) La unidad de control de procesos (PCU)

La PCU (Process Control Unit) es el elemento fundamental dentro de la arquitectura Bailey. Los equipos que conforman la PCU son módulos Infi90 y determinan las funciones de control, monitoreo, alarmas, tendencias y reportes.

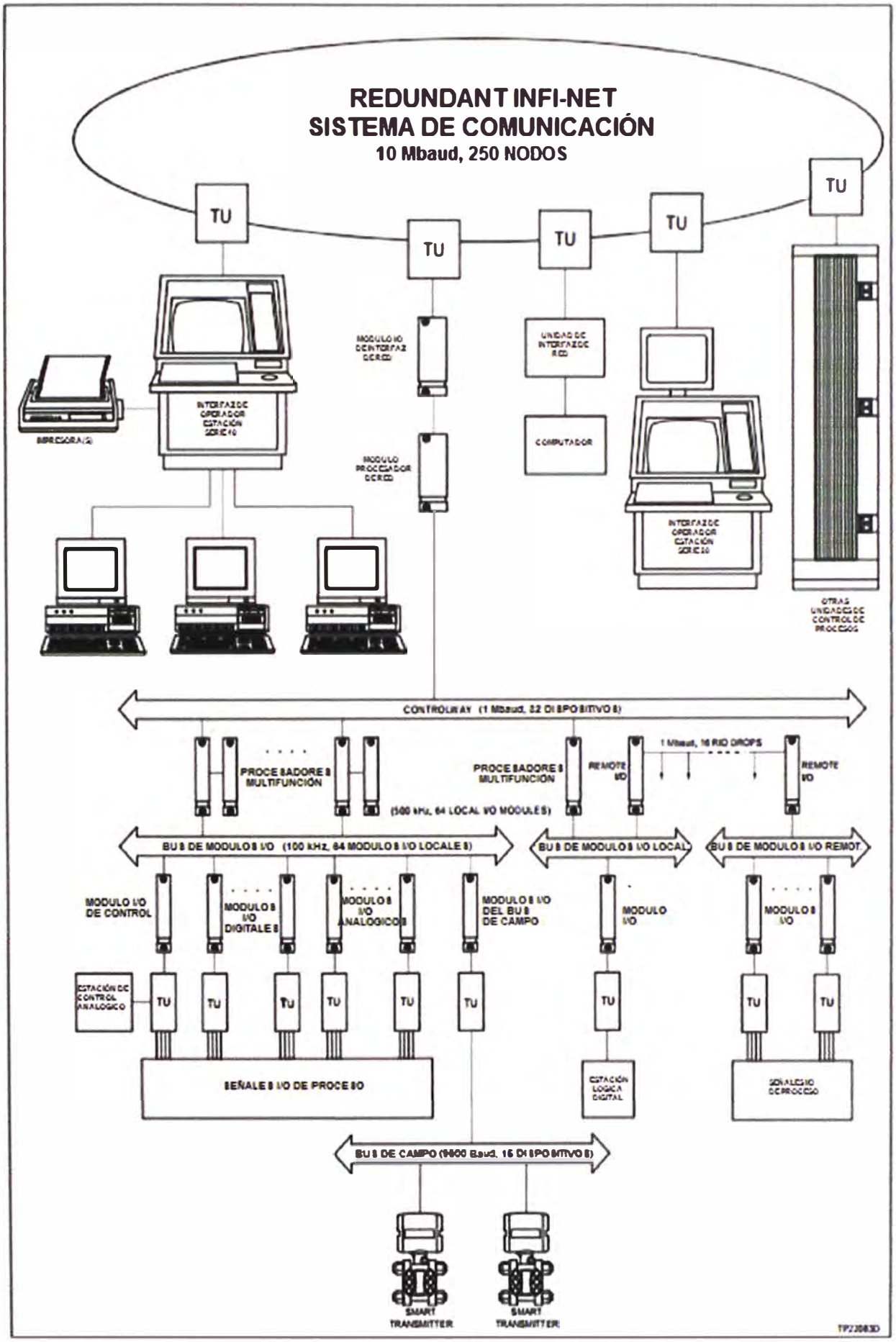

Figura 2.14 Arquitectura del sistema Infi90

La interfaz de comunicación del PCU con la red InfiNet se da a través de dos módulos:

- NIS: Enlaza el nodo a la InfiNet.
- NPM: Por un lado se comunica al bus Controlway y por el otro al NIS, con lo cual se completa la comunicación del bus con la InfiNet.

Los principales elementos de un PCU son:

- Las fuentes de poder.
- Los controladores.
- Los módulos I/O.
- Las unidades terminales. [8]

Respecto a los controladores se tiene que son de dos tipos, controladores MFP y controladores BRC.

Controladores MFP: Son controladores de múltiples propósitos y pueden funcionar en redundancia.

- Las versiones más conocidas son: IMMFP01, IMMFP02, IMMFP03, IMMFP11 y IMMFP12.
- Se programa con el software Composer. Su conexión es a través de un nodo de consola (ICI) conectándose a través de un anillo InfiNet.
- Requiere de una NIS y una NPM para poder acceder al lazo InfiNet. [3]

Controladores BRC: Son controladores de múltiples propósitos y pueden funcionar en redundancia.

- Las versiones más conocidas son: BRC100, BRC200, BRC300 y BRC400.
- Se programa con el software Composer. Su conexión es a través de un nodo de consola (ICI) conectándose a través de un anillo InfiNet.
- Requiere de una NIS y una NPM para poder acceder al lazo InfiNet. [3]

Las Unidades Terminales (TU): Son principalmente para comunicación y entradas y salidas eléctricas. Para comunicación el módulo más común es la NTCL01, la cual se utiliza para conectarse al anillo InfiNet a través de cable coaxial. [3]

b) Estación de Interfaz del Operador (OIS)

La OIS (Operator Interface Station) no es otra cosa que una computadora la cual viene a ser la interfaz de operación entre el sistema de control y el operador de planta. A través de la OIS se hace la configuración del sistema, como por ejemplo la configuración de los módulos del nodo, de la lógica de control y de las variables del proceso. Además es a través de una OIS que el operador de planta hace el monitoreo y control del proceso.

- El sistema operativo de las OIS12 es el QNX. [8]
- El sistema operativo de las OIS de la serie 40 es el VMS. [8]

- Es posible operar desde una OIS usando el sistema operativo Windows NT o Windows 95 usando el software semAPI para la plataforma Windows NT. [9]
- El número de consolas varía de acuerdo al planeamiento del sistema de control o del tamaño de la planta. [8]

c) Software del Sistema Bailey

Programa Composer: Permite programar los controladores MFP, BRC y los módulos HP800. También permite monitorear y diagnosticar los módulos Infi90 de los nodos de la red InfiNet.

Es en la interfaz del Composer donde se crea la red de control en anillo, la Unidad de Control o Nodo y el controlador. A través de la herramienta Inspect se puede revisar el estado de los módulos del nodo.

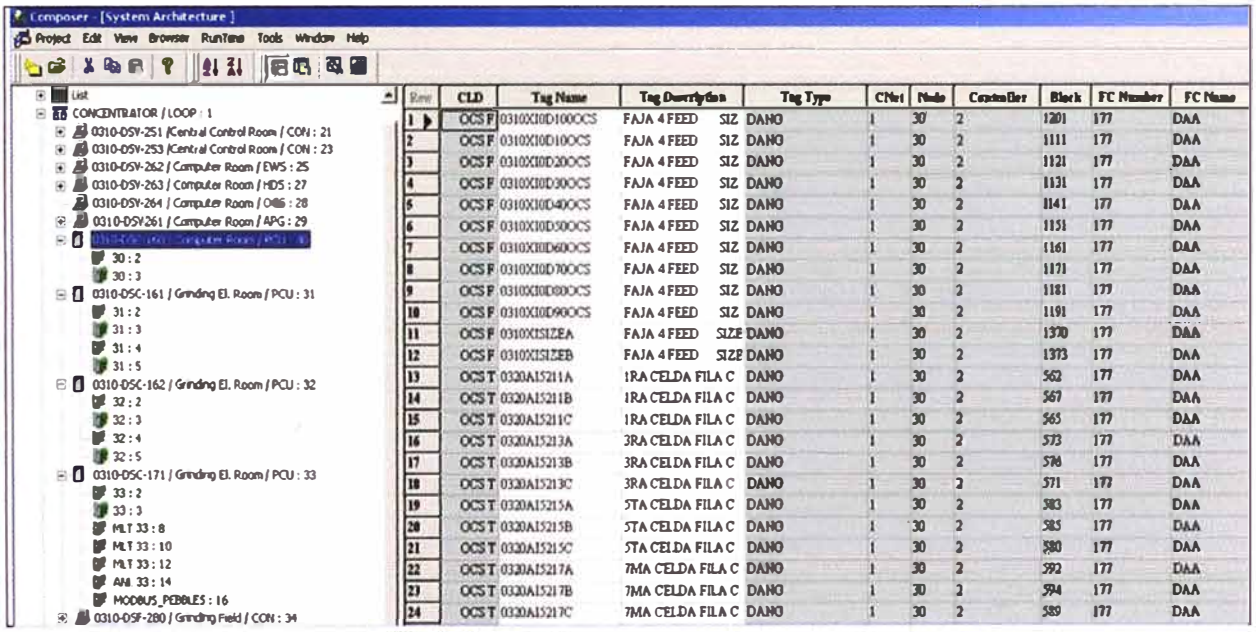

Figura 2.15 Entorno del Composer

Programa Conductor NT: El software de supervisión y control más usado de los DCS Bailey es el Conductor NT. Posee una arquitectura basada en Cliente - Servidor. Pueden tener servidores redundantes que comparten la base de datos de variables (Tags y hasta 10 clientes).

Servidores OPC: El conductor NT tiene la opción de comportarse como cliente OPC. [3]

Bloques de Códigos de Función: A diferencia de sistemas que dependen de aplicaciones que fuerzan el empleo de pasos de control específicos, los sistemas Infi 90 ofrecen una extensa librería de más de 200 códigos de función que residen en módulos activos. Estos algoritmos poderosos pueden ser usados en secuencias o anidadas dentro de bloques de funciones on-board (ubicaciones de memorias direccionables) para entregar una amplia variedad de estrategias de control sin tener un conocimiento especial de programación.

Los códigos de función del sistema Infi 90 son compatibles con una variedad de otras técnicas de configuración. Los códigos de funciones existen para BASIC, Batch 90™, Lenguajes Ladder y C. [10]

TABLA N° 2.6: Lista de los primeros 73 Códigos de Función

Código de Función	Descripción
FC 1	Function Generator
FC 2	Manual Set Constant (Signal Generator)
FC 3	Lead/Lag
FC 4	Pulse Positioner
FC 5	Pulse Rate
FC 6	High/Low Limiter
FC 7	Square Root
FC 8	Rate Limiter
FC 9	Analog Transfer
FC 10	High Select
FC 11	Low Select
FC 12	High/Low Compare
FC 13	Integer Transfer
FC 14	Summer (4 Inputs)
FC 15	Summer (2 Inputs)
FC 16	Multiply
FC 17	Devide
FC 18	PID Error Input
FC 19	PID (PV and SP)
FC 20	Indicator Station
FC 21	M/A Station (Basic)
FC 22	M/A Station (Cascade)
FC 23	M/A Station (Ratio)
FC 24	Adapt
FC 25	Analog Input (Same PCU Node)
FC 26	Analog Input Loop
FC 27	Analog Input
FC 28	Analog Output (Same PCU Node)
FC 29	Analog Output
FC 30	Analog Exception Report
FC 31	Test Quality
FC 32	Trip
FC 33	Not
FC 34	Memory
FC 35	Timer

TABLA N° 2.6: Lista de los primeros 73 Códigos de Función (continuación)

Código de Función	Descripción
FC 36	Qualified OR (8 Input)
FC 37	And (2 Input)
FC 38	And (4 Input)
FC 39	Or (2 Input)
FC 40	OR (4 Input)
FC 41	Digital Input/Controlway/Module Bus
FC 42	Digital Input/Loop
FC 43	TCS Digital Input
FC 44	TCS Digital Output
FC 45	Digital Exception Report
FC 46	Digital Input List
FC 47	Analog Input Exception Report (NAMM01)
FC 48	Reserved for future use
FC 49	Digital Output Buffer
FC 50	Manual Set Switch
FC 51	Manual Set Constant
FC 52	Manual Set Integer
FC 53	Executive Block (COM)
FC 54	Executive Block (NLMM01)
FC 55	Hydraulic Servo
FC 56	Executive Block (NAMM01)
FC 57	Reserved for future use
FC 58	Time Delay (Analog)
FC 59	Digital Transfer
FC 60	Group I/O Definition
FC 61	Blink
FC 62	Remote Control Memory
FC 63	Analog Input List (Same PCU)
FC 64	Digital Input List (Same PCU)
FC 65	Digital Sum With Gain
FC 66	Analog Trend
FC 67	Reserved For Future Use
FC 68	Remote Manual Set Constant
FC 69	Test Alarm
FC 70	Analog Point Definition
FC 71	Executive Block NAMM02/IMAMM03
FC 72	Analog Slave Definition
FC 73	Calibration

2.5 Protocolos de Comunicación Industrial

La automatización de las plantas industriales requiere que se pueda monitorear y controlar los equipos de campo, los cuales pueden ser la instrumentación, los actuadores o equipos controladores de terceros los cuales se encargan de una parte específica del proceso total de la planta, así como también en algunos casos una interacción entre uno y otro equipo es requerido. Para que esto pueda darse, es necesario que los equipos de campo sean capaces de comunicarse mediante lenguajes que sean conocidos, estos lenguajes vienen a ser los denominados Protocolos de Comunicación Industrial.

Los protocolos de comunicación que se usan hoy en día están estandarizados, lo cual permite una rápida integración de los diversos equipos que se usan en la industria. Dentro de los protocolos de comunicación más conocidos se tiene los siguientes:

- Hart
- Profibus
- Fundation Fieldbus
- OPC
- MODBUS
- Ethernet IP
- DeviceNet

Figura 2.16 Protocolos de Comunicación Industriales

Para efectos del presente informe se presenta un resumen del protocolo de comunicación industrial OPC, el cual es usado para comunicarse con la red InfiNet del sistema Bailey.

2.6 Protocolo OPC

OPC es conectividad abierta vía estándares abiertos. OPC llenó una necesidad en la automatización como los drivers de las impresoras lo hicieron para Windows.

OPC es todo sobre la conectividad y productividad abierta en la industria de la

automatización y sistemas empresariales que soporta la industria. La interoperabilidad es asegurada a través de la creación y mantenimiento de especificaciones de estándares abiertos. Actualmente hay siete especificaciones de estándares completados o en desarrollo.

Introducción a OPC

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

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 OPC Foundation. Cualquier software que sea compatible con estas especificaciones OPC proporciona a usuarios e integradores conectividad abierta e independiente tanto del fabricante del dispositivo como del desarrollador de la aplicación Cliente.

La comunicación OPC: Conceptualmente se puede representar como una capa de "abstracción" intermedia que se sitúa entre la fuente de datos y el Cliente de Datos, permitiéndoles intercambiar datos sin saber nada el uno del otro.

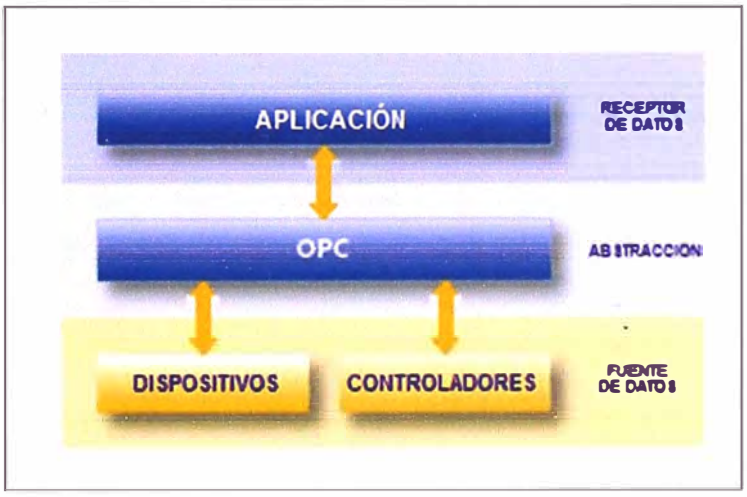

Figura 2.17 La comunicación OPC como una capa de Abstracción

La "abstracción de dispositivo" OPC se consigue utilizando dos componentes OPC especializados llamados Cliente OPC y Servidor OPC. Cada uno de ellos es descrito en la siguiente sección. Es importante resaltar que el hecho de que la Fuente de Datos y el Cliente de Datos puedan comunicar 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 y/o interfaces nativos siguen existiendo, pero sólo comunican con uno de los dos componentes del software OPC. Y 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 hablar directamente entre sí.

Tipos de datos que soporta OPC

Los tipos de datos más comunes transferidos entre dispositivos, controladores y aplicaciones en automatización se pueden encuadrar en tres categorías:

- Datos de tiempo real
- Datos históricos
- Alarmas y eventos

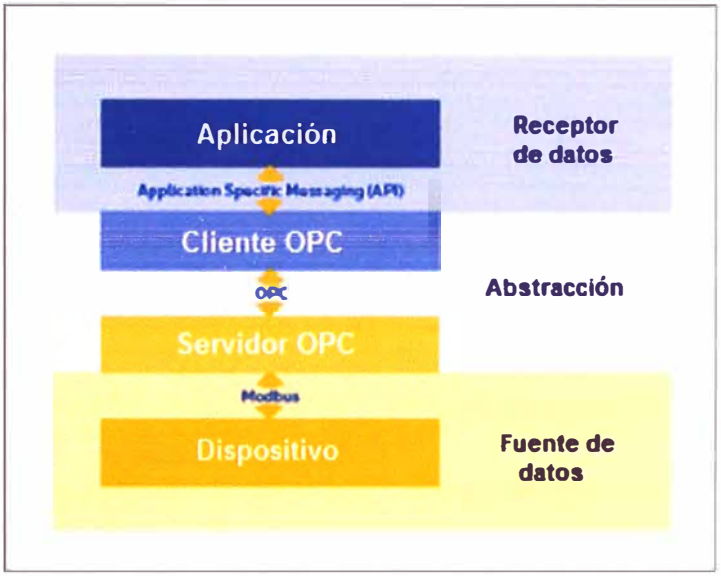

Figura 2.18 Componentes OPC - Cliente y Servidor OPC

A su vez cada una de las categorías anteriores soporta una amplia gama de tipos de datos. Estos tipos de datos pueden ser enteros, punto flotante, cadenas, fechas y distintos tipos de arrays, por mencionar algunos. OPC asume el reto de trabajar con estas distintas categorías de datos especificando de forma independiente cómo se va a transmitir cada uno de ellos a través de la arquitectura Cliente OPC - Servidor OPC.

Las tres especificaciones OPC que se corresponden con las tres categorías de datos son:

1. OPC Data Access Specification (OPC DA) - utilizada para transmitir datos de tiempo real.
2. OPC Historical Data Access Specification (OPC HDA) - utilizada para transmitir datos históricos.
3. OPC Alarms & Event Specification (OPC A&E) - utilizada para transmitir información de alarmas y eventos.

Históricamente, la mayoría de los servidores OPC solo soportan datos de tiempo real (OPC DA). No es obligatorio que todos los servidores OPC incluyan todas las especificaciones OPC. Es aconsejable investigar que especificaciones OPC incluye un Servidor OPC antes de elegirlo para un proyecto.

Servidor OPC

Un servidor OPC es una aplicación de software. Un driver "estandarizado" desarrollado específicamente para cumplir con una o más especificaciones OPC. La palabra "Server" en "OPC Server" no hace referencia en absoluto al ordenador donde este software se está ejecutando. Hace referencia a la relación con el Cliente OPC.

Los servidores OPC son conectores que se pueden asimilar a traductores entre el mundo OPC y los protocolos nativos de una Fuente de Datos. OPC es bidireccional, esto es, los servidores OPC pueden leer de y escribir en una Fuente de Datos. La relación Servidor OPC/Cliente OPC es de tipo maestro/esclavo, lo que significa que un servidor OPC solo transferirá datos de/a una Fuente de Datos si un cliente OPC así lo pide.

Los servidores OPC pueden comunicar prácticamente con cualquier Fuente de Datos cuyos datos pueden ser leídos o escritos por medios electrónicos. Una breve lista de posibles Fuentes de Datos incluye: dispositivos, PLCs, DCSs, RTUs, instrumentos de medición, bases de datos, historiadores, software de cualquier tipo (i.e. Excel), páginas web e incluso archivos CSV (texto separado por comas) de actualización automática. Para comunicar con cualquiera de estos dispositivos se requiere únicamente el uso de un Servidor OPC que utilice el protocolo o interfaz nativo apropiado. Una vez que se ha configurado dicho Servidor OPC, cualquier aplicación Cliente que utilice OPC (y tenga los permisos adecuados) puede empezar a comunicar con la Fuente de Datos sin que importe la forma en que esta comunica de forma nativa.

Un vistazo conceptual de cómo funciona un servidor OPC puede ser el siguiente:

- **Módulo de comunicaciones OPC:** Esta es la parte del Servidor OPC responsable de comunicar adecuadamente con un Cliente OPC. Los Servidores OPC bien diseñados deben ser plenamente compatibles con las especificaciones OPC que implementen, para asegurar que comunican correctamente con cualquier Cliente OPC.
- **Módulo de comunicaciones nativas:** El servidor OPC debe emplear el método de comunicación más eficiente con la Fuente de Datos. En algunos casos, esto implica comunicar con la Fuente mediante su protocolo propietario de datos, mientras que en otros casos, esto significa comunicar a través de una Interfaz de Programación de la Aplicación (API). Típicamente, cuanta más experiencia tenga el desarrollador del Servidor OPC con el dispositivo, mejor utilizará las posibilidades de comunicación que ofrece el dispositivo.
- **Módulo de traducción/mapeado:** Aquí es donde sucede toda la "magia" de un Servidor OPC. La función de este módulo es interpretar de forma adecuada las peticiones OPC entrantes de un Cliente OPC, convirtiéndolas en peticiones nativas que se envían a la Fuente de Datos y viceversa. Si esto se hace eficientemente, se puede mantener al

mínimo la carga sobre la Fuente de Datos mientras se maximiza la capacidad de transmisión de datos.

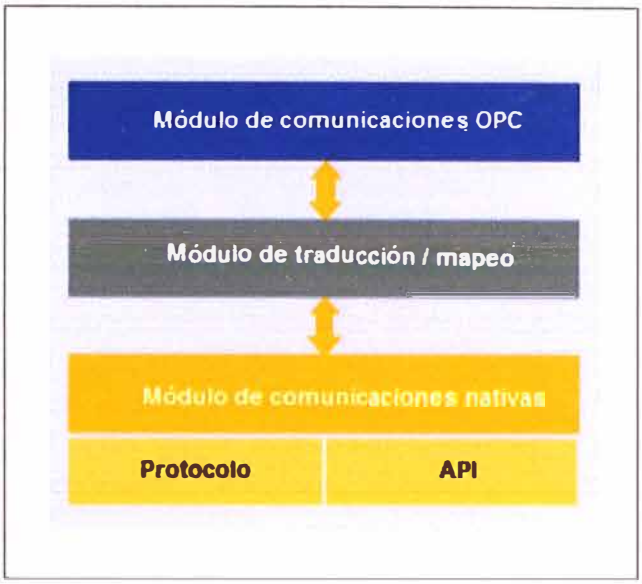

Figura 2.19 Anatomía conceptual de un servidor OPC

Un Cliente OPC de un determinado fabricante puede comunicarse con servidores OPC de otros fabricantes, esto siempre que tanto el Cliente OPC como el Servidor OPC cumplan con las mismas especificaciones OPC, independientemente de que suministrador vengan.

Los Servidores OPC no pueden compartir datos con otros Servidores OPC, es decir, no se comunican directamente unos con otros, solo están diseñados para comunicar con clientes OPC. Sin embargo, existen utilidades OPC diseñadas específicamente para que esta comunicación entre Servidores OPC sea posible (y fácil).

Cliente OPC

Un Cliente OPC es una pieza de software creada para comunicar con Servidores OPC. Utiliza mensajería definida por una especificación concreta de la OPC Foundation.

Para saber qué es lo que hace un Cliente OPC se puede dar un doble enfoque para un mejor entendimiento:

Conceptualmente: un Cliente OPC representa un destino de datos. Inician y controlan la comunicación con Servidores OPC basados en las peticiones recibidas desde la aplicación en la que están embebidos. Los Clientes OPC traducen las peticiones de comunicación provenientes de una aplicación dada en la petición OPC equivalente y la envían al Servidor OPC adecuado para que la procese. A cambio, cuando los datos OPC vuelven del Servidor OPC, el Cliente OPC los traduce al formato nativo de la aplicación para que ésta pueda trabajar de forma adecuada con los datos.

Técnicamente: Los Clientes OPC son módulos de software utilizados por una aplicación para permitirla comunicarse con cualquier Servidor OPC compatible visible en la red. Típicamente, los Clientes OPC están embebidos en aplicaciones como HMIs, SCADAs, graficadores, Historiadores o generadores de informes, convirtiéndolos en aplicaciones compatibles OPC.

Nota: Los Clientes OPC pueden comunicar de forma simultánea con múltiples servidores OPC. Esto significa que un Cliente OPC puede leer y escribir datos desde y hacia múltiples dispositivos (Fuentes de Datos) a través de sus respectivos Servidores OPC.

