COMPARACIÓN DE LA ARQUITECTURA DE LA HERRAMIENTA “SISTEMA DE ENTRENAMIENTO DE OPERADORES” (O.T.S) BASADA EN TECNOLOGÍA DE SISTEMAS EMULADOS Y DE CONEXIÓN DIRECTA.

ING. JORGE ANDRÉS PRADA MEJÍA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE FÍSICO-MECÁNICAS, ESCUELA E3T
ESPECIALIZACIÓN EN TELECOMUNICACIONES
BUCARAMANGA
2004.

COMPARACIÓN DE LA ARQUITECTURA DE LA HERRAMIENTA “SISTEMA DE ENTRENAMIENTO DE OPERADORES” (O.T.S) BASADA EN TECNOLOGÍA DE SISTEMAS EMULADOS Y DE CONEXIÓN DIRECTA.

ING. JORGE ANDRÉS PRADA MEJÍA

MONOGRAFÍA

DIRECTOR:
MAGISTER. JULIO GÉLVEZ FIGUEREDO
DOCENTE E3T

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE FÍSICO-MECÁNICAS, ESCUELA E3T
ESPECIALIZACIÓN EN TELECOMUNICACIONES
BUCARAMANGA
2004.

ABSTRACT

Title.

Evaluating architecture of an Operator Training System (OTS) based on Emulated Systems and Direct Connection technology*.

Author.

PRADA MEJIA, Jorge Andrés.**

Keywords.

Operator Training System.
Process Plants.
Emulated Systems.
Object Linking and Embedding (OLE)
OLE Process Control (OPC).
OLE Communications

Contents.

The aim of this work is to compare Direct Connection and Emulated Systems technologies, valuable tools in today's Operator Training Systems (OTS) development . The advantages and disadvantages of technologies and architectures are analyzed in order to give technical background for further suitable system selection and then assuring construction of reliable training systems in industrial plants. Besides, emphasis was made on communications between applications and computers.

Building of a Direct Connection technology and Emulated Systems-based OTS is shown, analyzing from earlier stages to final implementation but focusing on simulation rigorous model development and its implications by means of commercial simulation tools. It Involved the Analysis of several stages: Developing of rigorous model, planning and developing of proper technology and its communication architecture; those steps allowed to implement emulating interfaces for a Distribution Control System (DCS), specifically DCS I/A FOXBORO™ , establishing a communication link with the simulation rigorous model, whose features look alike real plant ones, permitting to operators a detailed manipulation of process variables in order to apply training methodologies and structures on an industrial-like environment.

***** Monography

****** Facultad de: Ingeniería Físico-Mecánicas, Programa: Especialización en Telecomunicaciones. Universidad Industrial de Santander. Director: MSc.Julio Gélvez Figueredo.

RESUMEN

Título.

Comparación de la arquitectura de la herramienta "Sistema de Entrenamiento de Operadores" (O.T.S) basada en tecnología de sistemas emulados y de conexión directa.

Autor.

PRADA MEJÍA, Jorge Andrés.**

Palabras Claves.

Sistema de Entrenamiento de Operadores.
Plantas Industriales.
Sistemas Emulados.
Object Linking and Embedding (OLE)
OLE Process Control (OPC).
Comunicaciones mediante OLE.

Descripción o contenido.

El documento pretende un análisis comparativo de las tecnologías de conexión directa y de sistemas emulados, probadas actualmente a nivel mundial en el desarrollo de Sistemas de Entrenamiento de Operadores (O.T.S) para plantas industriales, donde se analizan los pro y los contra de dichas tecnologías y arquitecturas utilizadas para la construcción de esta herramienta y en base a esta comparación, plantear criterios que permitan realizar la selección de la tecnología y arquitectura adecuada para el desarrollo de un sistema de entrenamiento de operadores de plantas industriales, haciendo especial énfasis en la comunicación de las aplicaciones y de los diferentes equipos de computo , que componen la herramienta.

Se lleva a cabo una ilustración del desarrollo de los OTS´s basados en tecnologías de conexión directa y de sistemas emulados haciendo énfasis en las etapas que preceden a la implementación final de estos y poder mostrar las exigencias requeridas en el desarrollo del modelo riguroso para simulación de una planta de procesos industriales, creado con herramientas comerciales de modelamiento y simulación.

Se analizan las siguientes etapas, luego de la creación del modelo riguroso, como la planeación, desarrollo de la arquitectura de comunicaciones y tecnología adecuada que permitieron implementar interfaces que emulan el entorno de operación de un Sistema de Control Distribuido (DCS), específicamente D.C.S I/A FOXBORO, con la finalidad de comunicar y conectar el modelo de simulación a un ambiente similar al de la planta real ,que es donde el operador debería realizar sus maniobras de operación para controlar la variedad de procesos existentes en la industria, al igual que toda una estructura y metodología de entrenamiento capaz de brindar resultados exitosos.

* Monografía
** Facultad de: Ingeniería Físico-Mecánicas, Programa: Especialización en Telecomunicaciones, Director: Magíster Julio Gélvez Figueredo.

TABLA DE CONTENIDO

LISTA DE FIGURAS

AGRADECIMIENTOS

Quisiera agradecer especialmente a mi familia, por su apoyo incondicional, mi novia por su paciencia, a todos mis compañeros de trabajo y profesores que a través de su colaboración desinteresada hicieron posible que esta monografía se culminara con éxito.

INTRODUCCIÓN

Este documento busca mostrar, analizar y concluir como abordar de la mejor forma la planeación, e implementación de los sistemas de entrenamiento de operadores para plantas industriales , abordando una critica de los sistemas basados en tecnologías de conexión directa y tecnologías de conexión emulada, analizando las diferentes arquitecturas de comunicaciones que aplican en la construcción de esta herramienta.

Brinda un significativo estudio del estado del arte de la evolución de las interfaces, objetos software e interacción entre estos, que son utilizados en la construcción de herramientas que permiten la comunicación en el control de procesos, haciendo especial énfasis en los OTS´s, que finalmente culminan en la implantación de estándares a nivel mundial, en la comunicación de procesos industriales mediante objetos vinculados mas conocidos actualmente como “OLE (Objec Linking and Embedding) for Control Process (OPC)” .

Ilustra las arquitecturas analizando y comparando dos implementaciones de OTS´s (De Conexión directa y de Sistemas emulados) haciendo referencia al software, hardware e interacción utilizado en dichas herramientas para concluir en base a los pro y los contra de cada tecnología aplicada en estos desarrollos, cuales son las pautas y criterios a tener en cuenta para llevar a cabo la construcción de esta herramienta de la mejor forma.

1. DESCRIPCIÓN

1.1. OBJETIVO GENERAL.

Describir las tecnologías utilizadas en la implementación de un Sistema de Entrenamiento de Operadores (O.T.S) para plantas Industriales, y plantear criterios que permitan realizar una comparación, selección de la tecnología y arquitectura adecuada para el desarrollo de esta herramienta.

1.2. OBJETIVOS ESPECÍFICOS.

- Describir las Tecnologías de Conexión Directa y de Sistemas Emulados, aplicada al desarrollo de Sistemas de Entrenamiento de Operadores (OTS´s) para plantas industriales.

- Analizar la arquitectura de los Sistemas de Entrenamiento de Operadores basados en las tecnologías de conexión directa y de sistemas emulados.

- Seleccionar la tecnología adecuada para la construcción de Sistemas de Entrenamiento de Operadores (OTS´s) de plantas industriales.

1.3. ALCANCES.

El presente documento pretende un análisis comparativo de las tecnologías probadas actualmente a nivel mundial en el desarrollo de un Sistema de Entrenamiento de Operadores (O.T.S.) para plantas industriales, donde se analizan los pro y los contra de las tecnologías y arquitecturas utilizadas para la construcción de esta herramienta y en base a dicha comparación, plantear criterios que permitan realizar la selección de la tecnología y arquitectura adecuada para el desarrollo de un sistema de entrenamiento de operadores de plantas industriales, haciendo especial énfasis en la comunicación de las aplicaciones y de los diferentes equipos de cómputo , que componen la herramienta.

Con la finalidad de marcar las pautas en el desarrollo de OTS´s y poder discutir criterios de selección que conlleven a tomar una decisión adecuada sobre cuál tecnología emplear en la implementación de una herramienta de tal magnitud y de tan alta utilidad.

Generar una ilustración de los Sistemas de Entrenamiento de Operadores desarrollados por el grupo de automatización del I.C.P – ECOPETROL S.A. y de las etapas que precedieron la implementación final de la herramienta, poder mostrar las exigencias requeridas en el desarrollo del modelo riguroso para simulación de una planta de procesos industriales, el cual fué creado utilizando una herramienta comercial de modelamiento y simulación, llamada Hysys, vendida a ECOPETROL S.A. por la empresa AspenTech e ilustrar y analizar las siguientes arduas etapas de planeación y desarrollo de la arquitectura de comunicaciones, y tecnología adecuada que permitieron finalmente implementar interfaces que emulan el entorno de operación del Sistema de

Control Distribuido D.C.S I/A FOXBORO, con la finalidad de comunicar y conectar el modelo de simulación a un ambiente similar al de la planta real ,que es donde el operador debería realizar sus maniobras de operación para controlar la variedad de procesos existentes en la industria del petróleo, al igual que toda una estructura y metodología de entrenamiento capaz de brindar resultados exitosos.

1.4. ESTADO DEL ARTE

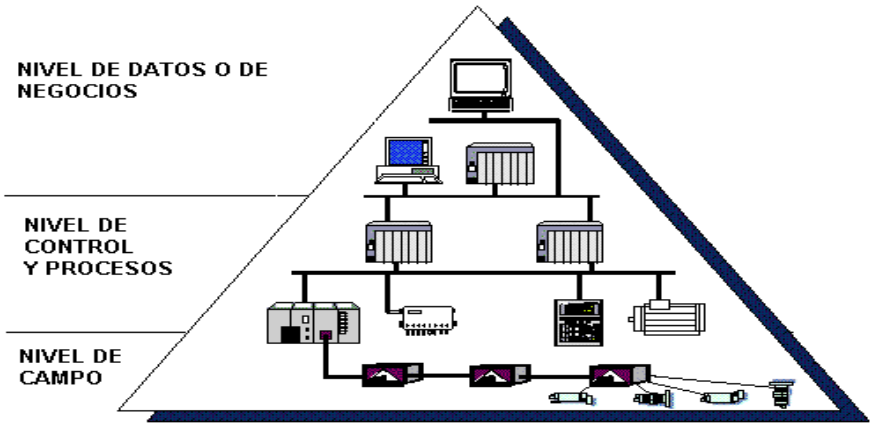
1.4.1. OPC: (OLE (Object Linking & Embedding) para Control de Procesos). La evolución de la automatización a lo largo de los últimos años ha difuminado la frontera entre las distintas capas de la pirámide industrial (ver fig1). El aumento de la importancia que se concede al intercambio de información ha llevado al objeto del OLE para Control de Procesos: OPC. OPC es un estándar que define una interfaz para la comunicación entre múltiples fuentes de datos. Se explicará que es OPC y qué aporta al mundo de la automatización desde el punto de vista de las tres partes involucradas en este mundo, los fabricantes de tecnología, las ingenierías de automatización y los clientes finales.

1.4.2 ¿Por qué surge OPC?. Si se analiza de forma general la que ha sido la evolución de la automatización a lo largo de estos años, es posible observar como el cambio ha sido revolucionario, incluso al nivel de concepto, sobre qué se considera que es tener una planta automatizada.

La arquitectura de un proceso industrial incluye los siguientes niveles (Fig.1):

- Nivel de campo o de dispositivos.
- Nivel de control o de procesos.
- Nivel de datos o de negocios.

Figura 1. Niveles en la Arquitectura de un proceso industrial.



Hoy en día existe una imperiosa necesidad de integración, que, simplificando mucho, no es ni más ni menos que establecer una comunicación entre sistemas diferentes o entre los distintos niveles de un mismo sistema (integraciones horizontal y vertical).

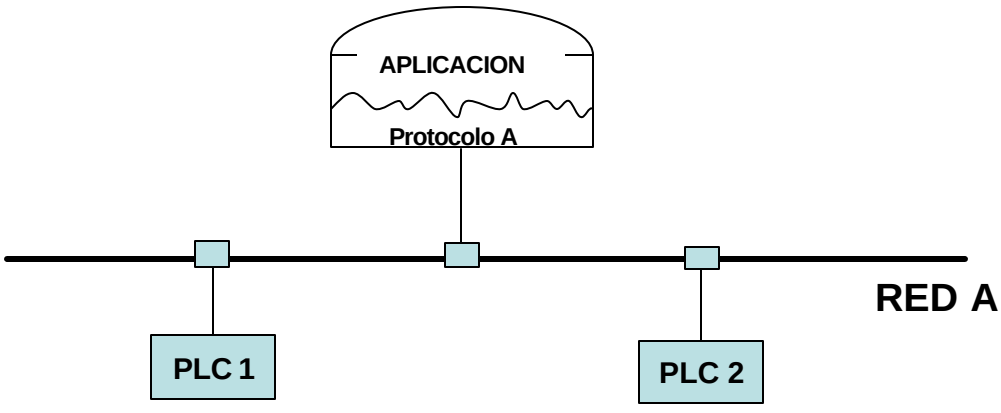
Inicialmente, esta comunicación se había venido abordando mediante el desarrollo de drivers.

Esto obligaba a tener un conocimiento del medio físico que sustentaba la comunicación así como del lenguaje o protocolo, propietario generalmente del fabricante.

Los fabricantes de hardware intentaron resolver estos problemas desarrollando sus propias librerías software, que aportaban a los desarrolladores de sistemas unas API´s (Application Programming Interfaces). Este paso, si bien facilita la tan ansiada integración, mantiene una dependencia del fabricante, que ahora también lo es del software.

El planteamiento óptimo para resolver estos problemas es dibujar una línea que separe los proveedores de hardware y software de los integradores de sistemas: un estándar.

