ProcSee

Graphical User Interface Management System

Technical Overview

Abstract

ProcSee is a versatile software tool for developing and displaying dynamic graphical user interfaces (GUIs), particularly aimed at process monitoring and control. ProcSee is a mature, high-quality software product used by organisations worldwide. ProcSee-based GUIs have been implemented on various screen sizes ranging from small handheld devices, via traditional operator screens to large wall-mounted overview displays.

ProcSee supports object-oriented definitions of dynamic GUIs, enabling the designer to visualise process states. Graphics, dynamic behaviour and operator dialogues are defined using an advanced GUI editor, and any aspect of the GUI can be linked dynamically to process parameter values. To support definitions of the dynamic behaviour, the GUI designer is provided with a full-featured programming language called *pTALK*. pTALK has the syntax and expressive power of C++, and additionally includes constructs for graphics manipulations. Thus, ProcSee provides GUI designers with a unique flexibility in building highly customised dynamic operator interfaces.

The ability to switch quickly between design and test modes makes ProcSee an excellent fast prototyping tool, supporting an evolutionary development process.

At run-time, ProcSee visualises dynamic GUIs on the operators' screens, updates graphics according to the dynamic behaviour whenever new process values are received, and handles operator input according to the designer's definitions.

ProcSee can be connected to process data acquisition systems, simulators and real-time databases using an open, high-level application programmer's interface (API). ProcSee can also act as an OPC client, reading and writing values from the process' OPC servers directly.

A versatile trend-system for logging and visualising historic data is fully integrated with the ProcSee run-time system and editor. The trend system provides functions for logging and displaying user-defined events as well as traditional trend curves with highly configurable presentation modes.

The ProcSee run-time, the OPC client, and the API are all optimised to handle large amounts of data and frequent display updates. ProcSee handles overlapping graphical objects automatically and provides flicker-free display updates using highly optimised algorithms.

Platforms

ProcSee runs under Microsoft Windows and various Unix platforms including Linux, HP-UX, Solaris and Mac OS X.

Why Use ProcSee?

Developing user-friendly GUIs for process monitoring and control requires advanced software tools. The tools should support the developers throughout the application's lifetime, from the initial prototyping to the final maintenance phase. ProcSee faces these challenges by offering an advanced, interactive editor and a ready-to-use run-time module for visualising the graphics and handling operator interaction. Dynamic behaviour and operator dialogues can be tested immediately while designing in the editor, and the editor can be connected to a running GUI at any time to edit, test and correct it.

Developers of large-scale systems will appreciate that ProcSee can also be configured using text input files. For instance, variable definitions and specifications for the logging of historic values are normally specified using a text editor or by some automatic generator program. In fact, designers can convert the entire GUI to textual format, modify any aspect in a text editor, and regenerate the GUI using tools from the ProcSee toolbox.

With its unique flexibility and performance, ProcSee is particularly attractive to simulator or SCADA system suppliers who can establish generic GUIs for easy configuration according to their customers' needs.

Main Features

- Advanced interactive GUI editor
- Highly flexible definition of customised objects with dynamic graphics and operator dialogues
- Easy integration with process data acquisition systems, simulators, and real-time databases
- Optimised for large amounts of data and frequent display updates
- Highly configurable historical trend data log and curve display system
- Supports immediate testing of GUI behaviour
- Platform independent with cross-platform data communication

References
ProcSee has been used for process monitoring and control in simulators and online applications for nuclear power plants, oil production platforms, electric power production and distribution, telecommunication networks, ship bridge systems, ship engine systems, paper mills, and environmental monitoring. For an extensive list of references, please visit ProcSee on the web.

System Architecture

Figure 1 describes the ProcSee system architecture and the roles involved in developing and using a ProcSee-based GUI.

