

The MORPHA Style Guide for Icon-Based Programming

Rainer Bischoff^{*}, Arif Kazi^{*}, Markus Seyfarth^{**}

^{*} KUKA Roboter GmbH
Bluecherstrasse 144
86165 Augsburg

{RainerBischoff, ArifKazi}@Kuka-Roboter.de

^{**} Reis Robotics

Industriegebiet an der B426
63785 Obernburg

M.Seyfarth@ReisRobotics.de

Abstract

Icon-based programming paves the way for making programming of modern industrial robots simpler and more intuitive. The flowchart-like representation of the program structure provides a superior overview. Programming becomes possible without detailed prior knowledge of a syntax. The style guide presented in this paper suggests a number of manufacturer-independent design rules for intuitive icon-based programming interfaces operated via touch screen and speech input. The complete style guide can be obtained from the authors free of charge by e-mail.

1 Introduction

Numerous visual programming systems have been developed to address both specific application areas, such as user interface design and physical simulation, and more general programming tasks [Boshernitsan, Downes 1996]. Well known graphical programming languages include, e.g., LabView, MATLAB/ Simulink, UML and RCX. LabView is an icon-based graphical programming package for rapidly developing and customizing instrumentation or data acquisition and analysis applications [Beyon 2000]. The Unified Modeling Language (UML) helps to specify, visualize, and document models of software systems, including their structure and design [Rambaugh et al. 1999]. Simulink is a simulation and prototyping environment for modeling, simulating, and analyzing real-world, dynamic systems providing a block diagram interface [Mathworks 2001]. In the robotics domain, icon-based graphical programming has been tremendously successful for educating both children and adults with, e.g., the LEGO Mindstorms robot assembly and programming kits [Bagnall 2002].

In recent years, also industrial robot manufacturers have learned that graphical languages could make robot programming easier and more intuitive. The ease of use is becoming increasingly important as the favorable price/performance ratio over the past few years has just enabled industrial robots to tap a whole range of new markets and fields of application in non-automotive sectors. In these new markets and fields specially trained robot operating and programming personnel are not always available.

At the same time, the functionality of industrial robot systems – and thus also the complexity of the user interfaces – has been steadily increasing. The situation for the user is complicated still further by the fact that different manufacturers produce vastly different user interfaces. For companies wishing to use robots from different manufacturers, this means a significant increase in training costs.

In the late 1990s, the European collaborative research project AMIRA (“Advanced Man Machine Interfaces for Robot System Applications”, ESPRIT Project 22646) addressed this problem and developed a style guide for industrial robot system user interfaces [AMIRA consortium 1999]. Since the end of the AMIRA project, icon-based programming in particular has met with a great deal of interest. Compared with conventional text-based programming, the flowchart-like representation of the program structure (see Figure 1) has the advantage of offering programmers a significantly improved overview and allowing them to work more intuitively. Special knowledge of syntax is no longer required, and syntax errors are precluded by the fact that the programs are no longer typed. Also, parameters and their boundaries for a specific command can be easily visualized and selected. For graphical programming of industrial robots a preliminary draft has been made for an international standard [ISO 15187] relating to the icons used. A first

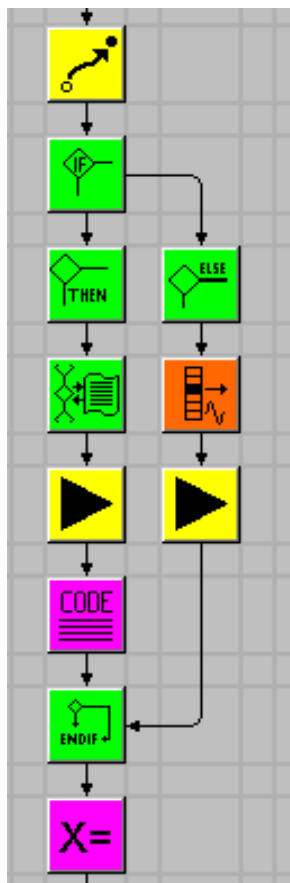


Figure 1: Sequence of program commands in an icon editor (the meaning of the icons is explained in the MORPHA style guide)

program control				robot motion			input / output			application							others	
program flow	wait	interrupt handling	error handling	motion cmds.	motion settings	sensor motion	input	output	computer comm.	workspace exclusion	handling	spot welding	arc welding	gluing	painting	(dis-) assembly	specific code	mathematics

