A Synthesis of the standards IEC 61131 and IEC 61499 within the design system SPaS – Software Project tool for automatic control Systems –

Andreas Pretschner, Jochen Alder

University of Applied Sciences Leipzig
Institute of Process Automation and Embedded Systems
Email: andreas@pretschner.com

Abstract: The contribution gives a short description of the software engineering tool SPaS, particularly considering the graphical flow chart design. SPaS is characterized by its technology-oriented engineering manner and by the in-built verification of its control layout. The graphical flow charts are compiled to a control program written in either of the languages: Instruction List, Structured Text, C or C++. This provides syntactically correct code for various PLC products, PCs or microcontrollers. The engineering tool is able to apply function blocks to distributed systems. Control engineers sometimes understand the IEC 61499 as a new programming language similar to those defined in IEC 61131-3 [Vya07]. That assumption does not cover the whole behavior of the new standard IEC 61499. The IEC 61499 incorporates advanced software technologies, such as object oriented software paradigms, which mostly are not per default usable in the IEC 61131-3. The contribution will show a bridge between both technologies under restrictive circumstances.

1 Preface

The system *SPaS* goes back on developments of the years 1980 to 1992 and is based on three graphic models, the process decomposition graph (*PDG*), the process cycle net (*PCN*) and the process flow chart (*PFC*).

1. The process decomposition graph is considered as the beginning of the system development, which ends with the functional module representation of the subprocess[1], i.e. the IEC 61499 function block instances accordingly.
2. The process cycle net, based on a petri-net model, is considered as an important assistance for system development. It is suitable to supply the possible instances including their interactions.
3. The process flow chart is the central model for the description of the technological process and arises as an type (i.e. software class), more exactly: type function block module.

[1] contentwise a subtask in the control system

4. The structure of an automation project is designed as an project-tree according to the rules of the IEC 61131-3 [61103], whereby the configuration, necessary resources, types and tasks are graphically arranged. Program types contain functional module instances derived by type function block modules. The type function blocks are software coded via IEC 61131-3 internally and adopted to the IEC 61499 behavior externally.

5. The user programming interface is the „Structured Text“ - language in accordance with IEC 1131-3 in order to write arithmetic formulations in functions (embedded into the *PFC* or independently). The function blocks by itself are programmed graphically with an underlying state machine behavior.

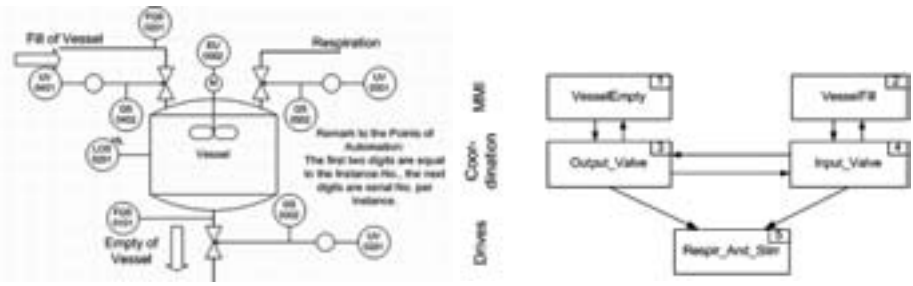
6. The SPaS tool can be considered as an „pre-compiler “which outputs different languages like Structured Text, C, C++ Siemens Instruction List and more. The special purposes of the tool are the consistent technology oriented project description and the mutable multi-language output.

The author refers to [AP07], in which the use of the Tool *SPaS* is described in detail and from which representations and illustrations were taken. The further sections of this contribution are concerned with the question, how the two standards are used and united, in order to understand an automation project more reliably and transparently.

2 Project engineering task

At the beginning stands the process, whose technological schematic diagrams should be present. The basis for this is the German standard [19283], which is based on the ISO 3511. The recommendations are obligatory [67703] as well. Based on the schematic diagrams the responsible editors are in the position to analyze subprocesses from the available entirely process. By subprocesses we e.g. understand drive controls for engines, valves, flaps and other so called servo units. The controlling of a machine can appear as subprocess as well, without regarding the above specified drive controls. These controls receive command signals from the supervisor control, in order to fulfill certain tasks of the machine. Such subprocesses belong into the range of the coordination. In addition subprocesses of man-machine-interface (MMI), such as signaling or congestion controls still arise. The mentioned procedure we call process decomposition and get connected from it the process decomposition graph (*PDG*) with subprocesses and coupling relations between them; this is the first step on the way for the functional module representation.

