

## ENTORNO DE COMUNICACIONES OPC PARA ECOSIMPRO

Jesús María Zamarreño Cosme

Dpto. Ingeniería de Sistemas y Automática (Fac. Ciencias) Universidad de Valladolid, jm@autom.uva.es

### Resumen

*En este artículo se muestra cómo se puede realizar la integración de OPC con una simulación EcosimPro de manera que el resultado final sea un servidor OPC que nos facilita el acceso a las variables de la simulación en tiempo real desde cualquier aplicación que soporte este estándar.*

**Palabras Clave:** OPC, Simulación en tiempo real, Integración, Comunicaciones.

### 1 INTRODUCCIÓN

Cuando se desarrolla una simulación de un proceso en un lenguaje como EcosimPro (o en cualquier otro lenguaje), los objetivos de la simulación pueden ser diversos (diseño, pruebas en diversas situaciones críticas, sintonización de controladores, etc.). Todas estas tareas, en la mayoría de los casos, pueden ser realizadas a través de experimentos programados en el mismo entorno de la herramienta de simulación. Sin embargo, una forma más flexible y abierta sería utilizar herramientas externas específicas que se comuniquen de alguna manera con la simulación. Para poder realizar esto, es necesario dotar a la simulación de posibilidades de comunicación bidireccionales accediendo al código generado en caso de que la propia simulación no soporte ningún tipo de protocolo de comunicación. Una de las ventajas de EcosimPro en este sentido es la posibilidad de utilizar las clases C++ generadas para su integración con el marco de comunicaciones elegido.

Existen diversas alternativas para comunicar el simulador con otras aplicaciones (como HMI, SCADA, controladores, aplicaciones a medida, etc), entre ellas podemos mencionar los ficheros de datos, sockets, DDE, etc, pero en cualquiera de los casos la comunicación obtenida es muy específica y poco generalizable a otras situaciones. Otra posible alternativa es usar OPC (OLE for Process Control) que es un estándar industrial pensado para que las aplicaciones informáticas en el ámbito del control intercambien y compartan datos.

En las siguientes secciones se verán los pasos que se deben realizar para conseguir esta integración de manera que cualquier simulación se convierta en un servidor OPC accesible por cualquier cliente OPC. Comenzaremos dando los conceptos fundamentales en los que se basa el estándar OPC, a continuación repasaremos las posibilidades que nos brindan las clases C++ generadas por EcosimPro, y finalmente describiremos la integración de la simulación en el servidor OPC.

### 2 OPC

OPC significa OLE for Process Control. Está basado en la tecnología OLE/COM/DCOM de Microsoft y es un estándar industrial que provee de un interface común para comunicaciones que permite que componentes software individuales interactúen y compartan datos. La comunicación OPC se realiza en la forma cliente/servidor. El servidor OPC es la fuente de datos (podría ser por ejemplo un dispositivo hardware a nivel de planta aunque en nuestro caso será la simulación) y cualquier aplicación basada en OPC (el cliente) tendrá acceso al servidor para leer/escribir cualquier variable suministrada por este (Figura 1). Así, una de las ventajas de dar capacidad de comunicación OPC a una simulación es que cualquier producto OPC que actúe como cliente puede acceder a las variables de la simulación. Las variables pueden ser consultadas utilizando un explorador y pueden ser seleccionadas de forma sencilla usando sus nombres naturales o sus tags.