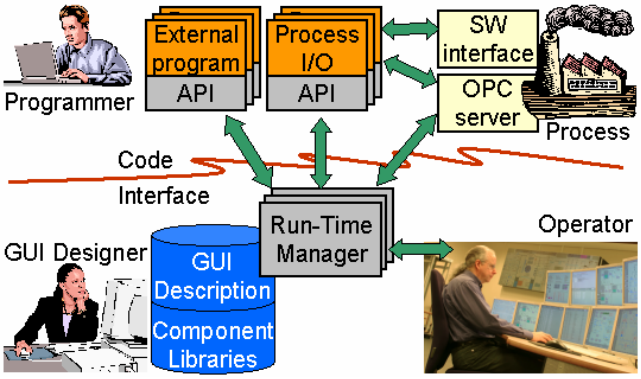

Figure 1: ProcSee system architecture.

GUI Editor
The GUI designer implements the GUI using the editor, and the result of his/her work is a set of files containing the complete GUI description, including re-usable component libraries.

Run-Time Manager
The Run-Time Manager (RTM) is the heart of ProcSee. The RTM is a ready-to-use executable that reads the GUI description at start-up, visualises the GUI on the operators' screens, receives variable values from the process and external programs, updates the displays according to the current values and states, and handles operator input according to the designer's definitions.

Process
The process may be a plant, a simulator, a real-time database, previously recorded data from simulations or anything else providing a set of variable values that changes over time.

There are two alternative ways to transfer variable values from the process to the RTM:

1) Configure ProcSee's integrated OPC client, provided the process includes an OPC server
2) Implement a C/C++ program to extract values from the process and send them to the RTM using the ProcSee API

Process control commands from the operators are routed the opposite direction.

External Programs
External programs may be complex computer codes and various operator support systems providing additional data to be visualised by the RTM. It may also be GUI applications integrating ProcSee displays into their own user interface.

Separation of GUI and Code
ProcSee's system architecture leads to a clear separation between GUI and application code, which has proven to be very useful when it comes to long-term maintenance.

Technical Overview

ProcSee-based GUIs are structured according to object-oriented principles with classes and object hierarchies.

The object hierarchies provide a structured way of organising the thousands of objects possibly involved in GUIs. At the top of the object hierarchy is the GUI itself, and lower levels include items like windows, pictures, classes and libraries as well as functions, attributes and dialogues. Within a picture, there may be basic shapes, instances of complex classes, and picture-specific functions, attributes and dialogues. A class may consist of basic shapes, instances of other classes and class-specific functions, attributes and dialogues. This way, the entire GUI is well organised and easy to browse.

Implementing a GUI Using the Editor
The ProcSee editor, shown in Figure 2, provides functionality for defining all aspects of the GUI, not only the graphics. Using the editor, the GUI designer builds generic components with dynamic behaviour, defines windows and window hierarchies, specifies resources like colours, fonts, patterns and cursors, imports libraries of reusable components, draws pictures, connects picture objects to process data, and specifies how operators shall interact with the GUI and the process.

