

Work-in-Progress: Evaluation Framework for End-to-End Latency of Cause-Effect Chains

Mario Günzel

Max Planck Institute for Software Systems (MPI-SWS) TU Dortmund University
mguenzel@mpi-sws.org

Robin Edmaier

Jian-Jia Chen
RWTH Aachen University
jian-jia.chen@rwth-aachen.de

Abstract—To compare different analyses for the end-to-end latency of cause-effect chains, empirical evaluation using a large number of (often synthetically generated) test cases plays an important role. However, conducting such an evaluation is often difficult and time-consuming, especially if implementations are not publicly available, or if they use different code structures or programming languages. In this work-in-progress, we develop an evaluation framework that collects benchmarks for the generation of cause-effect chains and approaches to analyze their end-to-end latency. The toolbox is implemented in Python 3 and prioritizes ease of use and extensibility. There is an ongoing effort to further enrich the framework and keep it up to date.

I. INTRODUCTION

Nowadays, many embedded cyber-physical systems consist of software pipelines that span sensing, processing, and actuation. Such pipelines are usually described as sequences of tasks, also known as *cause-effect chains*, and their timing behavior is specified as *end-to-end latency*. Over the years, multiple metrics to describe the end-to-end latency have been developed, with the most prominent ones being the Maximum Reaction Time (MRT), which is the longest time interval from an external cause until the earliest time where this external cause is fully processed, and Maximum Data Age (MDA), which is the longest time interval starting from sampling a value and ending at an effect that is caused by an actuation that processes the sampled data. A multitude of analytical approaches under different communication and scheduling policies for the end-to-end latency have been developed, and we refer to the tutorial [20] for a comprehensive overview.

To compare the performance of different analyses, synthetic evaluation is often used. That means, test cases are automatically generated based on some benchmark and the result of multiple analysis methods on the generated test cases is compared. However, the comparison is often difficult to perform: not all implementations are publicly available, different programming languages are utilized to implement the analysis, and the code structure from different implementations is usually incompatible.

In this paper, we present an evaluation framework for the end-to-end latency of cause-effect chains, consisting of:

- Task set and cause-effect chain generator.
- Collection of analytical results.
- Plotting component to visualize the produced results.

The framework is implemented in Python 3 and inspired by the evaluation framework for self-suspending task sets [15, 44].

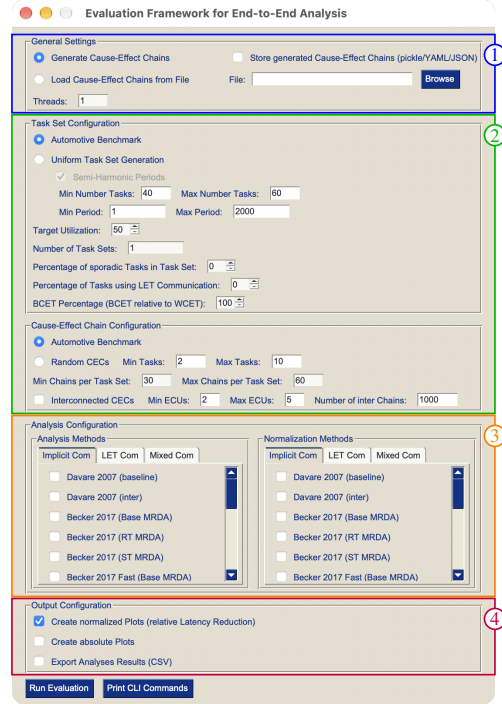


Fig. 1. Evaluation framework GUI.

It prioritizes ease of use, and supports both a Graphical User Interface (GUI) and a Command-Line Interface (CLI). Currently, the framework supports cause-effect chain generation based on an automotive benchmark [30] and a uniform benchmark [16] (based on the UUnifast [6] algorithm). The framework contains analyses from 12 papers (cf. Table I), and plots absolute values and relative latency reduction. The framework is published on GitHub¹ and there is an ongoing effort to keep the framework up to date.

II. FRAMEWORK DESCRIPTION

Fig. 1 presents the GUI of our framework, which consists of four components ①–④:

- ① General settings: Specification of the load/store behavior of the test case generator.
- ② Task set and cause-effect chain configuration: Adjustment of the benchmarks used for the test case generator.

¹<https://github.com/E2E-Evaluation-org/E2EEvaluation>

- ③ Analysis configuration: Choosing analytical approaches to be evaluated.
- ④ Output configuration: Choosing the output to be generated.

