https://doi.org/10.26552/com.C.2011.2A.44-48

Marek Hoger – Peter Bracinik – Marek Roch *

SIMULATION OF A POWER SUBSTATION'S CONTROL SYSTEM OPERATION

This paper describes a simulator of power substation automation system developed as part of software suit for complex simulation of power substation operation. This software simulates behavior of a distributed control system built from a group of interconnected and cooperating intelligent electronic devices. The basic structure and communication principles are derived from standard IEC 61850.

1. Introduction

Operation reliability of an electrical transmission or distribution system is markedly affected by the operation reliability of its nodes – power substations.

A power substation is a distributed system built from high number of power and control devices. Its key component is substation control system because it's necessary for proper cooperation of high number of power substation's devices (there can be hundreds of devices in a substation).

Many concepts of substation control system architecture were developed in the past, based on different control techniques (from oldest relay based systems to modern systems based on intelligent equipment and new communication technologies). The evolution of power equipment, protection and control devices, and of course evolution of communication technologies, led to development of new standards in substation automation like UCA2 and IEC 61850. These standards are very flexible. The system can be designed as strongly centralized or highly decentralized, according to local conditions and specific user needs. Improper design can cause markedly decrease of systems performance or unexpected behavior. Therefore, computer simulation can be very useful to simulate and predict the behavior and performance of such a system.

There are some tools for simulation of substation operation, mostly focused on simulation of protective relays and control system. But these tools are proprietary, have (relative) limited functionality and it is difficult (or impossible) to extend the application and add new functions.

Therefore, we decided to develop own substation operation simulation software. The software should be flexible and extensible, usable by solving wide range of problems. The scope of use should be primary the testing of new control and protection functions, developing of algorithms for autonomous substation control and analysis of system behavior in different operation situations (primarily by faults). Another important application of this software is in teaching various subjects related to control of power substations, protection of electrical systems and use of information systems in energetics.

2. The Power Substation Simulator

The power substation simulator consists of three independent cooperating applications – power system simulation (PSS), control system simulation (CSS) and graphical user interface (simple SCADA application). This structure is well corresponding with real operation of a power substation.

The power system simulator performs the calculation of state values (voltages, currents, device state signalization, etc.) and contains models of power equipment (switching devices, power transformers, instrument transformers or sensors). The physical model was created in Matlab/Simulink and uses user defined library of hardware models, based on SimPowerSystems library. The communication with the control system is realized using blocks for TCP communication from Instrument Control Toolbox library. The use of Simulink, as modeling environment, enables to modify the device models easily, so the user can add or remove their functionality according to actual demands.

The control system simulator performs simulation of control and protection devices and their communication. The control system reads the actual state values sent from physical model, process this data and if needed, sends requests for hardware state changes

* **Marek Hoger, Peter Bracinik, Marek Roch**
Department of Power Electrical Systems, Faculty of Electrical Engineering, University of Zilina, Slovakia, E-mail: Marek.Hoger@kves.uniza.sk

(e.g. opening a circuit breaker). It also generates reports for user interface application containing information about important state changes (change of switch position, reactions of protections). Detailed specification of this software is introduced later in this paper.

The user interface application provides user with information about actual system configuration, measured values and important events in graphical form. The application supports control of the power equipment using the hardware manipulation functions of the control system. It also creates an events log to a simple text file and optionally to a SQL database.

The data exchange between these applications is realized over TCP/IP protocol, so each of these applications can run on different computer, which allows effective use of computing capacity of multiple computers.

