Ensuring Real-Time Correctness for IEC 61499 Designs

PHD FORUM SUBMISSION

Sven Mehlhop
Carl von Ossietzky University of Oldenburg - sven.mehlhop@uol.de

***Abstract*—The increasing complexity of industrial control systems requires new modelling approaches such as IEC 61499, but verification tools for this approach are still lacking. Among other things, a tool for verifying temporal properties at design time is missing. One approach to fill this gap is to extend the standard to include temporal contracts and to design a simulator as an implementation of the standard, enriched with a time model and the possibility to implement monitors for the contracts and thus enable virtual integration testing.**

I. Introduction

Industry 4.0 is verging into establishing higher use of distributed systems. This increases the complexity and therefore the requirements to model a system in the industry 4.0 context. Since these systems are usually safety-critical systems, their behavior over time is also important. In this context, early verification is also of immense importance.

Ensuring such temporal requirements for complex industrial control systems is difficult to achieve by formal verification.

One approach to model is the IEC 61499 standard, a successor of the widely-used IEC 61131 standard. This standard provides a model-based approach for distributed systems. It is based on an event-based model of execution using graphical function block networks. While currently gaining succesessively more importance in industry and in academia, there are still tools missing that help ensuring functional and extrafunctional correctness in real-world systems. There are different attempts [1] for formal verification for models of the IEC 61499 standard, but they are not feasible for systems of this type. Therefore, different concepts are needed. One approach is layed out here, to ensuring timing correctness for embedded systems at design time.

This paper first presents the research questions for the planned dissertation before proposing the first approaches.

II. Research question

Fig. 1 shows a visualisation of the research questions and their relationship. The remainder of this section explains these in detail. The figure is intented to get interpreted as an refinement process of research questions.

RQ 1: **How to ensure and evaluate ICS timing correctness at design time?** This question splits into two sub-aspects. What is an appropriate modeling language for ICS and how can timing correctness be ensured at design time.

One approach to ensure timing correctness at design time is contract based design (CBD). CBD allows the integration of temporal requirements and assumptions already in the design process and has proven itself in the automotive design sector [2]. Contracts can be verified formal and simulative.

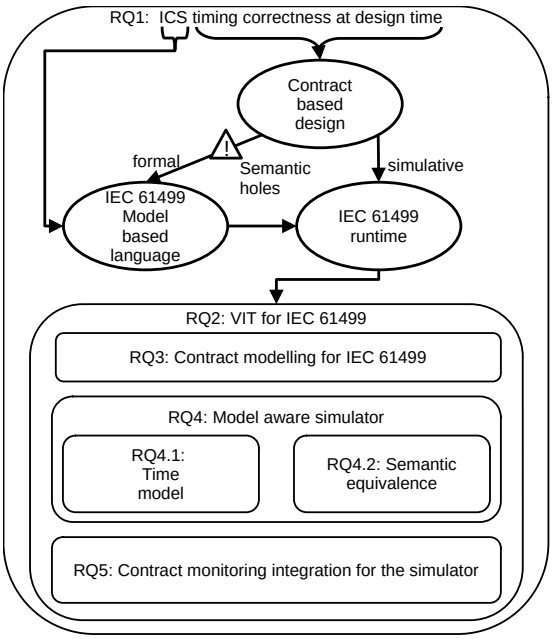

Fig. 1. The scientific research questions proposed for this work

A modeling standard for ICS is IEC 61499. The standard is suitable for the design of increasingly complex and distributed systems. Also it has proven itself in the work with CBD, as [3] and [4] prove. However, since the standard has semantic gaps [5], contracts cannot be verified in a formal way. Therefore, the decision is made in favor of the simulative approach.

In order to prevent the evaluation of the contracts from taking place until after the integration on the target platform, which would increase the costs for an already increasing effort in the design of ICS, it is necessary to have a possibility to check the contracts already at design time in order to detect errors as early as possible. VIT offers a solution for this, as works like [2] prove.

RQ 2: **How to enable VIT for IEC 61499?** It takes three pieces to enable VIT for IEC 61499. It needs a contract extension for IEC 61499, to integrate the contract modelling in the design process. The required IEC 61499 model-aware simulator has to be capable of executing given models and integrating contract monitoring.

RQ 3: **How to embed contract modelling in IEC 61499?** To ensure contract modelling for IEC 61499, the description language MULTIC Timing Specification Language (MTSL) [2] is used. The contracts can describe individual and coherent parts of an IEC 61499 model on different abstraction layers. This starts at the overall system level and can extend to the lower levels of the function blocks or the communication between the blocks. The contracts can provide these parts with individual assumptions and requirements.

RQ 4: **How to design a model aware simulator for VIT?** Given that there are numerous techniques of simulating IEC 61499 models, the question of why create a new simulator arises. The reason for this is the lack of a sufficient toolset for modeling and analyzing timing behavior, as well as the ability to not only simulate the model's behavior, but to actually execute it like a typical implementation. A simulator that can enable VIT with timing contracts for IEC 61499 models must therefore consider multiple factors.

