Using IEC CIM standards and SOA Technology for Coordinated Voltage Control Application

Shengye Lu and Sami Repo
Dept. of Electrical Engineering
Tampere University
Tampere, Finland
lvshengye@gmail.com, sami.repo@tuni.fi

Mikko Salmenperä, Jari Seppälä
and Hannu Koivisto
Dept. of Automation Science and Engineering
Tampere University
Tampere, Finland

***Abstract*—Distribution System Operators (DSOs) rely on an increasing number of software applications for network operation and management. These applications need to exchange data with each other smoothly and efficiently. To facilitate inter-application communications and improve interoperability, IEC Common Information Model (CIM) standards and enterprise integration technologies, such as Service-oriented architecture (SOA), can be used. This paper focuses on one distribution management use case – "Coordinated Voltage Control", explains how CIM standards and SOA technologies can utilized to integrate Coordinated Voltage Control Matlab software with external systems such as SCADA. The paper covers design, implementation and laboratory demonstration for the communication aspect of this use case.**

***Index Terms*—Information and Communication, Interoperability, IEC CIM standards, Distribution automation, voltage control**

I. Introduction

Today Distribution System Operators (DSOs) are relying on an increasing number of information systems for network operation and management, and these information systems – likely from different vendors – will engage in more and more inter-application collaborations and data exchanges. With the advent of Smart Grid, information integration and interoperability will become even more crucial, because various "smart grid management systems" and external information providers have to exchange data and collaborate in a seamless manner.

One of such smart distribution grid management systems is "Coordinated Voltage Control" application, which aims at solving voltage violation problems in medium voltage (MV) and low voltage (LV) network, by controlling resources like distributed generation (DG) units and transformer tap changers [1]. "Coordinated Voltage Control" (CVC) application relies on extensive data exchanges with external systems, such as SCADA and smart meters, etc. It needs to take in large amount of measurement data from multiple data sources and send out set-point values, in a timely way. Needless to say, efficient data exchange plays a crucial role in this use case.

To improve inter-application communication efficiency and reduce integration costs, two measures can be considered: first, use commonly agreed way to express the data content; second, pursue a loose-coupled pattern for integration. The former can be achieved by utilizing standardized information model, for example, IEC Common Information Model (CIM). The latter can be realized by using modern integration technologies such as Service-oriented architecture (SOA) and Web Services.

CIM, or Common Information Model, is one of important smart grid standards that standardizes the messages/files exchanged between systems by providing a common vocabulary and message schemas [2]. It aims to facilitate information exchange and system integration in the power system operations and planning domain. In Europe, the usage of CIM is mainly concentrated on transmission level, for network topology exchange use cases. On distribution level, there are more and more research initiatives using this standard. For example, some researchers use CIM to import grid data into simulation tool [3]; some proposes a new generation of distribution management system based on CIM [4].

This paper focuses on communication perspective, aims at exploring the possibility of utilizing CIM standards and SOA technologies to facilitate data exchange for typical MV/LV network management applications, using Coordinated Voltage Control as an example use case. The paper covers design, implementation and laboratory demonstration for the CIM-standard based data communication solution in the Coordinated Voltage Control use case.

II. Background

This section briefly introduce the Coordinated Voltage Control application, especially its data flow with external systems such as SCADA, and traditional way of integrating these systems.

A. Coordinated Voltage Control use case

Voltage control is a widely researched topic in distribution operation, and it plays a significant role in congestion management. A. Kulmala et al. proposes "Coordinated Voltage Control" (CVC) solution in [1], trying to solve voltage violation problems in medium voltage (MV) and low voltage (LV) networks. The main idea of CVC is: A centralized coordinated voltage control algorithm calculates a series of set points in real time. These set points include the set points for Automatic Voltage Regulator (AVR), to control voltage and reactive power of DGs; and the set points for Automatic Voltage Controller (AVC) relays, to control substation voltage with On-load tap changer (OLTC) of transformer. The centralized CVC Algorithm is executed every

This work is supported by Finnish Cluster for Energy and Environment (CLEEN) research program SGEM (Smart Grids and Energy Markets).

10 minutes, and whenever some network voltage exceeds the feeder voltage limits.

