This regular paper was presented as part of the main technical program at IEEE ETFA'2011

Object oriented extensions of IEC 61131-3 as an enabling technology of software product lines

Nikolaos Papakonstantinou, Seppo Sierla, Kari Koskinen
Department of Automation and Systems Technology
Aalto University
00076, Aalto, Finland
nikolaos.papakonstantinou@tkk.fi, seppo.sierla@tkk.fi, kari.o.koskinen@tkk.fi

Abstract

Software product lines (SPL) relying on UML technology have been a breakthrough in software reuse in the IT domain. In the industrial automation domain, SPL are not yet established in industrial practice. One reason for this is that conventional function block programming techniques do not adequately support SPL architecture definition and product configuration, while UML tools are not industrially accepted for control software development. In this paper, the use of object oriented (OO) extensions of IEC 61131-3 are used to bridge this gap. The SPL architecture and product specifications are expressed as UML class diagrams, which serve as straightforward specifications for configuring the IEC 61131-3 control application with OO extensions. A product configurator tool has been developed using PLCopen XML technology to support the generation of an executable IEC 61131-3 application according to chosen product options. The approach is demonstrated using a mobile elevating working platform as a case study.

1. Introduction

Software product lines (SPL) are an approach for software reuse to be applied for a set of applications, in which there is similar considerable similarity. Functionality that is not shared by all applications is packaged behind interfaces known as variation points, so that there are several optional software modules that implement the interface. Products are configured by specifying one of the options for each variation point. In the IT domain, SPL emerged as a breakthrough in software reuse during the first decade of 2000 [1-3].

The UML and object oriented programming languages have been very broadly adopted by researchers advocating the SPL approach, as they provide appropriate constructs for describing and implementing variation points. Despite the benefits of object-orientation, industrial practice in the automation domain still relies heavily on PLCs (programmable logic controllers) programmed in IEC 61131-3, so object-orientated extensions to the standard have been proposed to reap benefits such as improved reusability [4]. The purpose of this paper is to exploit these extensions to enable the use of SPL technology for generating IEC 61131-3 control applications.

Object-oriented extensions enable the straightforward implementation of PLC programs based on SPL architectural design expressed in UML, but it is also necessary to automatically configure a PLC program according to client options. A product configurator tool has been developed, which reads a PLCopen XML file describing the architecture, variation points and library of options. The user simply needs to specify the option for each variation point, and an executable IEC 61131-3 application is generated in the PLCopen XML format.

This paper is structured as follows. Section 2 reviews research in the fields of SPL and object-oriented extensions of IEC 61131-3, since our contribution builds on these two bodies of knowledge. Section 3 presents the case study, a mobile elevating work platform, which is used to illustrate a SPL architecture using UML2 structured class diagrams. In section 4, object-oriented extensions to IEC 61131-3 are applied in order to implement the architecture with greater readability, maintainability and reusability than would be possible with conventional IEC 61131-3. Section 5 presents a product configurator tool, which can be used to transform the product line level IEC 61131-3 PLCopen XML file to an executable application according to the product options selected by the user. Section 6 concludes the paper with a discussion on industrial benefits, the scalability of the proposed approach and further development needed to obtain an industrial strength machine control software product line.

978-1-4577-0018-7/11/$26.00

2. Related research

In this section state of the art is presented regarding the field of software product lines and object oriented paradigm in relation to IEC 61131 standard for PLC programming.

2.1. Software product lines

In the software engineering domain the SPL approach is considered a key technology for producing software products when time to market and software reuse are critical factors for the success of a product. The business advantages of the transition to a product line approach to software are discussed in [5] through a case study of the development of software for an ATM machine. A migration path and experiences gained from porting two large legacy industrial control applications to an SPL framework are described in [6].

The Object Oriented paradigm is a good match with the component driven architecture of SPLs. Object Oriented concepts and Feature Modeling techniques are combined in an Object Oriented Feature Modeling (OOFM) approach in [7] so that software variability is better captured and documented. Model Driven Engineering concepts are applied to software product lines in [8] for producing configurable applications for the communications equipment domain. SysML is used in [9] to model system variants within a product line.

