Towards facilitating software engineering for production systems in Industry 4.0 with behavior models

Bianca Wiesmayr
bianca.wiesmayr@jku.at
LIT CPS Lab, Johannes Kepler University Linz
Linz, Austria

ABSTRACT

With the growing adoption of Industry 4.0 concepts in production systems, new challenges arise in engineering control software. Highly distributed control with tight real-time constraints and safety regulations results in increasingly complex software. Current research focuses on increasing the abstraction with new architectures and modularization of software. The presented PhD research addresses modeling of the interactions between control software components, and of the emergent behavior of these compositions. Such behavior models can support the initial implementation, and facilitate (semi-)automated testing and monitoring of control software. Finally, visualizing behavior in a model can enhance understandability of existing control software, when software developers need not access abstracted hierarchy levels to deduct their functionality. This work aims at optimizing the benefit of behavior models in developing control software: Modeling the expected behavior directly for new software will allow using them throughout the software life-cycle. For legacy software, the initial development effort of behavior models will be minimized by automatically capturing behavior models from the implementation. The approach is evaluated in case studies and user studies to integrate experiences from the industrial domain into this software engineering research.

CCS CONCEPTS

• **Software and its engineering** → **Domain specific languages**; • **Computer systems organization** → **Embedded and cyber-physical systems**.

KEYWORDS

Model-driven software engineering, Control software, IEC 61499

ACM Reference Format:
Bianca Wiesmayr. 2022. Towards facilitating software engineering for production systems in Industry 4.0 with behavior models. In *44th International Conference on Software Engineering Companion (ICSE '22 Companion), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3510454.3517070

ICSE '22 Companion, May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9223-5/22/05…$15.00
https://doi.org/10.1145/3510454.3517070

1 INTRODUCTION

Mass customization emerged in modern industrial automation as part of Industry 4.0. It requires producers to flexibly adjust their goods to varying market demands, thus demanding highly adaptable manufacturing equipment, both in hardware and software [51]. Such Cyber-Physical Production Systems (CPPSs) are nowadays highly automated and interconnected with plant-wide systems that orchestrate the production, such as order tracking or quality monitoring. Based on the input from these systems and on data from sensors, control software manipulates actuators, such as motors. Due to high requirements on safety and throughput of a production system, its software is executed with tight real-time constraints on embedded devices, typically Programmable Logic Controllers (PLCs). Software complexity is introduced by the large number of interacting subsystems, which are often autonomous and operate in parallel [51]. Subsystems may be updated independently, thus additionally posing challenges for software evolution [35]. With Industry 4.0, the software share in industrial automation is increasing [37, 50] and at the same time the software complexity is increasing [13, 42]. Software engineering techniques, which can help tackling this complexity, are not a core competence of automation engineers and therefore not state of the art in the domain [40, 42].

Industrial standards (i.e., ISO/IEC) are significant for production plants [13], as the life-cycle of a production system can span up to several decades [38]. Industrial standards are not a core concern of current software engineering research [49]. The currently established graphical and textual DSLs (domain-specific languages) for implementing control software are defined in IEC 61131–3 [1]. A newer standard, IEC 61499 [30], addresses many modern challenges in developing control applications for distributed systems. It standardizes an executable domain-specific modeling language (xDSML) for control applications that is targeted at automation engineers, not software engineers [52]. This language abstracts the control logic from the hardware and low-level communication, thus fostering a component-based design [53]. IEC 61499 supports a model-driven architecture [54], but does not cover the whole software development process. For instance, requirements engineering is not possible within the means of the standard [36]. Furthermore, hierarchical structures help reducing complexity [53], but this abstraction reduces the visibility of the encapsulated functionality [5]. IEC 61499 mostly supports modeling of hierarchical structures, while its behavior models are limited to the intra-object behavior of software components. The hypothesis of this PhD research is that *using executable behavior models within IEC 61499 allows to support control software engineers in modeling or automatically visualizing interactions between software components, or finding inconsistencies*

between the expected and implemented behavior. Partially automating these processes allows to increase the acceptance of model-driven approaches among domain experts and to optimize the benefits of the proposed software engineering approach.

