(ITEA 3 – 17003)

PANORAMA

Boosting Design Efficiency for Heterogeneous^3 Systems

Deliverable: D 4.3

Tools for performance assessment

Work Package: 4

Result visualization and assessments

Task: 4.3 – 4.5

Specification of methods and strategies for visualizing system effectiveness
Analysis and evaluation of existing visualization frameworks and technologies
Implementation of Assessment Eclipse Tooling

Document Type:	Deliverable (Software)	**Classification:**	Public
Document Version:	V1.2	**Contract Start Date:**	2019-04-01
Document Preparation Date:	2022-03-31	**Duration:**	2022-03-31

History

Rev.	Content	Resp. Partner	Date
0.1	Initial template	Lukas Krawczyk	2019-11-06
0.2	Initial structure and Chapter on Response Time Analyzer added	Robert Höttger	2020-02-06
0.3	Minor restructuring, generic summary	Lukas Krawczyk	2020-03-20
0.4	Introduction and Chapter on Task Visualizer	Lukas Krawczyk	2020-03-25
1.0	Finalized RC1	Lukas Krawczyk	2020-03-31
1.1	Added content on Web-Services	Lukas Krawczyk Philip Okonkwo	2021-03-31
1.2	Added content on Activation Pattern Visualization, final cleanup	Philip Okonkwo Daniel Fruhner Fabian Kneer Lukas Krawczyk	2022-03-31

Contents

List of Figures

Summary

This document provides a brief documentation for the contributions related to the software deliverable D4.3 "Tools for Performance assessment" of the PANORAMA Work Package 4 "Result visualization and assessments". It contains a subset of the results related to following tasks:

- T4.3 "Specification of methods and strategies for visualizing system effectiveness" that supports assessing a systems efficiency along with the effectiveness of its subsystems by specifying quality attributes along with methods for comparing and recommending design alternatives as well as interfaces for visualizing these results.
- T4.4 "Analysis and evaluation of existing visualization frameworks and technologies" that enables delivering easily understandable results and reduced training periods by a coherent and uniform representation of the visualization results using a common and compatible basis of frameworks.
- T4.5 "Implementation of Assessment Eclipse Tooling" that aims at providing prototypical implementations of the visualizers and assessment tooling defined in Tasks 4.1 – 4.3 and integrating it along with the frameworks and technologies determined in Task 4.4.

While visualizers are required to provide an easily understandable representation of the results from WP3, such as details on the energy consumption of system aspects, performance characteristics or safety/security relevant information, assessment techniques will be required to support e.g. decisions or comparing large amounts of results.

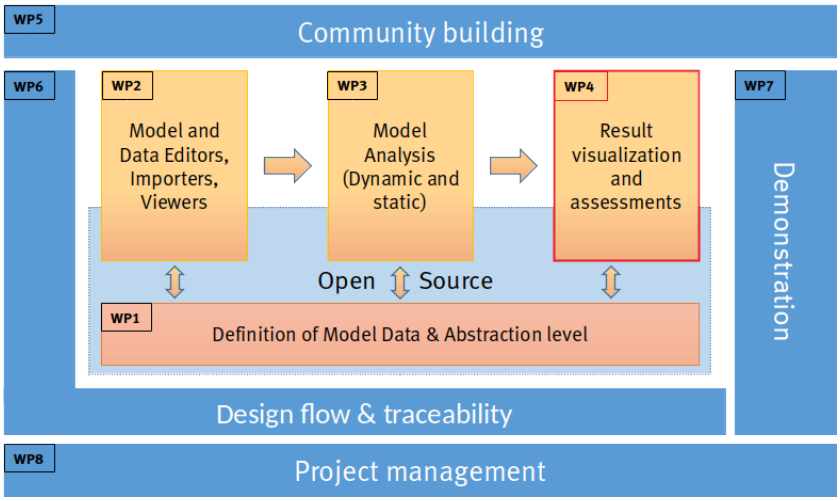


Figure 1: PANORAMA Work Package Structure including their relation to WP4

In order to put this document into context, the relation between WP4 (red box) and the other work packages is illustrated in Figure 1. The project is organized into two groups of work

packages: The technical solutions to the project objectives are developed by WP1 – 4 (colored in red / orange), whereas WP5 – 8 cover management, packaging, and dissemination related topics (colored in blue).

1 Introduction

The complexity of future solutions for mobility results in intricate and unforeseen impact of product and project decisions on systems level, even in late development phases. Especially software development in mobility, one of the main factors for innovation and value creation – and costs, must be evaluated also in system context. To cope with this fact, the early assessment of design decisions is a key factor for success.

The objective of WP4 is to enable this assessment by providing methods, processes and tools for visualizing and assessing the results of analysis and simulation (cf. Figure 1.1).

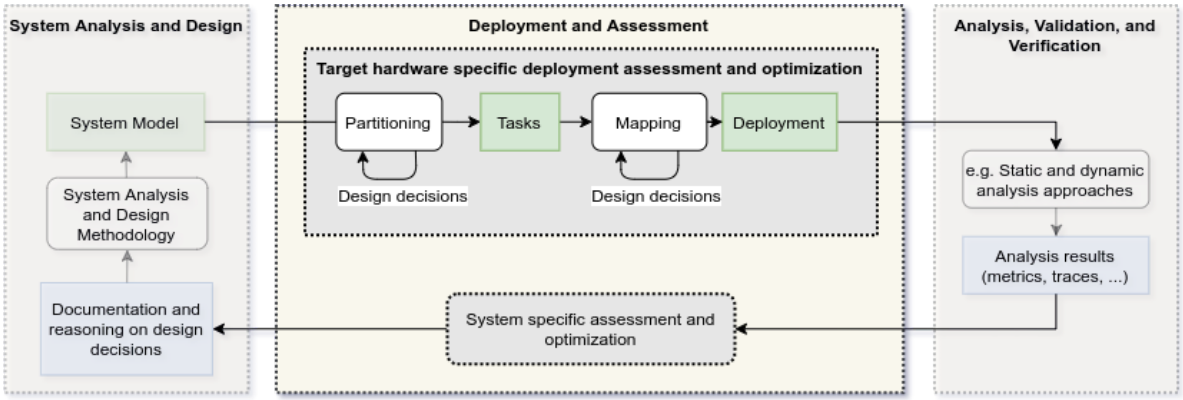


Figure 1.1: High-Level overview of a typical design flow within mobility systems development

The majority of inputs is expected to come from the analysis approaches related to WP3, such as details on energy consumption of system aspects, timing and performance characteristics, or safety/security relevant information. In order to support e.g. critical design decisions or comparing large amounts of such results there is a need for assessment techniques. Henceforth, WP4 deals with visualizers that are a key enabler to provide an easily understandable representation of the analysis results.