A formal approach for Feature Configuration Modeling is introduced in [10], where rules and constrains guide the selection of features. A methodology for customization of software to achieve non-functional requirements that are affected by many different choices of the variation points is presented in [11].

Tools are developed to enable better handling of Software Product lines. A commercial product line engineering solution is presented in [12] that focuses on the differences on the development of traditional applications and SPL software. A visual tool for managing the configuration of complicated software product lines with many points of variation is introduced in [13]. Another tool (MoSPL) that handles version management at the component level and logical constraints and derivation relations among components is introduced in [14]. The Visual and Interactive Tool for Feature Configuration (VISIT-FC) is introduced in [15], to facilitate the configuration of software products with a large number of variation points.

In the industrial software domain an approach towards product line industrial software development based on the IEC 61499 programming standard is presented in [16], although actual generation of executable IEC 61499 applications is left for further work. However, IEC 61499 follows only partially the object oriented paradigm and has not gained widespread industrial acceptance [17]. The IEC 61131-3 programming standard has been widely accepted in industry [4] but the SPL approach in this context has not yet been researched and no SPL tools to support IEC 61131-3 application generation are available.

2.2. Object-oriented extensions of IEC 61131-3

Attempts have been made to introduce or to integrate Object Oriented concepts to control software. A methodology for applying formal design and verification techniques to the development of industrial control systems is described in [18]. UML is used to capture the software design and provides the information needed for the verification phase.

A presentation of the Object oriented extensions of IEC 61131 [4] and how they are implemented in the CodeSys PLC programming environment are discussed in [19]. The mapping of UML class diagrams to hybrid function blocks (function blocks with object oriented extensions) is also described. An alternative object oriented paradigm is presented in [20]. In [21] a description of a methodology for using object oriented software development paradigm to automatically generate Instruction List code from Java code is introduced.

An object oriented tool for programming control applications generating IEC 61131 code (without the object oriented extensions) is presented in [22]. Also in [23] an attempt is made to produce IEC 61131 programs following an object oriented philosophy.

It is clear that combining object orientation with the industrially accepted IEC 61131-3 is a promising development recognized by many researchers. However, the potential of applying extensions to the standard to support the SPL approach and IEC 61131-3 application generation has not yet been researched. This paper proposes a SPL approach for machine control systems. As a proof-of-concept, a tool for generating IEC 61131-3 applications based on user choices of variation point options is presented. The tool exploits the PLCopen XML import/export format and object oriented extensions of IEC 61131-3.

3. Using UML2 structured class diagrams to describe software product line architecture

The application of SPL to the development of machine control software is demonstrated by using a mobile elevating work platform as a case study, see fig. 1. The platform supports vertical and horizontal movement and also supports the automatic movement to a stored position.

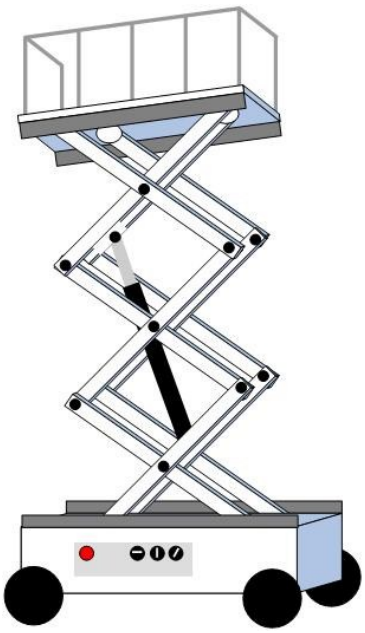

Figure 1, Concept figure of the case study, a mobile elevating work platform using a scissor lift mechanism

Starting the design of the product line it is decided that two variation points are going to be available. The first concerns the control of the rising or lowering of the platform, with the options of either a joystick or separate up and down buttons. The second variation point is about the control algorithm for the automatic movement to a stored position. The Control2D permits unlimited two dimensional automatic movement while the Control1D algorithm permits the automatic movement only vertically and only if the platform is in its horizontal home position (a safer variation of Control2D). Since all combinations are allowed, there are a total of four product variations that may be generated from the product line.

