From FMTV to WATERS: Lessons Learned from the First Verification Challenge at ECRTS (Invited Paper)

Sebastian Altmeyer, Étienne André, Silvano Dal Zilio, Loïc Fejoz, Michael González Harbour, Susanne Graf, J. Javier Gutiérrez, Rafik Henia, Didier Le Botlan, Giuseppe Lipari, et al.

From FMTV to WATERS: Lessons Learned from the First Verification Challenge at ECRTS

Sebastian Altmeyer
Universität Augsburg, Germany

Étienne André[1]
Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, F-93430 Villetaneuse, France

Silvano Dal Zilio
Univ. de Toulouse, INSA, LAAS, F-31400 Toulouse, France

Loïc Fejoz
RealTime-at-Work (RTaW), 615, Rue du Jardin Botanique, F-54600 Villers-lès-Nancy, France

Michael González Harbour
Universidad de Cantabria, Santander, Spain

Susanne Graf
Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, F-38000 Grenoble, France

J. Javier Gutiérrez
Universidad de Cantabria, Santander, Spain

Rafik Henia[1]
Thales Research & Technology, F-91767 Palaiseau, France

Didier Le Botlan
Univ. de Toulouse, INSA, LAAS, F-31400 Toulouse, France

Giuseppe Lipari
Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France

Julio Medina
Universidad de Cantabria, Santander, Spain

Nicolas Navet
University of Luxembourg, Luxembourg

Sophie Quinton[1]
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France

Juan M. Rivas
Universidad de Cantabria, Santander, Spain

Youcheng Sun
The University of Manchester, UK

Abstract

We present here the main features and lessons learned from the first edition of what has now become the ECRTS industrial challenge, together with the final description of the challenge and a comparative overview of the proposed solutions. This verification challenge, proposed by Thales, was first discussed in 2014 as part of a dedicated workshop (FMTV, a satellite event of the FM 2014 conference), and solutions were discussed for the first time at the WATERS 2015 workshop. The use case for the verification challenge is an aerial video tracking system. A specificity of this system lies in the fact that periods are constant but known with a limited precision only. The first part of the challenge focuses on the video frame processing system. It consists in computing maximum values of the end-to-end latency of the frames sent by the camera to the display, for two different buffer sizes, and then the minimum duration between two consecutive frame losses. The second challenge is about computing end-to-end latencies on the tracking and camera control for two different values of jitter. Solutions based on five different tools – Fiacre/Tina, CPAL (simulation and analysis), IMITATOR, Uppaal and MAST – were submitted for discussion at WATERS 2015. While none of these solutions provided a full answer to the challenge, a combination of several of them did allow to draw some conclusions.

2012 ACM Subject Classification Computer systems organization → Real-time systems; Computer systems organization → Embedded systems; General and reference → Verification; Software and its engineering → Software verification and validation

Keywords and phrases Verification challenge, industrial use case, end-to-end latency

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2023.19

[1] Corresponding author.

35th Euromicro Conference on Real-Time Systems (ECRTS 2023).
Editor: Alessandro V. Papadopoulos; Article No. 19; pp. 19:1–19:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Category Invited Paper

Supplementary Material *Software (ECRTS 2023 Artifact Evaluation approved artifact)*: `https://doi.org/10.4230/DARTS.9.1.4`

Funding This work was partially supported by MCIN/ AEI /10.13039/501100011033/ FEDER "Una manera de hacer Europa" under grant PID2021-124502OB-C42 (PRESECREL) and by the AIDOaRt project, an ECSEL Joint Under-taking (JU) under grant agreement No. 101007350.
Étienne André: Partially supported by the ANR-NRF French-Singaporean research program ProMiS (ANR-19-CE25-0015 / 2019 ANR NRF 0092) and ANR BisoUS (ANR-22-CE48-0012).

1 Introduction

Many model-based techniques and tools have been developped for the timing estimation and verification of critical real-time systems. One can classify these approaches into three categories: simulation, model checking and response-time analysis. The many existing tools apply to different but sometimes similar models and may provide different types of guarantees. This makes it difficult for researchers and practitioners to understand the advantages and drawbacks of one approach compared to another, or even simply to figure out which tools can perform a given type of analysis on a given system.

The WATERS industrial challenge was introduced in 2015 to address this issue by providing an opportunity for researchers to try their favorite tool on a practical problem. For Thales, who proposed the first challenge, this was an opportunity to better understand how various analysis methods and tools proposed by the research community can be applied to the large variety of real-time requirements of the Thales products (ranging from hard to soft real-time requirements). For the research community, the main motivation for participating to the challenge was to address concrete timing analysis problems issued from real industrial case studies. Such a challenge thus promotes discussions and closer interactions between research and industry.

A preliminary version of the challenge was presented and discussed at the FMTV 2014 workshop (FMTV standing for "Formal Methods for Timing Verification"), a satellite event of FM 2014 (the 19th International Symposium on Formal Methods) [11]. An improved version was then proposed for WATERS 2015 (the 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems) [22], a satellite event of ECRTS 2015. Five solutions, representing all model-based timing analysis techniques (model checking, response-time analysis and simulation), were submitted that year. Interestingly, no tool was able to solve all subchallenges, which shows that there is currently no unique solution that fits every timing verification problem. The combined use of several tools, however, led to better results than those provided by each individual tool, while increasing the confidence in the produced results.

The first WATERS industrial challenge provided Thales, the solution providers as well as all WATERS 2015 attendees with a better understanding of the various techniques and tools, and in particular of their strengths and weaknesses with respect to several aspects, such as: ease of modeling, level of automation of the verification process, verification time, reliability of the results, etc. Based on the feedback given by the solution providers with respect to the description of the challenge, Thales were also able to provide a consolidated version of the challenge including a corresponding model in Papyrus [13], for which solutions could subsequently be submitted (see e.g., [21]).

Although this challenge is quite ancient now, we believe that the lessons learned from this experience and the research perspectives that it opened are still relevant today. The purpose of this paper is therefore to share that knowledge and to make available to the community

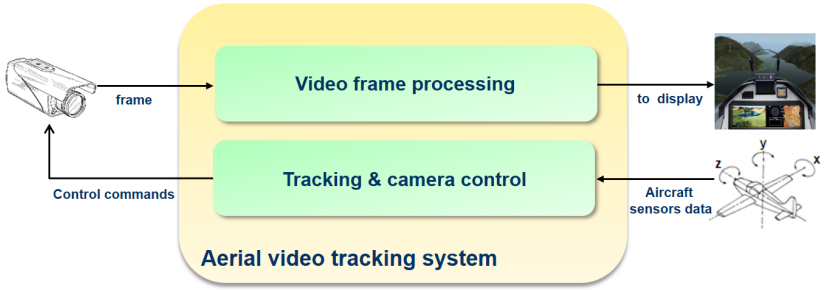

Figure 1 Subsystems of the aerial video tracking system.

the final description of the challenge and a comparative overview of the proposed solutions. The authoritative model of the challenge as well as the code for several of the solutions is available as additional material submitted together with this paper.

