A Time Model for an Event-Queue based IEC 61499 Runtime

WORK-IN-PROGRESS SUBMISSION

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

***Abstract*—The current development in the field of industrial control systems is moving further and further in the direction of distributed and increasingly complex structures. In order to be able to model these, tools such as the IEC 61499 standard come into play. But for such tools there is a lack of possibilities to validate them, as the increasingly complex structure overtaxes formal methods. Therefore, it needs a simulator for which it needs a time model. For this time model, the measurement concept is presented and evaluated, with the conclusion that it is sufficient for a proof-of-concept.**

I. INTRODUCTION

Although industrial control systems are becoming larger, more distributed, and more complicated, they still must be validated because they are part of a safety-critical infrastructure. This can be accomplished, for example, by evaluating them with tests on the target system. Given the high cost of testing at this late stage, a solution to verification during design time is required. Formal verification using analytical methods is one option. Initial approaches exist, but they are overwhelmed by the required complexity and scale.

Therefore, a simulative approach [1] for the IEC 61499 standard is proposed. The standard is a modelling language for distributed industrial control systems. It describes a control system model using function blocks. Each function block inherits a simple state machine, a cpp service interface, or is a composite block that inherits other function blocks. The simulator is not only an implementation of the standard, but also of the common IEC 61499 runtime Forte [2] to have interpretations for undefinitions of the standard [3]. It is written in SystemC and is designed to take advantage of existing SystemC features for virtual integration testing. The overarching purpose of that method is to validate temporal behaviour using contract-based design. The advantage of the simulator is the possibility to really perform functionalities and calculations and thus to generate real execution paths with the help of input sets.

In order to be able to simulate the temporal bahaviour of the model sufficiently, a time model for the simulator is required. For this purpose, execution times of function blocks are to be measured on realistic hardware, as well as the communication between the function blocks.

The procedure, in which the time model, is described graphically in Fig. 1. The behaviour model is converted into a simulator behaviour model. It is then extended by a temporal model there. This is the result of a series of benchmark testing on several hardware platforms. Predictions concerning temporal behaviour, among other things, can be made based on the simulation results.

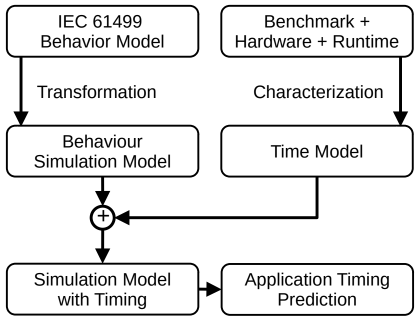

Fig. 1. The flow to enrich the runtime with a time model

This paper presents the first approaches for a measurement infrastructure to record time values on different hardware platforms. In the Method chapter, this measurement infrastructure is presented, followed by an initial evaluation. The findings are then analysed, followed by a summary of future work and current challenges.

II. METHOD

In the area of embedded systems, the measurement of individual functional sections to generate a time model for a simulation is a standard procedure. This procedure is implemented in different ways and already exists for most environments. For the IEC 61499 standard, however, there is still a need for a specific method. Therefore, a suitable measurement infrastructure will be presented here and an evaluation will be performed to check whether the measured values are meaningful enough.

A. *Measurement infrasturcture*

There are several ways to measure function blocks. For example, Forte's own tracing can be activated. This writes a trace to the command line during execution for input and output events, in which the block, the current time and whether it is an incoming or outgoing event is shown. Another method is the measurement by either an own Service Interface Function Block (SFIB) that contains a measurement infrastructure and is part of the model as an extra function block or a measurement infrastructure as part of a SIFB [4].

The disadvantage of both variants is the additional time influence of an FB as part of the complete execution, or the influence of command line outputs during runtime. In addition, the reusability of the console output is costly.

The proposed measurement infrastructure is implemented as part of the runtime. It makes use of the fact that standard functionalities that every FB must be able to do are available as a C++ class. Thus, incoming and outgoing messages are all processed in the same way before more specific functions are implemented. It is at this point that a point in time is taken for the measurement infrastructure described here. Therefore, the temporal influence also remains low. There is an instance for each FB in the model in which a previously defined number of measurements are recorded. With the help of a `Measure` FB designed for this purpose, the model is stopped after a specified number of runs and the measurements are saved. This is then added to the existing time model so that it can serve as a time reference for the simulator.