Leveraging the acceptance of modeling languages in industrial automation requires addressing developer needs and domain requirements. Several research challenges are specific to the automation domain and have to be addressed [37]: Developed approaches need to be usable by the domain experts (automation engineers). Automation engineers in industry have only limited knowledge of computer science, hence possible modeling methods are constrained. This constraint was also taken into account in the current specification of IEC 61499 [54]. Furthermore, languages need to support software evolution and versioning over decades. Software changes are frequently implemented directly on the embedded device at the production site, and such updates often take effect live during runtime of the software. Finally, control software closely interacts with the mechanical and electrical parts of the system (e.g., controlling a robotic arm). Control processes have tight hard real-time requirements, which particularly affect distributed systems due to the additional communication between subsystems. As control software is only one part of a CPPS [33], its relation to the full production system, including mechanical components, has to be considered. During commissioning of the plant, integration tests require software updates directly on the embedded system (PLC). Such changes are applied under time pressure, they may therefore not be propagated back to the original model [37]. Inconsistencies can be introduced when evolving the implementation without updating the model, or by erroneous semantic differences between the model and the implementation [8].

2 RELATED WORK

General-purpose modeling languages have been applied for control software. The most prominent example is the Unified Modeling Language (UML) [20]. The large variety of structural and behavioral diagrams in UML allow its use in various disciplines, as the language scope can be extended or restricted via profiles to adapt to domain-specific needs. Various profiles for the domain of control software have been proposed: A widely applied example is the UML profile for Modeling and Analysis of Real-Time and Embedded Systems (UML MARTE), which extends UML with real-time capabilities, and has been standardized [22]. UML for Real-Time Embedded Systems (UML-RT) focuses on systems with soft real-time constraints [4], not factory automation. Less extensive profiles were developed specifically for industrial control systems, e.g., the RT-ICS profile [15]. For most specific profiles, however, the lack of tool support restricts their practical applicability.

Modeling of full automation systems addresses the close interaction between control software and the physical components. The profile UML4IoT allows effectively integrating cyber-physical components into manufacturing systems [34]. Also the Systems Modeling Language (SysML, [21]) has been applied in the automation domain with dedicated extension profiles [32, 41]. SysML or the domain-specific GRAFCET [18] capture the expected behavior and structure of a full automation system, not only its software. In general, these languages are not directly executable and have therefore been applied in conjunction with executable languages, such as those defined for control software in IEC 61131-3 [39, 41] or IEC 61499 [10, 24, 33]. Such an approach acknowledges the heterogeneity of involved engineering disciplines and their different modeling needs. Control loops may require differential equations, while reactions of a system can be modeled as a Discrete Event System. Multi-paradigm modeling therefore combines various approaches to best support engineering of each aspect of a CPPS (e.g., MPM4CPS [2], or model-integrated mechatronics [33]).

For *control software engineering*, languages have to address the respective challenges [37]. Limitations of the current standard IEC 61499 can be summarized from the literature. The standard IEC 61499 only defines the language and the execution semantics, but neither a concrete implementation of a runtime [52] nor a suggested development process [9]. We can differentiate two main reasons for using general-purpose modeling languages together with IEC 61499: authors either aim at an integrated development of both physical and cyber components, or address disadvantages of IEC 61499 by:

- Introducing a standardized development process [14]
- Supporting developers in capturing requirements [36]
- Modeling the interaction between components in behavior diagrams before the actual implementation [6]
- Planning and designing an architecture for the hierarchical IEC 61499 models [11]
- Using established modeling tools from other languages to generate an IEC 61499 model [10]

Most model transformations were performed manually or code was generated directly from a UML model. However, early works from Thramboulidis [31] involves automated model transformations between UML diagrams and IEC 61499 software. These works provide tool support and cover additional stages of the software engineering process (such as requirements engineering). This unidirectional approach allows transforming models between different stages of the software engineering process, but does not support automatically deriving behavior models from an existing implementation. This increases the modelling effort and requires manually propagating changes from the CPPS back to the model, leading to potential inconsistencies that are difficult to detect.

3 RESEARCH QUESTIONS

The following research questions (RQs) aim at facilitating software engineering for production systems by automating parts of the model-driven process. A key goal is a bidirectional mapping between behavior models and the control software. As stated in the research hypothesis, this PhD research focuses on modeling interactions between software components. IEC 61499 currently standardizes two behavior models, the state-based ECC (Execution Control Chart), and scenario models at the interface of software components (Service Sequences). Both models are limited to individual software components.

RQ1: How can we increase the utility of behavior models defined in IEC 61499 w.r.t. the application areas consistency checking, testing, and documentation/visualization?

