Model-based Engineering for Designing Cyber-Physical Systems Control Architecture and Improving Adaptability from Requirements

Alexandre Parant, François Gellot, Philippot Alexandre, Véronique Carré-Ménétrier

Model-based Engineering for Designing Cyber-Physical Systems Control Architecture and Improving Adaptability from Requirements

A.PARANT[1[0000-0001-8433-8671]], F.GELLOT[1 [0000-0001-8797-1704]], A.PHILIPPOT[1[0000-0001-5229-9452]], V. CARRE-MENETRIER[1 [0000-0002-9576-9108]]

[1] CReSTIC, University of Reims Champagne-Ardenne, Reims, France
{alexandre.parant, francois.gellot, alexandre.philippot, veronique.carre}@univ-reims.fr

Abstract. Model-Based Engineering (MBE) is a method for reducing complexity in system design. This paper presents a methodology for designing Cyber-Physical System (CPS) and its control system using System Modeling Language (SysML) diagrams and IEC 61499 standard. The real-time control system is designed from high-level knowledge and tested software components on the plug and produce principle. This paper presents an application case to demonstrate the feasibility of our approach. In addition, a set of solutions is presented to reduce the time and improve the engineering and traceability of configuration changes during the system lifecycle.

Keywords: Cyber-Physical System, Model-Based Engineering, System Modeling Language, IEC 61499, Reconfiguration

1 Introduction

Production systems have become cyber-physical, i.e., integrating electronics and software, sensors and actuators known as intelligent with communication capability. The capabilities of the physical plant have evolved considerably with the advent of computing, communication, and control integrated into system components [1]. Therefore, there are expectations around integrating Cyber-Physical Systems (CPS) in industries to make systems more adaptable and robust than traditional production systems [2]. Adaptability, connectivity, and distribution of the control system are features that allow the system to adapt to the constant evolution of products and consumer needs [2].

Agent-based architectures, in particular, holonic architectures have the necessary abilities to integrate the specificities between the physical and virtual world of CPS [3]. Holonic control architecture is composed of autonomous, intelligent, and cooperative holons to meet Industry 4.0 requirements [4]. A holon is a particular agent integrating a physical part and its control system, divided into two parts: High-Level Control (HLC) and Low-Level Control (LLC). HLC is responsible for decision-making and cooperation with the other holon. LLC is responsible for the real-time control of the physical part. Wang and Haghighi combine agent, holon, and function block (FB) technologies to model an SCP [5]. HLC is composed of agents, while LLC is composed of

IEC 61499 FB. HLC must know itself and the system to collaborate with the other agents to plan, schedule, and adapt to disturbances affecting the environment.

One of the difficulties in designing holonic control architecture is providing and modeling system knowledge to the agent part. Another difficulty lies in using knowledge to design the real-time control in the first place and to reconfigure it when needed. This article proposes a Model-Based Engineering (MBE) of a CPS control architecture. The designer models the knowledge of the system through a set of System Modeling Language (SysML) diagrams. Future works will focus on the formalization of diagrams for agent decision-making. Physical structure, dynamic behavior, and production specifications are integrated into the knowledge base to integrate physical and software design into common models. In traditional design methods, the control system design does not begin until the physical system's design ending, leading to suboptimal solutions [6]. In this paper, physical and software design are integrated at the same time.

The main contribution is the design and implementation of the low-level real-time control system from the high-level knowledge of the system. Diagram Linkage Table (DLT) is the central element of the methodology; it allows to design control system from different points of view to cope with the new requirements of CPS. The advantage is that the control system is no longer limited to automation but integrates the relationships between physical and software part of CPS components. It also considers the specifications of the production. This body of knowledge helps facilitate configuration changes. Integrating different points of view reduces the time it takes to reconfigure. Indeed, a reconfiguration requires an identification phase of the elements affected by the configuration change. The link between domains reflects disturbances from one domain to another to effectively measure their influences.