3 Task-oriented structure of the user interface

The MORPHA style guide – like its AMIRA counterpart – works on the assumption of a task-oriented user interface structure: all the operator control functions required by the user to complete a task are offered in parallel. This may mean that it is necessary for one control function to be incorporated in several different places in a user interface, but the result for the user is an interface that is highly transparent and intuitive. In the case of a touch screen user interface, where only a very limited number of control elements can be displayed at the same time, a task-oriented structure gives rise to a natural division. For an icon-based programming interface, the MORPHA style guide distinguishes between three different categories of tasks:

- The program management screen provides administrative functions, e.g. copying and deletion of programs. The representation of directories and programs in a tree structure is similar to that used in the widespread Windows user interfaces.
- The editing screen contains the functions necessary for creating and modifying robot programs (e.g. insertion of a motion command into a program).
- For program testing, a test screen is required which allows the user to execute a program step-by-step while monitoring controller states and the contents of variables.

The style guide describes for each task category independently how the window should look like (which information should be placed or displayed at which location on the screen) and how the displayed items can be selected. Navigation paths between these three different task categories are proposed. They are illustrated in Figure 3. Eventually, the functions of each task category are explained. An example is given in Figure 4.

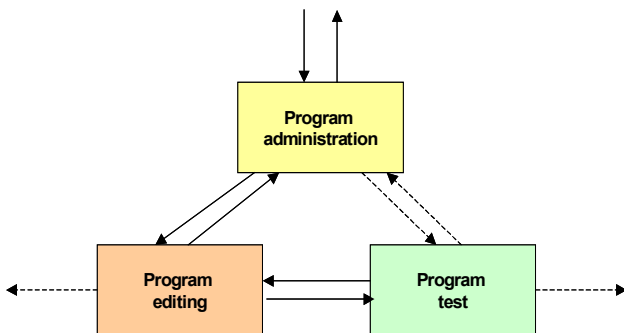


Figure 3: Task categories (main screens) and navigation options for an icon-based programming environment (solid lines: intended; dotted lines: optional navigation paths).

4 General Design Principles

In the field of user interface design, some general design rules should always apply. The MORPHA style guide has condensed the information of several sources of information (e.g., [Becker 2001], [Beu 1998], [Elo 2001], [Krauß 1998], [VDI 3850-3]), and provides it in a non-application specific way for both touch screen and speech-based interfaces. The main ideas are summarized in the sequel.

Task: Editing Programs

- ! The COPY ICON function is used to copy the icon to the clipboard. All command parameters are copied, too.
- ! The CUT ICON function will copy to the clipboard and at the same time remove it from the program
- ❖ The DELETE ICON function removes the selected icon from the program.

Figure 4: Example of the definition of a functionality within a task category (here: editing programs); please note that the “copy” and the “cut” functions are marked as mandatory while the “delete” function is optional (because the latter could be replaced by “cut” if space on the display has to be saved).

4.1 Design rules for touch screens

The use of a touch screen as an operator control element requires a fundamentally different user interface structure from a conventional windows-based application. The buttons in a windows-based application are generally too small to be operated with a finger. Furthermore, the mouse is no longer available as the input element, i.e., the functionality of the right mouse button is lost and actions such as the double click, drag&drop, etc., must be implemented in a different way.

In general, an application designed for touch screen operation should only be run in full screen mode, i.e., it should fill the entire screen. The application title bar and menu bar should be deactivated, as these are only significant for mouse operation. The mouse pointer should be switched off for the same reason. The operator now sees the entire screen and no longer subconsciously grapples with the problem of how to move the mouse pointer about the screen. Thus, user actions become more effective.

Traditionally, operators use their index finger to interact with a touch screen user interface. However, modern small size mobile devices like PDAs, WebPads, PenPCs etc., are often equipped with a pen as an input device. As interface objects require less space when designed for pen operation, a greater number of them can be placed on a given small screen size.

