SCADA SOFTWARE

«User Level» mode becomes a necessity

To modify a SCADA application, there is little choice but to use the development environment provided with the software. These tools, however, are designed for people who design applications from scratch rather than the technicians who maintain systems on site. There is hence a need for a customised development environment for each type of user. Whether the user is a programmer, an integrator, a maintenance engineer or an operator, everyone can benefit from how the information is displayed; with the additions of reliable applications and time saving in deploying them.

SCADA software products are provided with their own development environment. It is through these that the end users and their integrators develop their applications. They define functions that the operator will use and the ergonomics of the final application. No matter how much care is taken in developing an application, it will always require some sort of maintenance. This could well be the addition of new features or simply modifications to the existing ones (for example: the addition of a new piece of equipment). To make these changes, an end user requires the service of either an internal trained resource or a specialist system integrator. He could use his integrator for major changes to the application, however for small changes such as replacing a single piece of equipment for example, he would normally use his internal resources; this is when problems begin…

Essential

- The analysis of the SCADA market has shown that end users were confused by the complexity of a product's development environment.
- It is useful to reduce the functionality of a development tool for users who don't have advanced needs.
- The use of a "user level" mode guarantees a faster and more secure application navigation.
- An application developer has to continue to have access to all aspects of the development tool.

From the expert to the user: different skills

Industrial SCADA applications have long lives; these can often be 10 or more years. When changes have been made over the years by different people, even the most experienced developer or integrator will have some difficulties to overcome. It is a well known problem in IT that the more changes that are made to software, the more entrenched and even hermetic it becomes, which impacts its ability to be maintained.

An operator or a maintenance engineer can thus meet difficulties when making even the smallest of changes to an application. Simply training this person in the use of the base product doesn't make it any easier to modify the application and the idea of them "entering" into someone else's design is quite daunting. To do this at low risk, the user requires a total change of environment and even "vocabulary". A SCADA product will show its base objects: variables, alarms, logic, formulae, alarm windows, alarm history, etc., when the user only wants to add a valve, a pump or a tank.

Moreover, the large number of functions, objects and parameters makes it harder for a new user to find what he wants and to understand what people before him have designed. The principle of a tree structure for example can contribute to making the development tool difficult to access. Even for a simple problem such as adding a new sensor to a production line: the user will wonder if he needs to select a "new component", "new object" or "new object instance"… He can look for existing sensors; the number of properties, options and the possible choices may take him aback. The excess of information usually confuses a non-experienced user. Despite this, product suppliers keep on providing unique development environments, which are their standard development tools, despite the level of expertise needed to use them. The need for a "user level" mode is thus plain to see; one which allows not only the integrator to freely design a new application but one that allows an operator or a maintenance engineer to make changes using a different set of skills. Of course, from a product editor's point of view, they could provide a specific development environment for making only "small changes", this however would be even more confusing for the users and would lead to a higher costs of their developments which would in turn be passed on to their user base. It is possible, though, to change the way in which the information of an application is displayed such that a single development environment can be understood and used by any type of user. It is not a question of changing the way application designers work because their job consists of developing new applications and they understand the software; it is making the simpler tasks less deep-rooted for those who aren't designers.

In "developer" mode, on the left, all system folders are displayed, as well as the contents of the different components. Navigation is only done through the tree structure. However, in "user level" mode (photo right), the system folders are not displayed and only certain properties are available. Navigation is directly via a list of user objects. Depending upon the needs, "user level" mode can be reduced to the creation of new views (mimic diagrams that represent the current installation) or new object instances. When a pump needs to be added, for instance, it's a question of selecting a pre-defined pump model and then completing the parameters related to the equipment in question.

