

Embedded Graphics and Data Models – New approaches in the development of graphical user interfaces

Gerstendörfer Klaus
XiSys Software GmbH

Schachner Robert
RST Industrie Automation GmbH

Schuller Peter
MicroSys GmbH

Preamble

Combining visualization and data models for embedded graphical user interfaces is an entirely new and innovative approach for developing complex animated graphical functions for small, embedded systems. The realization of the graphical user interfaces is handled completely by the designer, without the programming required previously. Animated objects for the display and manipulation of process and system data merely need to be configured by reverting to the underlying data model. Extensive traditional code writing can be largely dispensed with. The German Federal Ministry of Research and Technology has been supporting this project within the context of the Research Initiative SPES 2020 (Software Platform for Embedded Systems) since January 2009, and will continue to do so for the next 2 years.

1. The environment for embedded graphical systems

Requirements and requests for improved display quality and operating properties of graphical user interfaces are increasing also for embedded systems and devices. Even applications on small systems with diminutive displays are no longer content with simple bit-mapped graphics. Here standards are being set by consumer equipment, e.g. the iPhone. Appealing animated icons and symbols, which allow intuitive operation, are becoming increasingly interesting for embedded applications. Therefore, what are the features important for a modern embedded graphical system?

The most important features are:

- Appropriate display of system processes so as to enable intuitive interaction with the user. This is facilitated by using animated objects and control via touch input.

- Tiled windows showing a simultaneous graphical display of several system operations and functions, in a clear and non-confusing manner.
- A simple generation and display of screen masks with only minimal or, even better, no programming effort. This leads to a significant reduction of traditionally high development costs.
- The integration into complex heterogeneous system environments: hardware, system software, application, networking, and redirection of graphical I/O onto different platforms.
- Display of information (data flow) connected with functional application flows. This leads to animated graphical objects which must be capable of being easily integrated into control programs.
- A low requirement of system resources for embedded applications and support of typical embedded architectures, ranging from PowerPC through x86 to ARM, MIPS and SH4.
- International language support is very important for our export-orientated industrial products.

The well-proven graphical system XiBase9 is a good starting point for satisfying (fulfilling) the above requirements.

The XiBase9 concept at a glance (Overview of the XiBase9 concept)

XiBase9 is an object-oriented graphical system which works independently from the operating system. New graphical objects are either created using a highly convenient vector editor or are appropriately adapted by modifying already existing objects. A graphics server communicates with the application via a defined interface and is responsible for displaying objects together with their functions. By uncoupling the graphical functions from the application it is no longer necessary to write program codes which have to be integrated into the application. Without the necessity of modifying the program code of the applications, changing the graphical user interface is very simple. This results in clearly arranged code structures and enables easier maintenance of the software during active operation. XiBase9 has already been used for a long time in embedded systems with high demands on software reliability and real-time capabilities and, at the same time, minimal resource requirements. Individual or customary “Windows-like” graphical user interfaces can be created easily by using the included development tools, and the internationality of the program allows it to run on all important Western, Asian or Arabic applications. Each of the functions missing in this

graphical system and which is necessary to fulfill the profile described above in its entirety requires a paradigm shift in the design of the software of embedded systems. The support of model-based software development is a modern approach hereto. What do we mean by this?

2. Basic principles of model based development

3.1 Model based software development, a short introduction

Essentially two aspects lead to this approach. On the one hand we have the increasing complexity of current and future embedded systems solutions, which must be structured and controlled clearly. This eliminates the familiar data graves and enables the user to concentrate completely on the essentials of his development work. On the other hand, constantly writing new program codes for systems operations is very time-consuming and annoying. The new approach is based on the construction kit principle. It is designed to work with existing and future system components and to be universal and reusable. In this context it is essential to simplify extensive programming so that complexity as well as long development times for projects can be kept under control. When working with model based software development, the ideal is to have formal models generate software automatically. I.e., methods and tools must exist which allow the description of a desired issue on an abstract level. The implementation into executable code will then be automated. Examples for this are products such as Matlab-Simulink from The MathWorks, or Rhapsodie from Telelogic, now IBM.

3.2 Model based software development for embedded systems

The purpose here is to simplify complex development projects. Below we have listed some of the basic conditions to be considered:

- Long-life products: Many years ago, projects for a VMEbus-architecture were developed using e.g. 68k processors. In the future, the same or only slightly modified applications will to be solved with a Compact-PCI system, new and modern hardware platforms will be required, capable of supporting established applications without extensive modifications.
- Modern I/O connections will be required.
- New telecommunication technology will have to be considered.
- New sensor technology and actuators will be integrated.

