Picasso-3 User Interface Management System

Håkon Jokstad and Carl-Victor Sundling
OECD Halden Reactor Project
Phone: +47 69 21 22 00, E-mail: Hakon.Jokstad@hrp.no, Carl.Victor.Sundling@hrp.no

Abstract

Picasso is a versatile SW tool for developing and displaying dynamic Graphical User Interfaces (GUIs), particularly aimed at process surveillance and control systems.

Picasso 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.

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

At run-time, Picasso 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.

Picasso can be connected to simulators, SCADA systems or real-time databases using a high-level Application Programmer's Interface (API). The API is open for integration with industry standards such as OPC and CORBA and Picasso and the API are optimised to handle large amounts of data and frequent display updates.

A versatile trend-system for logging and visualising historic data is fully integrated with the Picasso run-time system and editor.

Picasso runs under Microsoft Windows (NT, 2000, XP) and various Unix platforms including Linux, HP-UX, Solaris, Irix and Digital Unix.

1. WHY USE PICASSO?

Developing user-friendly GUIs for process surveillance and control systems requires advanced SW tools. The tools should support the developers throughout the application's lifetime, from the initial prototyping to the final maintenance phase. Picasso 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.

With its unique flexibility and performance Picasso is particularly attractive to SCADA system suppliers who can establish generic GUIs for easy adaption and configuration to their customers' needs.

1.1 Main Features

- Advanced interactive GUI editor

- Highly flexible definition of customised objects with dynamic graphics and operator dialogues
- Easy integration with simulators, SCADA systems 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

1.2 References

Picasso has been used for simulators and process surveillance and control systems in various application areas such as 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 Picasso on the web.

2. SYSTEM ARCHITECTURE

Figure 1 below describes the Picasso system architecture and the roles involved in developing and using a Picasso-based application.

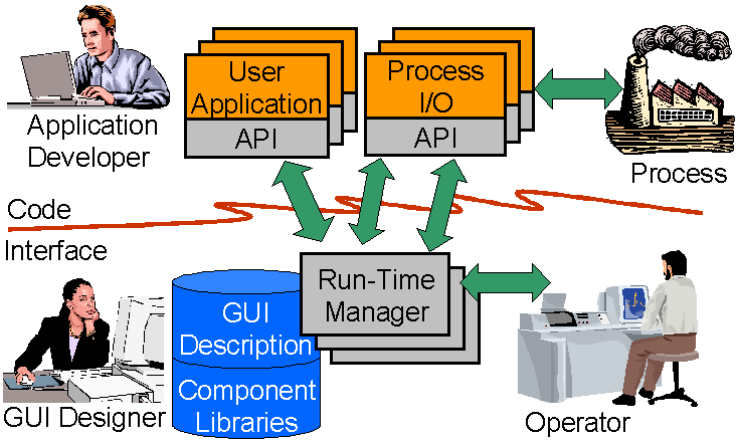

Figure 1: Picasso system architecture.

2.1 GUI Editor

The GUI designer implements the GUI using the GUI editor. All aspects of the GUI is defined through the editor, and the result of her work is a set of files containing the complete GUI description, including re-usable component libraries.

2.2 Run-Time Manager

The Run-Time Manager (RTM) is the heart of Picasso. 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 user applications and the process, updates the displays according to the current values and states, and handles operator input according to the designer's definitions.

2.3 Process

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

2.4 User Applications and Process I/O

User applications and process I/O modules are C/C++ programs using the Picasso API to communicate with the RTM. Their main task is to provide the RTM with updated process parameter values.

These programs may also issue predefined or user-defined commands to the RTM, for instance request the RTM to display a certain picture when an event occurs in the process. Further, these programs can declare user-defined functions that can be invoked from the RTM, e.g. in response to some operator input.

2.5 Separation of GUI and Code

Picasso'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 of applications. Changes frequently involve modifications to the GUI, but user applications and process I/O modules can often be left unchanged.

3. TECHNICAL OVERVIEW

Picasso-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 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.

3.1 Implementing a GUI Using the Editor

The Picasso editor, presented 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.

The Picasso editor offers general 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.