First, it must be ensured that the simulator's behavior corresponds to the standard. However, because the standard does not completely define several semantic criteria, the behavior of an IEC 61499 runtime implementation must be mimicked as well. Because the simulator's objective is to verify temporal behavior, it necessitates the second feature of a time model, which is as realistic a representation of an IEC 61499 implementation as possible. The modeling language SystemC supports time model integration and includes required features such as event tracing. As a result, the usage of SystemC for the simulator is intended. This has also worked for a first proof-of-concept, as explained in Method. Also, the inputs of the simulator should be stimulated with input sets from scenario close simulations or real world data sets.

RQ 4.1: **How to expand the simulator for VIT with a suitable time model?** Building a simulation to examine and verify temporal properties requires a time model. This time model should include the execution times of function blocks and the latencies due to communication and scheduling between function blocks on one or more hardware platforms. In this way, the runtime of an entire application can be predicted from the composition of time values recorded once.

RQ 4.2: **How to ensure semantic equivalence between simulator and an IEC 1499 runtime?** The necessity of semantic equivalence stems from the fact that the simulation should be as near to an IEC 61499 implementation as possible. As stated in [5], the standard has a number of unknown or at least diverse interpretable semantic behaviors. Different implementations handle these open spots differently. As a result, the simulation has to focus on a specific runtime implementation, and has limited transferability.

RQ 5: **How to extend the simulator with contract monitoring?** As previously stated, there is already work that combines IEC 61499 and contract-based design. MTSL is used there as the contract language. The idea is to annotate the contracts based on the concept of RQ 3 to the model and check while simulating, if a contract is harmed. The feasibility of this has yet to be determined.

III. METHOD

The planned flow to enable timing correctness for ICS is shown in Fig. 2. Starting with a model of an IEC 61499 system enriched with the contracts concerned with different system granularities. This model will be transformed into a model, which can be executed by a simulator and the contract specifications are extracted and afterwards transformed into Contract Monitors. The transformation process is a combination of using and converting parts of existing code fragments of the 4diac Forte code base [6] and implementing key features of the IEC 61499 standard beforehand.

A proof-of-concept simulator is already developed and first tests have proven the possibility to use this simulator as planned [7]. For ensuring the semantic equivalence of the simulator, there are currently two papers to be published. One paper targets a first approach to ensure correctness. The other paper [5] lays the foundation for ensuring correctness for IEC 61499 runtimes and simulators.

The simulator, in conjunction with a timing model previously generated from a set of benchmarks on different hardware and the Contract monitors, predicts the timing behaviour of the application. The process of developing this time model is still ongoing.

For the evaluation of this approach, it is planned to build a reference example with Factory IO and to examine if this approach can enable timing correctness.

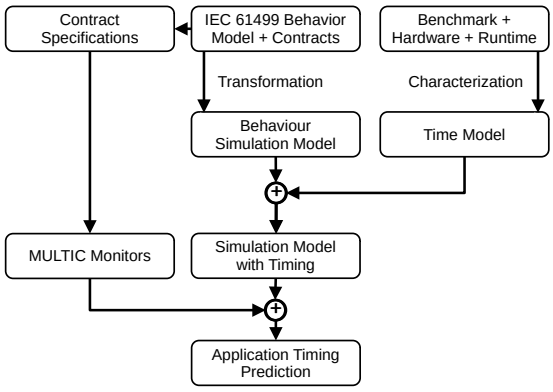

Fig. 2. The proposed flow to enable VIT for ICS

REFERENCES

[1] G. Cengic and K. Akesson, "On formal analysis of IEC 61499 applications, part A: Modeling," *IEEE Trans. on Industrial Informatics*, vol. 6, no. 2, pp. 136–144, 2010.

[2] E. Böde, M. Büker *et al.*, "Design Paradigms for Multi-Layer Time Coherency in ADAS and Automated Driving (MULTIC)," in *FAT-Schriftenreihe 302*, 10 2017.

[3] D. Do Tran, J. Walter *et al.*, "Towards time-sensitive behavioral contract monitors for IEC 61499 function blocks," in *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, vol. 1. IEEE, 2020, pp. 27–34.

[4] F. Bruns, W. Nebel *et al.*, "Modeling of real-time communication for industrial distributed automation systems," in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, 2020.

[5] Bianca Wiesmayr, Sven Mehlhop and Alois Zoitl, "Close Enough? Criteria for Sufficient Simulations of IEC 61499 Models," 2023, unpublished.

[6] Eclipse - 4Diac, "4diac FORTE – IEC 61499 Runtime Environment," https://www.eclipse.org/4diac/en_rte.php, (accessed on 07/16/2020).

[7] S. Mehlhop and J. Walter, "Model-aware Simulation of IEC 61499 Designs," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–4.