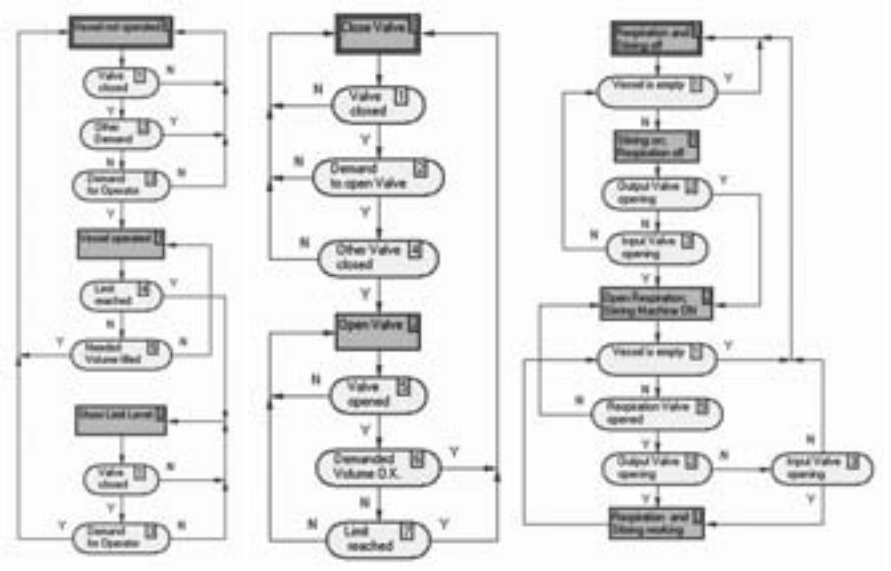
The figure 1a shows the cutout from a process engineering plant as technological schematic diagram. The process decomposition graph (PZG) is to be seen in its initial form made of figure 1b. The next step consists of developing the functional operational sequence of the subprocesses in which also the signal exchange with neighboring subprocesses is to be considered. We call this couplings between the subprocesses and these signals *coupling signals*. The subprocesses are named, which are instance names too, and numbered additionally. The associated types must be designed now in such a way that they seize as much



(a) Batch process (b) Process decomposition graph

Figure 1: Process Description

as possible instances concerning to the signal processing. In the example „vessel“ are it three types, the vessel operation - for subprocesses 1 and 2 - and the controlling of a valve - for the subprocesses 3 and 4 - as well as a type for the subprocess 5. As already mentioned, the process flow chart[2], further with *PFC* shortened, specifies the subprocesses concerning their data processing. We do not intend to describe the development of the *PFC* here, since this would blow up the frame of the article, but show the results in the figures 2a,2b and 2c.



(a) Vessel operating (b) Vessel type (c) Mechanical Stirrer and respiration

Figure 2: Process Flow Charts

[2]The *PFC* is in the sense of the IEC 61499 a basic function block type, even if a decomposition of a *PFC* is usually possible. It is theoretically a not clocked automaton, which is appropriate also for its graphical structure.

The designations of the graphic elements and their meaning are briefly mentioned:

