A Portability Study of IEC 61499: Semantics and Tools

Cheng Pang[1], Sandeep Patil[1], Chen-Wei Yang[1], *Members IEEE*, Valeriy Vyatkin[1,2], *Senior Member IEEE*, and Anatoly Shalyto[3]

[1]Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Luleå, Sweden

[2]Department of Electrical Engineering and Automation, Aalto University, Finland

[3] Computer Technologies Department, ITMO University, St. Petersburg, Russia

{cheng.pang.phd, vyatkin}@ieee.org, {sandeep.patil, chen-wei.yang}@ltu.se, shalyto@mail.ifmo.ru

***Abstract*—The second edition of the IEC 61499 standard aims to clarify the interpretation ambiguities of function block's execution semantics. This resolves the pivotal issue of realizing portable and interoperable implementations of the IEC 61499 reference architecture. As the IEC 61499 standard is about entering its technology takeoff phase, these clarifications are timely and important. It is hence expected that more innovators of automation software tools, runtime environments, and control hardware will start adopting this technology. To assist such adoption, this paper presents a study of existing IEC 61499 tools' portability issues. In particular, the features of currently active IEC 61499 tools, such as FBDK, ISaGRAF, 4DIAC, and nxtStudio are outlined. Their incompatibility issues due to different execution semantics are exemplified. Moreover, it is also illustrated in this paper how these issues can be addressed by complying with the updated norms.**

Keywords—IEC 61499; FBDK; ISaGRAF; 4DIAC; nxtStudio

I. INTRODUCTION

In the field of industrial automation systems, the ever increasing demands for decentralized control and the exponential growth of control complexity have accelerated the shift from monolithic and centralized design paradigms towards more reconfigurable and distributed engineering approaches. This trend has been reflected in the development of the IEC 61499 standard [1] for distributed industrial-process measurement and control systems. The IEC 61499 standard aims to establish an open, component-oriented, and platform-independent development framework to improve re-usability, re-configurability, interoperability, portability, and distribution of control software for complex distributed systems.

To achieve the above goal, the IEC 61499 standard first extends the *Function Block* (FB) notion from the IEC 61131-3 [2] standard with event-driven execution semantics. In IEC 61499, the basic design construct is FB. Each FB consists of a graphical event-data interface and a set of executable functional specifications. FBs can be interconnected into a network using event and data connections to specify the entire control system. Execution of an individual FB in the network is triggered by the events it received. This well-defined event-data interface and the encapsulation of local data and control algorithms make each FB a reusable functional unit of software. Secondly, in IEC 61499, an FB network should be easily partitioned and then distributed to control devices over various communication networks while still preserving its original control logic's execution semantics. As a result, distributed control systems can be designed and configured at system-level independent to implementation platform and hardware architecture.

The shift from centralized control systems to distributed control systems is one of the major evolution trends in industrial automation [3, 4]. This trend was also reflected in the development of the IEC 61499 standard, which establishes an event-driven modular design framework for distributed control systems. Over the past decade, the applicability of the IEC 61499 standard in distributed control systems has been extensively studied in many projects, such as airport baggage handling systems [5], manufacturing control [6], mechatronics [7], building automation systems [8], machining [9], process control [10], and smart grid [11]. These case studies have confirmed many advantages of IEC 61499 over the mainstream PLC technology based on the IEC 61131-3 standard in terms of design and re-design efficiency, and better interoperability and reusability. However, these studies also revealed many pitfalls of the first edition, which are primarily due to the non-exhaustive definition of FB's execution semantics. This, on one hand, gives software vendors sufficient freedom to adapt the IEC 61499 standard into their existing tool frameworks, such as ISaGRAF Workbench [12]. However, on the other hand, different IEC 61499 implementations may not be compatible to one another. Such incompatibility directly results in portability and inter-operability issues that are against the standard's original intention. As discussed in [13], the industrial adoption of the IEC 61499 technology is about entering its takeoff phase. It is therefore expected that more innovators of automation software tools, runtime environments, and control hardware will start adopting IEC 61499 technologies.

