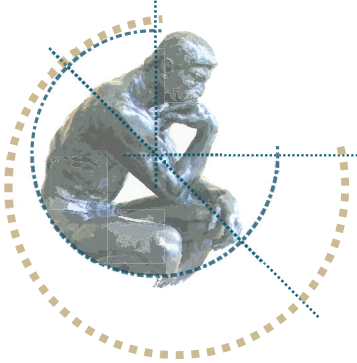

Programming support environments for safety applications MMI for complex safety-related control systems

T. BOROWSKI, M. HUELKE

BG Institute for occupational safety and health - BGIA,
Alte Heerstrasse 111, D-53754 Sankt Augustin, Germany

The degree of automation or of the "artificial" Intelligence of production plants further increases unmistakably in favour of increasing productivity. What are and what will be the tasks and concrete working activities of humans within this production processes? One recognizes a striking scenario by modern control systems which both for operators and for maintenance personnel and even more important for application implementer represents a central and elementary interface: the PC with its corresponding software tools for the respective tasks. This contribution (poster presentation) gives a current insight about what features these tools have to fulfil or how they should be designed so that all the activities can be executed as fault-freely as possible in practice. Furthermore existing product standards and known product solutions are presented.

Il est clair que le degré d'automatisation – ou d'intelligence « artificielle » – des installations de production ne cesse d'augmenter, dans une course à la productivité. Mais qu'advient-il et, surtout, qu'adviendra-t-il de la place et des tâches de l'opérateur humain dans ces processus de production ? On est frappé par l'exemple des systèmes de commande actuels, pour lesquels les opérateurs, mais aussi le personnel de maintenance et, mieux encore, les ingénieurs chargés de la réalisation du système, disposent d'une même interface élémentaire : un PC et des logiciels dédiés aux diverses tâches à accomplir. On indiquera dans cette présentation par poster ce que ces outils doivent savoir faire et comment ils doivent être conçus pour limiter dans toute la mesure du possible les risques d'erreur dans l'accomplissement de ces tâches au quotidien. On présentera en outre diverses normes produits et des exemples de produits connus.

Der Automationsgrad bzw. die "künstliche" Intelligenz von Produktionsanlagen nimmt zu Gunsten steigender Produktivität unverkennbar immer weiter zu. Was sind und was werden die Aufgaben und konkreten Tätigkeiten von Menschen in diesen Produktionsprozessen sein? Ein auffälliges Szenario erkennt man an modernen Steuerungssystemen, die sowohl für die Anlagenbediener als auch für Wartungspersonal und wichtiger noch für Applikations-Implementierer eine zentrale elementare Schnittstelle vorweisen: den PC mit entsprechenden Software-Tools für die jeweiligen Aufgaben. Dieser Beitrag (Posterpräsentation) gibt einen aktuellen Einblick darüber, was diese Tools können müssen bzw. wie sie gestaltet sein sollten, damit die Tätigkeiten im Praxisalltag möglichst fehlerfrei ausgeführt werden. Weiterhin werden Produktstandards und Produktlösungen vorgestellt.

Context

Control systems for machine and plant automation continue to grow in performance and complexity. Microprocessors and software are replacing hard-wired circuit technology. Technological change, in particular in high-performance components for numerical processing, data communication and data storage, has also given rise to numerous new products in the area of control systems for safety technology. Complex

safety control systems are being adopted in increasing numbers in everyday plant operations, in some cases in combination with safety field bus systems (figure 1).