The GUI can either run the evaluation specified in ①–④, or generate a CLI command that can be used to run the evaluation process without a GUI (e.g., to automate evaluation pipelines). In the following, we detail each of the four components.

General settings: In part ①, the user selects whether test cases should be generated or loaded from a file. The framework also supports storing the generated chains as pickle, YAML and JSON files for further processing. The ‘Threads’ parameter can be increased to parallelize the test case generation and the analysis on multiple cores.

Task set and cause-effect chain configuration: The synthesized test cases (task sets and cause-effect chains) can be configured in part ②. Task sets can be generated using an automotive benchmark [30] or a uniform benchmark (described in [16]), by specifying target utilization and task set count. To evaluate analyses [18] that allow a mixture of periodic and sporadic tasks, as well as a mixture of tasks using implicit communication and communication under the Logical Execution Time (LET) [26] paradigm, the ratio of tasks following the respective release and communication policy can be set. Analyses that are only applicable to one release or communication policy automatically interpret the task specification accordingly.

Analysis configuration: In part ③, on the left-hand side, the end-to-end latency analysis approaches for the evaluation are selected. Additionally, on the right-hand side, a normalization method for the generation of relative plots can be chosen (see the paragraph on ‘Output Configuration’ for further details). Available analyses are summarized in Table I. In the overview, we distinguish different release policies (i.e., periodic and sporadic) and communication policies (i.e., implicit communication and LET). The framework covers analyses for Maximum Reaction Time (MRT), Maximum Data Age (MDA), and Maximum Reduced Data Age (MRDA), following the terminology in the tutorial on end-to-end latency [20]. Plotting symbols use the first author’s initial and publication year, with additional characters added to avoid duplicates.

Output configuration: The output plots, generated in PDF (using matplotlib) and LaTeX (using tikz and pgfplots packages) formats, are specified in part ④. Exemplary plots for a comparison of some analyses for the MRT with the automotive benchmark are shown in Fig. 2. Besides the plotting of absolute values $latency_{absolute}$ returned by the analysis under consideration, relative latency reduction

$$latency_{reduction} = \frac{latency_{base} - latency_{absolute}}{latency_{base}} \quad (1)$$

can be plotted, using another method to calculate $latency_{base}$ for normalization (D07 in Fig. 2). The boxplots show median (red), quartiles (blue), and the range of minimum/maximum values (black) among all cause-effect chains. Additionally, analysis results can be chosen to be stored as CSV.

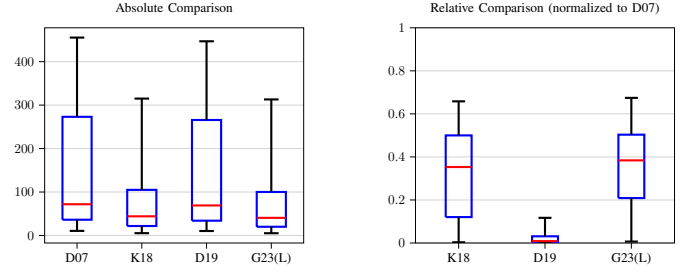


Fig. 2. Absolute and relative comparisons (showing absolute values and latency reduction from (1), respectively) of analytical end-to-end latency bounds generated by the framework.

III. EXTENDING THE FRAMEWORK

The evaluation framework is designed to be easily extensible, and this section details the steps necessary to integrate a novel analysis into the framework.

Step 1: Providing the analysis. First, a new file for the analysis method has to be created in the folder `e2eAnalyses`, which collects all analytical results. The analysis can be split into multiple functions within the file. Some useful methods (such as an event simulator) are placed in the `utilities` folder. There should be one method that is directly called by the framework for the analysis, taking one cause-effect chain as an input argument (as specified in `cechains/chain.py`). In case the analysis is able to analyze interconnected cause-effect chains [16], the method should take a variable number of local cause-effect chains as its input. The method called by the framework must return a single value, which is interpreted as the result of the analysis approach by the framework.

Step 2: Integrating the analysis into the framework. To integrate the analysis into the main file (to be accessible via the GUI and CLI), the framework provides an object, called `AnalysisMethod`, that needs to be added to the analysis dictionary `analysesDict` in the `framework.py` file. The dictionary key that is used to add the analysis to `analysesDict` is used internally by the framework and when calling it from the CLI. The `AnalysisMethod` specifies the analysis function and the names that are used in the GUI and in the output plots. Furthermore, a list of feature values describing the analysis method needs to be specified to categorize the analysis in the GUI. Currently, the framework uses features like `periodic` or `implicit` to categorize the analyses with respect to the assumed release and communication policies. After integration, the analysis appears in component ③ of the GUI, sorted into the correct tab.