The software product line is represented by a triangle containing the two variation points as a purple circle and a blue pentagon of fig. 2. The area outside the variation points represents software that is common to all precuts. The purple circle (the variation point representing the human machine interaction option) has two options represented by a red circle (joystick) and an orange circle (buttons). The blue pentagon (the variation point representing the control algorithm option) has two options: the Control1D algorithm (dark green) or the Control2D (light green). The Programmable Electronic Control System (PECS) diagram for the variation that uses the Control2D algorithm and the Joystick for input is presented in fig.3.

The problem in SPL design is to represent the concept illustrated in the triangle figure with a notation that describes software constructs at an appropriate

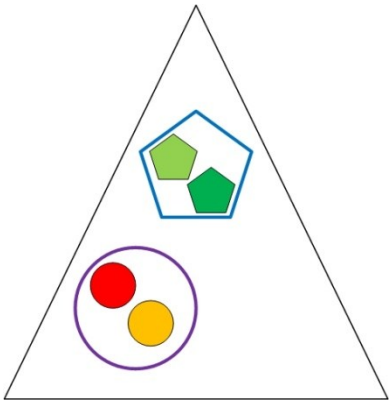

Figure 2, Conceptual representation of the software product line. Violet circle represents the variation point for the user interface device. Blue pentagon represents the variation point for the automatic control algorithm. Figure adapted from: C.W. Krueger, Software Mass Customization, BigLever Software Inc. March 2006, www.biglever.com

abstraction level. UML2 structured class diagrams have been chosen (fig. 4). There are two superclasses that capture the commonalities between the different options for the variation points. Optional implementations of the variation points are in derived classes, which can override the implementations of methods of their superclass and potentially add additional inputs/outputs. The "Joystick" and "Buttons" classes extend the "VerticalUserCommand" parent class. Similarly the "Control1D" and "Control2D" classes extend the "ModeCtrl" parent class.

The SPL designer will produce UML class diagrams, after which it will be necessary to configure

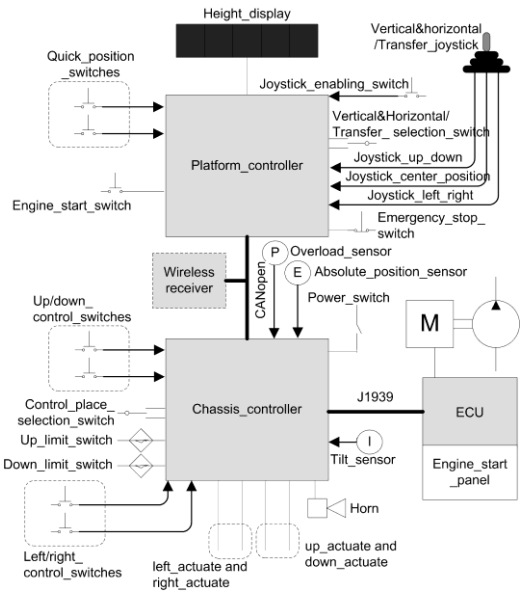

Figure 3, Programmable electronic control system for the product variation with a joystick and 2D control algorithm