Pens may also be suitable for interacting with touch screen robot teach pendants, if it is ensured that a pen is always available when required. Pen input may even be advantageous as it avoids the display getting covered with dirt, oil or grease on the operators’ fingers or gloves. However, it has to be taken into account that during programming, robot operators are also confronted with tasks that require operation of other input elements (‘hard’ keys, 6D mouse). Repeatedly taking a pen into the hand and putting it away again will prove to be annoying. In many cases, a mixed design featuring finger operation for certain functions and pen input for others will be a sensible solution. However, such a design needs to be based on a careful analysis of the work flow of the user.

Apart from the question whether fingers, a pen or a mixture of both should be used in touch screen applications the style guide addresses more issues in greater detail. The reader can find answers to questions such as: How should the buttons be arranged and what should be the size of the

spacing between the buttons? What labeling and colors are suitable for buttons? When should buttons be presented to the user and how? Since tactile feedback is not available on a flat screen surface how should feedback be provided to the user when he presses a button? Should a button be activated upon first contact or upon releasing it?

4.2 Guidelines for speech interfaces

We are used to communicate with machines via graphical interfaces. Currently, speech control is only regarded as an “add-on” that could allow the operation of graphical interface objects by voice (e.g., by pronouncing the labels of the interface objects). When designing new user interfaces, some characteristics of speech interaction should be taken into account. The information provided in the MORPHA style guide helps to design speech-based I/O interfaces. In general, the user should be allowed to speak a command or to press alternatively the corresponding graphical button, i.e., speech operation should only be another way to operate the graphical interface objects. Which way of interaction fits best to the desired task may vary from case to case and should be left to the user.

Numerous general recommendations are given that enable a designer to answer the following questions: How should graphical and auditory feedback be given and interwoven? Should earcons (an auditory icon, i.e., a sound that is used to represent a specific event or object) be used and how? How can speech recognition be improved and what recognition rate and what time span until a feedback occurs are accepted by a user? How can a speaker be supported during speech I/O?

Furthermore, the suitability of certain speech-operated interactions is discussed, e.g., it should be possible to call objects and menu items even if they are invisible. For instance, a sub-menu or a program name that are not displayed on screen but known by the user by heart may be called directly. In contrary, it is not recommended to navigate through tables or in a tree structure by calling the tree nodes to open a child node (e.g., the sub folder of a folder tree).

For operation of industrial robots, sophisticated dialogs via speech I/O are not playing a very important role up to date. Currently, only very limited single-word commands can be given to robots. In future, however, speech operation will become more and more important and speech dialogs will transcend simple one-word commands. In this section, the elements of more sophisticated dialogs are briefly described.

In general, dialogs of speech-based user interfaces are composed of several basic elements that are linked by specific structuring elements. Basic elements can be questions, statements or commands. They can be used, i.e., to output status information or to find out who is talking, e.g., to allow for user adaptation. Speech recognition is usually performed based on pre-defined words or commands that have to be spoken exactly as defined or that can be recognized from continuously spoken words by spotting for pre-defined key words. Key word spotting allows for an al-

most arbitrary ordering of words into phrases, is usually speaker independent, and is, thus, more intuitive and better suitable for machine control. Speaker-independent recognition of continuously spoken speech is not yet available under the typical environmental conditions present in today’s manufacturing sites with a high ambient level of noise.

Various prompts for structuring and demanding user input should be available. Prompts should be kept short for experienced users; they should be longer for novice users of the system. An advanced dialog manager would adapt its verbosity level automatically to suit the user’s needs (incremental prompts).

5 Interface objects

In the literature regarding the design of touch screen user interfaces, only a few suitable interaction objects are described. These objects consist primarily of combinations of simple buttons and text input boxes for alphanumeric entry via an on-screen keypad. These objects are insufficient, however, for the implementation of more complex user interfaces. One aim of the style guide is thus to identify a greater variety of interaction objects and demonstrate their uses (and limitations).

The style guide defines a total of 14 interaction objects with which the user can

- trigger an action (action object),
- select an option (selection object),
- adapt parameter values (manipulation object), or
- request assistance (supporting objects).

Each definition contains a description of the task and the appearance of the interaction object, the actions required to operate it (subdivided into the categories touch screen and speech recognition), information about alternative implementation options and recommendations about how to increase the user-friendliness.

