Towards Delta-Oriented Variability Modeling for IEC 61499

Hafiyyan Sayyid Fadhlillah*†, Bianca Wiesmayr*, Michael Oberlehner*,
Rick Rabiser*†, Alois Zoitl*†, *Member, IEEE*
†*CDL VaSiCS*
**LIT CPS Lab, Johannes Kepler University Linz*
Linz, Austria
{hafiyyan.fadhlillah, bianca.wiesmayr, michael.oberlehner, rick.rabiser, alois.zoitl}@jku.at

***Abstract*—Modern production plants are complex Cyber-Physical Production Systems with an ever-increasing share of software controlling and automating their operation. The customization of these systems to the needs of their customers and their frequent evolution over a typically long life-cycle result in a plethora of variants to be managed. However, reuse still is mainly done in an opportunistic way, relying on copy-paste-modify strategies. More systematic, strategic reuse would help to reduce costs and time to market, but requires approaches that can be applied to domain-specific languages for developing distributed control software. In this paper, we propose an approach to manage variability in IEC 61499-based systems using delta-oriented variability modeling. We discuss open challenges and outline a research agenda for variability management in IEC 61499.**

***Index Terms*—IEC 61499, Variability Modeling, Model-driven Development, Software Product Lines, Cyber-Physical Production Systems**

I. Introduction

Developing and maintaining modern production plants requires innovative methods for software engineering because such systems comprise complex automation software. A major challenge arises from the high variability of such Cyber-Physical Production Systems (CPPSs) [1]. Each production plant is designed as a unique system tailored to customer needs resulting in a plethora of variants over time, with commonalities and differences. Furthermore, the long life cycles of the physical components result in frequent adaptations of the (deployed) automation software [2]. In industry, copy-paste-modify is a frequently applied strategy for reusing software among variants. Systematically managing variants and versions would reduce software engineering costs and enable more planned, strategic reuse [3].

In software engineering, the concept of software product lines (SPL) [4] has been proposed to support systematic, large-scale reuse and address the need for managing variants [3]. An SPL is centered on the idea of creating variants of software by systematically reusing and customizing artifacts based on a common platform. A product line describes the commonalities and variability of a system for a particular target market or customer (type), typically in explicit variability models [5]. Together with specific configuration mechanisms, these models are the basis to support deriving customized products.

978-1-7281-2989-1/21/$31.00 ©2021 IEEE

IEC 61499 [6] is a domain-specific language for modeling distributed control software. It defines a hardware-independent application model that can be mapped to a system configuration and thus supports adapting the control hardware. The application itself is comprised of modular software components: Function Blocks (FBs) and subapplications. The usage of adapter to group common interface elements further increase modularity in the application. Hence, variants are stored in the library as separate types. Extracting common parts into FBs allows reuse and maintainability through composition. However, adapting to changing requirements frequently results in a number of manual changes. For instance, replacing a type may involve modifying diverse connections and parameters. This error-prone manual adaptation requires advanced capabilities for handling the variability of automated production systems.

We propose an approach for managing variability in IEC 61499 using delta-oriented variability modeling [7], [8]. This approach allows engineers to systematically transform an existing IEC 61499 model representing one system variant into other variants. The transformation process is enabled by describing a list of modification operations in a separate model called a *delta model*. Engineers can select any delta model that corresponds to the modification operations needed for transforming the existing model. Changes listed in the selected delta models will transform the existing model, resulting in a representation of other system variants that replaces the traditional approach of copy-paste-modify. Hence, delta-oriented variability modeling can provide better maintainability, adaptability, and reusability of a model [8].

II. Background and Related Work

Developing automated production systems requires close collaboration between various disciplines such as mechanical, electrical, and software engineering [9]. Decisions from one discipline may affect how the other involved disciplines design or implement the system. For example, realizing a specific requirement for a system (e.g., additional quality checking, increased throughput) may require additional hardware and implementation of the corresponding control code. Hence, describing variability for production systems must span all

disciplines of production systems development [10]. Covering all variability aspects of a CPPS in a single model will not be feasible because each discipline requires a representation of distinct concerns [11].

Several existing approaches deal with the variability of production systems. This includes UML-based variability modeling on an architectural level [12], adopting a variability mechanism based on Product-Process-Resource (PPR) models [13], and relating general software variability to process and topology in a multi-variability modeling approach [14]. Only one approach has so far investigated variability in IEC 61499-based systems, i.e., using a decision model to support (dynamic) variability in function block networks [15]. This approach, however, did not focus on integrating variability from multiple disciplines.

