The final publication is available at ACM via http://dx.doi.org/10.1145/3341105.3373871

CleanET: Enabling Timing Validation for Complex Automotive Systems

Sergi Vilardell[‡,†], Isabel Serra[†,⋆], Hamid Tabani[†], Jaume Abella[†], Joan Del Castillo[Ψ], Francisco J. Cazorla [†]
[‡]Universitat Politecnica de Catalunya (UPC), Spain
[†]Barcelona Supercomputing Center (BSC), Spain
[⋆]Centre de Recerca Matematica (CRM), Universitat Autonoma de Barcelona (UAB), Spain
[Ψ]Departament de Matematiques, Universitat Autonoma de Barcelona (UAB), Spain

***Abstract*—Timing validation for automotive systems occurs in late integration stages when it is hard to control how the instances of software tasks overlap in time. To make things worse, in complex software systems, like those for autonomous driving, tasks schedule has a strong event-driven nature, which further complicates relating those task-overlapping scenarios (TOS) captured during the software timing budgeting and those observed during validation phases. This paper proposes CleanET, an approach to derive the dilation factor r caused due to the simultaneous execution of multiple tasks. To that end, CleanET builds on the captured TOS during testing and predicts how tasks execution time react under untested TOS (e.g. full overlap), hence acting as a mean of robust testing. CleanET also provides additional evidence for certification about the derived timing budgets for every task. We apply CleanET to a commercial autonomous driving framework, Apollo, where task measurements can only be reasonably collected under 'arbitrary' TOS. Our results show that CleanET successfully derives the dilation factor and allows assessing whether execution times for the different tasks adhere to their respective deadlines for unobserved scenarios.**

I. Introduction

The software timing process for critical real-time systems starts in early design stages by deriving budgets for the execution time of tasks, i.e. estimating the Worst-Case Execution Time (WCET) of tasks [1], [2], along with task schedules. Both ensure that all software functions execute timely. As the complexity of critical systems increases, deriving estimates under the worst-case conditions that can arise during system operation becomes increasingly complex [2]. This is compounded by the fact that, in early stages, the system is not even fully integrated. During late design stages, as part of the incremental integration of the different subsystems, validation[1] tests are performed to assess whether the timing of the system still adheres to its specifications [3].

Software timing validation is more relevant as software gets integrated on the hardware platform under a close-to-final setup. It builds on *observability* to retrieve relevant and abundant timing information from the platform and *controllability* to exercise software under potentially-feasible operation conditions (e.g. with Simulink environments on a host computer). How the jobs of a task overlap with jobs (i.e. tasks activations) running on other cores plays a key role in software timing, as it has been shown that contention in the access to hardware shared resources has huge impact in tasks execution time [4], [5], [6]. Controlling task overlapping scenarios (TOS) is challenging in simple systems when a dynamic-priority scheduling is used (e.g. EDF). As hardware and software platforms for critical systems increase their complexity, it is even harder to achieve controlability on TOS as many tasks are not scheduled according to any well-know scheduling scheme for real-time systems, or they are simply event driven.

Tests during the validation phase measure timing in relevant scenarios as stipulated in safety guidelines for safety-critical real-time systems. A solid validation process based on qualified tools like RapiTime [7] helps achieving sufficient[2] coverage of relevant execution scenarios. The absence of any (timing) violation serves as means of validation of the correctness of the derived software timing budgets.

Tasks overlapping heavily impacts individual tasks' execution time and is an aspect of multicore-based systems for which no satisfactory solution exists yet. Given a reference analysis task, the number of other tasks (and the particular tasks) that overlap with it changes its execution time. Likewise the degree of overlap, i.e. how long tasks overlap, affects also its timing behavior. Task overlap varies in non-obvious manners across individual measurements and also during operation. Therefore, measurements are subject to hard-to-control contention, which brings uncertainty to the validation process. While the user might have means to observe the variability among tasks in observed TOS during tests, it is hard to enforce specific TOS.

We make the following contributions to tackle this challenge:

Contribution 1: We propose *CleanET*, that stands for *clean execution times*, an approach to distill both, contention-free measurements as well as measurements subject to specific contention levels, to enable a reliable timing validation process that captures any TOS that might occur during operation.

[1]Note that verification often refers to the evaluation of whether requirements and specifications are met by the items composing the system during the design phase, whereas validation occurs a posteriori once the system has been designed, in the different integration stages, building intensively on test campaigns. However, in some domains, such as avionics, these terms can be used interchanged.