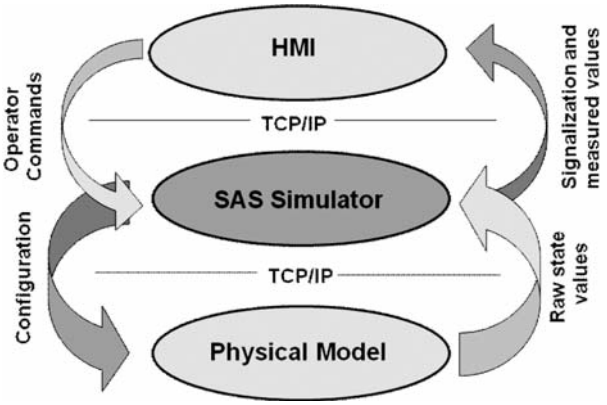

Fig. 1 Structure of power substation simulator

3. The Control System Simulator

The control system simulator is an application simulating the operation of intelligent electronic devices (IEDs), of which modern substation automation systems are built. The application simulates the behavior of each device (protective, control and system functions) and also simulates communication between devices. This application is a key component of the simulation suit of substation operation.

Today, an object oriented approach to modeling of substation automation system is used and this approach is also used in modern standards for communication in power substations (e.g. UCA2 or IEC 61850). Therefore, the application was developed in C# language – an object oriented language based on C language and .NET framework.

A power substation is a very complex system and it's difficult to identify all necessary functions and develop optimal object structure, so the object structure and basic communication principles were adopted from international standard IEC 61850. But, this software is not the implementation of this standard. The full implementation of this standard would be too complicated and in fact useless for the planned scope of use of the application. The object structure, mapping of communication services and configuration language were simplified and modified to meet our needs. The simplified structure of the application in UML notation is in Fig. 1.

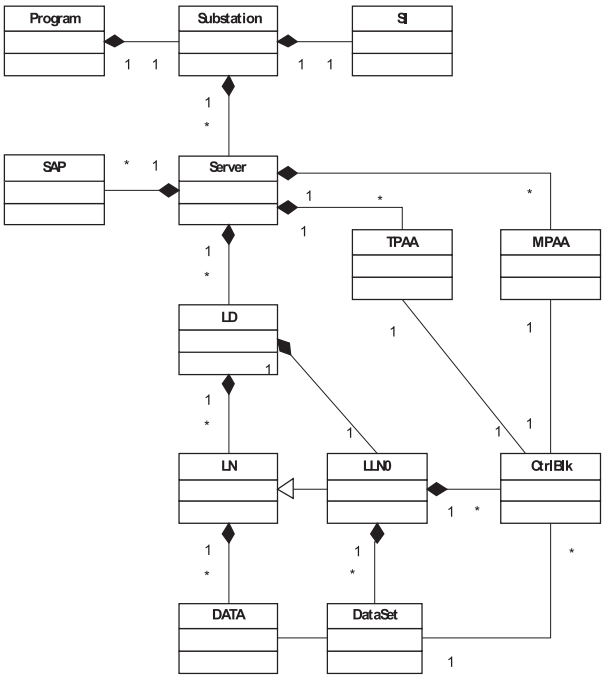

Fig. 2 Simplified structure of control system simulator

On the top of the structure is the *Program* class. It's the root element of the structure. This class loads the configuration on startup and builds the data structure according to configuration specified in a file in XML format. After startup, it provides user interface services and commands interpreting. Program class contains a single instance of *Substation* class.

Substation class acts as a container for *Server* class instances representing IEDs building the automation system. This class itself has no functionality and is used only for better organization of *Server* instances.

Simulation interface (SI) class provides the interconnection to physical model over a TCP/IP connection and controls the execution of functional cores of *Server* class instances. Simulation interface creates a TCP listener and then waits for incoming raw data from physical model. Each data is transmitted with a tag describing the source of the data. SI reads the tag, then searches the signals mapping table for this tag and if an association is found, writes the data to input buffer of associated logical node. If there is no association defined in the signal mapping table, the data is

ignored. Reversely, if a logical node needs to change state of a device in physical model, it writes the command and its own reference to command buffer of the SI, SI searches the signal mapping table and if an association is found, it sends the signal code (and optionally additional parameters) over TCP back to physical model. The last important function of SI is time synchronization. With each raw data message from the physical model the actual simulation time is transmitted to the SI. When a time update signal is detected, SI updates the system clock and runs functional cores of *Server* class instances. The signal mapping table is stored in a file in XML format and can be modified manually in every text editor or using an auxiliary tool (created as part of the software suit) able to automatically detect all available signals and related logical nodes.

