XIII SIMPASE
SIMPÓSIO DE AUTOMAÇÃO DE SISTEMAS ELÉTRICOS

AB/YY

METHODOLOGY TO AUTOMATE THE GENERATION OF SCADA APPLICATIONS IN ELECTRICAL SYSTEMS

Clovis Simoes (*)
Spin Canada Ltd.

Túlio Pereira da Silva
Spin Canada Ltd.

SUMMARY

This paper presents a methodology used in the development of SCADA software applications, oriented to the control of any process, which greatly reduces the time of development and implementation of the application, as well as increases its quality and minimizes the possibility of errors.

This methodology, called Lean Automation (LA), is based on the concept of a component, which becomes a SCADA object, included in the application in a single click, and contains all functionality oriented to monitoring and controlling the process or part of it. With a few clicks you can add some components to the SCADA that correspond to the entire application

This methodology allows the deployment of electrical applications in record time (parameterization, factory testing and commissioning), with high quality and level of standardization, very low level of errors and low cost. In addition, it maintains knowledge of the deployment process within the software and not only the technicians who participated in the enterprise.

KEY WORDS

SCADA; Lean Automation; Automatic generation of applications; automation of electrical systems; Asset management

1.0 INTRODUCTION

One of the authors participated in the development of four SCADA systems over the last 30 years and was integrator of dozens of applications using these developed systems. From this experience and the use of state-of-the-art tools in the development of its latest SCADA, the methodology was incorporated into the software itself, that is, software modules were developed that automate the process of generating applications, based on the knowledge of the procedures of integration used in the automation of electrical systems, as well as the knowledge of the communication protocols used and the regulatory aspect associated to the controlled electric process, from norms established by the regulatory agencies.

The process for the automatic generation of all the points of a system, screens, reports, etc., whether for automation of a substation or for the automation of a wind farm, is based on a set of procedures that must be developed in an organized way, following a sequence, which leads us to the end of SCADA parameterization activity, tests in a laboratory environment, tests in the factory, with the newly installed electric cubicles, and commissioning in the field after installation of the cubicles and passage of the cables.

If we consider the automation application of a distribution utility´s substation, the unit to be automatized is the bay. A substation, basically, consists of a few types of bays: bay of line, transformer, capacitor bank, auxiliary services, etc. These bays are treated in a standardized manner by the utility. Thus, for example, any bay of a distribution substation has a repeatable data model, with information of type: maneuver equipment data, electrical measurement data, data protection of existing equipment in the bay, data of events and alarms associated with the states and measures of the bays, data rules to store the information in historical files, etc. These bays information are placed in templates and, when you define the architecture of a new substation to be deployed, routines will automatically explode all the data that will be processed in those models, generating the entire database of the application, with events, alarms, and historical files, etc. In addition, the bay type itself is associated with its schematic draw, following rules associated with company culture. The drawings of these bays are available in symbol libraries and instantiated, automatically.

Thus, from the models associated with the company culture, when informing the types of bays of a given substation, and how many units of each bay will exist, it is possible to, automatically, generate all points of the substation, with associated alarms / events, their addresses in the IEDs, and the historical data tables. More than this, we will have all the screens of the substation, the rules to navigate between screens and all standard reports usually generated by the utility for the monitoring and control of the substation.

(*) Granja do Torto - Parque Tecnológico de Brasília - BIOTIC, Lote 4, 1º andar – CEP 70635-815 Brasília-DF– Brasil Tel: (+55 61)99284-2034 – E-mail: simoes@spinengenharia.com.br.

The same applies to wind farms, asset control, industrial substations, power plants, etc. Once a base application is generated, each feature is tested individually, as well as, every screen animation, every event, every alarm, etc. After these tests, new applications are generated very quickly, without errors.

