IEC 61499 Runtime Environments: A State of the Art Comparison

Laurin Prenzel[1], Alois Zoitl[2], and Julien Provost[1]

[1] Technical University Munich, Munich, Germany
{laurin.prenzel,julien.provost}@tum.de
[2] Johannes Kepler University, Linz, Austria
alois.zoitl@jku.at

Abstract. Networked automation devices, as needed for Industry 4.0 or Cyber Physical Production Systems, demand for new programming languages like the one defined in the IEC 61499 standard. IEC 61499 was originally released in 2005. Since then, different runtime environments—academic and commercial—surfaced: They partly differ in their execution semantics and behavior, and in the features they offer, e.g. Multitasking, Real-time performance, or Dynamic Reconfiguration. Users who want to apply this standard to their problem have to choose the right tool. This paper compares a selection of IEC 61499 runtime environments and outlines topics for further research.

Keywords: Archimedes · FBBeam · FBDK · 4diac FORTE · Fuber · ICARU_FB · ISaGRAF · nxtControl nxtIECRT · RTFM-RT

1 Introduction

We see a change in production automation towards more networked control devices demanding for new paradigms and languages allowing to more effectively and efficiently program them. The IEC 61499 defines a modeling language fulfilling these requirements [8]. Currently several runtime environments (RTEs) and IDEs provide implementations for IEC 61499.

Software tools for the IEC 61499 have been summarized before [5, 20]. Some new developments in the area of IEC 61499 RTEs make it necessary to take a closer look at the available systems. More specifically, this paper takes a closer look at the execution semantics and the prominent features of currently available IEC 61499 RTEs. Whether one wants to try out the standard or implement a new RTE, it is important to recognize the differences of already existing implementations. While the differences of IDEs only affect the user experience during modeling, the RTE has to interpret the execution semantics of the standard.

This paper reviews the differences between a collection of existing RTEs and tries to find unclaimed research opportunities. After the basics of the IEC 61499 standard and its execution models are presented, the examined RTEs are introduced and compared. The findings are discussed in Section 4 and research opportunities are summarized in Section 5.

2 Background

The IEC 61499 standard has been the topic of many research papers. It was developed as an architecture for distributed, flexible systems that may be reconfigured dynamically [19].

The Function Block (FB) is the main component of the IEC 61499, encapsulating the functionality. It is used in FB networks to build applications. There are different types of Function Blocks, e.g. the *Basic FB* with a state machine and algorithms, or the *Composite FB* containing a network of other FBs.

Since the introduction of the IEC 61499 standard, there has been a discussion about its execution semantics and possible ambiguities [15]. Most notably, [7] classified different execution semantics on a theoretical level. Thus, for researchers and commercial users of the standard, it is important to know the available execution semantics and the most prevalent solutions. The different runtime environments (RTEs) may be compared on different levels.

There are *organizational characteristics*, such as the license of the project (open source or other), the status (commercial, research, or inactive), or the programming language employed. The *execution semantics* may be described by the trigger mechanism (cyclic or event-based) and the execution model of the RTE. Finally, runtime environments may be distinguished by the *features* they offer, such as real-time performance, multitasking, or dynamic reconfiguration.

2.1 Execution Models

The IEC 61499 does not strictly define the execution semantics of its models. This has led to a number of papers outlining these ambiguities [4, 3, 15]. Currently, there is no consistent framework to describe the execution semantics of an IEC 61499 implementation. Two different views are discussed here. Ferrarini and Veber [7] use the factors *Multitasking* and *Scan order* to describe 7 groups of possible implementation approaches (see Table 1). The first factor is whether the order in which FBs are scanned is fixed or not fixed. The second factor is whether multitasking is used, and if yes, how it is controlled. This leads to a total of 8 combinations, but Ferrarini and Veber exclude the case of a fixed scan order and not controlled multitasking.

