20th European Symposium on Computer Aided Process Engineering – ESCAPE20
S. Pierucci and G. Buzzi Ferraris (Editors)

Process simulation as a domain-specific OPC Unified Architecture information model

Paolo Greppi

Università di Genova Dipartimento DICAT, Via Opera Pia 15 16145 Genova, ITALY, paolo.greppi@unige.it

Abstract

Currently one of the key challenges for the process industries is to react quickly to market changes; also the process automation and information technology infrastructure should follow seamlessly these changes without requiring frequent reconfiguration or manual adaptation of custom interfaces.
This driver for flexibility also applies to the model-based applications in use in the chemical, petrochemical and polymer industries: soft-sensors, inferentials, reconciliation, model predictive control and operator training systems. It is widely recognized that a Service-Oriented Architecture can match these requirements; in particular the recently released OPC Unified Architecture (OPC UA) specification provides a service-oriented architecture for interoperability in industrial automation.
In this work the prototype implementation of a simple process model as an OPC UA Server is presented; the model is a first-principle soft sensor that calculates a few useful physical properties from the composition of a process stream measured for example by a process gas-chromatograph.
The prototype demonstrates that an OPC UA soft sensor can be self contained, performant, transparent and easy to configure. Furthermore OPC UA is a very general technology that can be used to interface process models with other software components and implement distributed simulation systems; in this sense it challenges the existing interoperability standard CAPE-OPEN.

Keywords: Object-oriented, service oriented architecture, soft sensors, on-line

1. Introduction

According to a recent review [1] the current challenge in the process industry business environment is to react quicker and quicker to the market dynamics and to smoothly adapt the business and technological processes to these changes; but the current information technology infrastructure and in particular industrial process automation is not exactly flexible.
Service-Oriented Architecture (SOA) can bridge the gap, allowing the industry to use existing IT investments and achieve the flexibility required. With regards to the process industry, the recently released OPC Unified Architecture (OPC UA) specification [2] builds on the successes of the established Classic OPC standard [3] for interoperability between enterprise information systems and industrial automation. It provides platform-independence, the capability to expose the semantics of the data and a service-oriented architecture (SOA).
If a standard domain-specific information model based on OPC UA exists for an application, data and functionalities can be made available as services in a vendor-independent fashion; work is under way [4] to standardize information models for

physical device information, analyser devices, plant operation and maintenance, batch control, PLC programming.
Of course also process simulation capabilities can be encapsulated as OPC UA services; this can be done for simple models (such as a material stream or a unit operation) as well as for more comprehensive models (for example model predictive control or plant-wide mass balance reconciliation). What is more, not only the mere calculation of the model can take place through the OPC UA interface, but also discovery of services and configuration of the models.

In this work an existing soft sensor interfacing with the process via Classic OPC was upgraded to OPC UA; to our knowledge this is the first working prototype of an OPC UA server for process simulation.
In the following the modelling capabilities of the soft sensor and the approach used for the implementation are described, and the significance of this technology is commented.

2. The soft sensor

The simple process model which has been wrapped as an OPC UA Server in our prototype is a first-principle soft sensor that calculates the following useful properties for a material stream composed of short-chain hydrocarbons:

- Lower / higher heating value and Wobbe-Index;
- Explodibility-related quantities such as LEL / UEL (lower and upper explosive limits) and LOC (limiting oxygen content);
- Density, compression factor and temperature / pressure dew-point and bubble-point with an equation of state.

The typical application of this soft sensor is to provide inferential measurements when a process gas-chromatograph or other process analytic device is available and measures the composition of a process stream at regular intervals, such as in natural gas processing, transport and distribution, polymers and petrochemicals production.
This kind of application in the natural gas sector is well-known and regulated by international standards [5] [6], with an ongoing evolution towards the application of more sophisticated equations of state [7]. In the polymers and petrochemical sectors the interest is more in traditional cubic or statistical equations of state [8].