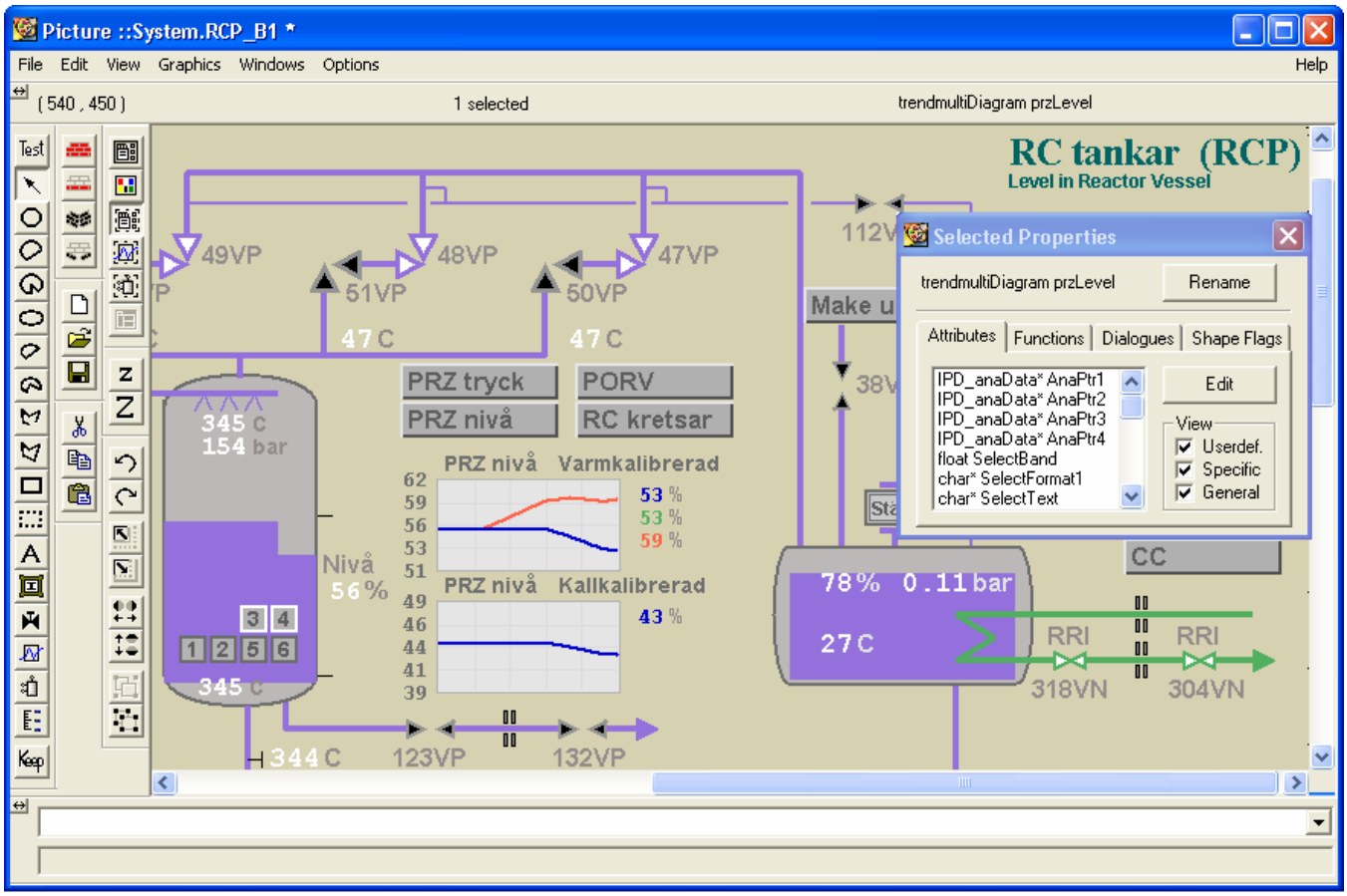

Figure 2: The GUI editor. Attributes, functions and dialogues anywhere in the object hierarchy can be viewed, set and modified. Switching between design mode and test mode is done simply by pressing a button.

The ProcSee editor offers editing functionality similar to that of general-purpose graphics editors as well as more specialised functionality for implementing a complete GUI as described above. Functionality like cut and paste, snap and grid, aligning and distributing, rotation, scaling, flipping, grouping, panning and zooming are available from toolbars, menus and keyboard shortcuts.

Testing the GUI

Traditionally, user interface development is carried out by iterating in a two-step cycle. First, to use an off-line tool to specify the interface and somehow compile this specification, and secondly to run an on-line system that uses the compiled interface specification.

In ProcSee however, there is no distinction between off-line and on-line. ProcSee's editor operates on-line, enabling the designer to immediately test the GUI while designing it. Dynamic behaviour as well as operator interaction can be tested. Testing can be accomplished by modifying ProcSee's copy of the process parameter values from the editor, by connecting some temporary data generators, or by connecting to the real process or simulator.

In addition, the ProcSee editor can be connected to the GUI for inspection, testing and modification at any time, even at run-time with the real process connected.