Un Cliente OPC se puede dividir conceptualmente en tres módulos:

- **Módulo de comunicaciones OPC:** Aunque no tan involucrado como en el Servidor OPC (en los Servidores OPC esta parte es más compleja) es crucial para que el Cliente OPC se comporte como debe al conectarse a un Servidor OPC, intercambiar datos con él y desconectarse sin desestabilizar al Servidor OPC.
- **Módulo de comunicaciones con la aplicación:** El Cliente OPC típicamente está diseñado para trabajar en una aplicación específica, por lo que, para permitir que la información pase de la aplicación al Servidor OPC pasando por el Cliente OPC, realiza una serie de llamadas al interfaz para la programación de la aplicación (API). También es posible que un Cliente OPC genérico comunique con una aplicación mediante un protocolo en lugar de con llamadas al API siempre que la aplicación soporte ese protocolo.
- **Módulo de traducción/mapeado:** Una de las funciones clave del Cliente OPC es la de traducir de forma bidireccional la información que su aplicación necesita leer de o escribir al dispositivo o Fuente de Datos.

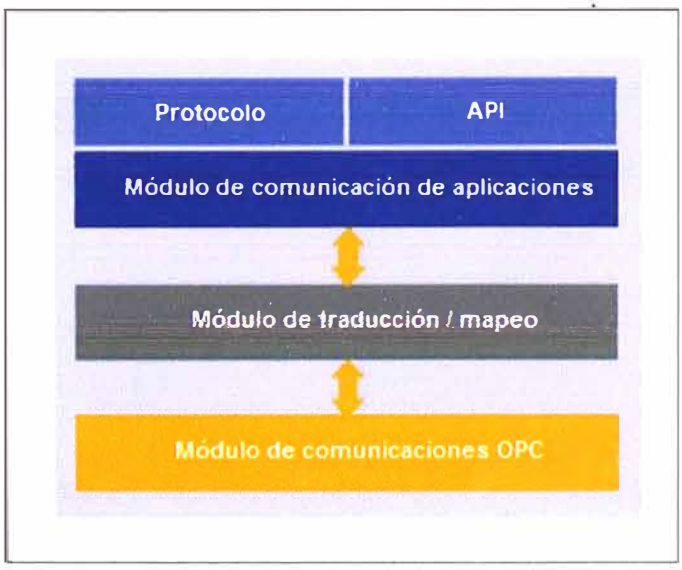

Figura 2.20 Anatomía conceptual de un Cliente OPC

Un Cliente OPC no puede comunicar directamente con otro Cliente OPC. En OPC las

comunicaciones Cliente - Cliente no están definidas, solo se soporta la arquitectura Cliente/Servidor. Por ello, si una aplicación debe proveer datos OPC a otros clientes, necesita tener su propio servidor OPC. [12]

2.6.1 OPC de Rovisys

La compañía RoviSys ha desarrollado una interfaz totalmente integrada entre los Clientes OPC y un DCS Bailey que promueve futuras actualizaciones a la planta utilizando OPC. La interfaz es llamada OPC90 Server. Permite un reemplazo fácil y extremadamente intuitivo (para el usuario) de las consolas de operador Bailey con consolas de Cliente OPC basados en Windows. También provee una ruta abierta para la migración desde la plataforma de control de Bailey a OPC, tanto como lo permita el tiempo de la planta, los recursos y los horarios.

El OPC90 Server hace posible un accesso similar al del DCS a los Sistemas de Control Distribuidos Bailey Controls Command Series, NETWORK 90 e INFI 90. El servidor puede comunicar al nivel de red a través de la apropiada Computer Interface Unit (NCIU0X) Plant Loop a Computer Interface (INPCI0X), InfiNet a Computer Interface (INICI01, INICI03, INICI12, INICI13, IIMCP01/02), o dentro de un único Process Control Unit, a través de un Computer Interface Command (CIC01), Serial Port Module (NSPM01, IMSPM01) y Computer Port Module (CPM02/03).

La conclusión es que el OPC90 puede comunicarse con todas las interfaces disponibles para los Sistemas de Control Distribuidos NETWORK 90 / INFI 90 y Command Series. La integridad del sistema, la funcionalidad y el rendimiento de los datos se mantiene utilizando técnicas de reporte por excepción estándares.

El OPC90 Server es un servidor compatible con OPC Data Access 1.0, 2.0 y 2.05. Esto se logra usando un juego de herramientas OPC de grado industrial para implementar su interfaz OPC. El mantenimiento y la certificación de este conjunto de herramientas es administrado por una compañía que es miembro fundador de OPC Foundation.

El OPC90 Server soporta DCOM. DCOM permite a una PC ejecutando el OPC90 proveer su información a través de una red de área local a otras PCs ejecutando softwares clientes OPC.

Descripción Funcional

El OPC90 Server puede ejecutar sobre los sistemas operativos Windows NT / 2000 / XP / 2003 Server / Vista / 7 / 2008 Server. Es implementado como una colección de bloques que son específicos para cada tipo de bloque de reporte por excepción encontrado dentro del sistema Bailey. Esta colección de bloques es almacenado como una configuración del OPC90.

Se configura un bloque especial llamado bloque Device (DEVICE) para configurar el (los)

puerto(s) de comunicación usado para acceder a la interfaz Bailey, define tasas de actualización requeridas y reporta diversos estados de los canales de comunicación.

Todos los otros bloques del OPC90 Server están vinculados a un DEVICE que provee la información necesaria para administrar la comunicación con el sistema Bailey. Los otros diversos bloques OPC90 Server vinculados al DEVICE contienen la dirección Bailey donde el bloque de reporte por excepción está ubicado. El DEVICE usa esta dirección y el tipo de bloque para determinar cómo el punto es establecido y administrado dentro de la interfaz Bailey.

Como la información es recibida desde la interfaz Bailey para cada uno de los puntos establecidos esta es analizada y copiada a los tags apropiados dentro del correspondiente bloque del OPC90 Server. Esta implementación tiene muchas ventajas:

- Extremadamente intuitivo para usuarios de Bailey.
- Desarrollado con un conjunto de herramienta OPC de grado industrial fabricado por un miembro fundador de la compañía OPC Foundation asegura un cumplimiento del 100% de la especificación OPC 1.0, 2.0 y 2.05.
- Validado asimismo usando las herramientas de prueba del OPC Foundation.
- La capacidad de puntos no está limitada por el tipo de puntos usados por la planta sino solo por el tipo de interfaz Bailey disponible (típicamente hasta 10000 tags).
- Los niveles de alarmas son configurados en tiempo real a través de la información recibida desde el sistema Bailey (elimina la necesidad de mantener configuraciones de niveles de alarmas en dos áreas).
- Soporta interfaces Bailey redundantes o canales de comunicación redundante para una única interfaz Bailey.
- Soporta la habilidad para fácilmente importar y exportar bloques de función OPC90 a o desde archivos delimitados por comas (*.csv). El archivo CSV puede ser creado fácilmente usando Microsoft Excel y luego simplemente importado a un OPC90 Server para crear una configuración OPC90.

Vista General de la Comunicación Bailey

Bailey utiliza una técnica de comunicación para el intercambio de información llamado reporte por excepción. Reporte por excepción significa que la información es enviada cuando ha cambiado significativamente o cuando un tiempo máximo ha expirado desde la última vez que esta fue enviada.

Toda la información de reporte por excepción se hace disponible para el resto del sistema a través de un conjunto de bloques de función relacionados al reporte por excepción los cuales están enlazado por un bloque equivalente de OPC90 Server. Bailey tiene una serie de dispositivos basados en RS232 Serial genéricamente llamados Computer

Interface Units (Unidades de Interfaz de Computador) que son usados para recibir esta información de reporte por excepción. Rovisys está muy bien informado en el uso de las diversas interfaces Bailey y ha implementado un único driver que puede utilizar los siguientes tipos de interfaces:

TABLA N° 2.7: Interfaces disponibles de acceso al sistema Bailey

Interfaz Bailey	Máx. Número de Bloques Bailey	Reporte por Excepción	Operación de Control
NSPM01	500	No	No
IMSPM01	500	No	No
IMCPM02	500	No	No
IMCPM03	500	Si	Si
NCIC01	500	Si	Si
NCIU01	500	Si	Si
NCIU02	2,500	Si	Si
NCIU03	5,000	Si	Si
NCIU04	10,000	Si	Si
INPCI01	500	Si	Si
INPCI02	5,000	Si	Si
IIMCP01	10,000	Si	Si
IIMCP02	30,000	Si	Si
INICI01	10,000	Si	Si
INICI12*	10,000	Si	Si
INICI13*	30,000	Si	Si
INICI03*	30,000	Si	Si

Cuando se usa esta interfaz, el entorno de software ABB Bailey semAPI no es requerido. El INICI12, INICI13 e INICI03 no soportan comunicación serial de canal doble. Cuando se usa comunicación serial para las INICI12, INICI13 e INICI03, se debe asegurar que el cable está conectado al puerto etiquetado con "terminal" en la unidad de terminación o módulo de terminación Bailey de ABB.

Se debe notar que unos pocos de los dispositivos mencionados líneas más arriba no soportan reporte por excepción. Específicamente el NSPM01, IMSPM01, IMCPM01 y IMCPM02. La colección de información desde estos dispositivos está básicamente

limitada a los bloques del OPC90 Server que encuestan para obtener los valores de los bloques. Los nombres de estos bloques son POLL, BLK y SPEC. Las interfaces restantes soportan los siguientes bloques del OPC90 Server y tipos de bloques de reporte por excepción asociados al Bailey:

TABLA N° 2.2: Relación entre Bloques del OPC90 Server y del Bailey

Tipo de bloque del servidor OPC90	Nombre del bloque Bailey	Número de F.C.
AIL	Salida analógica/ lazo - AOL	30, 70, 158
BLK	Configuración de bloque Bailey	cualquier código de función
DAANG	Adquisición de datos analógica - DAANG	177
DADIG	Adquisición de datos digital - DADIG	211
DD	Driver de dispositivo - DD	123
DIL	Salida digital / lazo - DOL	45
HAI	Entrada analógica de Harmony	222
HAO	Salida analógica de Harmony	223
HDI	Entrada digital de Harmony	224
HDO	Salida digital de Harmony	225
MODSTAT	Estado del módulo	cualquier módulo
MSDD	Driver de dispositivos multi estados - MSDD	129
MUXCIU	Multiplexa el puerto COM de una PC a la interfaz del Bailey para configuración relacionada al software.	Cualquier código de función
Poll	Encuestar valor	cualquier bloque
RCM	Memoria remota de control - RCM	62
RMC	Control remoto de motor - RMC	136
RMSC	Constante remota de configuración manual - RMSC	68
SOE	Secuencia de eventos	99, 243
SPEC	Especificación - SPEC	cualquier bloque
STN	Estación de control - STN	21, 22, 23, 80
TXT	Selector de texto - TEXT	151
TEXTSTR	Cadena de texto - TEXTSTR	194

Las interfaces de computador comunes de Bailey usados para ganar acceso a los bloques de reporte por excepción de Bailey presentados en la tabla de líneas arriba también soportan la habilidad de permitir a un sistema emular una salida de la mayoría de estos mismos tipos de puntos. Estos tipos de bloques son llamados "output report". Las consolas del Bailey reciben el tipo de punto, la información reportada por excepción directamente desde la interfaz Bailey en el mismo formato como si estuviese viniendo desde un controlador Bailey equivalente. Esto le permite a los faceplates predefinidos de la consola Bailey para cada uno de estos tipos de puntos de control ser utilizados

cuando se recibe información desde los tipos de puntos de reporte del OPC90 Server. En vez de direccionar los tags a un bloque en el controlador Bailey estos son direccionados a un número de bloque asignado dentro de la interfaz Bailey que el OPC90 está comunicando. Las direcciones de anillo y nodo a usar es el asignado a la interfaz con una dirección de módulo fija de dos.

La siguiente tabla presenta una lista de interfaces Bailey que soportan el tipo de bloque de reporte de salida (output report block type):

TABLA N° 2.8: Bloques de reporte de salida

Interfaz Bailey	Sistema Bailey	Max. Número de puntos	Soportado por el servidor OPC90
NCIU02	Network 90	2500	Si
NCIU03	Network 90	5000	Si
NCIU04	Infi 90	10000	Si
INPCI02	Infi 90 / Symphony	5000	Si
IIMCP01	Network 90 / Infi 90 / Symphony	10000	Si
IIMCP02	Infi 90 / Symphony	30000	Si
INICI01	Infi 90 / Symphony	10000	Si
INICI12	Infi 90 / Symphony	10000	Si
INICI13	Infi 90 / Symphony	30000	Si
INICI03	Infi 90 / Symphony	30000	Si

CAPÍTULO III
METODOLOGIA PARA LA SOLUCION DEL PROBLEMA

En este capítulo se realiza el planteamiento del problema, para lo cual primero se describe el problema, luego se fijan los objetivos del trabajo, se analiza y se propone una solución de ingeniería. Una vez propuesta una solución se fijará los alcances del trabajo. Finalmente se hace una síntesis del presente informe.

La automatización industrial es un campo de constante cambio tecnológico, esta sigue los pasos a las nuevas tecnologías emergentes con lo cual los servicios computacionales que brinda se vuelven cada vez más amplios y sofisticados y cubren así las crecientes necesidades de las empresas de los diversos rubros de la industria como son la minería, el petróleo y el gas.

En la actualidad hay cierto grupo de empresas que en el pasado optaron por usar un sistema de automatización que en su momento fue bueno, Sistema de Control Distribuido (DCS) Bailey que es el caso de estudio del presente informe, pero que ahora se ha vuelto una tecnología obsoleta debido a que el producto está descontinuado, esto es, ya no hay más desarrollo en software y no se cuenta con línea de fabricación del hardware del producto. Debido a este motivo es imperativo que las empresas que cuentan con el referido DCS hagan una migración urgente a una alternativa vigente y que cuente con un desarrollo constante de su sistema.

El primer paso para hacer la migración completa, tanto de hardware como de software, es hacer la migración de la interfaz hombre máquina (HMI) del sistema, esto es, migrar el HMI con toda la funcionalidad que este maneja, usando el hardware requerido para este caso (nuevas estaciones de operación e ingeniería).

3.1 Aplicación de estudio en el DCS Bailey

El sistema de control distribuido Bailey que es objeto de migración tiene el una topología similar a la que se muestra líneas más abajo.

La topología es una aproximación bastante general de la arquitectura detallada que se tiene en la planta minera donde se desarrolló las actividades de la migración, esta información detallada no puede ser divulgada debido a la política de confidencialidad que la empresa minera maneja para estos casos.

Para el monitoreo y el control desde la nueva plataforma de software se hace un

monitoreo y control (valga la redundancia) solo de los bloques necesarios del OPC90 Server. El criterio que determinó solo el uso de los bloques listados unas líneas más abajo es que son solo ese grupo de bloques los que usa el sistema Bailey en su lógica de control.

- AIL
- DIL
- MODSTAT
- MSDD
- RCM
- RMSC
- STN

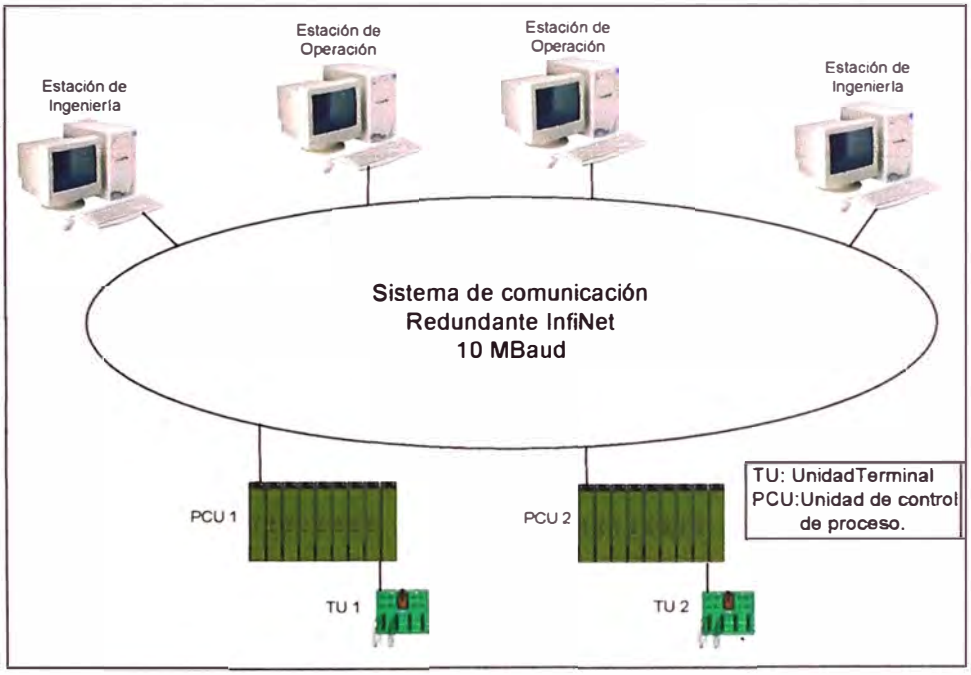

Figura 3.1 Esquema aproximado del Sistema Bailey a migrar

Se presenta una breve descripción de los bloques listados: [13]

AIL - Analog Input Loop: El bloque AIL OPC90 es usado para obtener la salida de reporte por excepción desde los bloques Analog Output/Loop (function code 30), Analog Point Definition (function code 70) y Enhanced Analog Point Definition (function code 158) de Bailey.

DIL - Digital Input Loop: El bloque DIL OPC90 es usado para obtener la salida de reporte por excepción desde los bloques Digital Output/Loop (function code 45) de Bailey.

MODSTAT - Module Status: El bloque MODSTAT OPC90 es usado para obtener información resumida del estado del módulo y reportes detallados de problemas en el modulo. Este bloque está diseñado para trabajar con todos los nodos Bailey y tipos de módulos.

MSDD - Multi State Device Driver: El bloque MSDD OPC90 es usado para obtener y controlar la salida de reporte por excepción de los bloques Multi State Device Driver (function Code 129) de Bailey.

RCM - Remote Control Memory: El bloque RMC OPC90 es usado para obtener y controlar la salida de reporte por excepción de los bloques Remote Control Memory (function code 62) de Bailey.

RMSC - Remote Manual Set Constant: El bloque RMSC OPC90 es usado para obtener y controlar la salida de reporte por excepción de los bloques Remote Manual Set Constant (function code 68) de Bailey.

STN - Station PID Control: El bloque STN OPC90 es usado para obtener y controlar la salida de reporte por excepción de los bloques Control Station (function codes 21, 22, 23 y 80) de Bailey.

La configuración que se tiene en el OPC90 Server tiene un Device Definition Block (DEVICE) llamado Antamina. El bloque DEVICE debe ser definido para cada interfaz Bailey al cual el OPC90 Server se va a comunicar, este bloque DEVICE debe ser configurado para que el OPC90 Server se pueda comunicar satisfactoriamente con el sistema Bailey.

La siguiente figura muestra una mejor vista de la configuración que se tiene para el OPC90 Server.

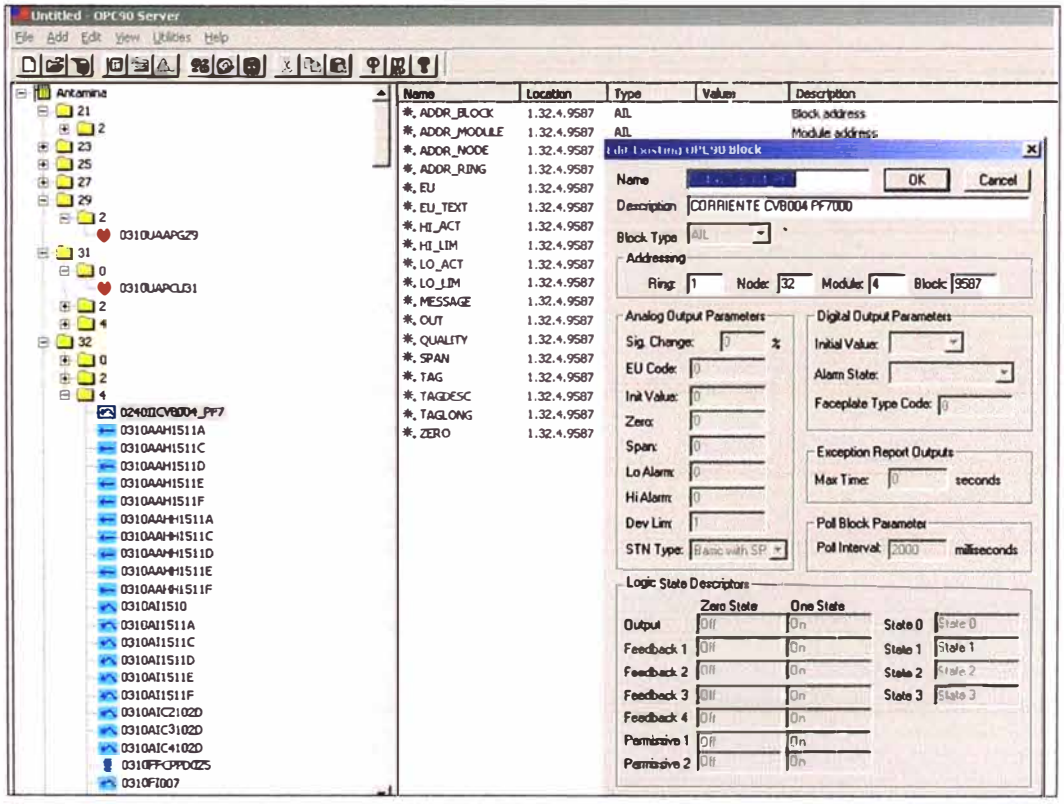

Figura 3.2 Vista de la configuración del OPC90 Server

Un nivel por debajo del bloque DEVICE se tienen unos grupos nombrados con número

(por ejemplo: 21, 23, 25, 27 y 29) los cuales representan los nodos que se están usando. Debajo de estos nodos se encuentran otros grupos los cuales representan a los módulos disponibles. Siguiendo con el árbol jerárquico, debajo de los módulos se encuentran los bloques de reporte por excepción, de estado del sistema, de lectura de configuración y sintonización entre otros, que el OPC90 Server usa para conectarse con sus pares del DCS Bailey, tal como se indica en la sección 2.3 OPC de Rovisys.

En esencia lo que se debe hacer desde la nueva plataforma de software es conectarnos al sistema Bailey a través de unas CIU (Communication Interface Unit) las cuales son Worstations que ejecutan la aplicación OPC90 Server que vienen a ser los servidores OPC y que cuentan con una tarjeta SCSI como canal de comunicación con el sistema Bailey. Del lado de la nueva plataforma se tendrán clientes OPC halando grupos de información del sistema Bailey, dicha información debe ser recibida y procesada para el monitoreo y control que se requiere.

3.2 Requerimientos del proyecto

Los objetivos que se tienen para el proyecto de migración son los siguientes:

- Implementar un sistema SCADA a través del cual se pueda hacer el monitoreo y control de la parte de planta que está siendo controlada por el sistema Bailey. La interfaz hombre máquina (HMI) del nuevo sistema deberá tener como mínimo la misma funcionalidad y su interfaz gráfica deberá ser muy parecida al del sistema Bailey.
- El sistema SCADA a implementar deberá contar con un mecanismo de redundancia de CPUs para aumentar la disponibilidad del sistema.

El objetivo de que la interfaz gráfica del nuevo sistema sea muy similar al del antiguo, es asegurar una fácil adaptación del personal de operaciones de la sala de control al nuevo sistema y así continuar con la operación regular de la planta sin tiempos muertos que se deban al cambio de interfaz gráfica.

El objetivo principal que se tiene desde la nueva plataforma es crear una base de datos que sea una réplica de los bloques de control y monitoreo indicados previamente en esta sección además de replicar su HMI, estas réplicas deben asegurar la misma funcionalidad del sistema anterior. En esta parte a nivel de aplicación no hay mucho por discutir en cuanto a su implementación, solo el hecho de cómo agrupar las señales provenientes del sistema Bailey, esta agrupación será de manera similar a los bloques de control del Bailey.

Para el caso del sistema de redundancia se tienen varias alternativas las cuales serán detalladas en la siguiente sección.

3.3 Alternativas de solución

Las alternativas que se tienen se enfocan principalmente en el mecanismo de

redundancia que se requiere. Una vez escogido el mecanismo de redundancia se presentará la arquitectura propuesta.

3.3.1 Redundancia en System Platform

La característica principal que permite realizar una configuración de redundancia en el System Platform es que el Application Server admite redundancia, la cual se realiza configurando los AppEngines y los WinPlatforms que los contienen. Esto es, se puede configurar un AppEngine en un nodo y su AppEngine Backup puede residir en otro nodo diferente.

Las configuraciones soportadas para poder realizar la redundancia son dos, las cuales se muestran a continuación. [14]

a) Conexión de red RMC - Par conectado por cable cruzado

Para octubre del 2009, la configuración de redundancia para un AppEngine del System Platform estaba fija en el sentido que siempre requería un par de plataformas y una conexión de red dedicada llamada RMC (Redundant Message Channel) vía cable cruzado Ethernet.

Si bien es cierto, esta es una alternativa que es sencilla y fácil de implementar, el funcionamiento de la conmutación por falla de una redundancia frecuentemente tropieza con los siguientes inconvenientes o inclusive problemas:

El número de plataformas habilitadas para redundancia dentro de una galaxia tiene que ser par por que la redundancia siempre requiere de un par.

El tiempo para una conmutación por falla exitosa y completa depende del número de AppEngines redundantes y del número de objetos ejecutando en cada AppEngine. Tanto el número de AppEngines y objetos crezca, la conmutación por falla puede tomar un largo tiempo para completarse.

Para un sistema grande, a menos que con parámetros optimizados y amplio recursos de sistemas disponibles, la tasa de éxito de la conmutación por falla es impredecible.