Section 2 presents the advantages of IEC 61499 for the design of CPS control system; Section 3 presents the design methodology based on SysML models, Section 4 shows the applicability on a simulated manufacturing system, and Section 5 shows the improvement of the reconfigurability thanks to the use of the knowledge base.

2 IEC 61499 Reconfiguration Abilities

IEC 61499 standard facilitates the design and development of distributed control systems [7]. IEC 61499 language is a network of FB incorporating code. The advantages lie in distributing FB between several controllers and in the principle of encapsulation which secures reuse of blocks already used in a previous program. IEC 61499 has interesting features for the reconfigurable part of the control thanks to its process way of thinking and structure that allows modular construction and easy integration or conversion of a component. Black and Vyatkin use reusable intelligent components to implement a component-based architecture in baggage handling systems [8]. They provide a decentralized control reconfigurable and fault-tolerant thanks to the specific architecture of IEC 61499.

One of the essential characteristics of IEC 61499 is the ability to reconfigure the system dynamically, i.e., during its execution. Thanks to a compliance profile, it is possible to send a set of controls to the device during its execution [1] [9]. The authors in [10] define specifications for a control architecture to support dynamic reconfiguration. Furthermore, they introduce reconfiguration services, and they describe impacts on the control application.

An agent-based approach to automatic control reconfiguration is proposed in [11]. An ontology describes the physical and functional interactions between the lower part of the real-time control elements based on the IEC 61499 architecture. The upper part of the control coordinates the set and sends reconfiguration orders if it detects an anomaly through the system knowledge base.

The authors in [12] use a reconfiguration manager linking the upper and lower part of the control. There are predefined configurations in an autochanger, so the designer must choose the correct configuration when an unexpected event occurs. The problem with this approach is that it is impossible to reconfigure the controlwhen no configuration is planned to mitigate the disruption. Another problem lies in the reconfiguration procedure because all the code is replaced even for a minor change.

3 Control Design Methodology

From the beginning of the IEC 61499 standard in 2005, researchers have developed design methods based on UML and SysML diagrams to facilitate control design systems [13]–[15]. UML is a modeling formalism suitable for IEC 61499 because it has an object-oriented vision with the class diagram corresponding to the principles of FB integrating their algorithm and variables. The problem is that the profiles and extensions presented remain close to the level of the control. The use of these diagrams requires knowledge of the programming languages used. Furthermore, these diagrams do not include the different parts of an automated system. The control of an automated system is highly dependent on the hardware part of the installation. The mechanical, electrical, and software parts of the installation are intimately linked: a change in one of these areas leads to a change in the other areas [16].

Vogel et al. have successfully integrated SysML into automation software development ranging from specification to implementation and maintenance [17]. However, they use the IEC 61131-3 language, and the method does not include predefined code during the code generation.

Our methodology uses a set of SysML diagrams to describe the production system from a general point of view without detailing the specifics of the programming language. One of the contributions is the use of models to facilitate design and reconfigurations of control architecture by integrating the mechanical connections between elements and customer's needs. The diagrams form a global knowledge base of the system that can be distributed among the holons for the agent part to enable cooperation and

[1] https://holobloc.com/doc/ita/index.htm

negotiation. Moreover, our approach allows the generation of the IEC 6199 FB network without writing any code for the real-time control part of holons.