Server class instance represents a single IED. It contains services for communication with other *Server* class instances and services for data manipulation on physical device level. Communication services are protocol independent and the final mapping to a real protocol is done using instances of *SAP* class (Service Access Point class used for general purpose communication), *TPAA* class (Two Party Application Association which supports peer to peer connection services) and MPAA class (Multiple Party Application Association class supporting multicast services). *Server* class also contains one or multiple instances of *LD* class. The *SAP* class developed with the simulator provides the mapping of services to TCP/IP and supports the binary serialization of data object instances, so they can be transmitted over TCP. In the future, it's possible to develop a service interface for other network technologies and protocols like SPABUS or MODBUS over a serial line, or map the services to MMS as defined by IEC 61850. For the binary serialization a standard binary formatter provided with .NET framework is used. For proper deserialization of received data in applications created in other programming environments (e.g. LabVIEW) a special data handler must be created in C# (or other .NET language) and imported as .NET component.

LD class acts as a container for logical node instances. Using multiple instances of *LD* class helps keep the data structure more compendious. Typical example of using multiple *LD* instances is using one instance of *LD* for bay control functions and other *LD* instance for protective functions. Each *LD* must contain one instance of *LLN0* (Logical Node Zero).

LN class represents an elementary system function of SAS (e.g. overcurrent protection). These elementary function blocks are called logical nodes. IEC 61850 defines 86 different logical node classes (one for every known elementary system function) divided to 13 basic categories. The requested functionality of an IED is than performed by a collection of cooperating *LN* instances. The LN instance uses *DATA* class instances as an interface for communication with other system blocks. *DATA* is a collection of attributes providing elementary information describing setting of logical node, current state of logical node and additional information, related to the specific function of the LN.

A special kind of logical node is the *LLN0*. This class doesn't perform any system function, but it contains data describing health, mode and vendor of the device and acts as a container for *CtrlBlk* (Control Block) instances and *DataSet* instances.

Control blocs are classes derived from *CtrlBlk* class. These blocks perform messaging and logging functions. Instance of a control block observes a collection of data or data attributes referenced by a *DataSet* instance. Each dataset member ought to have definition of trigger conditions indicating which event should invoke the control block to create and send a message (or write a new log entry). Trigger conditions can be set to data change, quality change or data update. When control block detects an event corresponding to trigger conditions, a report or data message is sent using the appropriate message handling object (e.g. *TPAA* or a *MPAA* instance associated with the control block), or a log entry is added to archive (if the control block is Log Control Block).

Of course, this is only brief description of the structure. The real structure is much more complex, similar to the object structure defined in standard IEC 61850-7.2

The configuration of a substation is described in a collection of configuration files. There are three types of configuration files.

The .sub file describes the structure of a substation and table of all IEDs building the system and device configuration files assigned to these IEDs. It is the main project file and normally each project contains only one file of this type.

The .cid file describes the configuration of a single IED. It contains all information about the internal structure of a *Server* class and setting of all necessary attributes. Format of this file is derived from CID file format defined in IEC 61850-6. There is one .cid file for each server instance.

The .sig file configures the simulation interface. It contains the signal mapping table, describing all transmitted signals, source devices and related logical nodes. If this file is not specified, simulation interface simple ignores all incoming data except the simulation time.

These file types are all in XML format and it is theoretically possible to create and edit these files in any ordinary text editor. But a single .cid file can contain more than two thousand lines of configuration code, so it is practically impossible to create and handle the configuration files without a specialized configuration tool developed as an auxiliary tool for the substation operation simulator. This tool enables to create the whole configuration through a user friendly graphical interface, keeps the consistency of the configuration and contains tools for automatic code generation witch rapidly speeds up the process of creating a new project configuration.