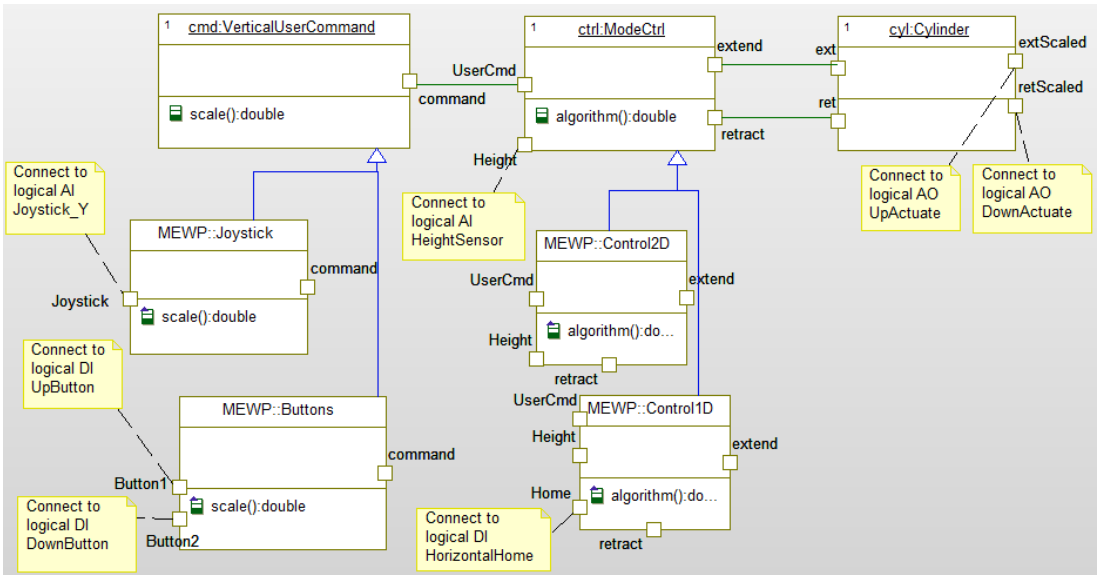

Figure 4, UML2 class diagram capturing the software product line architecture. Generalization relationships are used to identify alternative implementations of a variation point

an IEC 61131-3 application based on selected variation point options. This has not been previously attempted, and the remainder of this paper proposes a solution to this using object oriented extensions of IEC 61131 and the PLCopen XML import/export format.

4. Using object oriented extension to IEC 61131-3 to implement the architecture

The IEC 61131 with its latest addition of object oriented extensions is a good match to implement the model described in Section 3. The CodeSys is a commercial IEC 61131 platform that supports its object oriented extensions. Our approach relies on the possibility of a Function Block (FB) to "extend" other function blocks, override their methods and add new inputs or outputs; this makes it possible to implement the generalization relationships in the UML diagram (fig. 4). The connections between ports in the UML2 structured class diagram are mapped to connections between FB inputs and outputs in IEC 61131-3.

In our case, the IEC 61131-3 application was created manually, using the UML diagram as