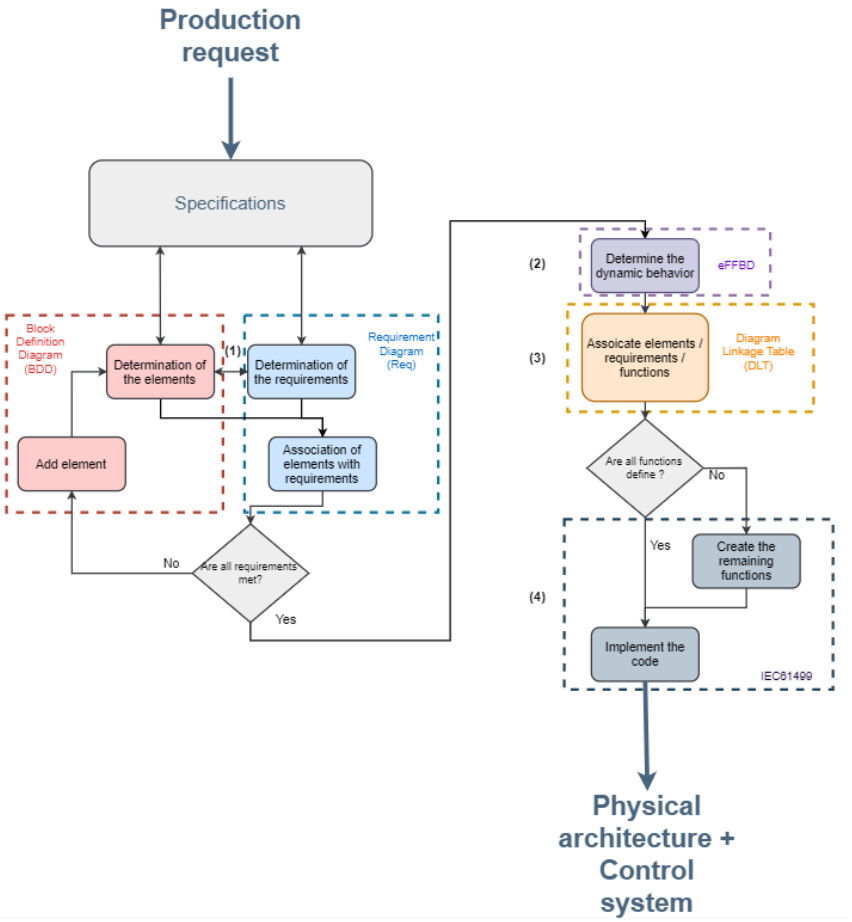

Fig. 1. Steps to design control system

The control design begins with the start of production following a customer's request (see Fig.1). The manufacturing system is a Reconfigurable Manufacturing System (RMS) where it is possible to add or remove production components to adapt the capabilities and functionalities of the system.

The first step is to define the specifications' requirements and the current production system (see Fig.1-1). Then, the requirement diagram (Req) defines the requirements for specifications and constraints. Next, Block Definition Diagram (BDD) defines the physical structure and links in the system. Finally, the components are linked to the requirements in Req. If all the requirements are met, the designer can proceed to the next step. Otherwise, it must add the available elements from a library that provides the necessary operations to validate the missing requirements.

The second step is to define the system's dynamic behavior to realize the requirements thanks to the enhanced Function Flow Block Diagram (eFFBD) (see Fig.1-2). Thus, the designer acts recursively to first satisfy the highest-level requirements before satisfying the requirements of the lowest levels.

Before moving on to code implementation, the final step is to link each diagram using the Diagram Linkage Table (DLT). DLT links the customer's requirements to the operations proposed by the system with the functions defined in the dynamic behavior (see Fig.1-3).

The control's implementation consists of replacing the functions described in the eFFBD with the IEC 61499 FB defined by system components (see Fig.1-4). The operations offered by the system components are basic operations such as enabling or disabling an actuator. Some functions in the eFFBD describe synchronization operations or operations that require the intervention of two independent components. The "Create the remaining functions" block corresponds to the creation of IEC 61499 blocks for these specific operations.

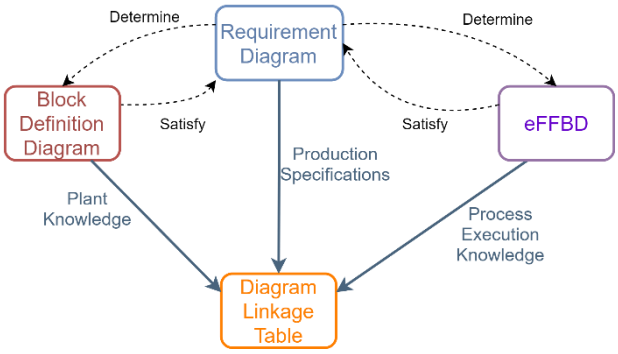

Fig. 2. Interdependencies of the knowledge base models

The specifications of the production define the physical components and the order of execution of the processes. Req, BDD, and eFFBD define high-level knowledge of the system from three points of view: (1) knowledge of the physical structure, (2) production specifications, and (3) knowledge of process execution (see Fig.2).