Traditional developments are at a serious disadvantage in this environment. Changes, adaptations and redevelopments in many instances require extensive new software programs down to the “C” code level. To date it has been very difficult to reuse software components. This is reflected in increased costs and prolonged project development periods. The approach described above adopts two entirely new levels of abstraction, allowing a component oriented system architecture and a dramatic reduction of the actual programming effort.

3. Modeling via a data-centric descriptive model

4.1 The CASE-Tool „GAMMA“, a so-called software plug-in

A software plug-in (figure 1) picks up the success story and the idea of standardized hardware interfaces (e.g. bus systems like VME or Compact-PCI) and realizes it on the respective software level and system solution. Constructing the system components in a pluggable and configurable way separates the development process from extensive system software adaptations, driver adjustments for peripherals, I/O functions and integration of network functions based on time-consuming C programming. This saves an enormous amount of time during the development process.

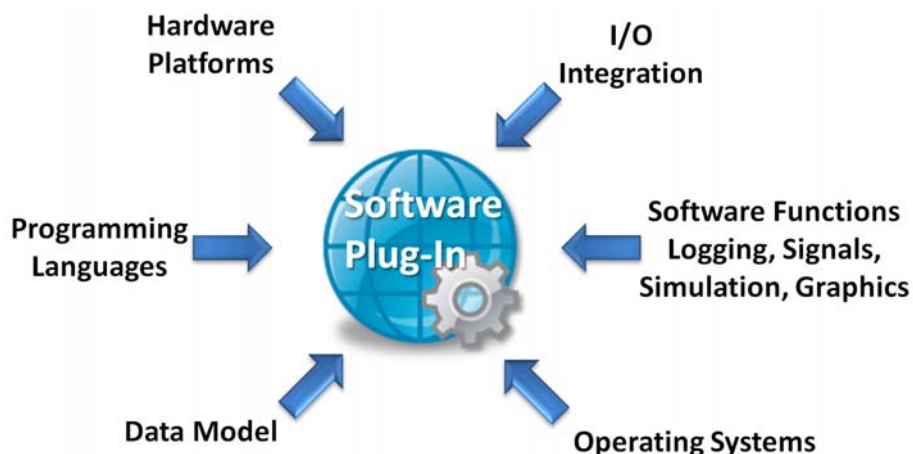


Figure 1: The software plug-in and the system components which it connects

The software plug-in abstracts, configures and integrates:

- Whole hardware platforms, including CPUs, BUS, memory and local I/O-systems
- Software systems



- Communication and network infrastructures: TCP/IP, field buses such as CAN, EtherCAT, ProfiNET, etc.
- Programming languages
- Graphics servers, e.g. XiBase9
- Graphical programming environments, e.g. MatLab-Simulink
- Heterogeneous computer architectures, also those integrated into other networks (mostly into TCP/IP)
- SPS systems
- Company specific software know-how in the form of components
- Additional techniques, e.g. visualization of processes via OPC (OLE for process control) interfaces

A significant component of this software plug-in is the data-centric organization of system variables via their mapping into a data model (figure 2).