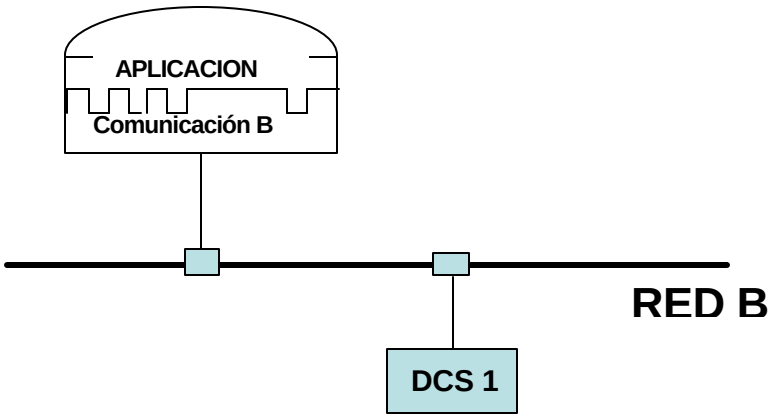
Figura 2. Desarrollo de una aplicación cliente sobre un protocolo de comunicaciones sobre varios PLC´S



Se expondrá un ejemplo de como abordar el desarrollo de una aplicación de visualización de información proveniente del nivel de control: antes, si se

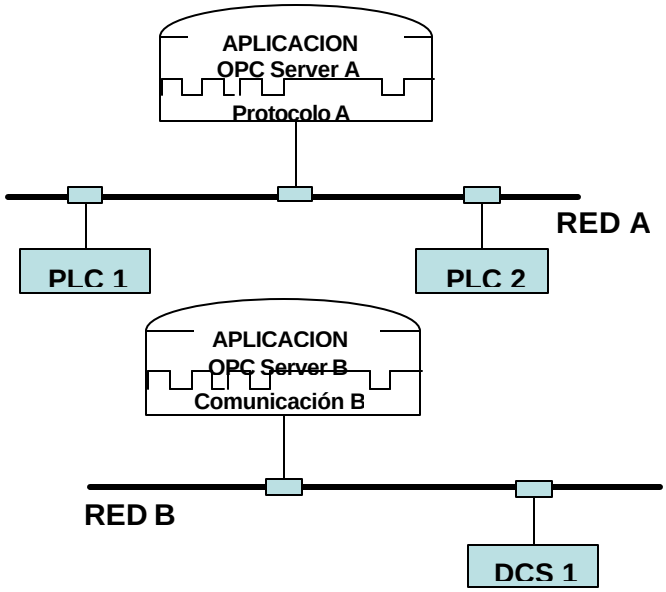
deseaba obtener una información de unos PLC´s 1 y 2 sobre una **red A**, se desarrollaba la aplicación cliente sobre el protocolo de comunicaciones A (Fig.2). Si en otro proyecto se deseaba realizar una aplicación similar, pero variaba el nivel de control, era preciso desarrollar otra aplicación cliente que implementase el tipo de comunicación B correspondiente al Sistema de Control Distribuido (D.C.S) afectado (Fig.3). Volviendo a iniciarse este planteamiento, si existe un servidor estándar de datos, nuestra aplicación cliente será capaz de adaptarse a las distintas fuentes de información a las que nos queramos conectar (Fig.4).

Figura 3. Desarrollo de otra aplicación cliente sobre un red de comunicaciones con un DCS.



Esto es OPC, que está diseñado para permitir a las aplicaciones cliente acceder a los datos del nivel de campo de una manera consistente. Es por esto que OPC aporta grandes ventajas y facilidades en la conectividad y comunicación de equipos de control en el mundo de la automatización.

Figura 4. Acceso de aplicaciones clientes a diversas fuentes de datos a través de un servidor OPC.



1.4.3. Historia. Cuando en 1990 surge Windows 3.0, se hace posible, en una plataforma "barata", correr múltiples aplicaciones simultáneamente.

Windows proporcionó un mecanismo estándar para el intercambio de datos en tiempo real: DDE (Dynamic Data Exchange).

Las principales limitaciones de DDE aparecieron pronto:

- Robustez.
- No soporte en red.
- Limitado ancho de banda.

Las mejoras a este sistema aparecieron de la mano de distintos fabricantes involucrados en el objetivo de lograr avanzar en la integración se sistemas: Wonderware, fabricante del sistema SCADA Intouch, creó el NetDDE, que trataba de resolver los problemas de soporte en red y el FastDDE para aumentar la capacidad de intercambios de datos: Rockwell creó el AdvanceDDE, etc. Pero todos estos permanecían como software propietarios y, por tanto, esto impedía adquirir el estatus de estándar.

En esta época, la tecnología para dar soporte al enlazado e incrustación de objetos se denominó OLE 1.0 (Object Linking & Embedding), que es la técnica que todos hemos utilizado para crear documentos compuestos en Microsoft Office.

En 1992 surge COM (Component Object Model). Es una especificación que establece como construir componentes que se pueden intercambiar dinámicamente (se define un componente como una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente en tiempo y espacio). Basándose en COM se desarrolló OLE 2.0. Ahora, OLE, incluye además OLE Custom Controls (OCXs) y OLE Automation (se define automatización de Microsoft como el uso del interfaz IDispatch, de forma que lenguajes de escritura (script lenguajes) pueden tener acceso a cualquier componente COM). OLE 2.0 surge, por tanto, como el sustituto de DDE: más robusto, más flexible y con un más eficiente uso de la red.

En esta misma época, un grupo que sé autodenominaba WinSEM (Windows in Science, Engineering and Manufacturing) comenzó a reunirse en las

instalaciones de Microsoft en Redmount. Los miembros de este grupo procedían del mundo del control y de la supervisión industrial, con Microsoft actuando como catalizador.

En 1994, había un claro interés en el uso de las técnicas OLE (realmente COM) para mover datos de procesos entre aplicaciones, en (casi) tiempo real. En particular los fabricantes de SCADA´s vieron la posibilidad de estandarizar la interfaz entre el núcleo de SCADA y los drivers que eran los actuales responsables de la adquisición de datos. Potencialmente esto podía beneficiar tanto a los fabricantes de SCADA como a los fabricantes de hardware:

- El fabricante de SCADA no necesitaría invertir recursos en escribir los drivers.

- El fabricante de hardware solo debería proporcionar un único driver, que podría trabajar con todo el software de Windows.

La propuesta más interesante fue expuesta por US Data en marzo de 1995. Aunque hoy ese documento, comparado con las actuales especificaciones de OPC, se vería ridículamente simple, contiene la clave de los principales conceptos actuales de OPC.

Tras el primer documento, los progresos hacia el estándar fueron muy lentos. Se vio que, en lugar de todo el grupo incluido, WinSEM, un pequeño y más enfocado grupo era necesario para asegurar la creación de un estándar. Ese grupo fue el origen de OPC Task Force.

Este grupo formado por fisher rosemount, Intellution, Intuitive Tecnology, OPTO 22 y Rockwell Software con Micrososft en el único papel de soporte y consulta emitió un primer borrador en noviembre de 1995, que fue presentado en la última reunión de WinSEM, en Redmont en enero de 1996. Tras un primer rechazo, debido a una percepción inicial de que un grupo de Elite había monopolizado el esfuerzo de la estandarización, la respuesta al borrador fue bastante favorable y muy constructiva [1].

La versión 1.0 de la OPC Specification fue emitida el 29 de agosto de 1996 y los primeros productos comerciales aparecieron apenas un año después. Se tomó la decisión de que la especificación OPC debía ser gestionada por una organización independiente y sin ánimo de lucro llamada OPC Foundation.

En1997 se publica una versión corregida 1.0 A de la OPC DATA ACCES SPECIFICATION, que es como se denomina ahora.

A mediados de 1998, el apoyo generalizado a OPC se confirmó con el Standard de la industria.

1.5. SISTEMAS DE ENTRENAMIENTO DE OPERADORES (O.T.S.).

El entrenamiento de operadores es una tarea indispensable en toda unidad de proceso y se realiza de forma incremental partiendo de los procedimientos básicos de operación segura de equipos, pasando por el conocimiento y manejo del Sistema de Control Distribuido (DCS), hasta el conocimiento

profundo de la dinámica con la que las diferentes partes del proceso responden a cambios en las variables manipuladas.

En cada una de estas etapas es indispensable el acompañamiento de operadores de mayor experiencia que guían a los novatos evitando que se puedan presentar accidentes. En todo caso, el operador aprende trabajando sobre la misma planta, creándose en algunas oportunidades situaciones inseguras que llevan a la pérdida de producto por mala operación, ó en el peor de los casos a accidentes de trabajo con las consiguientes lesiones a los trabajadores y el daño de equipos.

Estos sobrecostos debido a productos fuera de especificaciones y que deben ser reprocesados, así como las lesiones a los trabajadores y el daño de maquinaria costosa, pueden evitarse utilizando técnicas más modernas para entrenar a los operadores, acelerando el proceso de aprendizaje y reduciendo los riesgos que ocasiona el mismo al ser desarrollado ya no sobre la planta sino sobre un entorno virtual que emule la misma.

En varios proyectos de simulación a nivel mundial ya han sido ampliamente reconocidos los beneficios debidos al uso de modelos de planta virtual para entrenamiento de operadores, entre esos beneficios se resaltan:

- Garantizar las destrezas y competencias que el operador debe desarrollar para el buen desempeño de su labor.

- Arrancadas de planta más rápidas.

- Verificación de la configuración del DCS y pre-sintonía de lazos de control.
- Validación de las instrucciones de operación.

- Diseño de modificaciones para mejorar la operación.

- Modificaciones de control a partir de análisis de controlabilidad.

- Análisis del impacto ambiental de arrancadas y paradas de planta.

- Evaluación cuantitativa del riesgo para la elaboración de procedimientos de operación.

La confianza en los resultados del simulador es fundamental para su uso como herramienta de entrenamiento y para los análisis de operabilidad y controlabilidad. El modelo de la planta debe ser preciso y robusto, replicando los diferentes estados estables del proceso, así como la respuesta dinámica entre cada uno de ellos. Los equipos de la planta virtual deben ser modelados con alta fidelidad a partir de datos correspondientes a los equipos de la planta real. Igualmente las composiciones químicas y las propiedades físicas de las diferentes corrientes de entrada y salida de la unidad deben corresponder con la carga y productos del proceso real, y deben ser validados contra datos de corridas reales.

Para que la simulación se aproxime aún más a la realidad, el operador debe poder interactuar con el modelo virtual en un entorno idéntico al que enfrentará en la operación de su planta real, de esta forma también ganará conocimientos acerca del manejo de los diferentes Sistemas de Control Distribuido (D.C.S), que para el caso nuestro corresponde al D.C.S I/A FOXBORO, el cual es el encargado de realizar el control del proceso.

1.6. MODELOS Y SIMULACIÓN.

El modelo es la parte esencial, participa activamente en todo el ciclo de vida de la planta y es parte fundamental de todas las aplicaciones que se ejecutan para optimizar su funcionamiento (simulación, control y mantenimiento).

Las ventajas de obtener un modelo exacto del proceso son cada vez más claras. Las herramientas actuales ofrecen nuevas y mejores posibilidades para su desarrollo; los avances en la velocidad de cómputo hacen que modelos complejos se ejecuten en tiempos relativamente cortos. Todo esto conlleva a que cada vez se construyan más modelos de los procesos; sin embargo todavía aparecen una serie de inconvenientes:

a) **No se desarrolla un modelo Único:**

Existen modelos basados en propiedades físicas (que a su vez pueden ser estáticos o dinámicos); hay modelos empíricos basados en regresiones realizadas sobre datos experimentales, etc. Partiendo de una misma realidad, los modelos que se construyen, son sin embargo diferentes, ya que se desarrollan en función del uso (aplicación) que se va a hacer de los mismos. Se debe tender a la realización de un modelo único basado en principios físicos; este modelo se utilizará en todo el ciclo de vida de la planta: en el diseño del proceso (modelo en estado estacionario), en el estudio de la controlabilidad, comportamiento, arranques y paradas de la planta (modelo dinámico), en la adquisición de datos (por ejemplo realizar ensayos escalón para proporcionar datos para el control multivariable), en la detección de fallos, etc.

Puede haber procesos en los que no se conozca en detalle como modelarlos de forma rigurosa mediante ecuaciones que representen sus características fisicoquímicas; en este caso se debe investigar como desarrollar formalismos que permitan “mezclar e integrar” en un mismo modelo las ecuaciones que representen los fenómenos físicos y las obtenidas de forma empírica, manteniendo externamente la consistencia del modelo único.

b) Modelamiento de un proceso y no de todo el sistema:

No se suele considerar el sistema global a la hora de desarrollar el modelo. Esta es otra dirección en la que se debe avanzar y que permitirá realizar, entre otras cosas, el diseño inicial del proceso y el control de forma integrada, obteniendo un mejor comportamiento de la planta. Así los modelos únicos e integrados deberían incluir:

- La información contenida en los distintos planos y diagramas de tuberías e instrumentos entre los diferentes suministradores, constructoras y/o subcontratistas, clientes etc.

- Base de datos con las especificaciones técnicas de los distintos equipos, para utilizarla como fuente de información para mantenimiento.

- Diagramas de Ingeniería y construcción, útiles para su uso posterior como diagramas de operación y mantenimiento (Reformas, mejoras y actualizaciones).

- Características de materia prima y de productos terminados.

Toda la información necesaria para las etapas del ciclo de vida debería unificarse bajo un único modelo.

La simulación es la ejecución de un experimento sobre los modelos construidos, con el fin de obtener unos resultados que reflejen el comportamiento de ese modelo; idealmente debería ser distribuida y en tiempo real. Distribuida en el sentido de interoperabilidad y reusabilidad, de modo que se pueda tener el modelo de una planta a partir de modelos de diferentes propietarios que se comunican mediante una interfaz común. El requisito de tiempo real es importantísimo, ya que este permitiría:

- Sustituir el módulo de predicción del control multivariable, teniendo modelos rigurosos no linealizados en vez de los modelos linealizados identificados que se utilizan normalmente.

- Realizar un entrenamiento de operadores de modo que las interacciones con el proceso sean en tiempo real.

- Realizar el control de planta.

Con un buen modelo de la misma y una respuesta en tiempo real, se puede realizar un control basado en modelos que sustituya al control convencional.

Sin embargo, la simulación en tiempo real no es fácil, ya que implica que los resultados que obtengan los algoritmos del modelo se procesen en el tiempo esperado y garantizar este comportamiento para cada vez que se requiera

hacer uso de el. Esta parte es la más difícil de conseguir hoy en día y por eso es indispensable disponer de los equipos de simulación con niveles de procesamiento adecuados.

2. ESTÁNDAR QUE DEFINE LA INTERFAZ PARA LA COMUNICACIÓN ENTRE MÚLTIPLES FUENTES DE DATOS – OPC: OLE PARA CONTROL DE PROCESOS.