[2]In the context of automotive systems, whether a set of tests suffices or not relates to the coverage of relevant execution scenarios, as defined by highly qualified engineers, which enumerate and detail those scenarios so that the risk of occurrence of execution conditions different to those of the identified scenarios can be deemed as residual.

CleanET builds upon *statistical dependence analysis* to derive the dilation factor r affecting execution times due to simultaneous task execution. CleanET resamples the execution times measured during testing to derive tasks execution time under (1) no overlapping scenarios (i.e. in isolation), (2) full single overlapping (i.e. when the task overlaps 100% of the time with exactly one other task), and (3) data with any specific overlap representing the worst overlap of interest (e.g. full overlap with N other tasks). CleanET estimates provide additional means of validation for derived time budgets and alleviates the pressure on end users to produce tests cases with different TOS.

Contribution 2: We apply CleanET on a commercial Autonomous Driving (AD) framework, Apollo [8], where the different modules (tasks) of the framework overlap partially and in an 'arbitrary' manner, thus exposing the challenge that CleanET targets. Apollo is an intricate set of tightly-coupled software modules, including an Operating System, which jeopardizes the possibility of collecting measurements for each module under controlled TOS. CleanET successfully distills the dilation factor and resamples measurements to any TOS of interest.

Contribution 3: We illustrate the use of resampled measurements obtained with CleanET by estimating the execution times for specific Apollo modules for different overlap scenarios. Our results show that, in this particular case, we can account for the impact of multiple tasks fully overlapping with the one under analysis, thus being able to validate scenarios not triggered by the available test cases. Our results also show that accounting for full overlap with 2 tasks instead of 1, produces a negligible increase of the execution times for this particular AD framework. Furthermore, we show that, with the input data available for Apollo, we can also estimate the impact of the overlap of 3 other tasks, but confidence is lower due to the low frequency of measurements capturing this scenario. In any case, CleanET is not constrained to the number of tasks overlapping with the one under analysis, and allows reasoning on the impact of overlapping and the confidence of the values obtained.

The rest of the paper is organized as follows: Section II provides a brief background on the usual validation process for automotive systems. Section III analyzes the execution time measurements from the Apollo autonomous driving software. Section IV presents our approach to distill noise-free execution times. In Section V, we present our evaluation of CleanET on Apollo. Section VI reviews the related work and, finally, Section VII sums up our conclusions.

II. Background

This section provides some background on the usual validation process for some safety-related real-time systems, in particular those that mostly rely on measurements (e.g., automotive, railway). We use automotive terminology for the sake of consistency. We also introduce some details about the structure of the Apollo AD framework and provide some background on solutions to account for interference in high-performance systems for safety-critical applications with particular emphasis on measurement-based practices.

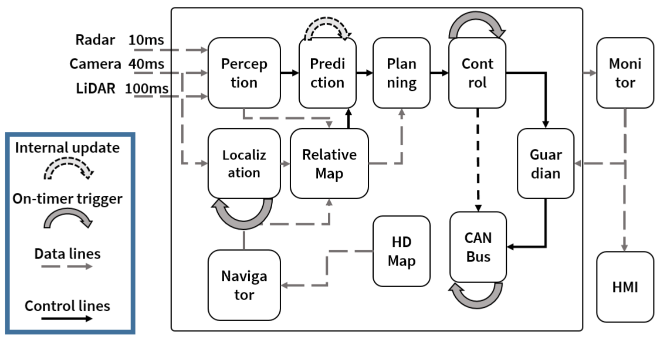

Fig. 1: Apollo 3.0 software architecture pipeline.

A. Apollo Autonomous Driving Framework

AD systems drive the vehicle towards a specified destination as safely and efficiently as possible. To this end, the AD system includes complex combinations of various input sensors such as cameras, short-range and long-range radars and LiDARs to scan the surrounding area around the car and track the moving objects. The system features sophisticated navigation schemes to locate the position of the vehicle with centimeter-level precision. Based on this information, the system plans the future paths, predicts the trajectory of moving objects, and controls the vehicle to follow the specified paths. These are the main stages of almost all practical state-of-the-art AD systems [8], [9], [10]. Figure 1 shows the main modules and architecture of a representative AD system, Apollo 3.0. The main modules are the following:
(1) *Perception* module identifies the obstacles in the surrounding of the autonomous vehicle.
(2) *Prediction* anticipates the future motion trajectories of perceived obstacles/objects.
(3) *Localization* leverages information received from different sensors to estimate the location of the vehicle precisely.
(4) *Navigator (Routing)* module tells the vehicle how to reach the specified destination.
(5) *Planning* plans the spatio-temporal trajectory of the vehicle.
(6) *Control* generates control commands such as accelerating, braking and steering based on the outcome of these modules.
(7) *CANBus* passes all the control commands to the vehicle hardware and also provides some information back to the AD system.
(8) *HD Map* module is a library that provides detailed structured information about the roads.
(9) *HMI* (Human Machine Interface) is a module for viewing the status and controlling the functions of the vehicle in real-time.
(10) *Monitor* is a surveillance system to check all the software and hardware modules.
(11) *Guardian* is a safety module responsible to intervene whenever *Monitor* detects a failure.