Our approach uses model-based engineering to structure production and plant knowledge. The knowledge base can be integrated into the holonic reference architecture: PROSA [18]. BDD gathers useful information for resource and order holons, Req information for product holons and eFFBD information for order and product holons.

4 Use Case

The methodology is implemented on a simulated manufacturing system using Factory I/O software[2] (see Fig.3) and EcoStruxure Automation Expert from Schneider

[2] https://realgames.co/

Electric. There are three types of parts that are provided from the entry of the system. Parts must be sorted according to their type to be brought to the appropriate exit. Sorting station is composed of two conveyors: an entry conveyor and an exit conveyor. The part detector determines the part type and is located on the entry conveyor. There are three sorters on the exit conveyor to push the part at the right exit.

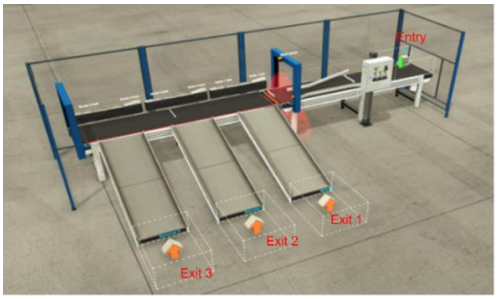

Fig. 3. Sorting station example

4.1 Requirement Diagram of the Sorting Station

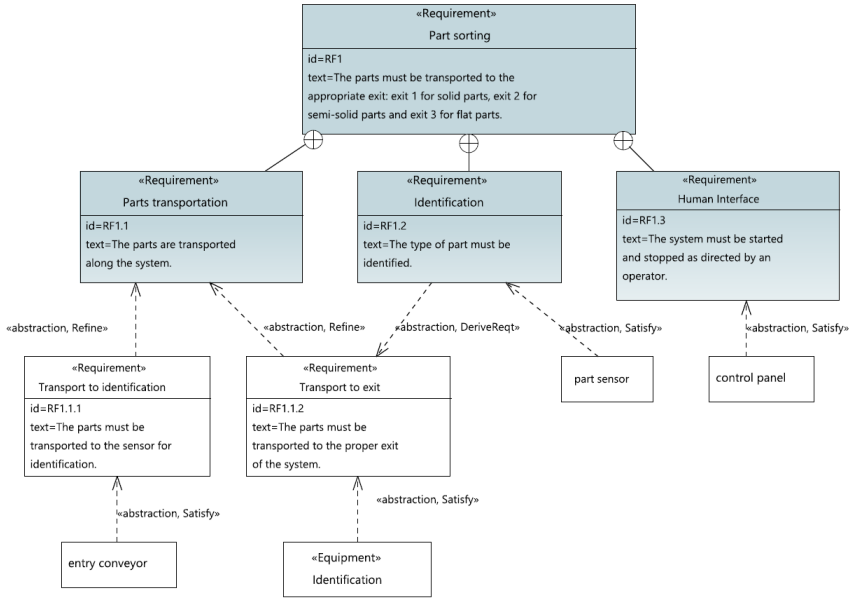

Fig. 4. Sorting station functional requirements

Requirement diagram (Req) is one of the diagrams defined by SysML. It graphically describes a requirement or constraint that the system must meet. First, the designer de-

duces the requirements of the specifications. This step is delicate because it is not formal, and the requirements may vary from one designer to another. Extensions of the stereotype "Trace" clarify the relationships between requirements and elements in other diagrams. They make it possible to establish a hierarchy in the requirements. They specify which requirement is satisfied by which element (see Fig.4).