The run of the application is as follows: on startup, application reads configuration files describing the automation system, configuration of each IED and configuration of simulation interface. After the initialization phase, system waits until physical model connects. As mentioned above, interconnection between control

system simulator and physical model realizes an instance of simulation interface class. When the connection is successfully established, simulation interface waits for incoming raw data (state signalization, instantaneous values of voltages and currents, actual simulation time). The simulation interface reads the signal code of each received signal and maps the signal to corresponding instance of logical node. SI also controls the execution of simulation; SI starts functional cores of IEDs each time when a time update signal is received from physical model, so both simulations are synchronized. If the connection to PSS is lost, execution of IED cores is terminated until the PSS reconnects. Sometimes, mostly when debugging new configuration, it's necessary to run the simulation without connection to physical model. Therefore, the SI enables to run the simulation cores using internal timer with user defined step size and delay. This internal timer is controlled using console commands.

The user interface is realized as a simple console with command prompt. Build–in command interpreter supports various commands for data manipulation, browsing the data structure and application control. Command interpreter also supports simple scripting – when frequently performing the same sequence of commands (e.g. initialization of substation to a specific configuration other then default state), the sequence can be stored to a text file and simple reused using the *run* command.

This console interface is primary used for testing and debugging of new configuration. As a standard user interface, SCADA application should be used.

This software suit should be primary used for analysis of control system's behavior. Therefore the simulator is able to store many types of events (value changes, state changes, sending and receiving messages) to SQL database for further analysis. This enables to study behavior of the system in various operation conditions and response of the system to various events.

4. Configuration Utility

The configuration of a single IED can contain thousands of configuration code lines. Even relative small substation contains about 15 IEDs. Because it's difficult to keep the configuration code consistent for a single device, on substation level it is practically impossible, the configuration utility has been developed.

This application has an easy to use graphical user interface. User simply creates the configuration in graphical form by drawing the configuration schema using built-in drawing tools. This process of configuration creation has two phases.

First phase is creating the power equipment structure of the substation. For better orientation in bigger substations, the equipment is concentrated to bays and each bay is related to a voltage level. This hierarchy is used also for addressing of equipment. Each component needs to have a unique system address. This so called object reference is a combination of the related voltage level name, related bay name and the bay component divided by dots (e.g. a circuit breaker QM1 located in bay Q1 related to voltage level E1 has the object reference E1.Q1.QM1). There are often more bays with the same equipment structure in a substation. Therefore a bay cloning function is implemented. It allows the user to use existing bay structures for creating new ones. Afterwards the user is asked to specify the name for a new bay and the program automatically changes the object references according to the new bay name. The user can also define physical connections between power equipment components, but for SAS configuration is the specification of interconnections not necessary (connections are used for proper physical model creation). The whole power system configuration is stored in an .xml file and can be also used in other application (e.g. SCADA application).

When the power system contains at least one bay with an IED, the configuration process can continue to the second phase – IED configuration creation. The user can start building the IED configuration from scratch using the generating tool in Device menu or assign an already existing configuration to the IED instance.

The generating tool creates a new configuration file, associates this file with the IED instance and creates a basic configuration structure with a single LD and all necessary logical nodes according to the structure of power system of the bay containing the IED instance. Then, user can freely add or remove logical devices, logical nodes and other configuration objects according to tasks the IED should perform. After creation of the configuration structure, user has to configure each LN, ControlBlock, DataSet instance (e.g. to define the DATA structure, set triggering conditions, add data references do dataset, etc.). The system automatically checks the consistency of the settings and doesn't allow to create improper data structures (e.g. it doesn't allow to add DATA to a logical node instance, if the specific LN class doesn't support this DATA class or to add a DATA reference to a DataSet, if the referenced DATA doesn't exist). At last, the user must define the relation between blocks. These relations are displayed as wires connecting the blocks.