Inter-Module Functional Dependencies: The dependencies between different modules of Apollo can be triggered in three different ways:

- Periodically, based on a timer (On-Timer).

- Once a module produces data to be served by its successor or successors.
- Whenever a module receives a *request* message from another module. Accordingly, a *response* message should be computed and published.

As shown in Figure 1, each module may be triggered by either two or just one of these ways.

- *Perception* receives sensor input data such as Radar data, LiDAR point-cloud data, and camera data. Based on these inputs, this module detects objects of interest and traffic lights and tracks them inside consecutive frames.
- *Localization* module has two modes: a RTK-based[3] mode with a timer-based function (On-Timer), and Multiple Sensor Fusion (MSF).
- *Prediction* receives output data of both *Perception* and *Localization*. It updates its internal status whenever it receives a *Localization* update. However, the main *prediction* function is triggered once *Perception* publishes an output message.
- *Navigator* module is triggered whenever a routing request is received.
- *Planning*, *Control* and *CANBus* modules are all triggered periodically and also when they receive *Prediction* output, human commands and *Control* commands respectively.

In Figure 1, control lines, On-Timer triggers, and inter-module updates are shown with different arrow types.

B. Timing Validation in Automotive Systems

The need for deriving time budgets for safety-related software components in automotive emanates from safety requirements, as described in ISO 26262 automotive functional safety standard [11]. Safety requirements determine the maximum response time affordable for a given functionality and the fault tolerant time interval (FTTI), which stands for the time allowed since a fault occurs until an action is taken to recover or to bring the system to a safe state. Both determine the end-to-end time for a given functionality to be performed (including sensing and actuation time). Once discounted the time devoted to interfacing with physical components, the remaining time is the time budget allocated for the software component to complete its execution.

Automotive systems are designed and verified following appropriate practices to achieve safety compliance. For instance, WCET estimation tools may be used, together with appropriate response time analysis methods for task scheduling. Nevertheless, a validation step is needed before deploying the system to detect system faults due to, for instance, the violation of some assumptions caused during integration. Whether an application adheres to its timing constraints during the validation process is normally assessed in hardware-in-the-loop testing environments, where the complete application can be run. Appropriate test equipment is used to collect information on the execution of the application under analysis in general, and each of its tasks in particular, thus with a much higher observability than in the system during operation. However, due to the high coupling between the different tasks, the operating system, and the input/output interface of the application, individual tasks cannot be run in isolation or at externally-controlled time instants. Hence, although start and end times can be obtained for the different jobs of each task (observability), it is not possible to enforce specific release times at will (controllability). Overall, tasks overlap with each other in an intricate manner depending on release times dictated by input data and timers, as well as by their duration.

Execution time measurements, therefore, reflect arbitrary-looking overlaps across tasks, representative for the input data used during tests. Whether other overlaps are possible and whether those can lead to higher execution times is, in general, unknown and hard to control in practice. Hence, other than considering an engineering margin on top of the high watermark execution time to account for scenarios not triggered by the tests used, end users lack tools to use those execution time measurements in a more informed manner for validation purposes. While engineering margins set based on experience have worked for hardware platforms with low execution time variability, the increasing use of GPUs and other high-performance hardware to match the performance needs of AD frameworks brings much higher performance variability due to contention in the use of shared resources (e.g. shared caches and main memory bandwidth). Hence, end users lack means to quantify whether unobserved scenarios could lead to timing violations.

C. Accounting for Contention on WCET

Multiple approaches exist to account for contention due to concurrency on the access to shared hardware resources, as part of the design and verification process of safety-related real-time systems. While it is not the purpose of this paper surveying on this topic, we identify the main families of techniques. Some techniques control the impact of contention by building upon some hardware and/or software support [5], [12], [4]. Other techniques use such support to completely avoid any impact due to contention [13], [14], [15], [16], [17]. Finally, some other approaches, rather than controlling or mitigating contention, aim at upper-bounding it [18], [19].