4.2 Block Definition Diagram (BDD)

The Block Definition Diagram (BDD) allows the designer to specify the system's physical hierarchical decomposition and obtain an overall view of the operations proposed by each system component. A block is a modular unit of system description that can integrate structural and behavioral features. BDD describes composition relationships, aggregations, dependencies, block generalizations.

The elements of the system are determined hierarchically, the decomposition is described in [19]. The stereotypes "Cell", "Equipment", "Component", and "Controller" are defined to describe the control architecture (see Fig.5). This control architecture makes it possible to have components with clearly established limits to define the overall system best. It takes over the object-oriented structure of the physical structure of the plant. Therefore, the design or redesign of the control system following a configuration change is facilitated. It will be sufficient to identify the element(s) impacted by this change and to replace them with components having the same physical and functional characteristics. An Internal Block Diagram (IBD) accompanies BDD to define the physical and logical connections between the elements and the logical connections to the controller.

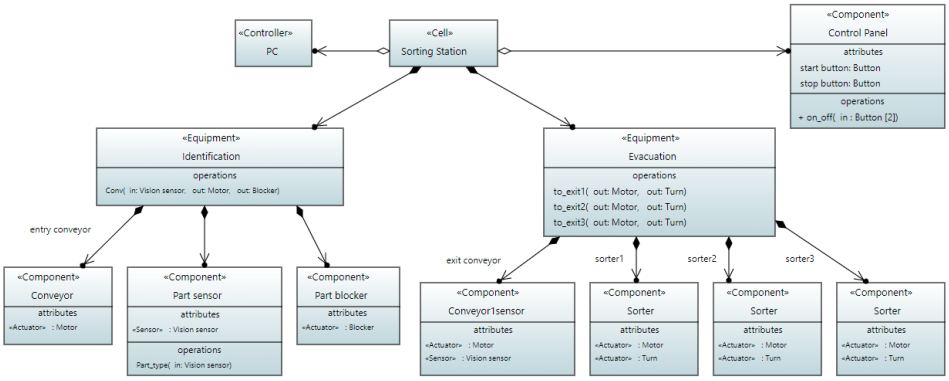

Fig. 5. Sorting station hierarchical decomposition

4.3 Enhanced Function Flow Block Diagram (eFFBD)

Enhanced Function Flow Block Diagram (eFFBD) models dynamic behavior, making it possible to model control flows between functions and data exchanges (see Fig.6). This diagram is ideal for modeling the dynamic behavior of the IEC 61499 FB network

because control flows between functions can be likened to events between function blocks. In addition, the modeling of data flows in parallel makes it possible to have a structure like the IEC 61499 code. Modeling in eFFBD allows a higher-level language with blocks of functions described by sentences. Therefore, it is more understandable than an IEC 61499 network.