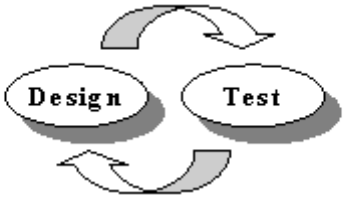

Figure 3: ProcSee offers a development environment where the GUI designer can test the GUI while designing, without recompiling or restarting the system.

pTALK and the Virtual Machine

The heart of the ProcSee RTM is a virtual machine able to compile and execute statements in the ProcSee language, *pTALK*. pTALK has the syntax and expressive power of C++, with additional constructs supporting graphics manipulation. pTALK statements can be transparently compiled at run-time, allowing source code to be added and executed at any time, even at run-time.

pTALK is used in all aspects of the GUI, most frequently to specify dynamic behaviour and operator interaction. With more than 300 standard functions available and the ability for the GUI designer to create his/her own functions as needed, pTALK's expressive power provides a unique flexibility for building customised GUIs.

Because of pTALK and the virtual machine, ProcSee-based GUIs can be modified at run-time. Using standard API and pTALK functions, application developers can add, modify and remove graphics at run-time. In fact, this is the key feature used by the ProcSee editor. It is actually the RTM that visualises all graphics in the editor, the editor itself merely sends commands to the RTM to create, modify or remove objects. So, ProcSee is a powerful tool to implement tailor-made editors for specific purposes.

The virtual machine makes ProcSee-based GUIs platform-independent, running on any platform supported by ProcSee.

Basic Interface Components

At his/her disposal in the editor, the GUI designer has a set of basic shapes, each with its unique set of attributes to be manipulated. For example, rectangles have 17 attributes, including visibility, position, width, height, rotation angle, scaling factors, foreground- and background colours, line- and fill patterns, etc. Any of these attributes can be dynamically connected to process variables through arbitrary complex pTALK expressions.

Among the basic shapes available are rectangle, circle and ellipse (including pie and chord), polygon, line, text and image. Regarding images, most major graphics formats are supported. To insert dynamic graphics on top of static images is a fast way to implement nice-looking dynamic objects.

Other, more complex shapes are also provided. For instance, a highly configurable scale shape provides excellent functionality for annotating bar graphs and trend diagrams.

On the Microsoft Windows platform, ProcSee supports the use of COM components and ActiveX controls in objects and pictures. This way, GUI designers can benefit from the large set of such components available from various suppliers. ProcSee is shipped with a set of such controls for some commonly used objects.

Dynamic Graphics

To create dynamic graphics, ProcSee offers functionality for connecting graphic elements to process variables. The connections are specified as pTALK expressions. For example, the expression below connects an object's height-attribute to the level in a tank. Thereafter, ProcSee will ensure that the object is always displayed with the correct height according to the value of the expression.

```
height = `tank1Level.value * 50`;
```

In more complex cases, the GUI designer may create a function. For instance, to connect the foreground colour of a shape to a process variable may involve the following steps:

1. Create a pTALK function that returns a colour given a variable of a user-defined data type. In this example the type TagData contains three attributes: *value*, *hiLim* and *hihiLim*, and the function returns one out of three different colours, *colNormal*, *colWarning*, or *colAlarm* depending on whether the attribute *value* is greater than one of the limits.

```
int statusCol( TagData* td )
{
   if ( td.value < td.hiLim )
      return colNormal;
   else if ( td.value < td.hihiLim )
      return colWarning;
   else
      return colAlarm;
}
```

2. Connect the shape's foreground colour attribute to the value of the function applied for the specific process variable connected to the shape:

```
foregroundColour=`statusCol(RS101)`;
```

This way, any attribute of a basic or user-defined graphical object can be assigned an arbitrary complex pTALK expression, providing a unique flexibility for building customised GUIs. Whenever updated values are received from the process, the dynamic expressions are automatically evaluated by the virtual machine and, if necessary, the graphics are updated accordingly.

Operator Interaction

An important aspect of any GUI is the interaction with the operator. In ProcSee, the interaction is specified as *dialogues* that are fully integrated with the graphical objects themselves. Dialogues can be defined at different levels in the object hierarchy: at the GUI level, picture level, class level, or shape level. A dialogue has three properties:

1. *An area where the dialogue is active.* Most frequently, dialogues are integrated with graphical object in the pictures, implying that the area covered by the object is the active area for the dialogue. However, dialogues may also be attached to more global areas, like a picture or an entire GUI.