2 Presentation of the verification challenge

The use-case provided by Thales consists of an aerial video system to detect and track moving objects, e.g., vehicles on a roadway. Aerial video tracking systems are mission critical real-time systems since they embed intelligence, surveillance, reconnaissance, tactical and security applications characterized by strict constraints on timing. The main system tasks consist in:

- displaying high quality video images to the user;
- following the tracked object even when it is temporarily hidden from view (e.g., the vehicle proceeds in and out of a tree obstructed area) through motion prediction;
- detecting patches of the image that may be moving differently from the background by combining image registration and motion prediction.

For simplicity, the use-case is limited to the timing related aspects of two subsystems of the aerial video tracking system, as represented in Figure 1: a video frame processing system and a tracking and camera control system. As suggested by its name, the first subsystem processes the video frames sent by the camera. This includes embedding tracking data into the video, converting the frames to the required format and displaying a high quality video running at 25 frames per second on the monitor. The second subsystem performs motion prediction for the tracked object. Based on this prediction and the aircraft sensors data (position, direction, speed, etc.) it calculates new camera angles and sends instructions to control the camera.

We propose timing verification challenges related to each subsystem of the aerial video tracking use-case.

2.1 Challenge 1: Video Frame Processing

The functional view of the video frame processing subsystem is illustrated in Figure 2. It consists of a sequence of 4 functions processing the video frames from the camera to the display. The `Pre-processing` function removes reflections from the frames and normalizes the intensity of the individual pixels. The `Processing` function embeds tracking information into

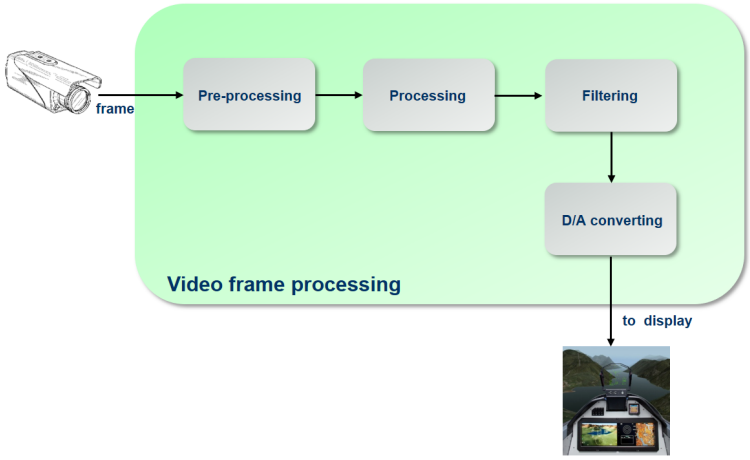

Figure 2 Functional view of the video frame processing subsystem.

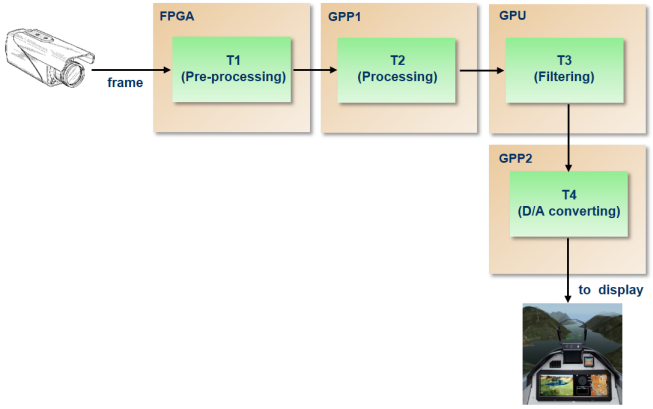

Figure 3 Architectural view of the video frame processing subsystem.

the pre-processed frames and executes zoom-in and zoom-out instructions. The `Filtering` function resizes the processed frames and removes noise. Finally, the `D/A converting` function converts the frames from digital to analog and sends them to the monitor.

For simplicity, we assume that each function is executed by a single task $\mathtt{T}_i$, as illustrated in Figure 3. All tasks are assumed to be mapped onto a different processor. Table 1 shows the execution time for tasks $\mathtt{T}_1$, $\mathtt{T}_3$ and $\mathtt{T}_4$ and the response time for task $\mathtt{T}_2$.[1]

Each frame sent by the camera activates task $\mathtt{T}_1$. The frames are sent strictly periodically with period $\mathtt{P}_1$, i.e., the time distance between two consecutive frames sent by the camera is *constant*. The exact value of $\mathtt{P}_1$ is however unknown since it may slightly vary from camera to camera. We know, however, that it ranges between $40\,ms - 0.01\%$ and $40\,ms + 0.01\%$. This *constant but unknown* value is a key aspect of the challenge.

[1] We need a response time rather than an execution time for $\mathtt{T}_2$ because it is running concurrently with other tasks that will be described in Challenge 2.

Table 1 Execution/response times of tasks from the video frame processing subsystem.

Task	Execution time
$\mathtt{T}_1$	$\mathtt{bcet}_1 = \mathtt{wcet}_1 = 28\,ms$
$\mathtt{T}_3$	$\mathtt{bcet}_3 = \mathtt{wcet}_3 = 8\,ms$
$\mathtt{T}_4$	$\mathtt{bcet}_4 = 1\,ms$; $\mathtt{wcet}_4 = 10\,ms$
Task	**Response time**
$\mathtt{T}_2$	$\mathtt{bcrt}_2 = 17\,ms$; $\mathtt{wcrt}_2 = 19\,ms$

After each execution, $\mathtt{T}_1$ sends a frame through its output that activates task $\mathtt{T}_2$. A register `R` is used for the communication between $\mathtt{T}_2$ and $\mathtt{T}_3$. At the end of each execution, $\mathtt{T}_2$ overwrites the register content with the new frame.

When activated, task $\mathtt{T}_3$ reads the current frame stored in the register `R`. For simplicity, we assume that there are no conflicts between read and write accesses to `R`. The activation of $\mathtt{T}_3$ is strictly periodic, i.e., the value of period $\mathtt{P}_3$ is *constant*. However, due to minor uncertainties in the clock implementation, the exact value of $\mathtt{P}_3$ is unknown: it ranges between $\frac{40}{3}\,ms - 0.05\%$ and $\frac{40}{3}\,ms + 0.05\%$. Note that, since task $\mathtt{T}_3$ is activated more frequently than task $\mathtt{T}_2$, it will process the same register content more than once.