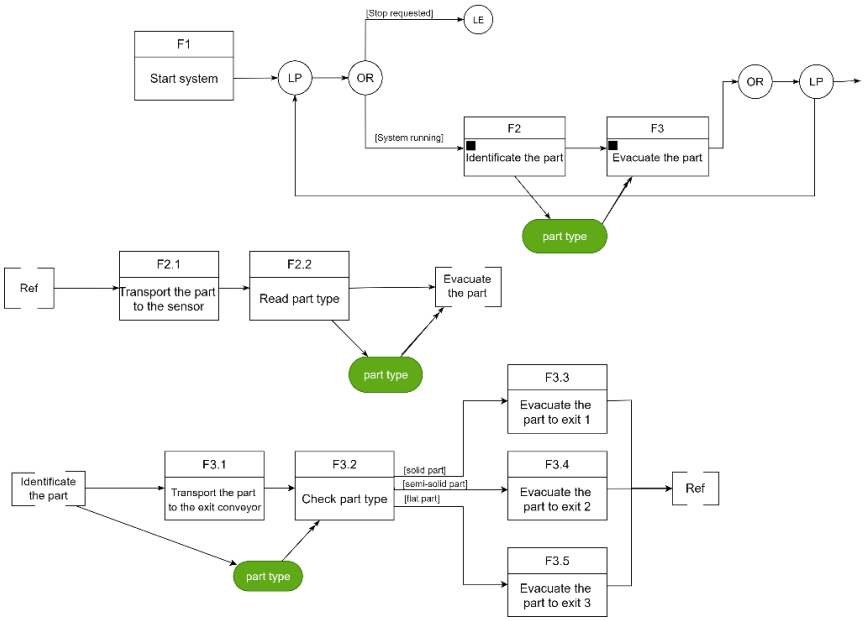

Fig. 6. Sorting station dynamic behavior

4.4 Diagram Linkage Table (DLT)

Figure 7 shows the DLT obtained by following the design steps. The rows of the table are the identifiers of eFFBD functions, and the columns are the identifiers of requirements in Req. The system operations must fill each DLT columns.

Specifications / Functions		RF1			
		RF1.1		RF1.2	RF1.3
		RF1.1.1	RF1.1.2		
F1					Pupitre.on_off()
F2	F2.1	Identification.conv()			
	F2.2			PartSensor.box_type()	
F3	F3.1		X		
	F3.2		X		
	F3.3		Evacuation.exit1()		
	F3.4		Evacuation.exit2()		
	F3.5		Evacuation.exit3()		

Fig. 7. Links between the knowledge base and the IEC 61499 FB

For example, the cell "Identification.conv()" corresponds to the operation "conv" proposed by the equipment "Identification" which includes the entry conveyor, the part detector, and the part blocker (see Fig.5). This operation meets the requirement RF1.1 (see Fig.4) and corresponds to the function F2.1 (see Fig.6). The crosses in the table correspond to operations that are not proposed by the system components. They can be coordination functions like the red cross in row F3.1.1 needs the simultaneous activation of entry and exit conveyor to bring the part from one to the other. The red cross in row F3.2 corresponds to a function with multiple exits, a choice of sequence. The designer will have to develop IEC 61499 FB containing the corresponding logic to complete the IEC 61499 network.

5 Knowledge base enabling reconfigurability

It is essential to measure the impact of a disruption on the system to provide an appropriate and optimal response. DLT is a central element in the control design but also throughout the life cycle of the system. It makes it possible to link all the system's knowledge and highlight the interdependence of the different models. The graphical representation of dependencies allows for a quick analysis of the impact of a disturbance in one model on the others. The designer can then act more quickly to address this disturbance and reduce the configuration change duration.

Manufacturing systems are subject to multiple internal and external disruptions. To overcome these disturbances, it is often necessary to change the configuration of the system. Reconfigurations can occur at the level of plant's physical structure or the level of organizational structure. We have identified three types of disturbances requiring a reconfiguration (see Table.1). Disturbances external to the system are specification changes, while failures of a component or controller are internal.

Table 1. Response to disturbances in our methodology

Reconfiguration types		Solution
Specification changes		Converting the functionality of a component and rearranging program connections or physical rearrangement
		Add/Remove Physical Component
Physical	Component failure	Identifying the failed component, converting the functionality of a component, and rearranging program connections or physical rearrangement
		Identifying the failed component and adding a new component
Software	Controller failure	Converting the functionality of a component and rearranging program connections or physical rearrangement
		Uploading code to other controllers
		Replacing the Controller

Table 2 presents the solutions to mitigate these disturbances and the impacts on the knowledge base. Adding or removing a component cause an upheaval in the entire knowledge base, but it is not necessarily the most complex operation. Indeed, this reconfiguration may result in only minor changes in each of the areas. Assisting in choosing the optimal solution for the designer according to the type of reconfiguration will be part of future work.