The remainder of this document is structured as follows. Chapter 2 presents a prototype for a Response Time Analysis tool that has been developed at Dortmund University of Applied Sciences and Arts along with a Google Summer of Code Project. It supports the assessment of embedded systems featuring self suspending tasks by performing various analysis on e.g. event chains following an implicit or direct communication paradigm. The APP4MC Task Visualizer is documented in Chapter 3.

It describes in particular the various assessment capabilities of an applications temporal behavior by visualizing e.g. the scheduling on a (heterogeneous) multi-core system, the various states of executed tasks including preemption points, communication behavior such as data propagation paths, identified over- and under-sampling effects, or the data flow within a systems task structure.

The Activation Pattern Visualizer is documented in Chapter 4. It is implemented in terms of

an Eclipse APP4MC plugin that determines and visualizes the lower- and upper bounds on the number of occurrances from activation events for different stimuli and specific time intervals.

The APP4MC Web Services related to assessing the performance of automotive embedded systems using Amalthea Data Models are described in Chapter 5. These allow performing rudimentary analysis and assessment operations, such as visualizing analysis results or quantifying and illustrating metrics using charts and diagrams.

Finally, Chapter 6 concludes this document with a brief summary and an outlook on the planned activities.

2 Response Time Analyzer

The `Response Time Analyzer` tool has been developed at Dortmund University of Applied Sciences and Arts along with a Google Summer of Code Project[1]. Its purpose is to provide an API for retrieving response times of tasks under Partitioned Fixed-Priority Preemptive Scheduling (P-FPPS). It requires an AMALTHEA software, hardware, and mapping model. Heterogeneous hardware is supported by using appropriate utility functions provided by APP4MC. Additionally, not only different processing unit properties such as frequency or instructions per second, but also processing unit types such as CPUs and GPUs are supported. While the former uses P-FPPS, the latter is analyzed for Weighted Round Robin (WRR) scheduling. If a mapping is not schedulable, the tool will inform its user accordingly.

Furthermore, the tool is able to retrieve end-to-end task chain latency values by analyzing reaction and age delays.

The tool comes with a small user interface (UI), which is shown in Figure 2.1.

The tool has already been published at the official APP4MC repositories at `https://bit.ly/370TT4T`. Furthermore, ideas and formal analyses have been published at [HKB+19].

[1] `https://gsoc-doc.readthedocs.io/en/latest/index.html`, visited 02.2020

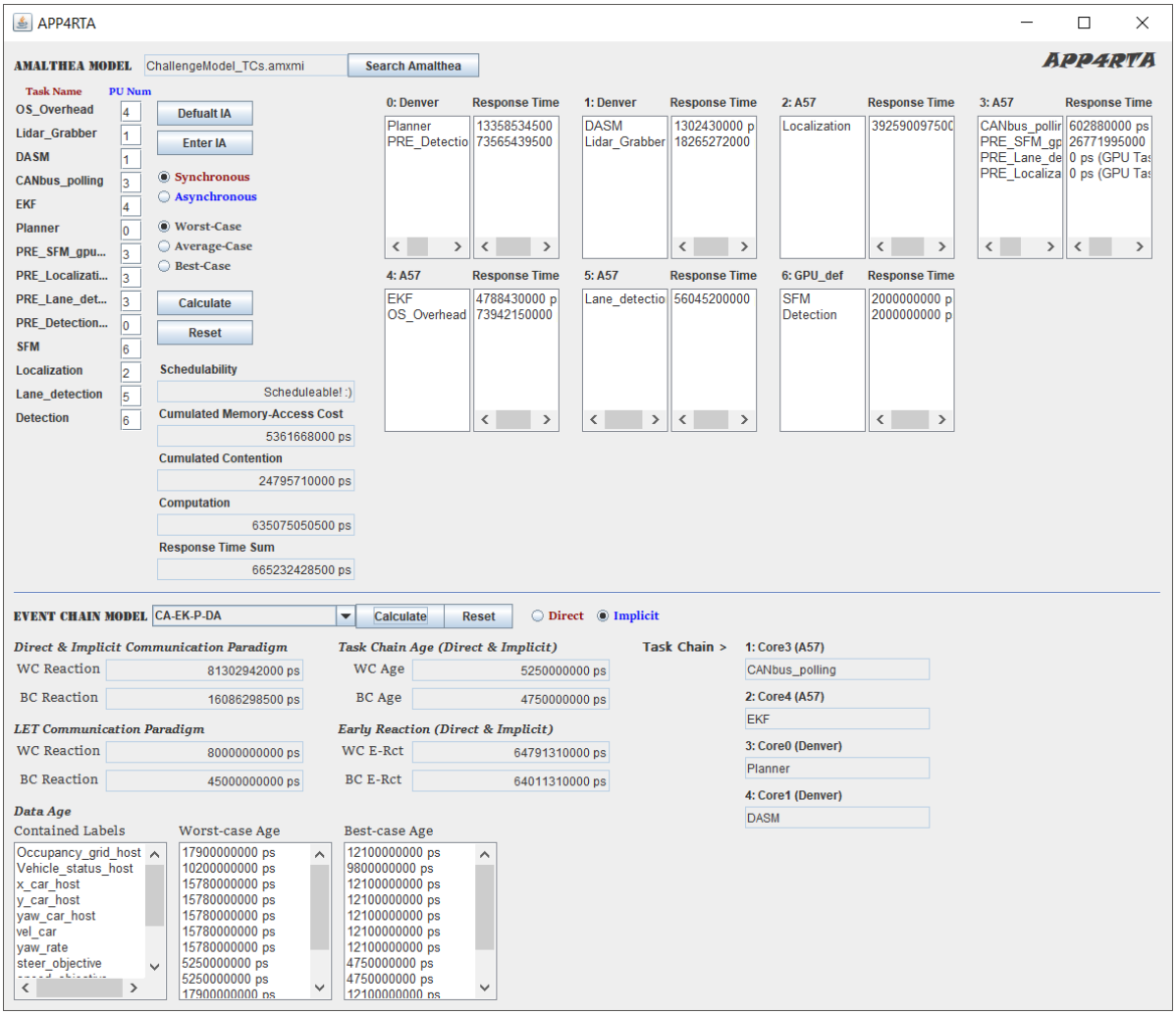


Figure 2.1: Response Time Analyzer UI

3 APP4MC Task Visualizer

The APP4MC Task Visualizer [APP] is a tool for assessing and embedded systems timing behavior by, among others, visualizing the execution of tasks along with their states and state changes on the resp. executing cores.