- The rectangular fields are called OPERATION and state with their designation, what has to happen in the subprocess gradually. To the OPERATION belong variables, like control actions and coupling variables to neighbor subprocesses. They are called in summary OPERATION VARIABLE, which are not visible in the operation element at the first glance[3]. Each operation variable is binary evaluated, thus a boolean variable.
- The long stretched graphics elements outlined with semi-circles, are called PROCESS VARIABLE and state with their designation, which input variables from the regarded subprocess are expected. Process variables may be coupling variables additionally, getting their input from neighboring subprocesses. Each process variable is a boolean size and therefore the values are YES or NO.
- Beyond the mentioned basic characteristics of the operation and process variables there also can be used embedded control functions. These control functions are linked with many arithmetic functions, additionally time and counting components as well as edge identification, belonging to process and operation variables.
- To each operation belongs at least one path of process variables, which begins and ends to the same OPERATION. Every such path is called STABILITY PATH. Other paths end at the respective subsequent operation and are called TRANSITION PATH. On the condition of a stability path the operation will not leave, the contained operation variables affects the process – steer it – until a transition path is fulfilled by the process. The quantity of all stability paths for operation is called STABILITY PROCESS STATE.
- Operations in relation with their stability process states are called SITUATIONs. Every such situation is assigned to an automaton. The doubly framed operation marks the initial operation and the initial situation too.
- The similarity to ECC (Event Execution Chart of IEC 61499) is obvious. The special purpose of using PFCs is their technological interpretation.

In figure 2a the „Demand for Operator“ means storing a certain volume, accordingly the path tracking is allowed by the other requirements. „Limit reached“ meant, the vessel is full or empty respectively, so that the requested volume cannot be served any longer. In figure 2b the „Demand to open the Valve“is a coupling signal, which have to be delivered by the vessel-type operation. The vessel instance, concerned to the mentioned operation, serves with the boolean value 1, if supply must, otherwise with 0.

Finally the figure 2c shows the process flow chart for the two drives of the respiration valve and the stirrer. One can recognize clearly the steering control of the coupling variables „Output Valve opening“ and „Input Valve opening“ . The formulation „...opening“ is to say the fact that the instruction to open is given, whereby is presupposed that each

[3]The Tool SPaS offers a fast announcement or an editing to the operation variable.

coupling variable is always present statically with the value YES and changes therefore in the example only on NO if the valves must be closed. Thus the types are briefly considered and over the coupling variables suggested their cooperation. The instances designated in figure 1b must get, of course, the real, actual variables valid for its assignment. The result is contained in figure 3, whose contents are to be described now and compared with [14903]:

- Because any hierarchical representation, as it is to be recognized in figure 1b, has no theoretical existence, and the arrangement of the initial and output variables prescribed in the standard [IEC1499] would to be kept only rarely, the arrangement of the instances takes place opposite to the *PDG* in signal direction. In addition it seems very meaningful to follow the signal flow from left to right in the representation.
- The designation (name) of the instance from figure 1b has attached the counting number, with which the subprocess name and/or instance name is considered as final and uniquely fixed. The formal variables are inside the functional module symbol, whereby the data types stand framed thereby.