Delta-oriented variability modeling supports expressing variability in modeling languages [7], [8]. In delta-oriented variability modeling, a target model is a result of transforming a core model based on changes described in one or more delta models. The core model must represent an existing implementation of a valid software variant. Each delta model may contain one or more changes (e.g., add, modify, remove) represented as delta operations and is targeted at a configurable aspect of the core model. A delta model also contains an application condition [8] that defines when it will be applied to the core model. This application condition is usually related to a variability model [5] (e.g., a feature model) representing commonalities and variability of (a set of) software systems. Specifying the application condition in each delta model prevents unrelated changes from being applied to the core model. For example, a delta model for adding Function Blocks that evaluate the room temperature must not be activated unless the corresponding heat sensor has been selected. This relation indicates the possibility of integrating delta models with other variability models (e.g., from other disciplines) via application conditions.

First works have applied delta modeling to automation software development. For instance, MontiArcAutomaton [16] introduced a modeling language for architecture and behavioral modeling in cyber-physical systems using the extended version of MontiArc [17] to describe variability in a delta model. Delta modeling has also been introduced to capture and express variability in multi-perspective modeling composed of UML activity, component, and state chart diagrams in manufacturing systems [18].

Our work is focused on expressing variability in IEC 61499 using delta modeling, which to the best of our knowledge, has not been done before. We aim to use delta modeling in IEC 61499 and various other models representing other CPPS disciplines' (e.g., mechanics, electrics) variability [10]. In the next section, we will present our idea of adopting delta modeling for IEC 61499.

III. Expressing Core & Delta in IEC 61499

A delta model describes the delta model name, a configured aspect in a core model, a list containing a sequence of delta operations, an application condition (using a *when*-clause), and (optionally) references to required delta models (using an *after*-clause) [19]. Given a valid configuration, both *when*-clause and *after*-clause will determine the order of applying delta models to the core model.

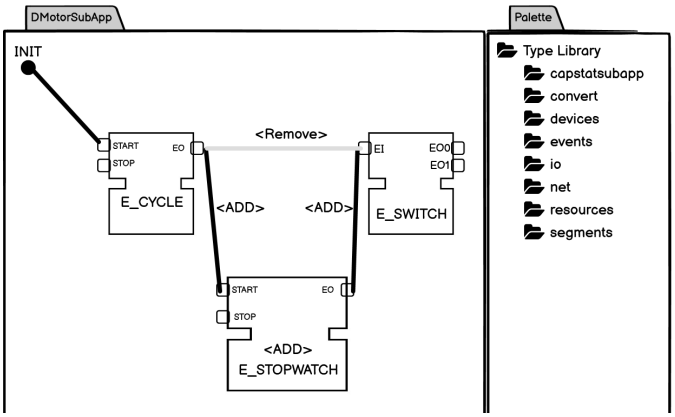
We define the configurable aspect in IEC 61499 to determine the granularity of the delta model and its delta operations. We currently use the definition of software components in IEC 61499 by Sünder [20] as a starting point to define a configurable aspect in a delta model. Sünder describes Basic Function Blocks (BFB), Service Interface Function Blocks (SIFB), Composite Function Blocks (CFB), and subapplications as software components in IEC 61499. We will refine this definition in future work when dealing with variability models from other disciplines. We use standard delta operations provided by Seidl et. al. [21] as a starting point for the IEC 61499 delta operations, which include (1) `Set/Unset` to alter a single value reference, (2) `Add/Insert/Remove` to manipulate a set of values in many-valued references, and (3) `Modify` to alter the value of an attribute. In future work, we will investigate whether further delta operations are required.

Creating a core model can be treated as developing a single production system using IEC 61499. Since the delta model represents changes to produce system variants according to a product line, the core model must also represent a valid system according to the product line. Since delta operations in the delta model can alter the core model in several ways (add, remove, or modify), the engineer can decide the core model's design accordingly. Engineers may create a big core model with all possible variations or a minimum IEC 61499 model representing one production system variant. In the former case more remove operations will be specified in deltas, in the latter case more add operations will be defined.