B. Evaluation concept

The overall goal of the simulator is to estimate the execution times of single function block ones and then use this information in other models. To evaluate, we created a reference sample consisting of three FBs (see fig. 2). These FBs are measured both individually and as a full chain, with each 10000 runs. The individual measurement consists of the same experimental setup, where the one block to be measured is triggered by `E_Cycle` and `Measure` terminates the experiment and saves the measurement. The time between the FBs is also recorded, as combination of communication latency and Forte event scheduling impact. The measurement time of the entire cycle is then compared to the sum of the single execution times of the FBs and the communication timings between them. The proof of concept is complete if they are near to each other.

To evaluate the impact of the measurement infrastructure, two identical models are run 10000 times each, once with and once without the measurement infrastructure. Instead of `E_CYCLE` FB, the cycle triggers itself. Following that, the execution times are compared.

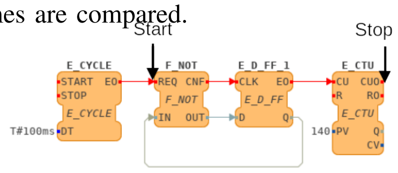

Fig. 2. The model under evaluation

Since the influence of competing tasks and services on the target platform falsifies the measurement results and makes them no longer reproducible, measures are taken to limit these influences.

A typical embedded system with the Tinkerboard is used as hardware. One of the four cores of the board is isolated so that only the Forte runtime runs on it. In addition, parallel processes are terminated if possible and the frequency is set static.

Since the first measurement results deviate observably from the rest, a start-up time is implemented in the experimental setup.

TABLE I
THE EXECUTION TIME OF THE DIFFERENT FBS AND THE THE MODEL

	F_NOT	**E_D_FF_1**	**E_CTU**	**Between two FBs**	**Full model**
Min	2.92 µs	2.33 µs	2.33 µs	1.46 µs	10.79 µs
Max	4.38 µs	3.21 µs	2.92 µs	1.75 µs	13.71 µs
Avg	3.68 µs	2.74 µs	2.58 µs	1.56 µs	11.77 µs
std. Dev.	0.517	0.254	0.192	0.138	0.741

III. EVALUATION AND FUTURE WORK

Table I shows the execution times of the individual FBs, the elapsed time between two FBs and the complete measurement. The minima, maxima, average and standard deviation were measured in each case. If we compare the values and add the latency between two FBs twice, we have an approximation to the values of the entire model and thus a proof-of-concept.

The analysis of the impact of the measurement infrastructure showed an influence of 0.033 µs per run and is therefore sufficiently low. The static setting of the CPU frequency had a positive impact on the reproducibility of the measurement results. Although higher frequencies allow a faster execution time, the standard deviation over several measurement series is not reproducible. Presumably concurrent processes still influence the measurement series despite the precautions. Therefore, the target platform should be made more real-time capable, e.g. by using an RTOS.

For a complete investigation, the latency between two FBs should be investigated further. It is interesting to see what influence the communication and scheduling procedures of the forte have here.

Future work then consists of a closer look at ECC and SFIB FBs. In ECCs, the measurement methodology presented allows the algorithms of the individual algorithms to be measured individually to the states of the state machine. A similar principle applies to the algorithms in an SFIB. Since the simulator implements the algorithms identically, the transisitons and branches can also be simulated in the programme. Also, the impact of the halting problem on the validation of a model by the simulator should be investigated here. Another open challenge for future consideration is the possible parallelism of FBs.

As a conclusion to the overall concept, it is also necessary to examine how the time model can be inserted into the simulator and whether the results then also reflect the behaviour of the runtime.

REFERENCES

[1] 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.

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

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

[4] F. Bruns, W. Nebel, J. Walter, and K. Grüttner, “Wip: Modeling of real-time communication for industrial distributed automation systems,” in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, 2020, pp. 1–4.