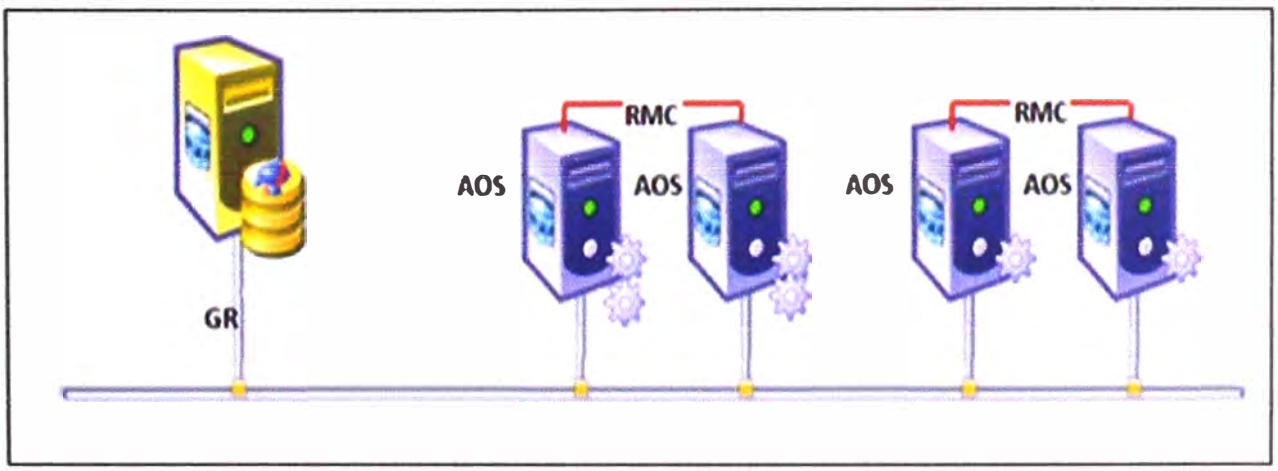

Figura 3.3 Conexión de red RMC - Par conectado por un cable cruzado

Para un sistema grande, particularmente cuando hay muchos AppEngines redundantes, problemas potenciales de funcionamiento pueden aparecer. Por ejemplo: alto uso de CPU, AppEngines fallando, etc. pueden ocurrir después de que todos los AppEngines han fallado sobre uno del par de la plataforma redundante. La conmutación por falla puede ocurrir cuando uno del par se cae inesperadamente o se apaga para mantenimiento programado.

b) Conexión de red RMC - Par conectado por Switch

Esta configuración es más reciente respecto a la anterior, dado que el uso de la tecnología de red Auto-MDIX la cual se ha venido convirtiendo en un estándar, el requerimiento de cableado cruzado Ethernet para el RMC se ha convertido en obsoleto. Cuando las interfaces de red cuentan con esta tecnología, dos de las restricciones de configuración de la RMC se eliminan:

- El cable cruzado Ethernet
- La conexión entre un par

En su lugar, todas las plataformas con RMC pueden ser conectadas vía una única red a través de un switch.

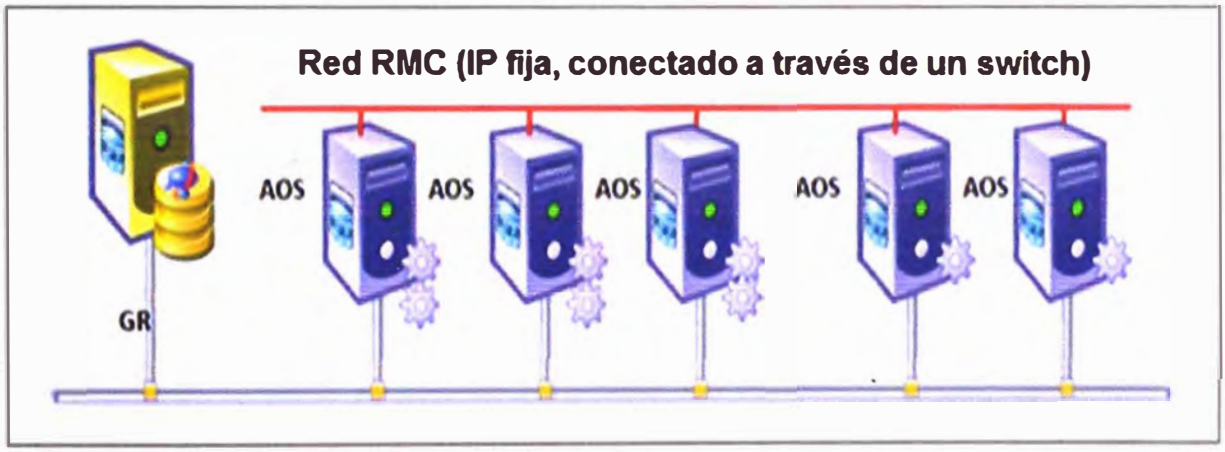

Figura 3.4 Conexión de red RMC vía un switch

Consideraciones a tener en cuenta

Se debe asegurar que la tecnología Auto-MDIX está ya embebida en la interfaz de red y la configuración en sitio estar hecha correctamente para asegurar un funcionamiento apropiado de esta característica.

Se debe asegurar que la velocidad de la red RMC es de al menos 16 Mbps, y que la red es dedicada para RMC, no a cualquier otro tráfico de red.

Si la probable distancia del cable de red RMC (entre dos nodos redundantes cualesquiera) es más largo que 91 metros, considerar en vez de ello el uso de fibra óptica.

Para una red redundante RMC, se debe aplicar hardware apropiado, como las tarjetas de

interfaz de red, switch, como también el software requerido (driver).

La significancia de estos cambios es que ahora, para cada plataforma con redundancia habilitada, su plataforma compañera no está limitada a única designada por la conexión RMC vía el cable cruzado. Todas las otras plataformas en la red RMC son sus compañeros como lo permite la nueva red RMC.

La siguiente matriz es un ejemplo de cómo los AppEngines backup pueden ser distribuidos entre las plataformas compañeras para cada plataforma que contiene AppEngines primarios.

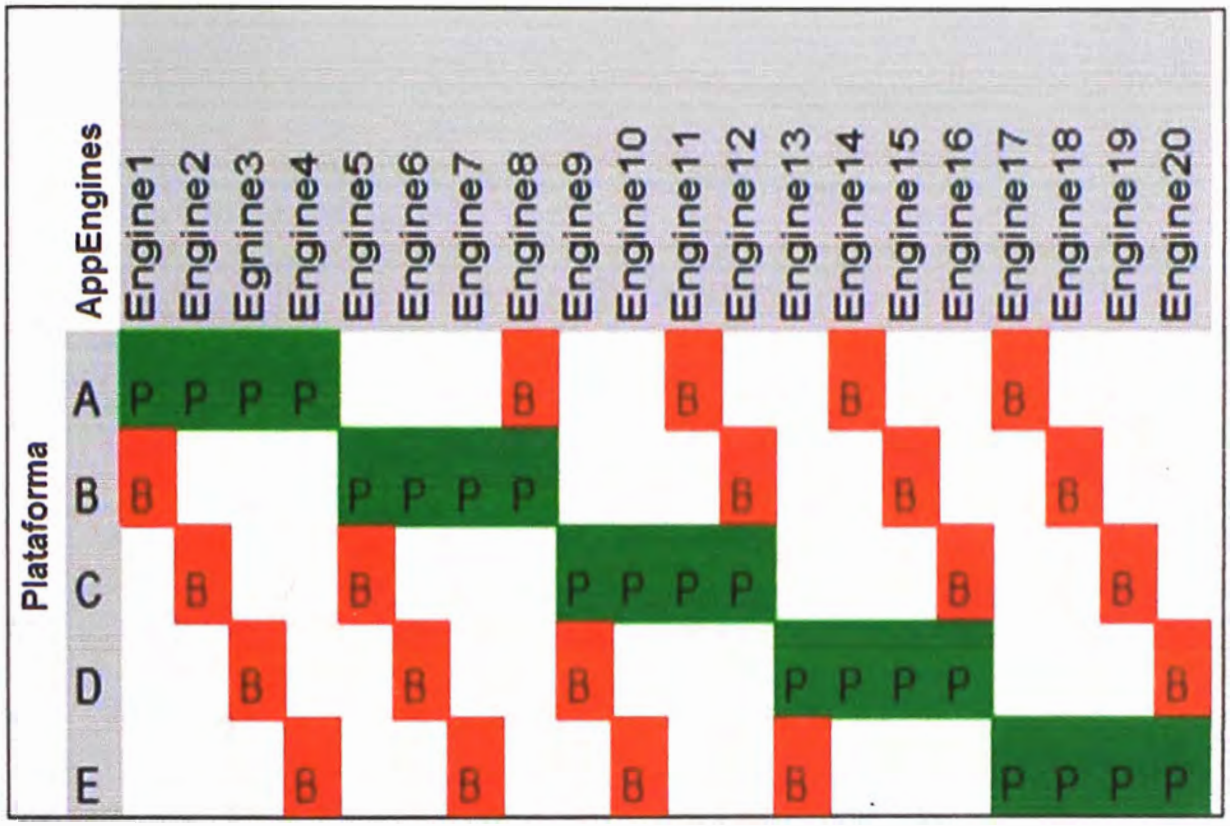

Figura 3.5 Distribución de AppEngines redundantes en múltiples compañeros

3.3.2 Redundancia usando virtualización

Beneficios de la virtualización [15]

- Costo de hardware más bajo.
- Mayor rendimiento.
- Mayor escalabilidad.
- Disponibilidad más alta.
- Administración más simple de software y sistemas operativos.
- Mayor confiabilidad.
- Mayor protección contra desastres.
- Elimina las dependencias entre el hardware físico y el software, esto da más opciones para administrar mejor las aplicaciones, servidores y equipamiento.

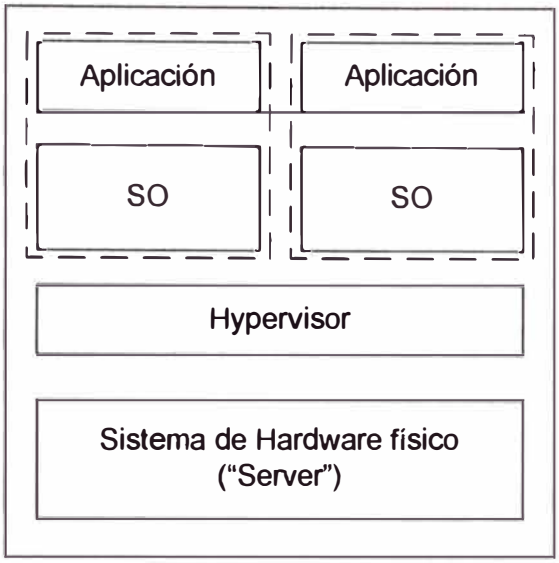

Figura 3.6 Entorno de virtualización

Tecnología de virtualización [15]

Las máquinas físicas pueden ser "mirrored" (crear copias iguales) usando menos hardware (para reducir costos)

Las aplicaciones pueden ser virtualizadas para tiempo de actividad y rendimiento.

El "*Wonderware System Platform*" está probado con soluciones basadas en Hyper-V y VMWare para el usuario industrial de HMI y control para recuperación por desastre y alta disponibilidad.

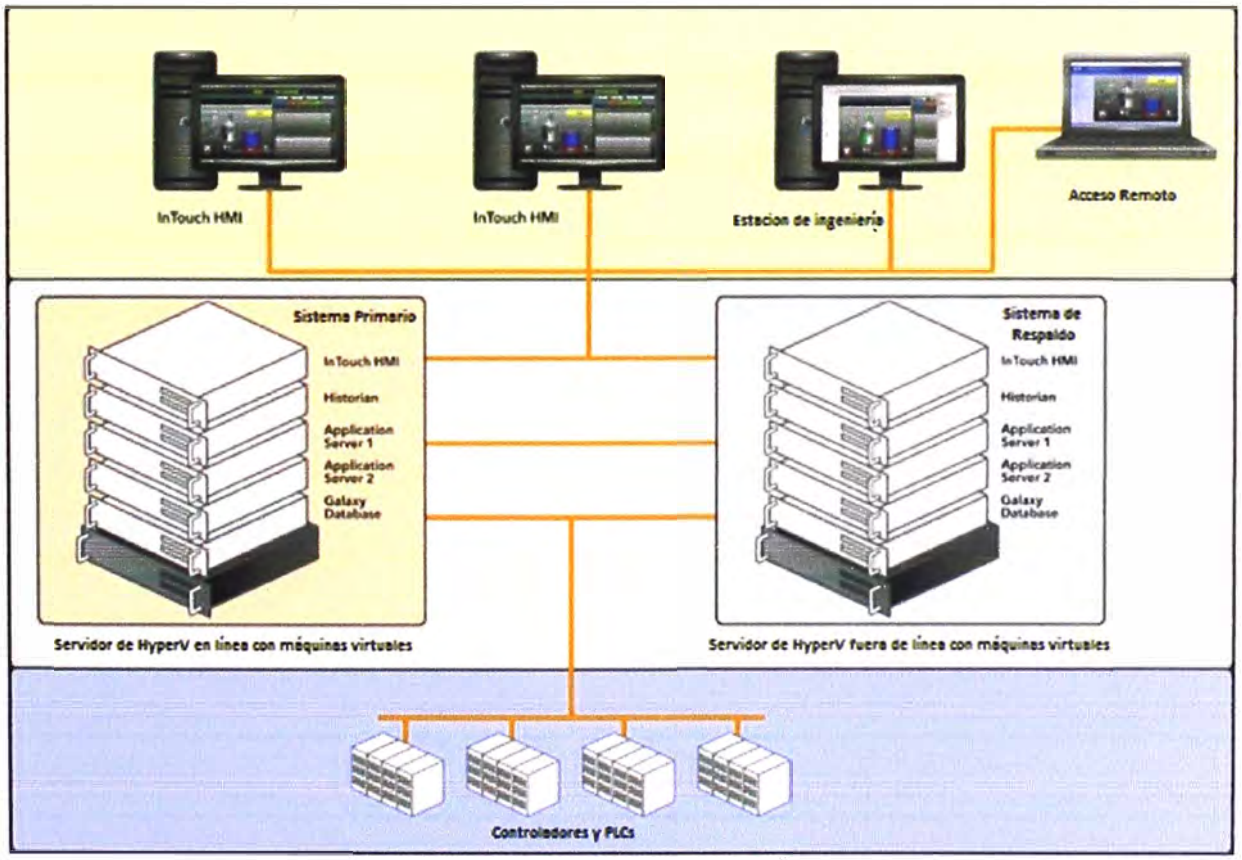

Figura 3.7 Tecnología de virtualización para el System Platform

Para que se pueda trabajar en un ambiente virtualizado como el mostrado en la figura anterior el servidor que contenga a las maquinas virtuales debe soportar todos los requerimientos de las aplicaciones que ejecuta cada máquina virtual, como son el número de núcleos y la cantidad de memoria RAM. El requerimiento del número de núcleos que requiere determinada aplicación, depende del tamaño del sistema, el cual se basa en el número de señales de entrada/salida (IO) y en el número de nodos que tiene. A continuación se presenta los números aproximados de IO y el número de nodos que definen los sistemas pequeños, medianos y grandes: [16]

- Sistema pequeño: 1 - 5k IOs por nodo - 1 a 3 nodos
- Sistema mediano: 5k - 50k IOs por nodo - 4 a 8 nodos
- Sistema grande: 50k - 400k IOs por nodo - Más de 8 nodos

Wonderware tiene recomendaciones mínimas de configuración para la implementación del sistema de acuerdo a su tamaño. Para el caso de sistemas medianos y grandes la recomendación mínima de Wonderware es la siguiente:

TABLA N° 3.1: Requerimientos mínimos del sistema para nodos del System Platform

	Núcleos	RAM (GB)	Almacenamiento (GB)
GR	4	4	250
Historian	4	4	500
Application Server	2 a 4	4	100
RDS Servers	4 a 8	4 a 8	100
Information Server	4	4	100
Historian Clients	2	4	100

Además es recomendable que el servidor físico que contiene las máquinas virtuales debe tener un 25% más de recursos que el mínimo necesario. Esto es, si el conjunto de aplicaciones que se ejecutan en las distintas máquinas virtuales que el servidor contiene necesita como mínimo 20 núcleos, el servidor debe tener como mínimo 25 núcleos. Este tipo de requerimientos hace que el hardware del servidor físico tenga características especiales y por ende se debe tener especial cuidado con el dimensionamiento del mismo antes de su adquisición.

3.4 Solución a implementar

La solución a implementar para el presente informe es la redundancia en el System Platform, específicamente la opción **3.3.1.2 Conexión de red RMC (Redundant Message Channel) - Par conectado por Switch.** Esta decisión se tomó debido a los siguientes factores:

- La Migración que se solicitó era un proyecto piloto el cual la iba a servir al usuario final para evaluar la alternativa del System Platform.

- Ya se contaba con un grupo de servidores y estaciones de trabajo para la implementación de la migración del sistema.
- El tiempo de ejecución era relativamente corto.

Entonces, dado que se eligió la alternativa de la redundancia en el System Platform a continuación se profundiza más sobre el tema para su posterior implementación.

3.4.1 Redundancia a nivel del Wonderware IAS

Se puede configurar y ejecutar el Application Server en un entorno de redundancia. En el Application Server, dos tipos de redundancias aseguran la continuidad de operación en tiempo de ejecución. Se puede configurar redundancia en: [17]

Emparejado de objetos AppEngine para fallas de software o de computador: Se debe configurar un AppEngine y dos WinPlatforms, ambos.

Comunicaciones de adquisición de datos a uno o más PLCs: Se debe configurar dos DIObjects (fuentes de datos) y un RedundantDIObject.

Una conmutación por falla (failover) es la condición durante la cual las operaciones en tiempo de ejecución son movidas desde un componente crítico a otro. La conmutación por falla puede ocurrir debido a condiciones de falla o puede ser forzado manualmente, llamado una conmutación por falla forzado (forced failover).

Configuración de redundancia de AppEngine [17]

Los AppEngines Primary/Backup forman un par redundante. La infraestructura ArchestrA generará el AppEngine backup automáticamente cuando la redundancia sea habilitada para un AppEngine. La jerarquía de ApplicationObjects puede ser solamente asignado al AppEngine primario. Los AppEngines primario y backup necesitan ser asignados a plataformas habilitadas para redundancia, las cuales pueden estar descargadas por separado (en distintos nodos).

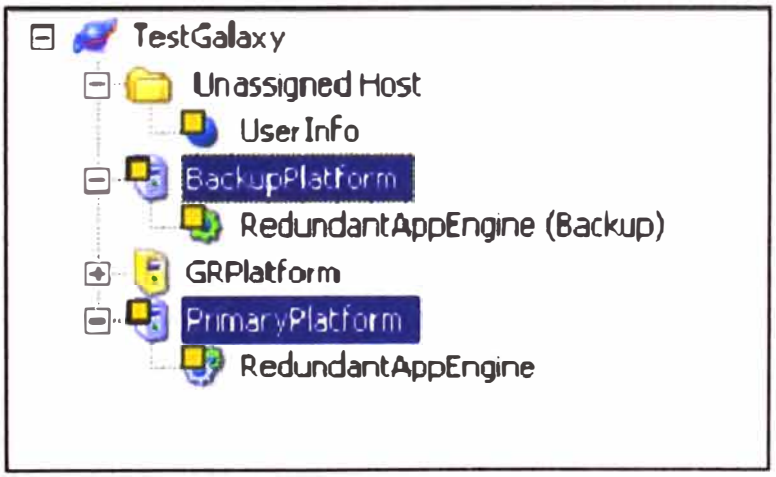

Figura 3.8 Redundancia de AppEngines en el entorno del IDE

Para adquisición de datos, los DIObjects Primary/Backup (fuentes de dato) deben ser creados por separado, configurados y descargados. También, se puede crear, configurar

y descargar un RedundantDIObject para controlar las conmutaciones por falla entre los dos objetos fuentes de dato.

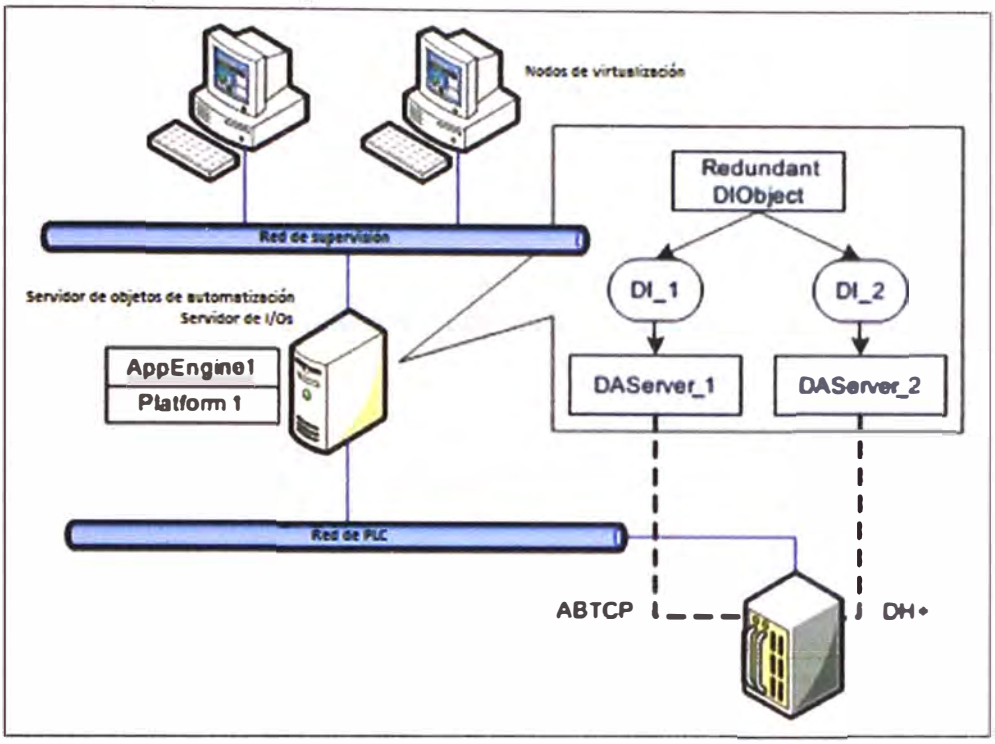

Figura 3.9 Redundancia de DIObjects (fuente de datos)

Cuando se configura redundancia, se debe configurar el objeto primario (Primary) y el objeto de respaldo (Backup).

Figura 3.10 Objetos Primary y Backup

Objeto Primario: El objeto central o principal que provee la funcionalidad durante el tiempo de ejecución (runtime). Para AppEngines, este es el objeto que se habilita para redundancia. Para adquisición de datos, este es el DIObject que se intenta usar primero como fuente de datos en tiempo de ejecución.

Objeto de respaldo: El objeto que provee la funcionalidad del objeto Primary cuando el objeto Primary falla. Para AppEngines este es el objeto creado por la infraestructura de ArchestrA cuando el objeto Primary es habilitado para redundancia. Para adquisición de datos, este es el DIObject que no se intenta usar primero como la fuente de datos en tiempo de ejecución.

Redundancia durante tiempo de ejecución

Cuando se descarga estos objetos en un entorno de tiempo de ejecución, la configuración de los objetos se convierte en objeto Activo y objeto Standby.

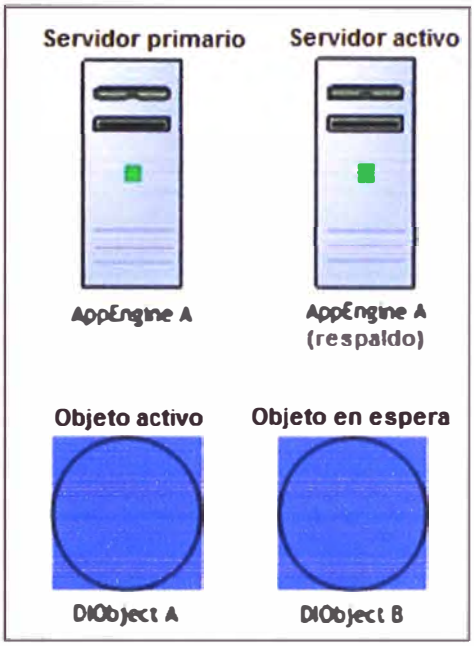

Figura 3.11 Objetos Activo y Standby

Objeto activo: El objeto que está actualmente ejecutando funciones. Para AppEngines, es el objeto que está conteniendo y ejecutando los ApplicationObjects. Para adquisición de datos, es el objeto que está proveyendo información de dispositivos de campo a través del RedundantDIObject.

Objeto en espera: El objeto que está esperando por una falla en el objeto activo o por una conmutación por falla forzado. Para AppEngines es el objeto que monitorea el estado del AppEngine activo. Para adquisición de información, es el objeto que no está proveyendo información de dispositivos de campo a través del RedundantDIObject.

En el ambiente de redundancia de AppEngine, los objetos Active y Standby monitorean uno el estado del otro y conmuta cuando la condición de falla ocurre.

En el ambiente de adquisición de datos, el RedundantDIObject monitorea el estado de los dos DIObject fuente de datos, y maneja la conmutación de los objetos de activo a standby.

Las WinPlatforms que contienen los AppEngines con redundancia habilitada, deben ser descargadas sobre computadores ejecutando el mismo sistema operativo.