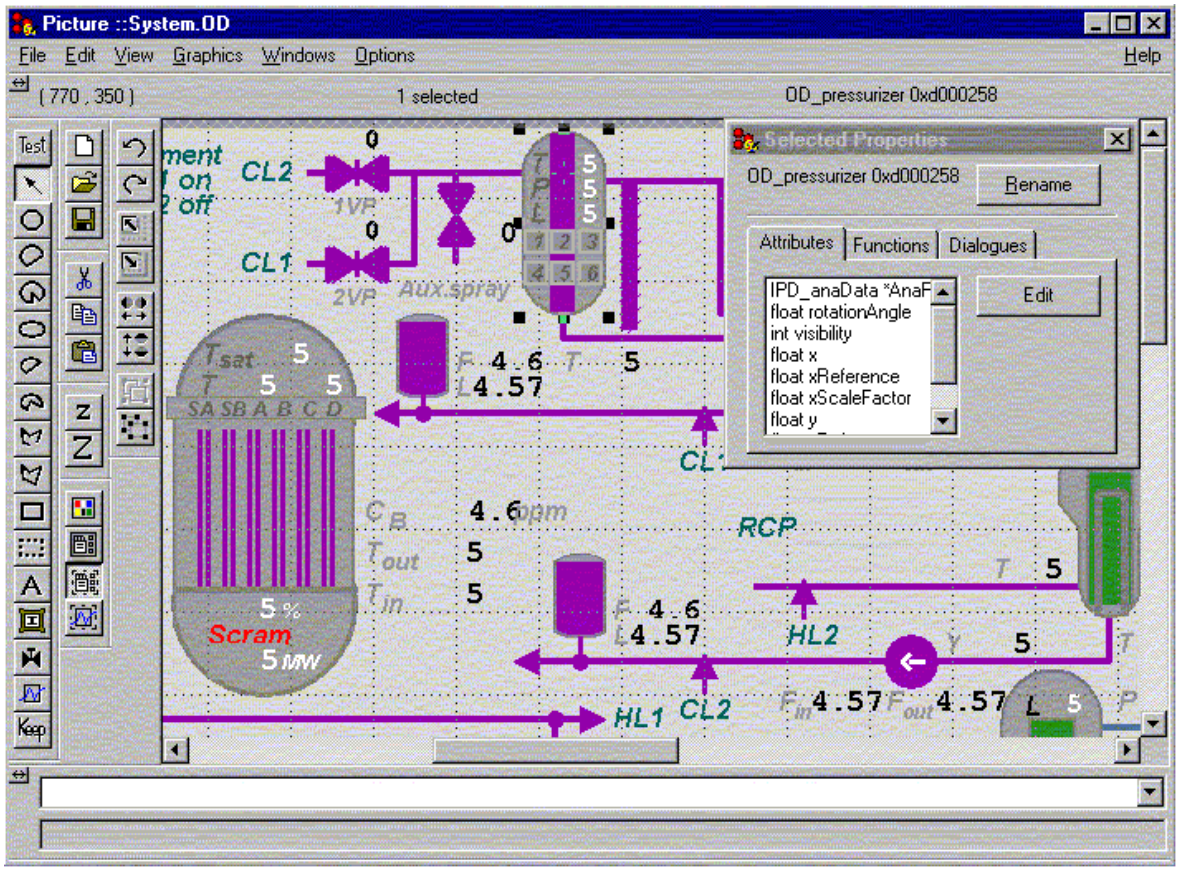

Figure 2: The GUI editor. Attributes, functions and dialogues anywhere in the object hierarchy can be viewed, set and modified. Switching between design mode and test mode is done simply by pressing a button.

3.2 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 then secondly to run an on-line system that uses the compiled interface specification.

In Picasso however, there is no distinction between off-line and on-line. The Picasso 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 Picasso's copy of the process parameter values directly from the editor, by connecting some temporary data generators, or by connecting to the real process.

In addition, the Picasso editor can be connected to the GUI for inspection, debugging or modifications at any time, even at run-time with the real process connected.

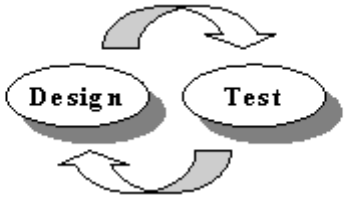

Figure 3: Picasso offers a development environment where the GUI designer can test the GUI while designing, without recompiling or restarting the system.

3.3 pTALK and the Virtual Machine

The heart of the Picasso RTM is a virtual machine able to compile and execute statements in the Picasso 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 her own functions as needed, pTALK's expressive power provides a unique flexibility for building customised GUIs.

Because of pTALK and the virtual machine Picasso-based GUIs can be modified at run-time. Using standard API and pTALK functions, the application developer can add, modify and remove graphics at run-time. In fact, this is the key feature used by the Picasso 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, Picasso can be, and in fact has been, used to implement tailor-made editors for specific purposes.

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

3.4 Basic Interface Components

At 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, size, 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, image, text and historic trend diagram. Regarding images, all major graphics formats are supported, including gif, tif, jpg, xwd, bmp, png, pcx, and tga.