Introducidos los conceptos básicos y generales, la finalidad es exponer la estructura del OPC, y su aplicación en los Sistemas de Entrenamiento de Operadores (OTS´s), así como la tecnología sobre la que se sustenta.

2.1 DESCRIPCIÓN.

***2.1.1 Conceptos de COM* (Component Object Model).** Como se ha mencionado al analizar la evolución histórica de OPC esta tecnología se sustenta sobre COM. Por tanto, y para una mejor comprensión trataremos los conceptos generales sobre COM que se van a manejar más adelante.

En 1992, surge COM como ya se ha dicho, y basándose en COM se desarrolló OLE 2.0. Aunque COM es en si mismo independiente del lenguaje de programación; esta desarrollado en C++, ya que es el leguaje en el que están escritos la mayoría de sus componentes. Programado COM directamente con C++, se entienden más cosas que si se emplean otros lenguajes como Visual Basic o Java.

COM proporciona el estándar que componentes y clientes siguen para asegurar que ellas puedan trabajar juntos.

En lugar de construir una aplicación monolítica, es preferible construirla a base de componentes. La manera más sencilla y básica de desarrollar una aplicación es reutilizando software.

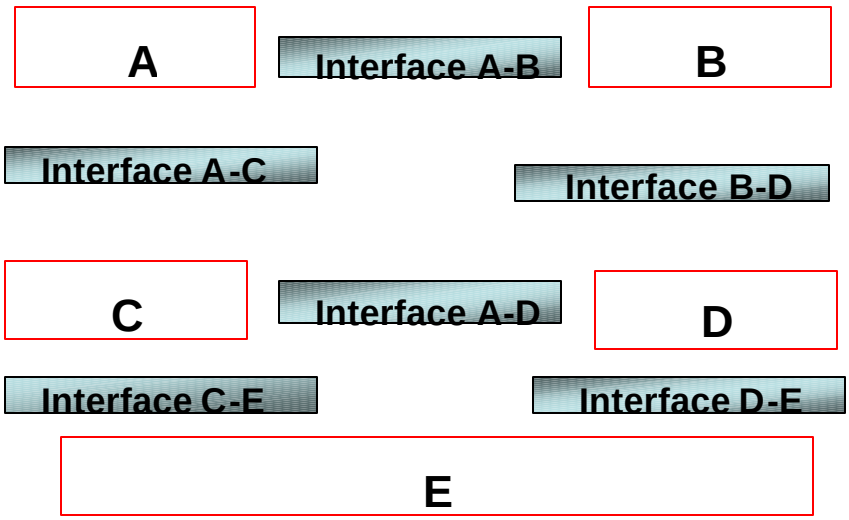
La idea de reutilizar software es buena y la forma de hacerlo es mediante el uso de componentes y no empleando el propio código fuente.

Un componente es una pieza de software reutilizable. Los componentes deben encapsular los detalles sobre como están implementados. El uso de los componentes se hace a través de interfaces (Fig. 5.).

Si volvemos a los planteamientos de integración antiguos, con API´s ya habría sido un avance importante que esas interfaces fuesen estándar, de forma que el mismo código simplemente se debiese ligar contra una librería u otra.

Sin embargo COM da un paso más: los componentes se ligan dinámicamente. Los componentes COM son DLL´s (Dinamic LINK Library) o EXE´s (execute).

Figura 5. Uso de componentes a través de interfaces.



COM es más que una especificación. COM tiene un API, la librería COM, que proporciona servicios de gestión de componentes que son útiles para todos los clientes y componentes.

Hasta ahora se ha empleado la palabra interfaces, que se refiere a las funciones que podría exportar una librería; en COM, una interfaz es algo más: es una estructura de memoria específica, que contiene una matriz de puntero a funciones. Cada elemento de la matriz contiene la función implementada por el componente.

En COM, las interfaces lo son todo. Un cliente, que requiera reutilizar un componente solo deberá conocer las interfaces que conforman el componente y su función.

Decir que un componente es solo la implementación de una interfaz no tiene mucho sentido. Después de todo, un interfaz sin implementación no hace nada. Sin embargo, un componente puede ser eliminado de una aplicación y reemplazado por otro componente, y mientras este soporte las interfaces que el antiguo componente, la aplicación seguirá funcionando.

En COM, los interfaces no cambian nunca. Una vez que un interfaz ha sido publicado, debe permanecer siempre igual. En lugar de cambiar un interfaz existente, para actualizar un componente, se debe crear una nueva interfaz y añadirla al componente. Esto es fundamental para que pueda funcionar el avance de versiones.

Características de COM.

Polimorfismo es la habilidad de diferentes objetos de ser tratados de la misma forma. Si dos componentes distintos soportan la misma interfaz, el cliente puede usar el mismo código para manipular ambos componentes. El cliente puede, por tanto, tratar a los diferentes componentes polimórficamente.

Múltiples interfaces facilitan el polimorfismo. Cuantos más interfaces soporten un componente, más pequeños deben ser estos. Una interfaz pequeña representa un único comportamiento, mientras que una interfaz grande representa muchos comportamientos. Cuantos más comportamientos represente una interfaz, más específica es la situación en que se usará. Cuanto más específica sea una interfaz, mas podrá ser reutilizada por distintos componentes. Si una interfaz no es reutilizada, el código que usa esa interfaz no puede ser reutilizado.

Una característica adicional de COM son los objetos conectables, que proporciona interfaces “salientes” a sus clientes además de ser interfaces “entrantes”, de forma que los objetos y sus clientes pueden establecer una comunicación bidireccional.

2.1.2 Estructura de OPC. OPC es un conjunto de interfaces COM que permiten intercambiar datos entre distintas fuentes.

Los objetivos en el diseño de OPC fueron:

- Fácil de implementar.
- Flexible para acomodarse a la necesidades de múltiples fabricantes.
- Proporcionar un alto nivel de funcionalidad.
- Permitir una operatividad eficiente.

La arquitectura de un sistema OPC está basada en el tradicional sistema cliente/servidor.

Las interfaces COM de los objetos OPC son implementados por OPC Server.
Un OPC Client puede conectarse a OPC Servers proporcionado por uno o más fabricantes, utilizando para ello las interfaces definidas en la especificación (Fig.6).

Figura 6. Conexión de un OPC cliente a diversos OPC servidores.

Estas interfaces pueden ser de dos tipos:

- Obligatorias: son las que todo OPC Server debe implementar para que sea considerado como tal.

- Optativas: ofrecen capacidades adicionales; el fabricante puede optar por incluirlas o no.

De cualquier modo, si un OPC Server ofrece una cierta interfaz, debe implementar todos los métodos que estén definidos para esa interfaz.
El servidor suministrado por el fabricante de hardware determina los dispositivos y datos a los cuales el servidor tiene acceso: los nombres de los datos y los detalles sobre como el servidor físicamente accede los datos.

La OPC Foundation, que en la actualidad cuenta con más de 250 miembros procedentes de los principales líderes mundiales en control y supervisión de procesos industriales, tiene como uno de sus objetivos prioritarios la publicación de especificaciones para la industria, tan rápido como sea posible. Con esto en mente, el alcance de los primeros documentos se ha limitado a las áreas comunes a todos los fabricantes:

- OPC Data Acces: proporciona acceso a datos en tiempo real (v2.05, Enero 2002).

- OPC Alarm & Events: proporciona información de ocurrencia de un evento o alarma especifico (v 1.02, Noviembre 1999).

- OPC Historical Data Acces: proporciona acceso a los datos históricos almacenados (v 1.1, Enero 2001).

Posteriormente se emitieron:

- OPC Batch: para el envío de procedimientos y su monitorización (v1.0 Abril 2000).

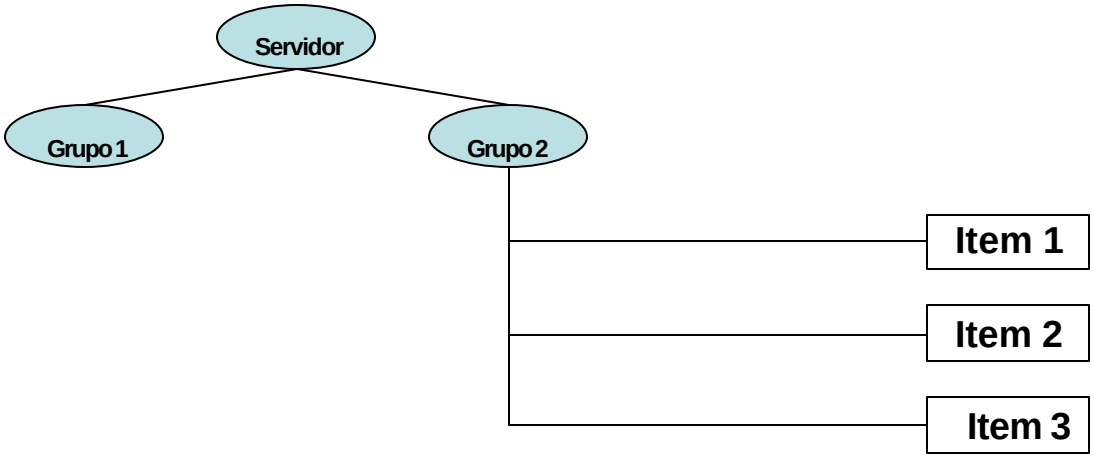
- OPC Security: para la identificación de los clientes (v1.0 Octubre 2000).

Ahora se está trabajando en la versión 3.0 de OPC Data Acces, en OPC XML, y en OPC Data eXchange.

OPC Data Acces proporciona un mecanismo eficiente de lectura y escritura de datos entre una aplicación y un dispositivo de control de procesos.

Un servidor de acceso a datos OPC comprende varios objetos: el servidor, el grupo y el ítem (Fig.7). El objeto servidor mantiene información sobre el servidor y sirve como contenedor para objetos de grupos OPC. Un objeto OPC Group mantiene información sobre si mismo y proporciona el mecanismo para contener y organizar lógicamente ítemes OPC.

Figura 7. Arquitectura de los Objetos de un servidor OPC.



Un grupo proporciona al cliente una forma lógica de organizar los datos. Hay dos tipos de grupos, públicos y locales o privados. Los públicos son para compartir a través de múltiples clientes, y los privados son, locales a un cliente determinado.

Dentro de cada grupo se pueden definir uno o más ítemes OPC. El ítem OPC representa la conexión a la fuente de datos, pudiendo tratarse de cualquier tipo de datos, desde un simple valor Booleano, hasta una cadena de caracteres.

La mayoría de los servidores OPC de acceso a datos proporcionan un "explorador de ítemes", que permite conocer la estructura jerárquica de sus datos y facilita la adición de ítems a los grupos.

A la hora de leer y escribir valores de los ítemes, se definen métodos síncronos y asíncronos, siendo los primeros adecuados para aplicaciones simples, mientras que los segundos se deben utilizar en procesos complejos, que requieran el intercambio de gran cantidad de información con la máxima eficiencia.

Además, en la especificación OPC de acceso a datos, se definen una serie de parámetros configurables, con la tasa de actualización, la actividad, o los porcentajes de banda muerta, que ayudan a crear aplicaciones cliente que aprovechen al máximo las capacidades del servidor.

OPC Alarm y Events permite que los clientes sean notificados de la ocurrencia de un evento determinado o una condición de alarma.

En OPC una alarma es una condición anormal, un caso especial de una condición, entendiéndose por condición el estado de uno de los objetos contenidos por el OPC Event Server. Las alarmas o condiciones pueden ser de estado único o multiestado.

Por otro lado, un evento es una ocurrencia detectable, que tiene significado para el OPC Server, el dispositivo que representa y sus Clientes OPC. Un evento puede o no estar asociado a una condición.

Los eventos pueden ser:

- Simples: no relacionados con estados.
- Tracking: Indican la realización de una operación y el alcance de un objetivo.
- Condition: son alarmas.

El cliente se suscribe a las alarmas o eventos que quiere ser notificados.

OPC Historical Data Access proporciona acceso a los datos históricos almacenados por los servidores. Con el objetivo de integrar datos, todos los niveles de negocio, los datos históricos deben considerarse como otro tipo de datos.

Hay varios tipos de servidores de históricos:

- Simples servidores de tendencias: simples registros de tiempo, valor y calidad.

- Complejos tratamientos de datos. Proporcionan resúmenes de datos, funciones de análisis como medias, mínimos, máximos etc. Pueden soportar actualizaciones de datos e historia de las actualizaciones.

OPC Batch proporciona un conjunto de interfaces para facilitar la creación, gestión y supervisión de procesos por lotes (recetas); se basan en las normas IEC61512-1.

OPC Security permite que los servidores OPC puedan identificar a los clientes que acceden a ellos y configurar sus permisos y niveles de acceso. Se definen tres niveles de seguridad:

- Sin seguridad.
- Seguridad Básica.
- Seguridad Extendida.

OPC XML posibilita el uso de OPC a través de Internet. La primera versión de la especificación, cuya aparición es inminente, utiliza el protocolo SOAP (Simple Object Acces Protocol), que es independiente de la plataforma utilizada, para intercambio de mensajes en formatos XML entre procesos.

OPC Data Exchange está en proceso de desarrollo. El estándar OPCDX proporcionara intercambio de datos y comunicaciones entre servidores en redes de tipo Ethernet, independiente del protocolo utilizado. Para la elaboración de la primera especificación, se ha creado un grupo de trabajo, en el que además de la OPC Foundation, participan organizaciones como

ControlNet Internacional, Fieldbus Foundation, OpenDevicesNet Vendor Association y PROFIBUS Internacional.