Table 2. Impact of disturbances on the knowledge base

Solution	Detail	Impact
Add/Remove Physical Component	1. Modification of BDD and link between the new operations and the specifications 2. Integration of new operations with eFFBD 3. Significant change in DLT and code if new specifications	BDD, REQ, eFFBD, DLT
Converting the functionality of a component and rearranging program connections	1. Identification of unfulfilled specifications and creation of links with physical components 2. Modification of the eFFBD to meet the new specifications 3. Impact of changes on DLT and code	REQ, eFFBD, DLT
Physical rearrangement	1. Modification of the IBD to reflect the change in physical connections 2. Modification of the eFFBD to meet the new specifications 3. Impact of changes on DLT and code	IBD, eFFBD, DLT
Uploading code to other controllers	1. Identification of the operations carried out by the controller 2. Mapping function blocks to other controllers 3. Downloading codes to controllers	BDD, IBD

6 Conclusion

Cyber-physical systems are composed of highly interconnected sub-systems to improve responsiveness, robustness, and flexibility. There are many works around integrating new technologies in the development of CPS control architecture, especially around holonic systems. In addition, technological advances in manufacturing are leading to increased complexity in system design. This paper presents a SysML-based design method for designing CPS with its control architecture. The architecture is designed on the principle of plug and produce where it is possible to add or remove components. The components propose different operations to the designer coded by already verified and validated IEC 61499 FB. SysML facilitates the assembly of components following the proposed methodology. Our approach provides knowledge for holon decision-making and real-time code for holon process control.

The control system based on IEC 61499 is developed from the knowledge base to reduce the complexity by raising abstraction [20]. The adaptation of the system to disturbances is one of the contributions of this paper. Reconfiguration time is reduced thanks to quick identification of the affected components and the proposed solutions according to the perturbations.

Future work will focus on transforming the knowledge base into an ontology and automatically generating code. Currently, each step of the methodology is done manually. The objective is to automate the passage of the real-time code after the design of the diagrams. MOF Model To Text Transformation Language (MOFM2T) standard allows the passage of the model into text. It is then possible to obtain an IEC 61499 application thanks to the XML file obtained with MOF2T. Moreover, combining an ontology with explicit semantics makes the manufacturing environment understandable to the control system and is a way to automatic reconfiguration [20].

Kruger and Basson define evaluation criteria for holonic control implementation in [21]. The following quantitative criteria will be used: development and reconfiguration time and code reuse and extension rate to evaluate the reconfigurability and the complexity of our approach. Our methodology will be used to design the architecture of a real manufacturing system: Cellflex 4.0[3] to study the increase of complexity on an industrial system use case.

Acknowledgment

This work is integrated into the project FFCA (Factories of Future Champagne-Ardenne). The authors would like to thank the region Grand-Est within the project FFCA (CPER PFEXCEL).

References

1. R. Baheti and H. Gill, "Cyber-physical systems," *The impact of control technology*, pp. 161–166, Mar. 2011.
2. L. Monostori *et al.*, "Cyber-physical systems in manufacturing," *CIRP Annals*, vol. 65, no. 2, pp. 621–641, Jan. 2016, doi: 10.1016/j.cirp.2016.06.005.
3. B. Vogel-Heuser, J. Lee, and P. Leitao, "Agents enabling Cyber-Physical Production Systems," *Automatisierungstechnik (at)*, vol. 63, no. 10, pp. 777–789, 2015, doi: 10.1515/auto-2014-1153.
4. W. Derigent, O. Cardin, and D. Trentesaux, "Industry 4.0: contributions of holonic manufacturing control architectures and future challenges," *ArXiv*, vol. abs/2002.04525, 2020.
5. L. Wang and A. Haghighi, "Combined strength of holons, agents and function blocks in cyber-physical systems," *Journal of Manufacturing Systems*, vol. 40, pp. 25–34, Jul. 2016, doi: 10.1016/j.jmsy.2016.05.002.
6. F. Mhenni, J.-Y. Choley, O. Penas, R. Plateaux, and M. Hammadi, "A SysML-based methodology for mechatronic systems architectural design," *Advanced Engineering Informatics*, vol. 28, no. 3, pp. 218–231, Aug. 2014, doi: 10.1016/j.aei.2014.03.006.
7. V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 768–781, Nov. 2011, doi: 10.1109/TII.2011.2166785.