Rethink the nature of functions

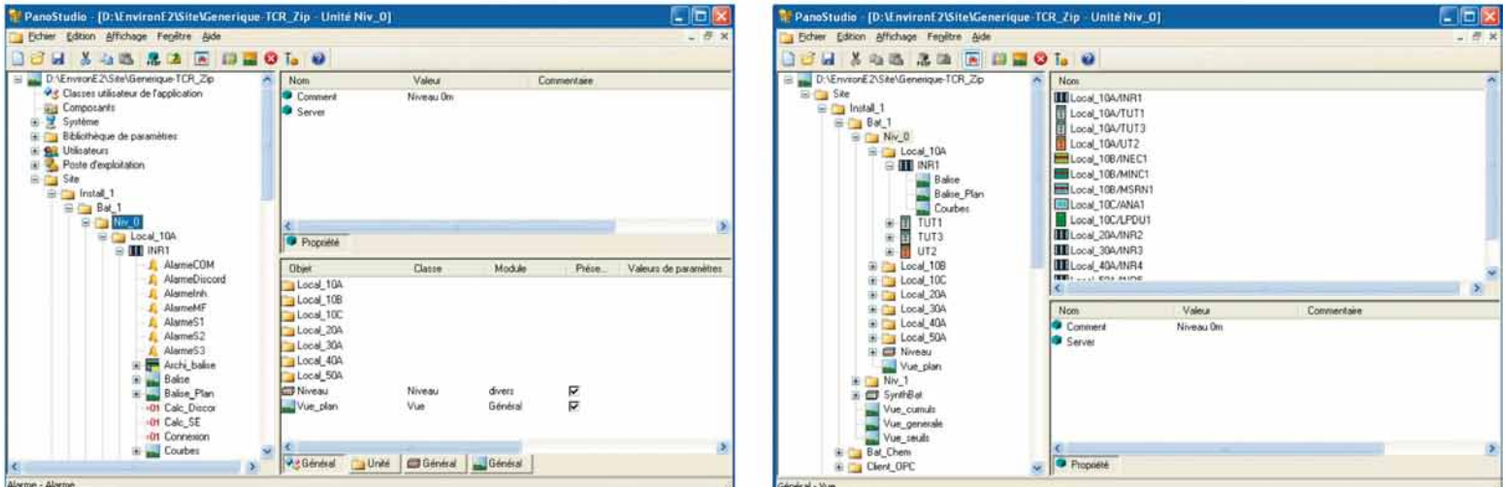
Let's take the example of an end user who wants to increase production. If he wants to base his shop floor on an existing model, he will hesitate to make a call his usual integrator to develop the application. A "copy and paste" should be enough, he might think. However for the operators who have to make this change, it won't be that simple. There are so many menus with parameter names that can be difficult to follow, so to establish a connection with new equipment can be difficult. A solution to this consists of offering different access to the development tool: an "developer" mode for users who develop application functions and a "user level" mode for users who only make modifications. To illustrate this point, let's make an analogy with a PC constructor. It is possible to build a PC from basic components (Microprocessor, memory, communication interfaces, etc.). A PC can be created and customized at chip level but it requires an expert to do so. The other method is to choose pre-made components off the shelf, attaching different modules to the motherboard and putting it together in a PC frame. To do this requires no expert knowledge of electronics. This is the principle of the user level mode. It is important to note that the current tendency is to provide "vertical" solutions (made simple for a specific task) to facilitate the task of an integrator. The availability of a certain number of pre-configured functions already saves application development time. From his point of view, the user will not see any difference: the designer's tool is still the same and it will be just as difficult to make modifications at this level as before.

An expert mode and a user level mode

Studies have shown that for restricted functionality, a user level uses on average only 5% to 10% of the information and of the development environment functionality. Once an application is delivered to site, it is then possible to mask less frequently used information in

Three approaches to SCADA

In order to install a SCADA application, several methods are possible.
The most basic approach consists of using programming languages and "classic" development environments. Any specialist application development company can do this, providing that they have the knowledge of industrial communication protocols.
The second approach is to use dedicated software products. This approach is used for very specific end user applications (flour milling, livestock pellet production, etc). Configuration is often done through tick boxes. However, even if the people who conceive these products are specialists in their areas, in the main they are neither control engineers nor computer scientists. That is why these products are more difficult to change or update (as it requires a call to the developer each time a new function is needed).
The third approach consists of employing generic SCADA software. Created by and for control engineers, these are widely used in industry because they communicate with many different types of equipment. The term "generic" doesn't mean they can do anything you want. Limitations related to the languages used make these products more or less attached to their original area (nuclear, in the case of Panorama P²). Fortunately over the years, product suppliers have started to adopt object programming techniques. This concept allows much more freedom to develop diverse applications. Using an object oriented tool, anything is now possible. The only one negative point being that it brings complexity for application developers who are not familiar with object programming techniques.



The "user level" mode makes the creation of objects simpler. We can see on the left the menus available in "developer" mode and on the right the menus available in "user level" mode, which only relate to "user" objects.

order to gain time. Operators will also be more autonomous and they won't need to call their system integrator to do simple jobs like adding one new object.

With the use of a user level mode, everyone can use their skill to the best effect. We can have software engineers and developers who create models using objects on one side and operators and maintenance engineers who can use those models on the other side. The former will be grateful not to have to learn brands or models of equipment and for the latter, there is no need to learn the deep object programming.

Security improvement

Besides the ease of use, there will be a clear separation of responsibilities regarding the development. We should keep in mind security and availability of the application. If end users learn to use the development tool, the production must not be stopped. In most cases SCADA software is used to control complex equipment. The supervision of critical applications and continuous process generally must not be interrupted. Furthermore it is rare that a production line functions outside a precise set of parameters. For all these industries, it is necessary not to upset production at the slightest modification (e.g. a sensor replacement). Certain SCADA software products allow the delivery of an "encrypted" application where the system integrator supplies an application in the form of an executable (that can't be viewed through the development tool); but this prevents any future evolutions being made to it. By offering a separation of responsibilities, there would be a reduction in the risks of errors while improving the range of actions and the reactivity of the end users.