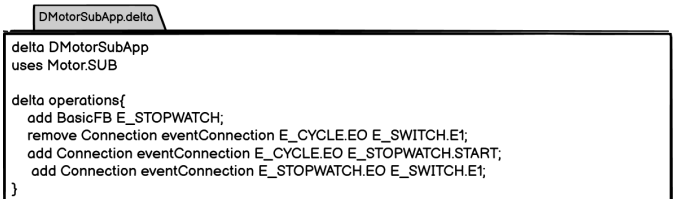
Creating a delta model differs from manually modifying the core model. Whenever engineers want to adjust a core model using a delta model, they must select which aspects (e.g., FBs, subapplications) will be modified and write it in the delta model. Engineers must only modify one aspect in one delta model. Also, the name for the delta model must be unique to avoid conflict and confusion when maintaining multiple delta models. Engineers must list every possible modification targeted to the selected aspects using delta operations. Thus, changes for the core model can now be maintained in a delta model instead of manually modifying the core model.

Fig. 1 shows an example of a delta model in IEC 61499. In this illustration, assume an engineer wants to add an `E_STOPWATCH` FB to extend a flow between `E_CYCLE` and `E_SWITCH` FBs. Engineers currently are required to instantiate the `E_STOPWATCH` FB manually from the Type Library. Engineers also have to manually restructure the event connections, so `E_CYCLE`, `E_STOPWATCH`, and `E_SWITCH` FBs are connected.

These changes would be automatically conducted by applying a predefined delta model. The delta model could be defined using a textual representation as illustrated in Fig. 1b with



(a) Graphical Editor



(b) Textual Editor

Fig. 1. The idea of representing deltas in IEC 61499.

(in this example) add and remove delta operations. The delta model could, however, also be graphically defined and represented as illustrated in Fig. 1a. An <ADD> annotation here describes the addition of the E_STOPWATCH FB and the new event connections. Removing the event connection between E_CYCLE and E_SWITCH is represented by a <Remove> annotation and fading the connection color (e.g., the event connection is drawn as a Grey line).

In the context of IEC 61499, having a graphical representation will probably make it easier for users to create delta models. A graphical representation for the delta model also complements IEC 61499 better as it can help users visualize the effect of applying a delta better. On the other hand, textual representations can be used more easily to exchange information with other tools. Thus, we argue that a delta model in IEC 61499 should be represented both, using textual and graphical representations, so that users can choose.

Delta-oriented variability modeling also assists engineers in maintaining the models when the product line is evolving due to changes in requirements. Instead of adjusting core and delta models, engineers can create new delta models to represent new requirements. Thus, engineers can now use both versions of models providing more flexibility when maintaining existing systems.

IV. Discussion & Challenges

The idea of applying delta modeling in IEC 61499 is our first step to master variability in CPPS using SPL concepts [22], but it is far from done. Several challenges must be addressed to use delta-oriented variability modeling for IEC 61499 in the CPPS context successfully. Currently, we work on solving the challenge of integrating delta modeling for IEC 61499 with other variability models, error checking mechanisms, a semi-automatic process for creating delta models, guidelines to create highly reusable deltas, and facilitating delta modeling for IEC 61499 in Eclipse 4diac[1].

Delta modeling mainly uses feature models to describe the application condition in a delta model [8]. Expressing variability for all disciplines in CPPSs using only feature models will, however, not be sufficient since we are dealing with heterogeneous models not only covering structural but also process/behavioral variability in and across multiple disciplines [11]. Thus, we need to investigate *how to describe application conditions for the delta model in IEC 61499 using heterogeneous variability models*. A potential solution could be to describe logical constraints based on heterogeneous variability models [10] and use them as application conditions for delta modeling in IEC 61499.

Since any small change may produce several delta models [9], *the number of delta models must be kept manageable and maintainable*. We argue that having a maintainable number of delta models is related to how reusable the delta model expresses changes between system variants. Having a reusable delta model will ensure that each delta model represents changes in several system variants, thus reducing the number of maintained models. One solution to this challenge is to make sure we have a highly modular design in the core model. Changes from a core model with high modularity also produce highly reusable delta models since the core model is easy to adjust. As a starting point, we can use existing works regarding design patterns to guide engineers in creating a highly modular IEC 61499 model [23], [24].

Error checking in delta-oriented variability modeling is also crucial as defining delta models manually is complex and error-prone [9]. We plan to address this task in future work by creating a visualization mechanism and formal verification [25] to provide an overview of the transformation results. A benefit of the visualization is to inform the engineers whether applying specific delta operations will introduce any errors and fix them accordingly before proceeding before the transformation process has been applied.