2.0 WORKING WITH COMPONENTS

2.1 Concept of component

The basic cell of the LA methodology is called the component. Thus, an empty SCADA software project requires that tens of hours be spent until an application is completed. An LA project, by default, already comes an initial component that creates the basic structure of an electrical systems automation project. This initial component establishes a pattern of screens, creates some templates and tags that allow you to automatically generate various reports for querying real-time and historical data, as well as have command windows common in automation systems such as opening and closing of breakers / disconnectors, increase and decrease transformer tap, etc. Figure 1, below, shows the default component of an application example, called the Spin Component, which contains part of an automation project with:

- Default layout of the process screens, with a header of reports and basic navigation bottons, a body to be added to the drawings, single-line types and a footer with the last alarms, date and time;
- Dictionaries in Portuguese and English of words used in the reports and in the messages of events and alarms type open/closed, normal/In Alarm, Raise/Low, etc. This way the project can be used in both Portuguese and English;
- Reports of: (1) Current alarms; (2) Events of the day, (3) Operation Log, (4) Project variable tags that meet a filter; (5) Historical events that meet a filter; (6) Historical measurements that meet a filter; (7) Real-time or historical trends that meet a filter; (8) Filter construction form; (9) Window to insert other reports built in the application; (10) Report export window for Excel;
- Scripts for initial system load;
- Tags and templates used in basic system reports.

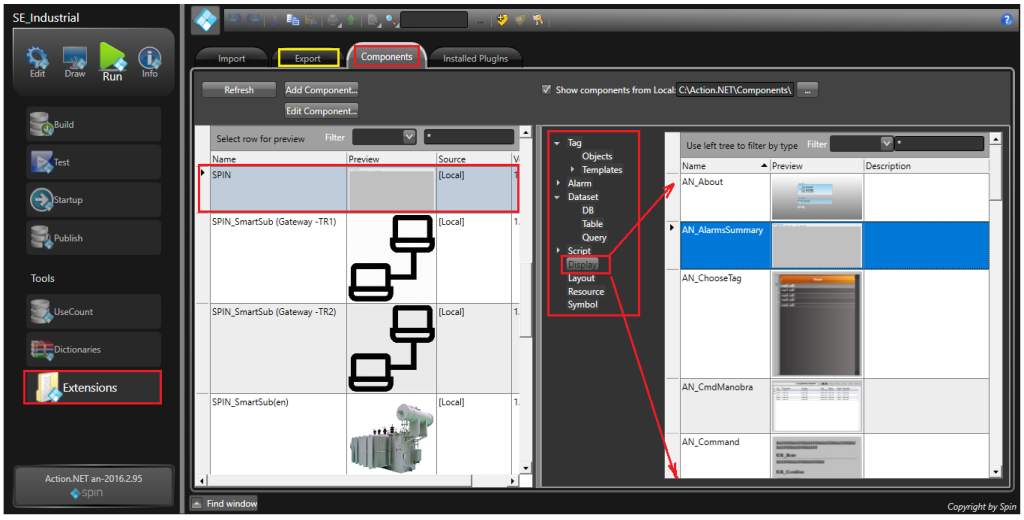

Figure 1 – Default component added to an empty project

- Creation of hierarchical default structure of objects;
- Definition of categories that associate objects with one or more functionalities such as historical or not, calculates current unbalance of two- or three-phase circuits, etc.;
- Project default security with types of users, their permissions and policies applied;
- Annotation windows associated with equipment, windows of equipment maneuvers, etc.

This basic component gives a common structure to any application developed, and can be designated as part of the company culture, its process automation methodology. If you want to change this or any other component, simply add it to the project, change it and export it, using the yellow highlighted tab in the figure above, creating a new component. This functionality gives a new dimension to the use of components, that

is, from a component with dozens of features the user can copy and modify it, creating a new component or more suitable to its culture or better than the previous component.

2.2 Components and the best practices

A manufacturer or a consultant to set best practices for using any IED such as a new relay line can use a component. Thus, the developer uses all the features available in the equipment to get the most out of it, and thus sets this knowledge in the component that should be used in dozens of projects, ensuring the propagation of best practices for the use of that equipment in new projects.

Considering electrical asset management solutions such as transformer monitoring [1], today there are IEDs from different manufacturers that are been used in a decentralized network [2] to monitoring the substation transformers and many of these IEDs are already sold embedded in the transformer. In addition, many relays used to protect transformers also accumulate features of monitoring the active. Using this basic idea, for example, the "Asset Monitoring" component was created that is generated from the inclusion of several components where each of them can be either a manufacturer's IED, as shown in the figure below, as a protection relay that accumulates asset management functionality.