The CVC algorithm needs the following input:

- substation voltage value (transformer secondary side)
- transformer on load tap changer position;
- measurement values of active, reactive power, voltage of each DG unit;
- measurement values of active and reactive power of each feeder (at beginning point);
- breaker status of each DG unit.

These input data can be collected from a variety of external system, including SCADA, smart meters, state estimator, etc. The output of the CVC Algorithm includes:

- voltage set points of AVC relays for transformer's OLTC;
- active and reactive power setting points for each DG unit (production curtailment and set point of DG unit AVR).

These set points are sent towards SCADA and smart meters.

B. Data flow between CVC and external systems in laboratory demonstration environment

In a typical DSO control center, voltage control component is provided as a part of Distribution Management System (DMS). In our laboratory demonstration environment, the CVC algorithm is implemented as an independent Matlab application. Network (including substation, DG generators and feeders) is simulated by RTDS simulator [5].

For the sake of simplicity, in our demonstration environment, SCADA system collects all the real-time measurements from the simulated network, and it also receives all set-points calculated by CVC algorithm and then sends towards transformer AVC relays and DG AVRs. SCADA system constantly exchanges measurement and control data with RTDS simulator, and their communication uses IEC 60870-5-104 protocol.

Hence, the major external system that CVC algorithm needs to cooperate with is SCADA system. The information exchange between CVC algorithm and SCADA is described by the sequence diagram in Figure 1.

C. Traditional way of System Integration

Many SCADA systems use internal OPC DA (OPC Data Access) server to expose process data towards external systems. Therefore, a straightforward way to integrate CVC algorithm and SCADA is using OPC DA standards.

In our demonstration environment, the SCADA system is ABB MicroSCADA, running on a Windows desktop. CVC algorithm, as a Matlab application, is running on a different computer. In order to read measurement values or write set point values from/to MicroSCADA using OPC DA, we have developed a built-in OPC DA client inside the CVC Matlab application, using Matlab OPC Toolbox.

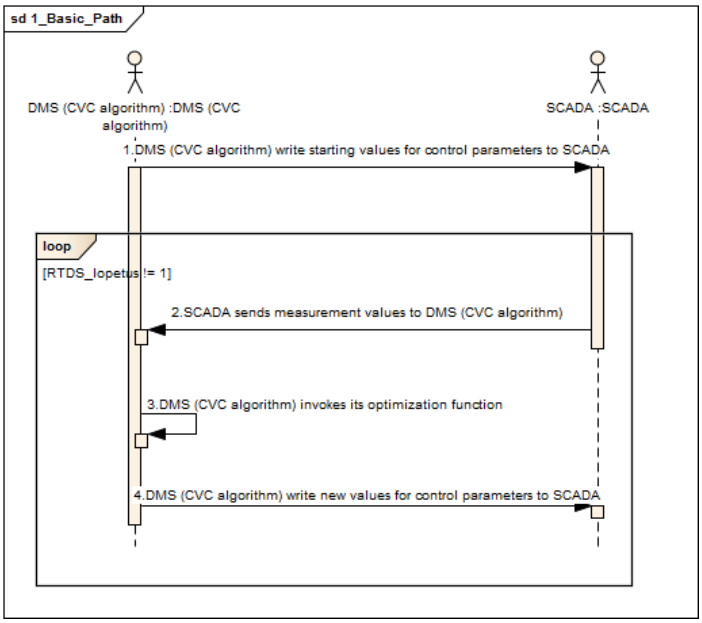

Figure 1. Sequence Diagram for CVC application

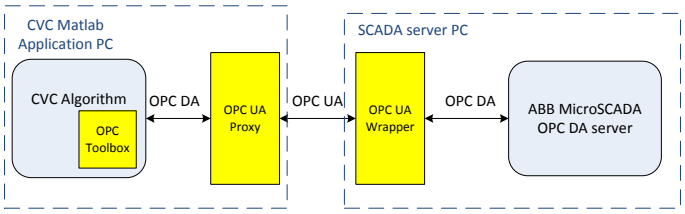

Figure 2. Using OPC UA to integrate CVC Matlab Application with ABB MicroSCADA

However, experiences have shown that remote connection to OPC DA server involves complicated configurations and is very difficult to achieve. To solve this problem, we use newer standard – OPC UA (OPC Unified Architecture), to integrate MicroSCADA and CVC application. This is realized by running OPC UA Wrapper on MicroSCADA desktop. OPC UA Wrapper connects with MicroSCADA using OPC DA standards, and in the meanwhile, presents MicroSCADA as OPC UA server towards external systems. On CVC Matlab application PC, runs OPC UA Proxy, which communicates with MicroSCADA desktop using OPC UA and interacts with CVC application using OPC DA. Because the communication between MicroSCADA desktop and CVC application is now using OPC UA standards, remote access becomes much easier. The whole communication structure is summarized in Figure 2.