Simple changes such as adding a sensor may require a large set of delta models to be defined [9]. Hence, we consider developing a *semi-automatic process* for creating delta models by comparing two or more IEC 61499 models representing valid system variants. Existing works for comparing models and capturing their differences [26], [27] can be used as a starting point. Engineers can group these differences into one or more delta models accordingly.

Lastly, we would like to *provide tool support* by integrating the IEC 61499 core and delta modeling concept in Eclipse 4diac. We can use a formal delta modeling definition [8] as a foundation to build the core and delta modeling concept for IEC 61499. To implement the concept of delta-oriented variability modeling inside Eclipse 4diac, we can use an existing

[1]https://www.eclipse.org/4diac/

framework that can enable delta modeling for any modeling language such as DeltaEcore [21] or SiPL framework [26].

V. Conclusion and Research Agenda

This paper described our idea of managing variability in IEC 61499-based systems using delta-oriented variability modeling. We use existing work on software component definition [20] and standard delta operations [21] in delta modeling as a starting point of the delta model implementation for IEC 61499. We also illustrated our idea of representing the delta model using both a graphical and textual representations. To ease the process of capturing the delta model in IEC 61499, we suggest to work on a semi-automatic approach using model comparison [26], [27].

As the next step, we will refine our general understanding of delta operations in IEC 61499 by conducting experiments with concrete systems. We will also discuss with an industry partner how we can better support the definition and representation of the delta model in IEC 61499. We will create a prototype for manually creating the core and delta model of IEC 61499 in Eclipse 4diac and a configuration mechanism to automatically produce a control software system. We will then deal with capturing deltas automatically, e.g., relying on model comparison mechanisms, and work on error checking, e.g., relying on consistency checking mechanisms. We will combine our delta-oriented approach with other variability models describing variability in other disciplines in the CPPS context.

Acknowledgment

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, and the Christian Doppler Research Association is gratefully acknowledged. We explicitly want to thank our industry partner for their continuous support.

References

[1] L. Monostori, "Cyber-physical production systems: Roots, expectations and r&d challenges," *Procedia CIRP*, vol. 17, pp. 9–13, 2014.

[2] B. Vogel-Heuser, S. Feldmann, J. Folmer, J. Ladiges, A. Fay, S. Lity, M. Tichy, M. Kowal, I. Schaefer, C. Haubeck, W. Lamersdorf, T. Kehrer, S. Getir, M. Ulbrich, V. Klebanov, and B. Beckert, "Selected challenges of software evolution for automated production systems," in *13th IEEE Int'l Conf. on Industrial Informatics*. IEEE, 2015, pp. 314–321.

[3] J. Bosch, R. Capilla, and R. Hilliard, "Trends in systems and software variability [guest editors' introduction]," *IEEE Software*, vol. 32, no. 3, pp. 44–51, 2015.

[4] K. Pohl, G. Böckle, and F. J. van Der Linden, *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.

[5] M. Raatikainen, J. Tiihonen, and T. Männistö, "Software product lines and variability modeling: A tertiary study," *Journal of Systems and Software*, vol. 149, pp. 485–510, 2019.

[6] International Electrotechnical Commission (IEC), TC65/WG6, "IEC 61499-1, Function Blocks - part 1: Architecture: Edition 2.0," Geneva, 2012.

[7] D. Clarke, M. Helvensteijn, and I. Schaefer, "Abstract delta modeling," *SIGPLAN Not.*, vol. 46, no. 2, p. 13–22, Oct. 2010.

[8] I. Schaefer, "Variability modelling for model-driven development of software product lines." in *Proc. of the 4th Int'l Workshop on Variability Modelling of Software-Intensive Systems*. ICB-Research Report 37, Universität Duisburg-Essen 2010, 2010, pp. 85–92.

[9] B. Vogel-Heuser, J. Mund, M. Kowal, C. Legat, J. Folmer, S. Teufl, and I. Schaefer, "Towards interdisciplinary variability modeling for automated production systems: Opportunities and challenges when applying delta modeling: A case study," in *13th IEEE Int'l Conf. on Industrial Informatics*. IEEE, 2015, pp. 322–328.

[10] H. S. Fadhlillah, K. Feichtinger, L. Sonnleithner, and R. Rabiser, "Towards heterogeneous multi-dimensional variability modeling in cyber-physical production systems," in *25th ACM Int'l Systems and Software Product Line Conf. - Volume B*. ACM, 2021, in press.