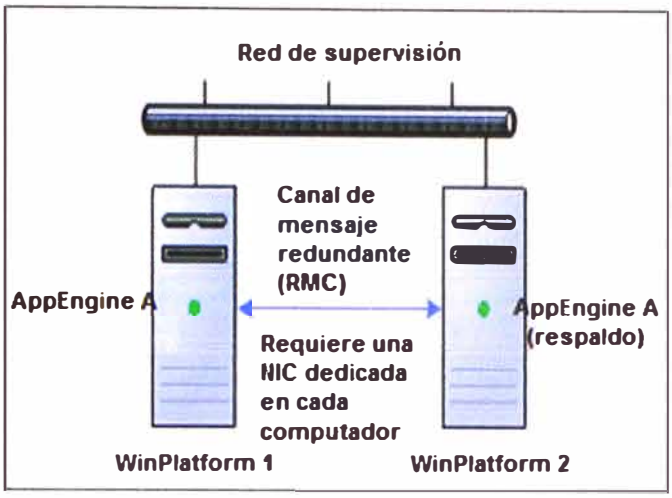

Figura 3.12 RMC requerido para AppEngines con redundancia habilitada

Cada computador de producción que contiene un AppEngine con redundancia habilitada debe tener un mínimo de dos tarjetas de red. Una NIC es para la red de supervisión y red de PLC, si el computador tiene solamente dos tarjetas de red. La otra debe ser para una conexión Ethernet dedicada entre computadores para el canal de mensaje de redundancia (RMC - Redundancy Message Channel).

3.5 Implementación de la solución

Para la implementación de la redundancia a nivel del Application Server se usa una arquitectura de red en estrella, teniendo como punto concentrador un switch. Esta topología es usada debido a que se requiere que todos los nodos estén convenientemente comunicados (es decir, que sea posible que un par cualquiera de nodos puedan verse entre sí sin tener que pasar por un tercer nodo), y también porque mediante el switch podemos administrar las dos redes que se usan (una red de supervisión y control y una red de redundancia - RMC) mediante el uso de dos VLANs (Virtual Local Area Network).

En esta arquitectura de red se usa 6 computadores de tipo estaciones de trabajo y 2 computadores de tipo servidor. De las 6 estaciones de trabajo, 4 son usadas como AOServers (Application Object Servers - Servidores de objetos de aplicación) y 2 son usadas como clientes de interfaz gráfica. De los 2 servidores, se tiene uno para que cumpla la función de almacenar la base de datos de configuración del sistema (nodo GR) y el otro para almacenar y proveer el servicio de datos históricos a los nodos con aplicaciones clientes. En la red implementada de 100 Mbps se usa patchcords de cobre. Dado el producto que se está usando para la implementación del sistema, y dado que el número de IOs es de 52070, para obtener un buen rendimiento del sistema es recomendable tener el nodo GR en un servidor y el nodo de Historian en otro servidor, por separado, aunque para ciertas aplicaciones el GR y el Historian pueden coexistir en el mismo nodo, como es el caso de aplicaciones muy pequeñas (menor a 5k IOs).

Para el caso de las estaciones de trabajo que funcionan como AOServers, con el número de IOs que el sistema debe manejar, el número mínimo de AOServers que el sistema debe tener es 4, esto para poder implementar la funcionalidad de redundancia, y es que el soporte de la marca Wonderware indica que cada AOServer debe manejar aproximadamente 25000 IOs como máximo para garantizar un buen rendimiento del sistema. Entonces, para el caso en que 2 de los 4 AOServer tengan una condición de falla, los 2 AOServer sin condición de falla deben asumir la carga (para este caso 26035 IOs cada uno) que los otros dos servidores no pueden manejar por su condición de falla. Se podría usar un par más de servidores para tener menos carga en cada AOServer pero de ninguna manera se podría usar menos de 4 AOServer. Entonces, con lo expuesto se tiene que se ha usado el mínimo número de AOServer.

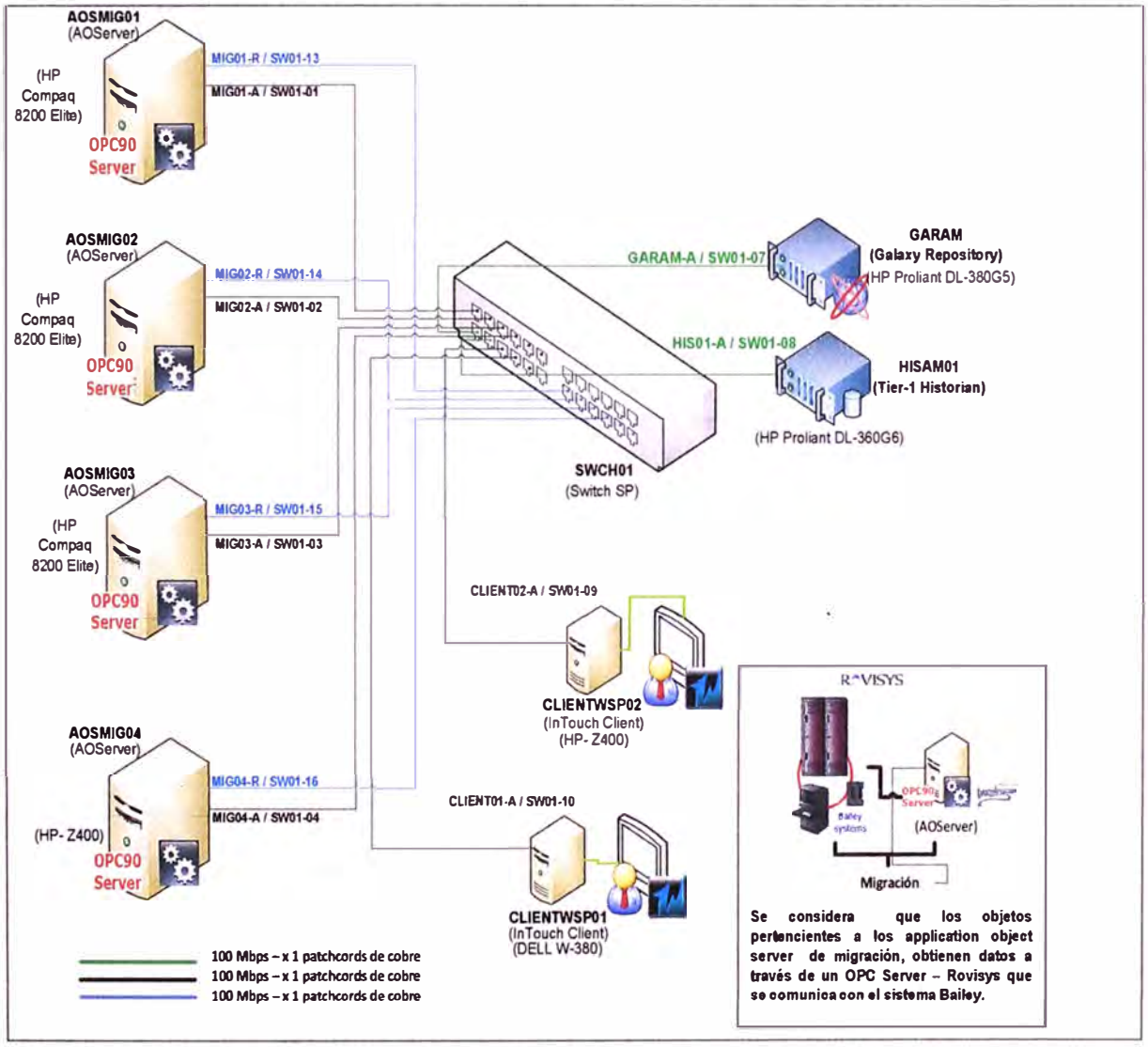

Figura 3.13 Arquitectura de Red usada para la implementación de la migración

Para tener una idea más clara del hardware sobre el cual se construye la plataforma, se muestra el detalle de los computadores de tipo Servidor y tipo Estación de trabajo.

TABLA N° 3. 2: Detalle de Servidores y Estaciones de trabajo - sistema operativo, memoria RAM y dimensión de las particiones C y D

ITEM	DESCRIPCION	CANTIDAD	NOMBRE	SISTEMA OPERATIVO	32/64 bits	RAM	Partición C:	Partición D:
	ELEMENTOS FUNCIONALES SYSTEM PLATFORM							
1	Workstation HP Compaq 8200 Elite	1	**AOSMIG01**	Windows 7 Professional SP1	64 bit	8 GB	80 GB	385 GB
2	Workstation HP Compaq 8200 Elite	1	**AOSMIG02**	Windows 7 Professional SP1	64 bit	8 GB	80 GB	385 GB
3	Workstation HP Compaq 8200 Elite	1	**AOSMIG03**	Windows 7 Professional SP1	64 bit	8 GB	80 GB	385 GB
4	Workstation HP Z-400 (Nuevo)	1	**AOSMIG04**	Windows 7 Professional SP1	64 bit	8 GB	80 GB	365 GB
5	Workstation DELL W380	1	**CLIENTWSP01**	Windows 7 Professional SP1	64 bit	8 GB	80 GB	152 GB
6	Workstation HP Z-400	1	**CLIENTWSP02**	Windows 7 Professional SP1	64 bit	8 GB	80 GB	216 GB
7	Servidor HP Proliant S-Buy DL380G5	1	**GARAM**	Windows Server 2008 R2 Standard SP2	64 bit	4GB	68.3GB	341 GB
8	Servidor HP Proliant S-Buy DL360G6	1	**HISAM01**	Windows Server 2008 R2 Standard SP2	64 bit	2GB	78.1 GB	154 GB

TABLA N° 3. 3: Detalle de Servidores y Estaciones de trabajo - procesadores y tarjetas de video

ITEM	DESCRIPCION	CANTIDAD	PROCESADOR	VIDEO
	ELEMENTOS FUNCIONALES SYSTEM PLATFORM			
1	Workstation DELL W-380	1	Intel Pentium 4 @3.2 GHz	Nvidia Quadro NVS285 - 512MB
2	Workstation HP Z-400	2	Intel XEON W3530 @2.80 GHz	Nvidia GForce 9500 - GT 1024 MB
3	Workstation HP Compaq 8200 Elite SFF PC	3	Intel (R) Core (TM) i7 2600 @3.40 GHz	Intel (R) HD Graphics Family - 1696 MB
4	Servidor HP Proliant S-Buy DL360G6	1	Intel XEON E5506 @2.13 GHz	ATI ES 1000 - 64 MB
5	Servidor HP Proliant S-Buy DL380G5	1	Intel (R) Xeon CPU 5150 @ 2.66 GHz	ATI ES 1000 - 32 MB

En el entorno de desarrollo integrado del System Platform se configuran las plantillas o "Templates" para los AutomationObjects, los cuales se pueden ver en el cuadro de vista "Template Toolbox" del entorno ArchestrA IDE.

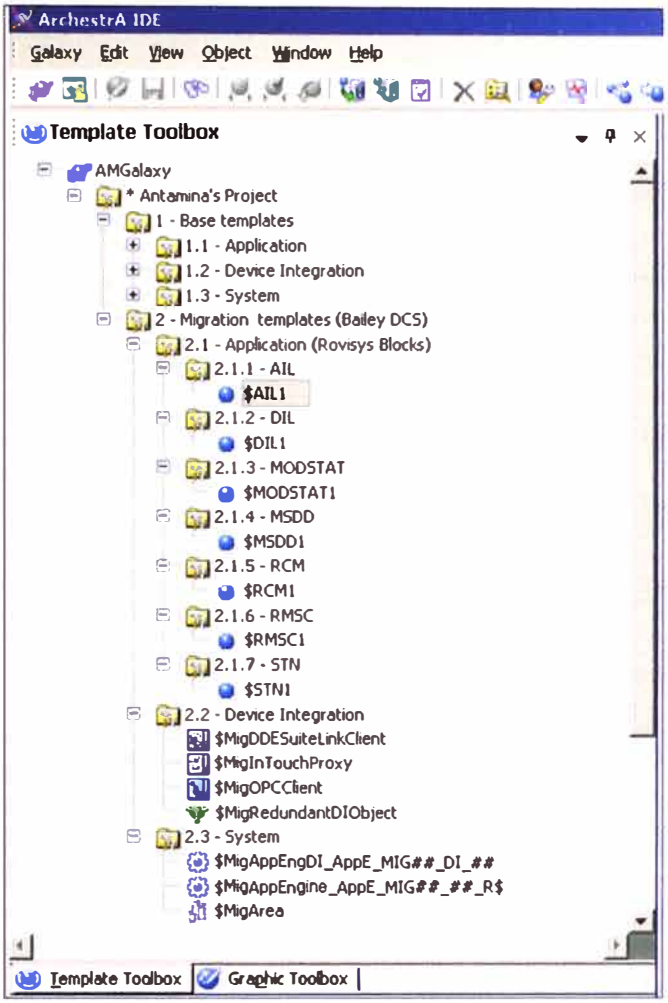

Figura 3.14 Vista de los objetos plantillas para la migración

En la figura anterior se muestra los bloques plantillas implementados en el nuevo sistema, los cuales son unas réplicas de los bloques de control que usa el sistema Bailey y que se listan a continuación.

- AIL
- DIL
- MODSTAT
- MSDD
- RCM
- RMSC
- STN

Para la implementación se representa cada nodo de la arquitectura con un objeto WinPlatform, los cuales son nombrados con el mismo nombre de los nodos. Se tiene que los objetos WinPlatforms de la aplicación son:

- AOSMIG01
- AOSMIG02
- AOSMIG03
- AOSMIG04
- GARAM
- HISAM01
- CLIENTWSP01
- CLIENTWSP02

A su vez, cada objeto WinPlatform que tiene la función de ser un AOServer contiene objetos del tipo AppEngine, los cuales contendrán y ejecutarán a su vez los objetos de aplicación que simbolizan los bloques del sistema Bailey. Son precisamente estos AppEngines los que son configurados para la implementación de la redundancia. También, ya que estos mismos nodos se usan como nodos de adquisición de datos, contienen un AppEngine que contiene a su vez dos DIObjects los cuales son las fuentes de datos para los objetos de aplicación.

TABLA N° 3.4: Detalle de AppEngines que contienen los nodos AOServers

	AOSMIG01	**AOSMIG02**	**AOSMIG03**	**AOSMIG04**
AppEngines primarios	AppE_MIG01_01_R2	AppE_MIG02_01_R1	AppE_MIG03_01_R4	AppE_MIG04_01_R3
	AppE_MIG01_02_R2	AppE_MIG02_02_R1	AppE_MIG03_02_R4	AppE_MIG04_02_R3
	AppE_MIG01_03_R2	AppE_MIG02_03_R1	AppE_MIG03_03_R4	AppE_MIG04_03_R3
AppEngine de Comunicación	AppE_MIG01_DI_01	AppE_MIG02_DI_01	AppE_MIG03_DI_01	AppE_MIG04_DI_01
AppEngines de Backup-	AppE_MIG02_01_R1 (Backup)	AppE_MIG01_01_R2 (Backup)	AppE_MIG04_01_R3 (Backup)	AppE_MIG03_01_R4 (Backup)
	AppE_MIG02_02_R1 (Backup)	AppE_MIG01_02_R2 (Backup)	AppE_MIG04_02_R3 (Backup)	AppE_MIG03_02_R4 (Backup)
	AppE_MIG02_03_R1 (Backup)	AppE_MIG01_03_R2 (Backup)	AppE_MIG04_03_R3 (Backup)	AppE_MIG03_03_R4 (Backup)

Como se puede ver en la tabla anterior cada nodo AOServer contiene tres objetos AppEngine primario, tres objetos AppEngines de backup y uno de comunicación. Se debe notar que la redundancia implementada es de pares, es decir, ante una falla de aplicación o de hardware del AOSMIG01, todos los AppEngines y los objetos de aplicación que se ejecutan en los mismos pasan a ejecutarse en el AOSMIG02 y viceversa. Lo mismo aplica para el par formado por AOSMIG03 y AOSMIG04.

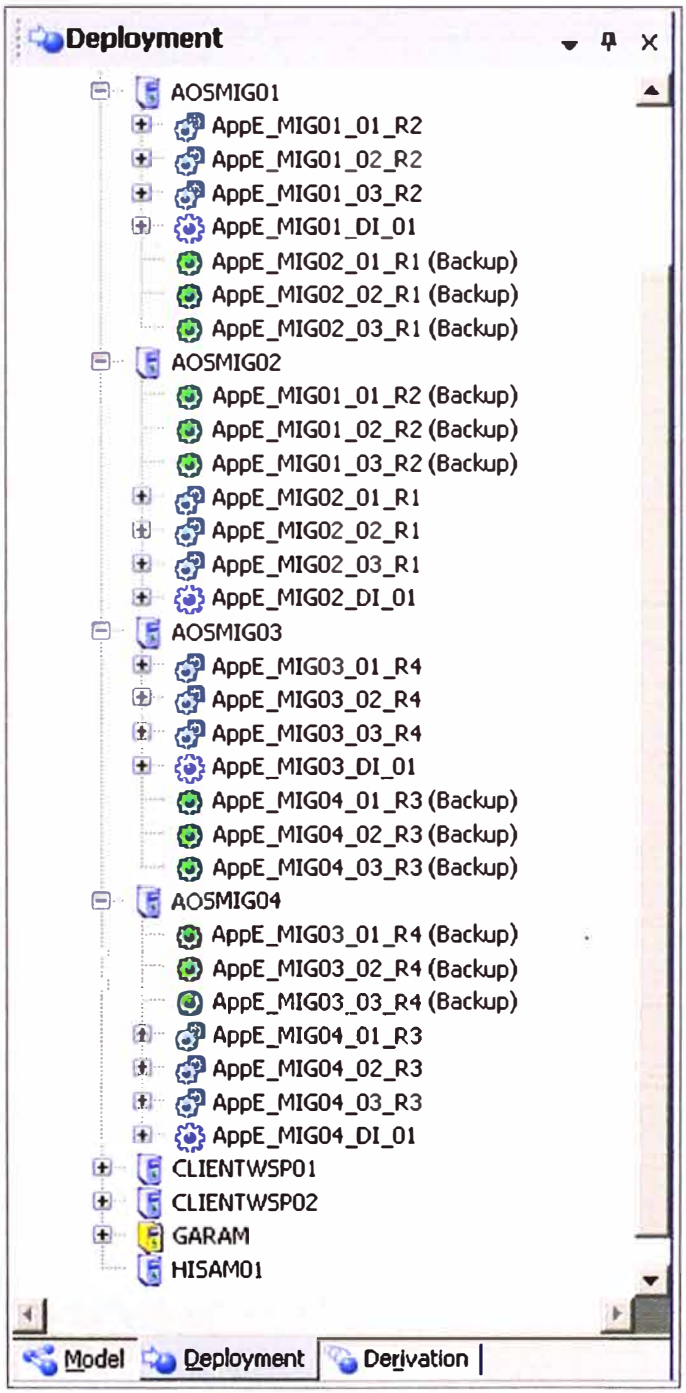

Figura 3.15 Plataformas configuradas en el System Platform

Dentro de cada AppEngine con redundancia se tiene un objeto del tipo RedundantDIObject, este objeto apunta a dos fuentes de datos, una en el mismo nodo y la otra en un nodo diferente, que para este caso se trata de su nodo par (AOSMIG01 con

AOSMIG02 y AOSMIG03 con AOSMIG04). Los objetos fuentes de datos se encuentran en los AppEngines de comunicación, y los objetos de comunicación redundantes - RedundantDIObject se encuentran dentro de cada AppEngine primario.

TABLA N° 3.5: Objetos de comunicación

AppEngine de comunicación	DIObject
AppE_MIG01_DI_01	DSL_MIG01_DI01
	DSL_MIG01_DI02
AppE_MIG02_DI_02	DSL_MIG02_DI01
	DSL_MIG02_DI02
AppE_MIG03_DI_03	DSL_MIG03_DI01
	DSL_MIG03_DI02
AppE_MIG04_DI_04	DSL_MIG04_DI01
	DSL_MIG04_DI02

TABLA N° 3.6: Objetos de comunicación redundante - RedundantDIObject por AppEngine

AppEngine primario	RedundantDIObject
AppE_MIG01_01_R2	OPC_MIG01RDI_01
AppE_MIG01_02_R2	OPC_MIG01RDI_02
AppE_MIG01_03_R2	OPC_MIG01RDI_03
AppE_MIG02_01_R1	OPC_MIG02RDI_01
AppE_MIG02_02_R1	OPC_MIG02RDI_02
AppE_MIG02_03_R1	OPC_MIG02RDI_03
AppE_MIG03_01_R4	OPC_MIG03RDI_01
AppE_MIG03_02_R4	OPC_MIG03RDI_02
AppE_MIG03_03_R4	OPC_MIG03RDI_03
AppE_MIG04_01_R3	OPC_MIG04RDI_01
AppE_MIG04_02_R3	OPC_MIG04RDI_02
AppE_MIG04_03_R3	OPC_MIG04RDI_03

Cada objeto de tipo RedundantDIObject dentro de AOSMIG01 tienen como fuente de datos primario al objeto de comunicación DSL_MIG01_DI01 (que se ejecuta en el mismo AOSMIG01) y como fuente de datos secundario al objeto de comunicación DSL_MIG01_DI02 (que se ejecuta dentro de AOSMIG02). A su vez, cada objeto de tipo RedundantDIObject dentro de AOSMIG02 tienen como fuente de datos primario al objeto

de comunicación DSL_MIG02_DI01 (que se ejecuta en el mismo AOSMIG02) y como fuente de datos secundario al objeto de comunicación DSL_MIG02_DI02 (que se ejecuta dentro de AOSMIG01). La misma analogía se cumple para las fuentes de datos de AOSMIG03 y AOSMIG04.

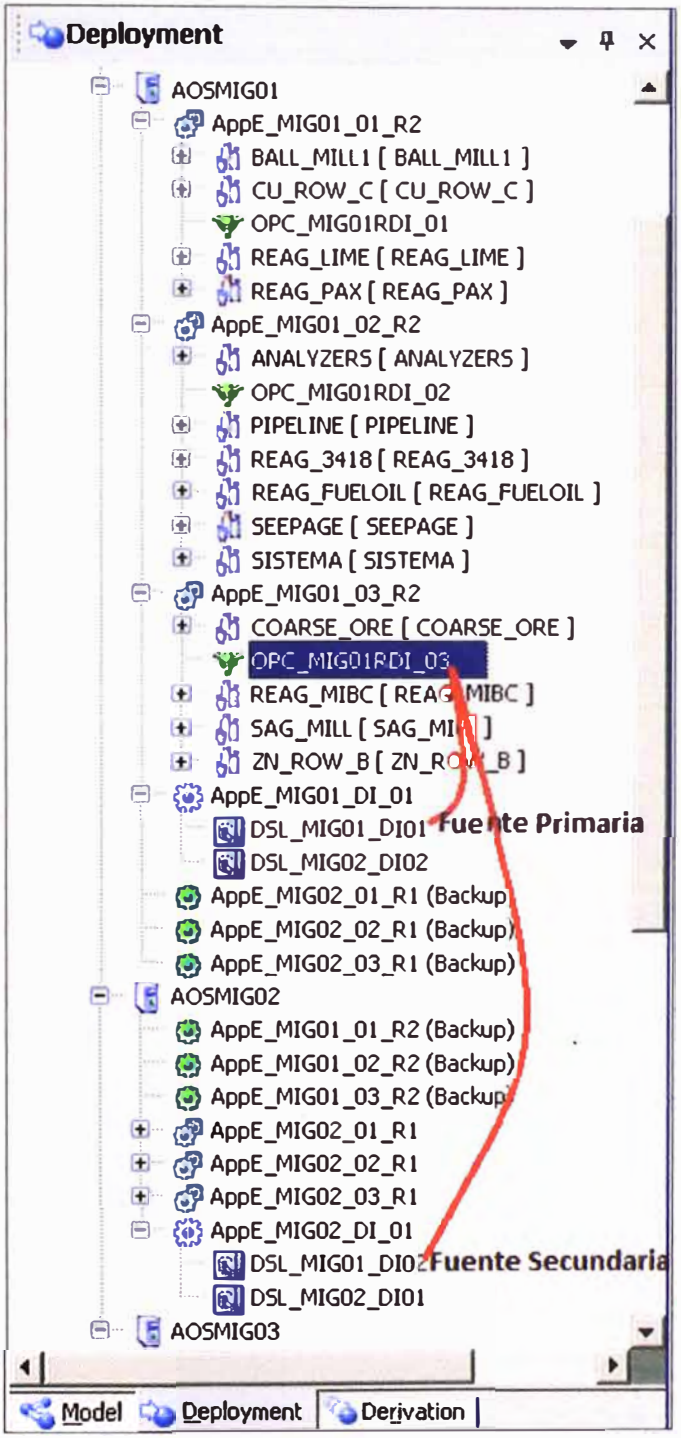

Figura 3.16 RedundantDIObject apuntando a sus dos fuentes de datos

Como ejemplo se tiene para el objeto de comunicación redundante OPC_MIG01RDI_03 que su fuente de datos primaria es el objeto de comunicación DSL_MIG01_DI_01 que se ejecuta en el AppEngine AppE_MIG01_DI_01 dentro del AOSMIG01 y su fuente de datos secundaria es el objeto de comunicación DSL_MIG01_DI_02 que se ejecuta en el

AppEngine AppE_MIG01_DI_02 dentro de AOSMIG02.

Como se ha mencionado líneas más arriba, dentro de los AppEngines se ejecutan los objetos de aplicación, estos objetos de aplicación están agrupados por un objeto contenedor del tipo Área dentro de los AppEngines, este objeto Área, como su nombre lo indica, representa un área determinada de la planta industrial.