In order to resolve the semantic incompatibility, the second edition of IEC 61499 has been published in 2012 to provide more rigorous behavioral specifications of the FB architecture. For new adopters' reference, this paper presents a survey of portability issues in existing IEC 61499 tools. In particular, issues that can be addressed by complying with the second edition are highlighted. Other potential problems that may cause portability issues are discussed as well. This work follows [14], where portability, configurability, and interoperability of the IEC 61499 compliant tools were first discussed.

The rest of this paper is structured as follows. Section II presents a brief overview of IEC 61499 tools. Then, Section III summarizes and exemplifies the portability issues in existing tools. In Section IV, semantics amendments in the second edition are outlined with illustration on existing tools' compliance. Section V further discusses other potential portability issues that must be considered when implementing the IEC 61499 standard. Finally, this paper is concluded in Section VI.

II. IEC 61499 ENGINEERING TOOLS

A variety of IEC 61499 tools has been developed for both academic research and commercial purposes. The *Function Block Development Kit* (FBDK) [15] is the first IEC 61499 engineering tool, which has been used to demonstrate the IEC 61499 technologies since its early conception time. FBDK contains two parts: an editor and a runtime (FBRT). FBRT implements a *Non-Preemptive Multi-Threading Resource* (NPMTR) execution model [16], which is based on a depth-first FB scheduling mechanism. As FBDK is developed using Java, its hardware support is limited (e.g. [17, 18]). The current FBDK 2.0 release is the first tool supporting the second edition of IEC 61499.

Inspired by FBDK, a number of research projects have been conducted to address the insufficiencies of FBDK or to incorporate the IEC 61499 technologies in their existing tool frameworks. This resulted in several IEC 61499 engineering tools or platforms. For example, FBench [19] provided an open-sourced version of FBDK for experimentation. The CORFU engineering support system [20] attempted to establish a framework for model-driven development of control applications using IEC 61499 and *Unified Modeling Language*. The TORERO engineering platform [21] further covered the entire development life cycle of distributed control systems using IEC 61499. At last, the 4DIAC initiative [22, 23] is currently the only active open-sourced research-based IEC 61499 compliant automation and control environment. Similar to FBDK, the 4DIAC framework consists of an editor based on the Eclipse platform [24] and a C++ runtime, called FORTE, implementing a sequential FB execution model with event dispatcher.

The first commercial engineering tool supporting IEC 61499 is called ISaGRAF Workbench [25]. Starting from adapting the IEC 61499 FB notation and event-driven notion into its existing PLC engineering framework, the ISaGRAF Workbench is gradually becoming an IEC 61499 compliant tool. However, due to the backwards compatibility issues, ISaGRAF Workbench still uses a cyclic scan-based execution model. In contrary, nxtStudio [26] is another industrial-grade engineering environment which fully implemented the IEC 61499 architecture. The nxtStudio's runtime, nxtRT61499F, is an extension to FORTE with additional infrastructure for testing, debugging, and deploying IEC 61499 applications. Both ISaGRAF Workbench and nxtStudio support a wide range of control hardware.

III. PORTABILITY ISSUES IN MAIN IEC 61499 TOOLS

This paper focuses on the four currently active tools: FBDK, ISaGRAF Workbench, 4DIAC, and nxtStudio. In this section, their portability issues are investigated.

A. Portability of Library Elements

In order to realize data exchange between different tools, the IEC 61499 standard has specified the textual syntax for storing library elements (e.g. data types, FB types, etc.) in XML files. Both FBDK and 4DIAC strictly conform to the normative syntax. This makes their library elements fully portable to each other. nxtStudio further extends the normative syntax to support additional design features. In general, with limited modifications, it is possible to import library elements from FBDK and 4DIAC to nxtStudio and vice versa. In contrary, ISaGRAF Workbench uses its proprietary file format and thus makes its library elements completely non-portable to other tools.

Moreover, as nxtStudio pursuits the *Model-View-Controller* design pattern, a concept called *Composite Automation Type* (CAT) FB has been implemented to combine control logic, visualization, and plant model into a single FB. The visualization part is coded in C#. To allow the data exchange between .Net runtime and nxtStudio's FB runtime (i.e. nxtRT61499F) a set of proprietary *Service Interface Function Blocks* (SIFBs) are automatically integrated in the CAT FBs. This prevents CAT FBs from being ported to other tools. FBDK and 4DIAC allows the FB algorithms to be programmed in Java or C++. This prevents such FBs to be ported to nxtStudio which supports only ST language. TABLE I. below summarizes the portability of library elements between different tools.