3. OPC UA server implementation

A key element of OPC UA is the capability to expose the semantics of the data: a temperature is different from a pressure; a variable reserved for the storage of a temperature measured in the field is conceptually different from a variable used to store a calculated temperature; the temperature, pressure and composition for a given process fluid belong together; the capability to compute a model is common to all different models.
To express these relationships, Variable and Objects types can be defined extending the built-in OPC UA types. The transposition of the above examples in our implementation are: we included an EngineeringUnit property to the base variable type to distinguish different physical quantities; we use the AccessLevel field to make a calculated variable read-only; temperature, pressure and composition for a process fluid are grouped in a "stream" abstract object type; all concrete stream models are subtyped from an abstract modelBase object which has pure virtual methods `Calculate`, `Reset` and `ResetErrors`.

For the implementation of the OPC UA server the first and currently only commercially available high-level C++ software development kit (SDK) [9] has been used. The thermodynamic calculations and the empirical correlations are implemented using LIBPF (LIBrary for Process Flowsheeting), a process flowsheeting and modelling tool arranged as a C++ library [10].
The object-oriented design and portability of the LIBPF library fit with the philosophy of OPC UA. This reduced the development time and resulted in an extremely lean implementation: coding the LIBPF / OPC UA bridge required a mere 5000 lines of C++ code. Key used LIBPF functionalities were:

- units of measurements: all quantities in LIBPF carry a unit field which was used to populate the EngineeringUnit property of the corresponding OPC UA variables
- reflection: the ability to programmatically inspect and use types at run-time)
- dynamic type manipulation via the object factory without reference to Windows-specific object broker mechanisms such as COM or .NET
- object persistency.

Both the OPC UA SDK as well as LIBPF library support Windows and Linux, so the prototype server was successfully tested in both environments. If desired the soft sensor could even be integrated directly into the automation device, whatever real-time operating system it is running, thanks to the portability of both libraries.

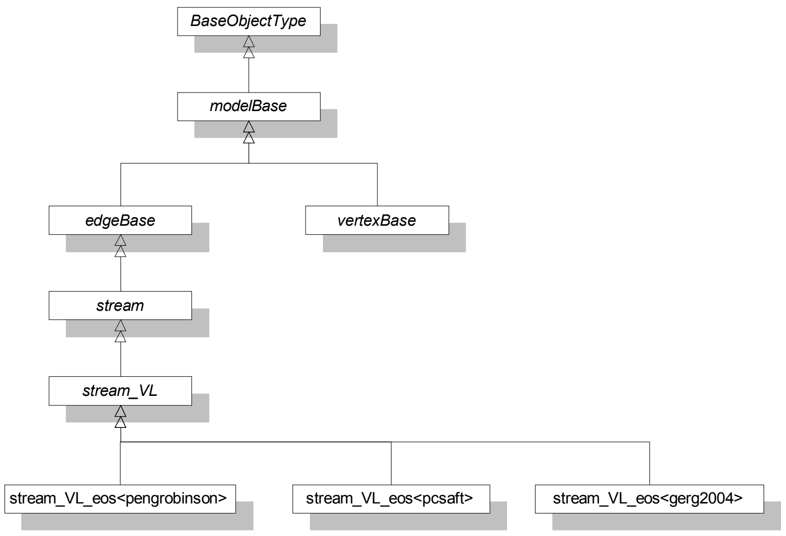

Figure 1 - OPC UA types from the implemented process simulation information model

An exemplificative subset of the implemented type hierarchy, expressed in the OPC UA (UML-derived) graphical notation is shown in figure 1.
Shaded boxes indicate object types; object type names in italic indicate that the respective objects are abstract i.e. they express an incomplete set of characteristics, and can not therefore be instantiated but only contribute to the definition of a more complex type. The double-arrows connection indicate the types are in a HasSubtype relation.

The instantiated objects exist in some form in the memory of the OPC UA server, as "live", instantiated objects of certain types. OPC UA clients can browse and access the objects, and execute methods on them. Either the server has a fixed set of live object instances, determined at compile-time, or it can expose a node creation service (note: this functionality is not yet implemented in our prototype). In this latter case clients can browse the hierarchy of available types known to the server, and request that one of them should be instantiated at run-time; this scenario makes sense for a simulation information model, where the types match virtual, non-physical entities, or during sensor/controller/actuator configuration for an hardware-related information model.