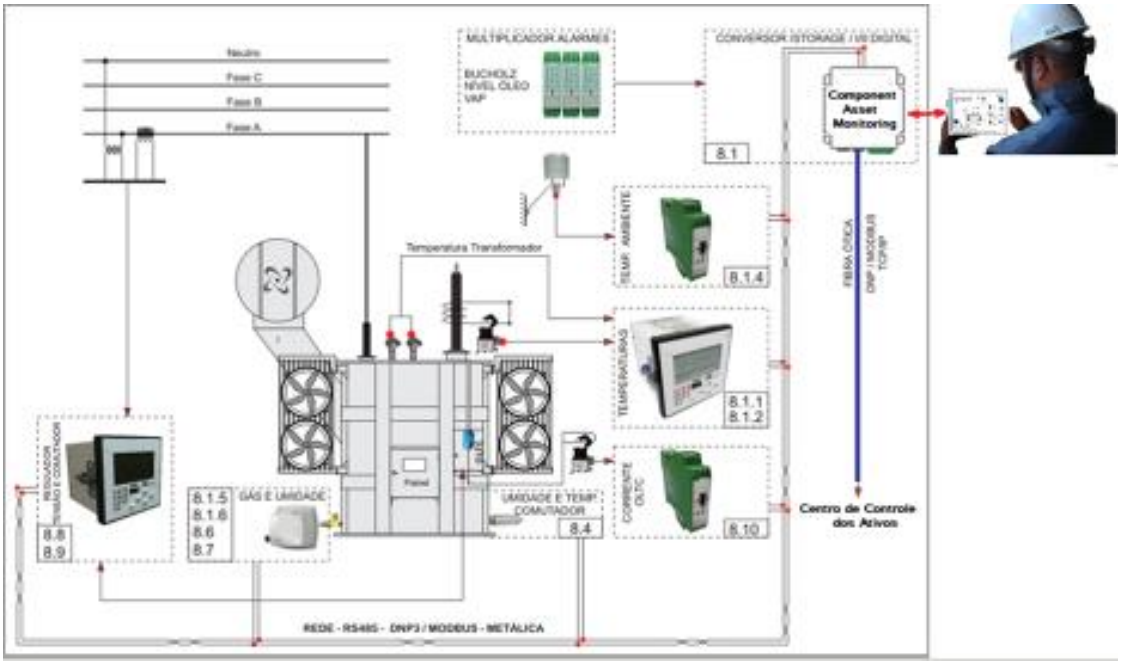

Figure 2 - IEDs from various manufacturers monitoring the transformer

This component is only an example, but the idea is to create a library with all the IEDs normally used to monitor the transformers of a utility and, according to the IEDs existing in a given substation, in a few minutes the application is generated which collects the data of these IEDs and presents them in local screens, when there is someone present in the substation, as well as send them to the central system, where the monitoring of all the assets will be done.

The use of components in the context of best practices, besides ensuring the creation of applications with quality, without errors, in record time, allows all knowledge of the best use of IED applied to the solution is in the component and not in the head of technicians.

2.3 Developing applications with components

Developing a component requires more time than a similar application because you must create a set of reusable features and you should define a somewhat more complex solution architecture. As for the homologation tests of the component, these are similar to those of an application, since both must guarantee the perfect operation of the solution, but in the case of components, once tested can be reused dozens of times without need for a test. That way, in a short time you will be more productive, more assertive, more competitive. To exemplify this behavioral change, the following is the development of a new application requested by a customer and how the vision of developing through component has changed the perspective of the company's solution. The client in this case has nine automated sites with this software, five wind complexes and four hydroelectric plants, and requested that the nine solutions be added to the reports on the availability of electric assets and, for this purpose, a single generic component was created that analyzes the availability of electrical assets. This same component has been added to nine different applications,

generating reports and dashboards that follow the same pattern. The figure below shows some dashboards and reports generated by the component.