TABLE I. LIBRARY ELEMENT PORTABILITY

	FBDK	**4DIAC**	**nxtStudio**	**ISaGRAF**
FBDK	-	Full	Partial	N/A
4DIAC	Full	-	Partial	N/A
nxtStudio	Partial	Partial	-	N/A
ISaGRAF	N/A	N/A	N/A	Full

B. Portability Issues due to Semantic Ambiguities

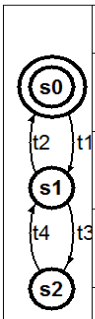
Another cause of portability issues is the incomplete or ambiguous specifications of FB execution semantics. The portability of an FB application A between platforms that comply with execution semantics s_1 and s_2 can be defined as the equivalence of behavior $B(A, s_1)=B(A, s_2)$. However, brute-force check of the equivalence can be prohibitively complex. Instead, one can check the A's model under semantics s, i.e. $M(A, s)$, against the comprehensive set of requirements R (functional and non-functional, including safety and liveness). Denoting the set of model-checking results as $C(M(A, s), R)$, we define the application A to be portable between semantics s_1 and s_2 if the model-checking gives equivalent results, i.e.:

$$P(A, s_1, s_2) \triangleq C(M(A, s_1), R) = C(M(A, s_2), R) \quad (1)$$

The rest of this section will delve into the illustration of some practical semantic issues.

1) Event Clearance Rule and EC Transition Evaluation

The sequence of algorithm invocations of a basic FB is defined in its *Execution Control Chart* (ECC). The operations of ECC, as per the first edition of IEC 61499, are specified in the *ECC Operation State Machine* (OSM) shown in Fig. 1:



State	Operations	Transition	Conditions
s0	Sample inputs	t1	Input event occurs
s1	Evaluate transitions	t2	No transitions clears
s2	Perform actions	t3	A transition clears
		t4	Actions completed

Fig. 1. ECC Operation State Machine [27].

As pointed out in many papers [28, 29], the problem here is the undefined event clearance rule: what is the lifetime of an event input in ECC. As a result, software vendors must implement their own event clearance rules. This led to different execution results. For example, in nxtStudio an FB run will only terminate until:

a. A guard condition is evaluated false; or,

b. The same EC state has been visited twice.

As illustrated in Fig. 2, at START state, upon the arrival of INIT event with QI=true (and assuming QI is not set to FALSE in the INIT Algorithm), the INIT event will be cleared after the second visit to the START state. Note that in nxtStudio events can be manually cleared by setting their values to FALSE in algorithms. For instance, by setting INIT:=FALSE in the INIT algorithm, the aforementioned FB run will stop at STATE1 state now. However, this trick is not portable and is specific to nxtStudio only.

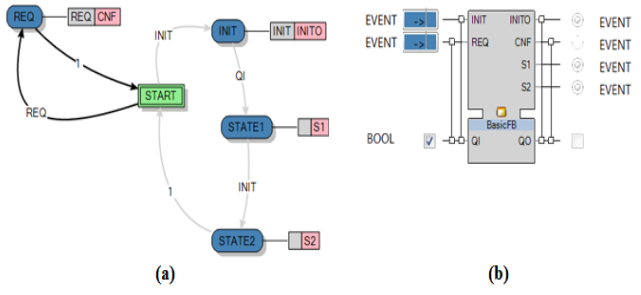


Fig. 2. nxStudio Event Clearance Example: (a) ECC and (b) FB Interface.

Both FBDK and 4DIAC behave the same for the above scenario. As shown in Fig. 3, under the same setting the FB run will stop at STATE1 state. This indicates that in FBDK and 4DIAC event input is only used once in a single FB run.

The problem arises when QI is FALSE when in START state and INIT event is triggered. In FBDK the control is stuck in the INIT state forever. The transition condition for OSM *t1* in the first standard is "invoke ECC" with a further comment stating that this "transition is activated by the presence of an event in an event input". In other words, the arrival of either INIT or REQ event input shall trigger the INIT→STATE1 EC transition. Therefore, the FBDK's implementation is incorrect. Contrarily, in nxtStudio and 4DIAC, the arrival of any input will trigger the INIT→STATE1 EC transition.