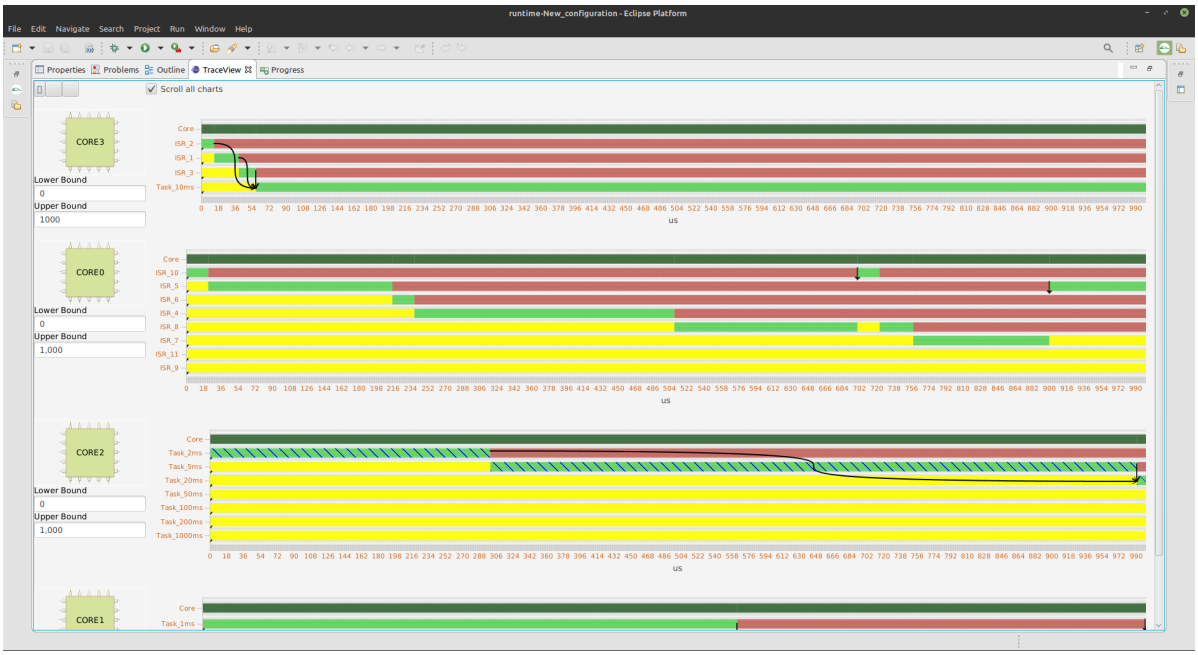


Figure 3.1: Gantt-Chart illustrating the execution in total 21 tasks and interrupt service routines on a quad-core ECU

The execution of such a system based upon the 2016 WATERS Industrial Challenge Model is illustrated in Figure 3.1. It shows the execution of 21 tasks and interrupt service routines on an ECU consisting of four cores. The colors of each bar segment indicate the status of a task, in particular:

- Green: Executing / Running
- Yellow: Ready
- Orange: Suspended / Interrupted
- Red: Finished

The APP4MC Task Visualizer has been further extended by support for assessing data propagation within event chains as well as communication among individual task sets for both, propagation at a structural level (i.e. among tasks, runnables, and labels regardless of their instances) as well as job level (i.e. the concrete instances created of a task's releases).

The visualization of data propagation on *job-level* is illustrated in Figure 3.2. In this example, data is written by task `TASK_ESSP3` upon its response time at time instant 10 and read by `TASK_ESSP8` at its release time at time instant 30. This view does not only allow visualizing under- and oversampling effects but also visually derive various end-to-end metrics such as the reaction or data age latency.