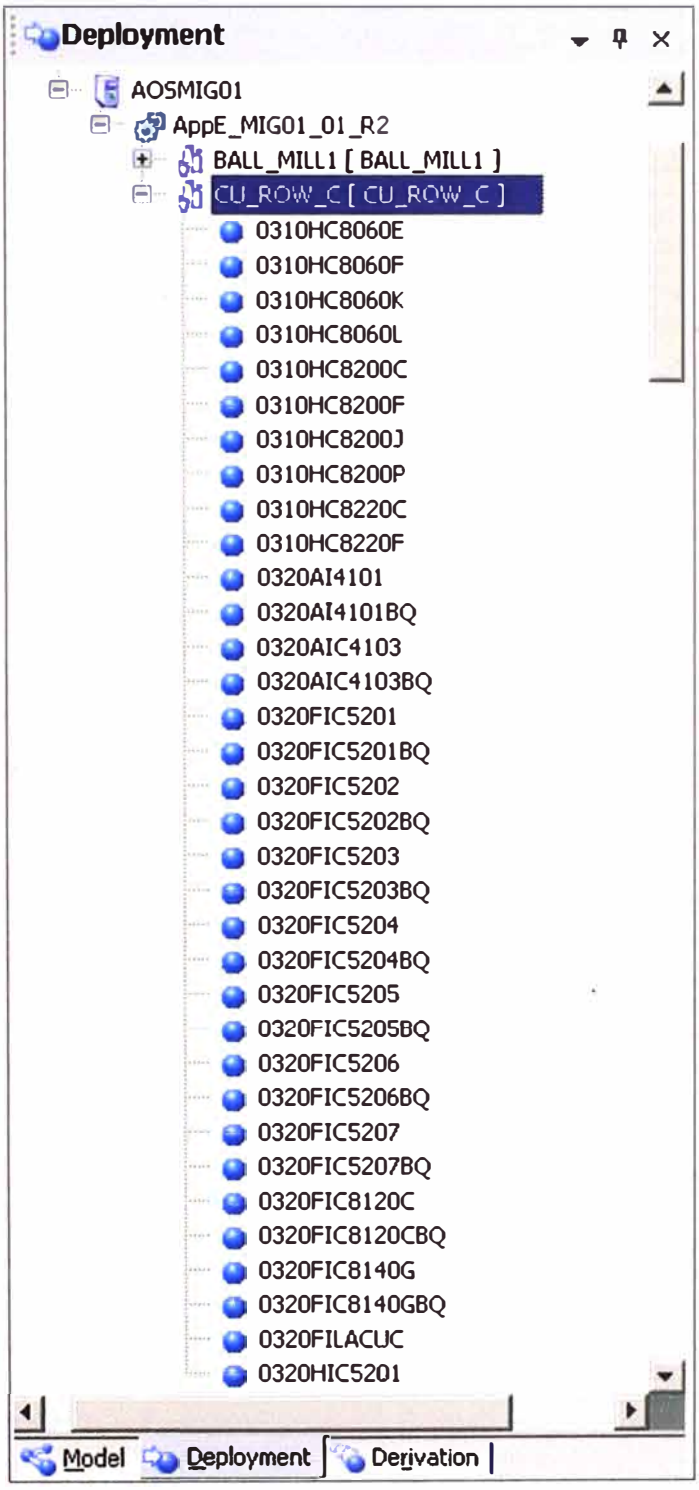

Figura 3.17 Objetos de aplicación contenidos dentro del objeto área CU_ROW_C

En el recuadro de la vista Model se puede apreciar la agrupación de las áreas que la planta presenta y los cuales para nuestro caso de estudio se muestran a continuación:

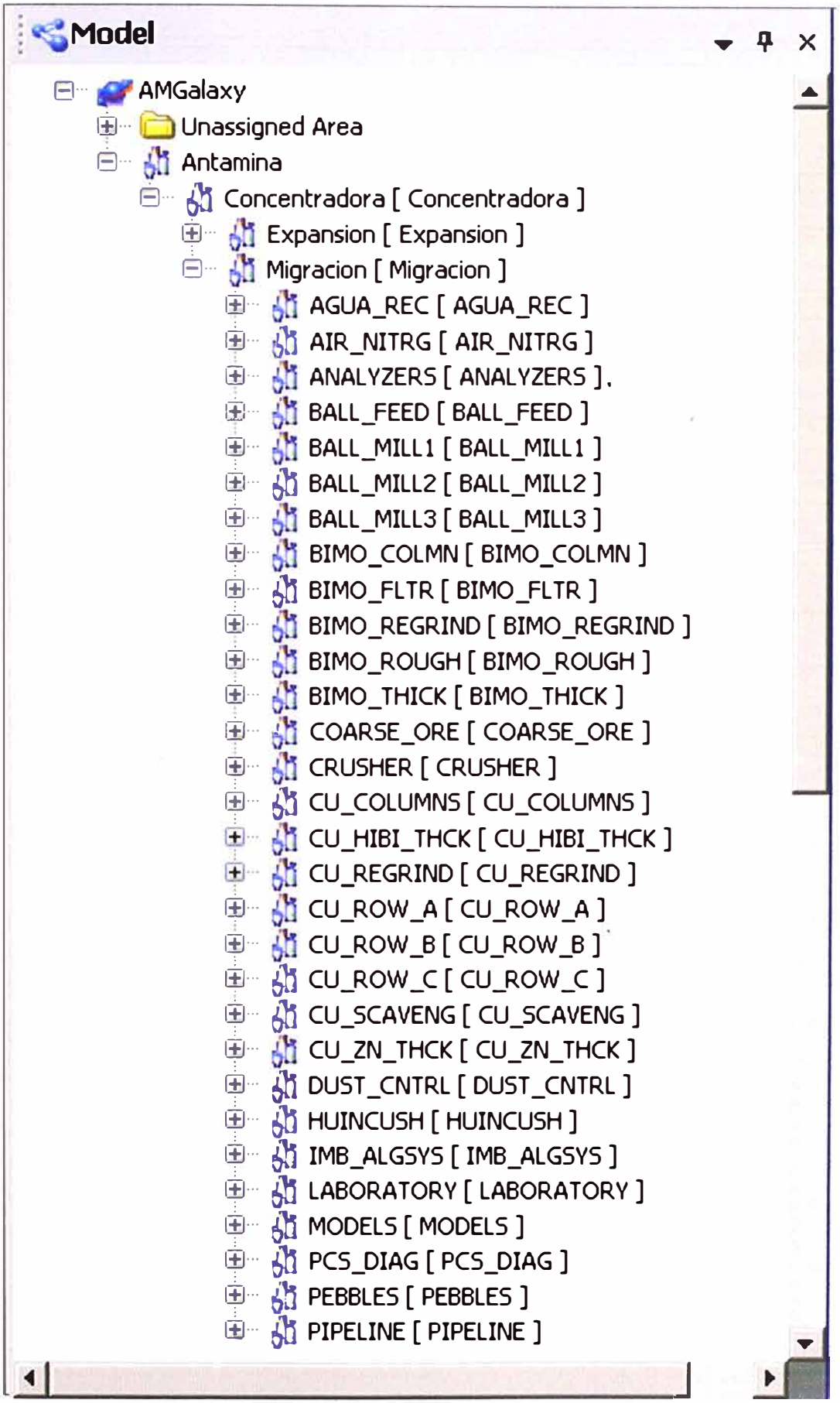

Figura 3.18 Áreas que forman parte del proyecto de migración

A continuación se muestra un grupo de 5 páginas gráficas de la planta minera las cuales representan la Interfaz Hombre Máquina o HMI, mediante las cuales se hace la supervisión y el control del proceso industrial.

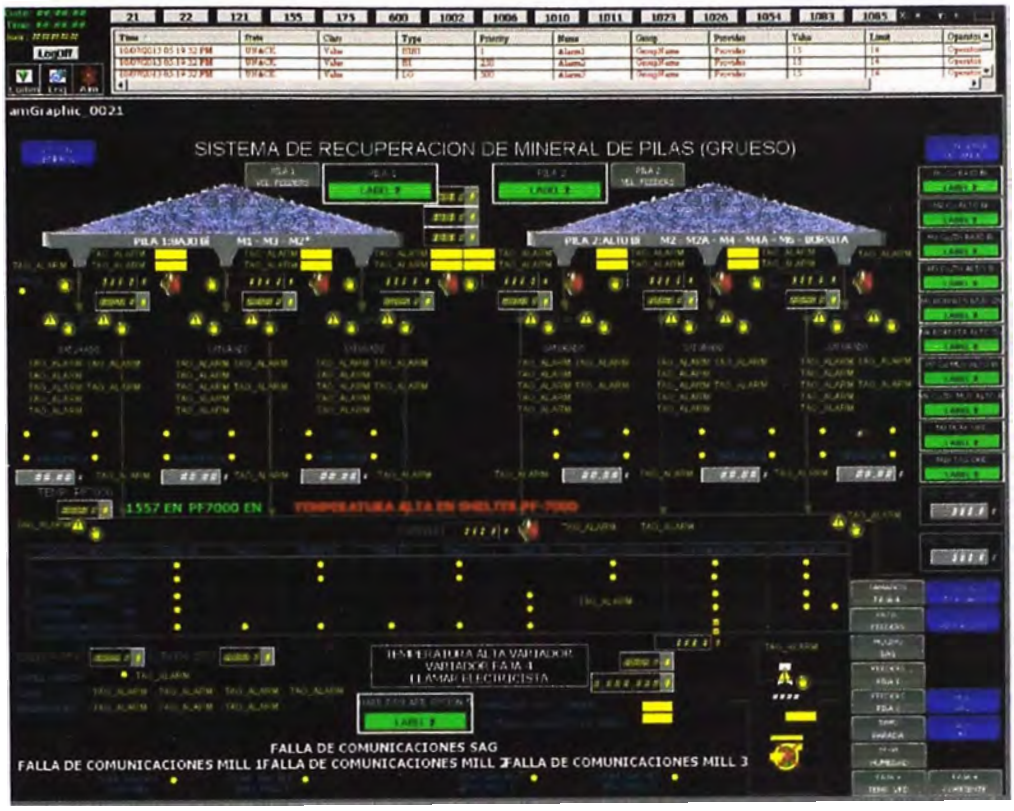

Figura 3.19 Pantalla 0021 - Sistema de recuperación de mineral de pilas (grueso)

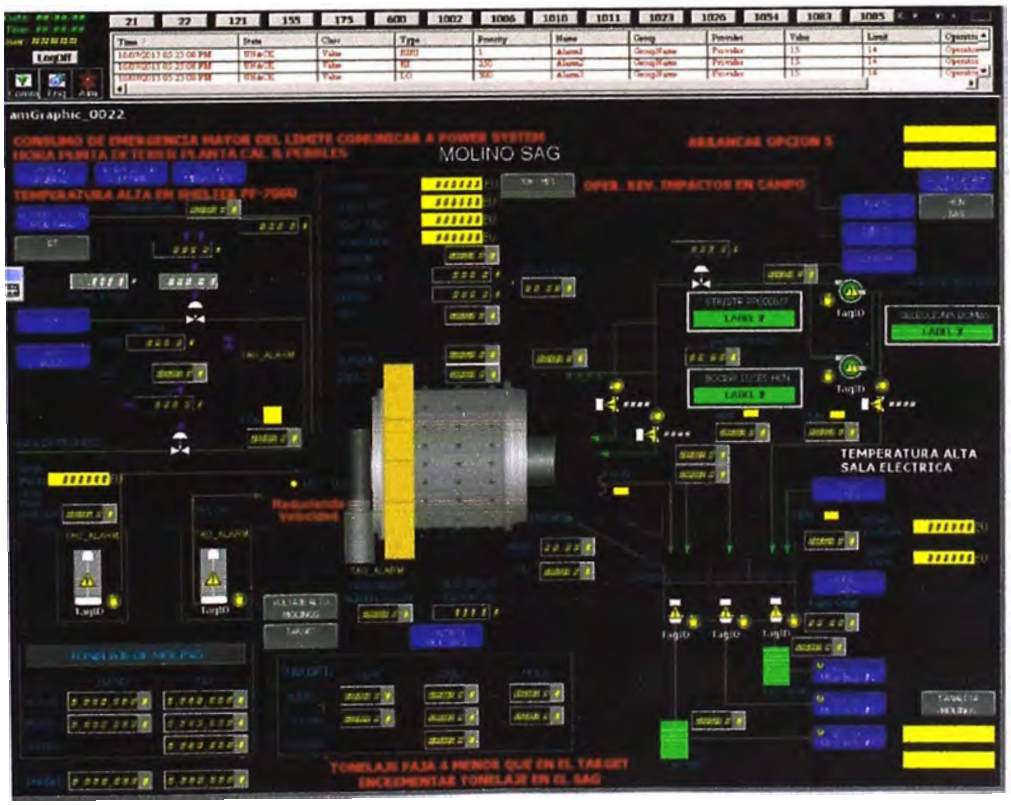

Figura 3.20 Pantalla 0022 - Molino SAG

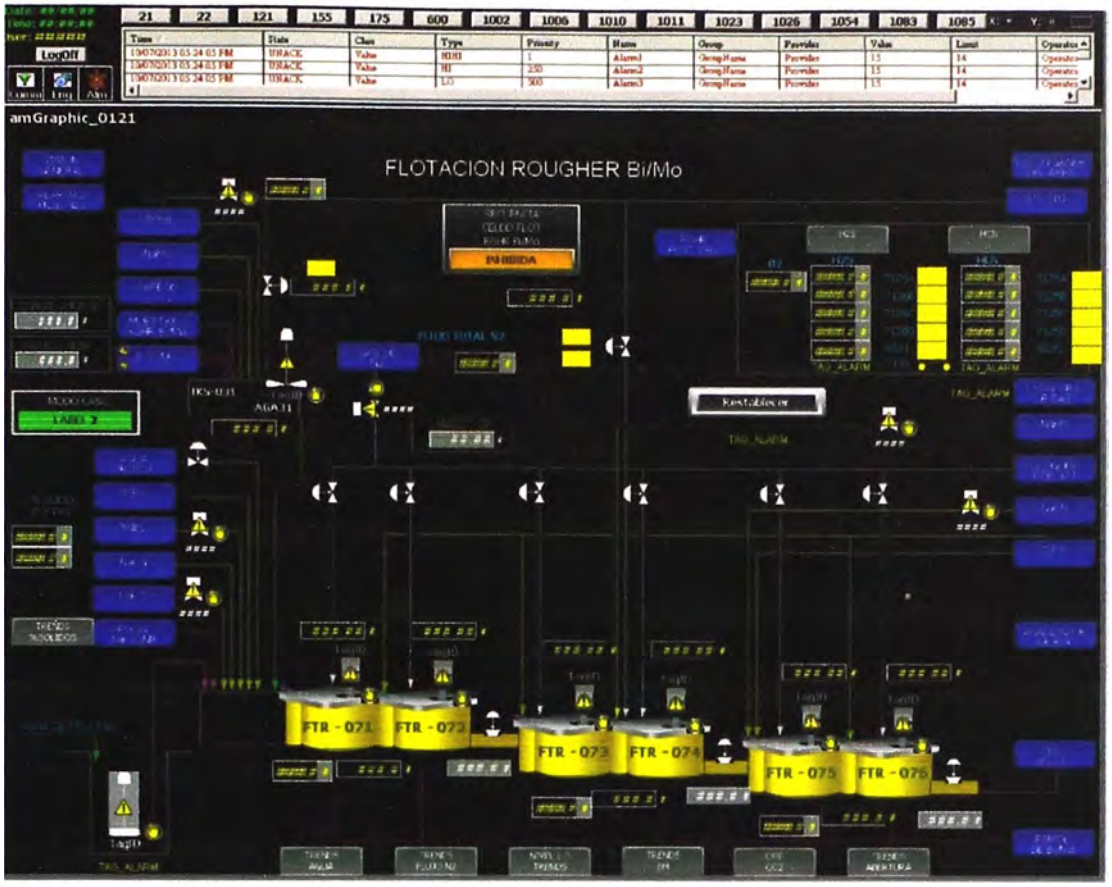

Figura 3.21 Pantalla 0121 - Flotación rougher Bi/Mo

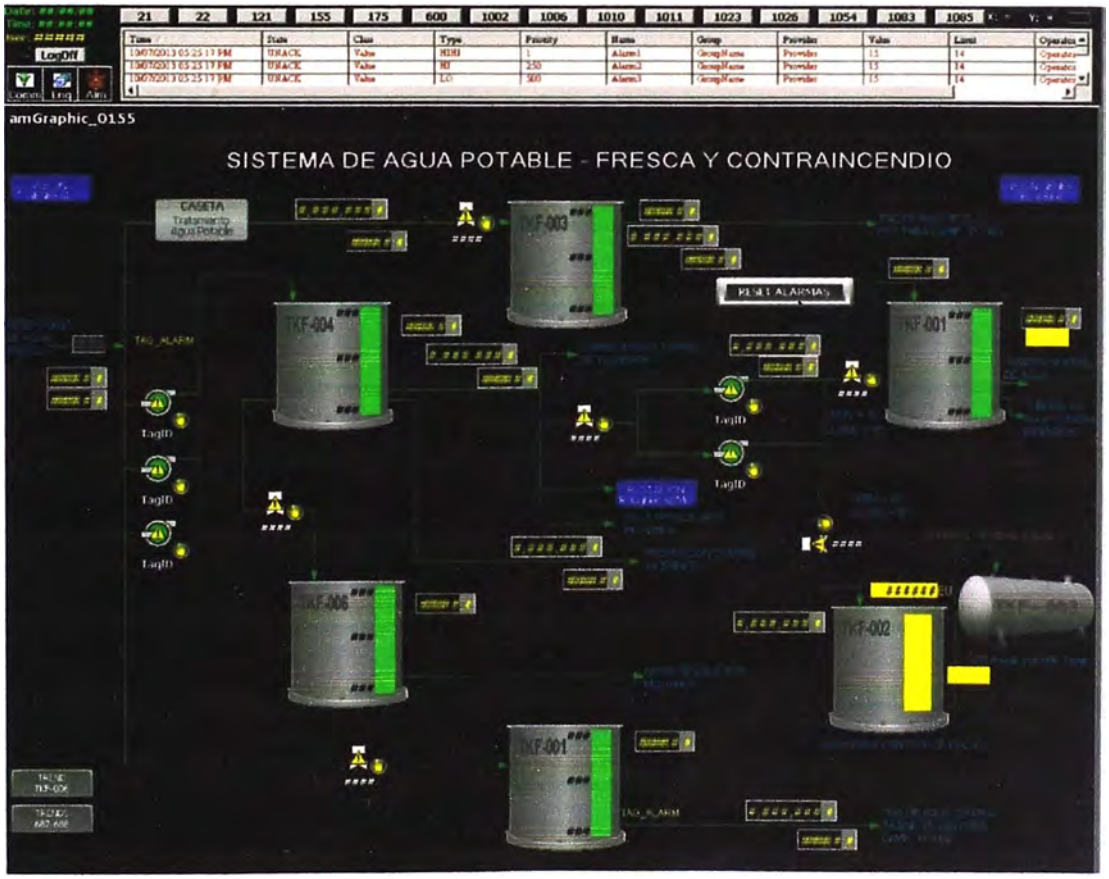

Figura 3.22 Pantalla 0155 - Sistema de agua potable - fresca y contra-incendio

Figura 3.23 Pantalla 0175 - Analizador courier para Cu

Las 5 pantallas mostradas representan a un grupo de interfaces gráficas que los operadores usan para supervisar y controlar la planta. Es a través de estas pantallas que los operadores pueden ver los niveles de los tanques de agua, el estado de las válvulas, el estado de las bombas, entre otros. Además, es a través de esta interfaz que se envían los comandos a los equipos de campo, esto cuando se da la necesidad por ejemplo de abrir una válvula o arrancar-detener un equipo como un motor. En ese sentido, una de las pruebas que el usuario final planteó como claves para la verificación de la funcionalidad del nuevo sistema fue la respuesta en tiempo que se tomaba la plataforma en ejecutar los comandos a los equipos de campo (equipos como bombas de agua). La respuesta ideal, para el usuario final, era que el tiempo que se demorase el nuevo sistema sea el mismo tiempo que le toma al sistema precedente. Dada las condiciones con las que se planteó la arquitectura mostrada anteriormente, esta respuesta ideal no va a ser posible. Esto debido al incremento en el número de interfaces que tiene que usar el System Platform respecto del sistema Bailey. Para ver de una manera más clara las razones del mayor retardo se tiene la siguiente sección, Retardo por Interfaces.

3.5.1 Retardos por interfaces

Para tener una idea más clara de cuál es el camino de la información desde su fuente original (Bloques del Bailey) hasta la visualización en el HMI del System Platform se

muestra el siguiente esquema:

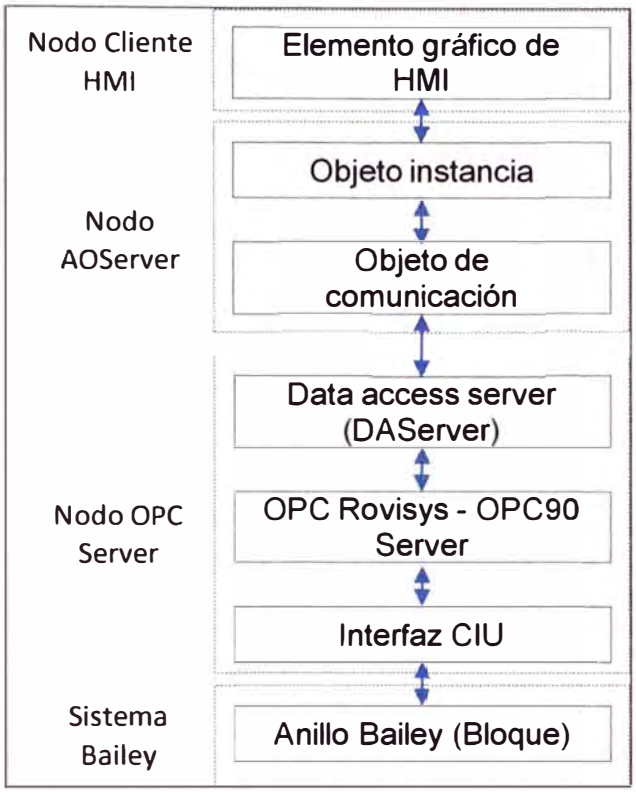

Figura 3.24 Recorrido de la información - desde su fuente origen hasta su visualización

De la figura anterior se nota que la información tiene que pasar por diversas interfaces que se encuentran en diversos nodos, estas interfaces son tanto hardware como software. Preliminarmente se puede pensar en que el paso por estas interfaces inyecta cierto nivel de retardo a la visualización de la información, desde su fuente origen hasta su visualización en el nodo cliente HMI. Esto debido a que cada programa o software y hardware necesitan de cierto tiempo para procesar la información que reciben. Dado esto, es conveniente cuantificar de alguna manera el retardo que inyecta una interfaz (software) al recorrido de la información.

Para el caso del sistema Bailey el camino aproximado que recorrería la información desde su fuente original hasta su software de visualización es como se muestra en la siguiente figura:

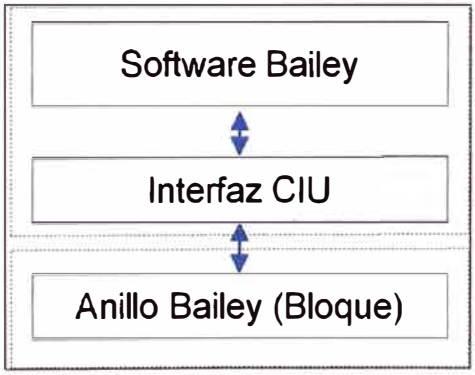

Figura 3.25 Recorrido de la información en el sistema Bailey

Para cuantificar el retardo que significa tener una interfaz más en el recorrido de la información, solo de manera referencial, se va usar un dispositivo indicador digital programable de multifunciones que será la fuente de datos (servidor), una laptop en la cual se ejecutará dos programas: un software A que halará la información del dispositivo y un software B que a su vez halará la información del software A. Para este caso el software A es un software cliente-servidor, y el software B es un software cliente del software A. En este caso de estudio solo se considerará halar un único dato o valor del dispositivo. Se usa la siguiente arquitectura:

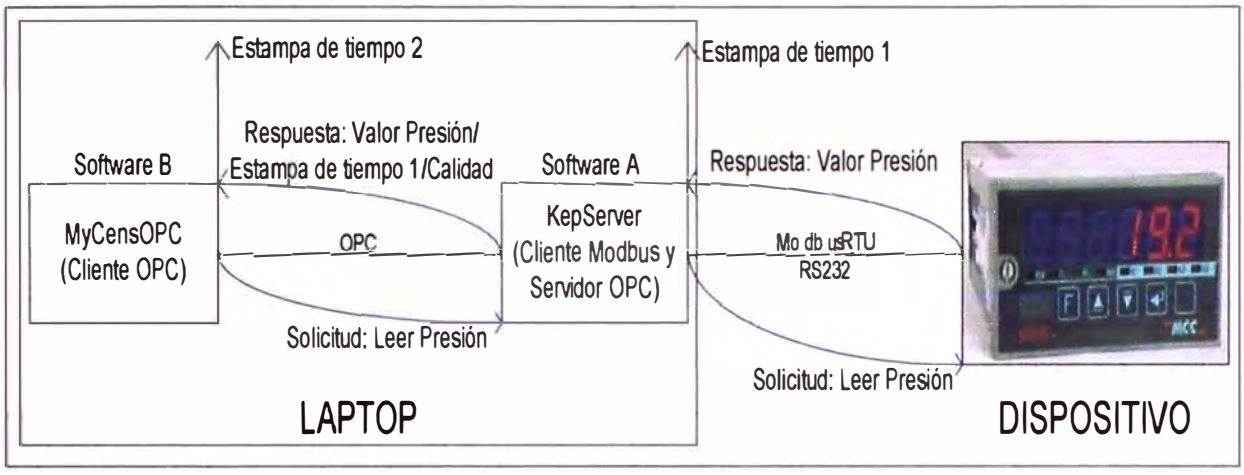

Figura 3.26 Arquitectura a usar para medición de retardo por interfaces

Para obtener el retardo que la interfaz de software A le inyecta a la propagación de la información, se debe obtener la diferencia "Estampa de tiempo 1 - Estampa de tiempo 2", para ello la secuencia a seguir es la siguiente:

a) El software B envía una solicitud de lectura de presión al software A.

b) El software A recibe la solicitud enviada por el software B y se la reenvía al dispositivo.

c) El dispositivo recibe la solicitud de lectura, y responde al software A con el valor de presión que este tiene en uno de sus registros.

d) El software A recibe el valor de presión del dispositivo, le establece su estampa de tiempo (Estampa de tiempo 1) y su calidad de dato. Hecho esto el software A responde a la petición inicial del software B.

e) El software B imprime una estampa de tiempo (Estampa de tiempo 2) al momento de recibir el dato del Software A. El valor de la presión, la calidad del dato y las estampas de tiempo 1 y 2 son mostradas en la interfaz de usuario del software B.