Manually modeled Service Sequences have been used for verification of connections [26] and specifying unit tests [12]. It will be necessary to describe the relations between Application model,

ECC, and Service Sequences, and to allow transformations between the models where feasible. Not all relevant information can be captured in current behavior models [26], leading to the RQ:

RQ2: What are limitations of the currently standardized behavior models w.r.t. implementation, consistency checking, testing and documentation/visualization?

Behavior models will be therefore used for the implementation of case studies to reveal the relations between the behavior specification and the IEC 61499 constructs. By focusing on software engineering processes such as testing, relevant limitations can be identified. These limitations form the input for *extending IEC 61499 with behavior modeling approaches*:

RQ3: Which domain-specific requirements does a behavior model have to fulfill for software models according to IEC 61499, i.e., applications and software components (Function Blocks)?

Experiences with industrial users and own case studies can help analyzing the required domain information (e.g., events, data, real-time constraints, non-functional requirements) for using a behavior model in our defined application areas. Potential extensions of the standard have to comply with the requirements from RQ3:

RQ4: Which human-comprehensible behavior models fulfill these requirements and can be integrated in IEC 61499?

The identified behavior models should be optimized for usability by domain experts. To increase the acceptance of model-driven design, different abstraction levels need to be closely integrated with each other. Furthermore, models should be transferred seamlessly between different phases in the design process [39] to reduce error proneness and improve reuse. Finally, we *use the newly identified behavior models* in the defined application areas:

RQ5: How can we integrate the behavior models into the design flow to provide a benefit for developers regarding the defined application areas?

Answering this research question requires a clear guideline on applying the behavior models, and efficient tool support for the engineering process. If a single model cannot fulfill all posed requirements, we have to outline which model is viable in a specific situation. Inconsistencies may furthermore occur if certain details are represented differently in alternative views of the same model. Such inconsistencies should be revealed automatically. To facilitate adoption of the new behavior specifications, required extensions of the standard are proposed. This includes a standard-compliant XML exchange format if the behavior models have to be stored separately and cannot be derived automatically from the IEC 61499 model. Such extensions have to be backwards compatible with preceding versions of the standard. A comprehensive assessment allows engineers to evaluate whether the proposed approach is applicable for their use case, leading to:

RQ6: What are the limitations of the new approach?

Limitations of the provided approach have to be investigated regarding functional requirements (e.g., supported languages and domains) and non-functional requirements (e.g., performance, effort, and usability of the approach).

In this PhD research, a Design Science approach [43] is followed. Experiences from users (automation engineers from production system manufacturers) are driving the investigation. The state of the art and practice is analyzed based on literature reviews and an initial user study that includes interviews with industrial experts. Case studies and cooperations with domain experts grant a better understanding of the problem domain. Prototyping technologies allows to demonstrate a proof-of-concept. All results will be available open source as part of the 4diac IDE from the Eclipse 4diac™ project [7]. New implementations are repeatedly evaluated in practice via case studies and a final user study, leading to continuous improvements of the designed methodology.

4 CONTRIBUTIONS

The main expected contributions are:

- **C1:** Enhancing software engineering for distributed control systems with visual behavior models, including a mechanism to capture behavior models from legacy software automatically [45, 46]
- **C2:** A model execution framework for the DSML IEC 61499 including a model interpreter [46]
- **C3:** Tool support for control software engineers to implement and test models, as well as to improve model comprehension [48]

This work does not replace, but supplements existing languages and tools that are used in the domain. Behavior models for applications (C1) will provide an important infrastructure for IEC 61499, e.g., as test specification and for software updates [27]. The work for C2 includes defining execution semantics for IEC 61499, which can serve as a reference implementation for tool developers, but also allows using infrastructure from the community of DSML engineering (e.g., [16]). Similar approaches were followed for UML with Foundational UML (fUML) [23] and the Action Language for fUML [19]. The latter is a high-level language for specifying executable behavior. Open source tool support (C3) will provide the required infrastructure for applying the designed concepts in practice. User studies furthermore add experiences made in the domain of industrial automation to the body of knowledge of developing visual modeling tools.