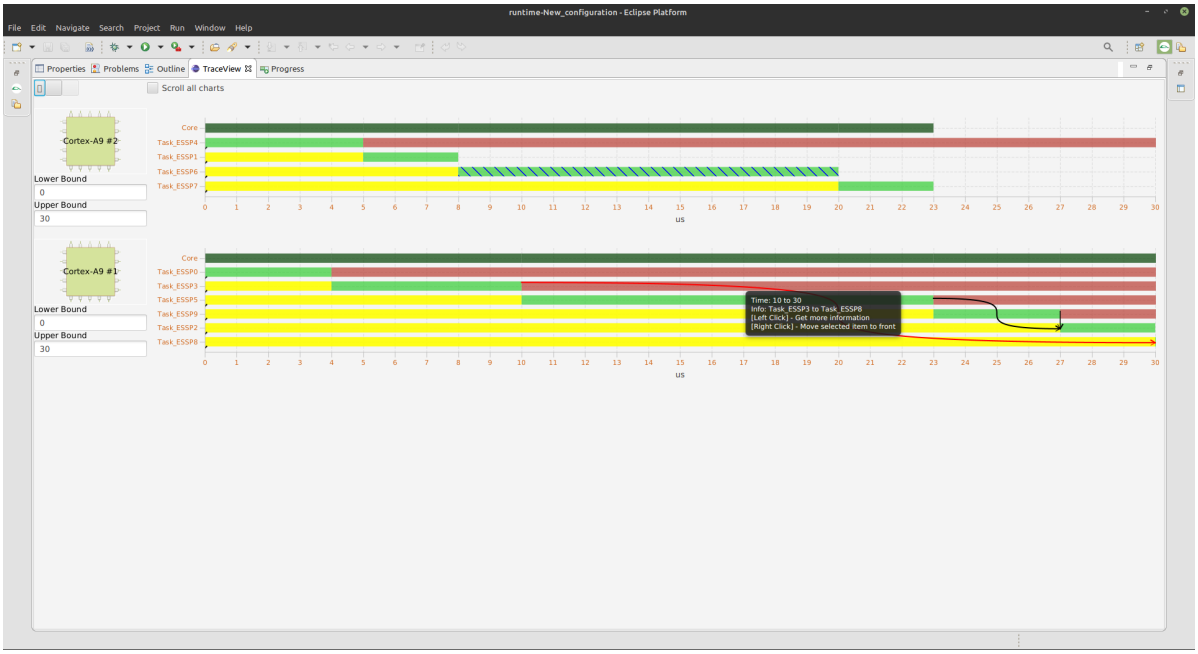


Figure 3.2: Gantt-Chart illustrating the execution of 10 tasks, including data propagation between the labels shared among those tasks

Propagation on structural level can be further visualized and assessed using the Event Chain View, which draws data propagation at the granularity of involved tasks (red boxes), runnables (teal boxes), and labels (white boxes).

This view is shown in Figure 3.3 and illustrates the second event chain of the 2016 WATERS Industrial Challenge model. Here, task `Task_100ms` is calling (red arrow) `Runnable_100ms_7` that writes (red arrow) its results to `Label_4258`. This Label is read (blue arrow) by `Runnable_10ms_19`, that is called by `Task_10ms` and writes its results back to `Label_2197`. Finally, `Label_2197` is read by `Runnable_2ms_8`, that is called by `Task_2ms`. Yellow arrows are used to further highlight the temporal order of label accesses, e.g. `Label_2197` is accessed after `Label_4258`.

Beyond that, the APP4MC Task Visualizer supports assessing custom data propagation paths. By selecting a set of task, runnables, and labels (cf. Figure 3.4) the tools automatically identifies all the available paths among that selection and draws it in a custom view.

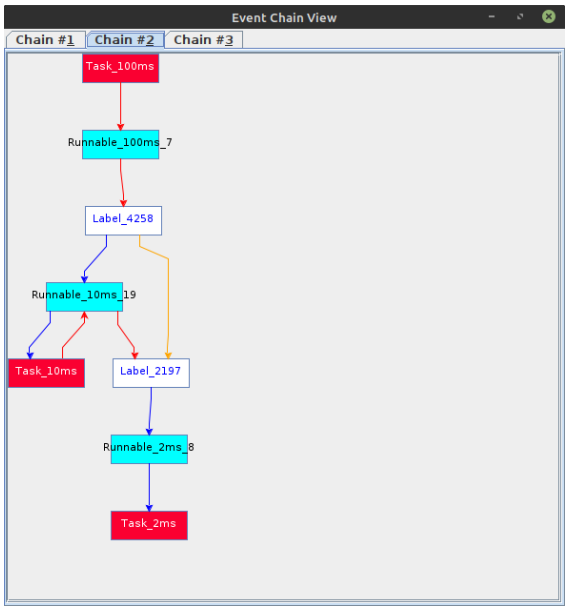


Figure 3.3: Event Chain View for the second Event Chain of the 2016 WATERS Industrial Challenge Model

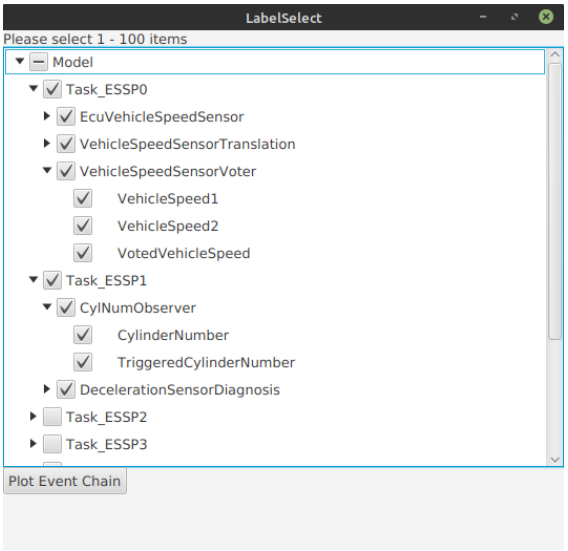


Figure 3.4: LabelSelect configuration dialog, based on the Democar Model Example

4 Activation Pattern Visualizer

The Activation Pattern Visualizer is a plugin that illustrates worst- and best case trigger patterns of processes according to their triggering stimuli. It utilizes the Eclipse APP4MC visualization framework and is based on the complex event model library presented in [KBM+22].

The activation pattern visualization plugin only generates the visualization based on the specific time window (also known as *time interval* Δt), which is specified in the user interface. Figure 4.1 illustrates the worst and best-case activation patterns of a periodic process with an inter-arrival interval of 200ms from the WATERS 2017 FMTV challenge model [HDK+17]. In this example, the time window is set to an interval of $\Delta t = 1000ms$ (see *Visualization Parameters* on the left side of the figure).