Las especificaciones OPC siempre contienen dos conjuntos de interfaces:

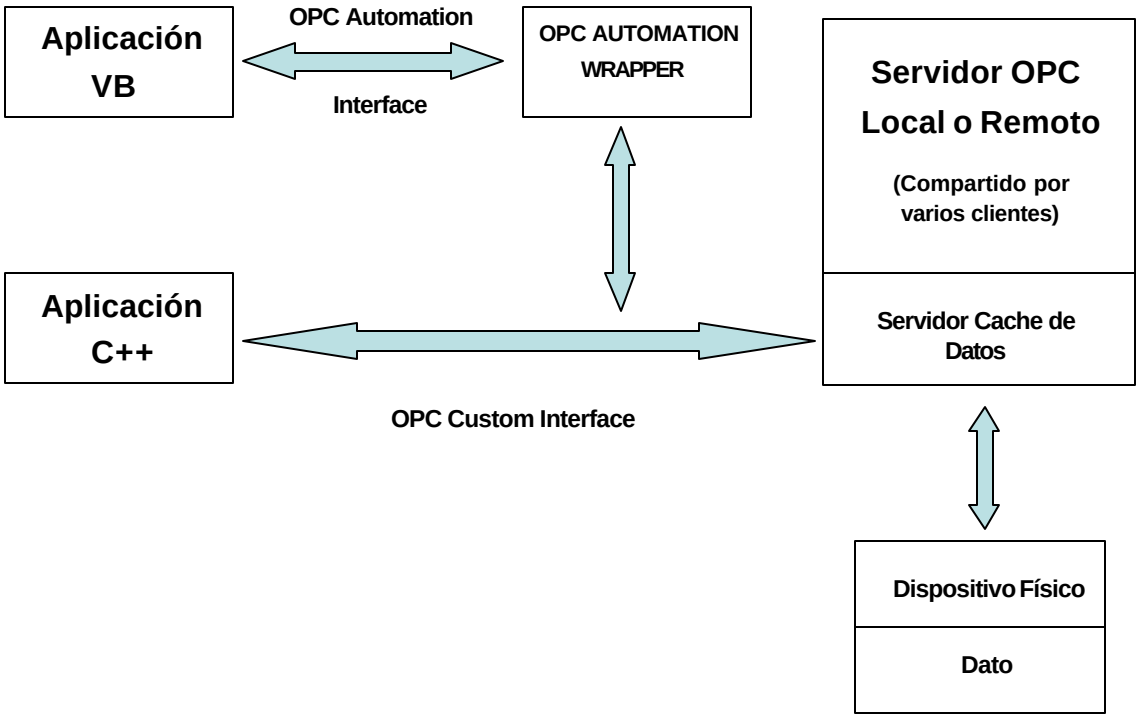
- Custom Interfaces.
- Automation Interface.

Las especificaciones establecen para las interfaces el comportamiento que se espera de ellos, no la implementación de las mismas.

Como toda implementación COM, la arquitectura de OPC es un modelo Cliente/servidor, donde el componente OPC Server Proporciona una interfaz a los objetos OPC y los gestiona.

Los OPC Servers están escritos en C/C++. Un OPC Client que se desarrolle en C++ deberá emplear el Custom Interface, mientras que un cliente que se desarrolle en Visual Basic deberá emplear el Automation Interface, debiendo usar un OPC Automation Wrapper, que actué como pasarela entre ambos conjuntos de interfaces (fig8).

Figura 8. Arquitectura de OPC.



2.1.3. Arquitectura Distribuida.

Análisis de como se plantea la arquitectura distribuida, la cual va a ser usada en la creación de la herramienta “Sistema de Entrenamiento de Operadores”.

Si en el pasado, el uso de un Driver propietario estaba restringido a una única aplicación, se ha planteado que ahora se puede acceder al mismo OPC Server desde diferentes aplicaciones OPC Clients.

Avanzando más, el acceso al OPC Server es aun más flexible. La capacidad del multicliente no solo cobra ventaja en el PC Local en que reside el servidor, si no que puede ser usada remotamente por medio de DCOM (Distributed COM).

DCOM apareció por primera vez en agosto de 1996 en el marco de Windows NT 4.0. Se puede entender como una extensión de COM que permite la comunicación entre objetos que se encuentran en máquinas distintas a través de una red de área local (LAN) o de una red de área extensa (WAN).

DCOM oculta por completo la ubicación de un componente, ya se encuentre en el mismo proceso o en una máquina a miles de Km de distancia. En todos los casos la forma en la que el cliente se conecta al componente y llama a sus métodos es idéntica.

Se ha evolucionado desde que el cliente y el servidor residen en la misma máquina, hacia la conectividad con un servidor residente en otra máquina.

COM y DCOM fueron creados por Microsoft para operar sobre un sistema operativo Windows, aunque en la actualidad existen empresas que han desarrollados versiones de DCOM para otros sistemas operativos y plataformas.

Como resultado de este planteamiento, uno de los temas que más preocupan, tanto a los desarrolladores como a consumidores de productos OPC, es la seguridad de sus aplicaciones distribuidas.
Afortunadamente con DCOM se pueden implementar aplicaciones distribuidas de forma segura, sin necesidad de añadir ningún código extra. Así como el modelo de programación DCOM encapsula la localización de un componente, también encapsula sus requerimientos de seguridad. De esta forma, el mismo código binario utilizado en un entorno de una sola máquina, donde no se aplican consideraciones de seguridad, se pueden usar en un entorno distribuido con un alto nivel de seguridad.

Para conseguir esta transparencia en cuanto a seguridad, DCOM permite que los desarrolladores y los administradores de las máquinas configuren los requisitos de seguridad de cada componente.

Cuando un cliente efectúa una llamada a un método o crea una instancia de un componente, DCOM obtiene el nombre de usuario de dicho cliente asociado con el proceso; Windows NT garantiza la autenticidad de esta credencial de usuario. A continuación, DCOM pasa este nombre de usuario a la máquina o proceso en el que “corre” el componente. El DCOM de la máquina en la que reside el componente se encarga de validar el nombre de usuario: chequea la lista de control de acceso para el componente. Si el nombre de usuario del cliente no está incluido en esta lista (ya sea directamente o indirectamente como un miembro de un grupo de usuarios), DCOM simplemente rechaza la llamada antes incluso de que el componente se entere.

Una vez que el usuario ha sido autentificado, DCOM define dos tipos de seguridad:

- Seguridad de Activación, que controla qué clientes pueden acceder a un determinado componente.

- Seguridad de llamada, que controla qué clientes pueden efectuar llamadas a los métodos del componente.

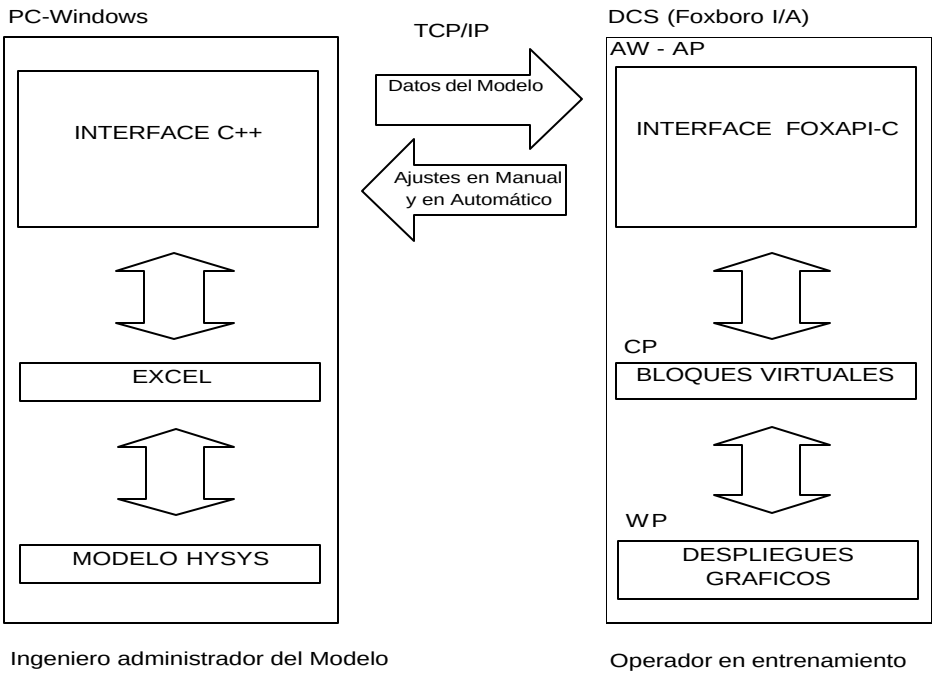
3. ARQUITECTURA DE LA HERRAMIENTA "SISTEMA DE ENTRENAMIENTO DE OPERADORES" BASADO EN TECNOLOGÍA DE CONEXIÓN DIRECTA

3.1 INTRODUCCIÓN A LA ARQUITECTURA.

El objetivo de entrenar al operador en un entorno muy similar al de operación no se cumpliría si no se le permitiera interactuar con interfaces similares a las del Sistema de Control Distribuido (DCS) de la planta. Es por esto que se diseñó una plataforma que le permitiera manipular en el modelo las variables de proceso disponibles en la planta real, a través de las mismas interfaces de operación que existen en el cuarto de control de la unidad, ocultando así los detalles del modelo construido en Hysys.

Esta plataforma permite leer/escribir datos del modelo en Hysys a través de una hoja de cálculo en Excel, que a su vez transmite/recibe datos por intermedio de una interfaz desarrollada en C++ hacia el servidor de aplicaciones del DCS I/A FOXBORO donde se encuentran configurados los despliegues de operación de la planta. El resultado final es que el operador observa, monitoriza y controla el modelo de la unidad construido en Hysys, utilizando para ello el mismo formato de información utilizado para controlar el proceso real.

Figura 9. Arquitectura de la herramienta “Sistema de Entrenamiento de Operadores” con tecnología de conexión directa”.



3.2. FUNCIONAMIENTO.

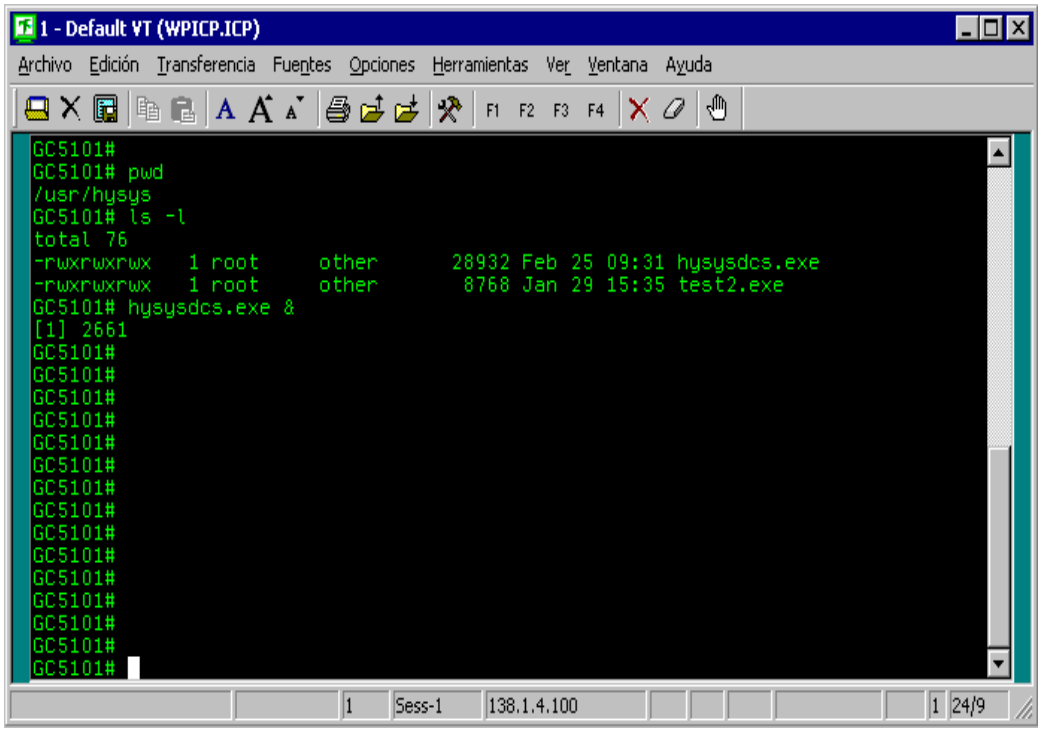
Para el correcto funcionamiento de toda la plataforma deben iniciarse de forma apropiada las diferentes aplicaciones relacionadas:

Lo primero que debe hacerse es iniciar en la estación correspondiente del DCS (AW ó AP) la interface con FoxAPI que permitirá recibir comandos en el Sistema de Control Distribuido y enviar los ajustes realizados por el operador hacia el computador corriendo el modelo en Hysys. Para esto es necesario iniciar una sesión remota con dicha estación (usando para ello la aplicación telnet) y ejecutar los siguientes comandos bajo el entorno Unix:

cd /usr/hysys

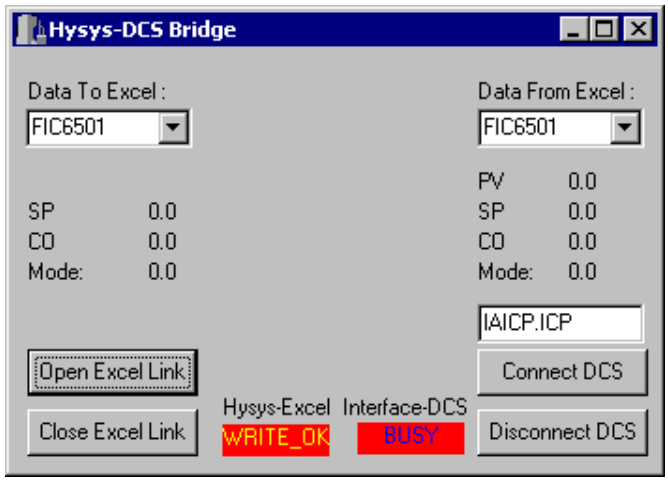
hysysdcs.exe &

Figura 10. Ventana de terminal remoto en estación del DCS.



