Hierarchization and Integration of IEC 61131-3 and IEC 61499 for Enhanced Reusability

M. Melik Merkumians, P. Gsellmann, and G. Schitter
Automation and Control Institute
TU Wien
Vienna, Austria
melik-merkumians@acin.tuwien.ac.at

Abstract—Today’s markets are short-lived and volatile. The producing industry needs to be able to react fast and cost-effective to product and process changes. Following the trend of Industry 4.0, current automation systems are designed as Cyber-Physical Production System (CPPS), which main properties are the distribution of capabilities and strict modularization to enhance reuse of off-the-shelf automation components. The currently dominant Programmable Logic Controller programming standard IEC 61131 was conceived with mainly centralistic systems in mind, which limits its use for nowadays CPPS scenarios. Focusing on the software aspect, the contribution of this work is an enhanced hierarchic development approach for automation systems, which aims to decouple all hardware from the control flow logic of an automation system for combined IEC 61131 and IEC 61499 systems. Therefore, a migration path for IEC 61131 Function Blocks into the development model of the event-driven IEC 61499 is presented, and issues are discussed. Both approaches pay special attention to the reusability of software.

Index Terms—CPPS, PLC, Modularization, IEC 61131, IEC 61499

I. INTRODUCTION

Plant engineering has become increasingly complex in recent years. The reasons for this are the increasingly demanding production processes that combine a variety of different technologies, but also the changed market conditions, with significantly higher diversity of variants, and shorter product- and process life cycles [1]. Industry and academia focused on modularization concepts to cope with these new demands. Following the trend of Industry 4.0, current automation systems are designed as Cyber-Physical Production System (CPPS), which main properties are the distribution of capabilities and strict modularization to enhance reuse of off-the-shelf automation components. Both properties can only be achieved if hardware (HW) and software (SW) are designed with this in mind. The currently dominant Programmable Logic Controller (PLC) programming standard, the IEC 61131 – Programmable controllers [2], unfortunately did not consider such highly distributed systems as CPPS, when it was first developed, as centralistic systems were sufficient back then. This limits the use of the IEC 61131 system and programming models for distributed systems, which were tried to remedy with additional external models or additions to the standard [3], [4]. The main reason for this limitation is the cyclic processing model of IEC 61131 [5], [6]. The IEC 61131 uses a cyclic processing model, which guarantees a unique system status during program processing. This regular “freezing” of the real world facilitates programming and is easy to understand for the user. At the same time, however, it is an obstacle for the modularization of the SW and the distribution of the program over several PLCs, since a uniform system state cannot be guaranteed. Even if the same cycle time is chosen for all PLCs, the individual PLC-cycles do not necessarily have to run isochronously. The PLCs can therefore have different snapshots of the system to be controlled, which can lead to errors in the overall application [7].

The IEC 61499 was developed with distribution aspects of HW and SW in mind, which also forces the modularization of SW [8]. IEC 61499 uses the Function Block (FB) as a basic programming element, which in principle similar to the IEC 61131 FB, but replaces the data-driven activation of FBs with event control. This means that IEC 61499 FBs only execute if they have received an event.

Still, as the IEC 61131 is the dominant standard for PLC development, a migration path for integrating existing PLC programs into IEC 61499 – Function blocks [9] systems has to be provided [10].

Focusing on the SW aspect, the contribution of this work are an enhanced hierarchic development approach for automation systems, which aims to decouple all HW from the control flow logic of an automation system, and a migration path for IEC 61131-based systems to be integrated into IEC 61499 systems.