To insert dynamic graphics on top of static images is a fast way to implement nice-looking dynamic objects.

3.5 Dynamic Graphics

In order to create dynamic graphics, Picasso offers functionality for connecting graphic elements to process variables. The connections are specified as pTALK expressions. For example, to connect an attribute like 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 green, yellow or red depending on whether the attribute value is greater than one of the limits.

```
int statusCol( TagData* td )
{
   if ( td.value < td.hiLim )
      return green;     // normal
   else if ( td.value < td.hihiLim )
      return yellow;    // high
   else
      return red;       // very high
}
```

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)';
```

If the pTALK expression is simple, it may be specified directly without defining a function:

```
height = 'tank1Level.value';
```

This way, any attribute of a basic or user-defined graphic 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 the graphics are updated accordingly.

3.6 Operator Interaction

An important aspect of any GUI is the interaction with the operator. In Picasso 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, or shape level. A dialogue has three properties:

1. *An area where the dialogue is active.* Most frequently dialogues are integrated with graphical components in the pictures, implying that the area covered by the component 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 user applications and process I/O modules may declare some of their functions to the RTM, so dialogues may also invoke functions in user applications and process I/O modules, for instance to start a pump or close a valve.

3.7 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 Picasso this is provided through the use of *classes*.

A Picasso class is a collection of basic or user-defined graphic 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 including 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. Classes can be modified at any time, and Picasso will ensure that all instances are updated accordingly.

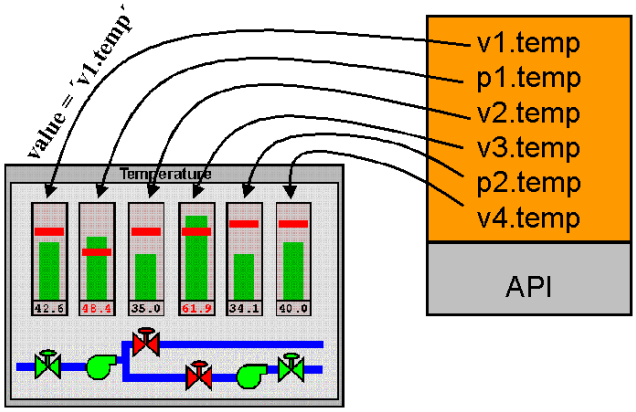

Figure 4: Instances of a bar graph class configured in a picture.

Easy retrieval and re-use of classes is 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 applications.

3.8 Historic Trend Diagrams

In addition to visualising the current state of the process, Picasso 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*. Picasso 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 future data, enabling Picasso to visualise for instance results from a predictive simulator.

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 the 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 user applications.