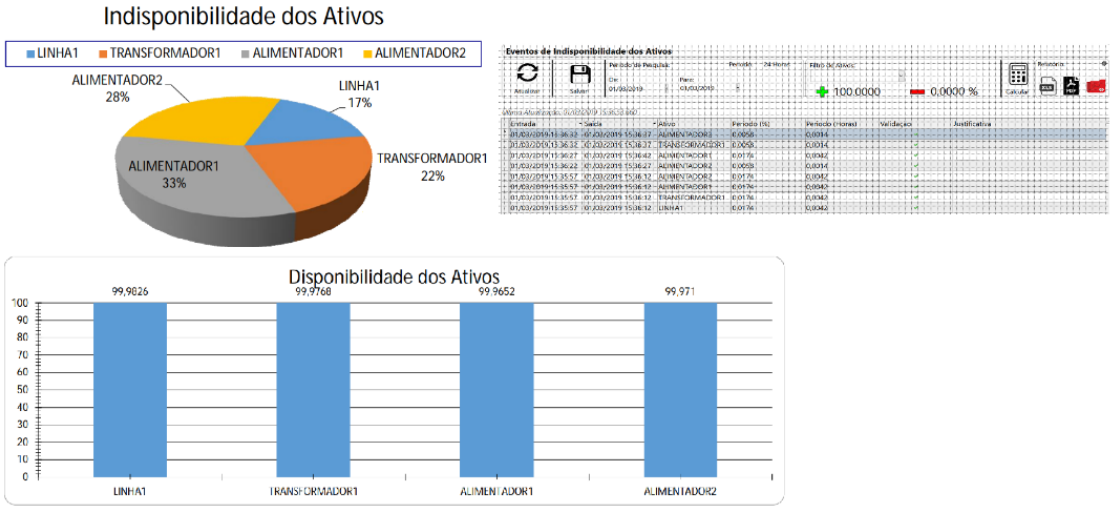

Figure 3 - Availability of electrical assets

Changing and testing an existing application by adding the availability report for their assets in all bays lasts less than four hours, and reports can be generated by shift, day, week, month, and year. This gives a substantive productivity gain, and manpower used is from an trainee, who does mechanical actions, from the component documentation.

After completing the development of this component, the company can use it at any other electrical site (power plant, substation, wind farm, etc.) to analyze the availability of the assets, obtaining standardized reports.

2.4 Automatic generation of applications

After the presentation of the concept and how to use the components, we will see how to create component libraries and how to use them to generate different applications in a few hours, with quality, robustness and low development and deployment costs.

The base project, called the default project, is no longer an empty project, since it has a first component that contains a set of basic tools of an electrical application, according to the authors' experience.

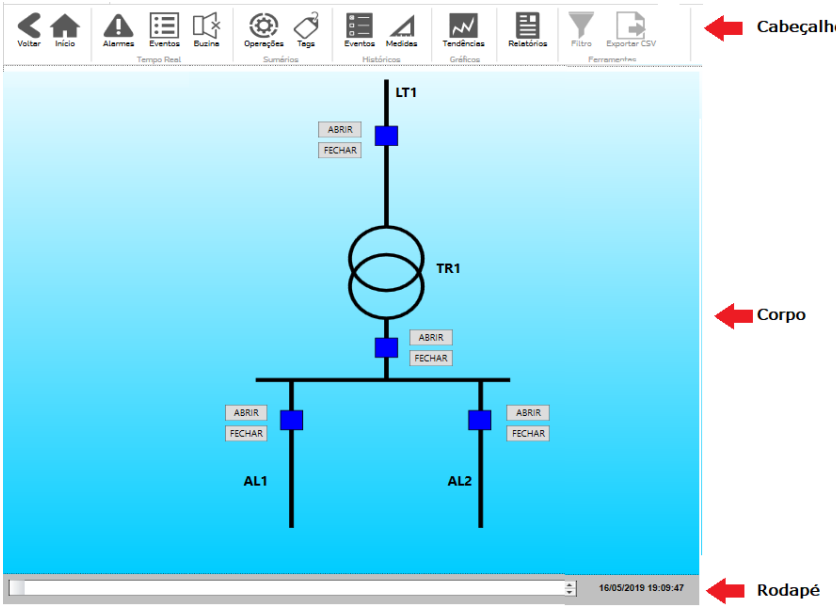

Figure 4 - Basic Design Screens Template

In this project, we have the model of all screens, with a header, a body and a footer where:

- The header has basic navigation rules (return to the previous screen and main screen), a set of real-time and historical reports, a generic filter applicable to any report, a beep mute button, and a button to export a selected report ;
- The body has process screens or the selected report body;
- The footer has the last alarm and, if we click on it, the last "x" alarms will appear. It also has the current date and time.