Several measurement-based WCET estimation techniques rely on (1) collecting measurements fully representative of the operation conditions (or the worst-case operation conditions), or (2) collecting measurements of the task in isolation. In the former case, measurements already account for contention, and they can be used to feed appropriate tools for timing analysis [7], [20], [21], [22], [23]. In the latter case, either execution time measurements or WCET estimates are inflated with an upper-bound to the impact of contention resulting in a contention-aware WCET estimate / response time [6], [24], [25].

While each of those techniques has its assumptions and requirements and has been proven appropriate for WCET estimation as part of the verification process, a validation step is still needed during system integration phases, in accordance with safety-related systems development processes.

[3]Real-time kinematic (RTK) positioning is a satellite navigation technique used to enhance the precision of position data derived from satellite-based positioning systems (global navigation satellite systems, GNSS).

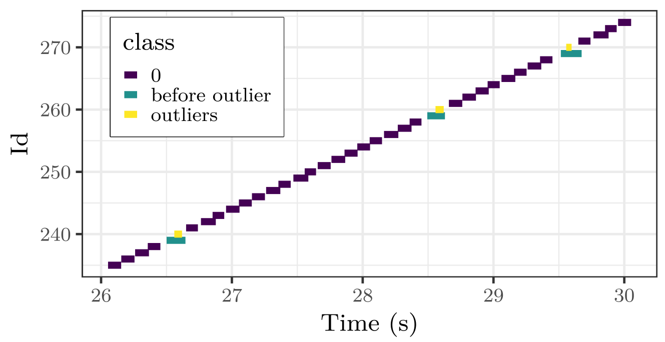

Fig. 2: Timeline interval of Prediction showing the behaviour of the outliers.

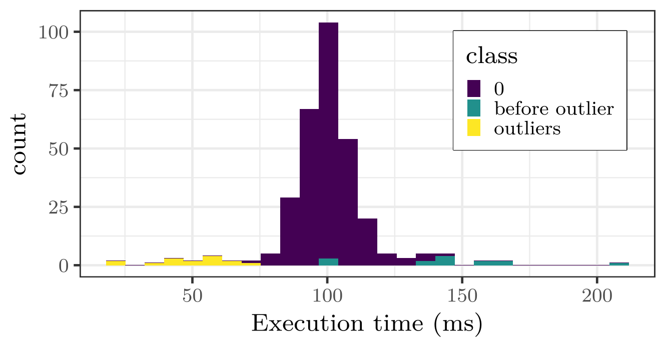

Fig. 3: Execution time histograms

In this context, performing a reliable timing validation process building on measurements with arbitrary overlap across tasks is a difficult challenge. This paper addresses this challenge as presented next.

III. Apollo Characterization

A. Experimental Setup

We target Apollo [8] AD framework, as the largest existing AD project with more than 120 partners including top-tier AI and tech companies, and car manufacturers. Since our target are real automotive systems, we have ported it to the NVIDIA's latest automotive platform, Xavier SoC[4], which is integrated in the Jetson AGX Xavier [27] platform. The Xavier SoC consists of an Octa-core ARM-based CPU, a 512-core NVIDIA Volta GPU, and several accelerators such as specialized deep learning accelerators (NVDLA) to meet the needs of automotive systems. From all Apollo modules, we focus on those with most intensive computation requirements, which are the ones using the GPU: Prediction and Perception. Those modules create mutual interference (contention), and hence, challenge timing validation. Both modules use the GPU in a similar manner since both build upon Deep Neural Networks (DNN) and Recurrent Neural Networks (RNN). Both call the same matrix multiplication primitives with similar parameters. Therefore, whenever they attempt to use the GPU simultaneously, they are expected to create significant interference on each other. In our analysis, for simplicity, we model how the execution time of the Prediction module is affected by the overlapping with the execution of Perception jobs. The symmetrical analysis can also be performed, but we omit it due to lack of space and since it does not offer further insights.