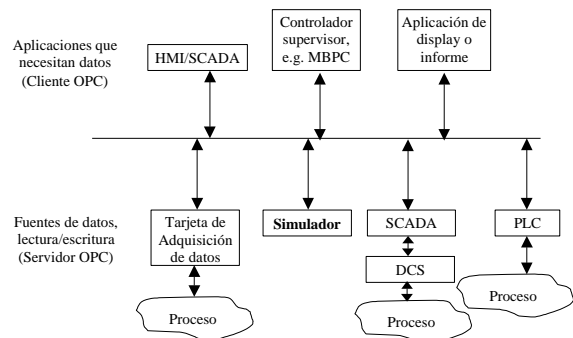


Figura 1: comunicaciones OPC entre elementos

Las variables suministradas por un servidor OPC se estructuran de forma jerárquica. En primer lugar, el servidor OPC puede estar localizado en un nodo distinto del cliente, por lo que la raíz de un servidor OPC se especifica como el nodo y el nombre del servidor. Las variables (o ítems) están incluidas en un espacio de nombres jerárquico a partir de una raíz. Los clientes incorporan estos ítems en grupos, por lo que la especificación de cualquier variable almacenada en un servidor OPC sería:

- ✓ Nodo
- ✓ Nombre\_Servidor
- ✓ Ruta\_por\_espacio\_nombres
- ✓ Ítem (contiene valor, calidad y fecha/hora)

Este espacio de nombres es un elemento importante del servidor OPC ya que permite “navegar” por él para encontrar las variables en las que estamos interesados. En la figura 2 se ilustra un caso concreto de espacio de nombres.

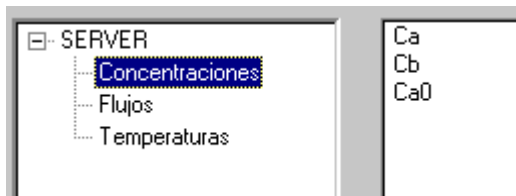


Figura 2: espacio de nombres de un servidor OPC

Una limitación de OPC es que sólo se ejecuta en plataformas Microsoft; se recomienda Windows NT (por otra parte, cada vez más extendido en entornos industriales), aunque puede ser instalado en Windows 95 con extensiones DCOM, Windows 98, ME y ahora en el reciente Windows 2000.

Para programar una aplicación con capacidad OPC existen varias herramientas en el mercado, la mayoría de ellas consisten en una serie de clases C++ que facilitan el desarrollo de la aplicación en lo que hace referencia a la comunicación. Esto, además, nos permite y facilita una perfecta integración con las clases C++ generadas por la simulación EcosimPro.

### 3 GENERACIÓN DE CLASES C++ POR ECOSIMPRO

Las clases C++ generadas por EcosimPro permiten su reutilización para otras necesidades. Las posibilidades que brinda son iguales o superiores a las que permite el concepto de experimento en EcosimPro. Cualquier cosa que se puede hacer en un experimento, se podrá realizar desde un programa en C++.

#### 3.1 CLASES GENERADAS AUTOMÁTICAMENTE POR ECOSIMPRO

Al compilar un experimento, se generan automáticamente dos clases en C++:

1. Una clase que se refiere a la partición. Se genera en dos ficheros con los nombres COMPONENTE.PARTICIÓN y extensiones h y cpp.
2. Una clase que se refiere al experimento. Se genera en dos ficheros con los nombres COMPONENTE.PARTICIÓN.EXPERIMENTO y extensiones h y cpp.

Será necesario utilizar la librería INTEG\_mt.lib suministrada también por EcosimPro, que contiene entre otras cosas los métodos de integración, además de los ficheros ecosim.h y MDL\_common.h.

### 3.2 POSIBILIDADES DESDE C++

#### 3.2.1 Inicialización del experimento

El primer paso es, desde nuestro programa C++, declarar un objeto experimento e iniciarlo:

```
modelo_simul_exp expe;
initEcosim(&expe);
```

A partir de este momento, el objeto expe puede ser utilizado para realizar cualquier función que se permita a un experimento.