5 EVALUATION

Regarding the technical implementation, case studies are the main mechanism for evaluation. Case Study 1 will be the capping station [53], which is commonly used as an automation example in the literature. As Case Study 2, with a realistic scope, the VDMA demonstrator will be used (e.g., [3]), which demonstrates key features from Industry 4.0 in a distributed control system. Scenarios of each application are manually modeled as behavior models (e.g., Activity Diagrams) and used as a basis for the IEC 61499 implementation. In a proof-of-concept, the standardized behavior models for software components will be first tightly integrated with the implementation (**RQ1**, i.e., Service Sequences and the state-based Execution Control Chart). Considering our application areas, this requires interpreting the models directly, visualizing traces of the software as Service Sequences, and using these specifications as test cases. The case studies will also demonstrate examples for limitations of the graphical models (**RQ2**). Requirements for behavior models (**RQ3**) are furthermore explained based on both case studies.

Table 1: Published papers (upper part) and papers that are in progress or planned (lower part).

Paper	Title/Topic	Venue	Case Study	RQ
[46]	Model-Execution Framework for IEC 61499 components	IEEE ETFA'21	-	1
[44]	Using behavior models to guide the application structure	IEEE ICPS'20	-	1, 2
[28]	Implementing interactions based on Activity Diagrams	IEEE ETFA'21	1	1, 2
[18, 47]	Translation patterns for implementing (hierarchical) Grafcets in IEC 61499	IEEE ETFA'20	-	1, 2
[48]	User Study on Maintaining Large-Scale Automation Software	MODELS'21	1	2
[45]	Requirements for dynamic interface models of IEC 61499	IEEE ETFA'20	-	1, 3
	Comparing behavior models for IEC 61499 Applications	IEEE INDIN'22	2	3, 4
	Model-Execution Framework for IEC 61499 applications	TII	1	4, 5
	User Study: Control software engineering with behavior models	SoSym	3	5, 6

RQ4 is evaluated by extending the implemented approach with the chosen new models. To successfully answer this question, the suggested approach has to facilitate the software engineering process. For instance, the engineering effort should be lower compared to manual testing, the round-trip engineering time should be reduced, and inconsistencies should be revealed automatically. For this evaluation, especially the counter-examples from RQ2 are relevant, as these previously could not be modeled in IEC 61499. Regarding the practical applicability of the approach, especially on RQ4 and **RQ5**, a user study with industrial experts is required. These experts are recruited from industry partners of our research lab. The user study aims at evaluating whether domain experts can understand and apply the new model(s) based on a real-world control application from their own field (Case Study 3). Furthermore, severe usability issues in involved tools could prevent developers from using the model(s). In a multi-stage approach, involved processes and tools will be evaluated based on the Cognitive Dimensions approach [5], before conducting an experiment with domain experts. Concerning **RQ6**, particularly the maximum supported complexity of the control software has to be identified based on our case studies. This complexity can be measured in lines of code (i.e., lines in the standardized XML), but also in software metrics, which provide a more comprehensive view on the software [29]. The quality of mined specification models can be also evaluated based on metrics [17, 25]. For the case studies, the execution time for recording traces will be measured to estimate the applicability of the approach for large-scale software.

6 INITIAL RESULTS AND TIMELINE

Regarding *understanding models*, a multi-stage usefulness study on maintaining large-scale control software was conducted based on case study 1, providing insights on how engineers interact with the programming tool and the software models. The publication at this year's MODELS conference [48] includes results from a user study with industrial experts, and has already resulted in numerous tool improvements for the 4diac IDE. The user study revealed that the industrial experts have difficulties working with hierarchical structures. The abstraction of hierarchical composition is furthermore incomplete, as the content of subapplications had to be accessed to understand the purpose of software parts.

Investigating the *implementation* of application behavior models resulted in a structured translation approach from the specification language GRAFCET (used by system engineers) to the executable language IEC 61499 (used by software engineers) [18, 47]. The identified translation patterns retain hierarchical structures and cover operation mode switching, which is also used for error handling (Best Paper Award for Factory Automation at the IEEE ETFA 2020 for [18]). The results can be generalized to other state-based models. For instance, activity diagrams were used as a means to design fully distributed architectures of IEC 61499 software [28].

Finally, automatically generating behavior models from the control software requires a semantics-aware tool infrastructure. Therefore, a model execution framework for software components of IEC 61499 (Basic Function Blocks with a simplified state-based diagram and textual algorithms) was created (cf. [46]). Using this framework allows automatically generating behavior models that capture the relation between inputs and outputs that is observable at the interface (i.e., Service Sequences). Furthermore, these diagrams can be used as a test specification. This approach is currently restricted by the limited expressiveness of Service Sequences. Interpreting control software has several advantages over executing it in runtimes: The interpreter allows executing partial models, and it improves the platform-independence because models are executed directly. Furthermore, immediate feedback is feasible, possibly reducing the round-trip engineering time. When component tests are created using the designed fluent interface developed in Java, the tests can also be executed remotely as unit-tests. Domain experts benefit from tool integration for this execution framework. They can then specify expected behavior graphically and run the interpreter, even without access to a runtime. Applying structured implementation guidelines (e.g., [47]) furthermore requires less knowledge about software architectures.