B. Prediction Module Timing Behavior

From our analysis of the prediction module's execution times we have realized that the vast majority of executions occur in a non-overlapping manner between each other with few exceptions. In particular, periodically a Prediction job starts with an unusual delay w.r.t. the finalization time of the previous job; then said job takes longer execution time than expected. Sometimes, before a long job finishes, a new Prediction job is released, but it is cancelled soon after its start since the previous one is still running. Therefore, sporadically (exactly once every 10 jobs at most), we have two jobs with anomalous timing behavior. This is illustrated in Figure 2, where we show an excerpt of the chronogram for the Prediction module. As shown, anomalies appear at time instants 26.5, 28.5 and 29.5 seconds, shown in green color (abnormally high execution times) and yellow color (abnormally low execution times after abnormally high ones). We, therefore, classify execution times into three classes: *normal* corresponds to the normal Prediction behavior, *before outlier* class corresponds to overly large execution times, and the *outliers* class corresponds to the outliers that start before the previous job has finished.

Figure 3 shows the histograms of each class with the same color code as for previous figure. The particular execution time mean (in milliseconds) for each class is 100, 144, and 50.3 respectively. Hence, outliers have shorter execution times, whereas the execution times before the outliers are particularly high. Finally, *normal* concentrates most jobs in a relatively narrow execution time interval.

Based on this evidence, we conclude that execution times for each class need to be treated separately. Moreover, we build on the common practice in safety-critical real-time automotive and robotics systems, among others, where some jobs can fail (e.g. due to an overrun) as long as the number of failures in the last M jobs does not exceed a particular threshold, N [28], thus building on top of the Typical Worst-Case Response Time (TWCRT) rather than on the absolute Worst-Case Response Time (WCRT). In our case, by simply studying *normal* jobs, we discard up to 2 jobs every 10, whose timing behavior, if it violates any deadline, would not be a problem as long as 2 out of 10 runs are allowed to fail, thus much in line with automotive and robotics common practice.

C. Aggregation of Overlaps

Once filtered the Prediction jobs of interest, we study their overlap with Perception jobs. Our analysis reveals that each Prediction job can overlap, during different parts of its execution, with up to 3 Perception jobs. The fraction of

[4]Xavier is the largest SoC ever built with more than 7 billion transistors and it is the main SoC integrated in Jetson AGX Xavier, NVIDIA Drive Xavier, and NVIDIA Drive Pegasus, all targeting the computation demand for the highest levels of automation (e.g., levels 4 and 5 according to SAE International [26]).

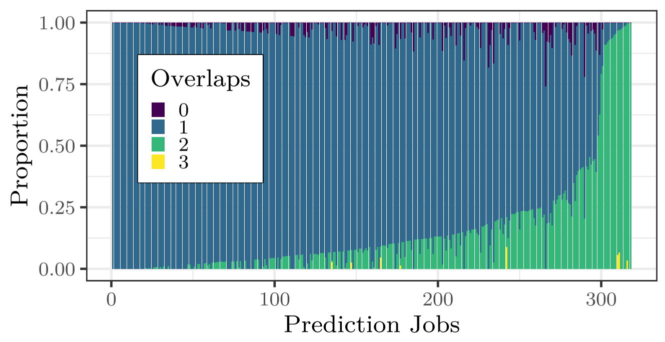

Fig. 4: Portion of normalized execution times that overlaps with different number of Perceptions simultaneously.

time overlapped with 0, 1, 2, or 3 Perception jobs for each of the Prediction jobs is depicted in Figure 4. Jobs have been sorted from highest to lowest fraction of overlap with 1 Perception job to ease graphical representation. As shown, most Prediction jobs spend most of their execution time overlapped with exactly 1 Perception job. The fraction of time overlapped with 2 Perception jobs is also significant. Little time is spent without overlapping in general. Finally, only sporadically some Prediction jobs overlap with 3 Perception jobs.

IV. CleanET

A. Setting the Objective

Complex SoCs like the NVIDIA Xavier have a large internal hardware and software (e.g. CUDA drivers) state. Modelling this state, which strongly influences tasks' execution time, is quite difficult. In such a complex system, the execution time of a task can be regarded as a random variable, J. When several tasks execute simultaneously, they affect each other timing behavior due to contention in the access to shared resources, even if tasks have no data or control dependence among them. The impact of the contention interference depends on (i) the number of tasks with overlapping execution; (ii) the percentage of time they overlap; (iii) the particular part of their code that overlaps; and (iv) the inputs to the program that might affect tasks access pattern to shared resources. In this work, we focus on the first two factors, and hence do not model the particular section of the code of a task that overlaps, nor tasks inputs.