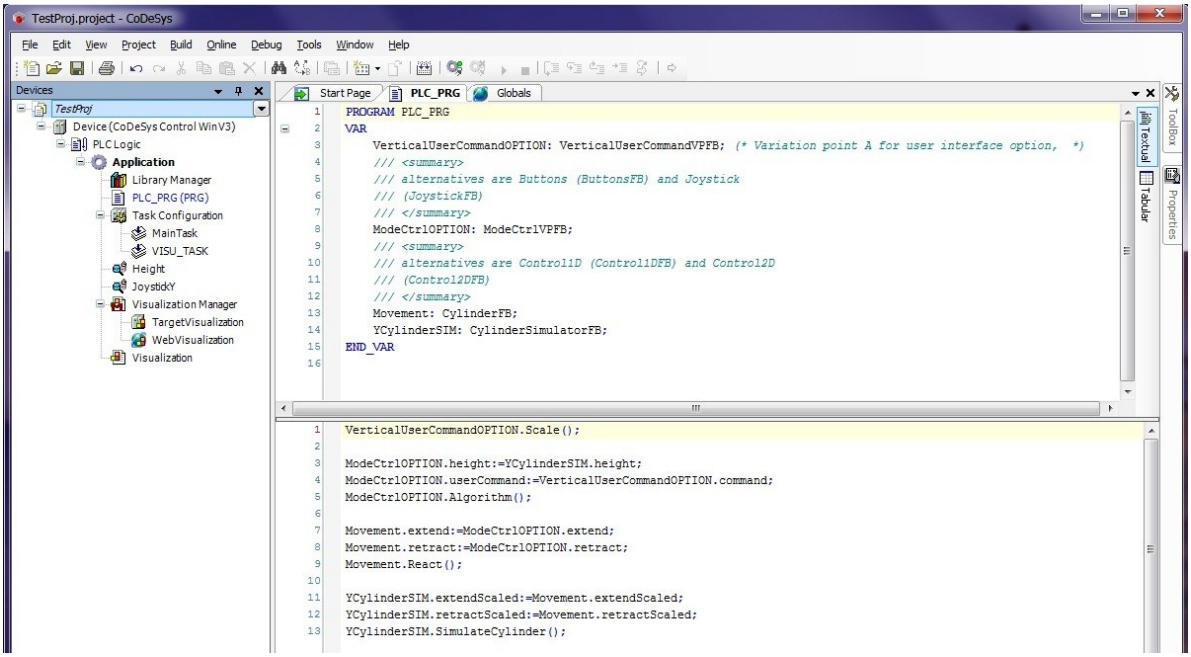

Figure 5, Product line level IEC 61131-3 application in CoDeSys, which does not contain any product specific configuration information

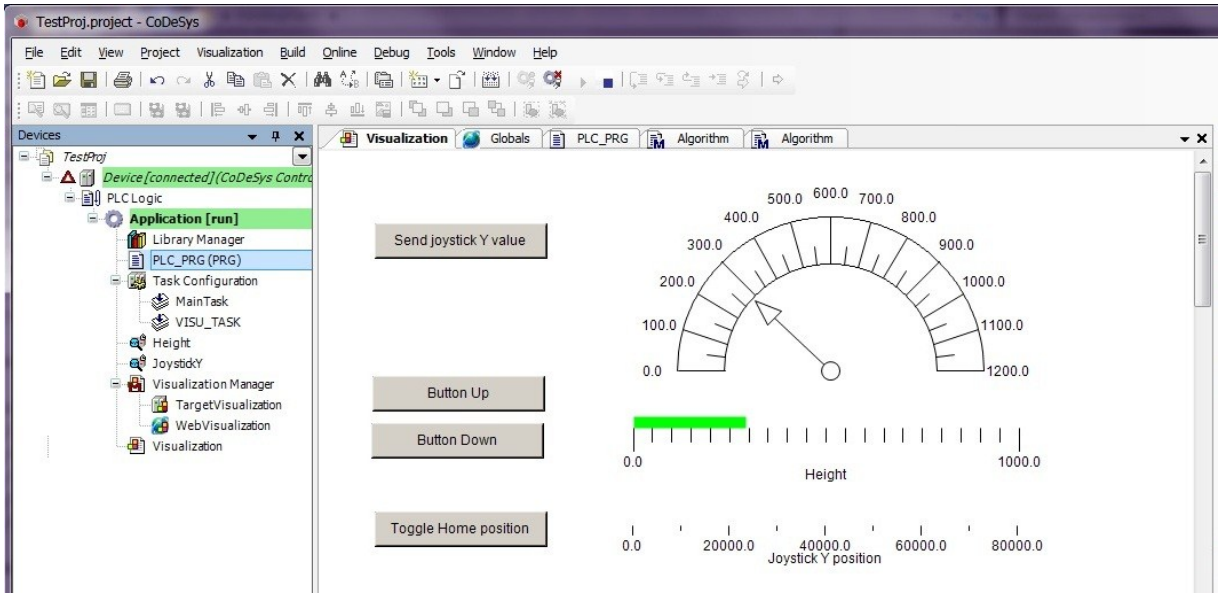

Figure 6, HMI for testing generated product specific control software. Boxes such as Button Up will be inactive if the Buttons option was not selected

specification. This only needs to be done once for the entire product line, after which products may be generated by selecting options for each variation point. When UML tool code generators are able to exploit IEC 61131-3 object-oriented extensions, the manual transition from UML to IEC 61131-3 is no longer needed. According to [19], development work into this direction has already been undertaken in projects developing UML modeling capabilities to CoDeSys V3.