However, integration in this way is very inefficient, consuming lot of resource from computer systems, because both OPC UA Wrapper and OPC UA Proxy require considerable amount of CPU resource.

The even bigger issue with this traditional way of integration is that it does not scale. To communicate with external systems like SCADA, CVC application has to "understand" other system's data format, semantics and message exchange protocol – in this case, OPC DA or OPC UA standards. Therefore, one

dedicated "adapter", e.g., OPC DA client, has to be developed. If CVC application needs to cooperate also with another external system, such as Smart Meter, then another dedicated "adapter" also needs to be added to CVC application, in order to "understand" Smart Meter's data format and protocol. As the number of integrated systems grows, the number of needed "adapters" will grow exponentially, which will result in very expensive development and maintenance cost.

To achieve better scalability and efficiency in communication, we switch to use CIM standards and Service-oriented architecture (SOA) for integrating CVC application and SCADA.

III. Using IEC CIM Model to define message payload

To let software components, such as CVC application and SCADA in aforementioned CVC use case, "understand" each other's data format and semantics in more scalable and efficient manner, CIM standards (IEC 61970/61968) have been chosen to design message payload for data exchange among these components.

CIM, or Common Information Model, is a set of standards developed by Electric Power Research Institute (EPRI), and now maintained by International Electrotechnical Commission (IEC)[6]. CIM standardizes the messages/files exchanged between systems by providing a common vocabulary (or semantics) and message/file schemas, in order to facilitate information exchange and system integration in the power system operations and planning domain. In Europe, although CIM is getting more and more attention, its applications have been mainly concentrated on transmission level. On distribution level, CIM is not yet commonly used.

In our demonstration, we mainly use IEC 61868 part of CIM standards. IEC 61868 series standards define contextual profiles, i.e., subset of the full CIM model, for all the major software elements for Distribution Management Systems (DMS), and is intended to be implemented with middleware services that broker messages among applications. It is the starting point to define message structure and payload for the CVC use case.

A. Message payload structure for CVC use case

Based on input and output data listed in section II-A, we design message payload for SCADA and CVC Application by modifying the contextual profile defined in "IEC 61968-3: Interface for Network Operations". The structure of the message payload looks like Figure 3.

In Figure 3, the root element "MeasurementsAndControls" contains the following optional sub elements:

- AnalogValue – for analog measurement values (from SCADA to CVC)
- DiscreteValue – for Breaker Status (from SCADA to CVC)
- SetPoint – for control signals (from CVC to SCADA)

Each of these sub elements contains detailed information such as name, value, timestamp, etc.

Note this message payload does not exchange network topology, because SCADA and CVC Application should already have network topology information.