In addition, there are windows with all possible commands associated with the user's culture, annotations so the operator can comment actions like, why he blocked the command of an equipment, rule of naming of tags, library with all types of visualization symbols usable in the user's culture (Bay type with all the functionalities already available), applicable types of alarm, rules to send events, alarms and measurements for history, etc.

Observe that all these elements are modules called by reference, that is, if a user clicks on the symbol of a circuit breaker, to execute a command, the tag name of the circuit breaker is sent by reference to the command window and its template is, automatically, mapped to that device, that is, the command window serves for all equipment of that type, as well as the details screens available. For example, if I have a detail screen of a feeder, this screen is unique to all feeders and when navigating to it, the selected feeder is passed by reference turning the screen into the detail screen of that feeder. Below, by way of example, a detail screen of a one-component feeder containing the automation of substation automation of a utility.

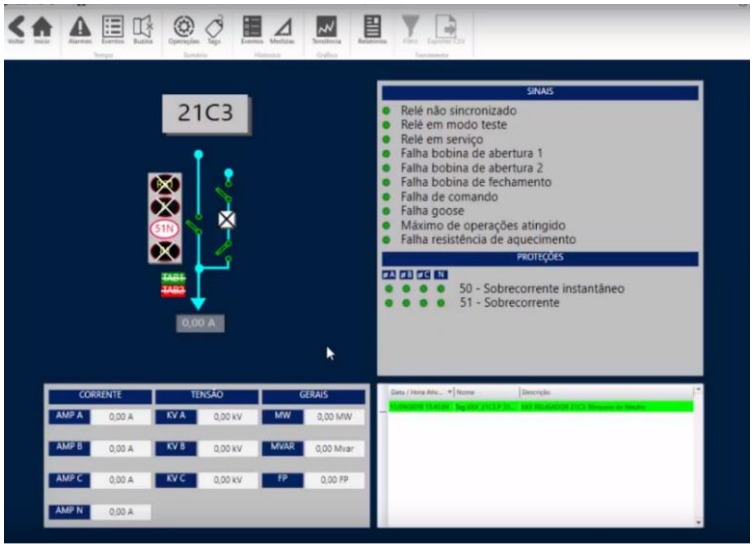

Figure 5 – Detail screen of a feeder according to the standard of a utility

Since the default project is a component, it is possible to modify it to fit the culture of different concessionaires.

On a default project, other components are added that allow the creation of different SCADA applications, such as a utility substation, a wind farm, an electric asset availability module, etc. From this concept, the software itself is already prepared to import components from a library, that is, when installed a directory is created that should be used as a repository of components, as shown in the figure below.

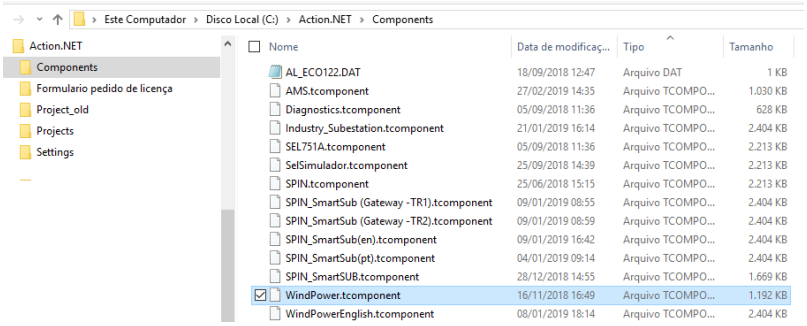

Figure 6 – Components Repository

Authorized users can access a library of components in the cloud and download one or more components and their documentation, as well as upload components developed by it, with its documentation.

To preserve the copyright of sophisticated components, it is possible to generate them as "Plug-Ins", which are non-editable components, that is, you can use them in your application, but you cannot see or modify your code.

Thus, with the use of component and LA methodology, the paradigm of developing SCADA applications is changed. An application can be created in minutes and over time users will produce extremely sophisticated applications with the guarantee of quality, robustness and zero errors.

3.0 WORKING WITH EXTERNAL LIBRARIES AND COMPONENTS

With the use of components, we show how it is possible to create a software factory for the generation of electric applications of SCADA systems with low cost, high quality and without errors.