In this example, the "VerticalUserCommandVPFB" is extended by the "JoystickFB" and by the "ButtonsFB". The method "Scale" is overridden with specific implementation in both option FBs. In "JoystickFB" an additional input called "JoystickYPosition" is added and in "ButtonsFB" the inputs "UpButton"/"DownButton" are added. In a similar way the "ModeCtrlVPFB" is extended by the "Control1DFB" and by the "Control2DFB". The method "Algorithm" is overridden with specific implementation in both option FBs. In "Control1DFB" an additional input called "HomePosition" is added. In our case study no new methods were added to the FBs that implement options. The example project as it was implemented in the CodeSys v3.4 development is presented in fig. 5.

Additionally to the control application, a cylinder simulator FB was used to test the example application (the "CylinderSimulatorFB"). A HMI was developed within the CodeSys environment in order to send inputs to the control application and monitor the outputs of the simulated cylinder FB (fig. 6).

5. A product configurator tool for generating IEC 61131-3 applications

A product configurator tool for generating IEC 61131-3 applications, ready to be compiled and run on a target platform, was developed. This tool can automatically read the variation points and options in an IEC 61131 generic project exported as a PLCopen XML file, and support the user for making the configuration of the application. Then the product specific project can be generated and imported into the development environment through a PLCopen XML file (fig. 7).

The tool exploits the IEC 61131 object oriented extensions in addition to the PLCopen XML import/output functionality of the CodeSys 3.4 platform. A generic project is created, with superclasses in the place of variation points (the VerticalUserCommandVPFB and ModeCtrlVPFB mentioned in section 4) and derived classes as library FBs. The application and its FB library is exported to a PLCopen XML file. The product configurator tool can parse this generic project xml file and by tracing the keyword "EXTENDS" in the FB declarations, a list of variation points and configuration options can be generated (see fig. 8). After a product specific option is selected for each variation point, the tool is able to generate a new PLCopen xml file that can then be imported to a compatible PLC programming environment, such as CoDeSys, be built and transferred to a target platform. In our examples the only changes that are made to the program itself are in

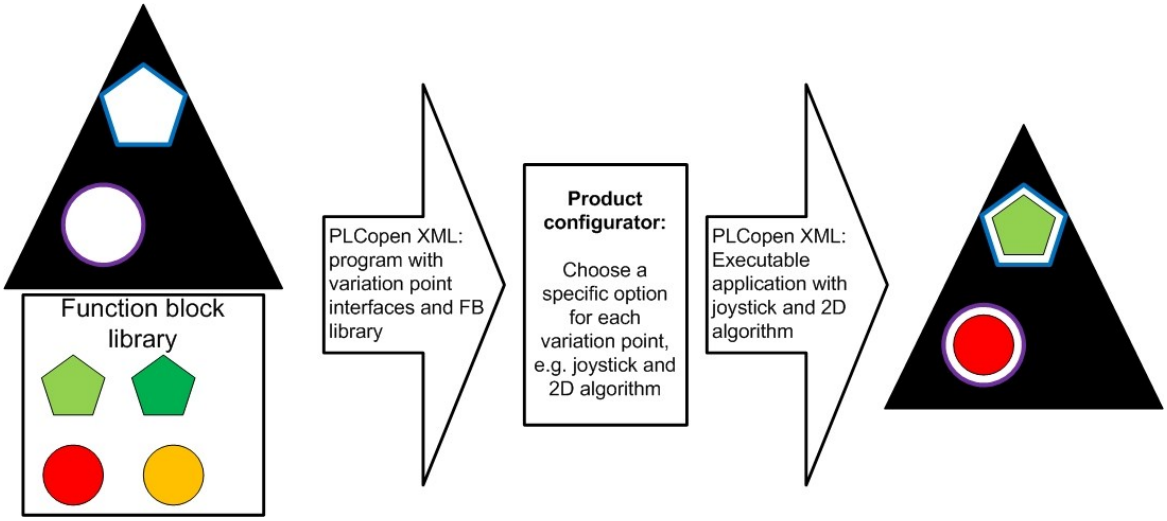

Figure 7, Product configuration process

the variable declaration section, where the correct types are chosen by the tool for the FBs that implement the variation points, e.g. the "VerticalUserCommandOption" is an instance of "ButtonsFB" see fig. 9.