In 2020 and 2021, 8 articles were presented at 4 international conferences (cf. paper plan in Table 1). The goal is to complete this PhD in 2023, after submitting core contributions to established journals from the modeling community (e.g., SoSym) and the automation engineering community (e.g., TII). The next step is extending the model execution framework of Basic Function Blocks to cover full IEC 61499 applications to (i) capture application behavior models that illustrate the interactions between the software components in the applications (scenarios), and (ii) test whether existing scenarios are consistent to the current implementation.

REFERENCES

[1] TC 65/SC 65B. 2013. *IEC 61131 - Programmable controllers, Part 3: Programming languages* (3 ed.). International Electrotechnical Commission (IEC), Geneva.
[2] Moussa Amrani, Dominique Blouin, Robert Heinrich, Arend Rensink, Hans Vangheluwe, and Andreas Wortmann. 2021. Multi-paradigm modelling for cyber–physical systems: a descriptive framework. *Software & Systems Modeling* 20, 3 (2021), 611–639.
[3] Virendra Ashiwal and Alois Zoitl. 2021. Messaging Interaction Patterns for a Service Bus Concept of PLC-Software. In *26th International Conference on*

Emerging Technologies and Factory Automation (ETFA). IEEE, Västeras, Sweden.
[4] Mojtaba Bagherzadeh, Karim Jahed, Benoit Combemale, and Juergen Dingel. 2019. Live-UMLRT: A Tool for Live Modeling of UML-RT Models. In *ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, Munich, Germany, 743–747.
[5] Alan Blackwell and Thomas Green. 2003. Notational Systems — The Cognitive Dimensions of Notations Framework. In *HCI Models, Theories, and Frameworks*, John M. Carroll (Ed.). Morgan Kaufmann, San Francisco, 103–133.
[6] Christos Tranoris and Kleanthis Thramboulidis. 2003. Integrating UML and the function block concept for the development of distributed control applications. In *International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Lisbon, Portugal, 87–94.
[7] Eclipse 4diac. 2021. 4diac IDE 2.0. https://www.eclipse.org/4diac
[8] Stefan Feldmann, Konstantin Kernschmidt, Manuel Wimmer, and Birgit Vogel-Heuser. 2019. Managing inter-model inconsistencies in model-based systems engineering: Application in automated production systems engineering. *Journal of Systems and Software* 153 (2019), 105–134.
[9] Georg Frey and Tanvir Hussain. 2006. Modeling Techniques for Distributed Control Systems Based on the IEC 61499 Standard - Current Approaches and Open Problems. In *8th International Workshop on Discrete Event Systems, 2006*. IEEE, Ann Arbor, MI, USA, 176–181.
[10] Carlos A. Garcia, Exteban Castellanos, and Marcelo V. García. 2018. UML-Based Cyber-Physical Production Systems on Low-Cost Devices under IEC-61499. *Machines* 6, 2 (2018), 22.
[11] Carlos A. Garcia, Esteban X. Castellanos, Cesar Rosero, Carlos Sanchez, and Marcelo V. García. 2017. Designing Automation Distributed Systems Based on IEC-61499 and UML. In *5th International Conference in Software Engineering Research and Innovation (CONISOFT)*. IEEE, Merida, Mexico, 61–68.
[12] Reinhard Hametner, Benjamin Kormann, Birgit Vogel-Heuser, Dietmar Winkler, and Alois Zoitl. 2011. Test case generation approach for industrial automation systems. In *5th International Conference on Automation, Robotics and Applications*. IEEE, Wellington, New Zealand, 57–62.
[13] Robert Harrison, Daniel Vera, and Bilal Ahmad. 2016. Engineering Methods and Tools for Cyber–Physical Automation Systems. *Proceedings of the IEEE* 104, 5 (2016), 973–985.
[14] Tanvir Hussain and Georg Frey. 2006. UML-based Development Process for IEC 61499 with Automatic Test-case Generation. In *International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Prague, Czech Republic, 1277–1284.
[15] Kamran Latif, Aamer Nadeem, and Gang Lee. 2011. A UML Profile for Real Time Industrial Control Systems. In *Software Engineering, Business Continuity, and Education*. Vol. 257. Springer, Berlin, Heidelberg, 97–107.