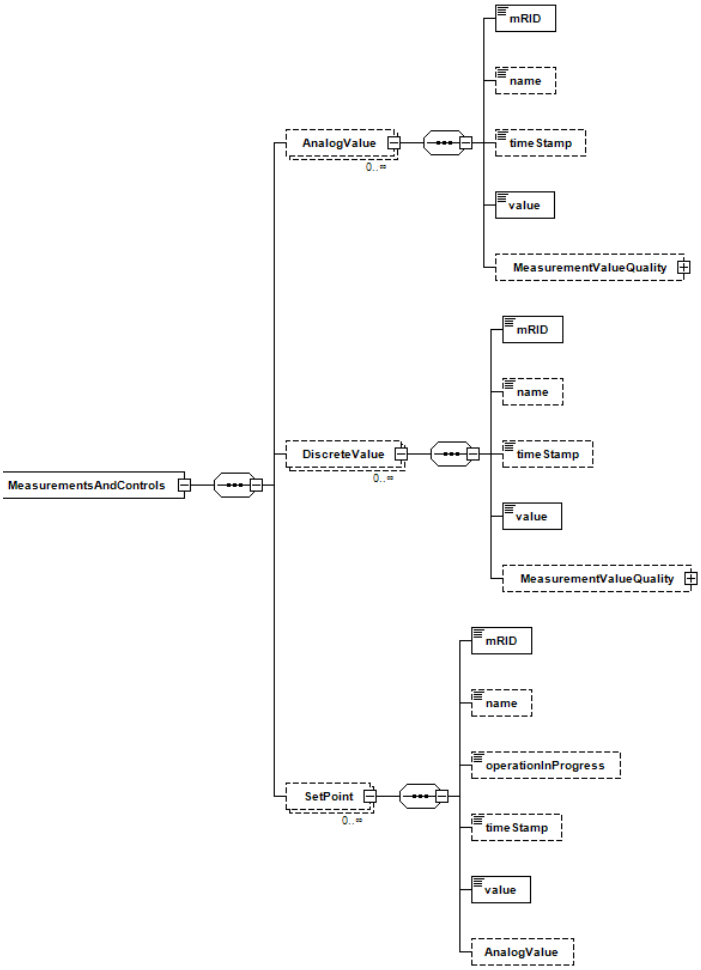

Figure 3. Message payload structure designed for CVC application

IV. Loose-coupled System Integration – using SOA technologies

Section III-A defines the "vocabulary" or semantics of messages. The actual message delivery is realized by using Service-Oriented Architecture (SOA) and Enterprise integration technologies. SOA provides a loose-coupled pattern for integration. In this pattern, application components provide services to other components via messaging, typically over a network. Message sender does not interact with message receiver; instead, Enterprise Service Bus (ESB) takes care of message delivery.

Using SOA as integration solution for CVC use case, there is no need to establish dedicated inter-application communication links between each message sender and message receiver among those CVC components – i.e., CVC Application, SCADA, and potentially several others. These dedicated links are now replaced by a single communication link, ESB. CVC components just need to connect with the ESB via their Web Service adapters and let the ESB take care of message deliveries. The communication structure looks like Figure 4. In Figure 4, every CVC component has one Web Service adapter or interface, which "wraps" the CVC component as Web Service.

The integration steps for CVC use case in our demonstration

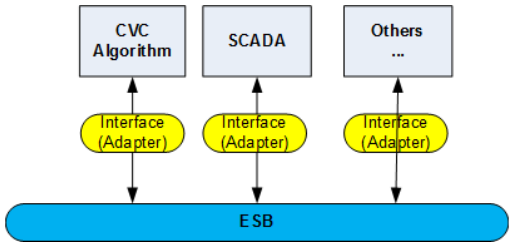

Figure 4. Message deliveries handled by a single communication link – ESB

environment are as follows:

1) Define service contract based on CIM standards.
   – In our demonstration, CVC Application and SCADA both need to be "wrapped" as SOAP (Simple Object Access Protocol) Web Services. Therefore, this step means using CIM standard to specify message payload for these two services, and generating WSDL (Web Services Description Languages) files, which are service contracts for SOAP Web Services.
2) Implement SCADA Service and CVC Service according to service contract.
   – SCADA Service is essentially an adapter for MicroSCADA. It "wraps" MicroSCADA with Web Service interface. Upon receiving SOAP request, it reads or writes data from/to MicroSCADA using OPC. Similarly, CVC Service "wraps" CVC Matlab application with Web Service interface, and interacts with it when receiving SOAP request.
3) Using ESB to assemble and orchestrate services based on business logic.
   – In this step, the direct OPC communication between CVC Matlab application and MicroSCADA as mentioned in Section II-C, is replaced by ESB. Measurement and control data will be fetched by CVC Service and SCADA Service, and then ESB will take care of message routing and delivery.

A. Step 1: Define service contract based on CIM standards

Service contract defines how request or response message to or from the service should be. Because SCADA and CVC application are "wrapped" as SOAP Web Services, their service contract has to be WSDL file, and the messages exchanged among them are in eXtensible markup language (XML) format, encapsulated as SOAP messages.