El paso d) es posible debido al uso del protocolo OPC, este protocolo permite el manejo de la calidad y estampas de tiempos de los datos que recibe.

En la siguiente imagen se muestra el entorno de usuario del software B mostrando los valores de presión, calidad de dato y estampas de tiempo 1 y 2.

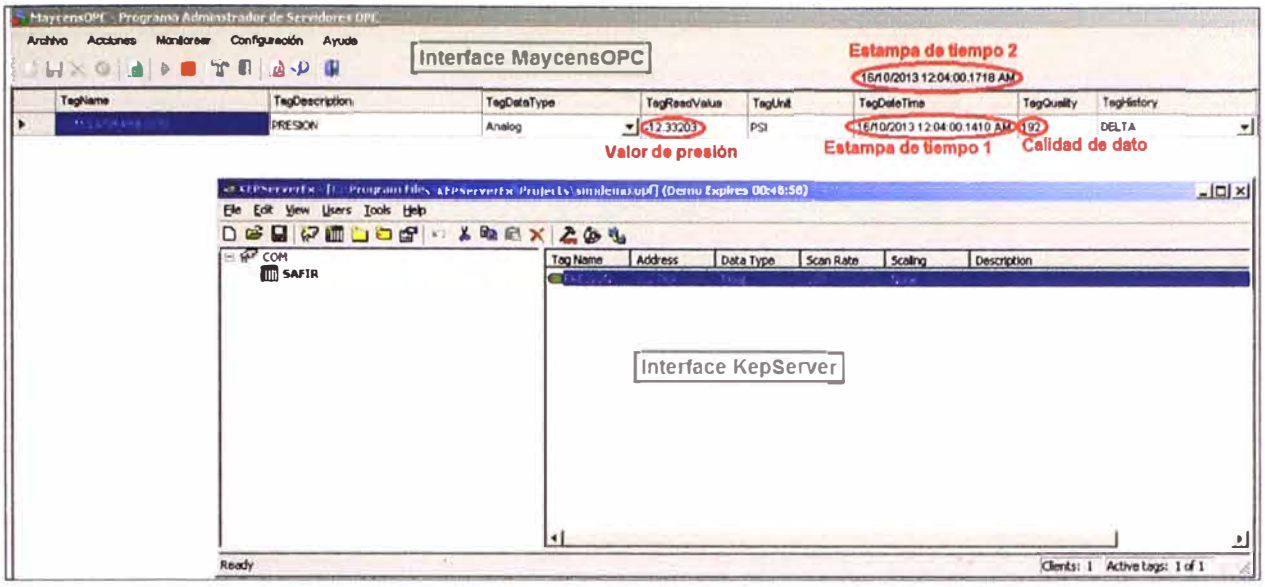

Figura 3.27 Interfaces de software MaycensOPC y KepServer

Los datos obtenidos fueron los siguientes:

- Valor de Presión: -12.33 PSI
- Calidad de dato: 192 (Calidad de dato igual a 192 significa lectura correcta, para valores de calidad de dato diferentes a 192 significan lecturas no correctas)
- Estampa de tiempo 1: 16/10/2013 12:04:00.1410 AM
- Estampa de tiempo 2: 16/10/2013 12:04:00.1718 AM

Entonces se obtiene que el retardo de la interfaz de software A es la diferencia de tiempos registrados en las estampas de tiempos 1 y 2, la cual al realizar la operación da un valor de 0.0308 segundos.

Retardo de Interfaz de Software A: 0.0308 segundos

Este resultado nos da una referencia de cuánto tiempo de retardo es inyectado por una interfaz de software a la difusión de la información en un sistema de supervisión y control donde regularmente se generan solicitudes y respuestas de datos.

CAPITULO IV
ANALISIS Y PRESENTACIÓN DE RESULTADOS

La solución planteada se implementó haciendo uso de las buenas prácticas tanto en ejecución como en programación y manejo de la plataforma de software, siguiendo siempre las recomendaciones del fabricante. Esta implementación contó con la supervisión regular de una persona de soporte técnico representante de la marca de la plataforma usada en la solución.

4.1 Resultados de funcionamiento de la nueva plataforma

A continuación se presenta los resultados de funcionamiento de la aplicación en la nueva plataforma de software, el "*Wonderware System Platform*". Estas pruebas se basan en tres aspectos:

Tiempo de visualización de datos en las pantallas del HMI.

Tiempo de respuesta de comandos enviados a equipos de campo.

Tiempo de respuesta de la redundancia.

4.1.1 Tiempo de visualización de datos en las pantallas del HMI

El siguiente cuadro indica los resultados obtenidos para las pantallas implementadas. Los tiempos mostrados fueron obtenidos mediante un cronómetro manual, desde el tiempo que se realizaba la acción de mostrar la pantalla hasta el momento en que se percibía que todos los datos eran mostrados en ella.

TABLA N° 4.1: Resultado del funcionamiento - visualización de datos en HMI

Pantalla	Estado	Comentario
Pantalla 0021	OK	2.53 segundos de apertura con todos los datos mostrándose.
Pantalla 0022	OK	2.50 segundos de apertura con todos los datos mostrándose.
Pantalla 0121	OK	2.13 segundos de apertura con todos los datos mostrándose.
Pantalla 0155	OK	2.75 segundos de apertura con todos los datos mostrándose.
Pantalla 0175	OK	3.82 segundos de apertura con todos los datos mostrándose.

En la tabla anterior se puede observar que hay diferencias entre los tiempos de visualización de las pantallas. Esto es debido a la diferencia de carga gráfica y de información que tiene cada pantalla, mientras más datos muestra la pantalla, mayor es el tiempo que se demora en mostrarlos a todos en tiempo de ejecución (visualización de datos en tiempo real). Dicho esto, de la tabla anterior se puede ver que la Pantalla 0175 tiene el mayor tiempo en mostrar todos los datos, ello debido a la carga que tiene esta pantalla, la cual podemos verificarlo en la última parte del capítulo "*3.5 Implementación de la solución*". También, se debe aclarar que los tiempos mostrados en la tabla anterior corresponden a la primera vez que se muestran las pantallas listadas, al mostrarlas por segunda vez y en adelante, el tiempo en que muestran todos sus datos se acorta considerablemente. Esto es posible debido a que el sistema maneja memoria cache para los datos ya mostrados, lo cual agiliza la visualización de datos.

4.1.2 Tiempo de respuesta de comandos enviados a equipos de campo

Para esta sección se toma como muestra los comandos de encendido y apagado de tres bombas del área de sistema de agua potable, fresca y contra-incendio. Para ello se ejecuta o envía el comando a la bomba usando el HMI y desde ese momento se toma el tiempo manualmente con un cronometro hasta recibir también en el HMI la señal de confirmación de la acción (feedback).

TABLA N° 4.2: Resultado del funcionamiento - Tiempo de respuesta de comandos

Bomba	Tiempo de respuesta de arranque (Seg.)	Tiempo de respuesta de parada (Seg.)
830HISPPC025	**4.44**	**3.94**
830HISPPC026	**4.31**	**4.44**
830HISPPC027	**4.43**	**4.46**

A manera de comparación también se muestra los resultados de la misma prueba efectuada desde la estación de operación del sistema Bailey usando su propio HMI.

TABLA N° 4.3: Resultado de funcionamiento - Tiempo de respuesta de comandos desde el sistema Bailey

Bomba	Tiempo de respuesta de arranque (Seg.)	Tiempo de respuesta de parada (Seg.)
830HISPPC025	**3.93**	**3.42**
830HISPPC026	**3.45**	**3.09**
830HISPPC027	**4.33**	**4.36**

De las dos tablas anteriores resulta conveniente hacer un cuadro con la diferencia de los tiempos encontrados entre el sistema Bailey y el "*Wonderware System Platform*".

TABLA N° 4.4: Diferencia de tiempos de respuesta entre el Bailey y el System Platform

Bomba	Δ Tiempos de respuesta de arranque (Seg.)	Δ Tiempos de respuesta de parada (Seg.)
830HISPPC025	**0.51**	**0.52**
830HISPPC026	**0.86**	**1.35**
830HISPPC027	**0.1**	**0.1**

En la mayoría de los casos se puede notar que la diferencia encontrada es menor a 1 segundo.

De los resultados anteriores se hace la anotación que el tiempo de respuesta obtenido no solo depende del sistema de supervisión y control, sino que también depende del tiempo de respuesta que tiene el equipo en campo (en este caso una bomba), una vez que recibe la señal de arranque (o parada) el equipo empieza su secuencia de encendido (o parada), la cual depende tanto de las características del equipo como de las características de la parte del proceso sobre la cual actúa, finalizada satisfactoriamente esta secuencia, el equipo envía la confirmación de arrancado (o detenido).

4.1.3 Tiempo de respuesta de la redundancia

Mediante las pruebas realizadas se pudo comprobar la funcionalidad de la redundancia entre pares de AppEngines. Se encontró que el tiempo de conmutación por falla entre un par de AppEngines configurados en redundancia estaba entre 35 a 50 segundos. Estas pruebas se hicieron de tres maneras:

- Forzando la conmutación manualmente mediante un comando desde el HMI (Forced failover).
- Simulando un falla de la red de comunicaciones (desconectando el cable de red del nodo AOServer).
- Simulando una falla de hardware (se quitaba la energía bruscamente al nodo AOServer).

El resultado obtenido mostró que la redundancia de AppEngines se da en 100%, pero que para llegar a esos 100% se necesita de 35 a 50 segundos, tiempo que el sistema necesita para que los AutomationObjects pasen del AppEngine Primario al AppEngine de respaldo. Durante este traspaso de objetos se produce inevitablemente pérdida de información. Esto es, al inicio de la conmutación la información mostrada en el HMI es la última data válida que el objeto ha leído (la información mostrada mientras se da el

proceso de conmutación no está siendo actualizada), luego hay un periodo de pérdida de datos y finalmente se recupera la lectura de datos de campo. Se debe mencionar que el nivel de la redundancia implementado no cumplió con las expectativas del usuario final.

De las tres pruebas realizadas, la que tuvo mayor relevancia para el usuario final fue la prueba "*4.1.2 Tiempo de respuesta de comandos enviados a equipos de campo*". Los resultados mostrados en esa sección muestran que en la mayoría de los casos hay similitud en los tiempos de respuesta para ambos sistemas pero que en todos los casos se encuentra cierta demora del nuevo sistema ("*Wonderware System Platform*") con respecto al sistema original (Bailey). Ello se debe en gran parte al retardo que le inyectan las diversas interfaces de software y hardware por las que tiene que viajar la información desde su fuente original hasta su visualización en el HMI. Si bien es cierto el valor de retardo encontrado para la aplicación de referencia es de 0.0308 segundos para el caso aislado de hacer la solicitud de tan solo un dato o valor del dispositivo, este valor puede incrementarse al aumentar el número de solicitudes al dispositivo ya que habrá una mayor cantidad de información a ser procesada por las interfaces de software, además se debe resaltar el hecho que para el nuevo sistema hay un mayor número de interfaces de software - hardware por los cuales la información debe pasar, por ende, es evidente que los tiempos de respuestas de comandos enviados a campo serán mayores en el nuevo sistema. Es importante señalar que para este punto se conto con la conformidad del usuario final, la compañía minera.

4.2 Tiempo de ejecución del proyecto

Para el cronograma de ejecución del proyecto se cuenta con una herramienta llamada Diagrama de Gantt, el cual se muestra más adelante. Este diagrama cuenta con varias etapas, y si bien es cierto, cuenta con una etapa denominada Implementación, esta se refiere al trabajo de implementación netamente en sitio, es decir, en las instalaciones de la empresa minera que solicitó el servicio, específicamente en la misma planta minera. Cabe aclarar que hubo un trabajo de implementación previo, que empezó días antes del inicio del proyecto, el cual se muestra en el Diagrama de Gantt. Ahora bien, en la ejecución del proyecto se presentaron tiempos retrasos, como por ejemplo malestar del personal debido a la altitud (aclimatación a 4200 m.s.n.m. aproximadamente), inclusive hubo que evacuar a un colega por afectarle dramáticamente la altura, demora en la implementación debido a nuevos requerimientos que estaban fuera del alcance del proyecto, programación forzosa de bajadas de mina (régimen de 14 días de trabajo en mina por 7 de descanso en Lima).

Debido a los retrasos presentados el cronograma fue extendido al final aproximadamente 22 días, terminando así el 03 de agosto del 2012.

ID	Nombre de tarea	Duration	Start	Finish	Predecessors	Resource Names
1	**1 PROYECTO ANTAMINA**	**51 days?**	**5/8/12**	**7/17/12**		
2	1.1 Orden de Compra	0 days	5/8/12	5/8/12		
3	1.2 Acreditacion de Personal	14 days?	5/10/12	5/29/12	2	
4	**1.3 Definiciones Previas al KOM**	**1 day**	**5/30/12**	**5/30/12**	**3**	
5	1.3.1 Arquitectura Preliminar	1 day	5/30/12	5/30/12		
6	1.4 Reunión de Arranque (KOM)	1 day	5/30/12	5/30/12	3	
7	**1.5 DOCUMENTACION**	**4 days?**	**6/5/12**	**6/8/12**		
8	**1.5.1 Elaboracion de Especificaciones**	**4 days?**	**6/5/12**	**6/8/12**	**6,14**	
9	1.5.1.1 Estandar Diseño HMI	2 days	6/5/12	6/6/12		
10	1.5.1.2 Estandar de Base de Datos de Objetos	1 day?	6/7/12	6/7/12	9	
11	1.5.1.3 Arquitectura de Comunicaciones y Redundancia	1 day?	6/8/12	6/8/12	10	
12	1.5.1.4 Entrega de Project Application Standard	0 days	6/8/12	6/8/12	11	
13	**1.6 IMPLEMENTACIÓN**	**18 days**	**5/31/12**	**6/25/12**		
14	1.6.1 Implementacion de Plataforma en Sitio	3 days	5/31/12	6/4/12	6SS	
15	1.6.2 Database Implementation	4 days	6/11/12	6/14/12	11,14	
16	1.6.3 HMI Implementation	4 days	6/15/12	6/20/12	15	
17	1.6.4 Afinamiento de Plataforma	3 days	6/21/12	6/25/12	16	
18	1.6.5 Entrega de Protocolos de Prueba	0 days	6/25/12	6/25/12	17	
19	**1.7 PRUEBAS**	**11 days?**	**6/26/12**	**7/10/12**		
20	1.7.1 SAT	1 day?	6/26/12	6/26/12	18	
21	1.7.2 Comisionamiento	2 days	6/27/12	6/28/12	20	
22	1.7.3 Site Performance Test	3 days	6/29/12	7/3/12	21	
23	1.7.4 Documentacion As Built	5 days	7/4/12	7/10/12	22	
24	1.7.5 Reunion de Cierre de Servicio	0 days	7/10/12	7/10/12	23	
25	**1.8 DESMOVILIZACION**	**5 days**	**7/11/12**	**7/17/12**		
26	1.8.1 Desistalar Plataforma en Sitio	2 days	7/11/12	7/12/12	24	
27	1.8.2 Embalaje y Despacho de Equipameinto	3 days	7/13/12	7/17/12	26	

Figura 4.1 Diagrama de Gantt - Primera parte

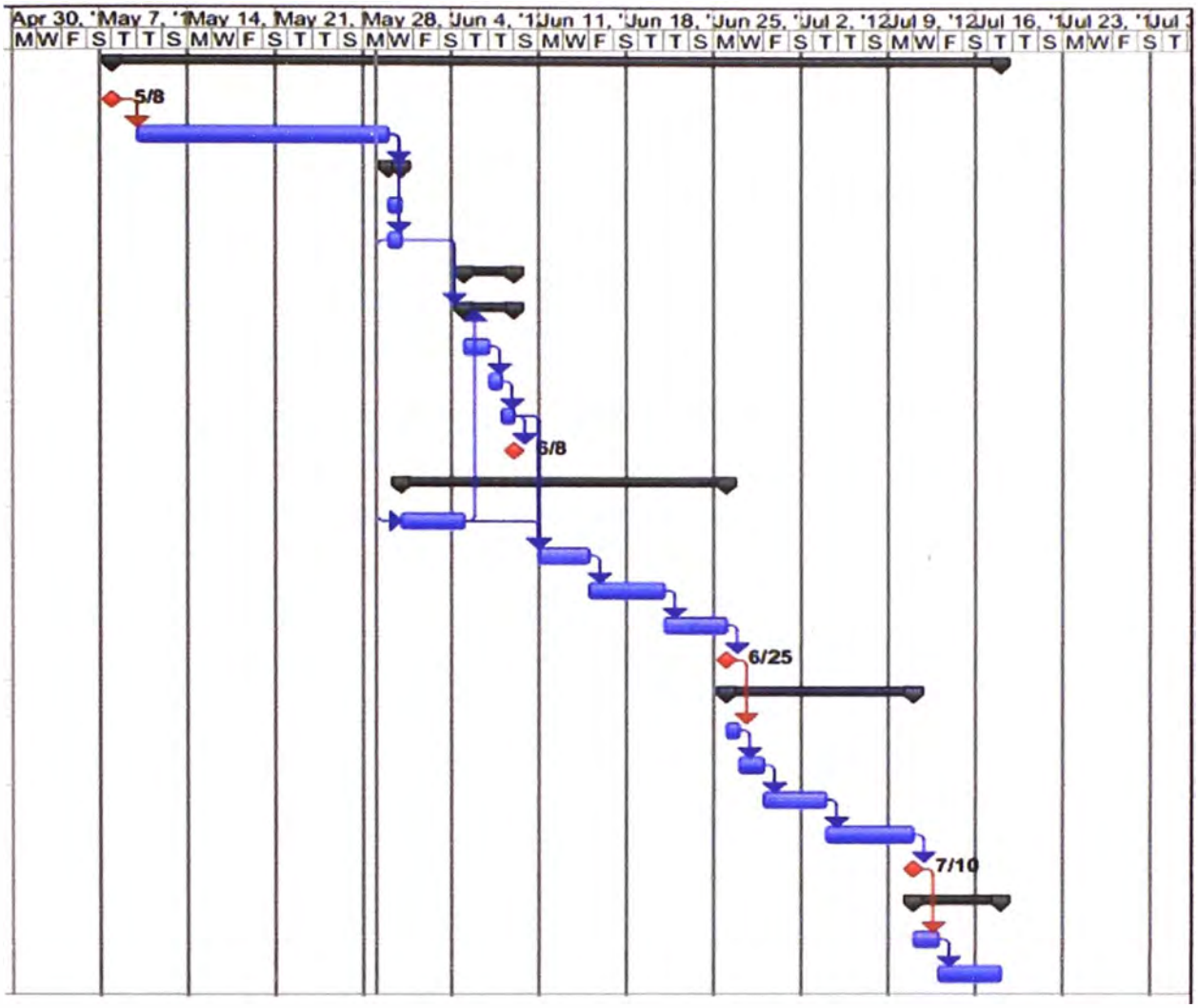

Figura 4.2 Diagrama de Gantt - Segunda parte

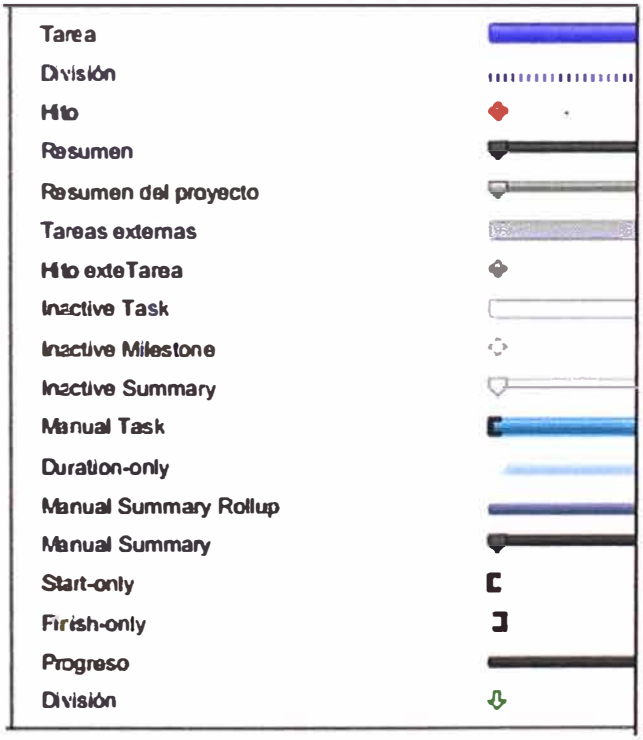

Figura 4.3 Diagrama de Gantt - Leyenda de segunda parte

4.3 Presupuesto

Dado que el presente informe se refiero a un proyecto piloto, desarrollado por la empresa Invensys Process Systems del Peru S.A, no fue necesario comprar hardware nuevo (servidores, estaciones de trabajo y periféricos) sino que se concedió a modo de alquiler el hardware que Invensys ya poseía, además la empresa minera cliente también proveyó cierta parte del hardware para la implementación del proyecto materia de este informe. El proyecto desarrollado era prácticamente un laboratorio, una herramienta que le iba permitir a la empresa minera verificar y validar las bondades y funcionamiento de la nueva plataforma a instalarse, en base a ello la empresa minera decidiría la adquisición completa de la plataforma y el trabajo de implementación del sistema completo de la planta concentradora de minerales.

Este proyecto con la estimación inicial según el diagrama de Gantt mostrado tuvo un costo aproximado de $ 40,000.

A modo de referencia se muestra el costo en licenciamiento que implicaría implementar el sistema presentado en este informe.

TABLA N° 4. 5: Licenciamiento para el Wonderware System Platform

Item	Descripción	Parte	Precio (USD)	Cantidad	Sub Total
1	System Platform 2012R2, 100K IO/25K History - Application Server 100K IO with 10 Application Server Platforms, Historian Server 25K Tag Enterprise Edition, 8 Device Integration Servers, Information Server with 1 IS Advanced CAL (local only)	SP-6560A	$87,695	1	$87,695
2	Development Studio 2012R2 Unlimited, Unlim/60K/500	97-1306	$12,445	2	$24,890
3	InTouch for SysPlatform 2012R2 w/ HistClient	01-2834	$3,510	2	$7,020

De la tabla anterior, el costo total por licenciamiento ascendería a la suma de $ 119,605 dólares americanos.

Para este caso en específico, la implementación del sistema se hizo usando un licenciamiento de tipo Demo las cuales cuentan con todas las funcionalidades habilitadas del sistema. Estas licencias encuentran disponibles para el personal de Invensys Process System del Perú en su sitio web.

Además, a esto habría que sumarle el costo por hora hombre del personal implicado en este proyecto, como referencia el costo para un ingeniero de aplicación es aproximadamente de $ 80 dólares americanos. Para la implementación de este proyecto se tuvo que contar con el servicio de 4 ingenieros de aplicación.

A modo de referencia, se indica que el personal implicado en este proyecto tuvo los siguientes cargos, de manera jerárquicamente descendente:

- 1 Gerente de país
- 1 Gerente de proyecto
- 1 Consultor de aplicación y software
- 1 Líder de proyecto
- 1 Ingeniero de Aplicación Líder
- 3 Ingenieros de Aplicaciones

CONCLUSIONES Y RECOMENDACIONES

1. De lo expuesto en el marco teórico se tiene que, respecto de los sistemas basados en tags, el sistema SCADA "*Wonderware System Platform"* permite ahorrar tiempo y dinero tanto en la implementación inicial como en el mantenimiento del mismo. Esto debido a que el sistema es orientado a objetos.

2. La arquitectura de red implementada soporta redundancia de servidores en tres niveles: falla de hardware, falla de sistema operativo y falla de aplicación. Esto es posible debido a que el software que usa el "*Wonderware System Platform*" está desarrollado de tal manera que la redundancia que se implementa es a nivel de aplicación, por ello la funcionalidad de redundancia actúa ante una interrupción de los servicios que este sistema usa.

3. De los resultados obtenidos en la sección "3.5.1 Retardos por interfaces", se ha comprobado y medido el retardo que significa usar una interfaz de software en la lectura de un dato, además el tiempo medido se incrementa con el número de datos. En tal sentido, mientras más interfaces tenga que atravesar la información desde el origen hasta la aplicación final, mayor es el tiempo que hay que esperar para la visualización de la misma.

4. El sistema implementado en la nueva plataforma de software permite la integración de los demás subsistemas de la planta, además cuenta con la facilidad de ser escalable, por lo que futuras expansiones de la planta no representarían ningún problema al sistema ya instalado. Además, pocos meses después de finalizada la implementación del sistema descrito en este informe, fue lanzada una nueva versión de la plataforma de software usada, la cual incluye nuevas funcionalidades y mejoras a las ya existentes.

5. Para la implementación del sistema mostrado en el presente informe, se recomienda juntar los nodos del OPC Server y del AO Server, con esto la información es transferida internamente en una PC, no tiene que viajar entre aplicaciones de nodos distintos, pasando por dos tarjetas de red y por el cable de transmisión. Con esto se puede acortar en cierta medida el tiempo de respuesta de los comandos enviados a equipos de campo.

6. Los resultados obtenidos en las diversas pruebas del nuevo sistema se obtuvieron haciendo uso de un cronómetro manual, cuya medición está sujeta el tiempo de

respuesta de la persona que lo maneja. Es decir, transcurre cierto tiempo entre la percepción del cambio y la acción de registrarlo manualmente al detener el cronómetro. Para una lectura más exacta se recomienda usar otra forma de medición, la cual puede ser la implementación de un cronómetro dentro del mismo sistema, el cual capture los tiempos exactos de cambio.