Step 3: Documentation. Since the framework strives for ease of use and clarity, new analyses should be properly documented. That includes proper code documentation as well as proper documentation in `README.md`. To that end, the analysis-related publication should be listed in the overview table, supported by a categorization of the analysis, as well as links to the publication and (potential) template repositories.

TABLE I
OVERVIEW OF INTEGRATED END-TO-END ANALYSES.

Year	Paper	Release	Communication	Metric	Symbol
2007	Davare et al. [8]	Periodic, Sporadic*	Implicit	MRT	D07
2016/17	Becker et al. [2, 3]	Periodic	Implicit	MRDA	B17
2017	Hamann et al. [22]	Periodic, Sporadic	LET	MDA/MRT	H17
2018	Kloda et al. [27]	Periodic	Implicit	MDA/MRT	K18
2019	Dürr et al. [9]	Periodic, Sporadic	Implicit	MRDA, MDA/MRT	D19
2021/23	Günzel et al. [14, 16]	Periodic	Implicit, LET	MRDA, MDA/MRT	G23 (L/I)
2022	Bi et al. [5]	Periodic	Implicit	MRDA	B22
2022	Gohari et al. [12]	Periodic	Implicit	MRDA	G22
2023	Günzel et al. [17]	Periodic	LET	MDA/MRT	G23 (EQ)
2023	Günzel et al. [18]	Periodic, Sporadic	Implicit, LET	MRT	G23 (MIX)

* Although proposed for periodic tasks, the results are also applicable to sporadic tasks.

IV. FUTURE WORK

While this work introduces a framework infrastructure for the evaluation of end-to-end latency of cause-effect chains with an initial set of benchmarks and analyses, enriching the framework further is part of future work. Specifically, there are multiple analyses [13, 19, 21, 28, 29, 31–35, 37, 39–43, 45, 46] that we plan to integrate. The framework provides the infrastructure to achieve that, and the procedure for integrating novel analyses is detailed in Section III.

Besides the collection of analytical approaches, the framework collects two benchmarks for test case generation, and the integration of further generators (such as [7]) is part of future work. While most of the analyses to be added should directly be compatible with the cause-effect chains currently generated by the framework, some analyses allow additional properties like event-triggered activation [41–43] or probabilistic behavior [13, 19, 31, 40], and require an extension of the current cause-effect chain generation procedure to be properly integrated. In addition to synthetic test case generation, we also strive to collect case studies and real-world application examples [1, 4, 10, 11, 23–25, 36, 38] into the framework.

V. CONCLUSION

This work-in-progress develops an evaluation framework for the end-to-end latency of cause-effect chains. The framework currently supports cause-effect chain generation based on two benchmarks, and includes analyses from 12 papers. In future work, we plan to enrich the framework with further benchmarks and analyses. Since analyses for the end-to-end latency of cause-effect chains are actively being developed, keeping the toolbox up to date will be a continuous effort.

We envision the further development of this framework to be a community-driven effort. With its focus on simplicity and extensibility, the framework is designed to ease the integration of new benchmarks and analyses. We invite researchers interested in integrating their own benchmark or analysis, using the framework for an evaluation, or otherwise contributing to its development to reach out to us.

VI. ACKNOWLEDGMENT

We would like to acknowledge the support from everyone involved in creating and extending this framework. Particularly, we would like to thank Matthias Becker from KTH Royal Institute of Technology, Prof. Jiankang Ren from Dalian University of Technology, Pourya Gohari from Eindhoven University of Technology, and Paolo Pazzaglia from Robert Bosch GmbH (formerly researcher at Saarland University) for contributing source code, guidance, and further information about their analyses.

This result is part of a project (PropRT) that has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 865170). This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project No. 569077889 (PEACH).