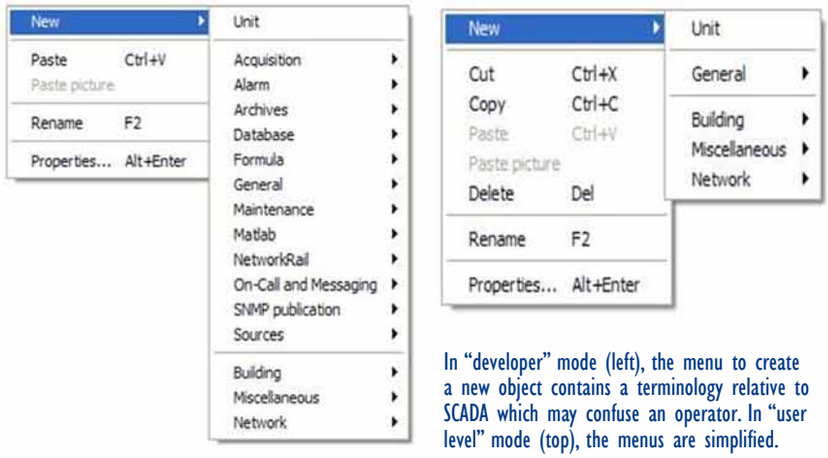
It is equally possible to reduce development time to deploy an application. We have seen that with a user level mode, standard objects and instantiated objects are developed by different teams, thus it is also possible to go on step further. By developing a "reference application", the time savings will be even greater. For example, a typical SCADA application is made up of a group of models with their instances and related parameters. This is used by those who wish to make a new deployment. The principle of referenced applications consists of keeping only the models in an application in order to make this deployment simpler. The operator will then use a runtime application that will contain only object instances and their parameters.

With this concept, product suppliers can provide user level modules via a reference application, instead of selling a solution that is based upon a library of functions, they will be able to provide complete factory elements. It will be necessary only to complete the equipment references and the application will be ready to run. If we take an example in the water industry: an operator will be more interested in the total flow of a pumping station as opposed to the number of pumps it has. He wants to easily connect this flow to the value of a level found in a monitoring function. To help him, the reference application will contain pre-configured parameters (the

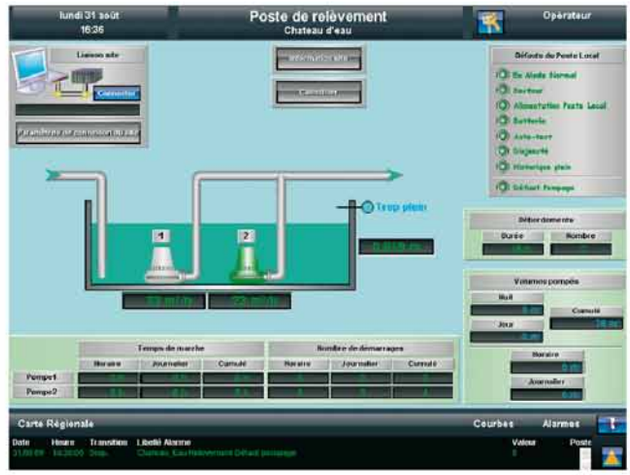
Native "user level" mode

Codra now provides a development tool that is highly adaptable to how it is used. With Panorama E², the application designer has access to every function, but the operators and the maintenance engineers can use a different "user level" interface oriented to their needs. Offering an adaptable configuration is not entirely new. Several software products already available on the market today allow the creation of configuration files using Excel. These files will then be imported into the SCADA product. However, until now no product company has offered an "open" method of configuration. The designer can now choose what information (objects, functions, parameters, etc…) that will be available in "user level" mode. This will improve both the simplicity of use and the security of applications. This "user level" mode has been provided with Panorama E² v3.01 and this release is fully compatible with older applications.

Another illustration of differences between user modes: Context menus can be adapted to the user's needs.



In "developer" mode (left), the menu to create a new object contains a terminology relative to SCADA which may confuse an operator. In "user level" mode (top), the menus are simplified.



The objective of a "user level" mode is not to simplify the work of those who already know the tool. It is done to make the same tool available to those who are unfamiliar to SCADA.

pump and PLC references). He will only need to connect the external references to the pumping station object.

And why not have other "level" modes?

The use of a user level mode and reference applications allows for more advanced products in terms of their ergonomics and their adaptability to a specific use. Through the use of software assistance, for example, we can plan a solution that only requires the elements to be cabled. When a new machine is installed, the software connections at a SCADA system often take as long as the equipment cabling. To connect each object to the related sensor or motor, the operator has to find its related data amongst a large choice of data items. Using "new generation" cabling assistants, the software is able to guide the user by associating each object model to the correct data. When the operator selects a "pressure detector" object, then to connect to the physical detector, a list of available data items related to only pressure will be offered to the user. This type of assistance will simplify the work of certain types of users, without changing the job of a developer who will continue to use the advanced features that he needs. Finally, we could even imagine that the arrival of a user level mode can bring about more economic ways of working. Effectively, product suppliers or any 3rd Party can provide "vertical" solutions thus capitalising their development efforts. These "plug-in" solutions can be sold directly to end users who would no longer need to use a system integrator.

Jean-Claude Hallynck
Codra Technical Director
and Frédéric Parisot