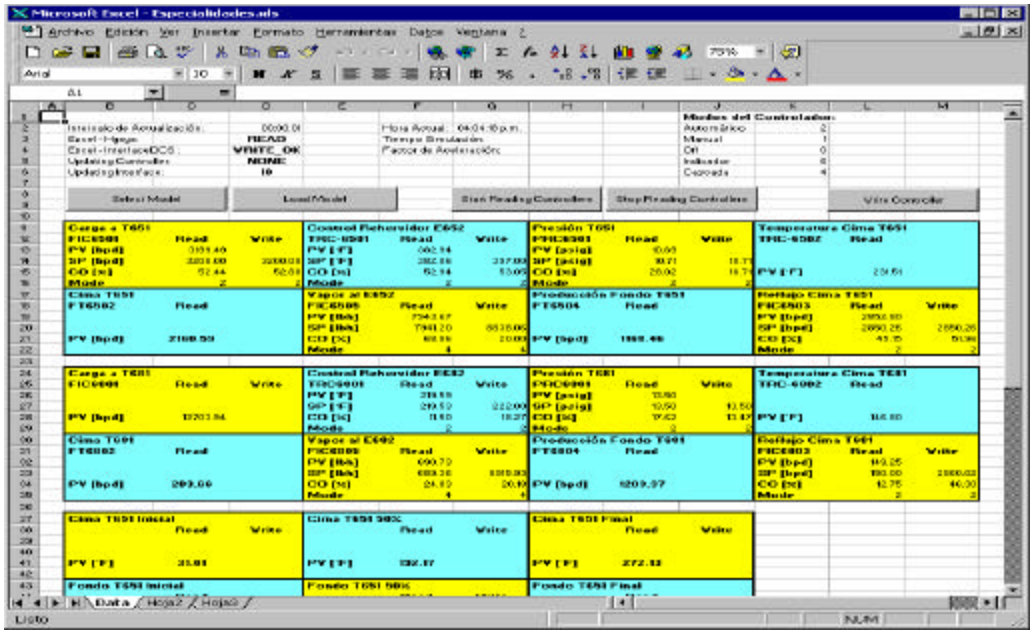
Luego de digitar estos comandos es importante anotar el número del proceso generado en Unix para posteriormente eliminarlo al terminar la sesión de entrenamiento. A continuación debe iniciarse la aplicación interfaz que corre sobre el computador con plataforma Windows:

Figura 11. Interface Excel - DCS.



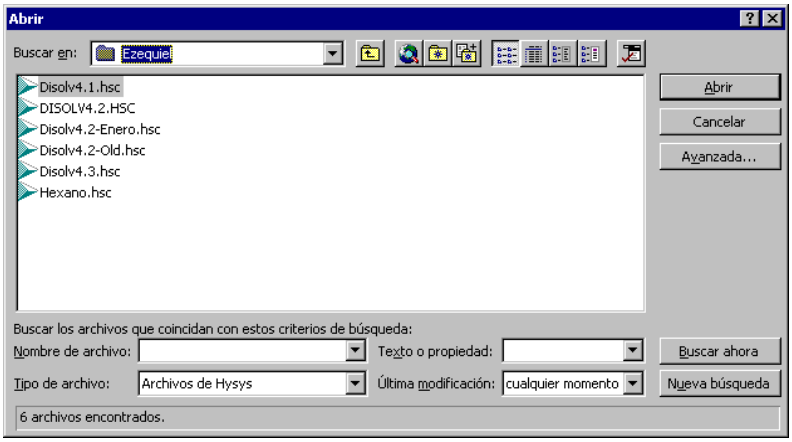
Oprimiendo el botón “Open Excel Link” esta interface abre de forma automática la hoja de cálculo en Excel que se encargará de interactuar directamente con el modelo en Hysys:

Figura 12. Hoja de Excel de lectura/escritura del modelo en Hys ys.



En la parte superior de esta hoja de Excel debe inicialmente pulsarse la opción “Select Model” a través de la cual se seleccionará el archivo .HSC correspondiente al modelo a simular.

Figura 13. Selección del modelo a simular.



Una vez se ha seleccionado el archivo correspondiente al modelo se debe oprimir la opción “Load Model” que inicializará Hysys.

Figura 14. Hysys - Herramienta de modelamiento riguroso.

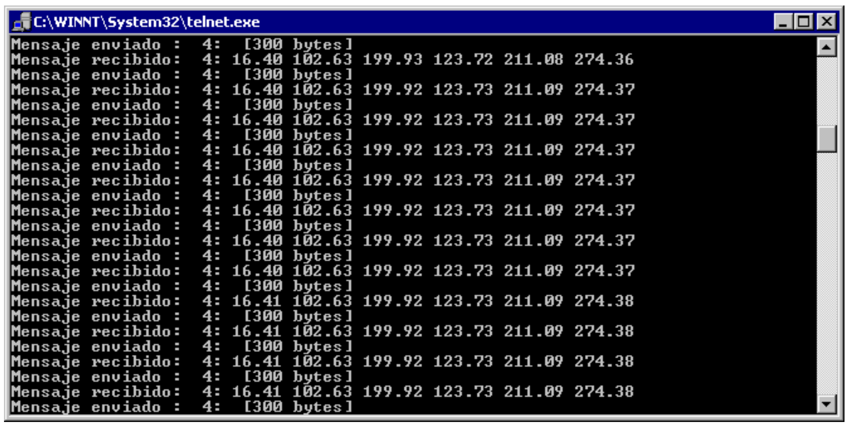
Una vez se ha iniciado apropiadamente Hysys el modelo debe automáticamente comenzar a correr (el módulo integrador de Hysys se activa automáticamente desde la hoja de cálculo de Excel). En dicho integrador es posible configurar la velocidad con la que Hysys solucionará las ecuaciones matemáticas asociadas al modelo de la planta.

A continuación debe seleccionarse el botón "Start Reading Controllers" ubicado en la parte superior de la hoja de cálculo de Excel. Con esta opción comienza el intercambio de información entre Hysys, Excel y la aplicación interface desarrollada en C++. Dichas transacciones de lectura/escritura se pueden observar a través de las celdas "Excel – Hysys" y "Excel – InterfaceDCS" que pueden contener los estados de READ/WRITE y READ_OK/WRITE_OK respectivamente. Cuando la primera de estas celdas se encuentra en estado READ y la segunda se encuentra en estado WRITE_OK, desde Excel se invoca la subrutina de lectura de los controladores PID del modelo desde Hysys (los valores leídos de Hysys corresponden al setpoint, medición de la variable controlada, señal de salida del controlador, y estado del controlador). Cuando la primera celda se encuentra en estado WRITE y la segunda se encuentra en estado READ_OK, desde Excel se invoca la subrutina de escritura de los controladores hacia Hysys (sólo se escribe el setpoint, ó la salida del controlador, ó el estado del controlador, de aquel lazo de control que ha cambiado sus valores con respecto a la última operación de escritura). Los valores escritos hacia el modelo en Hysys son aquellos que son recibidos por la interface que corre en el PC y que han sido enviados desde el DCS.

El siguiente paso es seleccionar el botón "Connect DCS" ubicado en la esquina inferior derecha de la aplicación de interface. Dicha opción inicia el envío de información del modelo hacia los despliegues gráficos en el DCS, y la

recepción de ajustes en manual ó automático realizados por el operador en el DCS.

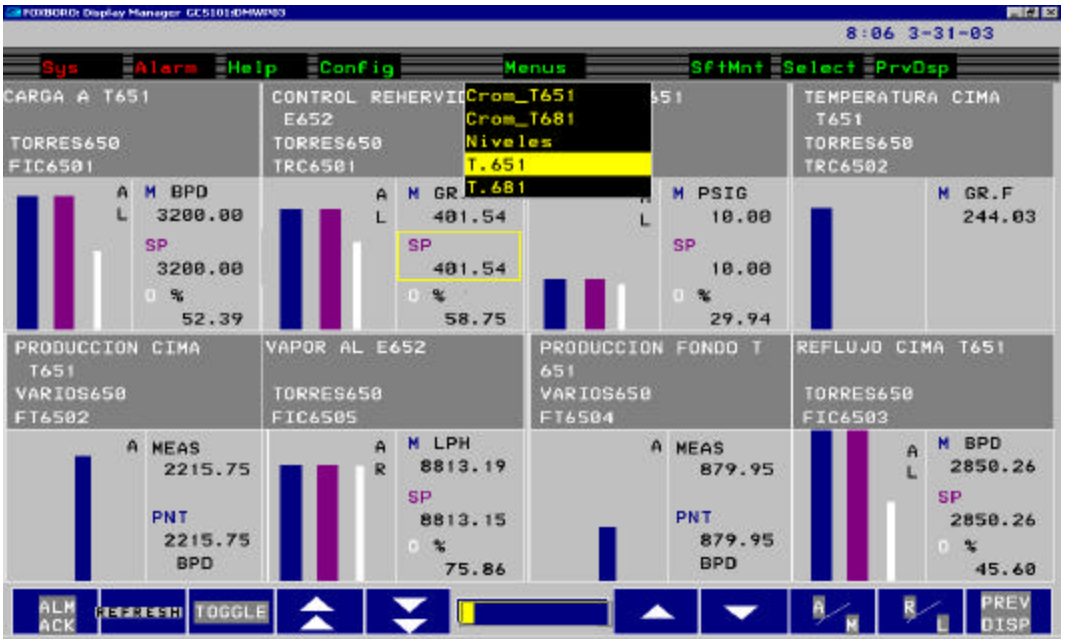
Figura 15. Ventana de terminal remoto en el DCS recibiendo/enviando información.



Cuando el operador realiza algún ajuste en los despliegues gráficos del DCS, dicho ajuste es recibido en la aplicación interfaz que corre en el computador y es comunicado a la hoja de cálculo de Excel en la celda “Updating Controller” en la cual se encuentra el nombre del controlador que debe ser actualizado en el modelo en Hysys.

El funcionamiento de las diferentes rutinas que realizan la transacción de información de entrada/salida se ha coordinado de forma tal que la demora en tiempo es mínima y el operador puede observar los efectos de cualquier cambio efectuado en el proceso (en manual manipulando directamente apertura de válvulas ó en automático manipulando setpoints) casi de forma inmediata. Las interfaces manipuladas por el operador en el DCS son las siguientes:

Figura 16. Interface de operación típica.



En todos estos despliegues gráficos existe un botón “Refresh” cuya función es indicarle a la interfaz tanto en el D.C.S I/A FOXBORO como en el computador cuáles datos deben ser enviados y recibidos, a fin de disminuir el tráfico de información por la red y aumentar la velocidad de respuesta de la plataforma de entrenamiento. El bloque de datos a actualizar se puede observar en la celda “Updating Interface” en la parte superior de la hoja de cálculo de Excel, y puede tener los siguientes estados: **I0/I1/I2/I3/I4/I5**.

3.3. DETALLES DEL DESARROLLO DEL PROTOCOLO DE COMUNICACIÓN DE LA HERRAMIENTA DE CONEXIÓN DIRECTA.

3.3.1 Interfaces en el D.C.S I/A FOXBORO. Para el desarrollo de la rutina ejecutándose sobre el DCS, se dispuso de la siguiente plataforma: Sistema de Control Distribuido (D.C.S) Foxboro I/A, estación de aplicaciones AP51 con sistema operativo SunOS versión 5.5.1 de Solaris (una versión comercial de Unix), software de aplicaciones de control de Foxboro versión 4.3 y librería FoxAPI para el desarrollo de interfaces externas al sistema de control.

La interfaz en el servidor de aplicaciones del DCS I/A Foxboro fué desarrollada en lenguaje ANSI C y compilada sobre la plataforma de Solaris. La estructura de esta aplicación contiene los siguientes módulos:

Rutinas de comunicación:

Establish get_connection Fireman	Se encargan de gestionar la conexión con la estación Windows utilizando para ello los sockets, o comunicación por paquetes de datos utilizando el protocolo TCP/IP.
read_data write_data	Se encargan de leer/escribir un número fijo de bytes (300bytes) del buffer de datos asociado a los sockets cuando se hace recibo/envío de información hacia el DCS I/A FOXBORO ó hacia la estación Windows.

Rutinas principales:

SendData	Se encarga de leer los valores de setpoints, aperturas de válvula y status de controladores modificados por el operador en su interfaz. Envía luego estos datos a la estación Windows.
process_request	Recibe de la estación Windows los valores actualizados de las variables que deben refrescarse en las interfaces de operación (variables controladas, indicadores, status de alarmas, status de controladores, setpoints y aperturas de válvula leídos del Modelo en Hysys).
Main	Dentro de un bucle se encarga de duplicar la rutina en memoria para atender cada nueva solicitud de conexión de una estación Windows. En cada oportunidad se crea un hilo que invoca Secuencialmente a process_request y SendData para refrescar las interfaces de operación y enviar los comandos del operador al modelo corriendo en la estación Windows.

Tanto en las rutinas SendData como en process_request fue necesario el uso de bloques configurados en memoria para guardar los valores leídos y escritos, hacia y desde las pantallas de operación. Estos bloques son creados por el software de aplicación de Foxboro y tienen como fin, entre otros, el servir de puente entre todas las aplicaciones gráficas con las que interactúa el operador, y cualquier aplicación desarrollada por usuarios sobre el sistema operativo. Para tener acceso a estos bloques de datos se utilizó la librería FoxAPI que

incorpora llamadas de lectura/escritura que pueden utilizarse para manipular el contenido de dichos bloques:

Los bloques utilizados permitieron el uso de variables enteras (para identificar por ejemplo el número de la pantalla de operación a refrescar en el DCS), variables flotantes (para guardar los valores de los setpoints y aperturas de válvula modificados por el operador), y variables de tipo booleano (guardando el status de alarma de los controladores).

Debido a la cantidad elevada de variables a transmitir en ambos sentidos (desde la estación Windows hacia el DCS, y desde el DCS hacia la estación Windows) fue necesario verificar la pantalla activa en la consola de operación del DCS, para refrescar tan solo sus variables asociadas.

Finalmente, en la pantalla activa en el DCS es leída toda la información asociada y enviada a la estación Windows:

La longitud de mensajes entre ambas estaciones fue limitada a 300 bytes, en los cuales debe enviarse toda la información requerida por la aplicación.

De igual forma, una vez es recibido un mensaje de la estación Windows para actualizar la información de las consolas de operación en el DCS, se sigue un proceso de lectura de mensaje del buffer asociado al socket de comunicaciones, y la posterior escritura de las variables apropiadas en los bloques de datos residentes en memoria del DCS:

3.3.2 Interfaz en la Estación Windows. La aplicación que se conecta con el DCS utilizando sockets y sirve de puente entre Excel y la interface del DCS fue construída en lenguaje C++ y compilada con el C++ Builder 5.0 de la empresa Borland. Esta aplicación recibe/envía información de/hacia el DCS utilizando el protocolo TCP/IP, y recibe/envía información de/hacia Excel utilizando DDE, Dynamic Data Exchange un protocolo de intercambio de datos en aplicaciones bajo la plataforma Windows. Los módulos principales que componen la aplicación son los siguientes:

Rutinas de inicialización:

TformMain:	Carga el archivo con los límites de alarmas definidos para cada variable.

Rutinas de comunicación:

ButtonOpenExcelClick:	Conexión a la hoja de cálculo.
TimerDDETimer	Luego de verificar el status de comunicación con el Modelo, puede ejecutar dos tareas: leer las celdas de la hoja de cálculo donde se actualizan los valores leídos del modelo de Hysys, y guardarlos en variables Globales para luego ser enviadas a la rutina en el DCS; ó verificar el controlador que ha sido modificado en el DCS, y con base en este dato, actualizar las celdas de la hoja de Cálculo que deben escribir hacia el modelo de Hysys.