REFERENCES

- [1] M. Becker and D. Casini, “The material framework: Modeling and automatic code generation of edge real-time applications under the qnx rtos,” *Journal of Systems Architecture*, vol. 154, 2024.
- [2] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte, “Synthesizing job-level dependencies for automotive multi-rate effect chains,” in *RTCSA*. IEEE Computer Society, 2016.
- [3] —, “End-to-end timing analysis of cause-effect chains in automotive embedded systems,” *J. Syst. Archit.*, vol. 80, 2017.
- [4] D. Bellassai, C. Scordino, D. Casini, and A. Biondi, “AP-LET: Enabling deterministic pub/sub communication in autosar adaptive,” *Journal of Systems Architecture*, vol. 162, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1383762125000621>
- [5] R. Bi, X. Liu, J. Ren, P. Wang, H. Lv, and G. Tan, “Efficient maximum data age analysis for cause-effect chains in automotive systems,” in *DAC*. ACM, 2022.
- [6] E. Bini and G. C. Buttazzo, “Measuring the performance of schedulability tests,” *Real Time Syst.*, vol. 30, no. 1-2, 2005.
- [7] S. S. Craciunas, M. Becker, and P. Pop, “Automotive cause-effect chains in the wild: A real-world industrial benchmark and generator,” 2026, presented at the 15th International Real-Time Systems Open Problems Seminar (RTSOPS 2026). [Online]. Available: <https://doi.org/10.5281/zenodo.20637979>