At the end of each execution, task $\mathtt{T}_3$ produces a frame. Frames originating from the same register content are identical copies and are therefore assigned identical indices. The frames are inserted into a buffer `Buf` read by $\mathtt{T}_4$. Buffer `Buf` has size `n`. For each frame, the following conditions must be met to get actually stored in `Buf`:

1. It is not full.
2. No other frame having the same index (i.e., identical copy) has already been stored in the buffer.

Otherwise, the frame is discarded. We call this a *smart insert* function. The time required to discard a frame or to store it in `Buf` can be ignored.

Task $\mathtt{T}_4$ is activated strictly periodically, i.e., the value of period $\mathtt{P}_4$ is *constant*. As for $\mathtt{T}_1$ and $\mathtt{T}_3$, the exact value of $\mathtt{P}_4$ is again unknown, but we know that it is in the range $40\,ms \pm 0.01\%$. Each activation of $\mathtt{T}_4$ leads to an execution. If buffer `Buf` is empty, the execution of $\mathtt{T}_4$ takes $1\,ms$. Otherwise, $\mathtt{T}_4$ consumes a single frame from the buffer, and in this case, its execution takes exactly $10\,ms$. Once a frame has been processed, task $\mathtt{T}_4$ sends it (at the end of its execution) to be displayed on the monitor.

Communication between processors, access to register `R` between $\mathtt{T}_2$ and $\mathtt{T}_3$ and access to buffer `Buf` between $\mathtt{T}_3$ and $\mathtt{T}_4$ are considered to not consume any time.

Challenge 1A

The first part of the video frame processing timing verification challenge is to analyze the latency $\mathtt{E2E}^{1A}$ (standing for end-to-end delay) of the frames sent by the camera that successfully reach the display.

1. Compute $\mathtt{E2E}^{1A}_{max}$, the maximum value of this latency, for a buffer size $\mathtt{n} = 1$.
2. Compute $\mathtt{E2E}^{1A}_{max}$ for a buffer size $\mathtt{n} = 3$.

Upper bounds for $\mathtt{E2E}^{1A}_{max}$ are also of interest.

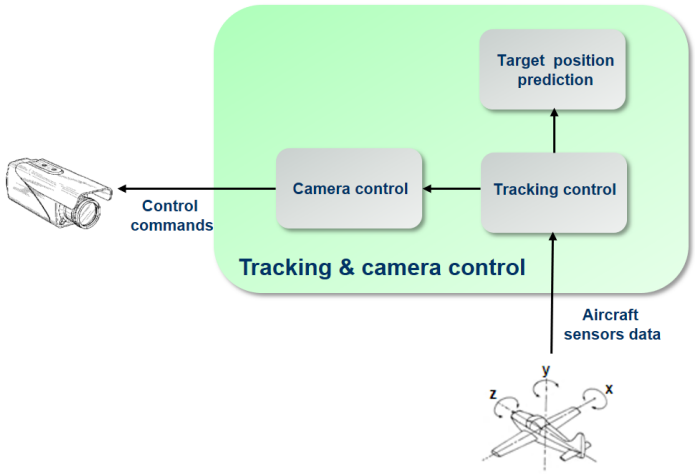

Figure 4 Functional view of the tracking and camera control subsystem.

Challenge 1B

Due to the small size of the buffer read by $\mathtt{T}_4$, it may happen that all frames with identical indices (i.e., all copies originating from the same register content between $\mathtt{T}_2$ and $\mathtt{T}_3$) are discarded at its entrance, e.g., when the period of $\mathtt{T}_4$ is smaller than the period of $\mathtt{T}_1$ (remember that the periods of $\mathtt{T}_1$, $\mathtt{T}_3$ and $\mathtt{T}_4$ are fixed, but their exact values are unknown). That is, no copy of the corresponding frame produced by the camera will ever reach the display. Losing frames is not very critical. However above a certain limit this may have an impact on the video quality and may be detected by the human eye.

The second part of the video frame processing timing verification challenge is therefore to analyze the distance between two frames produced by the camera that will be discarded at the buffer entrance, called `dist`. This distance `dist` may be expressed as a time distance or as a number of frames produced between two successive *losses*:

3. Compute the minimum loss distance $\mathtt{dist}_{min}$ for a buffer size $\mathtt{n} = 1$.
4. Compute $\mathtt{dist}_{min}$ for a buffer size $\mathtt{n} = 3$.

2.2 Challenge 2: Tracking and Camera Control

The functional view of the tracking and camera control subsystem is illustrated in Figure 4. It consists of 3 functions. The `Tracking control` function processes the aircraft sensors data (position, direction, speed, etc.), controls the whole tracking process and generates alerts and various tracking data. The `Target position` prediction function receives data about the aircraft speed, position and direction from the `Tracking control` function and performs motion prediction for the tracked object. The `Camera control` function receives data about the position of the tracked object from the `Tracking control` function and calculates a new angle for the camera based on the aircraft position, speed and direction and the tracked object motion prediction.

For simplicity, we assume that each function is executed by a single task, as illustrated in Figure 5. All tasks are mapped to a same processor GPP1 to which task $\mathtt{T}_2$ (which belongs to the video frame processing subsystem) is also mapped. All tasks are triggered by the arrival of data at their inputs. We assume fixed priority preemptive scheduling on the GPP1 with the following priority order: $\mathtt{T}_2 > \mathtt{T}_6 > \mathtt{T}_5 > \mathtt{T}_7$.

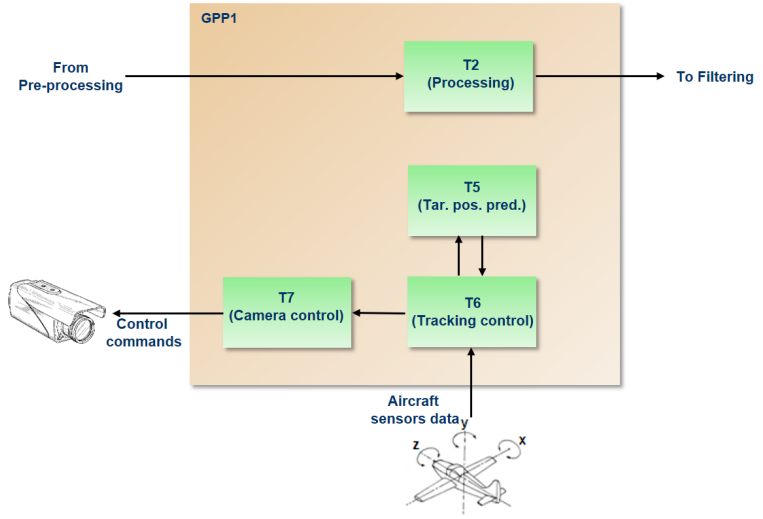

Figure 5 Architectural view of the tracking and camera control subsystem.