6. Conclusions

In this paper, the Software Product Line technology was applied to IEC 61131-3 machine control applications. The descriptive power of UML2 structured class diagrams was used to describe the product line architecture, the variation points, and the options at each point. A configuration support tool was presented, which can be used by personnel not familiar with UML or IEC 61131-3 technology. Without breaking the object oriented paradigm, the tool modifies the generic PLCopen XML file, resulting in a complete application that can be imported into any development environment that supports PLCopen XML and object oriented extensions of IEC 61131-3.

This approach was demonstrated with a case study having 2 variation points with 2 options each, resulting in 4 combinations, each of which represents a potential product. The approach and product configurator do not make any assumptions about that application, so the technology presented here can be used with a large number of variation points. The practical limit to scalability would be the testing of the large number of product variants, and our further research involves applying automatic SPL testing techniques to IEC 61131-3 machine control systems.

Another problem which is not evident from our case study and which is also left for further research is that all combinations may not be valid. For example, consider having the hydraulic cylinder (fig. 4) as a variation point with two options: binary or proportional valves. The binary valve enables on/off control only, so the joystick should not be allowed as a user interface device option if the binary valve option is selected. A future version of the tool can import a rules base containing such constraints.

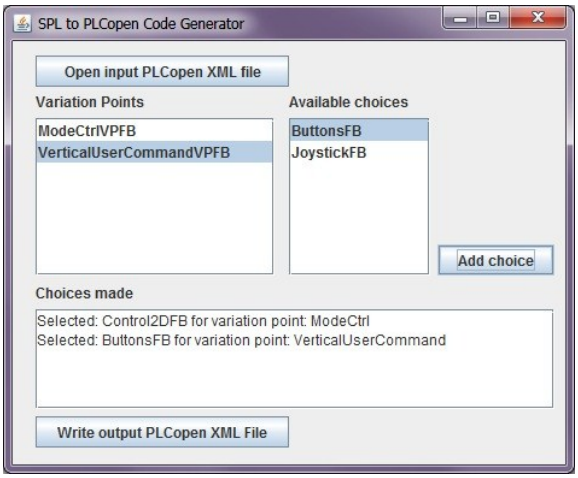

Figure 8, The product configurator tool. "Variation points" and "Available choices" are populated automatically based on the information in the input PLCopen XML file

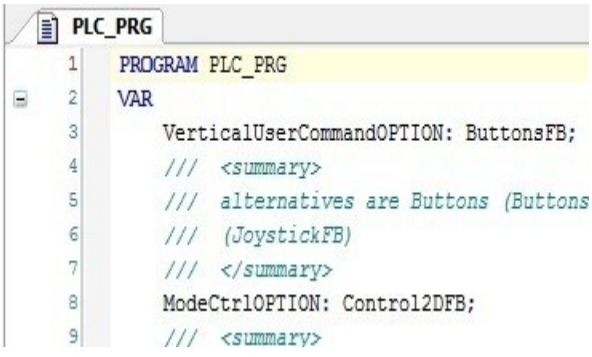

Figure 9, An excerpt from the automatically modified product code. FB type declarations match the user selected product specific options in Fig. 8

References

[1] J.D. McGregor, *et al.*, "Initiating software product lines," *IEEE Software,* vol. 19, p. 24, 2002.

[2] C. Krueger, "Eliminating the Adoption Barrier," *IEEE Software,* vol. 19, p. 29, 2002.