TimerTCPIPTimer	Crea de forma periódica un "thread" (hilo de ejecución), que se encarga de procesar las tareas de comunicación con el DCS.
ThreadTCPIP::Execute	Verifica si la aplicación ya está conectada con la interfaz del DCS (en cuyo caso invoca a Receive), o si desea conectarse (llamando a Connect) ó desconectarse (con Disconnect).
ThreadTCPIP::Connect : ThreadTCPIP::Receive ThreadTCPIP::Disconnect	Estas rutinas son invocadas desde el hilo de ejecución creado, y su función es la de tramitar el protocolo de envío y recepción de datos con la interfaz del DCS.

En la rutina de comunicación con Excel (TimerDDETimer) es necesario verificar primero que la hoja de cálculo haya podido realizar la lectura sobre los datos del modelo. Esto se hace a través de la lectura de una celda de la hoja de cálculo, en la cual se actualiza el status de la aplicación de Excel. Si las celdas de la hoja de cálculo han sido satisfactoriamente actualizadas con los últimos datos del modelo de Hysys, entonces dicho status será “READ_OK” y se podrá proceder a leer los datos de las celdas de Excel para posteriormente enviarlos al DCS:

Algo interesante de notar es que mientras esta aplicación está leyendo las celdas actualizadas con los valores del modelo, la aplicación de Excel (como ya se introdujo anteriormente en el parrafo correspondiente al funcionamiento de todas las aplicaciones) se encuentra escribiendo en el modelo de Hysys el contenido de las casillas que reflejan los datos modificados en las consolas de

operación por el operador. De esta forma se evita la colisión de operaciones de lectura/escritura simultáneas sobre las mismas celdas de la hoja de cálculo.

Igualmente debe notarse que la información relacionada con la pantalla activa en las consolas del DCS se tiene presente para solamente actualizar los datos relacionados con la misma y así evitar la saturación del canal de comunicaciones.

En la hoja de cálculo se implementaron cinco (5) pantallas, correspondientes a las carátulas de control de procesos de la planta.

Ahora bien, en el caso de que el status de comunicación de la hoja de cálculo con el modelo de hysys sea “WRITE_OK”, se procede a la escritura en las celdas correspondientes de los valores modificados en las pantallas de operación del DCS (únicamente se actualizan los datos del controlador modificado para evitar procesamiento de información innecesario):

Antes de realizar la escritura en las celdas de la hoja de cálculo, del controlador modificado, se verifican los rangos de la variable a escribir: Rango de la variable controlada en el caso del setpoint, rango de apertura del actuador en el caso de la válvula, o status del controlador.

Una vez se han leído los valores del modelo (o se ha escrito al mismo) se procede a enviar la información a la interfaz corriendo en el DCS. Para esto es necesario activar el temporizador que crea el hilo de ejecución donde se procesan las tareas de comunicaciones:

En el hilo de ejecución que se procesan las tareas de comunicaciones pueden existir tres posibles situaciones: Que la aplicación desee conectarse al DCS por primera vez, que la aplicación desee enviar datos al DCS (y recibir también), y que la aplicación desee desconectarse del DCS.

Las rutinas de conexión y desconexión se encargan de crear y liberar espacio en memoria para los sockets utilizados y sus búferes de datos, así como adelantar las diferentes etapas en el proceso de reconocimiento de ambas estaciones (la estación Windows y el módulo de aplicaciones AP51 en el DCS). En la rutina de recepción de información se recben los datos provenientes del DCS, entre los cuales se cuenta el número de la pantalla activa en la consola de operación, a partir del cual se determina la estructura de los datos enviados desde el DCS, ya que para cada pantalla activa se envían en secuencia los datos de sus controladores asociados (separados por espacios en blanco):

Debe notarse aquí también que a medida que se reciben los datos de setpoint, apertura de válvula y status del controlador del DCS, estos van siendo comparados con los datos actualmente leídos del modelo, y esta comparación es utilizada para determinar si alguno de los controladores ha sufrido cambios debido a la acción del operador. Igualmente, a partir del número de pantalla activa se envía hacia el DCS los datos correspondientes a los últimos datos actualizados provenientes del modelo.

El número de interfaz cero (0) indica ninguna pantalla activa en el DCS, lo cual genera que se reenvíen los valores de inicialización hacia el DCS de todas las pantallas, conteniendo los valores de setpoint, aperturas de válvula y status de los controladores leídos del modelo.

3.3.3. Hoja de Cálculo en Excel. Como ya se indicó con anterioridad, es finalmente Excel la aplicación que interactúa con el modelo riguroso de la planta construido en el software de simulación Hysys. En la hoja de cálculo se han representado las cinco (5) interfaces de operación previamente explicadas: Carátulas de control de los procesos de la planta. Oculta a la vista del usuario, se encuentra detrás de la hoja de cálculo, la aplicación desarrollada en Visual Basic for Applications (VBA) que se encarga de iniciar la aplicación Hysys, cargar el modelo en memoria, acceder a las diferentes variables del mismo utilizando COM (Component Object Model, un estándar que permite la ejecución de aplicaciones en modo esclavo y la manipulación de los objetos en memoria), y finalmente intercambiar datos entre el DCS y el modelo.

La gran ventaja de haber utilizando VBA para el desarrollo de la integración con Hysys es precisamente la transparencia con la que COM permite acceder a Hysys desde Excel.

4. ARQUITECTURA DE LA HERRAMIENTA "SISTEMA DE ENTRENAMIENTO DE OPERADORES" BASADO EN TECNOLOGÍA DE SISTEMAS EMULADOS

4.1. INTRODUCCIÓN A LA ARQUITECTURA.

El desarrollo de la herramienta Sistema de Entrenamiento de Operadores se llevó a cabo en el lenguaje de programación Visual Basic 6.0, debido a la estrecha vinculación que tiene con el simulador de procesos HYSYS, donde se desarrollan los modelos dinámicos de las plantas que van a representar a la planta real.

La herramienta basada en la tecnología de sistemas emulados desarrollada en un entorno Windows emula fielmente la interfaz de operación de los sistemas de control distribuido para este caso el D.C.S. FOXBORO, presentándoles a los clientes de la herramienta todas las interfaces de procesos, de faceplates de controladores, de históricos y tendencias, de la misma forma que lo haría el I/A DE FOXBORO, dado que es fundamental para los operadores que van a ser entrenados sentirse como en la planta real con la finalidad de generar confianza y seguridad a la hora de aplicar los conocimientos aprendidos en las lecciones de entrenamiento sobre la herramienta en los procesos de la planta real.

La arquitectura de la herramienta está fundamentada en una aplicación cliente /servidor, centralizada en una Estación Servidora encargada de la carga y ejecución de la planta virtual y el procesamiento de datos durante el entrenamiento mediante la estructuración y manejo de la base de datos sobre

esta estación. Además, permite el entrenamiento desde cualquier punto conectado a una red privada.

La arquitectura para el desarrollo de la herramienta consiste entonces en tres aplicaciones, capaces de comunicarse entre sí e interactuar en tiempo real mediante el protocolo de comunicación TCP/IP, estas aplicaciones representan cada una de las interfaces requeridas para el sistema de entrenamiento: Estación Servidora (modelo de simulación y registro de entrenamiento), Estación Instructora (Ingeniero instructor) y la Estación Operadora (Operador en entrenamiento).

Las tres aplicaciones de la que se conforman las herramientas se comunican entre si a través de sockets de la capa de transporte, bajo los protocolos TCP/IP, utilizando la seguridad que nos brinda dicho protocolo para la comunicación de los datos, las aplicaciones son capaces de comunicarse desde cualquier punto de la red, los sockets se programaron tomando como referencia una dirección IP fija que corresponde a la dirección de la estación servidora, asignándole un número de puerto a cada socket abierto sobre la estación servidora, configurándose como sockets servidores, los cuales siempre están en escucha esperando la petición de conexión de la Estación del Instructor y de las Estaciones del Operador.

La herramienta fue diseñada para la conexión de Una Estación Servidora, Una Estación Instructora y hasta cinco (5) Estaciones Operadoras a la vez.

Se realizaron pruebas con la totalidad de las estaciones conectadas y se observó que mejoró el tiempo de respuesta de interacción con los clientes

comparada con la herramienta basada en tecnología de conexión directa, pero se observaron algunos inconvenientes con colisiones de datos entre aplicaciones debido a que el lenguaje de programación utilizado para esta herramienta no posee programación por hebras o con procesamiento en paralelo y esto hace que se pierdan algunos paquetes mientras la aplicación principal está procesando un hilo.

La arquitectura y la topología de la herramienta pueden observarse en forma de diagramas de bloque a través de las siguientes figuras:

Figura 17. Arquitectura de la Herramienta "Sistema de Entrenamiento de Operadores" basado en Tecnología de Sistemas Emulados.

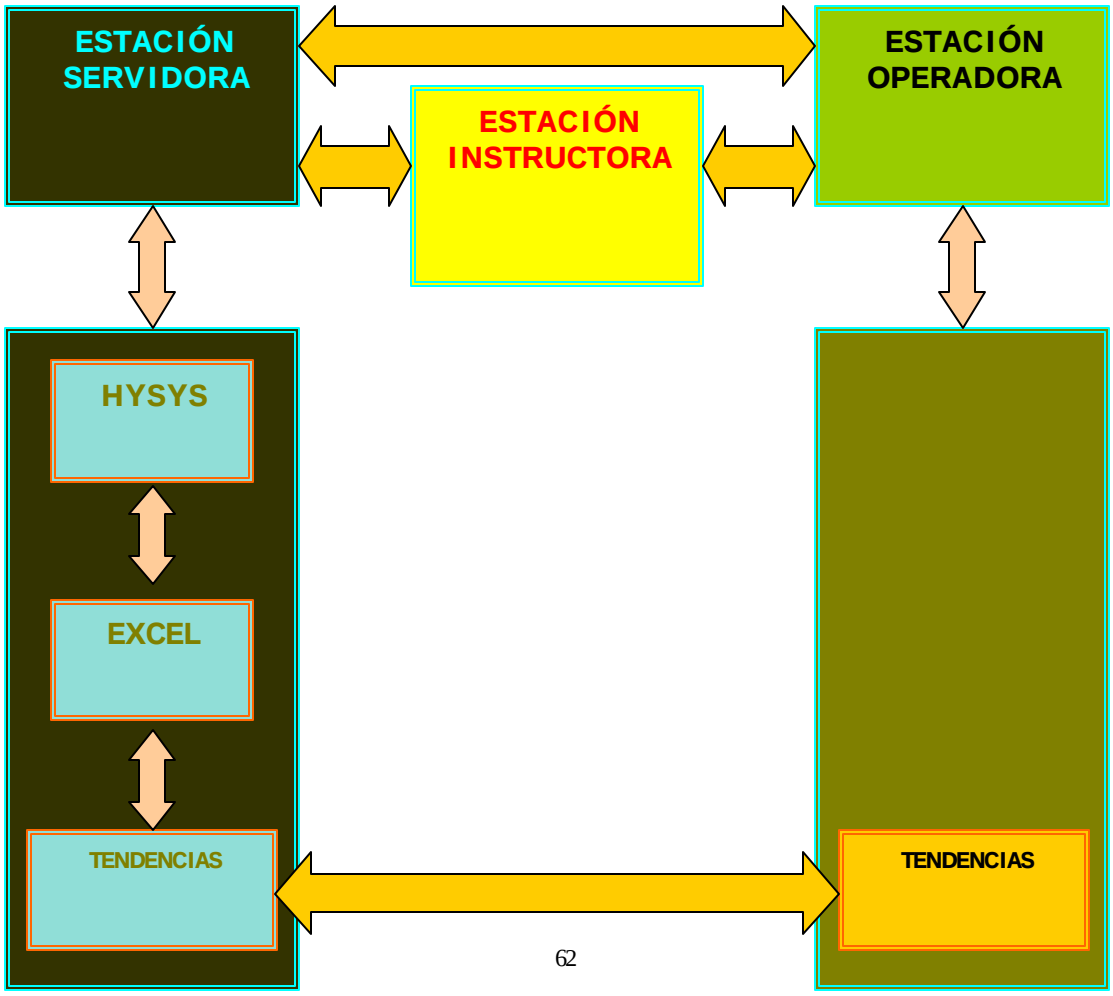
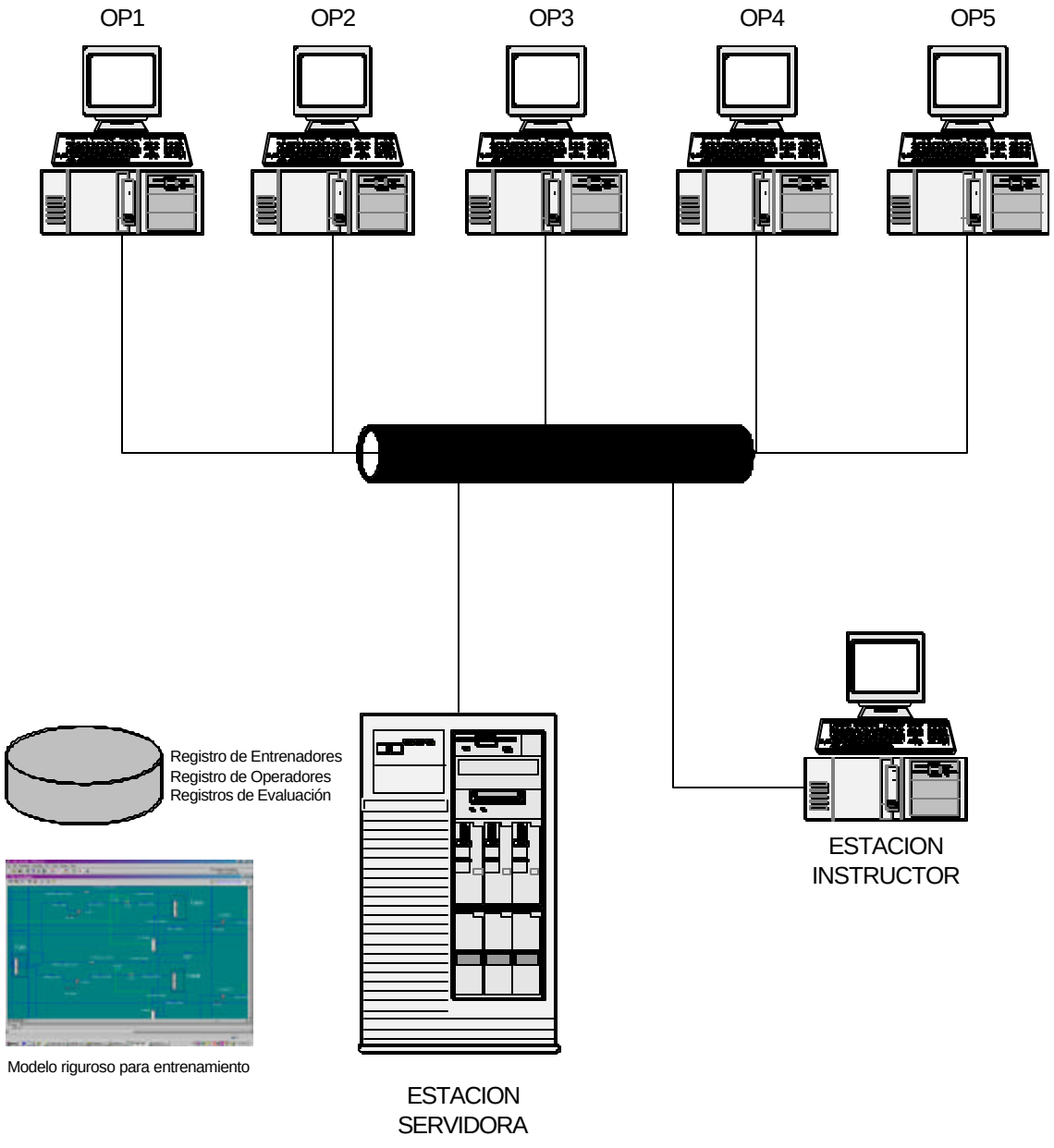


Figura 18. Topología de la Herramienta “Sistema de Entrenamiento de Operadores” basado en Tecnología de Sistemas Emulados.



