DOI 10.5162/SMSI2020/P3.3

# Safety Sensor Applications with Graphical Programming

Nick Berezowski[1], Markus Haid[1], Jeet Biswas[1], Ishak Boyaci[1]
[1] University of Applied Science Darmstadt, Schöfferstraße 3, Darmstadt, Germany
nick.berezowski@h-da.de

**Summary:**
This research recommend the use of a a graphical full variability programming language for safety-related sensor system developments, in order to create framework conditions that result in a general approach for graphical sensor applications. A graphical programming language represents a language whose basic elements consist of blocks, symbols and lines between them, not like text-based or superimposed visual languages with ASCII-formatted semantics.

**Keywords:** Functional Safety, Graphical Programming Language, Graphical Full Variability Programming Language, Recent Developments, LabVIEW

## Introduction

Graphical programming languages offer a visual development design that increasingly focuses on natural human thought structures, which frees up thinking resources for content-related problem-solving approaches [1]. They can serve as essential means of communication when using fourth generation programming languages [1]. This suggests that graphical programming languages, with the ability to visually represent abstract control flow and data flow structures, can be considered fourth generation programming languages, and thus serve further development as well as functional security.

From a technical point of view, graphical programming languages are just another depiction of the implementation that is very similar to the models of text-based languages, but represent the implementation of graphical languages. Thus, they can substitute for well-known semi-formal methods, such as UML, provided that appropriate regulations are adhered to.

Since there is no research on this topic so far, the question of feasibility arises.

## Theoretical Fundamentals

There are numerous standards for functional safety. Some are described in Fig. 1. These are updated and rewritten irregularly in order to provide descriptions of the current state of technology. Among other things, emphasis is placed on tendentious technology innovations in order to consider them for future projects. [2]

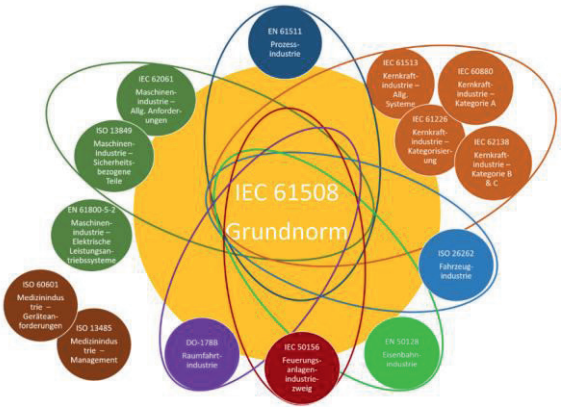


*Fig. 1. Overview of functional safety standards.*

For an initial process development in sensor systems only the basic standard IEC 61508 is needed.

To create a qualifiable development in a graphical development environment requires specific methods and approaches that can not be dictated solely by the development environment and require a clear definition in terms of functional safety.

## General structures

During the specification of requirements for safety-related systems, all relevant informations, such as requirements for the system, subsystems and components, must be recorded. Semiformal and formal methods can provide a detailed specification of the requirements. Due to their graphical structures, semi-formal methods are to be understood as similar to graphic code and thus preferable to formal methods. If possible, computer-aided specification tools should be used, which reduce the error potential during execution and lay the

basis for traceability. This is divided into forward and backward traceability. Forward traceability should be possible, especially at higher safety integrity levels between all phases of the software safety lifecycle [3]. A computerized tool could support these relationships.

**Programming Languages Structure**

To comply with the sensor requirements of the basic standard, a language subset must be defined which excludes the use of unsafe programming code constructs and checks their compliance with static analysis tools. [3]

A graphical programming language to be used must have a strict typing [3]. This means that type conversions must be obvious. Compliance checking can be done through in-program programming tests or, if necessary, additional static testing.

Since no already certified tools exist, tried-and-tested tools and translators should be used. These must be regarded as established and not error-prone in the relevant area of a safety-related system to be used. A test and verification environment that compares executable code with source code can provide additional confidence for individual sensor systems, but not for the complete environment. [3]

**Programming Structures**

There is a possibility that a graphical programming language may self-comply with some semiformal models, such as state transition diagrams and flowcharts, allowing for requirement determination and modeling close to programmatic implementation, potentially shortening development time and the necessary framework.

The rules of structured programming must be applied. Defensive programming can only be used in necessary places, since it also worsens the understanding of the complete program. [3]

The modular approach offers several sub-methods, all of which must be adhered to in a graphical programming language [3]. Some graphical languages inherently have a modular flow-controlled structure that supports these methods.

In general, a monitoring device should work with separation between monitoring and monitored computer in order to demonstrate a general independence for the introduction of a programming language. [3]

**Completed Work**

First, various standards and guidelines for functional safety and authoritative literatures for graphical programming languages were analyzed to make reference to the prior art. Based on the basic IEC 61508 standard, a rough concept with various possible solutions for software development in graphical programming languages was developed. The different methods and procedures let us derive an overall architecture using semiformal methods, which creates a direct relationship between design, development and programmatic implementation of a sensor application. It was possible to create a theoretical concept for an architecture framework, which should consist of project management, configuration management, test management, design and development tools in order to create a comprehensible link with the programming language and physical system.

**Further Work**

Currently, the previous knowledge for graphical full variability programming language apply. Here, an implementation was created that provides the basis for testing and validation through the use of semiformal methods and model-based analyzes. The basis for this is the establishment of safety functions via finite state machines. Based on the limitations of the language scope by means of language subsets, expert interviews should be conducted, if possible, with responsible persons involved in current established qualified procedures. This gives the basic work for recommendations for developing a policy in a special graphical programming language. Subdivisions into software, hardware and management have already been taken. Nevertheless, software structures can provide the basis for safe hardware and sensoric structures.

After developing the basic safety-related methods and procedures, as well as developing recommendations for a guideline, the findings must be tested and applied to a special graphical programming language and sensor systems. The programming language G in the development environment LabVIEW provides the best framework for this.

**References**

[1] Ludwig Coulmann (Hrsg.) (1993): Programmvisualisierung bei Sprachen der 4. Generation - Visualisierung von NATURAL-Programmen. SpringerLink.

[2] DIN EN 61508-1, Februar 2011: DIN EN 61508-1 (VDE 0803-1):2011-02.

[3] DIN EN 61508-3, Februar 2011: DIN EN 61508-3 (VDE 0803-3):2011-02.