4. *Displaying events*. Logged events are 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 event types, and the objects may visualise the data related to the event. Events and curves can be integrated into the same diagrams.

Figure 5 shows an example of a trend display from a simulator in the Halden Man-Machine Laboratory.

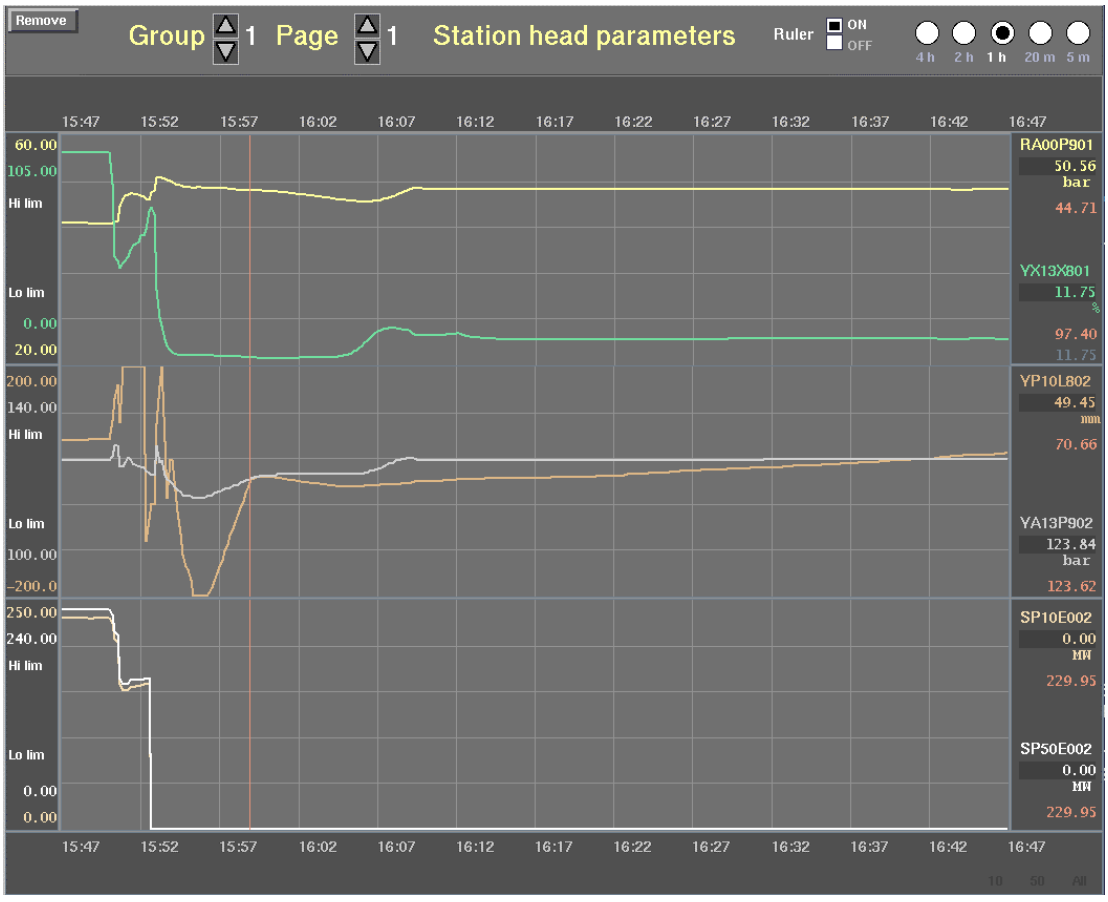

Figure 5: Historic trend diagrams. In this example three diagrams are put on top of each other, each diagram displaying curves for two process variables. For each of the six variables, the picture shows variable name, upper and lower limits, current value, and value at ruler position.

Figure 6 shows an example of a trend diagram where events are visualised in the diagram.

Figure 6: Trend diagram with events. The events visualised by the rectangles are related to the trended variable and its position follows the curve. The triangles at the bottom visualise events with relation to time only.

A trend diagram is a basic shape that can be included in pictures or in complex user-defined components. 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 or manual scaling of curves, linear or logarithmic scale, various presentation modes and more. All features can be controlled by pTALK statements and can thus 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 to logged variables. Figure 7 presents a few examples.

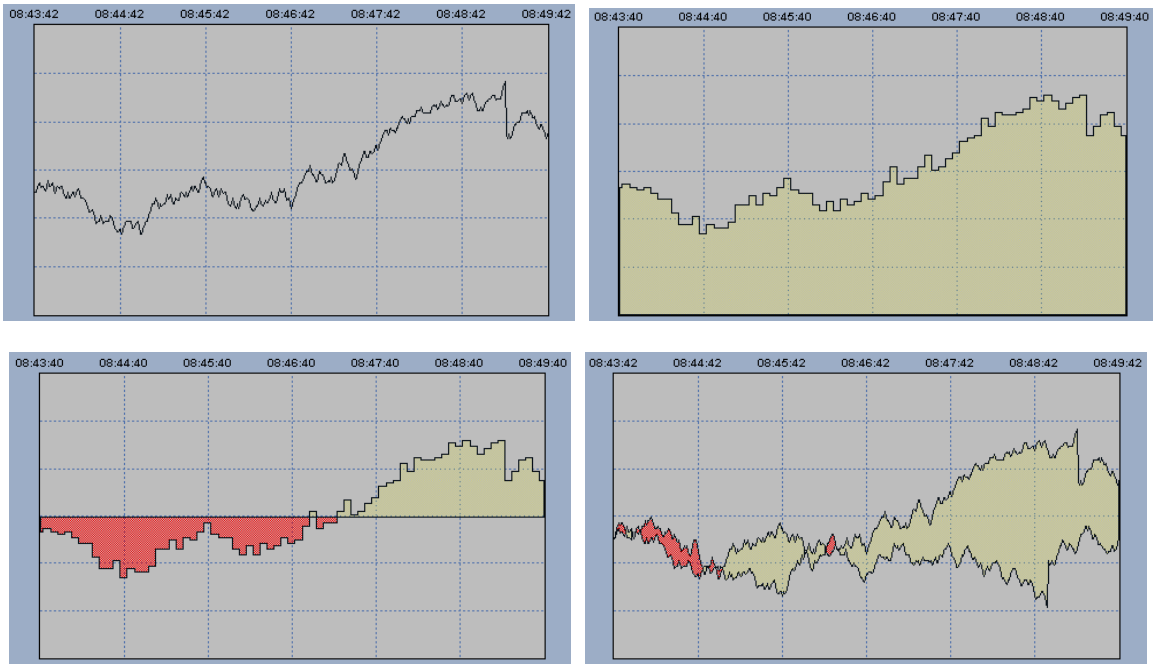

Figure 7: Various trend presentation modes.

In the internal Picasso system design, the logging part and the display part are strictly separated, and the SW interface between these two parts is a well-defined set of functions. So, if the user application provides a logging facility or a historic database, Picasso trend diagrams can use data from these sources directly, avoiding data logging in Picasso.

Further, this strong separation of logging and displaying supports the use of centralised loggers in large systems. Operator displays at several screens may request historical data from a central logger.

In any case, trend diagrams automatically request the required data from the logger and visualises the data according to its current attribute settings.

In Picasso, user applications fully control the definition of time. This is particularly important for simulator applications which may run slower or faster than real-time and where time may be set back to re-run some specific scenario. The Picasso trend system provides functions for handling such requirements.

3.9 Integration with the Process

Picasso is designed to operate in a network environment, and offers a high-level API for easy integration with user applications and the process. The API is a library of C functions and is open for integration with industry standards such as OPC and CORBA. It is optimised for large amounts of data and supports transparent cross-platform communication with very high throughput rates.