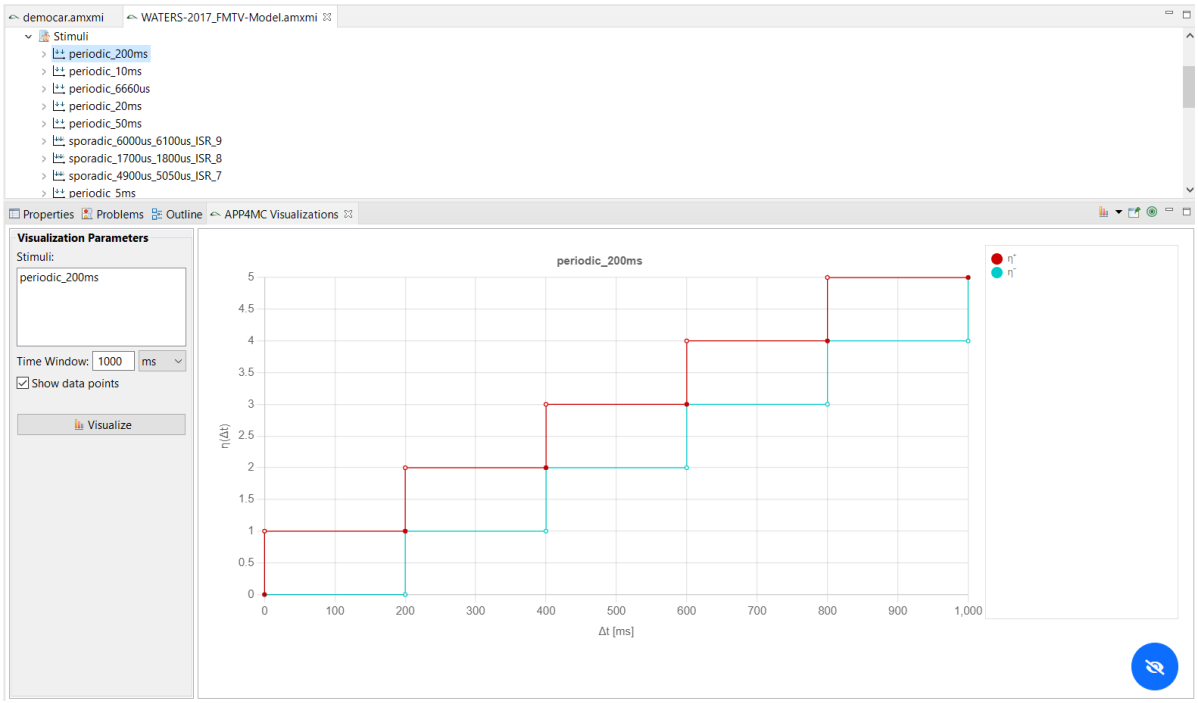
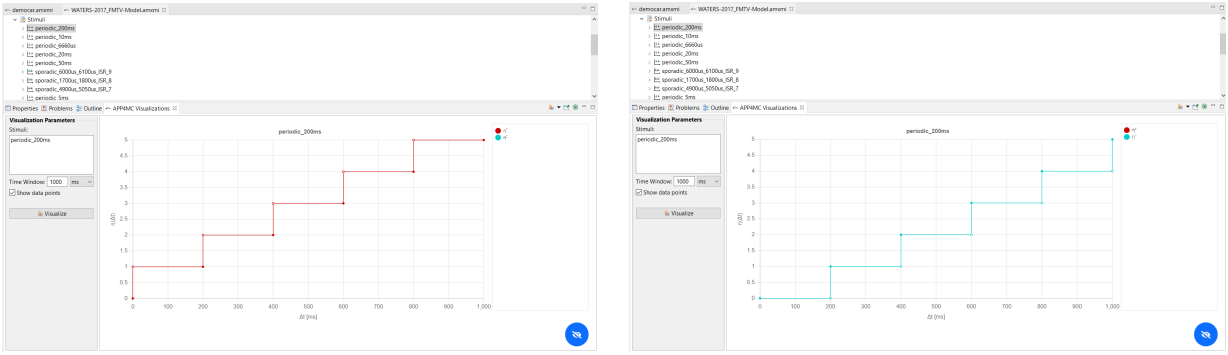


Figure 4.1: The Activation Pattern visualization of a single Stimulus

The red line represents the upper-bound arrival curve η^+, and the cyan line represents the corresponding lower-bound arrival curve η^-, both denoting the maximum and minimum numbers of activation events observed in any time interval Δt respectively. We differentiate between filled circles, which represent included points, and hollow circles, which denote excluded points.

The visibility of each line of the activation pattern can be toggled in order to filter specific functions and for the sake of comprehensibility, as shown in Figure 4.2.

Figure 4.3 shows an example of 4 different stimuli being visualized at the same time. All stimuli are visualized simultaneously in a single diagram. The plugin provides two visualization



(a) (b)

Figure 4.2: Activation Pattern Diagram with hidden lines

options: The first option is invoked by selecting the *stimuli model* of an Amalthea model and draws the arrival functions for all stimuli that have been modeled. The second option is triggered by selecting one or more explicit stimuli, which are visualized together.

The plugin also supports common navigational functions, such as zooming, for a proper visualization of stimuli with e.g., small periods.