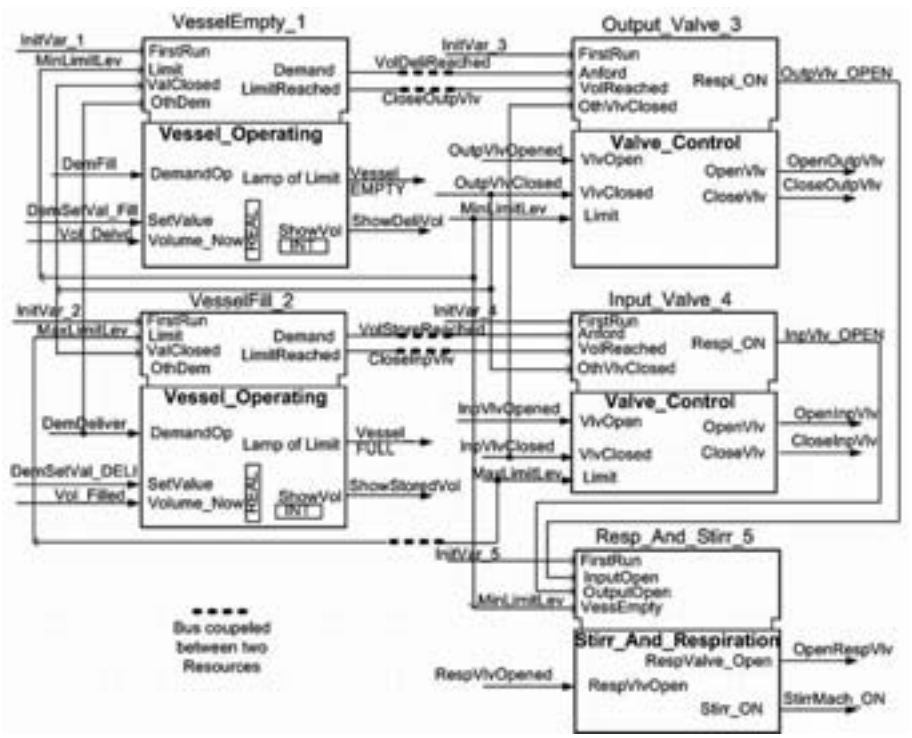


Figure 3: Function Block representation of the system

The actual variables depending on the instance are registered outside of the functional modules. The characteristics of this variables are alike to the type variables (concerning the data types). Which addresses are assigned to is here without meaning, because

they are distributed either locally in the PROGRAM „VesselControl“ or globally in the RESOURCE. To get a better overview of the variables, which do not originate from the associated PROGRAM, they should be marked if e.g. the initial variables are defined there: „VesselControl. InitVar_1“ . The event flow[4] is shown by connecting lines, which open arrows are helpful to distinguish from process data flow plotted through full arrows at the connecting lines. Those are pure outward appearances, which have nothing to do with the standard. Before we deal with contents of the standards, the project tree is presented in the figure 4a. From top to bottom the symbols mean:

- project root: *STORE_VESSEL_2_Res*,
- configuration: *Control of Vessel by 2 Ressources*,
- two resources: Man Machine Interface *MMI* and Valves and Vessel Control *Val_Vessel_Control*,
- task: in each ressource *Task_1*,
- function block types: *PCF*s in figures 2a, 2b and 2c
- program type: for each ressource, *Drives_Control* and *MMI_Activity*,
- function block instances: subprocesses, the five instances as in figure 3 shown,
- program instance: one in each ressource running with Task_1 initialized.

(a) Project tree (b) Initialization by example

Figure 4: Example Project Tree

In SPaS several resources can be registered in a project, each ressource can contain several programs, as well several tasks. Since the instantiating of a program arises as a result of the allocation to a task, the necessary interrupts can be engineered in a simple manner. Contentwise the subprocesses are absolutely alike in the project tree in figure 4a and in the functional module representation figure 3. The difference is only outward in the graphical

[4]Event variables, we say coupling variables, belong too and are to be understood in the sense of a coordination control.

description. The PROGRAM[5] „VesselControl“ is for itself seen a „composite function block“ (briefly: CFB) in accordance with 1.3.16 of the IEC 61499-1. If it is helpful to the overview, more such PROGRAMs can be introduced. In the example of figure 4a, are introduced 2 valve-instances as a CFB, Input_Valve_4 and Output_Valve_3 as a second CFB and also Resp_And_Stirr_5 alone in a further CFB. The fact of the packaging means to ensure in the project that into all CFB, thus in each PROGRAM, the necessary actual variables are at the disposal. Resources with its global variables, as suggested already above, serve this purpose or you would bring in derived data types (DDT). Defined variables in ressources stand then for all PROGRAMs for order. DDT instances, in order to be able to offer global variables, must stand below resources (thus not in a PROGRAM). At that point the most critical part in competition of both standards - using of global declared variables - came up. The conventional way of data access's stands against the object oriented data encapsulation, the event driven algorithms stand against the cyclic control program behavior of standard PLCs. Both design paradims have their proof of entitlement. Though often standard control applications are designed poorly. Unnecessarily much variables are designed globally. Using the SPaS design approach these defects cannot arise because of strictly bounding the global variable usage to input / output signals and coupling variables. The direct global variable access out of a program or an function block instance is forbidden. The way to an pure IEC 61499 design is now not far - just using the service interface function blocks, event function blocks and other adapters. We do not continue to pursue the different designations of several co-operating functional modules, even if they are distributed on several resources, because behind them you could not found new project qualities, which are not visible by the view of the cooperation of the instances by itself. All further ones are technical solution methods, which require however occasionally special system blocks, which must be included into the project tree. The translation forms the conclusion of a project engineering procedure into a language of the target machine. SPaS generates at present STEP 7, C, C++ and ST in accordance with IEC 61131-3. This procedure runs always related to resources, if the configuration in the project tree is marked and the translation is started. Each ressources can therefore have another ouput-language.