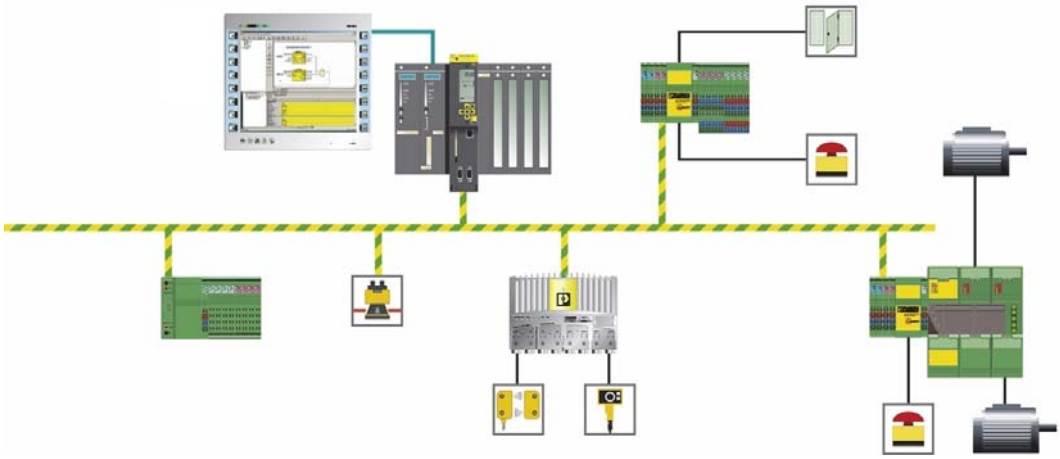

Figure 1 – Programmable safety logic controllers / field bus systems for machine and plant automation

This development induces change, both in technical performance and in the requisite interaction between Man and Machine. Man-Machine Interfaces (MMIs) have a particular function in this respect. They are of interest not only for operational machine control and process visualization, but for activities in the phase of application implementation: accident analyses reveal in fact particular blackspots in conjunction with deficient planning, implementation or integration of generic safety functions, especially in interlinked and interacting machines or plant components. In this context, modern safety controls support effective safety concepts. For the first time, they permit all-inclusive integration of safety technology and custom adaptation of functional aspects with a bearing upon safety. "Safety" is becoming more plannable. Nevertheless a further decisive development is taking place: the machine/plant-related functions are now being implemented in the way of “free programming” the application software. *Programming tools are therefore being employed as MMIs* which are used to perform several safety-related tasks within application development (figure 2) [1].

Figure 2 – Programming support environments – MMI for application software development

A growing number of programmable or at least configurable safety devices fulfil this function. Each device is associated with a proprietary tool and its own look and feel. This variety is often engineered either by programmers experienced with standard PLCs, or by electricians familiar with hard-wired safety circuits. Worse, the machine manufacturers seldom follow established good software engineering practice. In this

sector, application programming is often completed concurrently to field installation and under intense time pressure.

Could this scenario lead to an increase in occupational accidents due to systematic errors? Another issue are systematic software faults leading to failures which are not readily reproducible. Might such faults be the cause of a large number of unreported cases?

What counter-measures and activities are therefore appropriate? It is a known fact that most errors are introduced as early as the software specification and design stage, and are typically major conceptual errors. Conversely, oversights occurring at the coding stage can generally be detected by review or functional testing. Even application software can readily embody a strategy of fault avoidance in the initial phases of software development in preference to reliance upon subsequent bug fixing or the application of quality metrics. Besides the appropriate organisational measures to be implemented by the safety system integrator, the programming tool should play a major part in providing *technical measures for the avoidance of software faults* [2].

Method

For this purpose, safety PLC programming systems are to be provided with properties which satisfy the target criteria of "simplicity" and "user-friendliness", criteria frequently regarded as being only secondary. Furthermore, these properties are to be available to all individuals interacting with a control system, beginning with machine/plant engineering, through programming, commissioning and associated validation and operation, to maintenance. "Safety" should therefore also be attained by:

- *straightforward programming* by a reduction in the very awkward functionality of IEC 61131-3 (title: programmable controllers; programming languages) and the use of standardized and validated software function modules/libraries for safety functions such as emergency-stop, two-hand control, operating mode selection, safety door monitoring, motion control functions, etc. (figure 3);
- *simple application program generation* and fault avoidance through intuitive operation, clear functionality, transparent project administration, user guidance and input assistance (e.g. wizards, tooltips, combo boxes, drag & drop technique, cross-reference function, context-sensitive online help, interactive comprehensible dialogs and plain-text messages) (figure 4), project documentation assistance;
- *active fault detection* at the program and variable input stage to ease the work of the programmer (figure 5);
- *support for program verification* by simulation and debugging facilities, marking of the program flow, online checklists;
- *readily comprehensible but at the same time comprehensive and pinpoint diagnostics* with detailed status information