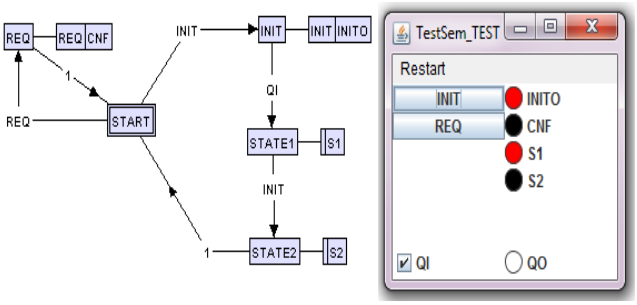


Fig. 3. FBDK Event Clearance Example: (a) ECC and (b) Execution Result.

At last, Fig. 4 presents the equivalent FB example implemented in ISaGRAF Workbench. In particular, the functions of ECC are realized in a *Sequential Function Chart* (SFC). As the execution model of ISaGRAF Workbench is based on cyclic scan, the event clearance rule is completely different from others. In the latest ISaGRAF Workbench 6.1, during each scan cycle all EC transitions will be evaluated. Events will not be cleared until the end of a scan. It can be expected that the execution results will differ much from other tools. A more comprehensive comparison between ISaGRAF and FBDK can be found in [30].

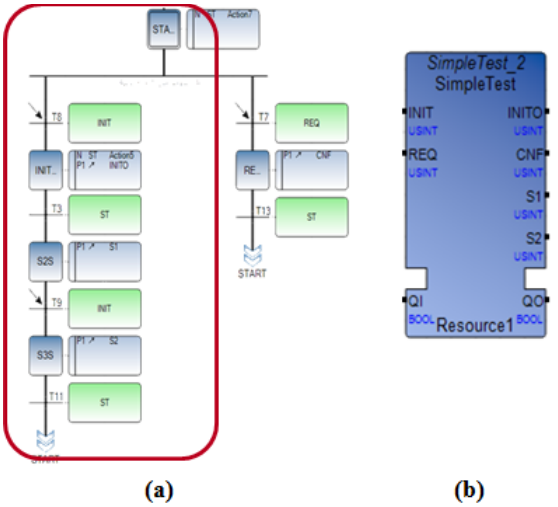


Fig. 4. ISaGRAF Workbench Event Clearance Example: (a) SFC and (b) FB Interface.

2) Arrival of Event Inputs

Event inputs can arrive in different ways in different execution models. As discussed in [31], there could be three problematic situations regarding arrival of event inputs as

shown in Fig. 5. The situation of simultaneous event arrival can only occur in ISaGRAF Workbench. Considering IEC 61499 applications running on ISaGRAF Workbench are, in general, non-portable to other three tools, this situation can be omitted here. The second situation states that an FB receives a new event while it is busy processing the previous event. In case of 4DIAC and nxtStudio this event will be buffered while in FBDK this event will be lost. If the lost event is critical, then the execution result will be completely different. For the last situation, as explained in the previous section, it will still trigger the evaluation of EC transition with guard condition only in nxtStudio and 4DIAC but not in FBDK.

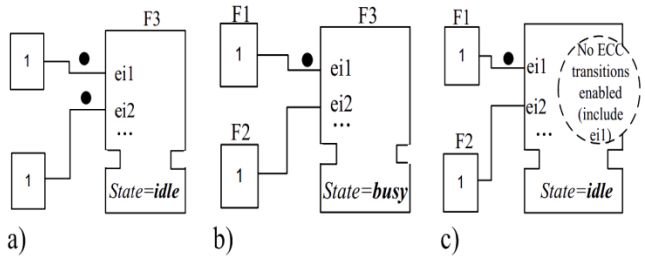


Fig. 5. Arrival of Event Inputs: (a) Simultaneous, (b) Arrival in Busy State, and (c) Arrival in Irrelevant State [30].