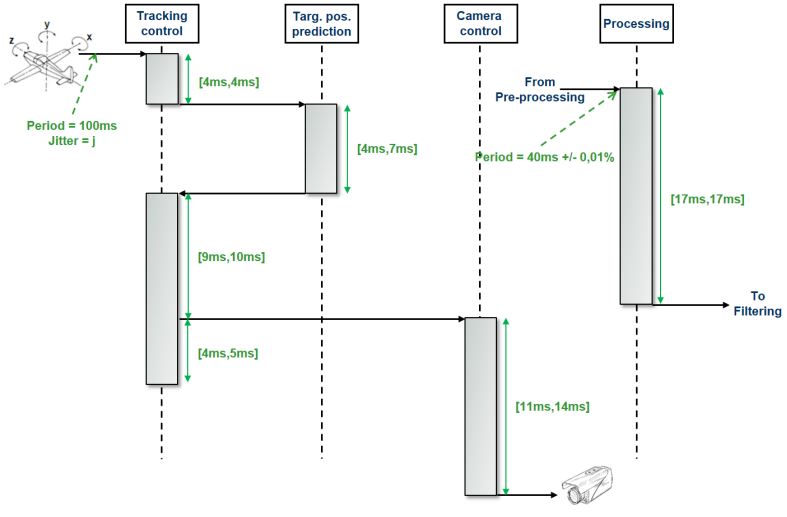

Figure 6 Sequence diagram of the functions on GPP1.

Table 2 Execution times of the individual functions and function segments by the tasks.

Function		Corresponding Execution Time [bcet, wcet] in *ms*
Tracking control	Segment 1	[4, 4]
	Segment 2	[9, 10]
	Segment 3	[4, 5]
Target position prediction		[4, 7]
Camera control		[11, 14]
Processing		[17, 17]

Figure 6 represents the sequence diagram of the functions on GPP1. The `Tracking control` function is activated periodically every 100 *ms*. Its periodic activation can however deviate by a jitter value `jitter`. As in Challenge 1, the `Process` function is activated strictly periodically, i.e., the time distance between two consecutive frames is *constant*. The exact value of the period is however unknown since it may slightly vary from camera to camera. However, we know that it ranges between $40\,ms - 0.01\%$ and $40\,ms + 0.01\%$.

A first segment of the `Tracking control` function is executed by task $\mathtt{T}_6$. Then the `Tracking control` function performs a synchronous call to the `Target position` prediction function and is suspended waiting for the answer. At the end of the `Target position` prediction function, task $\mathtt{T}_6$ resumes executing a second segment of the `Tracking control` function. An asynchronous call is then performed to the `Camera control` function executed by $\mathtt{T}_7$ while the last segment of the `Tracking control` function is executed by $\mathtt{T}_6$. All execution times by the tasks of the individual functions and function segments are given in Table 2.

Challenge 2A

The first part of the tracking and camera control timing verification challenge is to compute the best-case and worst-case end-to-end latencies from the activation of $\mathtt{T}_6$ to the termination of $\mathtt{T}_7$, $\mathtt{E2E}^{2A}_{min}$ and $\mathtt{E2E}^{2A}_{max}$ for different values of `jitter`:

1. Compute $\mathtt{E2E}^{2A}_{min}$ and $\mathtt{E2E}^{2A}_{max}$ for a jitter value $\mathtt{jitter} = 0\,ms$.
2. Compute $\mathtt{E2E}^{2A}_{min}$ and $\mathtt{E2E}^{2A}_{max}$ for a jitter value $\mathtt{jitter} = 20\,ms$.

Challenge 2B

Let us now assume that $\mathtt{T}_2$ and $\mathtt{T}_5$ have access to a shared resource (because the prediction requires information from the image). The resource is mutually exclusive and is protected by a priority ceiling protocol. The access to the shared resource takes 2 *ms* for both tasks. The second part of the tracking and camera control timing verification challenge is again to:

1. Compute $\mathtt{E2E}^{2B}_{min}$ and $\mathtt{E2E}^{2B}_{max}$ for a jitter value $\mathtt{jitter} = 0\,ms$.
2. Compute $\mathtt{E2E}^{2B}_{min}$ and $\mathtt{E2E}^{2B}_{max}$ for a jitter value $\mathtt{jitter} = 20\,ms$.
3. Compute the optimum priority assignment minimizing the worst-case latency `wcrt` for jitter values $\mathtt{jitter} = 0\,ms$ and $\mathtt{jitter} = 20\,ms$.

2.3 Discussion about the challenge

The challenge proposed by Thales is taken from a real industrial application. While the behavior described in the first part of the challenge conforms to a real application (only some execution times were modified), the second part of the challenge was synthesized based on

real timing behaviors encountered in several Thales real-time applications. A model of the application and its real-time behavior based on the Papyrus Modeling Environment and the MARTE profile was made available as a result of the WATERS 2015 workshop [13].

The second part of the challenge represents a classical scheduling verification problem that can be solved even manually by timing verification experts. This eases the evaluation of the quality of proposed solutions. The first part of the challenge, however, is rather untypical and more difficult to solve, thus requiring a tooled solution. In particular, the fact that the exact period for the arrival/sending of the video frames along the processing chain is constant but unknown represents a challenge for existing verification techniques. In addition, while the loss of data is usually not considered in classical scheduling problems, video frames may be discarded in the first part of the challenge depending on the buffer status.

3 Overview of the solutions provided

The proposed solutions fall into three categories: model checking, simulation and scheduling analysis. More specifically, the proposed solutions were based on the following tools:

- the timed model-checker Uppaal (Section 3.1);
- the parametric timed model-checker IMITATOR (Section 3.2);
- the timed model-checking framework Fiacre/Tina (Section 3.3);
- the tool for real-time systems schedulability analysis MAST (Section 3.4); and
- the simulation environment of CPAL (Section 3.5).

None of the proposed solutions could fully address the challenge, and it turned out that there were some misinterpretations of the model description, leading to very different answers for some of the solutions. In the rest of this section, we provide an overview of the various approaches. The objective is not so much to underline each individual contribution (some of which are outdated by now) than to provide a summary of the discussions that this challenge raised, and of the conclusions that were reached.