The message payload designed in Section III-A is only part of service contract. In addition to message payload, service contract also contains other parts such as Message Header. Message Header includes elements Verb and Noun [7]. In CVC use case, the choice of Verb are "Change", "Get", while Noun is "MeasurementsAndControls".

The remaining missing part of service contract can be created by referencing templates provided in IEC 61968-100[8]. IEC 61968-100 is an implementation profile, which provides XML Schema templates for Common Message Envelope, for defining web service operations, and WSDL templates. With these templates, as well as the message payload designed in III-A, we can generate service contract (i.e., WSDL file) for CVC use case actors – CVC Algorithm and SCADA.

B. Step 2: Implement services

The service contract (i.e., WSDL file) created in Section IV-A can be used for both CVC Algorithm service and SCADA service. The next step is to implement these two SOAP Web Services. Many software-programming frameworks, such as Spring Web Services and .NET Framework, can generate code skeletons directly from WSDL files, thus greatly simplify development work.

1) SCADA Service: SCADA Service is the SOAP Web Service interface for ABB MicroSCADA. It functions as a gateway, deployed at one URL (Uniform Resource Identifier) address. SCADA Service communicates with MicroSCADA locally using OPC DA. It can receive two types of SOAP request messages from external systems:

- GetMeasurementsAndControls – this message is for reading latest measurement values from SCADA. Upon receiving"GetMeasurementsAndControls" request message, SCADA Service will read measurement values from SCADA, encapsulate these values into response message and then send back.
- ChangeMeasurementsAndControls – this message is for writing set-points values for control parameters (starting values or new values) towards SCADA. Upon receiving "ChangeMeasurementsAndControls" request message, SCADA Service will write set-points values (from request message's payload) to SCADA.

Inside SCADA Service, there is one built-in OPC DA client implemented by using open source framework, which allows SCADA Service interact locally with SCADA's internal OPC server.

2) CVC Algorithm Service: Similarly, CVC Algorithm Service is the SOAP Web Service interface for CVC Matlab algorithm, deployed at another URL. It communicates with CVC Matlab application locally using file system read/write operations. It can also receive two types of SOAP request messages:

- GetMeasurementsAndControls – this message is for retrieving set-point values from CVC algorithm. Upon receiving "GetMeasurementsAndControls" message, CVC Web Service will read set-point values from a file, which is constantly updated by CVC Matlab application, and then encapsulate these values into response message and send back.
- ChangeMeasurementsAndControls – this message is for providing measurement value as input for CVC Algorithm. Upon receiving "ChangeMeasurementsAndControls" message, CVC Web Service will write latest measurement values (from request message's payload) into a file, from where CVC Matlab application can read and then start launching its calculation functions.

It is worth noting that the CVC Matlab application, as mentioned in II-C, can now be simplified – there is no need to have a built-in OPC DA client component in CVC Matlab application anymore, because CVC Matlab application won't directly communicate with any external systems. Instead, it simply reads SCADA measurements from a local file and

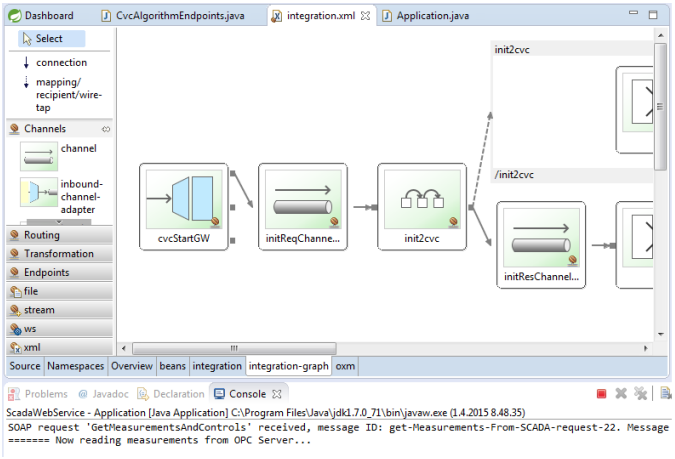

Figure 5. Screenshot on Spring Integration Framework IDE. It shows part of the configuration for this integration application, which assembles and orchestrates SCADA Service and CVC Algorithm Service, in order to realize the whole CVC use case logic.

write control values to another file. CVC Service will update or retrieve these files when receiving request messages. This manifests the loose-coupled nature of Enterprise integration: the software component itself does not need to care about message flow. It simply produces or consumes data, and there is no need to take care of where to send or receive message, which will be handled by ESB. This topic will be further elaborated in next section.