If the server has to serve node creation service requests, and the clients require that server-side objects "survive" a restart of the server process, the server needs a facility to persist the live instances. For this purpose LIBPF offers a built-in persistency mechanism to an external relational database.
The persistency mechanism is also useful if the server exposes a large number of live instances, which are infrequently accessed: with persistency unused objects can be cached to disk reducing the consumption of resources.

3.1. Comparison with Classic OPC

A similar soft sensor had been previously implemented with a Classic OPC DA interface. The main difference with the new OPC UA server prototype is that the old-style soft sensor was an OPC client: the OPC tags required to store the results of the calculations had to be manually created on the Classic OPC server of the DCS; next the tag names had to be manually configured with the soft sensor using an XML file. Any change to the OPC server configuration would require a manual update of the client configuration; this is where the limited flexibility of a non-service oriented architecture becomes evident.
Another difference is that the Classic OPC client contained a bare-bone state machine to schedule the repeated execution of the calculations, but this state machine was opaque and again only configurable with a custom interface (in that case, values stored in the operating system registry). The orthodox way to achieve this in OPC UA (although not yet implemented in our prototype) would be to use the State Machine information model and the Program abstraction provided by the specification.

In the resulting OPC UA soft sensor there would be then no need for configuration files, registry settings or a maintenance user interface: the configuration could be performed using any OPC UA client, requiring no client-side configuration thanks to the discover capabilities.
The preferred use case would be to have a higher level OPC UA server, i.e. the one provided by the DCS vendor, controlling the soft sensor according to the “chained server” pattern. The elected OPC UA client to perform the set-up in this case would be the generic control system maintenance user interface itself.

3.2. Off-line application

Once the process simulation domain is exposed as an OPC UA information model, and objects such as material streams, unit operations and flow-sheets can be instantiated, the models could be invoked also for off-line use, without any interaction with plant data.
OPC UA would then be used as a communication protocol between a calculation kernel and any client, be it a standalone user interface or a web server or a light client on a mobile device. Although the OPC UA specification has been created for distributing applications, and designed for efficient communication when the client and the server

sit on different machines, the architecture can be simplified with the two coexisting on the same workstation.

3.3. Comparison with CAPE-OPEN

It is interesting to compare the OPC UA off-line application scenario above with CAPE-OPEN [11], the de-facto standard for interoperability of CAPE (Computer-Aided Process Engineering) applications or components based on COM middle-ware. The supported operating systems, architecture and network layouts, configuration requirements and performance issues are reviewed for both technologies.

The equivalent to the CAPE-OPEN simulation executive would be a specially designed OPC UA client. This client would act as host for complete flow-sheets, unit operation or thermo models that would be discovered, browsed and controlled on either local or remote servers using OPC UA service oriented architecture. Any complex model could in turn be a host for a thermo model pulled from another server, using the chained OPC UA server architecture.

CAPE-OPEN is based on COM middle-ware and therefore bound to the Windows platform, whereas OPC UA is not tied to a specific operating systems, which is very desirable both for low-end (real-time, embedded systems) as well as for high-end (servers) environments.

CAPE-OPEN can connect to remote objects using the distributed COM (DCOM) technology and its proprietary protocol, but not across a WLAN since a transparent http-based SOA protocol is not supported. OPC UA on the other hand can operate as a web service using SOAP/HTTP transport protocol, making it possible to access services across a firewall.

CAPE-OPEN requires that all object types (unit operation, reaction or thermo) are assigned a Class ID and registered with the operating system as COM objects - the instantiation of the objects is performed by calling the operating system CreateObject function. OPC UA on the other hand is zero-configuration at the client side, as all server capabilities can be discovered and browsed directly from the server.

CPU performance issues are very important in particular for the thermo interface, which lies at the heart of any process model. Real-world, quantified profiling data are not available at the time of writing for our prototype, apart from the qualitative observation that the overhead caused by OPC UA was not significant in this specific case.