3.1 Solution using UPPAAL

The Uppaal solution [19] uses timed model checking as the main technique for answering challenge 1. The input formalism is timed automata (Ta) [2] – a dense-time extension of finite-state automata with a set of clocks measuring time – and the software used is Uppaal [14], a tool for the analysis of real-time systems described by a system of communicating timed automata.

A solution to challenge 1 was proposed[2] by a straightforward representation of the task system that almost directly maps to an implementation: each task is represented as a Ta executing a simple time- or event-triggered loop. Frames are represented by indexes counted modulo a sufficient[3] window size $\mathtt{N}_w$. For answering the challenge questions, for each frame `id`, an observer Ta `Thread(id)` is defined, which starts counting time when the frame is created and follows its progress through the tasks until the frame is lost or successfully processed. The variable execution time of task $\mathtt{T}_2$ is expressed by a time interval that is handled symbolically by the verification tool.

[2] Uppaal can also express Challenge 2, but the use of a schedulability analysis tool seemed to us a more obvious choice for that.

[3] The window size defines the number of frames which may be "in the system" simultaneously. In practice, we fix a size $\mathtt{N}_w$, and prove that it is sufficient by verifying that no frame lives longer than the corresponding time window $\mathtt{P}_1 \times \mathtt{N}_w$ (a new frame is created every $\mathtt{P}_1$ time units).

In order to model the assumed uncertainty about the periods of the different tasks, for each period, an integer constant (a "parameter"[4]) is defined, which is instantiated with a set of relevant values. This may be considered as a limitation, as the challenge specification specifies that the periods can take *any* value within some interval. Still, explicitly distinguishing different cases provides some insight: that only cases where $\texttt{P}_1 < \texttt{P}_4$ will lead to losses was not really a surprise, but the fact that even large deviations of $\texttt{P}_3$ (of $\frac{1}{3}$ or more) have only very little influence on the results obtained, was probably less evident. The regularity of the results obtained for decreasing deviations (Table 3 shows results for $\frac{1}{3}$, $\frac{1}{6}$, $\frac{1}{12}$) allows us to extrapolate the results for smaller deviations.

The basic verification with Uppaal can be considered in the present case as a sort of compromise between simulation and parametric verification as proposed by Section 3.2. It provides exact results for given parameter instances, and there was no issue with scaling to compute minimal or maximal `E2E` or `dist` when excluding the first few hundred frames (600 and 1000 frames respectively in the results reported in Table 3). The solution uses the basic model-checking algorithms of Uppaal based on a symbolic clock representation. This allows us to achieve exact verification results for the model under study. Using relevant properties, a set of timings can thus be derived. The simulation capacities of Uppaal were also used but only to get some initial understanding or for debugging. Results were then confirmed by model checking.

3.2 Solution using IMITATOR

The IMITATOR solution [20] uses parametric timed model checking as the main technique to derive the end-to-end timings for answering Challenge 1. The underlying formalism is parametric timed automata [3], an extension of timed automata [2] with unknown timing parameters, in addition to the clocks used in timed automata invariants and guards. The software used is IMITATOR [5], a parametric timed model checker taking as input networks of parametric timed automata augmented with useful features such as rational-valued variables, stopwatches, etc.

The key solution to solve Challenge 1A is to consider a single *arbitrary* frame processing. Thanks to the *symbolic* representation of the state space offered by IMITATOR, the system can start from an arbitrary state, and perform a finite number of discrete actions simulating this arbitrary frame. Measuring the time from its input to the output, IMITATOR therefore derives a (parametric) best and worst case time. Parameters (i.e., unknown constants) are used to model the uncertain periods; an additional parameter $\texttt{E2E} \geq 0$ is also used to represent the end-to-end latency of the target frame. And, for a given run, this value is *unique* as a single frame is considered.

The key aspect of this solution is the use of rational-valued *timing parameters* to model perfectly the unknown (but constant) periods. This solution is correct, as opposed to intervals – in which case the actual period can vary at each cycle, which is not the intended specification in the challenge. The solutions to Challenge 1A (see Table 3) for both buffer sizes are exact, while the proposed solution to Challenge 1B is only approximated, due to the increased system complexity.

In addition, a solution is derived for Challenge 2A by the same authors [20] using analytical methods, and then confirmed by IMITATOR. The extension of the solution to Challenge 2B was not modeled in [20] due to lack of time but seems straightforward to the authors.

[4] Note that this is different from the timing parameters (unknown constants) that will be described in Section 3.2, as they are handled manually here.

Finally, this way of modeling the solution was a source of inspiration for subsequent work on addressing scheduling problems under uncertainty using (extensions of) parametric timed automata [6].

3.3 Solution using Fiacre and Tina

The Tina approach [8] relies on several models to answer questions from Challenge 1, defined using either Time Petri Nets (TPN) [16] or Fiacre [7], a component-based specification language that extends TPN with data and priorities. All our models derive from a common specification, but are each specifically instrumented in order to check a given property: minimal and maximal traversal time; influence of different clock rates; possibility to lose frames; etc.

Our results are computed using the model-checking tool `sift`, part of the Tina toolbox [9], that provides state-space exploration and reachability algorithms for both TPN and Fiacre models. Like with IMITATOR and Uppaal, our approach is based on a dense-time hypothesis. We are not totally faithful to the specification though. In particular, we decided to use time intervals to model the uncertainty on the period instead of a fixed value inside an interval. This means that, inside the same execution trace, the periods of a task may vary. Because of this, each experiment that we perform only returns approximate results. Nonetheless, by using several iterations, and by using together methods that over- or under-approximate the possible behaviors, we were able to compute results within a given accuracy threshold (we use $10\mu s$ in Table 3). It would have been possible to derive an "exact" model using an extension of TPN with stopwatches, but this is way more computationally expensive and not usable in practice.

The frame processing challenge turned out to be a very interesting case study for our model-checking toolbox. First, since the description is highly modular, it is well-suited for component-based modeling languages. Also, it provides a good motivation for the use of high-level data structures in a specification language. In our Fiacre models, for instance, we use a queue of identifiers with a dedicated insertion function to model an unbounded number of frames. As a result, in the case $n = 3$, we can prove that there can be at most 5 different frames traveling at a given time in the pipeline, without the need to provide a bound *a priori*. Finally, many requirements can be reduced to safety properties, that is, checking that some "bad state" cannot occur. In this case, we often do not need to explore the whole state space of the system to return a meaningful result. We can also use more aggressive abstractions, that are able to speed up our computations. We give an example of such optimization in [8] that was able to reduce some model-checking tasks by a factor of $\times 250$ (from about $100s$ to less than $0.4s$).