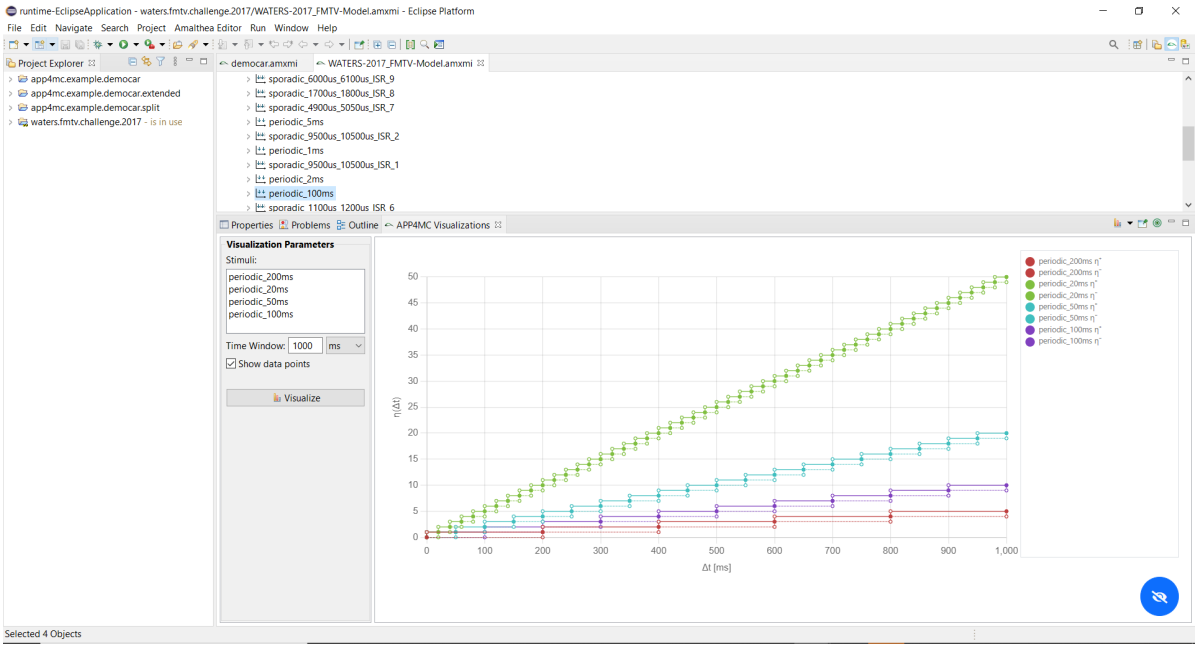


Figure 4.3: The Activation Pattern Eclipse plugin visualization

5 APP4MC Web Services

5.1 Visualization Service

The APP4MC Visualization Service is a microservice capable of visualizing different charts, diagrams, and plots, which are commonly used to present analysis results or a system's information. It consumes a JSON file that contains a set of metrics and to be visualized, and produces a downloadable HTML file as an output. Though it was designed to work with APP4MC Cloud Manager, it implements a REST interface that enables it to be used, if needed, as a stand-alone service, or in conjunction with other tools, such as [KBGW19a; KBGW19b; BKGW20; BKW21]. Currently, four types of charts are supported:

- Bar Charts
- Stacked Bar Charts
- Box Plot Charts
- Spider Charts

The visualized data is usually a collection of different kind of charts, along with the data to be displayed, which are organized in the form of a grid, with each chart as an item in the grid (cf. Figure 5.1).

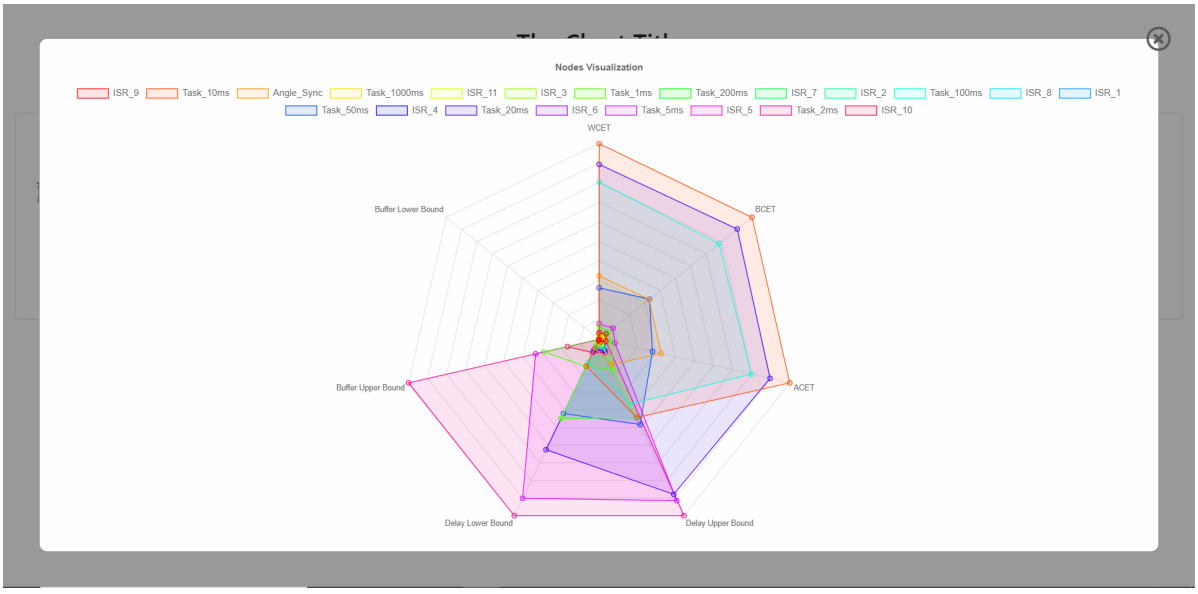


Figure 5.1: Chart Visualizer Showing a bar chart, a boxplot chart and a spider chart

Each item can be expanded into a detailed view by clicking on the corresponding chart. The detailed view shows the legends and any other data required for fully understanding the illustrated information from the particular chart (cf. Figure 5.2).

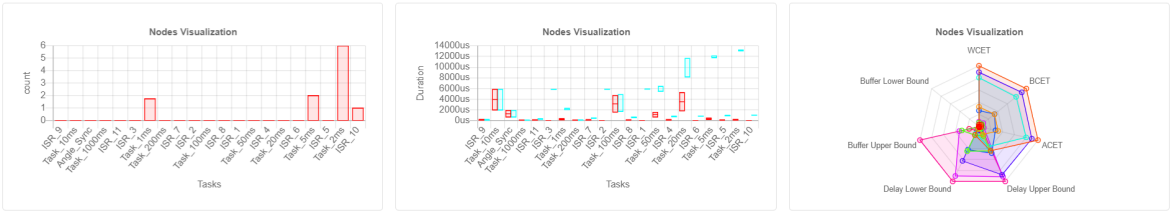


Figure 5.2: Detailed view of a spider chart illustrating the analysis results of a system consisting of 21 processes.

Both a demo[1] of the services as well as the source code[2] are available online.

5.2 Label Services

The APP4MC Label Services allow assessing the label read and write metrics from Amalthea models. It provides an interface between Amalthea model and the Chart Visualizer service by consuming Amalthea models, quantifying label accesses, and producing a downloadable JSON output containing the visualization data. The APP4MC Label Services supports the generation of four kinds of label metrics:

- Labels per Task: Quantifies the number of read, write, as well as read-and-write labels for each *Process*.
- Labels per Core: Quantifies the number of read, write, as well as read-and-write labels for each *Processing Unit*.
- Labels per Memory: Quantifies the number of read, write, as well as read-and-write labels for each *Memory*.
- Label per Size: Counts the number of labels in the input model per size (bit width) and visualizes a corresponding histogram.

The visualization of any or all out of these 4 label metrics can be configured in the APP4MC Cloud Manager. Additionally, it implements a REST interface that allows its usage as stand-alone application.

Both a demo[3] of the services as well as the source code[4] are available online.

Example Workflow (Static Analysis): Figure 5.3 depicts a possible workflow in the APP4MC Cloud Manager that utilizes the Label Services as well as the Visualizer Services in order to conduct a static analysis. It reads the Amalthea model, performs the label analysis in order to get a set of label metrics, and finally visualizes the results by generating an HTML output (cf. Figure 5.4).