4.2. ESTACIÓN DEL OPERADOR

La Estación del Operador posee entre sus funciones el registro del operador y habilitar las conexiones mediante sockets clientes conectados con sockets servidores de la Estación Servidora, para realizar el intercambio de datos tanto de escritura generados por el Operador que está siendo entrenado como los de lectura capturados del modelo hysys que simula la planta real por la Estación Servidora (datos que son enviados hacia la estación de operación los cuales permiten la carga de los formularios emuladores de la interfaz del DCS).

En las figuras siguientes es posible observar algunas de las interfaces del DCS I/A FOXBORO ya implementadas en la Estación Operador y que emula la consola real del DCS de la planta:

Figura 19. Interfaces con diagrama de operación del D-201.

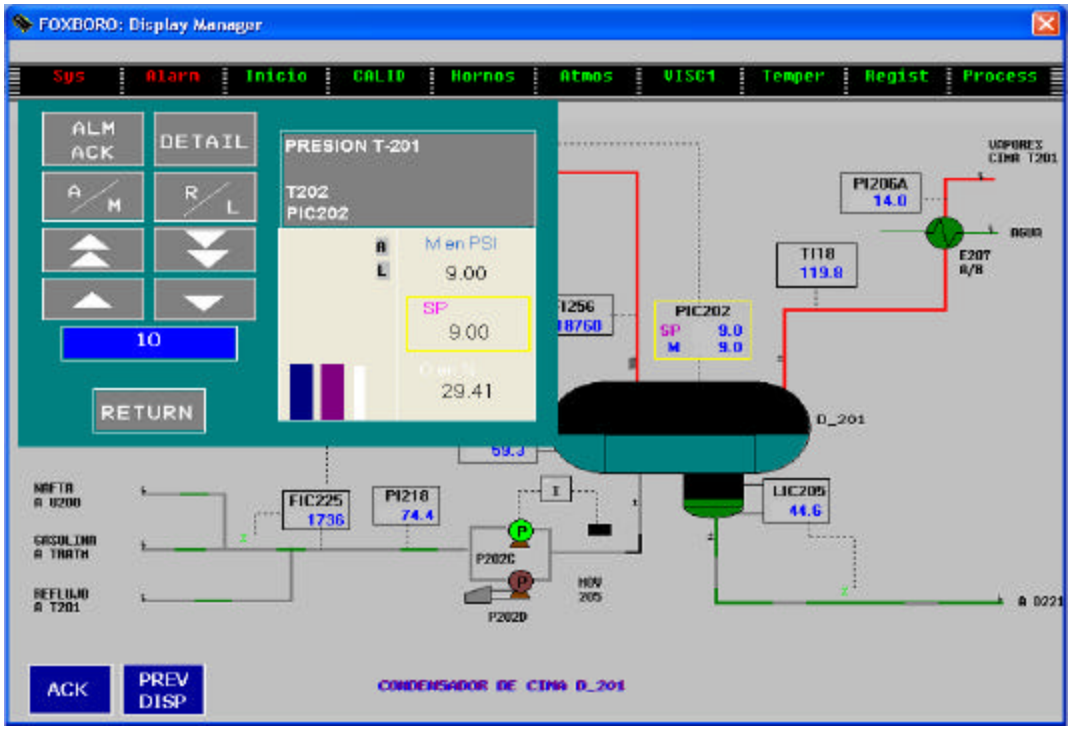
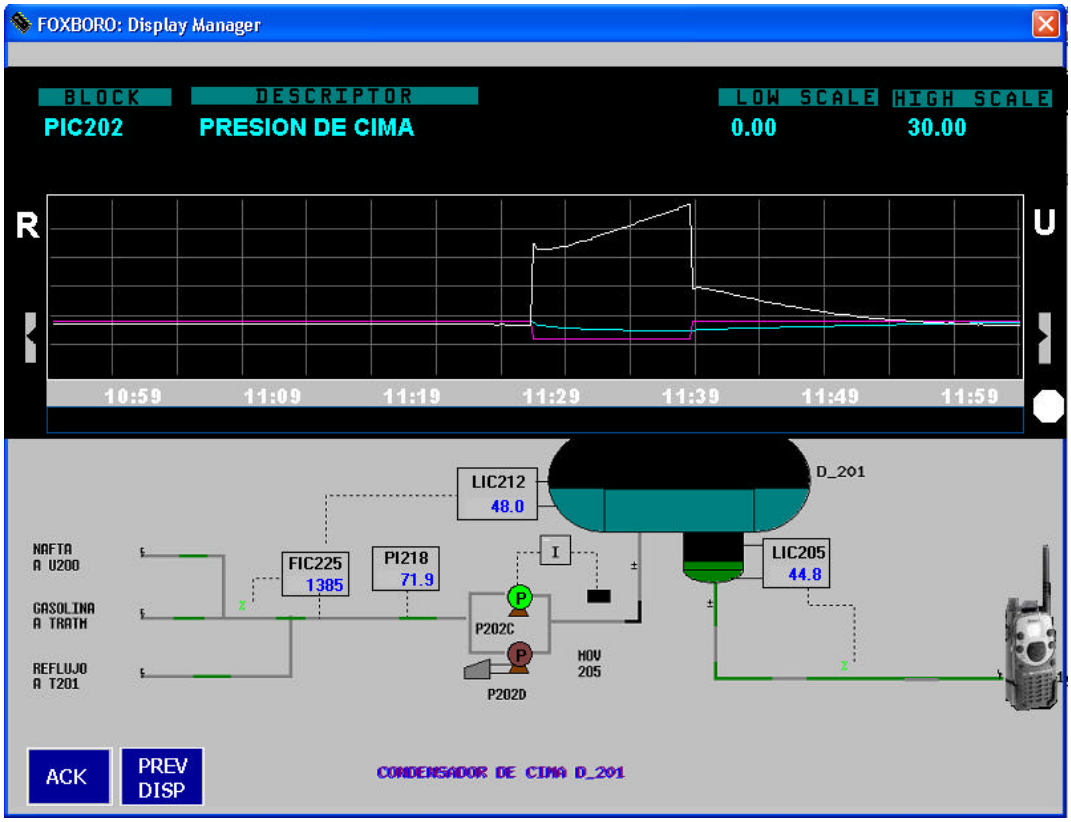


Figura 20. Interfaces con diagrama de operación del D-201 y tendencias.



4.3. ESTACIÓN SERVIDORA.

La Estación Servidora, cuyas funciones ejecutan las tareas de simulación desarrolladas en Hysys, administra la inscripción del personal de entrenamiento, registra los eventos y estados del proceso de entrenamiento y ejecuta el algoritmo de calificación, permitiendo que las estaciones de Instructor y de operario utilicen todos sus recursos en el procesamiento de los procedimientos realizados por estas y el intercambio de datos entre ellas y con la estación servidora, además concibe la posibilidad de llevar a cabo el

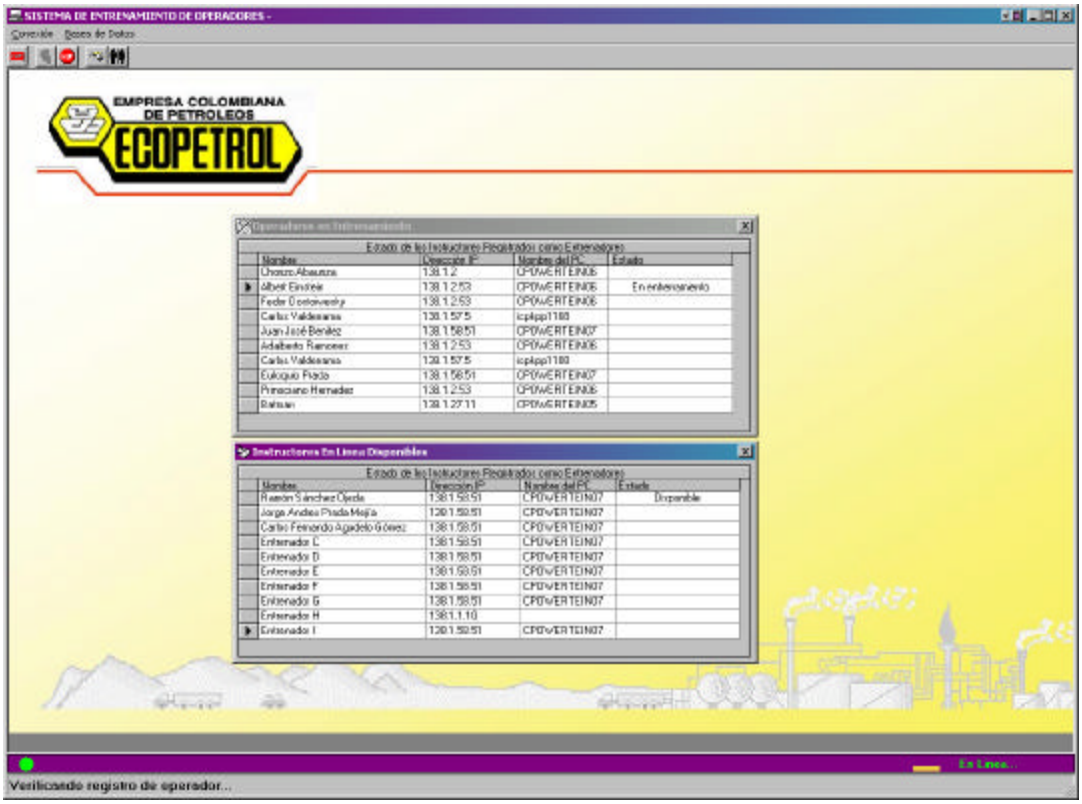
entrenamiento desde cualquier punto conectado a la red, tanto para el instructor como para el operador, otra muy importante característica es la posibilidad de centralizar el manejo de la información, es decir, en ella se encuentra la base de datos de entrenamiento encargada del registro de operarios y entrenadores, la información de desempeño del entrenado, historial de resultados de sesiones de entrenamiento realizadas, además del preestablecimiento y almacenamiento de rutinas de perturbación y de estados objetivos monitoreados para calificación.

La Estación Servidora permite inicialmente el registro de inicio de sesión tanto de los entrenadores como de los operadores, ya que centraliza el manejo de los datos en la base de datos diseñada y construida en Microsoft Access, en ella se registran las direcciones de red (IP), los nombres de las estaciones de trabajo de los entrenadores y operadores autorizados para entrenamiento y los registros de evaluación.

La Estación Servidora controla los accesos y efectúa el llamado a la aplicación Hysys y a los casos establecidos para entrenamiento, así como también el algoritmo de lectura/escritura sobre el modelo de Hysys.

La arquitectura cliente servidor sobre la que esta estructurada el sistema de entrenamiento de operadores, requiere que antes de las peticiones de conexión del instructor y de los operadores esté disponible y abierta la Estación Servidora, por esta razón para dar inicio a la sesión de entrenamiento es necesario iniciar esta aplicación y establecer su estado en "Escuchar".

Figura 21. Listado de instructores y operadores registrados en sesión de entrenamiento.



Para el sistema de comunicaciones entre las estaciones, se estableció el uso del protocolo TCP/IP.

4.4. ESTACIÓN DEL INSTRUCTOR.

La Estación Instructora permite establecer comunicación directa con el modelo y con los operarios en entrenamiento de manera tal que pueda manipular eventos sobre la planta virtual que especifiquen casos de entrenamiento para los operarios. Se estructuró una herramienta de comunicación directa entre el

instructor y el operador a manera de chat, que permite el intercambio de ideas, sugerencias y prescripciones.

El registro, verificación y permisos de conexión a entrenamiento se establecen por lectura directa desde la base de datos que se encuentra sobre la estación servidora.

Otra característica que se ha tenido en cuenta es el registro de hasta cinco operadores en sesión de entrenamiento, un instructor y hasta cinco casos de simulación abiertos dependiendo de sus especificaciones.

Figura 22. Salón de entrenamiento virtual (Estación del Instructor).

4.5. ALGORITMO DE CALIFICACIÓN.

El algoritmo de calificación se basa en el cálculo de la distancia euclídea entre el estado actual del sistema y el estado objetivo propuesto como las especificaciones deseadas para los productos. La calificación del operador se calcula con base en la distancia que existe entre cada estado asociado a los movimientos ejecutados por este, y el estado objetivo al que debe llevar a su planta. Cada movimiento incrementa los puntos de la calificación si su distancia asociada es menor a la distancia obtenida con el movimiento anterior, de esta forma se evalúa el sentido en el que el operador mueve el proceso.