The next step toward robust solutions that reduce the cost of production with quality assurance is the addition of external libraries whether they are "open source" or purchased. The DotNET environment allows to add external libraries (DLLs), as exemplified in the figure below, where three open source external libraries were added, the first one that allows the use of georeferenced maps of different manufacturers, the second one that allows to develop three-dimensional screens and the third , which is the EPRI library called the Open Distribution System Simulator (OpenDSS). The latter is a simulation program for electrical systems of electric power distribution. OpenDSS is able to perform new types of analysis that are necessary to meet future needs related to Smart Grids and many of its features were originally developed to support the needs of analyzes in which there is distributed generation (GD).

These libraries, however, are a set of code, properties, attributes, and methods that should be appropriately used to be useful in the SCADA configuration. For example, the GIS map shown below was used in a client to map the distribution system switches and reclosers. To use the map, it should be possible during the customization time to insert georeferenced switches on the map, with properties that allow the real-time state, its main measures, to draw the electrical networks between them, as well as to create methods that must command them.

In the case of external libraries, the component comes as a response to integrating libraries external to the SCADA. Explaining better, the component is the part of the project that not only includes the library external to the system but also manipulates its properties and methods in the context of SCADA, making it an object customizable by any client that wants to use it.

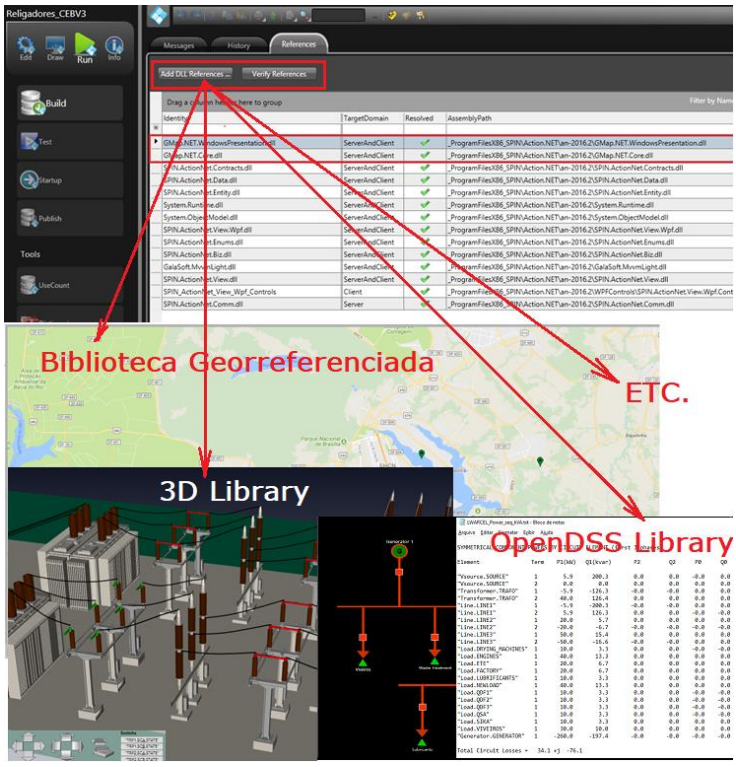

Figure 7 – Inserting external libraries through components

Continuing the previous chapter, through the methodology, expands the functionalities of the software with the new libraries that are already ready to be used as a component and if in the future the user decides to use more features of the new library, just change the component for such. Likewise, if new versions of the library are released, simply incorporate the new features into the existing component.

The integration of Open Source libraries into SCADA software may be a shortcut to the Smart Grid [3].

4.0 USING EXTENSIONS

Finally, one last tool to automate the generation of real-time solutions is the use of extensions, which are software modules developed in C # or VB.NET, used in parameterization time to automate the generation of the real-time database and historical, as well as facilitate the parameterization of the application. Below are some examples of the use of extensions already developed, which were used to accelerate the parameterization of SCADA applications:

a) Since several users frequently used a certain PLC (could be a UTR, an IED, etc.), an extension was developed that reads the configuration file of this PLC, and automatically generates the templates with the variables tags defined in it, creating a naming rule for them. This was done with the Allen-Bradley ControlLogisx PLC.