3) *Data Sampling and Data Latching*

Another portability issue is caused by the data sampling rule and data latching mechanism. As specified in the first edition, on the arrival of an event input all the associated data inputs will be sampled. Similarly, prior to emission of event output, the associated data outputs will be updated first. In nxtStudio, whenever an event input arrives or an event output is to be emitted all the data inputs/outputs, no matter associated or not, will be sampled/updated. Similarly, as ISaGRAF Workbench uses a cyclic execution model, all the data inputs/outputs are sampled/updated in each scan cycle. The ignorance of data association may cause potential issues due to the use of inconsistent data in algorithms.

Furthermore, in nxtStudio, all composite FBs are treated as flattened FB network without latching the corresponding data inputs/outputs. As discussed in [28] and illustrated in Fig. 6, incorrect flattening will omit the corresponding data sampling. Therefore, the identical application will behave differently in, for example, 4DIAC which ensures strict event and data associations. To illustrate this, a testing FB network has been set up as shown in Fig. 7. The main FB of interest in this testing FB network is the composite FB called TestAlgoComp, whose detailed interface and internal composition are presented in Fig. 8.

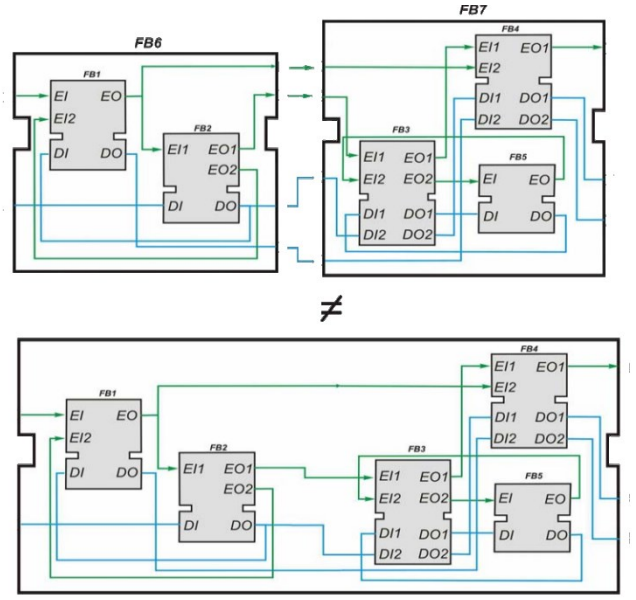


Fig. 6. Incorrect Flattening of Composite FBs [28].

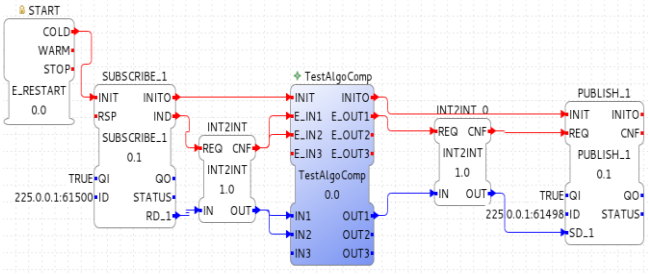


Fig. 7. Testing FB Network.

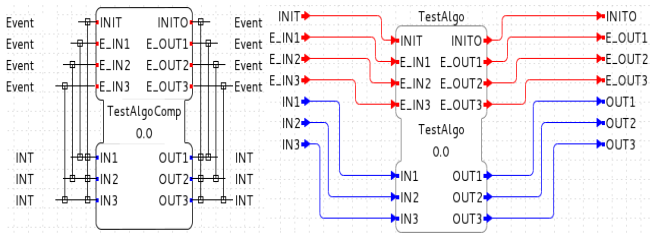


Fig. 8. TestAlgoComp FB: Interface and Internal Composition.

The TestAlgoComp FB consists of a single basic FB called TestAlgo, whose ECC and interface are further shown in Fig. 9. When TestAlgo receives E_IN1 event, the value of IN2 data input is copied to OUT1 data output. On E_IN2 event, the value of IN3 is copied to OUT2. Similarly, on E_IN3 event the value of IN1 is copied to OUT3. In 4DIAC, consider the case when there is connection to TestAlgoComp.E_IN2 (Fig. 7) and no connection from TestAlgoComp.E_IN2 to TestAlgo.E_IN2. Since IN2 is associated with E_IN2 of TestAlgo, the value of TestAlgo. IN2 will never be updated. The same is not true in case of nxtStudio. It is not applicable to ISaGRAF as it does not have event and data associations.