C. Step 3: Using ESB to assemble and orchestrate services

To let messages actually flow between CVC Service and SCADA Service, we need to use Enterprise Service Bus (ESB) to assemble and orchestrate them. There are many ESB frameworks for enterprise integration. In our demonstration, we use one open source framework called Spring Integration.

Spring Integration provides a Java-based framework with support for Enterprise Integration Patterns. It functions as a lightweight ESB, facilitates asynchronous, message-driven behavior among software applications.

When using Spring Integration solution, CVC Algorithm Service and SCADA Service are treated as external endpoints. On top of that, one special integration application is created for the purpose of assembling CVC Service and SCADA Service. This integration application communicates with these two endpoints using SOAP protocol and orchestrates the message flow. It retrieves a message from an inbound queue, parses and transforms it, and then transmits to one endpoint (i.e., CVC Web Service or SCADA Web Service). After receiving reply, it transforms the SOAP response to a new SOAP request and transmits it to another endpoint. All these steps can be realized simply by using configuration files. Figure 5 is a screenshot on Spring IDE (Integrated Development Environment). It shows part of the configuration for this integration application.

V. Result and Conclusion

Using IEC CIM standards and SOA technology as integration solution, the communications between SCADA and CVC Application become modular, reusable and easier to scale. If one component is changed, e.g., ABB MicroSCADA is replaced by another SCADA system using different interface rather than OPC, then only SCADA Service (the adapter) needs to be modified. If the CVC use case needs a third component, then we can just create a service for the new component, and modify the integration configuration files. This way can greatly increase interoperability and reduce integration complicity. It is much more efficient than traditional way of integration, which uses dedicated channel and adapter for each communication sender and receiver, like in Figure 2.

On the other hand, SOA technology is not suitable for time-critical applications, e.g., protection, because real-time requirement is not the focus of SOA technology. It does not provide straightforward way for handling Quality of Service (QoS) in automation context. It is designed for conserving messages and delivering them when possible. This approach is in strong contrast to most automation related functions where data has limited lifespan and should be discarded if time limit is exceeded.

References

[1] A. Kulmala, S. Repo, and P. Järventausta, "Coordinated voltage control in distribution networks including several distributed energy resources," *IEEE Transactions on Smart Grid*, vol. 5, pp. 2010–2020, Jul. 2014.

[2] M. Uslar, M. Specht, S. Rohjans, and J. Trefke, *The Common Information Model CIM: IEC 61968/61970 and 62325 - A Practical Introduction to the CIM*. Springer, 2012.

[3] M. Armendariz, A. Saleem, L. Nordstrm, and M. Brugeron, "Facilitating distribution grid network simulation through automated common information model data conversion," in *Proc. 2015 IEEE Eindhoven PowerTech*, Eindhoven, Netherlands, Sep. 2015.

[4] L. Fiaschetti, M. Antunez, E. Trapani, L. Valenzuela, A. Rubiales, M. Risso, and G. Boroni, "Monitoring and controlling energy distribution: Implementation of a distribution management system based on common information model," *International Journal of Electrical Power & Energy Systems*, vol. 94, pp. 67–76, Jan. 2018.

[5] V. Tuominen, H. Reponen, A. Kulmala, S. Lu, and S. Repo, "Real-time hardware-and software-in-the-loop simulation of decentralised distribution network control architecture," *IET Generation, Transmission & Distribution*, vol. 11, pp. 3057–3064, Sep. 2017.

[6] EPRI, "Intelligrid common information model primer: second edition," Palo Alto, CA, Tech. Rep., Oct. 2013.

[7] *Application Integration at Electric Utilities - System Interfaces for Distribution Management - Part 1: Interface Architecture and General Requirements*, IEC Std. 61 968-1, 2002.

[8] *Application Integration at Electric Utilities - System Interfaces for Distribution Management - Part 100: Implementation profiles*, IEC Std. 61 968-100, 2013.