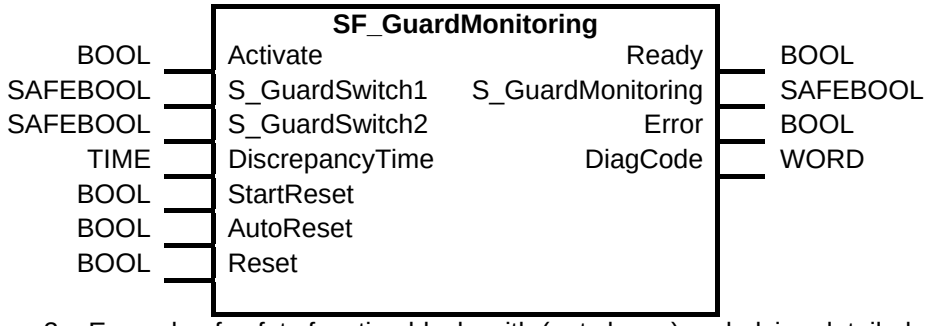

Figure 3 – Example of safety function block; with (not shown) underlying detailed specifications

Figure 4 – Interactive dialogs simplify “What-to-do” questions

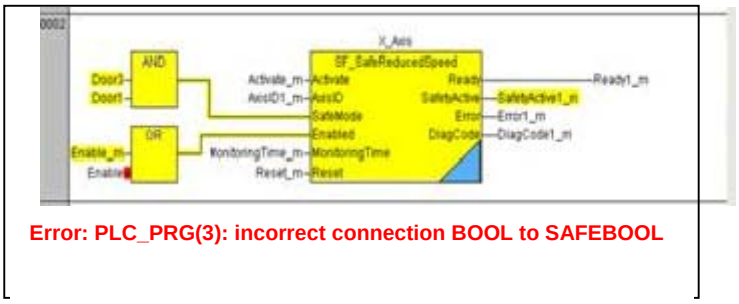

Figure 5 – Automatically detected mismatch between data types

Results

As the enumeration in the previous chapter points out, there is the basic approach of the software ergonomics and user-friendliness for programming systems. Furthermore it is recognizing the far-reaching fault detecting measures and particularly the use of standardized safe function blocks which in the best possible way relieves the programmer at his fault-prone work.
The contribution wouldn't like to do a market overview in this place, the result, however, is confirmed that some systems were already certified with the performance profile introduced here or are in the process of the testing and certification. The BGIA participates in charge in the standardization. Worth mentioning is the new standard (FDIS) ISO 13849-1 (safety related parts of control systems) [4]. It contains requirements for

safety-related application software which can partially fulfilled by safety-related programming support environments.
The “standard” for the programming elements arises in the PLCopen [3]. With participation of many manufacturers and authorities in the relevant working group (TC5) topics like

- *definition of language subsets*;
- *programming guidelines*;
- *definition of a function block library* and
- *style guide for safety function blocks*

are worked out here.

Bibliography

[1] M. Huelke, Applikationssoftware: mit Sicherheit !, Computer & Automation, February 2005, pp. 30-33

[2] M. Huelke, Software takes the lead in safety applications, 4th International Conference Safety of Industrial Automated Systems, 26.-28.09.2005, Chicago, Illinois/USA – Proceedings, Ed.: Automation Technologies Council (ATC), ANN ARBOR, Michigan/USA

[3] PLCopen-TC5, 2005, Safety Functionality – Part 1: Concepts and Function Blocks, Technical Specification Version 0.99, www.plcopen.org

[4] FDIS ISO 13849-1, Clause 4.6.3, Requirements for application software