[3] https://www.univ-reims.fr/meserp/cellflex-4.0/cellflex-4.0,9503,27026.html

8. G. Black and V. Vyatkin, “Intelligent Component-Based Automation of Baggage Handling Systems With IEC 61499,” *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 2, pp. 337–351, Apr. 2010, doi: 10.1109/TASE.2008.2007216.
9. J. Yan and V. Vyatkin, “Extension of reconfigurability provisions in IEC 61499,” in *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, Sep. 2013, pp. 1–7. doi: 10.1109/ETFA.2013.6648026.
10. A. Zoitl, C. Sunder, and I. Terzic, “Dynamic reconfiguration of distributed control applications with reconfiguration services based on IEC 61499,” in *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS’06)*, Jun. 2006, pp. 109–114. doi: 10.1109/DIS.2006.28.
11. W. Lepuschitz, A. Zoitl, M. Vallée, and M. Merdan, “Toward Self-Reconfiguration of Manufacturing Systems Using Automation Agents,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 1, pp. 52–69, Jan. 2011, doi: 10.1109/TSMCC.2010.2059012.
12. S. Olsen, J. Wang, A. Ramirez-Serrano, and R. W. Brennan, “Contingencies-based reconfiguration of distributed factory automation,” *Robotics and Computer-Integrated Manufacturing*, vol. 21, no. 4, pp. 379–390, Aug. 2005, doi: 10.1016/j.rcim.2004.11.011.
13. V. Dubinin, V. Vyatkin, and T. Pfeiffer, “Engineering of Validatable Automation Systems Based on an Extension of UML Combined With Function Blocks of IEC 61499,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Apr. 2005, pp. 3996–4001. doi: 10.1109/ROBOT.2005.1570732.
14. T. Hussain and G. Frey, “UML-based Development Process for IEC 61499 with Automatic Test-case Generation,” in *2006 IEEE Conference on Emerging Technologies and Factory Automation*, Sep. 2006, pp. 1277–1284. doi: 10.1109/ETFA.2006.355407.
15. C. A. García, E. X. Castellanos, and M. V. García, “UML-Based Cyber-Physical Production Systems on Low-Cost Devices under IEC-61499,” *Machines*, vol. 6, no. 2, 2018, doi: 10.3390/machines6020022.
16. K. Kernschmidt and B. Vogel-Heuser, “An interdisciplinary SysML based modeling approach for analyzing change influences in production plants to support the engineering,” in *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2013, pp. 1113–1118. doi: 10.1109/CoASE.2013.6654030.
17. B. Vogel-Heuser, D. Schütz, T. Frank, and C. Legat, “Model-driven engineering of Manufacturing Automation Software Projects – A SysML-based approach,” *Mechatronics*, vol. 24, no. 7, pp. 883–897, Oct. 2014, doi: 10.1016/j.mechatronics.2014.05.003.
18. H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters, “Reference architecture for holonic manufacturing systems: PROSA,” *Computers in Industry*, vol. 37, no. 3, pp. 255–274, Nov. 1998, doi: 10.1016/S0166-3615(98)00102-X.
19. A. Parant, F. Gellot, A. Philippot, and V.Carre-Menetrier, “Hierarchical Intelligent Component-Based Development for the Design of Cyber-Physical Control Architecture,” presented at the 5th Int. Conference on Control and Fault-Tolerant Systems (SysTol), 2021.
20. M. Merdan, T. Hoebert, E. List, and W. Lepuschitz, “Knowledge-based cyber-physical systems for assembly automation,” *null*, vol. 7, no. 1, pp. 223–254, Jan. 2019, doi: 10.1080/21693277.2019.1618746.
21. K. Kruger and A. H. Basson, “Evaluation criteria for holonic control implementations in manufacturing systems,” *null*, vol. 32, no. 2, pp. 148–158, Feb. 2019, doi: 10.1080/0951192X.2018.1550674.