3.4 Solution using MAST

This approach [15] is based on the Modeling and Analysis Suite for Real-Time Applications (MAST) [12], which is an open source set of tools for developing real-time applications to perform various kinds of schedulability analysis.

For Challenge 1, the authors focused on a different research question than the one posed by Thales, namely finding the worst-case conditions under which the system is still schedulable (i.e., it loses no frame). Under these assumptions, 3 out of the 4 questions can be answered by combining the information directly provided by MAST and ad-hoc external methods. In particular when $n = 3$, results for the required latencies and distances can be calculated as the maximum number of frames enqueued is 2. Furthermore, distance between two frames can be calculated for $n = 1$ based on the MAST method for calculating the buffer size.

The relevant characteristics for modeling Challenge 2 are: Each function is mapped to one task. All tasks are in the same processor using FP preemptive scheduling with priority order: $\mathtt{T}_2 > \mathtt{T}_6 > \mathtt{T}_5 > \mathtt{T}_7$. All tasks are triggered by the arrival of data at their inputs. $\mathtt{T}_7$ is invoked from an action in the middle of $\mathtt{T}_6$. Here the authors need to define the models to explore timing responses and priority assignments for tasks in GPP1 that encompass a fork, activation jitter for one of the two concurrent flows of execution, and shared resources. The approach in this case is to adapt the models used to the sets of equivalent cases for the relative values of priorities and the capacities of the analysis techniques available.

The presence of a fork leads to a multipath model for which only the holistic technique was available [15]. If the fork is decomposed using the values of the priorities to linearize it, then, this linear model is analyzed using offset-based techniques. A newer technique enabling the offset-based analysis of the multipath model has been more recently published in [4].

3.5 Solution using CPAL

This approach [1] is based on CPAL (Cyber Physical Action Language) [10], a language meant to model, simulate, verify and program Cyber-Physical Systems (CPS). CPAL does not provide any fully automatic analysis to compute a solution to the FMTV challenge. However, it helps to identify and validate best and worst-case scenarios. Thus, the language features of CPAL used for solving the challenges are the formal description, the edition, graphical representation and simulation of CPS models. The two challenges as specified in the original description of the FMTV challenge could come out as a little ambiguous: the CPAL model in contrast must be unambiguous and adhere to the well-defined semantics of the execution and simulation environment. Furthermore, the models can be written directly in the graphical CPAL editor, providing an immediate feedback on where and in which aspects the informal problem descriptions were underspecified. Since CPAL is a domain-specific language with native support for real-time scheduling, the modeling effort is small, and the risk of translation errors is limited.

4 Discussion

Let us now discuss briefly the results obtained using the different tools, and provide some overall conclusions. Table 3 shows an overview of the results for Challenge 1, and Table 4 those for Challenge 2. A first outcome is that no tool solved all challenges, even with an approximate solution. Challenge 1A is the one with the highest number of answers, with 4 out of 5 tools being able to make some answers – this challenge also has the highest diversity rate in terms of results (see below). Then, Challenge 1B had 4 out of 5 tools offering a solution, while the other challenges (2A and 2B) had 3 out of 5 tools being able to provide a (partial) solution. A second outcome is that the range of solutions is highly diverse: that is, different tools obtained different answers. This is not entirely surprising, as several tools knowningly analyzed slightly different problems (as detailed in Section 3), leading to completely different answers. Still, a certain (partial) consensus can be obtained by looking closely at the results, as will be discussed later.

[5] $\mathtt{P}_1 = \mathtt{P}_4 = 3 \times \mathtt{P}_3$.
[6] All measured results were below or equal to 146, the true worst-case is thus at least $146ms$.

Table 3 Overview of the results for Challenge 1.

Tool	**1A**: E2E or $E2E_{max}$ *ms*		**1B**: dist or $dist_{max}$ *ms* or frames	
	n = 1	n = 3	n = 1	n = 3
UPPAAL				
no deviation[5]	$[63, 118.33]$ *ms*	$[63, 118.33]$ *ms*	none	none
$P_4 = P_1 - \frac{1}{3}$	$[63, 118]$ *ms*	$[63, 118]$ *ms*	none	none
$P_4 = P_1 + \frac{1}{3}$	$[63, 145.33]$ *ms*	$[63, 226]$ *ms*	$[42, 240]$ frames	$[79, 480]$ frames
after ≥ 600 fr	$[90, 145.33]$ *ms*	$[170.66, 226]$ *ms*	$[79, 161]$ frames	$[79, 161]$ frames
$P_4 = P_1 + \frac{1}{6}$	$[63, 145.16]$ *ms*	$[63, 225.66]$ *ms*	$[84, 480]$ frames	$[159, 958]$ frames
after ≥ 1000 fr	$[89.84, 145.16]$ *ms*	$[162.66, 225.66]$ *ms*	$[159, 321]$ frames	$[159, 321]$ frames
$P_4 = P_1 + \frac{1}{12}$	$[63, 145.08]$ *ms*	–	–	$[319, -]$ frames
IMITATOR	$[63, 145.008]$ *ms*	$[63, 225.016]$ *ms*	–	$< 5,000$ frames
Fiacre/Tina	$[89.66, 145.33]$ *ms*	$[89.66, 225.33]$ *ms*	2 frames	between 35 and 4085 frames
MAST	–	If $P_1 \geq 39.996$ *ms* and $n \geq 2$ $[63, 118.344]$ *ms*	55.344 *ms*	28 *ms* for the highest possible frame production rate of $P_1 = 28$ *ms*
CPAL simulation	≥ 146 *ms* [6]	≥ 220 *ms*	2 frames	4 400 frames
CPAL analysis	$[89.6656, 146]$ *ms* ($E2E_{min}$ is 63 *ms* for the 1st frame)	$[89.6656, 226]$ *ms* ($E2E_{min}$ is 63 *ms* for the 1st frame)	–	–

Table 4 Overview of the results for Challenge 2.

Tool	**2A**: [bcrt,wcrt]		**2B**: [bcrt,wcrt]		
	jitter = 0 *ms*	jitter = 20 *ms*	jitter = 0 *ms*	jitter = 20 *ms*	Optimization
UPPAAL	–	–	–	–	–
IMITATOR	$[49, 74]$ *ms*	$[49, 94]$ *ms*	–	–	–
Fiacre/Tina	–	–	–	–	–
MAST	$[32, 74]$ *ms*	$[32, 94]$ *ms*	$[32, 78]$ *ms*	$[32, 98]$ *ms*	$T_5 > T_7 > T_6 > T_2$; if jitter = 0 *ms* then wcrt = 39 *ms*; if jitter = 20 *ms* then wcrt = 59 *ms*
CPAL simulation	–	–	–	–	–
CPAL analysis	$[33, 75]$ *ms*	$[33, 112]$ *ms*	$[33, 75]$ *ms*	$[33, 112]$ *ms*	$T_7 > T_6 > T_5 > T_2$ (37 *ms*)

4.1 Modeling

One important lesson from this challenge is related to the fact that there were ambiguities in the description, leading to misinterpretations (the same problem was understood and interpreted differently by different participants to the challenge). This shows the importance of providing unambiguous models to avoid misunderstandings. Based on this observation, Thales published a consolidated version of the challenge [13] including a Papyrus model annotated with MARTE.