[11] J. Krüger, S. Nielebock, S. Krieter, C. Diedrich, T. Leich, G. Saake, S. Zug, and F. Ortmeier, "Beyond software product lines: Variability modeling in cyber-physical systems," *ACM Int'l Conference Proceeding Series*, vol. 1, pp. 237–241, 2017.

[12] R. Behjati, T. Yue, L. Briand, and B. Selic, "Simpl: A product-line modeling methodology for families of integrated control systems," *Inf. Softw. Technol.*, vol. 55, no. 3, pp. 607–629, 2013.

[13] K. Meixner, R. Rabiser, and S. Biffl, "Feature identification for engineering model variants in cyber-physical production systems engineering," in *14th Int'l Working Conf. on Variability Modelling of Software-Intensive Systems*. ACM, 2020, pp. 18:1–18:5.

[14] M. Fang, G. Leyh, J. Doerr, and C. Elsner, "Multi-variability modeling and realization for software derivation in industrial automation management," in *Proc. of the ACM/IEEE 19th Int'l Conf. on Model Driven Engineering Languages and Systems*. ACM, 2016, pp. 2–12.

[15] R. Froschauer, A. Zoitl, and P. Grunbacher, "Development and adaptation of iec 61499 automation and control applications with runtime variability models," in *2009 7th IEEE Int'l Conference on Industrial Informatics*. IEEE, 2009, pp. 905–910.

[16] J. O. Ringert, B. Rumpe, and A. Wortmann, "Architecture and behavior modeling of cyber-physical systems with montiarcautomaton," *arXiv preprint arXiv:1509.04505*, 2015.

[17] A. Haber, H. Rendel, B. Rumpe, I. Schaefer, and F. van der Linden, "Hierarchical variability modeling for software architectures," in *2011 15th Int'l Software Product Line Conference*. IEEE Computer Society, 2011, pp. 150–159.

[18] M. Kowal, C. Legat, D. Lorefice, C. Prehofer, I. Schaefer, and B. Vogel-Heuser, "Delta modeling for variant-rich and evolving manufacturing systems," in *1st Int'l Workshop on Modern Software Engineering Methods for Industrial Automation*. ACM, 2014, pp. 32–41.

[19] C. Pietsch, C. Seidl, M. Nieke, and T. Kehrer, "Delta-oriented development of model-based software product lines with DeltaEcore and SiPL: A comparison," in *Model Management and Analytics for Large Scale Systems*. Academic Press, 2020, pp. 167–201.

[20] C. Sünder, "Evaluation of downtimeless system evolution in automation and control systems; how to decide whether a system under operation can be changed without disturbances?" Ph.D. dissertation, TU Wien, 2008.

[21] C. Seidl, I. Schaefer, and U. Aßmann, "Deltaecore - A model-based delta language generation framework," in *Modellierung 2014*, ser. LNI, vol. P-225. GI, 2014, pp. 81–96.

[22] R. Rabiser and A. Zoitl, "Towards Mastering Variability in Software-Intensive Cyber-Physical Production Systems," *Procedia Computer Science*, vol. 180, pp. 50–59, 2021.

[23] S. Patil, D. Drozdov, and V. Vyatkin, "Adapting software design patterns to develop reusable IEC 61499 function block applications," in *16th IEEE Int'l Conf. on Industrial Informatics*. IEEE, 2018, pp. 725–732.

[24] L. Sonnleithner, B. Wiesmayr, V. Ashiwal, and A. Zoitl, "Iec 61499 distributed design patterns," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, in press.

[25] V. Dubinin, V. Vyatkin, and H. Hanisch, "Modelling and verification of IEC 61499 applications using prolog," in *Proc. of 11th IEEE Int'l Conf. on Emerging Technologies and Factory Automation*. IEEE, 2006, pp. 774–781.

[26] C. Pietsch, T. Kehrer, U. Kelter, D. Reuling, and M. Ohrndorf, "Sipl - A delta-based modeling framework for software product line engineering," in *30th IEEE/ACM Int'l Conf. on Automated Software Engineering*. IEEE Computer Society, 2015, pp. 852–857.

[27] M. Zadahmad, E. Syriani, O. Alam, E. Guerra, and J. de Lara, "Domain-specific model differencing in visual concrete syntax," in *Proc. of the 12th ACM SIGPLAN Int'l Conf. on Software Language Engineering*. ACM, 2019, pp. 100–112.