		Multitasking Implementation			
		Not used	Used not controlled	Used controlled, time slice	Used controlled, FB slice
Scan Order	Not fixed	A0	A1	A2	A3
	Fixed	A4	x	A5	A6

Table 1. Possible implementation approaches according to Ferrarini and Veber [7]

	Organizational			Execution			Features		
Name	**Open / Closed src**	**Research / Commercial**	**Language**	**Cyclic / Event**	**Execution Model**	**Ferrarini Model**	**Real-Time**	**Multi-Tasking**	**Recon-figuration**
Archimedes		R	Java C++	E	NPMTR	A1	Hard	Yes	Yes
FBBeam	O	R	Erlang	E	PMTR	A2	Soft	Yes	Yes
FBDK	C	R	Java	E	NPMTR	A1		Partly	Partly
4diac FORTE	O	R	C++	E	PMTR	A1	Hard	Yes	Yes
Fuber	O	R	Java	E	BSEM	A0		Yes	Yes
ICARU_FB	O	R	C	C	CBEM	A4	Hard	No	Yes
ISaGRAF	C	C	IEC 61131-3	C	CBEM	A4	Hard		
nxtIECRT	C	C	C++	E	PMTR	A1	Hard	Yes	Yes
RTFM-RT		R	C	E	PMTR	A1	Hard	Yes	

Table 2. Comparing key characteristics of IEC 61499 RTEs

In addition to this classification, many publications have introduced their own names for the most common implementation. The earliest model is arguably NPMTR (*Non-Preemptive Multithreaded Resource*), which is employed in FBDK, and mentioned already in 2006 [16]. At a similar time, a sequential model was discussed in [21] and [4]. This model was later termed *Buffered Sequential Execution Model* (BSEM) [2]. Finally, [3] termed a third model, named *Cyclic Buffered Execution Model* (CBEM).

3 Methods

As introduced in the previous section, the IEC 61499 does not strictly define its execution semantics and thus different implementations are possible. This section presents a collection of runtime environments that have been implemented since the inception of the standard. An overview of the comparison is displayed in Table 2. A total of 9 different RTEs were compared based on information that was available from websites and publications.

In addition to the three execution models already introduced in the literature, an additional model (*PMTR*) was added. *NPMTR* describes non-preemptible multitasking resources. This explicitly excludes the preemptible multitasking resources, that nevertheless do not fall into the categories of buffered sequential

or cyclic execution semantics. Thus, the *PMTR* name was chosen, to indicate the set of preemptible multitasking resources.

The assignment and collection was performed to the best of our knowledge. Where no reliable data was found, and the clues were inconclusive, the field was left blank. Following, the 9 RTEs are shortly presented.

3.1 Archimedes

There are three different runtime environments using similar execution semantics: *RTSJ-AXE*[17], *RTAI-AXE*[6], and *Luciol-AXE*[18]. They are implemented in Java and C++, and allow both reconfiguration and multitasking. FBs may be implemented as independent tasks / threads, or combined in Function Block Containers.

3.2 FBBeam

In this Erlang-based IEC 61499 runtime environment, every FB is implemented as its own process, and scheduling is left to the Erlang Virtual Machine. Erlang processes do not share memory, and messages between processes are sent asynchronously. Because of the fair round-robin scheduling, only soft real-time performance can be guaranteed. Erlang includes sophisticated frameworks for distribution, dynamic reconfiguration, debugging and monitoring of distributed, highly concurrent systems [14]. Its execution model may be best described by *PMTR*, since FBs may be preempted.

3.3 FBDK FBRT

The *FBDK* (Function Block Development Kit) and the accompanying *FBRT* (Function Block Runtime Environment) allow the modeling and execution of IEC 61499 systems in a Java-based runtime environment [9]. Function Blocks are compiled to Java classes and scheduled in a depth-first manner. Instead of emitting events, the *FBRT* uses method calls to communicate between Function Blocks [20]. The execution model of the *FBRT* was referred to as Non-Preemptive Multithreaded Resource (NPMTR) [16].

3.4 4diac FORTE

4diac FORTE is the runtime environment provided by the Eclipse 4diac open source project [1, 22]. The implementation is based on C++ and uses the Event Chain concept described in [23] to achieve deterministic real-time performance by allowing the introduction of real-time constraints for Event Chains. Execution of an Event Chain may preempt execution of other event chains, thus the execution model *PMTR* seems the most appropriate.

3.5 Fuber

Fuber was build to investigate the different execution semantics of the IEC 61499 [4]. It executes in two threads: One for the execution of the ECC, and one for the scheduling of algorithms. Function Blocks and algorithms are assigned to FIFO queues and algorithms are interpreted on the fly instead of static compilation, thus allowing modification of the algorithm code during the execution. As the focus of this implementation is research about the execution semantics, real-time performance is not considered. *Fuber* employs the Buffered Sequential Execution Model (BSEM), where FBs are put in a FIFO ready queue [2].