[16] Dorian Leroy, Erwan Bousse, Manuel Wimmer, Tanja Mayerhofer, Benoit Combemale, and Wieland Schwinger. 2020. Behavioral interfaces for executable DSLs. *Software & Systems Modeling* 19, 4 (2020), 1015–1043.
[17] David Lo and Siau-cheng Khoo. 2006. QUARK: Empirical Assessment of Automaton-based Specification Miners. In *2006 13th Working Conference on Reverse Engineering*. 51–60.
[18] Oscar Miguel-Escrig, Julio-Ariel Romero-Perez, Bianca Wiesmayr, and Alois Zoitl. 2020. Distributed implementation of Grafcets through IEC 61499. In *25th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Vienna, Austria, 402–409.
[19] Object Management Group. 2017. Action Language for Foundational UML: Version 1.1. https://www.omg.org/spec/ALF/1.1/About-ALF/
[20] Object Management Group. 2017. OMG Unified modelling language (OMG UML): Version 2.5. https://www.omg.org/spec/UML/About-UML/
[21] Object Management Group. 2019. OMG Systems Modeling Language (OMG SysML): Version 1.6. https://www.omg.org/spec/SysML/1.6/PDF
[22] Object Management Group. 2019. OMG UML Profile for MARTE. https://www.omg.org/spec/MARTE
[23] Object Management Group. 2021. Semantics of a Foundational Subset for Executable UML Models (FUML): Version 1.5. https://www.omg.org/spec/FUML/
[24] Seno Panjaitan and Georg Frey. 2006. Combination of UML Modeling and the IEC 61499 Function Block Concept for the Development of Distributed Automation Systems. In *11th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Prague, Czech Republic, 766–773.
[25] Michael Pradel, Philipp Bichsel, and Thomas R. Gross. 2010. A framework for the evaluation of specification miners based on finite state machines. In *2010 IEEE International Conference on Software Maintenance*. 1–10.
[26] Herbert Prähofer and Alois Zoitl. 2013. Verification of hierarchical IEC 61499 component systems with behavioral event contracts. In *11th IEEE International Conference on Industrial Informatics (INDIN)*. IEEE, Bochum, Germany, 578–585.
[27] Laurin Prenzel and Sebastian Steinhorst. 2021. Automated Dependency Resolution for Dynamic Reconfiguration of IEC 61499. In *26th International Conference on Emerging Technologies and Factory Automation*. IEEE, Västeras, Sweden.
[28] Lisa Sonnleithner, Bianca Wiesmayr, Virendra Ashiwal, and Alois Zoitl. 2021. IEC 61499 distributed design patterns. In *26th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Västeras, Sweden.
[29] Lisa Sonnleithner and Alois Zoitl. 2020. A Software Measure for IEC 61499 Basic Function Blocks. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Piscataway, NJ, 997–1000.
[30] TC65/WG6. 2012. *IEC 61499-1, Function Blocks - part 1: Architecture* (2 ed.). International Electrotechnical Commission (IEC), Geneva. www.iec.ch
[31] Kleanthis Thramboulidis. 2005. Model-integrated mechatronics - Toward a new paradigm in the development of manufacturing systems. *IEEE Transactions on Industrial Informatics* 1, 1 (2005), 54–61.
[32] Kleanthis Thramboulidis. 2010. The 3+1 SysML View-Model in Model Integrated Mechatronics. *Journal of Software Engineering and Applications* 3, 2 (2010), 109–118.
[33] Kleanthis Thramboulidis and Andrea Buda. 2010. 3+1 SysML view model for IEC61499 Function Block control systems. In *8th International Conference on Industrial Informatics*. IEEE, Osaka, Japan, 175–180.
[34] Kleanthis Thramboulidis and Foivos Christoulakis. 2016. UML4IoT-A UML-based approach to exploit IoT in cyber-physical manufacturing systems. *Computers in Industry* 82 (2016), 259–272.
[35] Martin Törngren and Paul Grogan. 2018. How to Deal with the Complexity of Future Cyber-Physical Systems? *Designs* 2, 4 (2018), 40.
[36] Chris Tranoris and Kleanthis Thramboulidis. 2002. From requirements to function block diagrams: a new approach for the design of industrial control applications. In *10th Mediterranean Conference on control and automation (MED)*. IEEE, Lisbon, Portugal, 9–12.