7. Dada la pérdida de datos que se obtiene con la redundancia a nivel de la aplicación, para los eventos de falla de hardware o falla de sistema operativo, se recomienda usar la tecnología de virtualización, ya que en esta tecnología es posible la implementación de redundancia a nivel de tolerancia a fallas ("*Fault-Tolerance*"). En este nivel de redundancia no existe pérdida de datos. Sin embargo, se debe tener en cuenta que la implementación del sistema con redundancia sin pérdida de datos requiere de un mayor gasto en licenciamiento y además adquisición de hardware con características específicas para el sistema a implementar.

8. Se recomienda en la medida de lo posible contar con las últimas versiones del software por las mejoras que estos incluyen.

ANEXO A
GLOSARIO DE TERMINOS

Application Server: Servidor de aplicaciones de Wonderware.

Application Objects: Objetos de aplicación que representan equipos físicos o construcciones lógicas dentro de la Galaxia.

ArhcestrA: Tecnología en la cual se basa el desarrollo de la plataforma de software de Wonderware ("*Wonderware System Platform*").

Automation Objects: Objetos usados en el IDE para la configuración del sistema SCADA.

DCS: Sistema de Control Distribuido.

Fault Tolerant: Característica que permite continuar sin detenerse cuando una falla ocurre.

GR: Galaxy Repository - Repositorio de galaxia, es el nodo donde reside la base de datos del sistema SCADA de Wonderware.

High Availability (HA): Habilidad para continuar rápidamente cuando una falla ocurre.

HMI: Interface Hombre Máquina.

I/O: Señales de entrada/salida. Transferencia de información entre un dispositivo fuente (como un PLC) y el sistema SCADA.

IDE: Integrated Development Environment - Entorno de desarrollo integrado, es el entorno de la plataforma del sistema de Wonderware usado para la configuración de la base de datos del sistema SCADA.

SCADA: Sistema de adquisición de datos, supervisión y control.

SOA: Service Oriented Architecture - Arquitectura orientada al servicio.

ANEXO B

CODIGO DE MYCENSOPC - CLIENTE OPC

```
OPTION STRICT OFF
OPTION EXPLICIT ON
IMPORTS SYSTEM.RUNTIME.INTEROPSERVICES
IMPORTS OPCAUTOMATION
IMPORTS SYSTEM.DATA
IMPORTS SYSTEM.DATA.ODBC
IMPORTS SYSTEM.IO
IMPORTS SYSTEM.WINDOWS.FORMS
IMPORTS MICROSOFT.OFFICE.INTEROP.EXCEL
IMPORTS MICROSOFT.WIN32

PUBLIC CLASS FRMPRINCIPAL
   INHERITS SYSTEM.WINDOWS.FORMS.FORM
   PUBLIC CONEXIONBD AS SYSTEM.DATA.ODBC.ODBCCONNECTION
   DIM PROCESO = 0 '1:NUEVO, 0:EDITAR
   DIM WITHEVENTS OPCMYSERVER AS OPCAUTOMATION.OPCSERVER
   DIM WITHEVENTS OPCMYGROUPS AS OPCAUTOMATION.OPCGROUPS
   DIM WITHEVENTS OPCMYGROUP AS OPCAUTOMATION.OPCGROUP
   DIM OPCMYITEMS AS OPCAUTOMATION.OPCITEMS
   DIM OPCMYITEM AS OPCAUTOMATION.OPCITEM
   DIM SERVERHANDLES AS ARRAY
   DIM OVAL(1) AS OBJECT
   DIM SH(1) AS INTEGER

   PRIVATE MINI AS NEW CINIARRAY

   DIM CONTADOR(512) AS INTEGER 'DATOS PARA EL HISTORIAL
   DIM READVALUE(512) AS STRING
   DIM READVALUEANTERIOR(512) AS STRING

   PRIVATE SUB FORM1_LOAD(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E AS
SYSTEM.EVENTARGS) HANDLES MYBASE.LOAD
      CONECTARMYSQL()
      LLENARDGV3()
      TABCONTROL1.WIDTH = ME.WIDTH - DGV1.WIDTH
      DGV1.COLUMNS(6).DEFAULTCELLSTYLE.FORMAT = "DD/MM/YYYY
```

```
HH:MM:SS.FFFF TT"

      'INICIO AUTOMATICO
      DIM KEY AS REGISTRYKEY =
REGISTRY.CURRENTUSER.OPENSUBKEY("SOFTWARE\MICROSOFT\WINDOWS\CU
RRENTVERSION\RUN", TRUE)
      KEY.SETVALUE("MYAPP",
SYSTEM.WINDOWS.FORMS.APPLICATION.EXECUTABLEPATH)
   END SUB

   PUBLIC SUB CONECTARMYSQL()
      "" MOSTRAR LAS CLAVES DE ESTA SECCIÓN
      DIM SFICINI AS STRING =
SYSTEM.WINDOWS.FORMS.APPLICATION.STARTUPPATH & "\FILEINI.INI"
      DIM SERVIDOR, USUARIO, CONTRASEÑA, BASEDATOS AS STRING
      DIM MODOAUTO AS STRING

      SERVIDOR = MINI.INIGET(SFICINI, "MYSQL", "SERVIDOR", "")
      USUARIO = MINI.INIGET(SFICINI, "MYSQL", "USUARIO", "")
      CONTRASEÑA = MINI.INIGET(SFICINI, "MYSQL", "CONTRASEÑA", "")
      BASEDATOS = MINI.INIGET(SFICINI, "MYSQL", "BASEDATOS", "")
      MODOAUTO = MINI.INIGET(SFICINI, "OPC", "AUTO", "")

      TRY
         CONEXIONBD = NEW
SYSTEM.DATA.ODBC.ODBCCONNECTION("DRIVER={MYSQL ODBC 5.1
DRIVER};SERVER=" & SERVIDOR & ";USER=" & USUARIO & ";PASSWORD=" &
CONTRASEÑA & ";DATABASE=" & BASEDATOS)
         CONEXIONBD.OPEN()
         CREARRUNTIME() 'CREAMOS LA TABLA RUNTIME SI NO EXISTE
         LLENARDGV1()
         DGV2.ROWS.ADD(NOW, "CONECTADO CORRECTAMENTE A MYSQL
SERVER")

         'INICIO AUTOMATICO
         IF MODOAUTO = "TRUE" THEN
```

```
            CONECTAOPC()
          END IF
        CATCH EX AS ODBCEXCEPTION
          MSGBOX(EX.MESSAGE, MSGBOXSTYLE.CRITICAL, "MAYCENSOPC")
        END TRY
    END SUB

    PUBLIC SUB CREARRUNTIME()
        TRY
          DIM SQL AS STRING = "CREATE TABLE IF NOT EXISTS RUNTIME(TAGNAME
VARCHAR(100),TAGDESCRIPTION VARCHAR(200),TAGDATATYPE VARCHAR(10),
TAGREADVALUE VARCHAR(20),TAGWRITEVALUE VARCHAR(20),TAGUNIT
VARCHAR(20),TAGDATETIME DATETIME,TAGFUNCTION INT,TAGQUALITY
INT,TAGHISTORY VARCHAR(15),TAGID INT AUTO_INCREMENT,PRIMARY
KEY(TAGID))"
          DIM COMANDOSQL AS ODBCCOMMAND = NEW ODBCCOMMAND(SQL,
CONEXIONBD)
          DIM RESULTADOSQL AS ODBCDATAREADER =
COMANDOSQL.EXECUTEREADER()
        CATCH EX AS ODBCEXCEPTION
          MSGBOX(EX.MESSAGE, MSGBOXSTYLE.CRITICAL, "MAYCENSOPC")
        END TRY
    END SUB

    PUBLIC SUB LLENARDGV1()
        TRY
          DIM SQL AS STRING = "SELECT * FROM RUNTIME"
          DIM COMANDOSQL AS ODBCCOMMAND = NEW ODBCCOMMAND(SQL,
CONEXIONBD)
          DIM RESULTADOSQL AS ODBCDATAREADER =
COMANDOSQL.EXECUTEREADER()
          DIM I AS INTEGER

          DGV1.ROWS.CLEAR()

          WHILE RESULTADOSQL.READ
```

```
            DGV1.ROWS.ADD()
            FOR I = 0 TO RESULTADOSQL.FIELDCOUNT - 1
              DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(I).VALUE =
RESULTADOSQL(I).TOSTRING()
            NEXT I
          END WHILE
        CATCH EX AS ODBCEXCEPTION
          DGV1.ROWS.CLEAR()
          MSGBOX(EX.MESSAGE, MSGBOXSTYLE.CRITICAL, "MAYCENSOPC")
        END TRY
    END SUB

    PUBLIC SUB LLENARDGV3()
        ' MOSTRAR LAS CLAVES DE ESTA SECCIÓN
        DIM SFICINI AS STRING =
SYSTEM.WINDOWS.FORMS.APPLICATION.STARTUPPATH & "\FILEINI.INI"

        DGV3.ROWS.CLEAR()

        DGV3.ROWS.ADD("NODO:", MINI.INIGET(SFICINI, "OPC", "NODO", ""))
        DGV3.ROWS.ADD("OPC SERVER:", MINI.INIGET(SFICINI, "OPC",
"OPCSERVER", ""))
        DGV3.ROWS.ADD("TIEMPO DE ESCANEO(SEG):", MINI.INIGET(SFICINI, "OPC",
"ESCANEO", ""))
        DGV3.ROWS.ADD("ACTUALIZACIÓN DE LA BD(SEG):", MINI.INIGET(SFICINI,
"OPC", "ACTUALIZACION", ""))

        DGV3.ROWS.ADD("SERVIDOR:", MINI.INIGET(SFICINI, "MYSQL", "SERVIDOR",
""))
        DGV3.ROWS.ADD("USUARIO:", MINI.INIGET(SFICINI, "MYSQL", "USUARIO", ""))
        DGV3.ROWS.ADD("CONTRASEÑA:", MINI.INIGET(SFICINI, "MYSQL",
"CONTRASEÑA", ""))
        DGV3.ROWS.ADD("BASE DE DATOS:", MINI.INIGET(SFICINI, "MYSQL",
"BASEDATOS", ""))
        DGV3.ROWS.ADD("NOMBRE DE LA TABLA:", "RUNTIME")
    END SUB
```

```
    PRIVATE SUB OPCMYGROUP_DATACHANGE(BYVAL TRANSACTIONID AS
INTEGER, BYVAL NUMITEMS AS INTEGER, BYREF CLIENTHANDLES AS
SYSTEM.ARRAY, BYREF ITEMVALUES AS SYSTEM.ARRAY, BYREF QUALITIES AS
SYSTEM.ARRAY, BYREF TIMESTAMPS AS SYSTEM.ARRAY) HANDLES
OPCMYGROUP.DATACHANGE
        DIM I%

        'CAPTURAR ESTAMPA DE TIEMPO AL RECIBIR EL DATO"""""""""""""""
        LBLSTARTTIME.TEXT = DATETIME.NOW.TOSTRING("DD/MM/YYYY
HH:MM:SS.FFFF TT")
        """""""""""""""""""""""
        FOR I = 1 TO NUMITEMS
            DGV1.ROWS(CLIENTHANDLES(I) - 1).CELLS(3).VALUE = ITEMVALUES(I)
            DGV1.ROWS(CLIENTHANDLES(I) - 1).CELLS(6).VALUE = TIMESTAMPS(I)
            DGV1.ROWS(CLIENTHANDLES(I) - 1).CELLS(8).VALUE = QUALITIES(I)
        NEXT I

    END SUB

    PRIVATE SUB BTNNUEVO_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL
E AS SYSTEM.EVENTARGS) HANDLES BTNNUEVO.CLICK
        DGV1.ROWS.ADD("", "", "ANALOG", "0", "0", "",
DATETIME.NOW.TOSTRING("DD/MM/YYYY HH:MM:SS.FF TT"), "", "", "(NINGUNO)",
"")
        PROCESO = 1 'NUEVO
        BTNNUEVO.ENABLED = FALSE
        MNUNUEVO.ENABLED = FALSE
    END SUB

    PRIVATE SUB BTNGUARDAR_CLICK(BYVAL SENDER AS SYSTEM.OBJECT,
BYVAL E AS SYSTEM.EVENTARGS) HANDLES BTNGUARDAR.CLICK
        TRY
            DGV1.READONLY = TRUE

            DIM ADAPTADOR AS NEW SYSTEM.DATA.ODBC.ODBCDATAADAPTER
```

```
        DIM MIDATASET AS DATASET = NEW DATASET
        DIM MYRECORDS AS INTEGER
        DIM SQL AS STRING = ""

        IF VALIDAR() = FALSE THEN ' UTILIZA LA FUNCIÓN DE VALIDACIÓN
          IF PROCESO = 1 THEN    ""GUARDA NUEVO REGISTRO
            IF DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(0).VALUE <> "" THEN '
REGISTRO EN BLANCO
              SQL = "INSERT INTO RUNTIME VALUES('" &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(0).VALUE & "'," &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(1).VALUE & "'," &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(2).VALUE & "','0','0'," &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(5).VALUE & "'," &
FORMAT(NOW.DATE, "YYYY-MM-DD HH:MM:SS") & "','0','0'," &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(9).VALUE & "',0)"
              ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL, CONEXIONBD)
              MYRECORDS = ADAPTADOR.FILL(MIDATASET)
              BTNNUEVO.ENABLED = TRUE 'BOTON GUARDAR
              MNUNUEVO.ENABLED = TRUE
              PROCESO = 0
              LLENARDGV1()
              CREATABLA() 'CREA LA TABLA PARA HISTORIZAR EL NUEVO TAG
              DGV2.ROWS.ADD(NOW, "EL TAG " &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(0).VALUE & " SE HA AGREGADO A LA
LISTA")
            ELSE
              MSGBOX("REGISTRO EN BLANCO. OPERACIÓN CANCELADA",
MSGBOXSTYLE.CRITICAL, "MAYCENSOPC")
              BTNNUEVO.ENABLED = TRUE
              MNUNUEVO.ENABLED = TRUE
              LLENARDGV1()
              PROCESO = 0
            END IF
          ELSE              ""EDITA TODOS LOS REGISTROS
            FOR I = 0 TO DGV1.ROWS.COUNT - 1
```

```
                SQL = "UPDATE RUNTIME SET TAGNAME='" &
DGV1.ROWS(I).CELLS(0).VALUE & "',TAGDESCRIPTION='" &
DGV1.ROWS(I).CELLS(1).VALUE & "',TAGDATATYPE='" &
DGV1.ROWS(I).CELLS(2).VALUE & "',TAGUNIT='" & DGV1.ROWS(I).CELLS(5).VALUE
& "',TAGHISTORY='" & DGV1.ROWS(I).CELLS(9).VALUE & "' WHERE TAGID='" &
DGV1.ROWS(I).CELLS(10).VALUE & "'"
                ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL, CONEXIONBD)
                MYRECORDS = ADAPTADOR.FILL(MIDATASET)
            NEXT I
            LLENARDGV1()
            DGV2.ROWS.ADD(NOW, "LA LISTA HA SIDO EDITADA
CORRECTAMENTE")
          END IF
        ELSE
          DGV2.ROWS.ADD(NOW, "ERROR AL GUARDAR LA INFORMACIÓN.
OPERACIÓN CANCELADA")
          MSGBOX("ERROR AL GUARDAR LA INFORMACIÓN. OPERACIÓN
CANCELADA", MSGBOXSTYLE.CRITICAL, "MAYCENSOPC")
          BTNNUEVO.ENABLED = TRUE
          MNUNUEVO.ENABLED = TRUE
          LLENARDGV1()
          PROCESO = 0
        END IF

        DGV1.READONLY = FALSE
      CATCH EX AS ODBCEXCEPTION
        MSGBOX(EX.MESSAGE)
      END TRY
    END SUB

    PRIVATE SUB BTNELIMINAR_CLICK(BYVAL SENDER AS SYSTEM.OBJECT,
BYVAL E AS SYSTEM.EVENTARGS) HANDLES BTNELIMINAR.CLICK
      TRY
        IF (MESSAGEBOX.SHOW("DESEA ELIMINAR EL TAG: " &
DGV1.ROWS(DGV1.CURRENTCELL.ROWINDEX).CELLS(0).VALUE,
```

```
"CONFIRMACIÓN", MESSAGEBOXBUTTONS.YESNO,
MESSAGEBOXICON.QUESTION) = DIALOGRESULT.YES) THEN

            DIM ADAPTADOR AS NEW SYSTEM.DATA.ODBC.ODBCDATAADAPTER
            DIM MIDATASET AS DATASET = NEW DATASET
            DIM MYRECORDS AS INTEGER
            DIM SQL AS STRING = ""

            SQL = "DELETE FROM RUNTIME WHERE TAGID='" &
DGV1.ROWS(DGV1.CURRENTROW.INDEX).CELLS(10).VALUE & "'"
            ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL, CONEXIONBD)
            MYRECORDS = ADAPTADOR.FILL(MIDATASET)

            ELIMINARTABLA() 'ELIMINA LA TABLA HISTORIAL CORRESPONDIENTE AL
TAG ELIMINADO

            DGV2.ROWS.ADD(NOW, "EL TAG " &
DGV1.ROWS(DGV1.CURRENTCELL.ROWINDEX).CELLS(0).VALUE & " SE HA
ELIMINADO DE LA LISTA")
            LLENARDGV1()
          END IF
        CATCH EX AS ODBCEXCEPTION
          MSGBOX(EX.MESSAGE)
        END TRY
    END SUB

    PRIVATE SUB BTNCANCELAR_CLICK(BYVAL SENDER AS SYSTEM.OBJECT,
BYVAL E AS SYSTEM.EVENTARGS) HANDLES BTNCANCELAR.CLICK
        BTNNUEVO.ENABLED = TRUE
        MNUNUEVO.ENABLED = TRUE
        PROCESO = 0
        LLENARDGV1()
    END SUB

    PRIVATE SUB BTNRUN_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E
```

```
AS SYSTEM.EVENTARGS) HANDLES BTNRUN.CLICK
        CONECTAOPC()
    END SUB

    PRIVATE SUB BTNSTOP_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E
AS SYSTEM.EVENTARGS) HANDLES BTNSTOP.CLICK
        DESCONECTAOPC()
    END SUB

    PRIVATE SUB BTNOPC_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E
AS SYSTEM.EVENTARGS) HANDLES BTNOPC.CLICK
        FRMOPC.SHOWDIALOG()
    END SUB

    PRIVATE SUB BTNMYSQL_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL
E AS SYSTEM.EVENTARGS) HANDLES BTNMYSQL.CLICK
        FRMMYSQL.SHOWDIALOG()
    END SUB

    PRIVATE SUB BTNAYUDA_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E
AS SYSTEM.EVENTARGS) HANDLES BTNAYUDA.CLICK
        PROCESS.START(SYSTEM.WINDOWS.FORMS.APPLICATION.STARTUPPATH &
"\MAYCENSOPC.PDF")
    END SUB

    PRIVATE SUB BTNSALIR_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E
AS SYSTEM.EVENTARGS) HANDLES BTNSALIR.CLICK
        IF (MESSAGEBOX.SHOW("AL CERRAR EL PROGRAMA SE DETENDRÁ LA
SUPERVISIÓN Y CONTROL DEL SISTEMA", "¿QUIERE SALIR DEL PROGRAMA?",
MESSAGEBOXBUTTONS.OKCANCEL, MESSAGEBOXICON.WARNING) =
DIALOGRESULT.OK) THEN
            IF BTNSTOP.ENABLED = TRUE THEN
                DESCONECTAOPC()
            END IF
            END 'SALIR
        ELSE
```

```
        END IF
    END SUB

    PRIVATE SUB CREATABLA()
        DIM ADAPTADOR AS NEW SYSTEM.DATA.ODBC.ODBCDATAADAPTER
        DIM MIDATASET AS DATASET = NEW DATASET
        DIM MYRECORDS AS INTEGER
        DIM SQL AS STRING = "CREATE TABLE ID" &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(10).VALUE & "(TAGREADVALUE
VARCHAR(20),TAGDATETIME DATETIME)"

        ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL, CONEXIONBD)
        MYRECORDS = ADAPTADOR.FILL(MIDATASET)
    END SUB

    PRIVATE SUB ELIMINARTABLA()
        DIM ADAPTADOR AS NEW SYSTEM.DATA.ODBC.ODBCDATAADAPTER
        DIM MIDATASET AS DATASET = NEW DATASET
        DIM MYRECORDS AS INTEGER
        DIM SQL AS STRING = "DROP TABLE ID" &
DGV1.ROWS(DGV1.CURRENTCELL.ROWINDEX).CELLS(10).VALUE

        ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL, CONEXIONBD)
        MYRECORDS = ADAPTADOR.FILL(MIDATASET)
    END SUB

    PRIVATE SUB MNUNUEVO_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL
E AS SYSTEM.EVENTARGS) HANDLES MNUNUEVO.CLICK
        DGV1.ROWS.ADD("", "", "ANALOG", "0", "0", "",
DATETIME.NOW.TOSTRING("DD/MM/YYYY HH:MM:SS.FF TT"), "", "", "(NINGUNO)",
"")
        PROCESO = 1 'NUEVO
        BTNNUEVO.ENABLED = FALSE
        MNUNUEVO.ENABLED = FALSE
```

```
  END SUB

  PRIVATE SUB MNUGUARDAR_CLICK(BYVAL SENDER AS SYSTEM.OBJECT,
BYVAL E AS SYSTEM.EVENTARGS) HANDLES MNUGUARDAR.CLICK
    TRY
      DGV1.READONLY = TRUE

      DIM ADAPTADOR AS NEW SYSTEM.DATA.ODBC.ODBCDATAADAPTER
      DIM MIDATASET AS DATASET = NEW DATASET
      DIM MYRECORDS AS INTEGER
      DIM SQL AS STRING = ""

      IF VALIDAR() = FALSE THEN ' UTILIZA LA FUNCIÓN DE VALIDACIÓN
        IF PROCESO = 1 THEN     ""GUARDA NUEVO REGISTRO
          IF DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(0).VALUE <> "" THEN '
REGISTRO EN BLANCO
            SQL = "INSERT INTO RUNTIME VALUES('" &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(0).VALUE & "','" &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(1).VALUE & "','" &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(2).VALUE & "','0','0','" &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(5).VALUE & "','" &
FORMAT(NOW.DATE, "YYYY-MM-DD HH:MM:SS") & "','0','0','" &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(9).VALUE & "',0)"
            ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL, CONEXIONBD)
            MYRECORDS = ADAPTADOR.FILL(MIDATASET)
            BTNNUEVO.ENABLED = TRUE 'BOTON GUARDAR
            MNUNUEVO.ENABLED = TRUE
            PROCESO = 0
            LLENARDGV1()
            CREATABLA() 'CREA LA TABLA PARA HISTORIZAR EL NUEVO TAG
            DGV2.ROWS.ADD(NOW, "EL TAG " &
DGV1.ROWS(DGV1.ROWS.COUNT - 1).CELLS(0).VALUE & " SE HA AGREGADO A LA
LISTA")
          ELSE
            MSGBOX("REGISTRO EN BLANCO. OPERACIÓN CANCELADA",
```

```
MSGBOXSTYLE.CRITICAL, "MAYCENSOPC")
                BTNNUEVO.ENABLED = TRUE
                MNUNUEVO.ENABLED = TRUE
                LLENARDGV1()
                PROCESO = 0
              END IF
            ELSE                ""EDITA TODOS LOS REGISTROS
              FOR I = 0 TO DGV1.ROWS.COUNT - 1
                SQL = "UPDATE RUNTIME SET TAGNAME='" &
DGV1.ROWS(I).CELLS(0).VALUE & "',TAGDESCRIPTION='" &
DGV1.ROWS(I).CELLS(1).VALUE & "',TAGDATATYPE='" &
DGV1.ROWS(I).CELLS(2).VALUE & "',TAGUNIT='" & DGV1.ROWS(I).CELLS(5).VALUE
& "',TAGHISTORY='" & DGV1.ROWS(I).CELLS(9).VALUE & "' WHERE TAGID='" &
DGV1.ROWS(I).CELLS(10).VALUE & ""'
                ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL, CONEXIONBD)
                MYRECORDS = ADAPTADOR.FILL(MIDATASET)
              NEXT I
              LLENARDGV1()
              DGV2.ROWS.ADD(NOW, "LA LISTA HA SIDO EDITADA
CORRECTAMENTE")
            END IF
          ELSE
            DGV2.ROWS.ADD(NOW, "ERROR AL GUARDAR LA INFORMACIÓN.
OPERACIÓN CANCELADA")
            MSGBOX("ERROR AL GUARDAR LA INFORMACIÓN. OPERACIÓN
CANCELADA", MSGBOXSTYLE.CRITICAL, "MAYCENSOPC")
            BTNNUEVO.ENABLED = TRUE
            MNUNUEVO.ENABLED = TRUE
            LLENARDGV1()
            PROCESO = 0
          END IF

          DGV1.READONLY = FALSE
        CATCH EX AS ODBCEXCEPTION
          MSGBOX(EX.MESSAGE)
```

```
    END TRY
  END SUB

  PRIVATE SUB MNUELIMINAR_CLICK(BYVAL SENDER AS SYSTEM.OBJECT,
BYVAL E AS SYSTEM.EVENTARGS) HANDLES MNUELIMINAR.CLICK
    IF (MESSAGEBOX.SHOW("DESEA ELIMINAR EL TAG: " &
DGV1.ROWS(DGV1.CURRENTCELL.ROWINDEX).CELLS(0).VALUE,
"CONFIRMACIÓN", MESSAGEBOXBUTTONS.YESNO,
MESSAGEBOXICON.QUESTION) = DIALOGRESULT.YES) THEN
      TRY
        DIM ADAPTADOR AS NEW SYSTEM.DATA.ODBC.ODBCDATAADAPTER
        DIM MIDATASET AS DATASET = NEW DATASET
        DIM MYRECORDS AS INTEGER
        DIM SQL AS STRING = ""

        SQL = "DELETE FROM RUNTIME WHERE TAGID='" &
DGV1.ROWS(DGV1.CURRENTROW.INDEX).CELLS(10).VALUE & "'"
        ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL, CONEXIONBD)
        MYRECORDS = ADAPTADOR.FILL(MIDATASET)

        ELIMINARTABLA() 'ELIMINA LA TABLA HISTORIAL CORRESPONDIENTE AL
TAG ELIMINADO

        DGV2.ROWS.ADD(NOW, "EL TAG " &
DGV1.ROWS(DGV1.CURRENTCELL.ROWINDEX).CELLS(0).VALUE & " SE HA
ELIMINADO DE LA LISTA")
        LLENARDGV1()
      CATCH EX AS ODBCEXCEPTION
        MSGBOX(EX.MESSAGE)
      END TRY
    END IF
  END SUB

  PRIVATE SUB MNUCANCELAR_CLICK(BYVAL SENDER AS SYSTEM.OBJECT,
BYVAL E AS SYSTEM.EVENTARGS) HANDLES MNUCANCELAR.CLICK
```

```
        BTNNUEVO.ENABLED = TRUE
        MNUNUEVO.ENABLED = TRUE
        PROCESO = 0
        LLENARDGV1()
    END SUB

    PRIVATE SUB MNUSALIR_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E
AS SYSTEM.EVENTARGS) HANDLES MNUSALIR.CLICK
        IF (MESSAGEBOX.SHOW("AL CERRAR EL PROGRAMA SE DETENDRÁ LA
SUPERVISIÓN Y CONTROL DEL SISTEMA", "¿QUIERE SALIR DEL PROGRAMA?",
MESSAGEBOXBUTTONS.OKCANCEL, MESSAGEBOXICON.WARNING) =
DIALOGRESULT.OK) THEN
            IF BTNSTOP.ENABLED = TRUE THEN
                DESCONECTAOPC()
            END IF
            END 'SALIR
        ELSE
        END IF
    END SUB

    PRIVATE SUB MNURUN_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E
AS SYSTEM.EVENTARGS) HANDLES MNURUN.CLICK
        CONECTAOPC()
    END SUB

    PRIVATE SUB MNUSTOP_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E
AS SYSTEM.EVENTARGS) HANDLES MNUSTOP.CLICK
        DESCONECTAOPC()
    END SUB

    PRIVATE SUB MNUOPC_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E
AS SYSTEM.EVENTARGS) HANDLES MNUOPC.CLICK
        FRMOPC.SHOWDIALOG()
    END SUB

    PRIVATE SUB MNUMYSQL_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL
```

```
E AS SYSTEM.EVENTARGS) HANDLES MNUMYSQL.CLICK
        FRMMYSQL.SHOWDIALOG()
    END SUB

    PRIVATE SUB TIMER1_TICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E AS
SYSTEM.EVENTARGS) HANDLES TIMER1.TICK
        ' MOSTRAR LAS CLAVES DE ESTA SECCIÓN
        DIM SFICINI AS STRING =
SYSTEM.WINDOWS.FORMS.APPLICATION.STARTUPPATH & "\FILEINI.INI"

        TIMER1.INTERVAL = CONVERT.TODOUBLE(MINI.INIGET(SFICINI, "OPC",
"ACTUALIZACION", "")) * 1000

        TRY
            DGV1.READONLY = TRUE
            ""CARGA EL VALOR DE TAGFUNCTION DE LA BASE DE DATOS AL DGV1
            DIM SQL1 AS STRING = "SELECT TAGWRITEVALUE, TAGFUNCTION FROM
RUNTIME"
            DIM COMANDOSQL AS ODBCCOMMAND = NEW ODBCCOMMAND(SQL1,
CONEXIONBD)
            DIM RESULTADOSQL AS ODBCDATAREADER =
COMANDOSQL.EXECUTEREADER()
            DIM J AS INTEGER = 0
            DIM TAGVALUE(DGV1.ROWS.COUNT) AS OBJECT

            WHILE RESULTADOSQL.READ
                DGV1.ROWS(J).CELLS(4).VALUE = RESULTADOSQL(0).TOSTRING()
""TAGVALUE
                DGV1.ROWS(J).CELLS(7).VALUE = RESULTADOSQL(1).TOSTRING()
""TAGFUNCTION
                J = J + 1
            END WHILE

            ""ACTUALIZA LA BASE DE DATOS CON LOS VALORES DEL DGV1 Y
ESCRIBE EN EL OPC SERVER SI TAGFUNCTION ES 1
            DIM ADAPTADOR AS NEW SYSTEM.DATA.ODBC.ODBCDATAADAPTER
```

```
        DIM MIDATASET AS DATASET = NEW DATASET
        DIM MYRECORDS AS INTEGER
        DIM SQL2 AS STRING = ""

        FOR I = 0 TO DGV1.ROWS.COUNT - 1
          IF DGV1.ROWS(I).CELLS(7).VALUE = 1 AND
DGV1.ROWS(I).CELLS(8).VALUE = 192 THEN ""SI LA TAGFUNCTION ES IGUAL A 1
ESCRIBE EN EL OPC SERVER
            DIM ERRORS AS ARRAY

            SH(1) = SERVERHANDLES(I + 1)
            OVAL(1) = DGV1.ROWS(I).CELLS(4).VALUE
            OPCMYGROUP.SYNCWRITE(1, SH, OVAL, ERRORS)
          END IF

          SQL2 = "UPDATE RUNTIME SET TAGREADVALUE='" &
DGV1.ROWS(I).CELLS(3).VALUE & "',TAGWRITEVALUE='" &
DGV1.ROWS(I).CELLS(4).VALUE & "',TAGDATETIME='" &
FORMAT(CONVERT.TODATETIME(DGV1.ROWS(I).CELLS(6).VALUE), "YYYY-MM-DD
HH:MM:SS.FF") & "',TAGFUNCTION=0" & ",TAGQUALITY='" &
DGV1.ROWS(I).CELLS(8).VALUE & "' WHERE TAGID='" &
DGV1.ROWS(I).CELLS(10).VALUE & "'"
          ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL2, CONEXIONBD)
          MYRECORDS = ADAPTADOR.FILL(MIDATASET)
        NEXT I
      CATCH EX AS ODBCEXCEPTION
        MSGBOX(EX.MESSAGE, MSGBOXSTYLE.CRITICAL, "MAYCENSOPC")
        TIMER1.ENABLED = FALSE
      END TRY
    END SUB

    PRIVATE SUB CONECTAOPC()
      DIM SITEMNAME(DGV1.ROWS.COUNT) AS STRING
      DIM CH(DGV1.ROWS.COUNT) AS INTEGER
      DIM SH1(DGV1.ROWS.COUNT) AS INTEGER
```

```
        DIM DTIME(DGV1.ROWS.COUNT) AS DATE
        DIM WQUALITY(DGV1.ROWS.COUNT) AS SHORT

        DIM I%
        DIM ERRORS AS ARRAY
        DIM RET AS INTEGER

        DIM SFICINI AS STRING =
SYSTEM.WINDOWS.FORMS.APPLICATION.STARTUPPATH & "\FILEINI.INI"

        TRY
            '--- CONECTA OPCSERVER
            OPCMYSERVER = NEW OPCSERVER
            OPCMYSERVER.CONNECT(MINI.INIGET(SFICINI, "OPC", "OPCSERVER", ""),
"")

            OPCMYGROUPS = OPCMYSERVER.OPCGROUPS
            OPCMYGROUP = OPCMYGROUPS.ADD("GROUP1")
            OPCMYGROUP.UPDATERATE = VAL(MINI.INIGET(SFICINI, "OPC",
"ESCANEO", "") * 1000)
            OPCMYGROUP.ISACTIVE = FALSE
            OPCMYGROUP.ISSUBSCRIBED = OPCMYGROUP.ISACTIVE
            OPCMYITEMS = OPCMYGROUP.OPCITEMS

            FOR I = 1 TO DGV1.ROWS.COUNT
                SITEMNAME(I) = DGV1.ROWS(I - 1).CELLS(0).VALUE
                CH(I) = I
            NEXT

            OPCMYITEMS.ADDITEMS(DGV1.ROWS.COUNT, SITEMNAME, CH,
SERVERHANDLES, ERRORS)

            FOR I = 1 TO DGV1.ROWS.COUNT
                IF ERRORS(I) = 0 THEN
                    SH1(I) = SERVERHANDLES(I)
                ELSE
```

```
            DGV1.ROWS(I - 1).CELLS(8).VALUE = 0
            DGV2.ROWS.ADD(NOW, "ERROR AL REGISTRAR EL ITEM: " +
SITEMNAME(I))
          END IF
        NEXT

        'LEE TAGS
        OPCMYGROUP.ISACTIVE = NOT OPCMYGROUP.ISACTIVE
        OPCMYGROUP.ISSUBSCRIBED = OPCMYGROUP.ISACTIVE

        DIM ANITEM AS OPCAUTOMATION.OPCITEM

        FOR EACH ANITEM IN OPCMYGROUP.OPCITEMS
          ANITEM.READ(OPCAUTOMATION.OPCDATASOURCE.OPCDEVICE) ',
VALUE, QUAL, TIME    ' IF SUBSCRIBED, DON'T DO THIS!
          DGV1.ROWS(ANITEM.CLIENTHANDLE - 1).CELLS(3).VALUE =
ANITEM.VALUE
          DGV1.ROWS(ANITEM.CLIENTHANDLE - 1).CELLS(6).VALUE =
ANITEM.TIMESTAMP
          DGV1.ROWS(ANITEM.CLIENTHANDLE - 1).CELLS(8).VALUE =
ANITEM.QUALITY
        NEXT ANITEM
        ANITEM = NOTHING

        BTNNUEVO.ENABLED = FALSE
        MNUNUEVO.ENABLED = FALSE
        BTNGUARDAR.ENABLED = FALSE
        MNUGUARDAR.ENABLED = FALSE
        BTNELIMINAR.ENABLED = FALSE
        MNUELIMINAR.ENABLED = FALSE
        BTNCANCELAR.ENABLED = FALSE
        MNUCANCELAR.ENABLED = FALSE
        BTNRUN.ENABLED = FALSE
        MNURUN.ENABLED = FALSE
        BTNSTOP.ENABLED = TRUE
        MNUSTOP.ENABLED = TRUE
```

```
        TIMER1.ENABLED = TRUE
        TIMER2.ENABLED = TRUE

        DGV2.ROWS.ADD(NOW, "EJECUTANDO OPC SERVER...")
    CATCH EX AS EXCEPTION
        MSGBOX(EX.MESSAGE, MSGBOXSTYLE.CRITICAL, "OPC CLIENTE")
        BTNNUEVO.ENABLED = TRUE
        MNUNUEVO.ENABLED = TRUE
        BTNGUARDAR.ENABLED = TRUE
        MNUGUARDAR.ENABLED = TRUE
        BTNELIMINAR.ENABLED = TRUE
        MNUELIMINAR.ENABLED = TRUE
        BTNCANCELAR.ENABLED = TRUE
        MNUCANCELAR.ENABLED = TRUE
        BTNRUN.ENABLED = TRUE
        MNURUN.ENABLED = TRUE
        BTNSTOP.ENABLED = FALSE
        MNUSTOP.ENABLED = FALSE
        TIMER1.ENABLED = FALSE
        TIMER2.ENABLED = FALSE

        DGV2.ROWS.ADD(NOW, "ERROR AL EJECUTAR OPC SERVER.")
    END TRY
END SUB

PRIVATE SUB DESCONECTAOPC()
    '--- DESCONECTA OPCSERVER
    OPCMYGROUP.ISACTIVE = FALSE
    OPCMYGROUPS.REMOVE(OPCMYGROUP.SERVERHANDLE)

    OPCMYITEMS = NOTHING
    OPCMYITEM = NOTHING
    OPCMYGROUPS = NOTHING
    OPCMYGROUP = NOTHING

    OPCMYSERVER.DISCONNECT()
```

```
        'RET = MARSHAL.RELEASECOMOBJECT(OPCMYSERVER)
        OPCMYSERVER = NOTHING
        SYSTEM.GC.COLLECT()

        BTNNUEVO.ENABLED = TRUE
        MNUNUEVO.ENABLED = TRUE
        BTNGUARDAR.ENABLED = TRUE
        MNUGUARDAR.ENABLED = TRUE
        BTNELIMINAR.ENABLED = TRUE
        MNUELIMINAR.ENABLED = TRUE
        BTNCANCELAR.ENABLED = TRUE
        MNUCANCELAR.ENABLED = TRUE
        BTNRUN.ENABLED = TRUE
        MNURUN.ENABLED = TRUE
        BTNSTOP.ENABLED = FALSE
        MNUSTOP.ENABLED = FALSE
        TIMER1.ENABLED = FALSE
        TIMER2.ENABLED = FALSE

        DGV1.READONLY = FALSE
        DGV2.ROWS.ADD(NOW, "OPC SERVER DETENIDO")
    END SUB

    PUBLIC SUB ESCRIBETAG()
        TRY
            DIM ADAPTADOR AS NEW SYSTEM.DATA.ODBC.ODBCDATAADAPTER
            DIM MIDATASET AS DATASET = NEW DATASET
            DIM MYRECORDS AS INTEGER
            DIM SQL AS STRING = "UPDATE RUNTIME SET TAGWRITEVALUE='" &
FRMESCRIBIRTAG.TXTTAGVALUE.TEXT.TRIM() & "',TAGFUNCTION=1" & " WHERE
TAGID=" & DGV1.ROWS(DGV1.CURRENTCELL.ROWINDEX).CELLS(10).VALUE

            ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL, CONEXIONBD)
            MYRECORDS = ADAPTADOR.FILL(MIDATASET)
        CATCH EX AS ODBCEXCEPTION
```

```
        MSGBOX(EX.MESSAGE)
    END TRY
  END SUB

  PRIVATE SUB DGV1_DOUBLECLICK(BYVAL SENDER AS SYSTEM.OBJECT,
BYVAL E AS SYSTEM.EVENTARGS) HANDLES DGV1.DOUBLECLICK
    IF BTNRUN.ENABLED = FALSE THEN
      FRMESCRIBIRTAG.SHOWDIALOG()
    END IF
  END SUB

  FUNCTION GRIDAEXCEL(BYVAL ELGRID AS DATAGRIDVIEW) AS BOOLEAN

    'CREAMOS LAS VARIABLES
    DIM EXAPP AS NEW MICROSOFT.OFFICE.INTEROP.EXCEL.APPLICATION
    DIM EXLIBRO AS MICROSOFT.OFFICE.INTEROP.EXCEL.WORKBOOK
    DIM EXHOJA AS MICROSOFT.OFFICE.INTEROP.EXCEL.WORKSHEET

    TRY
      'AÑADIMOS EL LIBRO AL PROGRAMA, Y LA HOJA AL LIBRO
      EXLIBRO = EXAPP.WORKBOOKS.ADD
      EXHOJA = EXLIBRO.WORKSHEETS.ADD()

      ' ¿CUANTAS COLUMNAS Y CUANTAS FILAS?
      DIM NCOL AS INTEGER = ELGRID.COLUMNCOUNT
      DIM NROW AS INTEGER = ELGRID.ROWCOUNT

      'AQUI RECORREMOS TODAS LAS FILAS, Y POR CADA FILA TODAS LAS
COLUMNAS Y VAMOS ESCRIBIENDO.
      FOR I AS INTEGER = 1 TO NCOL
        EXHOJA.CELLS.ITEM(1, I) = ELGRID.COLUMNS(I - 1).NAME.TOSTRING()
        EXHOJA.CELLS.ITEM(1, I).HORIZONTALALIGNMENT = 3
      NEXT

      FOR FILA AS INTEGER = 0 TO NROW - 1
        FOR COL AS INTEGER = 0 TO NCOL - 1
```

```
                EXHOJA.CELLS.ITEM(FILA + 2, COL + 1) =
ELGRID.ROWS(FILA).CELLS(COL).VALUE
            NEXT
        NEXT
        'TITULO EN NEGRITA, ALINEADO AL CENTRO Y QUE EL TAMAÑO DE LA
COLUMNA SE AJUSTE AL TEXTO
        EXHOJA.ROWS.ITEM(1).FONT.BOLD = 1
        EXHOJA.ROWS.ITEM(1).HORIZONTALALIGNMENT = 3
        EXHOJA.COLUMNS.AUTOFIT()

        'APLICACIÓN VISIBLE
        EXAPP.APPLICATION.VISIBLE = TRUE

        EXHOJA = NOTHING
        EXLIBRO = NOTHING
        EXAPP = NOTHING

      CATCH EX AS EXCEPTION
        MSGBOX(EX.MESSAGE, MSGBOXSTYLE.CRITICAL, "ERROR AL EXPORTAR
A EXCEL")
        RETURN FALSE
      END TRY

      RETURN TRUE

   END FUNCTION

   PRIVATE SUB BTNEXPORTAR_CLICK(BYVAL SENDER AS SYSTEM.OBJECT,
BYVAL E AS SYSTEM.EVENTARGS) HANDLES BTNEXPORTAR.CLICK
      GRIDAEXCEL(DGV1)
   END SUB

   PRIVATE SUB MNUEXPORTAR_CLICK(BYVAL SENDER AS SYSTEM.OBJECT,
BYVAL E AS SYSTEM.EVENTARGS) HANDLES MNUEXPORTAR.CLICK
      GRIDAEXCEL(DGV1)
   END SUB
```

```
  PRIVATE FUNCTION VALIDAR() AS BOOLEAN
    DIM VAL AS BOOLEAN = FALSE
    TRY ' RECORRO TODO EL GRID PARA VER SI EXISTEN DOS TAGS IGUALES
      IF DGV1.ROWS(0).CELLS(0).VALUE = " " THEN
        VAL = TRUE
      END IF

      FOR I = 0 TO DGV1.ROWS.COUNT - 2
        FOR J = I + 1 TO DGV1.ROWS.COUNT - 1
          IF DGV1.ROWS(I).CELLS(0).VALUE = DGV1.ROWS(J).CELLS(0).VALUE
OR DGV1.ROWS(J).CELLS(0).VALUE = "" THEN
            VAL = TRUE
          END IF
        NEXT J
      NEXT I

      VALIDAR = VAL
    CATCH EX AS ODBCEXCEPTION
      'MSGBOX(EX.MESSAGE, MSGBOXSTYLE.CRITICAL, "MAYCENSOPC")
    END TRY
  END FUNCTION

  PRIVATE SUB TIMER2_TICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL E AS
SYSTEM.EVENTARGS) HANDLES TIMER2.TICK
    FOR I = 0 TO DGV1.ROWS.COUNT - 1 'RECORRE EL DGV1
      IF DGV1.ROWS(I).CELLS(9).VALUE <> "(NINGUNO)" THEN

        CONTADOR(I) = CONTADOR(I) + 1 'INCREMENTA EL CONTADOR

        IF DGV1.ROWS(I).CELLS(9).VALUE <> "DELTA" THEN 'HISTORIZA POR
TIEMPO
          IF CONTADOR(I) >= DGV1.ROWS(I).CELLS(9).VALUE THEN
            TRY 'GUARDA EL TAG
              DIM ADAPTADOR AS NEW
SYSTEM.DATA.ODBC.ODBCDATAADAPTER
```

```
                    DIM MIDATASET AS DATASET = NEW DATASET
                    DIM MYRECORDS AS INTEGER
                    DIM SQL AS STRING = "INSERT INTO ID" &
DGV1.ROWS(I).CELLS(10).VALUE & " VALUES('" & DGV1.ROWS(I).CELLS(3).VALUE
& "','" & FORMAT(DATETIME.NOW, "YYYY-MM-DD HH:MM:SS.FF") & "')"

                    ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL, CONEXIONBD)
                    MYRECORDS = ADAPTADOR.FILL(MIDATASET)

                    CONTADOR(I) = 0 'RESETEA EL CONTADOR
                  CATCH EX AS ODBCEXCEPTION
                    MSGBOX(EX.MESSAGE)
                  END TRY
                END IF
              ELSE 'HISTORIZA DELTA
                READVALUE(I) = DGV1.ROWS(I).CELLS(3).VALUE '  CARGA EL VALOR
ACTUAL DEL TAG

                IF READVALUE(I) <> READVALUEANTERIOR(I) THEN
                  TRY 'GUARDA EL TAG
                    DIM ADAPTADOR AS NEW
SYSTEM.DATA.ODBC.ODBCDATAADAPTER
                    DIM MIDATASET AS DATASET = NEW DATASET
                    DIM MYRECORDS AS INTEGER
                    DIM SQL AS STRING = "INSERT INTO ID" &
DGV1.ROWS(I).CELLS(10).VALUE & " VALUES('" & DGV1.ROWS(I).CELLS(3).VALUE
& "','" & FORMAT(DATETIME.NOW, "YYYY-MM-DD HH:MM:SS.FF") & "')"

                    ADAPTADOR.SELECTCOMMAND = NEW
SYSTEM.DATA.ODBC.ODBCCOMMAND(SQL, CONEXIONBD)
                    MYRECORDS = ADAPTADOR.FILL(MIDATASET)
                  CATCH EX AS ODBCEXCEPTION
                    MSGBOX(EX.MESSAGE)
                  END TRY
```

```
                READVALUEANTERIOR(I) = READVALUE(I)
              END IF
            END IF
          END IF
        NEXT I
    END SUB

    PRIVATE SUB MNUAYUDA_CLICK(BYVAL SENDER AS SYSTEM.OBJECT, BYVAL
E AS SYSTEM.EVENTARGS) HANDLES MNUAYUDA.CLICK
        PROCESS.START(SYSTEM.WINDOWS.FORMS.APPLICATION.STARTUPPATH &
"\MAYCENSOPC.PDF")
    END SUB

    PRIVATE SUB BTNACERCADE_CLICK(BYVAL SENDER AS SYSTEM.OBJECT,
BYVAL E AS SYSTEM.EVENTARGS) HANDLES BTNACERCADE.CLICK
        FRMACERCADE.SHOWDIALOG()
    END SUB

    PRIVATE SUB MNUACERCADE_CLICK(BYVAL SENDER AS SYSTEM.OBJECT,
BYVAL E AS SYSTEM.EVENTARGS) HANDLES MNUACERCADE.CLICK
        FRMACERCADE.SHOWDIALOG()
    END SUB

    PRIVATE SUB FRMPRINCIPAL_FORMCLOSING(BYVAL SENDER AS
SYSTEM.OBJECT, BYVAL E AS
SYSTEM.WINDOWS.FORMS.FORMCLOSINGEVENTARGS) HANDLES
MYBASE.FORMCLOSING
        IF (MESSAGEBOX.SHOW("AL CERRAR EL PROGRAMA SE DETENDRÁ LA
SUPERVISIÓN Y CONTROL DEL SISTEMA", "¿QUIERE SALIR DEL PROGRAMA?",
MESSAGEBOXBUTTONS.OKCANCEL, MESSAGEBOXICON.WARNING) =
DIALOGRESULT.OK) THEN
            IF BTNSTOP.ENABLED = TRUE THEN
              DESCONECTAOPC()
            END IF
            END
        ELSE
```

```
        E.CANCEL = TRUE
    END IF
  END SUB

END CLASS
```