The different methods for implementing simple buttons are used here by way of example to illustrate briefly the options available for the design of touch screen user interfaces. Alongside traditional pushbuttons, which trigger a defined action directly, there is a whole range of other useful alternatives:

- In the case of “jog buttons”, the action continues as long as the button remains pressed.
- A “toggle button” is used to toggle between exactly two alternatives.
- The “multi-state button” is used to cycle through more than two alternatives, returning to the first selection option after the last one.
- The “radio button bar” presents the selection options as a row of adjacent buttons. Only one alternative is ever selected at any one time.
- A “menu button” can be used to offer a range of different actions or selection options.

Given that these different implementation types vary greatly in terms of functionality and response, it would

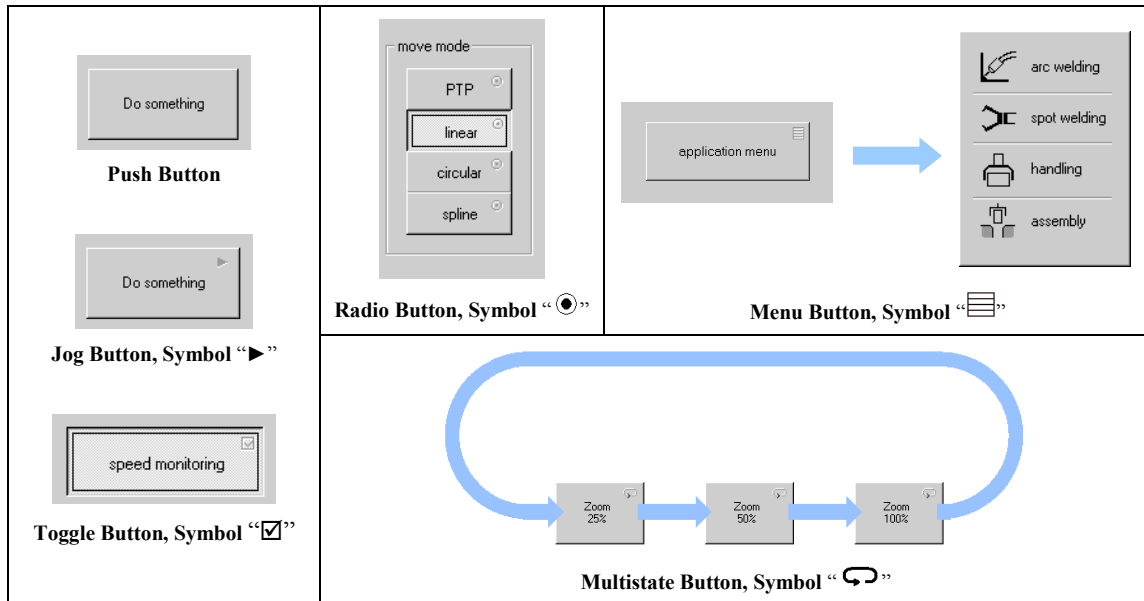


Figure 5: Different methods for implementing buttons.

appear sensible to indicate visually to the user the type of button used. The variants discussed are depicted together in Figure 5 with suggested identifying graphic symbols.

6 Prototypical implementation and trials

In order to validate the MORPHA style guide, a prototype user interface was created and improved in several iteration cycles (see Figure 6). It is based on an adapted implementation of the [KUKA Icon Editor 2000] and provides the icons shown in Figure 2. The prototype has no functional connection to a robot controller, but provides the user with all necessary control functions and allows