[37] Birgit Vogel-Heuser, Christian Diedrich, Alexander Fay, Sabine Jeschke, Stefan Kowalewski, Martin Wollschlaeger, and Peter Göhner. 2014. Challenges for Software Engineering in Automation. *Journal of Software Engineering and Applications* 07, 05 (2014), 440–451.
[38] Birgit Vogel-Heuser, Alexander Fay, Ina Schaefer, and Matthias Tichy. 2015. Evolution of software in automated production systems: Challenges and research directions. *Journal of Systems and Software* 110 (2015), 54–84.
[39] Birgit Vogel-Heuser, David Friedrich, Uwe Katzke, and Daniel Witsch. 2005. Usability and benefits of UML for plant automation - some research results. *atp international* 3, 1 (2005).
[40] Birgit Vogel-Heuser and Alexis Sarda-Espinosa. 2017. Current status of software development in industrial practice: Key results of a large-scale questionnaire. In *15th International Conference on Industrial Informatics (INDIN)*. IEEE, Emden, Germany, 595–600.
[41] Birgit Vogel-Heuser, Daniel Schütz, Timo Frank, and Christoph Legat. 2014. Model-driven engineering of Manufacturing Automation Software Projects – A SysML-based approach. *Mechatronics* 24, 7 (2014), 883–897.
[42] Valeriy Vyatkin. 2013. Software Engineering in Industrial Automation: State-of-the-Art Review. *IEEE Transactions on Industrial Informatics* 9, 3 (2013), 1234–1249.
[43] Roel Wieringa. 2014. *Design science methodology for information systems and software engineering*. Springer, Berlin and New York and Dordrecht.
[44] Bianca Wiesmayr, Lisa Sonnleithner, and Alois Zoitl. 2020. Structuring Distributed Control Applications for Adaptability. In *3rd International Conference on Industrial Cyberphysical Systems (ICPS)*. IEEE, Tampere, Finland, 236–241.
[45] Bianca Wiesmayr and Alois Zoitl. 2020. Requirements for a dynamic interface model of IEC 61499 Function Blocks. In *25th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Vienna, Austria, 1069–1072.
[46] Bianca Wiesmayr, Alois Zoitl, Antonio Garmendia, and Manuel Wimmer. 2021. A Model-based Execution Framework for Interpreting Control Software. In *26th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Västeras, Sweden. https://epub.jku.at/obvulioa/content/pageview/6408776
[47] Bianca Wiesmayr, Alois Zoitl, Oscar Miguel-Escrig, and Julio Romero-Perez. 2021. Distirbuted implementation of hierarchical Grafcets through IEC 61499. In *26th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Västeras, Sweden.
[48] Bianca Wiesmayr, Alois Zoitl, and Rick Rabiser. 2021. Assessing the Usefulness of a Visual Programming IDE for Large-Scale Automation Software. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. ACM, Fukuoka, Japan (virtual).
[49] Andreas Wortmann, Olivier Barais, Benoit Combemale, and Manuel Wimmer. 2020. Modeling languages in Industry 4.0: an extended systematic mapping study. *Software & Systems Modeling* 19, 1 (2020), 67–94.
[50] Hang Yin and Hans Hansson. 2018. Fighting CPS Complexity by Component-Based Software Development of Multi-Mode Systems. *Designs* 2, 4 (2018), 39.
[51] Ray Y. Zhong, Xun Xu, Eberhard Klotz, and Stephen T. Newman. 2017. Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering* 3, 5 (2017), 616–630.
[52] Alois Zoitl and Robert W. Lewis. 2014. *Modelling control systems using IEC 61499* (2. ed. ed.). IET Control engineering series, Vol. 95. IET, London.
[53] Alois Zoitl and Herbert Prähofer. 2013. Guidelines and Patterns for Building Hierarchical Automation Solutions in the IEC 61499 Modeling Language. *IEEE Transactions on Industrial Informatics* 9, 4 (2013), 2387–2396.
[54] Alois Zoitl and Valeriy Vyatkin. 2009. Different perspectives [Face to face; "IEC 61499 architecture for distributed automation: The '"glass half full" view. *IEEE Industrial Electronics Magazine* 3, 4 (2009), 7–23.