2. *A triggering condition.* The triggering condition must include one of the more than 20 predefined event functions, optionally combined with any pTALK expression returning a boolean value. Here are some examples:

```
buttonPressed(LeftButton)
cursorMoved()
keyPressed(AnyKey)
windowClosed()
shapeEntered() && myState==Open
```

3. *The action to be executed when the triggering condition is fulfilled*. The action can be one or more pTALK statements. Note also that external programs and process I/O modules may declare some of their functions to the RTM. So, dialogues may also invoke functions in external programs and process I/O modules, for instance to start a pump or close a valve.

User-Defined Interface Components

When designing user interfaces, it is important to be able to re-use interface components throughout a GUI and across related projects. In ProcSee, this is provided through the use of *classes*.

A ProcSee class is a collection of basic or user-defined graphical objects as well as class-specific attributes, functions and dialogues. The GUI designer can add functions and attributes of any predefined or user-defined data type to configure its use. A typical attribute is a pointer to a complex data type with the relevant data for the component. Dynamic behaviour is assigned to the individual objects making up the class to obtain the desired appearance of the class as a whole, and dialogues are added to handle operator interaction.

Once a class is defined it can be inserted into pictures in a number of *instances* by a simple point-and-click operation. An instance is a copy of its originating class and its attributes should be connected to the proper process parameter values as shown in Figure 4.

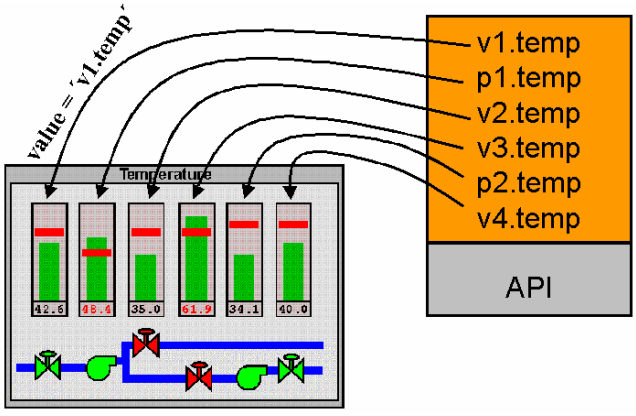

Figure 4: Instances of a bar graph class configured in a picture. For each instance, the class' attribute "value" *is connected to the proper process variable.*

Classes can be modified at any time, and ProcSee will ensure that all instances are updated accordingly.

Easy retrieval and re-use of classes are accomplished by grouping related classes in *libraries*. In addition to class definitions, a library can contain definitions of all graphic resources needed by the classes. These resources include colours, fonts, patterns, functions, and attributes. A library of classes and resources can thus be defined as an independent part of the user interface and can easily be re-used in a number of projects.

Historic Trend Diagrams

In addition to visualising the current state of the process, ProcSee can also display historic trend diagrams visualising the history leading up to the current situation. The trend system provides functions for:

1. *Logging data at user defined intervals*. ProcSee can log historic data in memory or on disk, and the logger is optimised to handle large amounts of data. The logger can also be used to store predicted data, enabling ProcSee to visualise results from predictive simulators.
2. *Displaying trend curves*. The logged data can be visualised as multi-colour curves in a diagram, and a diagram can present any number of curves simultaneously.
3. *Logging user-defined events*. GUI designers can define event types and related data to be logged when user-defined events occur. At run-time, requests to log events can be issued from the GUI itself or from external programs.
4. *Displaying events*. Logged events can be visualised in trend diagrams by user-defined objects that move in the diagram as time passes. Different types of objects may be used to represent different types of events, and the objects may visualise the data related to the event. Events and curves can be integrated into the same diagrams.

Trend diagrams can be included in pictures and user-defined classes, and there is no limit on the number of diagrams that can be visualised. Trend diagrams have a wide range of attributes that can be manipulated and connected to process parameter values just like attributes of other shapes.

Features for trend diagrams include time-labels, grid, ruler for exact read-out of values from the curves, panning to see values outside the current time-span, automatic and manual scaling of curves, linear and

logarithmic scale, various presentation modes and more. All features can be controlled by pTALK statements and can be modified at any time.