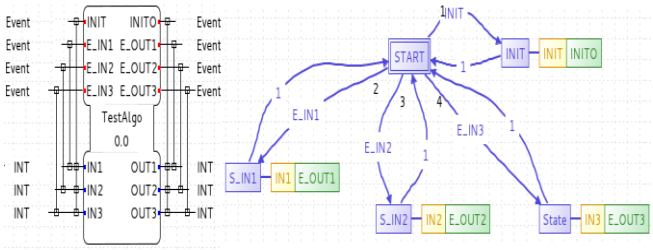


Fig. 9. The Interface and ECC of TestAlgo Basic FB.

IV. Semantic Amendaments in 2ND Edition

Most of the portability issues discussed in the previous section are caused by the non-exhaustive definition of FB execution semantics. Thus, significant amount of refinement work has been done in the second edition to remove or clarify the semantic ambiguities. The second edition aims to establish a well-defined model of FB execution semantics for device vendors and application developers. The second edition primarily addressed the following issues causing ambiguities:

- Event semantics issue;
- Concurrency issue; and,
- Data consistency issue.

A. Event Semantic Issue

Within an ECC, an event input now can only be used once during the evaluation of EC transitions. After the new state has been reached, only EC transitions with pure guard conditions may be further evaluated. Moreover, evaluations of EC transitions with pure guard conditions may be triggered by irrelevant events. Considering the ECC Operation State Machine (OSM) shown in Fig. 1, the second edition mentioned that if *s1* state is reached due to *t1* then the guard conditions associated with the sampled event or any transitions that without any events will be evaluated and when *s1* state is reached due to *t3* then only transitions that contains purely non-event based guard conditions will be evaluated. Referring to the nxtStudio example illustrated in Fig. 2, under the second edition, *OSM* is initially in *s1*, the INIT event is sampled, OSM moves to *s2* and evaluates the transitions and since there is a valid transition the OSM moves to *s3*. It executes the INIT algorithm and emits the INITO event and this completes the action. The OSM now moves back to state *s2* and evaluates for any non-event associated guard conditions and it finds QI and executes that transition by again changing the OSM state to *s3* and hence emits S2 event (Fig. 3) and again moves back to OSM *s1* state. Since there is no more non-event based guard condition, the OSM moves back to start state, *s0*, and waits for sampling any new input events. In case at INIT state and QI is FALSE, then arrival of either INIT or REQ event will trigger the evaluation of INIT→STATE1 EC transition. If in any of the evaluations, QI is TRUE, then STATE1 will be reached.

B. Concurrency Issue

In the previous edition, the standard only specifies that the invocation of an FB is atomic. However, this does not guarantee that no multiple algorithms will be executed at the same time within an FB. In the second edition, it is prescribed in the ECC OSM descriptions that within a resource at any instance of time only one event input of an FB can occur. As a result, it is impossible to have multiple algorithms executing at the same time, which makes the FB execution more deterministic. For example, situation a) and b) illustrated in Fig. 5 will not occur with the new execution semantics.

C. Data Consistency Issue

At last, in the new edition, the data sampling rule is enforced and further clarified. It further prescribes that during the execution of algorithms the values of data variables used must remain stable. This implies the implementation of data latching mechanisms.

V. Potential Portability Issues

The second edition of IEC 61499 has addressed many crucial semantic issues causing interpretation ambiguities. However, the standard still leaves sufficient freedom for implementing different execution models of resources. This may also cause potential portability issues. For example, although the standard states that "resources might need to schedule the execution of algorithms in a multitasking manner", further details are not specified.

Moreover, the event scheduling mechanism implemented by the resources would also affect the portability. As discussed in [32], if only events are scheduled by, for instance, sending all event outputs to an event queue; then the events and associated data might be inconsistent. Contrarily, if both events and associated data are buffered in the queue, this might be too resource consuming. If one runtime buffers event only while another runtime buffers both event and data, there may be portability and interoperability issues.