[1] `https://app4mc.eclipseprojects.io/app4mc/visualization`
[2] `https://gitlab.idial.institute/panorama.project/src/app4mc-chart-visualizer`
[3] `https://cloud.panorama-research.org`
[4] `https://gitlab.idial.institute/org.eclipse.app4mc.cloud`

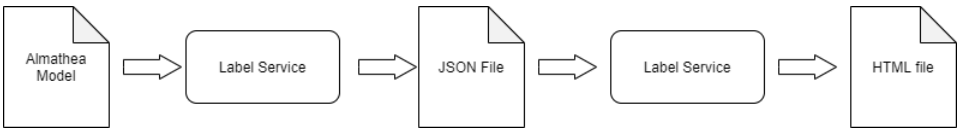


Figure 5.3: Static Analysis Workflow diagram.

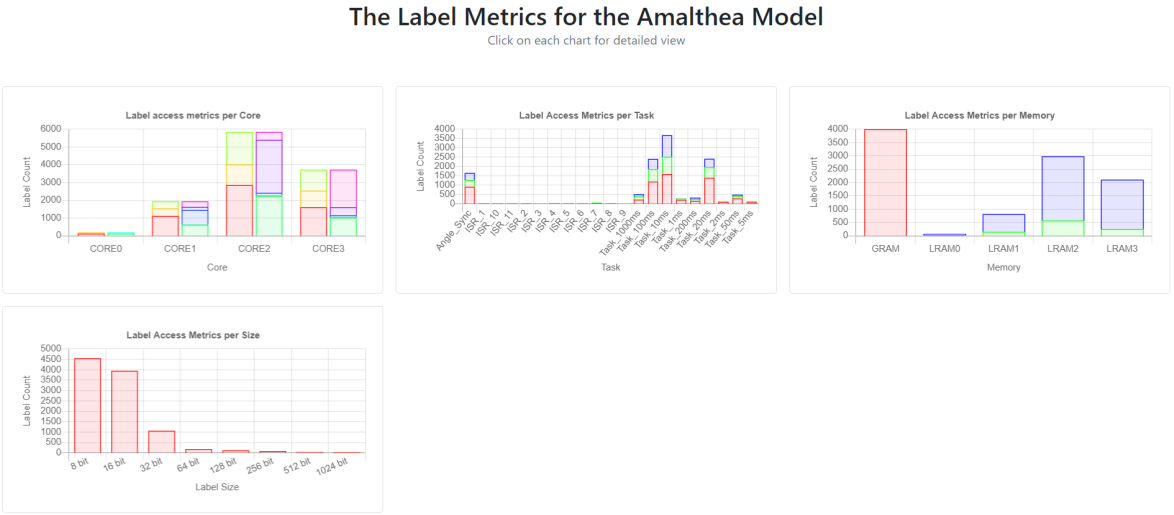


Figure 5.4: The result of a static analysis workflow.

5.3 Activation Pattern Service

The Activation Pattern service is a web service that visualizes best and worst-case trigger patterns corresponding to the stimuli described by Amalthea models. It utilizes the event model library [KBM+22] for determining the different trigger patterns of a stimulus and transforms the resulting lower and upper bound curves into JSON files. These generated files can be consumed by the visualization service to illustrate the activation patterns. A general explanation on how the activation pattern visualization works has been previously provided in Chapter 4. The difference between the web service and the Eclipse APP4MC plugin is mainly the level of integration and the way the visualization is called. Both, the previous described Activation Pattern Visualizer and the cloud service, share an option to specify the size of the time window. The cloud service also has an option to constrain the number of visualized activation events. This option becomes in particularly useful whenever the frequencies, at which processes are triggered, deviate by order of magnitude. Another minor difference lies in the generated visualization. The web service exploits a web-browsers capabilities and produces results that are represents in a grid of line charts. Each chart contains two lines that represent the upper and lower bound arrival curves. Figure 5.5 shows an example of an activation pattern result which was consumed by the Chart Visualizer service.

For each chart, a detailed view can be opened by clicking the corresponding XY-Plot. Figure 5.6 shows an example of the detail view.

Both a demo[5] of the services as well as the source code[6] are available online.

[5] `https://cloud.panorama-research.org`
[6] `https://gitlab.idial.institute/org.eclipse.app4mc.cloud`