Trend curves may be multi-colour, open or filled, smooth or step-wise, focused around an offset value or presenting the difference between two logged variables. Figure 5 presents a few examples.

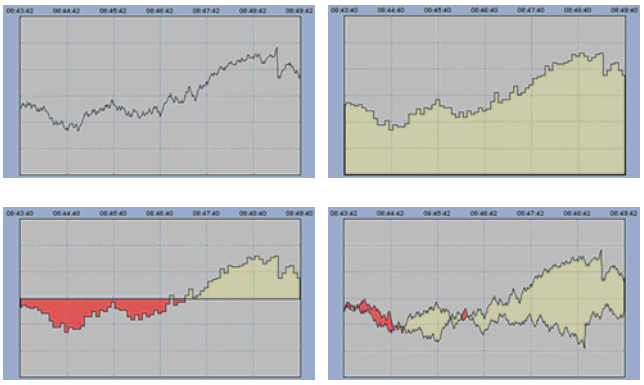

Figure 5: Various trend presentation modes.

Figure 6 presents a traditional trend display from a nuclear power plant simulator and Figure 7 shows how events may be visualised in a trend diagram.

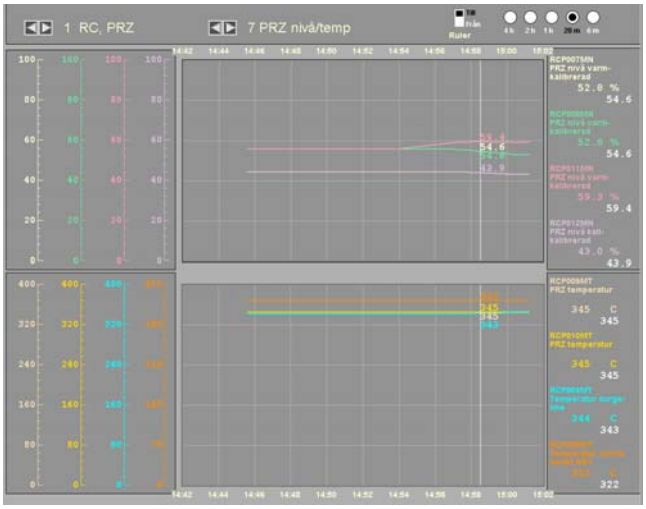

Figure 6: Two historic trend diagrams on top of each other, each diagram displaying four curves. Individual scales for each curve at the left.

Figure 7: Trend diagram with events. Events visualised by rectangles are related to the trended variable and their position follows the curve. Triangles at the bottom visualise events with relation to time only.

In the internal ProcSee system design, the logging part and the display part are strictly separated, and the software interface between these two parts is a well-defined set of functions. Trend diagrams normally request historical data from a central logger, but if an external program provides a logging facility or a historic database, ProcSee trend diagrams can use data from these sources directly. In any case, trend diagrams automatically request the required data from the logger and visualises the data according to its current attribute settings.

In ProcSee, external programs fully control the definition of time. This is particularly important for simulators, which may run slower or faster than real time and where time may be reset to re-run a scenario. The ProcSee trend system provides functions for handling simulator requirements, including saving and loading snapshots.

Integration with the Process

ProcSee is designed to operate in a network environment and offers a high-level API for easy integration with the process and external programs. The API is a library of C functions. It is optimised for large amounts of data and supports transparent cross-platform communication with very high throughput rates.

If the process includes one or more OPC servers, the RTM can read and write process values directly from the OPC servers using ProcSee's integrated OPC client. In such cases, no programming, only configuration, is needed to get regularly updated process values into the RTM. The OPC client is configured using a configuration tool available from the GUI editor. Using the configuration tool, the designer browses the OPC servers for the relevant process values and specifies the desired update frequency. The resulting configuration is text-based, enabling developers to add, remove or modify items using their favourite text editor.

If no OPC server is available, a C/C++ program, denoted "Process I/O" in Figure 1, must be developed to regularly update the RTM with process values. This program should use the process' software interface to access process values, and the ProcSee API to send the values to the RTM. API functions exist to support the programmer to deliver the values to ProcSee in an efficient way, without having to deal with low-level communication issues.