Our objective is to enable the estimation of the execution time of tasks under different TOS. In particular, by modelling the dependence of the execution time on the TOS (i.e. how much overlapping and with how many tasks), we enable the reliable estimation of the dilation factors, r_1, …, r_m, impacting execution time due to the task overlapping with 1, …, m tasks respectively. Hence, r_i is the factor by which the execution time with no overlap of the task under analysis must be multiplied when overlapping with i tasks. For instance, if $r_i = 1.5$, the non-overlapping execution time of the task under analysis is 10ms, and it overlaps with i tasks during 4ms, then the execution time is $6 + 4 \cdot 1.5 = 12$ms.

Once dilation factors r_i are obtained, and given that we know the start and end time for all jobs – and so their particular overlappings, we can deduce execution time measurements under any TOS (i.e. those regarded relevant but that could not be tested) since we can obtain the pristine (non-overlapped) execution time measurements removing the impact of contention in the measurements collected, and apply dilation factors to model any TOS of interest for validation purposes.

The derived dilation factors can be used to validate the timing behavior of the task under analysis by comparing them against the corresponding bounds from the verification phase, e.g. under worst overlapping conditions. They can also be used to contrast the output of timing analysis techniques that predict WCET estimates under different TOS. For instance, since execution times under a given (homogeneous) overlap correspond to random variables independent and with identical distribution, then we could use Measurement-Based Probabilistic Timing Analysis (MBPTA) techniques [20], [21], [22], [23] for that purpose.

It is worth noting that, unlike previous works studying dependent execution times [29], [30], [31], which focus on stationary processes and hence, on dependencies across measurements, our work considers external dependencies that are unlikely to be periodic and that cannot be studied only based on the actual execution time measurements of the jobs of the task under analysis, but accounting for the execution times (more precisely, for the start and end times) of the jobs of the other tasks running simultaneously in the system. Stationary processes could be normally studied on top of the measurements obtained dilating them with our approach to match worst-case overlaps during operation. Thus, our work is orthogonal and complementary to previous works since they consider different types of dependencies (i.e. we target dependencies on the TOS).

B. Modelling Approach

CleanET assumes a baseline (basal) state without any impact of dilation that corresponds to the execution in isolation with no overlap with any other task. In practice the basal state may never occur in the system as it may be not possible to run the task under analysis without overlapping its execution with others. We denote the hardware/software state information in this basal state as $\mathcal{I}_0$.

CleanET builds around the mathematical expectation E[5]. Let $\mathcal{I}$ be the available information on the state of the system for any arbitrary state, and $\mathrm{E}(T|\mathcal{I})$ the expected execution time T assuming that the system is in a given (observed) state $\mathcal{I}$. The dilation coefficient, r is defined as follows, thus relating the execution time of the basal state with that of any other state:

$$\mathrm{E}(T|\mathcal{I}) = r\mathrm{E}(T|\mathcal{I}_0) \tag{1}$$

Whenever the system is not in the basal state, then the expectation of the execution time, $\mathrm{E}(T|\mathcal{I})$, is dilated by the coefficient r, where in general $r > 1$ (i.e. execution overlapping normally leads to increased execution times). In

[5]The mathematical expectation is given by $E[X] = \sum_{i=1}^{n} x_i p_i$, where x is a random variable with the probability function, $f(x)$, p is the probability of occurrence, and n is the number of all possible values, or, informally, the mean of an infinitely large sample.

fact, Equation 1 holds for any state where the amount of overlapping in $\mathcal{I}$ is higher than in $\mathcal{I}_0$, thus meaning that any TOS necessarily results in an increase of tasks execution time.

The basal state of the system (i.e. the execution in isolation) cannot be observed in general. Otherwise, by observing such state and any other state, we could trivially derive r. Instead, we build upon the measurements collected with arbitrary TOS from a real system where no practical control can be exercised to enforce specific TOS, as it is the case, for instance, for most automotive systems. Given a finite sample of execution times and variables characterizing the state of the system (i.e. how jobs overlap), we want to estimate a basal state of the system as a random variable, as well as the dilation coefficient r. For the sake of this discussion, we consider that jobs either overlap or do not overlap, neglecting whether they overlap with one or more jobs. Later we extend the discussion to arbitrary numbers of overlapping jobs.