A second observation related to modeling is that the ease of modeling differed quite a lot between the different tools. Some were quite well adapted to the nature of the challenge while others required quite heavy work to address the problem at hand. This made it difficult to validate the correctness of the proposed models.

4.2 Different solutions

Focusing on the solutions that did model the constant, yet unkown period parameter, a certain (partial) consensus can be obtained on some results.

For Challenge 1A, the lower bound for both $n = 1$ and $n = 3$ is around 89 *ms*, which is obtained by most tools (with some differences due to rounding approximations). The value 63 *ms* found by IMITATOR and UPPAAL corresponds to the first frame, which has a specific behavior.

In addition, 4 out of 5 tools agree that the upper bound is between 145 and 146 *ms* for $n = 1$, and between 225 and 226 *ms* for $n = 3$. Most importantly, the solution obtained by simulation for $n = 1$ (CPAL: 146 *ms*), and that acts as a *lower bound* on the desired maximum, matches the result obtained by parametric model checking (IMITATOR: 145.008 *ms*), which acts as an *upper bound* on the desired maximum. (The fact that the "lower" upper bound found with simulation by CPAL is lower than the upper bound found by model-checking comes from the discrete-time setting of CPAL simulation tool.) That is, the desired maximum is necessarily in $(145, 145.008]$ *ms*.

Concerning Challenge 1B, results are completely different. This challenge was the most difficult according to the participants. The actual value is probably around the result of CPAL, i.e., 4 400 frames, according to the participants, but no conclusion has been reached.

Although it was considered easy according to the participants, results for Challenges 2A and 2B vary quite a lot and more research would be needed to converge towards a consensual answer.

4.3 No perfect tool

The developers of the challenge were hoping to have a single tool being able to solve all sub-challenges immediately. Instead, it turned out that different tools solved parts of the challenge, and that had different strengths and weaknesses: ease of modeling, level of automation of the verification process, computational complexity of verification, reliability of the results, etc.

An additional lesson learned from the challenge is that the use of different techniques to solve the same timing verification problem may increase the confidence in the produced results. For example by applying simulation to an execution scenario identified by scheduling analysis as worst-case, we may determine if the analysis results are too pessimistic or not and thus evaluate the analysis quality. This point is very important in the industrial development of real-time systems since overestimation margins are usually small (over approximation up to 20 % are in general accepted in industry, larger over-approximations are refused due to the cost of the corresponding over dimensioned solutions).

For example, in Challenge 1A, combining the formal methods based result of IMITATOR (a possibly overapproximate result) with the simulation based result of CPAL (a possibly underapproximate result, due to the weakness of simulation) allowed us to confirm the exactness of both methods. Similarly, in [17], two different formalisms (parametric timed Petri nets and parametric timed automata) were used in academic software developed by two different teams in order to solve a problem close to the problem described in the current paper. The fact that both solutions led to the same result increases significantly the confidence we can have in these results. Finally, some tools could potentially be used together in a sequential manner: for example, the results of IMITATOR could be fed into non-parametric tools such as Uppaal or CPAL, so as to first infer a set of possible solutions, and then verify them using non-parametric methods. It remains an open question whether we can easily derive a holistic tool that solves the complete challenge.

5 Conclusion and perspectives

In this paper, we have presented the main features and lessons learned from the first edition of what has now become the ECRTS industrial challenge. This verification challenge, proposed by Thales, was first discussed in 2014 as part of a dedicated workshop (FMTV, a satellite event of the FM 2014 conference), and solutions were discussed for the first time at the WATERS 2015 workshop.

The use case for the verification challenge is an aerial video tracking system. The first part of the challenge focuses on the video frame processing system. It consists in computing maximum values of the end-to-end latency of the frames sent by the camera to the display, for two different buffer sizes, and then the minimum duration between two consecutive frame losses. The second challenge is about computing end-to-end latencies on the tracking and camera control for two different values of jitter. Solutions based on five different tools – Fiacre/Tina, CPAL (simulation and analysis), IMITATOR, Uppaal and MAST – were submitted for discussion at WATERS 2015. While none of these solutions provided a full answer to the challenge, a combination of several of them did allow to draw some conclusions.

From Thales' point of view, the timing verification challenge was a success. The submitted solutions to the challenge represent all three model-based timing verification techniques: timing simulation, scheduling analysis and model checking using timed automata. This gave Thales the opportunity to evaluate these techniques and better understand their strengths and weaknesses. As a direct result of the challenge, point to point collaborations were set-up with several participants. Thales and RTaW set-up a research project (FUI WARUNA) focusing on timing verification and its integration into the industrial design process. The solution using IMITATOR was further evaluated by a trainee at Thales and the results were published in a joint paper [17]. Another joint paper between Thales and UNICAN about the integration of response-time analysis and optimization in the industrial design process was also published [18].

From an academic perspective, we also consider this first edition as a success. This event provided a unique opportunity for researchers to try out their pet tool on a problem of industrial relevance, and to discuss and compare its performance and limitations with colleagues in a collaborative way. Although the challenge was fairly simple in its formulation, it was sufficient to underline the gap that exists between academic tools and industrial real-time systems. Still, the proposed verification problem did strike a difficult balance between practical relevance and feasibility: it was challenging, yet within reach of academic tools. Furthermore, the fact that a combination of solutions could provide a much better

answer to the challenge than any single tool opens up a scientific line of research that has not been explored since: when is it useful to combine different formalisms and analysis techniques to solve a verification problem, and how can this be done in an efficient manner?

This first positive experience triggered a series of subsequent industrial challenges at WATERS, most notably two iterations by Bosch GmbH. The WATERS challenge has now grown to become a full-fledged event of the ECRTS conference.

Let us conclude this paper by a few general comments. We believe that the noncompetitive way in which the challenge was organized contributed a lot to its success. The review process focused on clarifying assumptions and limitations of the proposed solutions and did not intend to declare a winner. In fact, participants to the challenge were invited to review other submissions, considering that they were the best suited for this. This process ensured that everyone could focus on the scientific discussions without the additional burden of organizing or participating in a competition.