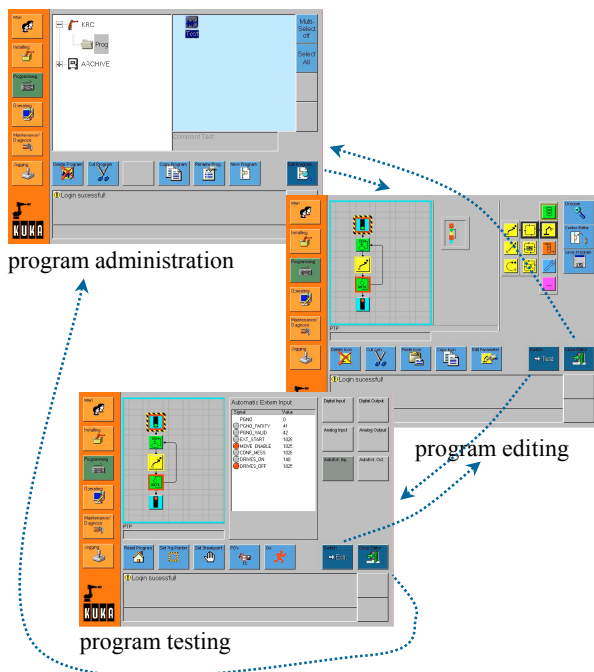


Figure 6: Example of an implementation of a development environment for icon-based programming consistent with the MORPHA style guide; arrows indicate the implemented navigation paths between the main screens.

evaluation by means of usability tests. Several tests with beginners, service staff, application and expert programmers have been carried out. Their task was to write a simple motion program and a more complex application program using special icon editor functions. They had to compare text-based programming versus icon-based programming. The time needed for programming was measured, and each user had to fill out a questionnaire and was individually interviewed. The results of these usability tests are quite encouraging (Figure 7). They suggest that icon-based programming of industrial robots is indeed an option not only for beginners but also for experts who are used to text-based programming.

In order to make a more realistic evaluation possible (operator standing up, hand-held teach pendant), a conventional teach pendant housing was fitted with a touch screen. Preliminary trials showed a high degree of acceptance for the prototype, particularly from users with little experience of working with robots.

7 Conclusions

The “MORPHA Style Guide for Icon-Based Programming“ is meant to be followed by developers of intuitive user interfaces. Icon-based programming is taken as an example how man-machine-interaction via touch screen and speech can be manufacturer-independently realized. Preliminary trials with beginners, service staff, application and expert programmers indicate that icon-based programming is indeed making robot handling easier by giving a better overview than conventional text-based programming and making program changes easier.

Touch screen and speech input interaction conforming to the suggestions of the MORPHA style guide could prove to be viable alternatives over conventional input devices. In the long run, this more intuitive type of interaction could help to open up new markets and fields of applications. If all robot manufacturers would adopt a

similar approach to graphical programming based on a set of standardized icons – as suggested by the style guide – it can be envisaged that even non-experts will be able to handle complex industrial robots, even from different manufacturers without requiring special training.

The complete MORPHA style guide can be obtained by the authors free of charge via e-mail or can be downloaded from the MORPHA web-site.

References

AMIRA Consortium (1999): *User Interface Style Guide for Robot System Applications*. Final report of the ESPRIT Project 22646 “Advanced Man Machine Interfaces for Robot System Applications” AMIRA. Distributed on behalf of the project consortium by Fraunhofer IPK, Berlin.

Bagnall, B. (2002): Core LEGO Mindstorms Programming, Prentice Hall PTR, Upper Saddle River, NJ.

Becker, B. (2001): Entwicklung eines Bedienkonzeptes für Spracheingaben. Diplomarbeit am Zentrum Mensch-Maschine-Systeme, Technische Universität, Berlin (in German).

Beu, A. (1998): Final Online Styleguide. Deliverable D20 of the ACTS Project AC010 „Multimedia User Interfaces For Interactive Systems and TV“, GSM GmbH, Stuttgart. Available at www.gsm.de/musist, last accessed June 15, 2002.

Beyon, J. Y. (2000): LabVIEW Programming, Data Acquisition and Analysis. Prentice Hall PTR, Upper Saddle River, NJ.

Boshernitsan, M.; Downes, M. (1996): Visual Programming Languages: A survey. www.cs.berkeley.edu/~maratb/cs263/paper/paper.html, last accessed on June 15, 2002.

ELO Touchsystems (2001): Touchscreen application tips. Elo TouchSystems Inc., Fremont, CA, USA. Available at www.elotouch.com/Support/10tips.asp (in German).

ISO 15187 (2000): ISO 15187:2000. Manipulating industrial robots - Graphical user interfaces for programming and operation of robots (GUI-R). Work on this standard is still in progress.