BIBLIOGRAFÍA

[1] Wonderware, "Wonderware System Platform", http://www.wonderware.es/contents/WonderwareSystemPlatform.asp

[2] Luis Corrales Paucar, "Interfaces de Comunicación Industrial", 2007, http://bibdigital.epn.edu.ec/bitstream/15000/10020/2/PARTE%202.pdf

[3] Roberto Zacarías Cirilo Correa, " Evaluación de un Sistema de Control Distribuido Bailey para Reemplazo por el System 800xA de ABB", Universidad Nacional de Ingeniería - Lima - Perú, 2011

[4] Wonderware Training, "Training Manual, Revisión A, Wonderware System Platform 3.0 course - Part1", 2007

[5] Wonderware, "InTouch HMI Concepts and Capabilities Guide", 2007

[6] Invensys, White Paper "The Benefits of Object-Based Architectures for SCADA and Supervisory System", 2012

[7] Alejandro S. León O., "Sistemas de Control Distribuido (DCS)", Universidad de Chile, http://www.slideshare.net/alleonchile/sistemas-de-control-distribuido-dcs-7298975

[8] Erick Hernán Sánchez Ponce, "Migración del Sistema Bailey al Delta V en la fundición de SPCC - Ilo Perú", Universidad Nacional del Callao - Callao - Perú, 2011

[9] Infi90, "Application Programming Interface Windows NT Platform", http://www.infi90.com/Files/Bailey%20Infi90%20Documentation/semAPI%20Windows%20NT%20Platform.pdf

[10] Bailey, "Function Code Theory", 1995

[11] OPC Foundation, What is OPC?", http://www.opcfoundation.org/Default.aspx/01_about/01_whatis.asp?MID=AboutOPC

[12] Matrikon OPC, "¿De qué se trata, y cómo funciona? Guía para entender la tecnología OPC", http://www.matrikonopc.com/portal/downloads/whitepapers/Guia_para_entender_la_tecnologia_OPC.pdf, https://www.matrikonopc.com/downloads/836/whitepapers/index.aspx

[13] The Rovisys Company, "OPC90 Server for Bailey Command Series/Network 90/Infi 90".

[14] Wonderware, "Tech Note 667 - Supported Redundancy Configuration for Enhanced Failover Performance", 2009, https://wdn.wonderware.com/sites/WDN/Pages/Articles/TechNotes.aspx

[15] Invensys, "Invensys Expands Support for Disaster Recovery and High Availability through Virtualization Technology".

[16] Wonderware, "ArchestrA System Plarform in a Virtualized Environment Implementation Guide", 2012

[17] Wonderware, "Application Server User's Guide", 2010