- [8] A. Davare, Q. Zhu, M. D. Natale, C. Pinello, S. Kanajan, and A. L. Sangiovanni-Vincentelli, "Period optimization for hard real-time distributed automotive systems," in *DAC*. IEEE, 2007.
- [9] M. Dürr, G. von der Brüggen, K. Chen, and J. Chen, "End-to-end timing analysis of sporadic cause-effect chains in distributed systems," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5s, 2019.
- [10] P. Frey, "Ulmer Informatik Berichte Nr 2010-03 - Case Study: Engine Control Application," University Ulm, Tech. Rep., 2010.
- [11] K.-B. Gemlau, L. KÖHLER, R. Ernst, and S. Quinton, "System-level logical execution time: Augmenting the logical execution time paradigm for distributed real-time automotive software," *ACM Trans. Cyber-Phys. Syst.*, vol. 5, no. 2, 2021. [Online]. Available: <https://doi.org/10.1145/3381847>
- [12] P. Gohari, M. Nasri, and J. Voeten, "Data-age analysis for multi-rate task chains under timing uncertainty," in *RTNS*. ACM, 2022.
- [13] M. Günzel, "Property-based timing analysis of distributed real-time systems," Ph.D. dissertation, Technical University of Dortmund, Germany, 2024.
- [14] M. Günzel, K. Chen, N. Ueter, G. von der Brüggen, M. Dürr, and J. Chen, "Timing analysis of asynchronized distributed cause-effect chains," in *RTAS*. IEEE, 2021.
- [15] M. Günzel, H. Teper, K. Chen, G. von der Brüggen, and J. Chen, "Work-in-progress: Evaluation framework for self-suspending schedulability tests," in *RTSS*. IEEE, 2021.
- [16] M. Günzel, K. Chen, N. Ueter, G. von der Brüggen, M. Dürr, and J. Chen, "Compositional timing analysis of asynchronized distributed cause-effect chains," *ACM Trans. Embed. Comput. Syst.*, vol. 22, no. 4, 2023.
- [17] M. Günzel, H. Teper, K. Chen, G. von der Brüggen, and J. Chen, "On the equivalence of maximum reaction time and maximum data age for cause-effect chains," in *ECRTS*, ser. LIPIcs, vol. 262. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.
- [18] M. Günzel, N. Ueter, K. Chen, and J. Chen, "Timing analysis of cause-effect chains with heterogeneous communication mechanisms," in *RTNS*. ACM, 2023.
- [19] M. Günzel, N. Ueter, K. Chen, G. von der Brüggen, and J. Chen, "Probabilistic reaction time analysis," *ACM Trans. Embed. Comput. Syst.*, vol. 22, no. 5s, 2023.
- [20] M. Günzel, H. Teper, G. von der Brüggen, and J. Chen, "End-to-end latency of cause-effect chains: A tutorial," *ACM Trans. Embed. Comput. Syst.*, vol. 24, no. 1, 2025.
- [21] M. Günzel, M. Becker, and D. Casini, "Shape-aware analysis of end-to-end latency under LET," in *RTAS*. IEEE, 2026.
- [22] A. Hamann, D. Dasari, S. Kramer, M. Pressler, and F. Wurst, "Communication centric design in complex automotive embedded systems," in *ECRTS*, ser. LIPIcs, vol. 76. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
- [23] A. Hamann, D. Dasari, S. Kramer, M. Pressler, F. Wurst, and D. Ziegenbein, "Waters industrial challenge 2017," in *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2017.
- [24] A. Hamann, D. Dasari, F. Wurst, I. Sañudo, N. Capodieci, P. Burgio, and M. Bertogna, "Waters industrial challenge 2019," in *Proceedings of the 10th International Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS)*, 2019.
- [25] P. Iyengar, L. Huning, and E. Pulvermueller, "Automated end-to-end timing analysis of autosar-based causal event chains," in *ENASE*, 2020.
- [26] C. M. Kirsch and A. Sokolova, "The logical execution time paradigm," in *Advances in Real-Time Systems*. Springer, 2012.
- [27] T. Kloda, A. Bertout, and Y. Sorel, "Latency analysis for data chains of real-time periodic tasks," in *ETFA*. IEEE, 2018.
- [28] T. Kloda, J. Chen, A. Bertout, L. Sha, and M. Caccamo, "Latency analysis of self-suspending task chains," in *DATE*. IEEE, 2022.
- [29] A. M. Kordon and N. Tang, "Evaluation of the age latency of a real-time communicating system using the LET paradigm," in *ECRTS*, ser. LIPIcs, vol. 165. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- [30] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmarks for free," in *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2015.
- [31] H. Lee, Y. Choi, T. Han, and K. Kim, "Probabilistically guaranteeing end-to-end latencies in autonomous vehicle computing systems," *IEEE Trans. Computers*, vol. 71, no. 12, 2022.
- [32] X. Li, Y. Ma, Y. Chen, J. Sun, W. Chang, N. Guan, L. Chen, and Q. Deng, "Priority optimization for autonomous driving systems to meet end-to-end latency constraints," in *RTSS*. IEEE, 2024.
- [33] X. Luo, H. Liang, Y. Tang, X. Jiang, N. Guan, and W. Yi, "Timing analysis of cause-effect chains for external events with finite validity intervals," in *SETTA*, ser. Lecture Notes in Computer Science, vol. 15469. Springer, 2024.
- [34] J. Martinez, I. Sañudo, and M. Bertogna, "End-to-end latency characterization of task communication models for automotive systems," *Real Time Syst.*, vol. 56, no. 3, 2020.
- [35] M. D. Natale, W. Zheng, C. Pinello, P. Giusto, and A. L. Sangiovanni-Vincentelli, "Optimizing end-to-end latencies by adaptation of the activation events in distributed automotive systems," in *IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE Computer Society, 2007.
- [36] C. Pagetti, D. Saussière, R. Gratia, E. Noulard, and P. Siron, "The ROSACE case study: From simulink specification to multi/multi-core execution," in *19th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2014.
- [37] P. Pazzaglia and M. Maggio, "Characterizing the effect of deadline misses on time-triggered task chains," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 41, no. 11, 2022.
- [38] PerceptIn, "RTSS 2021 industry challenge," in *RTSS*, 2021, <https://2021.rtss.org/wp-content/uploads/2021/06/RTSS2021-Industry-Challenge-v2.pdf>.
- [39] J. Ren, S. Li, R. Bi, J. Zhou, Y. Zheng, X. Qi, and X. He, "Efficient maximum reaction time analysis for data chains of real-time tasks in multiprocessor systems," *J. Syst. Archit.*, vol. 168, 2025.
- [40] X. Shi, Y. Zhu, and L. Li, "End-to-end probability analysis method for multi-core distributed systems," *J. Supercomput.*, vol. 80, no. 19, 2024.
- [41] Y. Tang, N. Guan, X. Jiang, Z. Dong, and W. Yi, "Reaction time analysis of event-triggered processing chains with data refreshing," in *DAC*. IEEE, 2023.
- [42] H. Teper, M. Günzel, N. Ueter, G. von der Brüggen, and J. Chen, "End-to-end timing analysis in ROS2," in *RTSS*. IEEE, 2022.
- [43] H. Teper, T. Betz, M. Günzel, D. Ebner, G. von der Brüggen, J. Betz, and J. Chen, "End-to-end timing analysis and optimization of multi-executor ROS 2 systems," in *RTAS*. IEEE, 2024.
- [44] G. von der Brüggen, M. Nayeibi, J. Shi, K.-H. Chen, and J.-J. Chen, "Evaluation framework for self-suspending task systems," in *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2019.
- [45] S. Wang, E. Bini, Q. Deng, and M. Maggio, "Jitter propagation in task chains," in *RTSS*. IEEE, 2025.
- [46] Q. Zhu, H. Zeng, W. Zheng, M. D. Natale, and A. L. Sangiovanni-Vincentelli, "Optimization of task allocation and priority assignment in hard real-time distributed systems," *ACM Trans. Embed. Comput. Syst.*, vol. 11, no. 4, 2012.