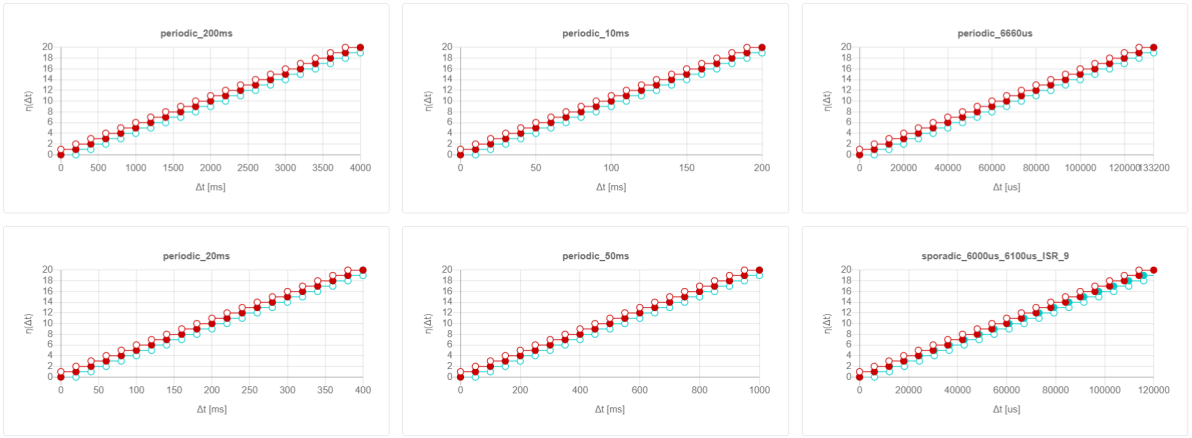


Figure 5.5: Chart Visualizer showing results from the Activation Pattern service

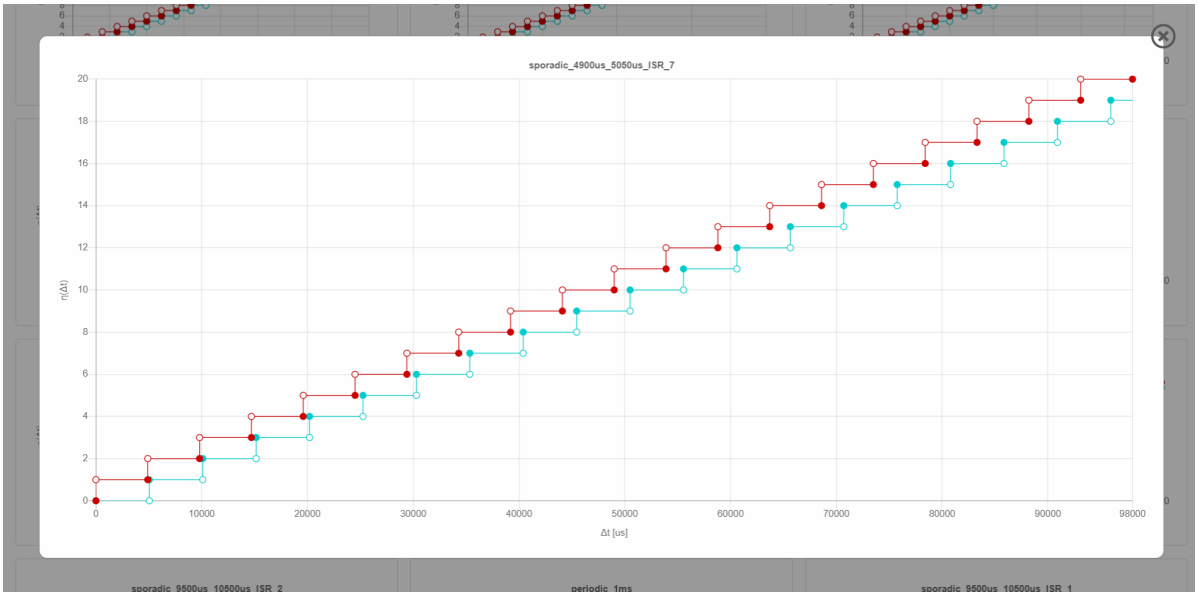


Figure 5.6: Detailed view of the trigger pattern generated by a relative periodic stimulus, which starts to draft for larger time intervals

6 Conclusion and Outlook

This document documented the prototypical implementations related to software deliverable D4.3 of Work Package 4 in the PANORAMA project. It illustrated various selected applications that were developed alongside Tasks T4.3 – T4.5 and highlights their applicability towards visualizing and assessing an applications timing behavior.

It briefly describes the following software contributions:

- **Response Time Analyzer** that supports evaluating and verifying the deployment of software to hardware by performing schedulability tests and calculating various timing information such as response times or end-to-end latency.
- **APP4MC Task Visualizer** that supports the visual verification of deployment by visualizing the simulation of a system and highlighting various information such as (among others) a tasks response times, an event chains end-to-end latency, over- and under-sampling effects, or memory accesses.
- **Activation Pattern Visualizer** supports the development of heterogeneous systems by visualizing the activation times of stimuli.
- **APP4MC Services** that support assessing heterogeneous embedded systems by e.g. quantifying metrics and visualizing analysis results, such as bounds on response times and the number of consecutive deadline misses.

We hope that this documentation will allow others to efficiently utilize the performance assessment tooling, provide guidelines on how to develop and extend web-services tailored around the Eclips APP4MC ecosystem, and inspire the open-source community to contribute its own ideas.

Bibliography

[APP] APP4MC Consortium, *App4mc documentation*, https://www.eclipse.org/app4mc/help/app4mc-0.9.7/.

[BKGW20] M. Bazzal, L. Krawczyk, R. P. Govindarajan, and C. Wolff, "Timing Analysis of Car-to-Car Communication Systems Using Real-Time Calculus: A Case Study," in *2020 IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*, vol. 17, IEEE, Sep. 2020, pp. 1–8, ISBN: 978-1-7281-9960-3. DOI: 10.1109/IDAACS-SWS50031.2020.9297100. [Online]. Available: https://ieeexplore.ieee.org/document/9297100/.

[BKW21] M. Bazzal, L. Krawczyk, and C. Wolff, "RTCAnalysis: Practical Modular Performance Analysis of Automotive Systems with RTC," in *Communications in Computer and Information Science*, 2021, pp. 209–223. DOI: 10.1007/978-3-030-88304-1_17. [Online]. Available: https://link.springer.com/10.1007/978-3-030-88304-1%7B%5C_%7D17.

[HDK+17] A. Hamann, D. Dasari, I. Kramer, M. Pressler, F. Wurst, and D. Ziegenbein, "WATERS Industrial Challenge 2017," in *8th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2017.

[HKB+19] R. Höttger, J. Ki, T. B. Bui, B. Igel, and O. Spinczyk, "CPU-GPU Response Time and Mapping Analysis for High-Performance Automotive Systems," *10th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS '19), co-located with the 31st Euromicro Conference on Real-Time Systems (ECRTS '19)*, Jul. 2019.

[KBGW19a] L. Krawczyk, M. Bazzal, R. P. Govindarajan, and C. Wolff, "An analytical approach for calculating end-to-end response times in autonomous driving applications," in *10th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2019)*, 2019.

[KBGW19b] ——, "Model-Based Timing Analysis and Deployment Optimization for Heterogeneous Multi-core Systems using Eclipse APP4MC," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, IEEE, Sep. 2019, pp. 44–53, ISBN: 978-1-7281-5125-0. DOI: 10.1109/MODELS-C.2019.00013. [Online]. Available: https://ieeexplore.ieee.org/document/8904877/.

[KBM+22] L. Krawczyk, M. Bazzal, H. Mackamul, R. Weber, and C. Wolff, "Complex event models for automotive embedded systems," *Journal of Systems Architecture*, vol. 123, p. 102 343, Feb. 2022, ISSN: 13837621. DOI: 10.1016/j.sysarc.2021.102343. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1383762121002368.