Thirdly, when real-time constraints must be considered by the runtime, events must be prioritized to determine their processing orders. It is undoubtedly that there will be portability issues between real-time and non-real-time IEC 61499 runtimes.

VI. Conclusions

This paper presented a brief survey of portability issues of existing engineering tools for the IEC 61499 standard. Many of these portability issues are due to the incomplete specifications of FB execution semantics. The release of IEC 61499's second edition has clarified most of the semantic ambiguities found in the first edition. 4DIAC and FBDK already support the new edition. It is expected that this will establish a more rigorous common understanding for vendors of IEC 61499 software tools, runtime platforms, and control hardware.

In this paper, some other potential issues that may still cause portability problems were also discussed. In order to achieve better interoperability and portability, additional techniques should be applied. For example, by formally modeling the IEC 61499 architecture and its execution semantics, it is possible to formally analyze and validate its portability against certain runtime platforms. Alternatively,

approaches based on semantics-robust design patterns [31] can be applied for developing execution-model-independent IEC 61499 applications. Despite of the effort made in [31], there is an open research question on the existence of more practical design rules and patterns, following which an FB application would behave equivalently under different execution models and under known restrictions.

ACKNOWLEDGMENT

The authors are grateful to the nxtControl GmbH, ICS Triplex ISaGRAF, PROFACTOR GmbH, and HOLOBLOC Inc. for providing the IEC 61499 tools for evaluation.

REFERENCES

[1] "Function blocks - Part 1: Architecture," *IEC 61499-1: 2012,* p. 245, 2012.

[2] "Programmable controllers - Part 3: Programming languages," *IEC 61131-3: 2013,* p. 226, 2013.

[3] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review," *IEEE Transactions on Industrial Informatics,* vol. 7, pp. 768-781, 2011.

[4] V. Vyatkin, "Software Engineering in Industrial Automation: State-of-the-Art Review," *IEEE Transactions on Industrial Informatics,* vol. 9, pp. 1234-1249, 2013.

[5] J. Yan and V. V. Vyatkin, "Distributed Execution and Cyber-Physical Design of Baggage Handling Automation with IEC 61499," in *9th IEEE International Conference on Industrial Informatics (INDIN 2011)*, Caparica, Lisbon, Portugal, 2011, pp. 573-578.

[6] M. Colla, A. Brusaferri, and E. Carpanzano, "Applying the IEC-61499 Model to the Shoe Manufacturing Sector," in *11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2006)*, Prague, Czech Republic, 2006, pp. 1301-1308.

[7] M. Sorouri, V. Vyatkin, and S. Xie, "Distributed Control Design of Medical Devices Using Plug-and-Play IEC 61499 Function Blocks," in *19th International Conference Mechatronics and Machine Vision in Practice (M2VIP 2012)*, Auckland, New Zealand, 2012, pp. 450-455.

[8] C. Pang, V. Vyatkin, Y. Deng, and M. Sorouri, "Virtual Smart Metering in Automation and Simulation of Energy-Efficient Lighting System," in *18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2013)*, Cagliari, Italy, 2013, pp. 1-8.

[9] L. Wang, G. Adamson, M. Holm, and P. Moore, "A review of function blocks for process planning and control of manufacturing equipment," *Journal of Manufacturing Systems,* vol. 31, pp. 269-279, 2012.

[10] D. Dimitrova, S. Panjaitan, I. Batchkova, and G. Frey, "IEC 61499 Component Based Approach for Batch Control Systems," in *17th World Congress*, Seoul, Korea, 2008, pp. 10875-10880.

[11] G. Zhabelova and V. Vyatkin, "Multiagent Smart Grid Automation Architecture Based on IEC 61850/61499 Intelligent Logical Nodes," *IEEE Transactions on Industrial Electronics,* vol. 59, pp. 2351-2362, 2012.

[12] J. Chouinard and R. Brennan, "Software for Next Generation Automation and Control," in *4th IEEE International Conference on Industrial Informatics (INDIN 2006)*, Singapore, 2006, pp. 886-891.

[13] T. Strasser, J. H. Christensen, A. Valente, J. Chouinard, E. Carpanzano, A. Valentini*, et al.*, "The IEC 61499 Function Block Standard: Launch and Takeoff," presented at the ISA Automation Week 2012, Orlando, US, 2012.