II. HIERARCHIC DEVELOPMENT

Modularization and hierarchization allow automation components to be easily reused in various plant sections and applications. To be composable, each automation component must be self-sufficient and complete [11], [12]. It is important that the components do not pass or expect any requirements of a specific HW-implementation via their interfaces [12], [13]. Unfortunately, automation programs are often tightly coupled to the used HW, due to the fact how the HW is supposed to be controlled [12]. The resistor sorting system in Fig. 1 serves as an example. Resistors are filled into the vibrating conveyor as bulk good. The resistors are separated, the resistance value is measured, and based on the measurement result put in one of the two storage pallets. Following the state-of-the-art methods for hierarchizing automation systems [14]–[16], the system can be divided into three subcomponents: the measuring station,
the vibrating conveyor and separator, and the gantry robot with a vacuum gripper. The individual subcomponents can be further divided. For the portal robot-component this would be the three axes or axis drives for $x$, $y$, and $z$, and the gripping tool. On the HW side, these three components are completely decoupled, as there are no dependencies between the individual components. However, inspecting the SW-components, it is evident that the components are coupled to each other, due to expectations on the representation of process parameters. The axis drive control of the individual axes of the portal robot serves as an example (see Fig. 2). The servomotor-axis drive expects the position representation in Cartesian coordinates ($x$, $y$, or $z$ for each individual axis) as input, visible as the output _Position_ of the FB’s on the right in Fig. 2, which seems to be the obvious choice, due to the gantry robot structure. This assumption is passed on to the next higher hierarchical level via the _Controller_Interface_ interface, which expects a specific position value (_Final_Position_).

Thus, the next higher component is forced to implement this interface and is therefore strongly linked to subcomponents that expect coordinates as input. However, if the servomotor is replaced by, e.g., a pneumatic cylinder, the representation of position changed drastically, as a pneumatic cylinder is usually moved into its end positions, without intermediate positioning possible. This is also represented by its control FB in Fig. 3, where the movement is controlled by the _Extend_ and _Retract_ events to perform the respective movements. The proposed component-based design approach tackles this issue, by further decoupling the control logic from the HW implementation abstracting HW-dependent parameters into abstract symbols.

These symbols only state the intent of a control-logic action,
III. MIGRATING OF IEC 61131 APPLICATIONS TO IEC 61499

As mentioned in Section I, a migration path from classic IEC 61131 based systems to IEC 61499 systems has to be provided, as it is not economical to completely revamp existing automation systems to be able to use new technologies and methods. Still, a method is needed to integrate those legacy systems into newer methodologies, such as the IEC 61499 development model, to reap the benefits of its distributed execution model, which usually builds the base of CPPS [17]–[19]. Several approaches to either combine IEC 61131 and IEC 61499 environments have been proposed [5], [20] or the automatic transformation of IEC 61131 applications to IEC 61499 models [10], [21], [22] to provide. Aside from the migration path aspect, certain classes of automation applications are easier implemented in either IEC 61131 or IEC 61499, so the ability to mix and match is highly desired to reduce the development effort [23].

The integration approach presented by Goellmann et al. [24] not only combines IEC 61131 and IEC 61499 execution environments, but also development models. IEC 61131 TASKS are represented as specialized IEC 61499 RESOURCES, which implement the cyclic execution model of the IEC 61499, running independent of other IEC 61499 RESOURCES on the same DEVICE, as the IEC 61499 demands.

Interaction between the IEC 61131 and IEC 61499 parts is enabled via communication FBs, which respect the IEC 61131 cyclic execution model for the IEC 61131 parts and utilizing the event model for IEC 61499 RESOURCES. A very interesting property of this approach is the dual use of SFBs as non-Object-Oriented (OO) IEC 61131 FBs (see Fig. 5), which enables an easy automatic transformation from legacy IEC 61131 FBs into IEC 61499 FBs, by simply adding an

Fig. 5: The Simple Function Block (SFB) is used in a dual way to represent IEC 61499 Simple Function Blocks (SFBs) and IEC 61131 FBs simultaneously. This reduces development effort and IEC 61131 FBs can easily be transferred to the IEC 61499 [24].
transfer them, similar to the main code body of IEC 61131 FBs, as an additional algorithm and to represent the methods interface as an additional set of data in- and outputs and the respective events. This first idea seems unsatisfying, as the method could only be called from the outside, but not from the inside of an IEC 61499 FB.

REFERENCES