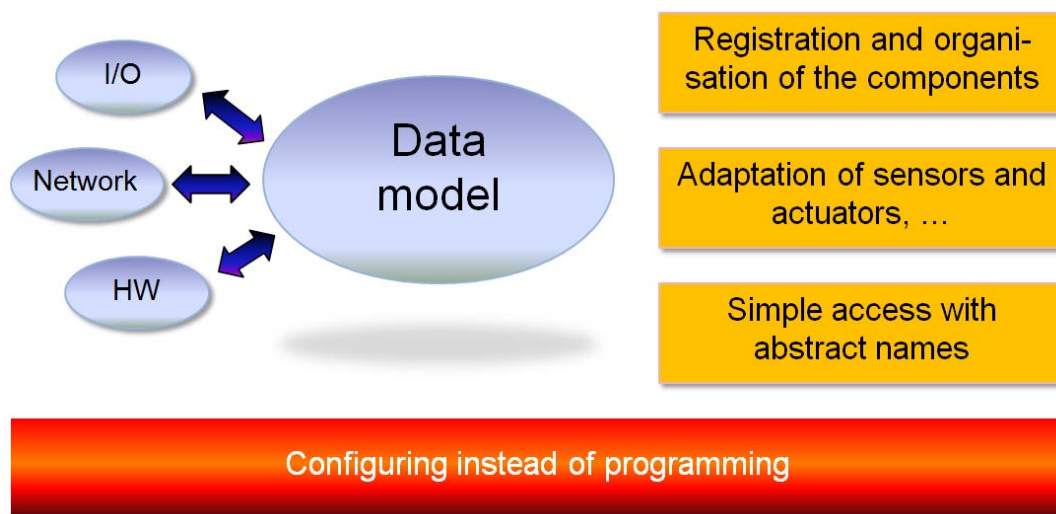


Figure 2: The data model represents the system components

In this data-centric model, the application is organized around a software model whose data is the central focus and, above all, stands as its own construct outside of (separate from) the application. Access to this data is conceived such that it functions locally, via networks and/or different computers with different software systems and programming languages. All control functions are designed in a single process, and conversion into executable programs on different system components is realized via the software plug-in GAMMA. Prerequisite is, of course, previous integration of the components (both hardware and software) into the “GAMMA” framework. With this method, systems can be built in which heterogeneous

computer environments, data loggers, I/O systems, sensor systems and actuators interact with each other. Intelligent variable structures are an additional feature of this model. A value that is to be processed automatically communicates, e.g. within the data model, every change in its condition to other processes, which, in turn, no longer have to sample data changes via laborious polling but are automatically informed and activated by the value itself. Thus, in addition to containing mere data information, a variable has the additional capability of triggering functions directly. This results in considerably simplified and more clearly arranged program architectures. Most system functions can be converted via simple configuration instead of extensive programming. With the ability to simulate the process variables and to log all changes, simple test procedures can be set up separately from the actual controlling software, resulting in greatly simplified development and testing. Testing and development work seamlessly together making extension of the controlling software unnecessary (superfluous).

4.2. The data model and its linkage to the graphical system

The data model describes the states of the system processes represented by variables. To handle the visualization of these states and the interaction with the users, the graphical system must have the following features:

- Linkage of the variables in the data model to the display units is to be automated. This is particularly essential for systems that have to process thousands of values and extensive I/O data.
- Representation and animation of displays without extensive programming by using standardized and configurable processes of graphical objects.
- Interaction with the sequence programs that control the system functions.

In order to realize these features within the graphical system, the GUI builder is upgraded with the following functions:

- Determination of the available variables of the data model and direct linkage thereof to graphical objects.
- Adjusting the features of variables to display units, e.g. scaling or display of limits.
- Determination of run-time rules without programming.

5. Practical implementation of this approach

5.1. The new development process, demonstrated with the example of designing an aviation electronics instrument

The graphical objects, such as the attitude indicator used in this example (figure 3), are designed with the GUI builder. Their visual features and kinematics are defined and determined by means of a design process using a configurator. As a result, object programming and designing are uncoupled. This enables experts, such as graphic designers, to realize complex ergonomic and interactive graphical surfaces. Animation and movement of graphical elements are configured with a so-called vector editor. The attitude indicator (virtual horizon), (figure 3) must be able to display complex aircraft movements:

- Display of an artificial horizon which includes climbing, descent and declination.
- Display and masking of limit values such as angle of incidence, declination, etc.
- Display of angles of approach, programmed landmarks and warnings.

The vector editor defines and describes the individual objects as well as their features. Separate low level programming is not necessary.

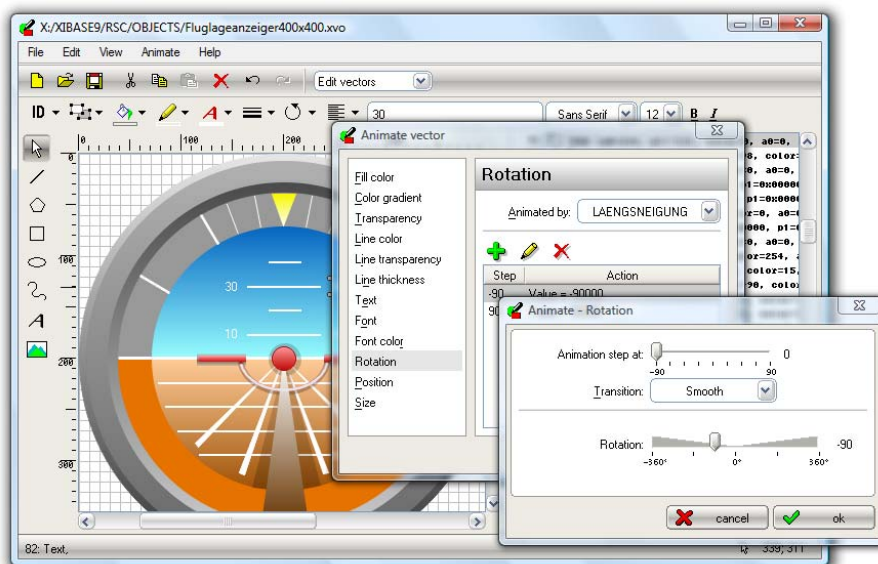


Figure 3: Designing instead of programming, using an aviation electronic instrument as an example