3 Relations to the standards

Because the process initialization is realized implicitly by the pre-compiler, no special initialization sequence is provided in figure 4a. However it works with cold start only. For each *PFC* a local variable „FirstRun“ , which gets „TRUE“ at the beginning of the initial value assignment, is added additionally. All instances are processed with the FirstRun-procedure in order of the project tree. The initial state of each instance is set, all other state reset. Though the warm start must be organized in the project expressly, so that after the stop of ressources the initialization process is feasible. The FUNCTION „InitState“ contains the variable „InitVar“ for all instances, as it is shown in figure 4b. The call of this function is the solely content of the PROGRAM „Init_Program“ . Here takes exactly place what happens in the IEC 61499 with INIT and INITO. The variable „FirstRun“ must be

[5]Program type, as it is designated in the IEC 61131-3

inserted into each *PFC*, thus into each type explicitly. More smart solutions are possible like application evolution shown in [SSSZ07] by adopting methods from the IEC61499. All instances are called in the PROGRAM (see the PROGRAM with Task_1) after initialization. Every now and then the sequence of the instances is of importance. There can be obligations to have to specify the call sequence by reasons of process control and the coupled instances. In principle, the calling sequence is the same, as of the PROGRAM. The obtained source code is organized in such a way that after two runs all results are present. That is valid for coupled instances also.

4 Conclusions

The contribution gives a short introduction to the software design tool SPaS and its underlying design paradigms. A certain level of function block compatibility was shown to the IEC 61499 function block design. Applying the tool's strict software constraints leads to an object oriented software design and gives further the chance to approach IEC 61499 methods later on. Contrary to [HOL, SZ09] the SPaS tool is software technically based on the IEC61131 design methods, and only adopted partwise to the IEC 61499 function block design. The multi-language facility of the generated software source code provides a wide variety of target compiler or cross compiler, runnable on different hardware targets like embedded Linux boards(ARM, PowerPC or x86 Computers-On-Modules, PC/104+ and ATX boards) as also conventional PLCs (SIEMENS, PHOENIX CONTACT, B&R).

References

[14903] DIN IEC 1499-1. Function blocks for industrial-process measurement and control. Technical report, DIN Deutsches Institut für Normung e.V., 2003.

[19283] DIN 19227-1. Graphische Symbole und Kennbuchstaben für die Prozessleittechnik. Technical report, DIN Deutsches Institut für Normung e.V., 1983.

[61103] DIN EN 61131-3:2003. Programmierbare Steuerungen, Teil 3: Programmiersprachen. Technical report, DIN Deutsches Institut für Normung e.V., 2003.

[67703] DIN 6779-12. Kennzeichnungssystematik für technische Produkte und technische Produktdokumentationen, Teil 12: Bauwerke und technische Gebäudeausrüstung. Technical report, DIN Deutsches Institut für Normung e.V., 2003.

[AP07] J. Alder and A. Pretschner. *Prozess-Steuerungen Projektierung und Inbetriebnahme mit dem Softwaretool SPaS*. Springer Verlag, 2007.

[HOL] Inc. HOLOBLOC. FBDK - The Function Block Development Kit. http://www.holobloc.com/doc/fbdk/index.htm.

[SSSZ07] Heinrich Steiniger, Dr. Thomas Strasser, Christoph Sünder, and Alois Zoitl. Evolution Control Environment for Distributed Automation Components. *Computer and Automation*, 11, 2007.

[SZ09] Thomas Strasser and Alois Zoitl. Framework for Distributed Industrial Automation and Control. Technical report, Vienna University of Technology, PROFACTOR GmbH, 2009.

[Vya07] Valeriy Vyatkin. *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*. ISBN 978-0-9792343-0-9. ISA - Instrumentation, Systems and Automation Society, 2007.