Kazi, A.; Seyfarth, M. (2002): Style Guide for Icon-Based Programming. Public-domain document in the BMBF lead project MORPHA. KUKA Roboter GmbH, Augsburg / Reis Robotics, Oberburg. Available at www.morpha.de.

Kraus, L. (1998): Einsatz von Touchscreens auf Basis von Flachbildschirmen. Universität Kaiserslautern, Lehrstuhl für Produktionsautomatisierung (pak).

KUKA Icon Editor (2000): Product information from KUKA Roboter GmbH, Augsburg (in German).

Mathworks (2001): Online Documentation of MATLAB/Simulink (June 2001). Available at www.mathworks.com/access/helpdesk/help/toolbox/simulink/simulink.shtml.

MORPHA Consortium (2002): Internet presentation of the BMBF lead project MORPHA, available at www.morpha.de.

Rumbaugh, J.; Jacobson, I.; Booch, G. (1998) The Unified Modeling Language Reference Manual (Addison-Wesley Object Technology Series), Addison-Wesley, Boston, MA.

VDI 3850-3 (2001): Dialoggestaltung für Touchscreens - Nutzergerechte Gestaltung von Bediensystemen für Maschinen. VDI/VDE 3850 Blatt 3. Internal draft version, May 2001 (in German).

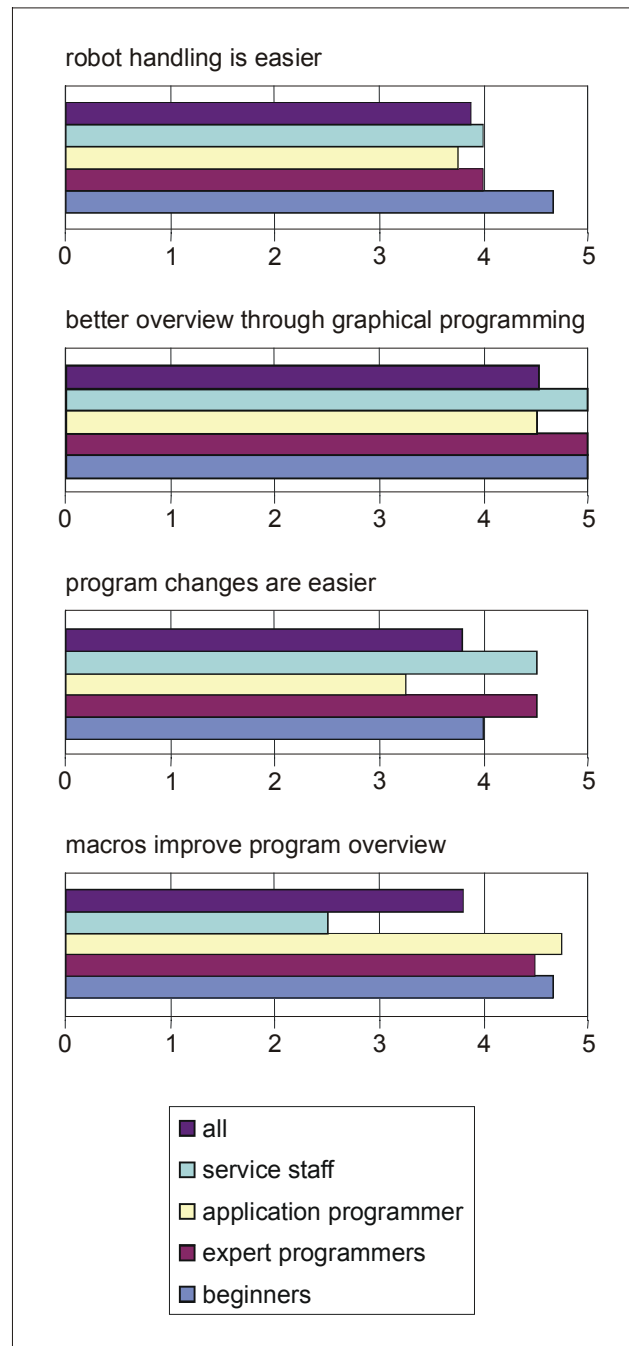


Figure 7: Results of a usability test which compared graphical programming to conventional text-based programming (based on the KUKA Icon Editor).