[3] V. R. Ommering, "Software Reuse in Product Populations," *IEEE Transactions on software engineering,* vol. 31, p. 537, 2005.

[4] B. Werner, "Object-oriented extensions for iec 61131-3," *Industrial Electronics Magazine, IEEE,* vol. 3, p. 36, 2009.

[5] L. M. Northrop, "Software product lines: reuse that makes business sense," presented at the Software Engineering Conference, Sydney, NSW, 2006.

[6] H.P. Breivold, *et al.*, "Migrating Industrial Systems towards Software Product Lines: Experiences and Observations through Case Studies," presented at the Software Engineering and Advanced Applications, Parma, 2008.

[7] V.T. Sarinho and A. L. Apolinario, "Combining feature modeling and Object Oriented concepts to manage the software variability," presented at the IEEE International Conference on Information Reuse and Integration, Las Vegas, NV, 2010.

[8] B. Trask, *et al.*, "Using model-driven engineering to complement software product line engineering in developing software defined radio components and applications," presented at the 10th International Software Product Line Conference, Baltimore, MD, 2006.

[9] C.R. Maga and N. Jazdi, "An approach for modeling variants of industrial automation systems," presented at the IEEE International Conference on Automation Quality and Testing Robotics, Cluj-Napoca, 2010.

[10] Li Yi-yuan, *et al.*, "Feature configuration modeling and problem solving for software product line," presented at the Second International Multi-Symposiums on Computer and Computational Sciences, Iowa City, IA, 2007.

[11] J. Sincero, *et al.*, "Approaching Non-functional Properties of Software Product Lines: Learning from Products," presented at the 17th Asia Pacific Software Engineering Conference, Sydney, NSW, 2010.

[12] C. W. Krueger, "Software product line reuse in practice," presented at the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, Richardson, TX, 2000.

[13] G. Botterweck, *et al.*, "Visual Tool Support for Configuring and Understanding Software Product Lines," presented at the 12th International Software Product Line Conference, Limerick, 2008.

[14] Thao Cheng, *et al.*, "Software Configuration Management for Product Derivation in Software Product Families," presented at the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, Belfast, 2008.

[15] G. Botterweck, *et al.*, "Visual Configuration in Automotive Software Product Lines," presented at the 32nd Annual IEEE International Computer Software and Applications Conference, Turku, 2008.

[16] R. Froschauer, *et al.*, "Managing the Life-cycle of Industrial Automation Systems with Product Line Variability Models," presented at the 34th Euromicro Conference Software Engineering and Advanced Applications, Parma, 2008.

[17] K. Thramboulidis, "Different perspectives [Face to Face; "IEC 61499 function block model: Facts and fallacies"]," *IEEE Industrial Electronics Magazine,* vol. 3, p. 7, 2009.

[18] M. Bonfe and C. Fantuzzi, "Design and verification of mechatronic object-oriented models for industrial control systems," presented at the IEEE Conference Emerging Technologies and Factory Automation, Lisbon, 2003.

[19] D. Witsch and B. Vogel-Heuser, "Close integration between UML and IEC 61131-3: New possibilities through object-oriented extensions," presented at the IEEE Conference on Emerging Technologies & Factory Automation, Mallorca, 2009.

[20] V.M. González, *et al.*, "MIOOP. An object oriented programming paradigm approach on the IEC 61131 standard," in *IEEE Conference on Emerging Technologies and Factory Automation*, Bilbao, 2010.

[21] G. Aiello, *et al.*, "An Agile methodology for Manufacturing Control Systems development," presented at the 5th IEEE International Conference on Industrial Informatics, Vienna, 2007.

[22] S. Ono Kajihara, *et al.*, "Development and products of the object-oriented engineering tool for the integrated controller based on IEC 61131-3," presented at the SICE Annual Conference Sapporo, 2004.

[23] M. Bonfe and C. Fantuzzi, "Object-oriented approach to PLC software design for a

manufacture machinery using IEC 61131-3 norm languages," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Como, 2001.