As the ProcSee RTM is based on a virtual machine, pTALK statements can be issued also from external programs. API functions exists to send pTALK code from an external program to the RTM for immediate compilation and execution. For instance, suppose the programmer of an operator support system would like to inform the operator specifically when a particular process state is encountered. The GUI designer may implement a pTALK function handling the desired visualisation, for instance by bringing up a specific display. The programmer can then call the pTALK function from the operator support system like this:

```
PfExecute(NULL,"{onTurbineTrip();}");
```

On the other hand, functions in external programs or process I/O modules may be invoked from the GUI, for instance as a result of an operator interaction. At start-up external programs and process I/O modules may declare functions to the RTM, i.e. provide the functions' names and arguments. Thereafter, the functions can be used as any other pTALK function in the GUI. Figure 8 shows how a function in a process I/O module can be invoked from the GUI.

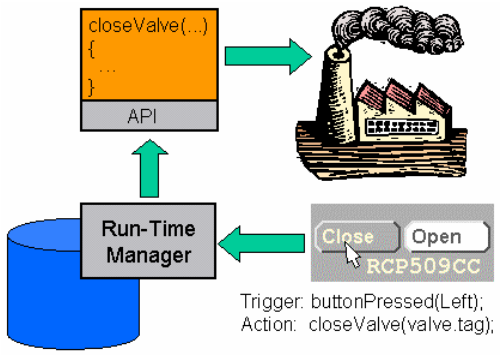

Figure 8: A user-defined function invoked from an operator dialogue in the GUI.

GUI Integration

ProcSee displays may be integrated into the user interface of external programs. The external program creates one or more windows and leave the control of these windows to ProcSee using one the functions in the API. To the operator, such an integrated user interface will look like a single GUI.

The ProcSee editor itself is implemented using this technique, and Figure 9 shows another example of a ProcSee display integrated into an MFC-based application.

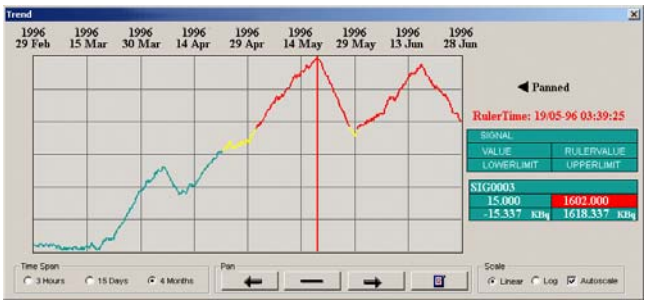

Figure 9: Integration of a ProcSee display into an MFC-based application.

Colours, Fonts and Cursors

The GUI editor includes a colour editor where the GUI designer can define new colours, including flashing colours. Colours can be specified using RGB or HLS values, or selected from a standard colour palette. In addition, colours can be easily grabbed from anywhere on the designer's screen. User-defined colours can then be used in any graphical object, for instance as shown in the function `statusCol` in the example on page 4.

All fonts installed on the computer can be used in ProcSee GUIs. The GUI editor includes a font selector where the GUI designer can easily pick the fonts he/she wants. Double-byte character sets like in Japanese, Korean and Chinese are supported.

ProcSee can use font-cursors, bitmap cursors and animated cursors that are installed on the computer.

Documentation

ProcSee is shipped with a complete set of user-documentation including a user's guide, a reference manual and a tutorial. The documents are also available in electronic format for easy retrieval and search.

Conclusion

ProcSee is a versatile tool for implementing dynamic GUIs for process monitoring and control. It provides an editor for defining and testing GUIs, a ready-to-use run-time executable for displaying and updating the graphics and handling operator interaction, and finally an API and an OPC client for connecting ProcSee to the process and external programs.

Contact Information

Project leader: Håkon Jokstad
Institute for Energy Technology
P.O. Box 173, NO-1751 Halden, Norway
Tel: +47 69 21 22 00, Fax: +47 69 21 24 90
Email: ProcSee-support@hrp.no
Web: www.ife.no/procsee