#### 3.2.2 Integración del experimento (simulación)

En particular, será interesante realizar una integración del modelo con el objeto de simular la evolución del sistema a lo largo del tiempo. Esto se puede realizar en C++ a través del método INTEG\_CINT() del experimento, que realiza la integración de un intervalo de comunicación definido en el atributo CINT del experimento (supondremos que está dado en horas). Para conseguir que la simulación avance en tiempo real podríamos escribir un código como el siguiente, en el que se debe cumplir que la integración pueda realizarse más rápido que en tiempo real y el tiempo que “sobre” se queda en espera. Con el objetivo de poder acelerar sistemas lentos y que la simulación se realice a un ritmo superior al tiempo real, se introduce un parámetro fac (factor de aceleración).

```
UINT ThreadExpe (LPVOID lparam)
{
    LARGE_INTEGER principio, final, frecuencia;
    double factor;
    // 'factor' es la relación entre el intervalo de
    // comunicación y el tiempo de cálculo
    expe.INTEG_CINT();
    // integración previa

    while(true)
```

```

{
QueryPerformanceCounter(&principio);
if (expe.INTEG_CINT() == INTEG_END)
    break;
QueryPerformanceCounter(&final);

QueryPerformanceFrequency(&frecuencia);
t_calculo = (double) (final.QuadPart -
    principio.QuadPart) / (double)
    frecuencia.QuadPart;

factor = (expe.CINT)*3600.0/t_calculo;
lmaxac = (lmaxac < factor) ? lmaxac : factor;

if (factor>1)
    Sleep((unsigned long)((factor -
        1)*t_calculo*1000/fac));
}
return 1;
}
    
```

### 3.2.3 Acceso a las variables

Otro aspecto interesante es el acceso a las variables de la simulación tanto para leer los valores y poder comunicarlos a otros módulos como poder escribir nuevos valores en caso de que se requiera. Para ello, existe el método `getValueReal` del experimento que lleva como argumento el nombre de la variable EcosimPro y que nos permite leer el valor actual de la simulación; por ejemplo, para leer la variable `Ca` del experimentos escribiríamos:

```
valor = expe.getValueReal("Ca");
```

Para escribir un valor en una variable, se utiliza el método `setValueReal` del experimento que lleva como argumentos el nombre en EcosimPro de la variable y el valor que le queremos asignar:

```
expe.setValueReal("FI",value);
```

Estos son los aspectos más destacados y que nos permiten un acceso fácil y sencillo al modelo EcosimPro desde C++.

## 4 INTEGRACIÓN DE OPC Y ECOSIMPRO

Una vez explicado qué es OPC y la forma básica de utilizar el modelo EcosimPro desde C++, pasemos a ver cómo se integran ambos elementos. Para ello, en primer lugar, dispondríamos de un “marco” tipo de servidor OPC. Este se escribe en C++ y en principio sería un servidor con toda la funcionalidad pero que no contiene todavía las variables de la simulación ni la propia simulación. En dicho marco, se deben incluir las clases C++ generadas por EcosimPro y hacer las llamadas adecuadas en determinados puntos del programa principal. Por ejemplo, cuando se arranca el servidor, habría que declarar el objeto experimento e inicializarlo como se ha indicado anteriormente. Otro aspecto importante sería construir el espacio de nombres de acuerdo con las variables de simulación de que dispongamos. Finalmente, y también igualmente importante es establecer cómo se realizará la lectura/escritura de valores de las diversas variables cuando algún cliente OPC lo solicite. Esto se realiza introduciendo los métodos vistos en la sección 3.2.3 en la rutina que gestiona las peticiones de comunicación por parte de los clientes.

De forma esquemática aparecen en la figura 3 los diversos pasos a seguir en la construcción de este servidor OPC.