b) In order to replace old SCADA applications with the new SCADA, an importer has been developed that reads both the old SCADA database and the existing screens and generates an application in the new software, with guarantee that all points already commissioned will be generated correctly, already that the addresses are copied.

c) As the IEC-61850 and OPC object-oriented protocols are often used, two extensions have been developed that read the IEC-61850 IEDs and OPC servers, respectively, and generate the real-time database. In the case of IEC-61850, the SCD file can also be used as a way to import the data.

d) Since certain users made extensive use of the OsiSoft PI database, an extension was developed that imports the data from this base at design time and allows in real time to use this data to generate reports and Dashboards.

e) In the use of a FLISR (fault location, isolation, system restoration) algorithm in a SCADA application, there is a first program that generates a multi-loop distribution network that will be monitored by FLISR and populate its data from system data GIS of the concessionaire. At the end of the program, you have all the data in a relational database that has not only the distributed network that will be controlled, but also the screens represented by the loops. An extension has been developed that not only imports all data from the distribution network to the SCADA, but also generates the screens with the loops that will be controlled [4].

f) If the user always uses the same worksheet in a standard format, in order to enter the data of all his tags, with addresses, operational limits, events, alarms, etc., he can develop an extension that reads this worksheet and generates its base data, events, alarms, history, etc.

g) In the case of the Lean Automation methodology, an extension has been generated that starts from the defined templates automatically generates all the alarms / events of the application, as well as the conditions of recording historical records and the list of addresses of all the points.

Generalizing, with the extension functionality, whenever there is a standard used many times to generate the SCADA database, it is possible to develop an extension that automates this process, minimizing the possibility of errors and reducing the time of configuration and testing.

5.0 CONCLUSION

in this work we have shown a new trend applied to software type SCADA which is to provide a set of tools in the software that allow to create pre-ready components that, like a game of Lego, allow to automate the process of generating SCADA applications, connecting components. In addition, the methodology establishes the technologies for the use of IEDs (best practices) to software, allowing it to stay in the company and not in the head of its technicians.

Following the development of technology, the next step will be the distribution of component libraries that will allow users to build and test new applications over hours instead of several days.

In addition to the automation of application generation, third-party libraries, whether they are "open source" or not, can be added to the solution and, with the use of the components, mold that library to operate within certain limits, time, and the obligation to study the external library to use it, that is, the component provides

the methods for using the library as if it were a feature of the SCADA software.

Concluding the process, we added a last tool called extension that allows to automate the import of data from sources widely used by a set of software users.

The end result is increased productivity of SCADA system integrators with cost reduction and design time.

6.0 BIBLIOGRAPHICAL REFERENCES

[1] Silva. C.A, Silva Junior A.P and Borges. R.L. “Development of Specialist System for Transformer Management Installed in Substations”, Cigré-Brasil, “IX Workspot - International workshop about transformers and substations equipment’s, 2018.

[2] Rampazzo, W.A, Teixeira, R.C, and Amorim, F.G.A, “Decentralized Solution for Substation Digitalization Using the DNP3.0 Protocol”, XVII SENDI, Olinda, jan/2015.

[3] Silva, T. P, Simões, C. “Open Source Tools Integration in SCADA Software - A Shortcut to Smart Grid”, Cigrè Latino Americano, XVIII ERIAC, Foz do Iguaçu, Maio 2019.

[4] Simões, C., Porto, J.A.S.B., Kagan, H. Pelegrini, M.A. and Guaraldo J.C. “Implementation of FLISR Algorithm”, Cigrè Latino Americano, XVII ERIAC, Foz do Iguaçu, Maio 2017.

7.0 BIBLIOGRAPHICAL DATA

Clovis Simoes, borned in 1951 - Curitiba-PR/BR. Graduated in mechanical engineering, UFRGS, 1973 and MSC in Computer Science, UFRGS 1977. He is a founding partner of Spin Canada, having participated in the development of four SCADA software, two already in Spin (ActionView [EMS / SCADA] and Action.NET [ADMS / SCADA]). Participated in dozens of automation projects for power plants, substations, wind complexes and generation and distribution control centers. He has written and presented dozens of papers at national and international congresses on software development and automation of electrical systems, paper machine and propulsion control and ship failure. He was a professor at the University of Brasilia and UFRGS, in the area of computer science.