C. *Mathematical Development of CleanET*

The most simple way for describing the state of a system along the execution period T is by summarizing the information of its state. This can be done with only one property through a binary measure for each instant of time, t of a fixed duration (e.g. 1 processor clock cycle in the extreme), see Equation 2 where t is the particular time instant when we assess the property. In this work the particular property is whether the execution of the particular job of the task under analysis overlaps with the execution of any other job.

$$W_t = \begin{cases} 1 & \text{if the property holds,} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

We can summarize this process through the total time during which the property holds, i.e. $W_t = 1$, denoted by V. The remaining time, $T - V$, is denoted by U. This approach approximates the basal state of the system by $\mathcal{I}_0 = (V = 0)$ and Equation 1, so that:

$$\mathrm{E}(T|V = v) = r(v)\mathrm{E}(T|V = 0) \tag{3}$$

In Equation 3, $\mathrm{E}(T|V = 0)$ stands for the expectation $(E(\cdot))$ of the execution time (T) when there is no overlap at all $(V = 0)$, $r(v)$ is the dilation factor given a specific overlap v, and therefore, $\mathrm{E}(T|V = v)$ is the expectation of the execution time given the specific overlap v.

Likewise, let Q be the proportion of the execution time, T, where the property holds (V/T) and $1 - Q$ where it does not (U/T). This approach approximates the basal state of the system by $E(T|Q = 0)$ and Equation 1, so that:

$$\mathrm{E}(T|Q = q) = r(q)\mathrm{E}(T|Q = 0) \tag{4}$$

Both Equation 3 and 4, allow using regression methods for parameter estimation [32]. In particular, we want to obtain the dilation coefficient r, which we obtain as $r(1)$, so when $q = 1$ (full overlap).

D. *Obtaining r*

Let $X = (T|Q = 0) = (T|V = 0)$ be the execution time of the task when the state of the system is basal. Note that $X = (T|V = 0)$ is the actual random variable, which may realize into any specific execution time, instead of $E(T|V = 0)$, the expectation, that can only be the Expectation, thus a single value (i.e. the mean of an infinitely large sample). Let $Y = (T|V) = (T|Q)$ be the observed execution time, thus with V time of overlapping (or Q fraction of overlapping). If an observation of Y is given (the actual execution time measured), then V is fixed to v, and Q fixed to q, for some v and q. Note that v and q are directly obtained from observation of the system since we have the start and end time for each job of each task, and hence, we can determine whether the specific job of the task under analysis overlaps its execution with any other job at any time instant.

We can decompose the observed execution time, Y, as follows:

$$Y = U + V = (1 - q)Y + qY \tag{5}$$

$(1-q)Y$ is the part of the execution time of the job in basal state (no overlapping). Hence, we can define $(1 - p)$ such that $(1 - q)Y = (1 - p)X$, thus relating the execution time with overlapping (Y) with the execution time in isolation (X). From this equivalence, we can describe the dilation coefficient r as follows:

$$Y = U + V = (1 - q)Y + qY = (1 - p)X + rpX \tag{6}$$

where the rightmost part of the equation decomposes the measured execution time into $(1 - p)X$, so the fraction of non-overlapping execution $(1-p)$ multiplied by the basal state execution time (X), and rpX, so the fraction of overlapping execution (p) multiplied by the dilation factor (r) and the basal state execution time.

Equivalently, we can apply the distributive property:

$$Y = ((1 - p) + rp)X \tag{7}$$

Moreover, since $U + \frac{V}{r} = X$, then $U = X - \frac{V}{r}$, and we can obtain a linear expression:

$$Y = U + V = X - \frac{V}{r} + V,$$
$$Y = \frac{r-1}{r}V + X \tag{8}$$

Then, we can apply linear regression to obtain r and X in Equation 8 since Y and V are known.

Note that the basal state corresponds to $V = 0$, so $Y = X$. Also note that if all execution time is overlapped with other jobs (so $V = Y$), we would have the following:

$$Y = \frac{r-1}{r}Y + X \tag{9}$$

We could transform it into the following expression:

$$Y\left(1 - \frac{r-1}{r}\right) = X \tag{10}$$

Thus having that $\frac{Y}{r} = X$, and $Y = rX$, which matches Equation 7 when $p = 1$ (overlapping fraction).

Based on the assumptions of the model, X and Q are independent, which is equivalent to X and P being independent. Then, it follows that, in general, given X and P being non-observed independent random variables, and U and V observed random variables such that

$$U = (1 - P)X \quad \text{and} \quad V = rPX$$

for some coefficient $r > 0$, then a point estimation for r, P and X is given by

$$\hat{X} = U + V/\hat{r} \tag{11}$$

$$\hat{P} = V/(\hat{r}U + V) \tag{12}$$

$$\hat{r} = min_r \Big\{ cov(\hat{X}, \hat{P}) \Big\} \tag{13}$$

However, point estimations are not satisfactory from a safety perspective due to the uncertainty caused by the potential error in the estimation of execution times. Therefore, we propose a pessimistic method that overestimates the coefficient r, which we detail next.