Advantages of integrating the graphical system into the data model

To date, integration of the graphical functions into a dynamic system environment have required extensive programming. E.g. the graphical display of pressure within a sequential program has to be combined with functions, limits, alarms, animated reports and perhaps also with color changes or other changes to objects. To date, all of this must be explicitly programmed. Given the need to program hundreds or even thousands of values, as is often the case in industrial plants, a methodical simplification will lead to a significant reduction of costs and saving of time. The XiBase9 GUI builder, equipped with the respective new interfaces to the data model, acts as a coupling link between the variables in the model and their associated functions in the overall visualization environment. This of course includes the graphical display of all stated variables as well as the handling of I/O functions.

5.3 Configuration instead of programming

Via this new interface automatic access to the variables in the data model is now possible without the need to manually program each single dependency.

In a menu controlled interface, all variables derived from the model are first assigned to the previously created graphical objects. In the next step, the specific features of the variables are adjusted to the desired display functions. The objects can be animated, i.e. their motion properties are configured using the screen display. In the attitude indicator (virtual horizon) shown in figure 3, this method is used to define that the background display is able to rotate within a defined angular range of possible movements in front of the body reference plane.

5.4 The sequential program as connecting link to the runtime features of display objects

The graphical objects have now been provided with a specific appearance consisting of form, color, dimensions, transparency and description of their movement characteristics. Needless to say, the sequential program, i.e. the program for logical processing of the process levels, is an essential part of the application. It provides an additional level of abstraction in which the rules previously defined in the GUI builder are automatically executed. Functions processed by the sequential program are, for example:

- Initializing the graphical environment when booting up the system
- Scaling the graphical objects
- Processing the changes in the variables, e.g. response to signals or polling

- Manipulation of variables by GUI events, e.g. responses to external inputs such as alterations of waypoints, trim values, etc.
- Split readout, i.e. I/O functions on different displays

5.5 Advantages of this new approach

Recurring programming is rationalized, simplified and logically packed using abstractions. The GUI builder is upgraded in its function as mask generator with the ability to configure and include variables from a data model. Data are found and taken over by polling, administrated in lists and provided with descriptions of their features. Thus they can be connected to display functions and characteristics, scaling and runtime rules. This method of designing graphical objects, i.e. configuration and linkage instead of extensive programming, can potentially save 80–90% of the development work customary for traditional embedded graphical systems.

Involvement in the research project SPES 2020

Representatives of industry, universities and research institutes have formed an alliance striving to define a more standardized software method for the development of embedded systems. The goal of this project is to set up a procedure for the development of combined systems consisting of microcontrollers, mechanical data and software, allowing for the first time a model based approach which will do away with all the previous ad-hoc solutions. The innovative alliance “Software Platform Embedded Systems (SPES) 2020” is part of the BMFT research program “IKT2020” and is being financed initially for 3 years with a budget of approximately 38.5 million Euros.

XiSys Software GmbH, together with its partners ELMA TRENEW GmbH, FTI Engineering Network GmbH, IMACS GmbH, MicroSys Electronics GmbH, N.A.T. GmbH and RST Industrial Automation GmbH, a member of Embedded4You, an incorporated association, is one of the representatives of the small and intermediate companies participating in this project. Step by step, innovative model based embedded software solutions “Made in Germany“ are explored. The results contribute to the development of new products thereby strengthening Germany’s position as an industrial site in the area of technology.

Contact members

Dipl.-Ing. Klaus Gerstendörfer:
XiSys Software GmbH
Klosterstrasse 24
97236 Randersacker
Tel: +49 (931) 467 70 90
Fax: +49(931) 467 70 98
Email: kg@xisys.de
Web: www.xisys.de

Dipl.-Ing. Peter Schuller
MicroSys Electronics GmbH
Mühlweg 1
82054 Sauerlach
Tel: +49 (8104) 801-133
Fax: +49 (8104) 801-110
email: Schuller@microsys.de
Web: www.microsys.de

Dipl.Ing. Robert Schachner
RST Industrie Automation GmbH
Rosenheimer Landstr. 145
85521 Ottobrunn
Tel: +49 (89) 961 6018-00
Fax: +49(89) 961 6018-10
Email: rschachner@rst-automation.com
Web: www.rst-automation.com