More generally, events like this help bringing closer researchers in academia and industrial practitioners from different application domains. For real-time systems research, which tends to measure its success by its practical relevance and transfer to industry, this is obviously very useful. Now, this raises an interesting question, which is rarely discussed by the community: how much research should focus on addressing the needs of industry? A useful expansion of the challenge in future years could experiment with other types of collaborative efforts, e.g., research approaches that would investigate alternative trajectories to those of immediate interest to industry – for example, approaches that would strike a different balance between predictability and complexity – or even focusing on society's needs via another proxy than industry – for example by collaborating with nonprofit organizations. In any case, one can only hope that more use cases will be made available to the community in the future, and that venues for discussing them and potential solutions will spread.

References

1 Sebastian Altmeyer, Loïc Fejoz, and Nicolas Navet. Using CPAL to model and validate the timing behaviour of embedded systems. Solutions to the FMTV Challenge, Informal proceedings of the 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2015), July 2015. URL: `https://nicolas.navet.eu/publi/challenge.pdf`.

2 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994. `doi:10.1016/0304-3975(94)90010-8`.

3 Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *STOC*, pages 592–601, New York, NY, USA, 1993. ACM. `doi:10.1145/167088.167242`.

4 Andoni Amurrio, Ekain Azketa, J. Javier Gutiérrez, Mario Aldea Rivas, and Michael González Harbour. Response-time analysis of multipath flows in hierarchically-scheduled time-partitioned distributed real-time systems. *IEEE Access*, 8:196700–196711, 2020. `doi:10.1007/s10009-017-0467-0`.

5 Étienne André. IMITATOR 3: Synthesis of timing parameters beyond decidability. In Rustan Leino and Alexandra Silva, editors, *CAV*, volume 12759 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2021. `doi:10.1007/978-3-030-81685-8_26`.

6 Étienne André, Emmanuel Coquard, Laurent Fribourg, Jawher Jerray, and David Lesens. Parametric schedulability analysis of a launcher flight control system under reactivity constraints. *Fundamenta Informaticae*, 182(1):31–67, September 2021. `doi:10.3233/FI-2021-2065`.

7 Bernard Berthomieu, Jean-Paul Bodeveix, Patrick Farail, Mamoun Filali, Hubert Garavel, Pierre Gauffilet, Frédéric Lang, and François Vernadat. Fiacre: an intermediate language for model verification in the topcased environment. In *Embedded Real Time Software (ERTS)*, 2008.

8 Bernard Berthomieu, Silvano Dal Zilio, and Didier Le Botlan. Latency analysis of an aerial video tracking system using Fiacre and Tina. Solutions to the FMTV Challenge, Informal proceedings of the 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2015), July 2015. URL: `https://www.ecrts.org/forum/download/FMTV15_Solution_Fiacre_Tina.pdf`.

9 Bernard Berthomieu and François Vernadat. Time Petri nets analysis with TINA. In *QEST*, pages 123–124. IEEE Computer Society, 2006. `doi:10.1109/QEST.2006.56`.

10 CPAL : A Cyber-Physical Action Language. URL: `https://www.designcps.com/model-based-design-with-cpal/`.

11 Formal Methods for Timing Verification (FMTV) Workshop, a satellite event of FM'14, the 19th International Symposium on Formal Methods. URL: `https://www.comp.nus.edu.sg/~pat/FM2014/`.

12 Michael González Harbour, J. Javier Gutiérrez García, José Carlos Palencia Gutiérrez, and J. M. Drake Moyano. MAST: Modeling and analysis suite for real time applications. In *ECRTS*, pages 125–134. IEEE Computer Society, 2001. `doi:10.1109/EMRTS.2001.934015`.

13 Rafik Henia. Consolidated version of the WATERS industrial challenge by Thales including a corresponding model in Papyrus. URL: `https://www.ecrts.org/forum/viewtopic26ef.html?f=34&t=86&sid=d74079af129d5480a5ac4fd1778eecc1`.

14 Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2):134–152, 1997. `doi:10.1007/s100090050010`.

15 Julio L. Medina, Juan M. Rivas, J. Javier Gutiérrez, and Michael González Harbour. Solving the 2015 FMTV challenge using response time analysis with MAST. Solutions to the FMTV Challenge, Informal proceedings of the 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2015), July 2015. URL: `https://www.ecrts.org/forum/download/FMTV15_Solution_MAST.pdf`.

16 Philip Meir Merlin. *A study of the recoverability of computing systems*. PhD thesis, University of California, Irvine, CA, USA, 1974.

17 Baptiste Parquier, Laurent Rioux, Rafik Henia, Romain Soulat, Olivier H. Roux, Didier Lime, and Étienne André. Applying parametric model-checking techniques for reusing real-time critical systems. In Cyrille Artho and Peter Csaba Ölveczky, editors, *FTSCS*, volume 694 of *Communications in Computer and Information Science*, pages 129–144. Springer, November 2016. `doi:10.1007/978-3-319-53946-1_8`.

18 Juan Maria Rivas, J. Javier Gutiérrez, Mario Aldea Rivas, César Cuevas, Michael González Harbour, José María Drake, Julio L. Medina, Laurent Rioux, Rafik Henia, and Nicolas Sordon. An experience integrating response-time analysis and optimization with an MDE strategy. In Paolo Milazzo, Dániel Varró, and Manuel Wimmer, editors, *MELO @ STAF*, volume 9946 of *Lecture Notes in Computer Science*, pages 303–316. Springer, 2016. `doi:10.1007/978-3-319-50230-4_23`.

19 Lijun Shan and Susanne Graf. Timing verification of an aerial video tracking system using UPPAAL. Solutions to the FMTV Challenge, Informal proceedings of the 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2015), July 2015. URL: `https://www.ecrts.org/forum/download/FMTV15_Solution_UPPAAL.pdf`.

20 Youcheng Sun, Étienne André, and Giuseppe Lipari. Verification of two real-time systems using parametric timed automata. Solutions to the FMTV Challenge, Informal proceedings of the 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2015), July 2015. `https://lipn.univ-paris13.fr/ andre/documents/verification-of-two-real-time-systems-using-parametric-timed-automata.pdf`.

21 Youcheng Sun, Étienne André, and Giuseppe Lipari. Verification of an aerial video system. Solutions to the consolidated 2015 industrial Challenge, Informal proceedings of the 8th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2017), July 2017. URL: https://www.ecrts.org/forum/download/FMTV17_submitted.pdf.

22 Website of the WATERS'15 workshop. URL: http://waters2015.inria.fr/.