3.6 ICARU_FB

ICARU_FB is a RTE for lightweight embedded systems, e.g. 8-bit Arduino boards. The IEC 61499 model is converted into C code. FBs are implemented as objects and events are passed directly to a variable in the destination FB object [13]. Since the execution is cyclic, and the FB are scanned in a fixed order, the msot appropriate execution model for this RTE is CBEM and A4. Hard real-time performance may be achieved and dynamic reconfiguration is available.

3.7 ISaGRAF

ISaGRAF was the first commercial IEC 61499 implementation [5]. IEC 61499 Function Blocks are compiled to IEC 61131-3 code that may be executed on traditional IEC 61131-3 devices. Because of the IEC 61131-3 base, the execution is cyclic instead of event-triggered. Its execution model is referred to as Cyclic-Buffered Execution Model (CBEM) [3].

3.8 nxtControl nxtIECRT

According to [5], the solution provided by *nxtControl*, *nxtIECRT*, is based on the open source RTE *4diac FORTE*. Thus, the execution semantics should mostly be identical. The *nxtIECRT* RTE is a hybrid runtime system, that may execute both IEC 61131-3 and IEC 61499 systems [12]. Furthermore, *nxtIECRT* provides extensive features for changing control applications during system operation.

3.9 RTFM-RT

RTFM-RT is a RTE for the IEC 61499 built on the RTFM core language [10]. It is using the Event Chain concept and implements them as synchronous task chains [11]. The RTE is mostly build for real-time research. Threads of execution are preemptible and multitasking is possible, thus the model *PMTR* was assigned.

4 Discussion

Table 2 summarizes the findings of this paper. Up until now, the IEC 61499 has been implemented numerous times with various execution semantics. Most RTEs are open source and research projects, but there are at least 2 commercially available IEC 61499 RTEs. Both of them do not only implement the IEC 61499, but support also the languages of the IEC 61131-3. The implementation languages vary, but are mostly focused on Java and C / C++.

All RTEs except for two employ an event-triggered execution. Using the classification introduced by Ferrarini and Veber [7], most RTEs employ the semantics A0, A1, or A2, where no fixed scan order exists. *ISaGRAF* and *ICARU_FB* are the only implementations with a fixed scan order, falling into the category A4. To the knowledge of the authors, the categories A3, A5, and A6 are currently not used, i.e. there are no RTEs with a fixed scan order and multitasking, or RTEs using FB slice multitasking. For categories A5 and A6 this may be because a fixed scan order with multitasking can be contradictory, since a multitasking implementation by itself may disturb a fixed scan order. If the next FB in the fixed scan order must wait for the previous FB to finish, multitasking is not possible. If it does not have to wait for the previous FB to finish, this would disturb the determinism of a fixed scan order implementation, since the previous FB might want to send events to the next FB in the scan order. For A2 and A3, only one implementation currently exists, that uses a fair scheduler with time slice preemption. Most other implementations do not prescribe the scan order, and either do not use multitasking, or do not control it.

Since the standard is aimed at industrial process measurement and control systems, most implementations claim to offer hard real-time performance. Multitasking is available in some RTEs but not all. Although Dynamic Reconfiguration has been the topic of multiple research papers, and many RTEs seem to support it, information about the usability or performance of the reconfiguration process is rare.

5 Conclusion

This paper summarizes some developments with respect to runtime environments of the IEC 61499 for users and researchers alike interested in working with IEC 61499 or wanting to implement their own RTE. Since the introduction of the standard, it has been implemented numerous times. Despite the ambiguities of the execution semantics, there exist both commercial and research runtime environments that may be used to control physical systems.

From a theoretic perspective, the existing and possible execution models call for a deeper investigation. The current classification frameworks help distinguish fundamental differences between the RTEs, but fail to describe the different execution models of the standard precisely. Given that the execution semantics of the standard have room for interpretation, it is even more important to differentiate between the implementations.

Given the availability of lightweight, multitasking embedded systems that require real-time performance, the IEC 61499 may offer suitable models for this application. In this regard, deterministic real-time scheduling of multitasking IEC 61499 systems may require further investigation.

Although the topic of Dynamic Reconfiguration has been addressed from a modeling perspective, and many runtime environments claim to allow Dynamic Reconfiguration, examples of Dynamic Reconfiguration with the IEC 61499 are rare. Most RTEs focus on the execution semantics, whereas the frameworks for deployment, distribution, configuration and reconfiguration are also key selling points of the IEC 61499.