4.6. DETALLES DE IMPLEMENTACIÓN.

A diferencia de la primera propuesta de arquitectura de comunicaciones y tecnología aplicada para la herramienta Sistema de Entrenamiento de Operadores, en esta nueva propuesta no se requiere de la plataforma del DCS. Los computadores utilizados para implementar las estaciones del operador y del instructor, tienen los requerimientos de software y hardware comunes a todo ordenador: Procesador Pentium 4 de 1.6 GHz ó superior, 256 MB de RAM ó superior, disco duro de 40 GB ó superior, sistema operativo Microsoft Windows 2000 con Service Pack 2, Microsoft Excel 97. La estación servidora, por el contrario si requiere unos recursos superiores, debido al cálculo intensivo que debe realizar con la simulación en Hysys, así como las transacciones de información con las otras dos estaciones.

Estas tres aplicaciones fueron desarrolladas en Visual Basic 6.0, y su estructura de módulos se resume a continuación.

4.6.1. Estructura de la Aplicación Servidor.

Módulos:

ServerSub	CondIni()	Toma valores máximo y mínimo de todas las variables para enviar a la aplicación del operador y configurar los rangos de visualización de los indicadores gráficos.
	Escribir()	Tiene el programa principal de escritura sobre el modelo de Hysys para cada una de las 7 interfaces (ajustes de SP, OP, Auto, Manual, Cascada).
	Lectura()	Programa principal de lectura del modelo de Hysys .
	SendPacketOP()	Enviar datos a la aplicación del operador.
	SendPacketIN()	Enviar datos a la aplicación del instructor.
	Escritura	Rutinas de escritura de SP y OP. Llena tabla eventos.
	Leer	Rutinas de lectura de cada una de las interfaces (7). Se invocan desde ServerSub. Construye el bloque de datos para ser enviado a la estación del

operador.

Formularios: Inicio.

Instructores en línea disponibles.

Operadores en entrenamiento.

MDIServer (Tramita el intercambio de información entre aplicaciones usando winsockets).

Historia de las perturbaciones generadas por el instructor.

4.6.2. Estructura de la Aplicación Operador.

Módulos :

Modulo_Principal	DCS()	Es invocada en el FormLoad de cada interfaz. Verifica si hay o no conexión y envía al servidor el número de interfaz a actualizar.
	FirtRun()	Habilita "Conectarse a Entrenamiento" (abre el caso Especificado por el operador).
	Empaquetar()	Llama a Mostrar() que a su vez invoca las rutinas que actualizan cada interfaz con el contenido del vector Valores. Igualmente invoca a MostrarEntrenador() si el instructor ha solicitado espiar interfaces del operador, que envía un paquete de datos hacia la aplicación del instructor.

	EmpCondIni()	Solo se invoca la primera vez y almacena en los vectores de rangos de los indicadores gráficos los valores recibidos de la aplicación servidor.
	IntermXX()	Hacen parpadear los botones Sys y Alarm, así como las alarmas de cada controlador. Es invocada desde el temporizador de cada formulario.
ModuloEntrenador	InterfazEntXX()	Envía los datos de cada interfaz a la estación del instructor. Los datos ya se envían escalados para evitar cálculos en la estación del instructor.
	MostrarEntrenad()	Es invocado desde Empaquetar(), y llama a la rutina InterfazEntXX() apropiada. Utiliza FaceSock para comunicarse con el instructor.
Interfaz1_7	InterfazX()	Guarda los valores de PV, SP, OP. El status de alarmas por controlador:BAJO-BAJO, BAJO, ALTO, ALTO-ALTO.

Formularios :

Ventana Principal Aplicación		Botón Conectar, Desconectar, Terminar, Nombre de la máquina servidora.
	FaceSock	Winsock – Conexión a la máquina instru ctora
	Eslabon	Winsock – Solicitud de conexión del operador al instructor, chat instructor-operador, conexión a la máquina instructora, perturbaciones.
	Server	Winsock – Conexión al servidor.
	Timer	Recoger y expandir las interfaces de chat.
Interfaces de Operación		Interfaces de Operac

4.6.3. Estructura de la Aplicación Instructor.

Módulos :

ModuloEspia	Se encarga de la actualización de las interfaces configuradas en la estación del instructor. Estas son copia de las de operación y permiten hacer seguimiento de las

	acciones del operador. Los datos son enviados desde el ModuloEntrenador de la aplicación en la estación del operador.
Formularios :	
Ventana Principal de Aplicación.	Contiene los tres (3) objetos Winsock que se encargan de tramitar el canal de comunicaciones con la estación servidora y con la estación del operador: Cadena, Server y FaceSock. Cadena recibe Peticiones de entrenamiento de operadores y mensajes de chat. Server recibe los datos de los instructores registrados desde la Estación servidora, muestra perturbaciones creadas y registra eventos de perturbación. FaceSock recibe los datos de las interfaces que son "espiadas" de la estación del operador.
Registro	Código de conexión al servidor.
EventGen2	Crear nuevas perturbaciones. Es el mismo formulario para enviar Perturbaciones al modelo.
Operador	Salón de entrenamiento en el que se pueden monitorear hasta 5 Operadores, enviándoles perturbaciones, espiando interfaces, ó enviando mensajes por chat.

5. COMPARACIÓN Y ANÁLISIS DE LAS TECNOLOGÍAS DE CONEXIÓN DIRECTA Y SISTEMAS EMULADOS BASADAS EN EL SISTEMA DE OPERACIÓN O LA ARQUITECTURA HARDWARE.

En este capítulo se realiza una comparación y análisis de dichas tecnologías haciendo énfasis en : los equipos de cómputo utilizados, las comunicaciones entre los equipos, la comunicación entre las aplicaciones cliente/servidor, la tecnología y arquitectura adecuada para el diseño y la implementación de un Sistema de Entrenamiento de Operadores (OTS) para plantas industriales.

5.1 SISTEMAS MAINFRAMES O GRANDES COMPUTADORES VS. SISTEMAS BASADOS EN PC'S DE ESCRITORIO.

Tradicionalmente, los sistemas de entrenamiento de operadores fueron diseñados para correr sobre Mainframe o computadores como DEC ALPHA usando sistemas operativos como UNIX/VMS, estos poderosos computadores fueron usados ya que corrían todo el sistema de entrenamiento en una sola máquina, y suplían las necesidades de cómputo, cálculo y actualización de los datos que requieren estos sistemas debido a la necesidad de estos de hacer cada vez más real el ambiente de simulación donde se entrenan los operadores, muchas ecuaciones algebraicas y diferenciales necesitan ser resueltas entre 250-500ms, además que los datos tienen que ser actualizados en diferentes formatos como los mostrados al operador entre los que se tienen como mímicos de la planta, tendencias, faceplates de controladores, faceplates de alarmas, diagramas de procesos.

La necesidad de una máquina con tales características restringe el potencial de los sistemas de entrenamiento de operadores haciéndolos muy costosos y la

administración de la herramienta una tarea especializada y solo para unos pocos.

Los avances de la tecnología de los computadores y su alto rendimiento actual, están eliminando estas barreras, justamente pocas décadas atrás los PC's (Desktop) no eran tan poderosos y nunca fueron considerados para correr aplicaciones en tiempo real o aceleradas como los requieren los OTS. Pero hoy en día los PC's se han fortalecido y han mejorado su capacidad hasta el punto de ser capaz de alcanzar niveles de procesamiento de los Mainframes antiguos, especiales para cálculos numéricos y soluciones de todo tipo de ecuaciones.

Con el advenimiento de estos PC's con características tan poderosas, los Sistemas de Entrenamiento de Operadores pueden desarrollarse sobre plataformas de dichos PC's y explotar el potencial de esta herramienta a muy bajo costo. Algunos de estos Sistemas de Entrenamiento de Operadores pueden mantenerse funcionales y sin problemas de procesamiento y velocidad de actualización de datos durante unos pocos años sin que haya necesidad de cambiar los requerimientos de hardware y software, y además se acabaría la preocupación sobre el sistema operativo a seleccionar para el Sistema de Entrenamiento de Operadores por que fácilmente podría utilizar la versatilidad y la amigabilidad que le ofrece Windows, haciendo estos sistemas de fácil mantenimiento y de fácil actualización.

El desarrollo del Sistema de Entrenamiento de Operadores bajo la tecnología de Sistemas Emulados el cual fue expuesto en paginas anteriores, aprovecha de forma completa las bondades de los sistemas basados en PC`s de escritorio, dando resultados satisfactorios culminando con una herramienta muy

versátil, de bajo costo comparado con las otras tecnologías y con mayor facilidad para la personalización de las interfaces de operación y características del modelo de simulación, ya que son aplicaciones estándar bajo un sistema operativo muy amigable como lo es Windows.

5.2. SISTEMAS DE CONEXIÓN DIRECTA VS. SISTEMAS EMULADOS EN PC'S DE ESCRITORIO.

Actualmente en la industria pocos "Sistemas de Entrenamiento de Operadores" con tecnología de Conexión Directa están siendo usados. Tecnología de Conexión Directa, como su nombre lo dice, irónicamente el sistema de entrenamiento no está conectado directamente con la planta real, solo se encuentra conectado a la estación de aplicaciones del Sistema de Control Distribuido (D.C.S.) real.

Los Sistemas de Entrenamiento de Operadores que están basados en tecnología de conexión directa, son como el expuesto en esta monografía el cual posee un modelo de simulación de la planta real que se encuentra conectado mediante interfaces de comunicación al D.C.S. I/A DE FOXBORO, en donde fueron diseñados previamente los faceplates de controladores y diagramas de procesos de la planta real para luego ser direccionados a bloques dependientes que actúan como buffer de transmisión de datos entre el modelo y el operador que manipula el DCS, esta replica nuevamente de las interfaces se hace con la finalidad de no interferir con los procesos de la planta real.

Las interfaces de comunicación son aplicaciones desarrolladas sobre distintas plataformas, para el caso expuesto fue necesario realizar interfaces sobre plataformas tipo Windows (la estación donde corre el modelo de simulación) y sobre plataformas tipo UNIX, ya que este es el sistema operativo que posee las estaciones de trabajo del DCS I/A DE FOXBORO, además de las interfaces entre el modelo y la aplicación que corre sobre la estación de simulación.

Como se puede observar el trabajo requerido para desarrollar una herramienta de conexión directa para dichos sistemas es una tarea dispendiosa y poco atractiva , ya que para cada DCS de diferente marca es necesario emprender un nuevo desarrollo implementando nuevamente interfaces de comunicación e interfaces de operación sobre cada DCS; y además que la mano de obra debe ser especializada, lo que implica elevados costos, igualmente para poder correr dichos sistemas es necesario disponer de estaciones de trabajo tipo DCS y utilizar recursos de la infraestructura del DCS tales como tiempo de procesamiento y canal de comunicaciones, lo que hace incrementar de manera exorbitante los costos de dicho sistema.

Un Sistema de Entrenamiento de Operadores basado en aplicaciones que emulen las interfaces de los Sistemas de Control Distribuido tiene ventajas sobre los de Tecnología de Conexión Directa si la emulación de los diagramas de operación es de alta fidelidad, además que su hardware es mucho más reducido, la necesidad de hardware y software del D.C.S. es eliminada y reemplazada por la disponibilidad, el bajo costo del hardware y un software standard para las estaciones de trabajo que emulen los sistemas.

Otras de las grandes ventajas de la Tecnología de Sistemas Emulados sobre plataformas de PC's de escritorio son la flexibilidad, transportabilidad y los

múltiples usos de la emulación, el usuario puede rápidamente y fácilmente hacer copias del sistema que puede ser modificado y adaptado para diferentes plantas y diferentes sistemas de control distribuidos D.C.S.

6. CONCLUSIONES

Las conclusiones están orientadas a la Selección de la Arquitectura y Tecnología Adecuada para el Desarrollo de los Sistemas de Entrenamiento de Operadores en base a los análisis anteriores.

Resulta poco recomendable el uso de las propias instalaciones industriales como lugar de entrenamiento, debido al alto costo logístico o al riesgo de pérdidas materiales e incluso de vidas humanas, por esto el entrenamiento debe llevarse en ambientes simulados dotados de alto grado fidelidad y desempeño.

Las ventajas de los sistemas de Entrenamiento de Operadores basados en Tecnología de Sistemas Emulados como el O.T.S expuesto en este trabajo son:

- El bajo costo ya que no es necesario los Mainframe ni el hardware y arquitectura de un D.C.S. real, como también los bajos costos de soporte y mantenimiento del hardware y el software de estos sistemas emulados.

- El diseño compacto de estos sistemas lo hacen un sistema portátil y flexible

- Permite realizar cambios sobre el modelo, Estación de Operación y Estación del Instructor de una manera más fácil, a bajo costo y en corto tiempo.

- La facilidad de las múltiples sesiones de entrenamiento que pueden ser abiertas y los múltiples modelos de simulación que pueden estar corriendo simultáneamente.

- El sistema de Entrenamiento de Operadores se puede comprar e implementar en un tiempo muy corto, y evitaría la demora que generaría un sistema de conexión directa con la compra de las nuevas estaciones de operación del D.C.S y el diseño por parte del personal especializado de las interfaces de operaciones necesarias para el entrenamiento.

- El Sistema de Entrenamiento de Operadores se puede utilizar sin interferir con el hardware y software de control de la planta real, a diferencia de los sistemas basados en conexión directa que utilizan los recursos de hardware y software del D.C.S. de la planta.

7. BIBLIOGRAFIA

Conceptual Design Document(CDD2) for CAPE OPEN Project, CAPE OPEN PROJECT TEAM, Version 4 January 2000.

Dynamizing a plantwide Steady State Model. Seminario AspenTech. Instituto Colombiano del Petróleo, Bucaramanga, 2003.

Instructor Guidelines for Use of an Operator Training Simulator (OTS). A Compilation of Experiences and Lessons Learned from OTS Users Group Meetings. EPRI, Palo Alto, CA: 2001.

HYPROTECH HYSYS PROCESS, Customization Guide, Version 3.2, August 2003.

Training Simulators: Engineering and Use. Gunter Reinig, et al. Chemical Engineering Technology, 21 (1998).

Improve process training with dynamic simulation. S.W. Morgan, et al. Hydrocarbon Processing, Abril 1994.

Conceptual Design Document(CDD2) for CAPE OPEN Project, CAPE OPEN PROJECT TEAM, Version 4 January 2000.

Borland C++ Builder 5, Developers Guide, Borland Inprise , 1993