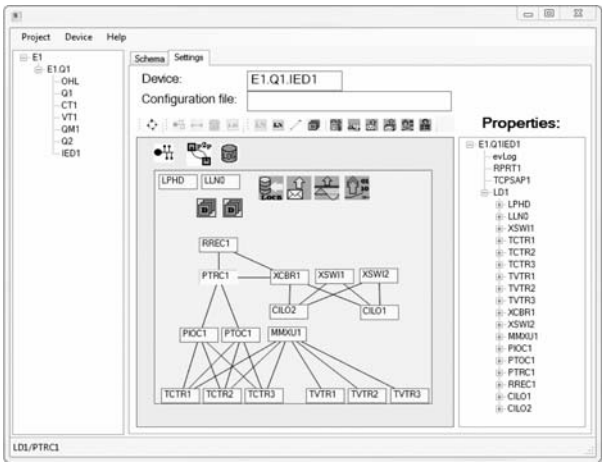

Fig. 3 Example of a simple IED configuration

Assigning an existing configuration is useful when the substation contains bays with the same or very similar configuration. When assigning an existing configuration to an IED, the system checks if the configuration structure is compatible with related power equipment structure. If the configuration is compatible, a copy of the source configuration file is created and all object references are updated automatically.

The configuration of each device is stored in a separate file in XML format, so it's possible to edit the configuration using an ordinary text editor or XML editor, but this is not recommended (because the consistency of so created configuration is not longer guaranteed).

5. Conclusion

The software described in this paper is a key part of an integrated software system for complex simulation of a power substation operation. It provides the simulation of a substation control system operation. The complete suit also contains a power system simulator, SCADA application and some additional tools speeding up the process of configuration of a new substation model (control system configuration tool and physical model configuration tool). In contrast to proprietary software, which is mostly closed and difficult to modify or extend, this system can be modified, extended and upgraded according to our current needs. There are several possible applications of this software.

The first possible application is analysis of data exchange within the substation for purpose of minimization of transferred data volume. The experiences with the real operation of control systems based on IEC 61850 showed that some optimization is often necessary.

The second application is analysis of processes and reactions of the system on faults on distribution lines and acquisition of knowledge usable by fault location on 22 kV distribution lines – the localization of failures in compensated networks with tree structure is a difficult problem. The tree structure of the network causes that many fault location methods can't exactly specify the fault point and are returning more possible fault points. The use of some kind of knowledge processing algorithm could provide additional information about the fault, because the system could specify the probability of correct fault location for each detected potential fault point and so provides additional information for dispatcher saying how trustworthy is the information form fault locator.

Developing and testing of algorithms for autonomous substation operation control is another possible application of this software suit. Such a system could handle most of situations occurring by the operation of a power substation and solve most of problem situations without need of operator's intervention.

Of course this software is well applicable in education process and can help by teaching subjects related to substation automation.

The development of this system has not been finished yet, but the software components are now in final stage of development (testing of stability, bugs fixing, some code cleanup). After finishing this phase, the system will be ready for planned use and start of development of a new generation is planned.

The new generation of the control system simulator shall have more modular structure based on plug–in modules for faster and easier development of new function blocks (or modifying the existing). Also some changes in configuration language are planned, the syntax is still unnecessarily complicated. It is also planned to create new service access point classes implementing real protocols (e.g. SPABUS), so it will be possible to connect the substation model with real devices and create some kind of hybrid model (the system will be built particularly from simulated and particularly from real devices).

Acknowledgement

This work was supported by the Slovak Research and Development Agency under the contract No. APVV-0560-07.

References

[1] HOGER, M.: *Model of Operation for Electrical Station (in Slovak),* Proc. of 10. Int'l conference Electric Power Engineering 2009, Kouty nad Desnou, ISBN: 978-80-248-1947-1

[2] IEC 61850-6: Configuration Description Language for Communication in Electrical Substations Related to IEDs, *IEC*, 2004

[3] IEC 61850-7.2: Basic Communication Structure for Substation and Feeder Equipment –Abstract Communication Service Interface (ACSI), *IEC*, 2004

[4] IEC 61850-7.4: Basic Communication Structure for Substation and Feeder Equipment – Compatible Logical Node Classes and Data Classes, *IEC*, 2004

[5] PENDER, T.: *UML Bible,* Wiley, USA, 2003.