The main responsibility of the user applications and the process I/O modules is to update the Picasso RTM with variable values as they change. API functions exist to support the application developer to deliver the values to Picasso in an efficient way, without having to deal with low-level communication issues.

As the Picasso RTM is based on a virtual machine, pTALK statements can be issued also from user applications. An API function exists to send pTALK code from a user application to the RTM for immediate compilation and execution. Suppose the user application would like to trigger an action in the GUI, for instance to print a hardcopy of some displays. The following statement will do that, provided that the user-defined pTALK function `doPrint()` is defined and implemented in the GUI:

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

Functions in the user applications or process I/O modules may be invoked from the GUI, for instance as an action in a dialogue. At start-up the user application declares its selected functions to the RTM, i.e. provides their names and arguments. Thereafter, these 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.

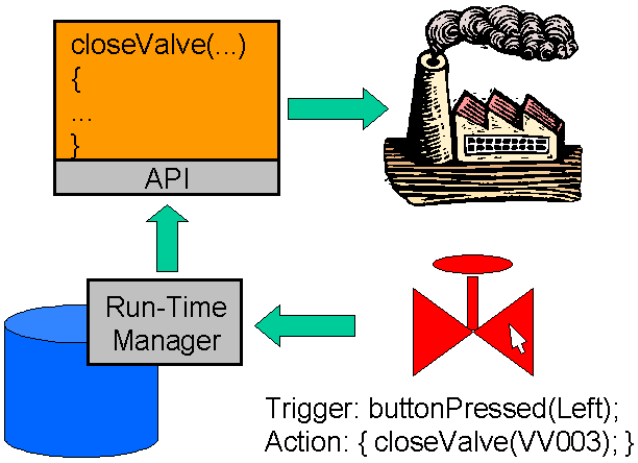

Figure 8: A function in a process I/O module is invoked from a dialogue in the GUI.

3.10 GUI Integration

Picasso displays may be integrated with the user interface of other applications. For instance, Picasso can easily be integrated into MFC- or Motif-based applications. The external application will then create one or more windows and leave the control of these windows to Picasso. To the operator, such an integrated user interface will look like a single GUI.

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

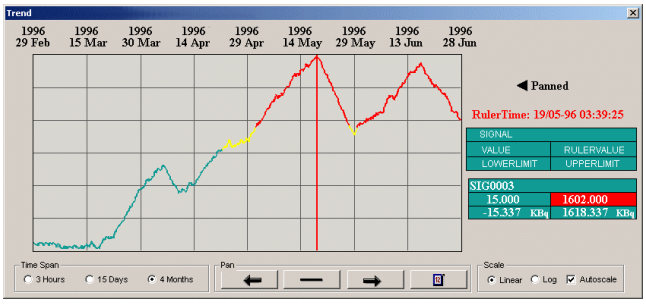

Figure 9: Integration of a Picasso display into an MFC-based application.

3.11 Colours, Fonts and Cursors

The Picasso editor includes a colour editor where the GUI designer can define new colours, including blinking colours. Colours can be specified using RGB or HLS values, or be selected from standard colour palettes. In addition, colours can be easily grabbed from anywhere on the designer's screen.

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

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

3.12 Documentation

Picasso 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.

4. CONCLUSION

Picasso is a versatile tool for implementing dynamic GUIs for process supervision and control systems. 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 for connecting Picasso to the process and external user applications.