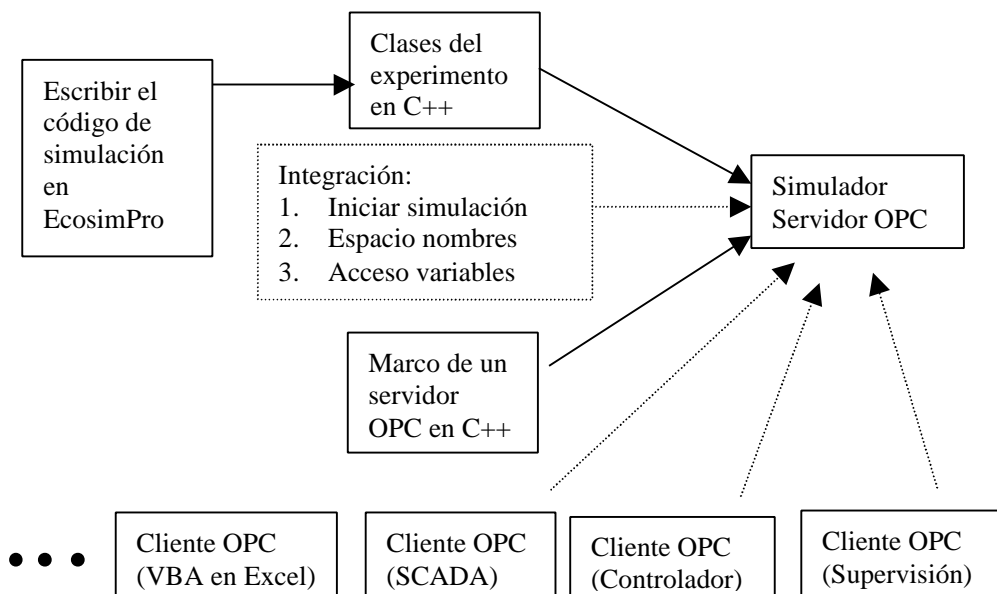


Figura 3: Procedimiento de generación del servidor OPC de la simulación EcosimPro

## 5 APLICABILIDAD INDUSTRIAL

OPC se ha convertido en un estándar *de facto* para tareas de intercambio y acceso a datos dentro del ámbito industrial. Por esta razón, prácticamente todos los SCADAs actuales son capaces de, actuando como clientes, adquirir datos a través de OPC. Como se explica en [1], esto nos permite realizar el diseño y configuración del SCADA de una planta sin necesidad de que dicha planta esté ni siquiera construida, utilizando la simulación-servidor OPC como fuente de datos para realizar tanto el mencionado diseño como pruebas de funcionamiento que pudieran sugerir un cambio en los parámetros de la planta. Una vez comprobado en simulación que todo funciona como se desea, y una vez montada la planta, simplemente hay que reasignar los tags para que estos provengan, por ejemplo, de una tarjeta de adquisición de datos que disponga de servidor OPC.

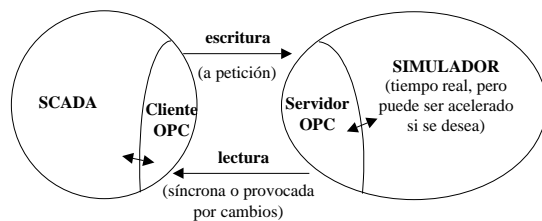


Figura 4: Enlace OPC entre un SCADA y la simulación

## 6 CONCLUSIONES

En este artículo se ha mostrado cómo se puede convertir un modelo cualquiera de EcosimPro en un servidor OPC que contiene la simulación ejecutándose en tiempo real. De esta forma, la simulación queda totalmente abierta a comunicaciones por parte de terceros programas de una forma estándar y flexible. Si se dispone de un marco tipo con el servidor OPC, la incorporación del modelo EcosimPro en la arquitectura es cuestión de horas siguiendo un procedimiento totalmente sistemático.

### Agradecimientos

El autor agradece a la CICYT el soporte obtenido a través del proyecto FEDER “Supervisión y Control Optimizado de Procesos (TAP 1FD97-1690)”.

### Referencias

[1] Acebes, L.F., Zamarreño J.M. (2001) “Diseño de un SCADA para una planta piloto usando

un simulador OPC”, *Automática e Instrumentación*, 316, pp. 76-80.

[2] Opc Task Force, (1998) “OPC Overview”, *OPC Foundation*.

[3] Softing, (1999) “OPC Server Toolkit Programming Guide”.

[4] Zamarreño, J.M., Acebes, L.F., Alvés, R. (2000) “OPC-based real time simulator: architecture and practical example”, *41<sup>st</sup> SIMS Simulation Conference*.