[14] J. H. Christensen, T. Strasser, A. Valentini, V. Vyatkin, and A. Zoitl, "The IEC 61499 Function Block Standard: Software Tools and Runtime Platforms," presented at the ISA Automation Week 2012, Orlando, US, 2012.

[15] Holobloc Inc. (2014). *FBDK 2.1 - The Function Block Development Kit* [Online]. Available: http://www.holobloc.com/fbdk2/

[16] C. Sünder, A. Zoitl, J. H. Christensen, V. Vyatkin, R. W. Brennan, A. Valentini*, et al.*, "Usability and Interoperability of IEC 61499 based distributed automation systems," in *4th IEEE International Conference on Industrial Informatics (INDIN 2006)*, Singapore, 2006, pp. 31-37.

[17] P. Tait, "A path to industrial adoption of distributed control technology," in *3rd IEEE International Conference on Industrial Informatics (INDIN 2005)*, Perth, Australia, 2005, pp. 86-91.

[18] J. L. M. Lastra, A. Lobov, and L. Godinho, "Closed loop control using an IEC 61499 application generator for scan-based controllers," in *10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2005)*, Catania, Italy, 2005, pp. 323-330.

[19] The University of Auckland. (2014). *FBench - Open Source Function Block Engineering Tool* [Online]. Available: http://oooneida-fbench.sourceforge.net

[20] K. Thramboulidis and C. Tranoris, "Developing a CASE Tool for Distributed Control Applications," *The International Journal of Advanced Manufacturing Technology,* vol. 24, pp. 24-31, July 2004 2004.

[21] C. Schwab, M. Tangermann, and A. Lueder, "The modular TORERO IEC 61499 engineering platform - Eclipse in automation," in *10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2005)*, Catania, Italy, 2005, pp. 8 pp.-272.

[22] T. Strasser, M. Rooker, G. Ebenhofer, A. Zoitl, C. Sunder, A. Valentini*, et al.*, "Framework for Distributed Industrial Automation and Control (4DIAC)," in *6th IEEE International Conference on Industrial Informatics (INDIN 2008)*, Daejeon, Korea, 2008, pp. 283-288.

[23] 4DIAC. (2014). *Framework for Distributed Industrial Automation (4DIAC)* [Online]. Available: http://www.fordiac.org

[24] Eclipse Platform. (2014). [Online]. Available: http://www.eclipse.org

[25] ICS Triplex ISaGRAF. *ISaGRAF Workbench* [Online]. Available: http://www.isagraf.com

[26] nxtControl. (2014). *nxtSTUDIO - Engineering software for all tasks* [Online]. Available: http://www.nxtcontrol.com/en/products/nxtstudio.html

[27] "Function blocks - Part 1: Architecture," *IEC 61499-1: 2005,* p. 111, 2005.

[28] V. Dubinin and V. Vyatkin, "Towards a Formal Semantic Model of IEC 61499 Function Blocks," in *4th IEEE Conference on Industrial Informatics (INDIN 2006)*, Singapore, 2006, pp. 6-11.

[29] V. Vyatkin, "The IEC 61499 standard and its semantics," *IEEE Industrial Electronics Magazine,* vol. 3, pp. 40-48, 2009.

[30] V. Vyatkin and J. Chouinard, "On comparisons of the ISaGRAF implementation of IEC 61499 with FBDK and other implementations," in *6th IEEE International Conference on Industrial Informatics (INDIN 2008)*, Daejeon, Korea, 2008, pp. 289-294.

[31] V. Dubinin and V. Vyatkin, "Semantics-robust Design Patterns for IEC 61499," *IEEE Transactions on Industrial Informatics,* vol. 8, pp. 279 - 290, 2012.

[32] W. Rumpl, F. Auinger, C. Dutzler, and A. Zoitl, "Platforms for Scalable Flexible Automation Considering the Concepts of IEC 61499," in *Knowledge and Technology Integration in Production and Services*. vol. 101, V. Mařík, L. Camarinha-Matos, and H. Afsarmanesh, Eds., ed: Springer US, 2002, pp. 237-246.