References

1. 4diac: 4diac FORTE - the 4diac runtime environment. `https://www.eclipse.org/4diac/en_rte.php` (2019), accessed: 2019-5-24
2. Cengic, G., Akesson, K.: Definition of the execution model used in the fuber IEC 61499 runtime environment. In: International Conference on Industrial Informatics. IEEE (2008)
3. Cengic, G., Akesson, K.: On formal analysis of IEC 61499 applications, part b: Execution semantics. IEEE Transactions on Industrial Informatics (2010)
4. Cengic, G., Ljungkrantz, O., Akesson, K.: Formal modeling of function block applications running in IEC 61499 execution runtime. In: Conference on Emerging Technologies and Factory Automation. IEEE (2006)
5. Christensen, J.H., Strasser, T., Valentini, A., Vyatkin, V., Zoitl, A., Chouinard, J., Mayer, H., Kopitar, A.: The IEC 61499 function block standard: Software tools and runtime platforms. ISA Automation Week (2012)
6. Doukas, G.S., Thramboulidis, K.C.: A real-time linux execution environment for function-block based distributed control applications. In: International Conference on Industrial Informatics. IEEE (2005)
7. Ferrarini, L., Veber, C.: Implementation approaches for the execution model of IEC 61499 applications. In: International Conference on Industrial Informatics. IEEE (2004)
8. Harrison, R., Vera, D., Ahmad, B.: Engineering methods and tools for Cyber–Physical automation systems. Proceedings of the IEEE **104**(5), 973–985 (2016)
9. holobloc: FBDK 8.0 - the function block development kit. `https://www.holobloc.com/fbdk8/index.htm`, accessed: 2019-5-24
10. Lindgren, P., Lindner, M., Lindner, A., Pereira, D., Pinho, L.M.: RTFM-core: Language and implementation. In: Conference on Industrial Electronics and Applications. IEEE (2015)
11. Lindgren, P., Lindner, M., Lindner, A., Vyatkin, V., Pereira, D., Pinho, L.M.: A real-time semantics for the IEC 61499 standard. In: Conference on Emerging Technologies Factory Automation. IEEE (2015)
12. nxtcontrol: nxtcontrol - nxtIECRT. `https://www.nxtcontrol.com/en/control/` (2019), accessed: 2019-5-24
13. Pinto, L.I., Vasconcellos, C.D., Rosso, R.S.U., Negri, G.H.: ICARU-FB: An IEC 61499 compliant multiplatform software infrastructure. IEEE Transactions on Industrial Informatics **12**(3), 1074–1083 (2016)
14. Prenzel, L., Provost, J.: FBBeam: An erlang-based IEC 61499 implementation. In: International Conference on Industrial Informatics. IEEE (2019)

15. Strasser, T., Zoitl, A., Christensen, J.H., Sünder, C.: Design and execution issues in IEC 61499 distributed automation and control systems. IEEE Transactions on Systems, Man and Cybernetics **41**(1), 41–51 (2011)
16. Sünder, C., Zoitl, A., Christensen, J.H., Vyatkin, V., Brennan, R.W., Valentini, A., Ferrarini, L., Strasser, T., Martinez-lastra, J.L., Auinger, F.: Usability and interoperability of IEC 61499 based distributed automation systems. In: International Conference on Industrial Informatics. IEEE (2006)
17. Thramboulidis, K., Zoupas, A.: Real-time java in control and automation: a model driven development approach. In: Conference on Emerging Technologies and Factory Automation. vol. 1. IEEE (2005)
18. Thramboulidis, K., Papakonstantinou, N.: An IEC 61499 execution environment for an aJile-based field device. In: Conference on Emerging Technologies and Factory Automation. IEEE (2006)
19. Vyatkin, V.: IEC 61499 as enabler of distributed and intelligent automation: State-of-the-Art review. IEEE Transactions on Industrial Informatics **7**(4) (2011)
20. Vyatkin, V., Chouinard, J.: On comparisons of the ISaGRAF implementation of IEC 61499 with FBDK and other implementations. In: International Conference on Industrial Informatics. IEEE (2008)
21. Zoitl, A., Grabmair, G., Auinger, F., Sunder, C.: Executing real-time constrained control applications modelled in IEC 61499 with respect to dynamic reconfiguration. In: International Conference on Industrial Informatics. IEEE (2005)
22. Zoitl, A., Strasser, T., Valentini, A.: Open source initiatives as basis for the establishment of new technologies in industrial automation: 4DIAC a case study. In: International Symposium on Industrial Electronics. IEEE (2010)
23. Zoitl, A.: Real-time Execution for IEC 61499. Instrumentation, Systems, and Automation Society (2009)