On this topic there are certain general considerations that apply. The recent benchmark data in [12] indicate that the worst-case communication latency (round trip) for an OPC UA client-server configuration on a typical laptop is about 0.5 ms for 10 floating-point values (comparable to a thermo remote procedure call) and about 50 ms for 10000 values (comparable to a unit operation remote procedure call). Based on these figures a 10% performance degradation bound for off-line CAPE applications based on OPC UA can be achieved provided the design of the interface makes sure that whenever 10 variables are transferred, the associated calculation overhead is in the range of 5 ms, while whenever 10000 variables are transferred the calculation overhead is in the range of 500 ms. The technique of grouping remote procedure calls is well-known in the automation world and is integral already to the legacy, Classic OPC DA (data access) standard. From a process modeling point of view, it is perfectly feasible for unit operations which are simultaneously solved to call the thermo engine once to perform a number of flashes and physical property computations in a single remote procedure call.

4. Conclusions

We have presented a first-of-a-kind soft sensor prototype implemented as an OPC UA server. To fully exploit the OPC UA promise for portability, flexibility and performance the modelling calculation kernel used to implement the process simulation information model should be modern, designed around an object-oriented paradigm and portable; the C++ LIBrary for Process Flowsheeting satisfies these requirements and allowed fast prototyping of the application.

The soft sensor calculates certain properties such as heating value, explodibility, density, compression factor and dew-/bubble-points for hydrocarbon mixtures. The latter either with industry-standard cubic equations of state, or with more sophisticated and accurate equations of state for specific mixtures (GERG-2004) or finally with the PC-SAFT statistical equation of state. To make the prototype complete would require implementing the node creation interface, the State Machine information model and the Program OPC UA interface; in this way it would be possible to completely steer the soft sensor using OPC UA for set-up and configuration as well as for communication at run-time.

OPC UA, if accepted as an industry-wide standard, is a potentially revolutionary technology that could reshape the automation industry and its business model.
The challenge for the CAPE community now is to leverage CAPE-OPEN and define a basic, standard, expandable, designed-for-performance OPC UA information model for process simulation.

References

[1] Credle R., Akibola V., Karna V., Pannerselvam D., Pillai R. and Prasad S. “Discovering the Business Value Patterns of Chemical and Petroleum Integrated Information Framework” IBM redbooks 2009 ISBN 0738433136
[2] OPC Unified Automation Specification 1.01 Parts 1-9, OPC Foundation 2009
[3] OPC Data Access Specification, OPC Foundation 1996
[4] Wolfgang Mahnke, Stefan-Helmut Leitner and Matthias Damm “OPC Unified Architecture” Springer-Verlag 2009
[5] INTERNATIONAL STANDARD ISO 6976:1995 “Natural gas — Calculation of calorific values, density, relative density and Wobbe index from composition”
[6] INTERNATIONAL STANDARD ISO 12213:2006 “Natural gas -- Calculation of compression factor -- Parts 1, 2 and 3”
[7] O. Kunz, R. Klimeck, W. Wagner and M. Jaeschke, "The GERG-2004 Wide-Range Equation of State for Natural Gases and Other Mixtures", GERG TM15 2007
[8] J. Gross and G. Sadowski, "Perturbed-chain SAFT: An equation of state based on a perturbation theory for chain molecules", Industrial & Engineering Chemistry Research, vol. 40, pp. 1244-1260, Feb 21 2001
[9] C++ based OPC UA Server/Client SDK V1.0.0, Unified Automation 2009
[10] P. Greppi, “LIBPF: A LIBRARY FOR PROCESS FLOWSHEETING IN C++” Proceeding of the International Mediterranean Modelling Multiconference 2006, pp. 435-440
[11] CAPE-OPEN Laboratories Network, CAPE-OPEN Open Interface Specifications - Thermodynamic and Physical Properties Version 1.0, 2002
[12] Intel, AscoLab, UnifiedAutomation, Reducing Product Development Effort for OPC Unified Architecture (white paper) (2009)