E. Intrinsic Pessimism

Equation 8, states the linear relationship between Y and V. On the other hand, the linear regression of Y and V provides us with (α, β, Z) such that $\alpha, \beta \in \mathbb{R}$, and Z is a random variable that maximizes the independence between Y and V such that $Y = \alpha + \beta V + Z$. Given the definition of β in simple linear regression [32] and Equation 8, we can yield the following expression:

$$\beta = \frac{\text{cov}(Y, V)}{\text{var}(V)} = \frac{\text{cov}(\frac{r-1}{r}V + X, V)}{\text{var}(V)} = \frac{r-1}{r} + \frac{\text{cov}(X, V)}{\text{var}(V)} \tag{14}$$

Therefore, the equivalence $\beta = \frac{r-1}{r}$ would hold if X and V were independent. Given that this is not the case, the relationship between r and β yields:

$$\beta > \frac{r-1}{r} \tag{15}$$

since $cov(X, V) > 0$, which can be proven from the independence of X and P. Note that the covariation between X and V is generally unknown. Thus, by ignoring it, we overestimate the factor $\frac{r-1}{r}$, which, given that $r > 1$ as indicated before, implies that we overestimate r.

The consequence of overestimating r is that, by considering scenarios where the overlapping is higher than measured, the overestimated r leads to higher predicted execution times than those in the real system. In our case, this implies that, if those predicted measurements respect timing budgets, then real system measurements would necessarily also respect the budgets. Also, if the system can overrun the timing budget, predicted measurements obtained with CleanET will indicate even higher overruns, so faults will be reliably detected. However, in some cases, time budgets may be respected and CleanET report that they are violated. Hence, while this may cost some tightness by imposing the allocation of larger time budgets than needed, our approach does not challenge the safety of the system tested.

F. CleanET for Multiple Overlappings

In the previous section we focused the case where one specific job either does not overlap or partially overlap with another job. The model can be easily extended to account for several jobs overlapping at the same time (from 1 to m). This is done by defining Equation 2 for each number of overlaps, so having W_t^1,..., W_t^m, where the property in each case is whether there are *exactly* i overlaps. For instance, for $i = 2$, $W_t^2 = 1$ if and only if the job overlaps with exactly 2 other jobs in time t. For any other number of overlapping jobs (e.g. 0, 1, 3, etc), $W_t^2 = 0$. Hence, V_i stands for the total time where property W_t^i holds.

We can, therefore, extend our original definition of Y ($Y = U + V = (1 - q)Y + qY$) decomposing V and q across the different types of overlapping (from 1 to m overlapping jobs) as follows:

$$Y = U + \sum_{i=1}^{m} V_i = \left(1 - \sum_{i=1}^{m} q_i\right) Y + \sum_{i=1}^{m} q_i Y \tag{16}$$

This allows us to reformulate Equation 7 as follows:

$$Y = \left(\left(1 - \sum_{i=1}^{m} p_i\right) + \sum_{i=1}^{m} r_i p_i \right) X \tag{17}$$

which can be transformed into the following equation where linear regression can be directly applied:

$$Y = \sum_{i=1}^{m} \left(\frac{r_i - 1}{r_i} V_i \right) + X \tag{18}$$

where Y and all V_i are known, and X and all r_i dilation factors are obtained through linear regression.

V. Evaluation

For the evaluation of CleanET, we first estimate r_i dilation factors for the different task overlaps. Then, we validate how dilation factor estimation matches expectations. Finally, we show how those dilation factors can be used to estimate execution time bounds for different overlapping scenarios, thus allowing to validate whether time budgets are violated.

A. Dilation Factors (r_i)

1) Single dilation factor: We have applied CleanET to obtain the dilation factors, r_i, for the different numbers of overlapped tasks. Results are shown in Table I considering a single dilation factor (whether overlap exists or not), as per Section IV-D. As shown, execution time with no overlap at all is around 16.32ms, with a standard deviation of 1.64ms. However, the impact of overlap is huge since $r = 6.85$ with a standard deviation of 0.78. Hence, on average, we could expect the execution time will full overlap to be around 112ms ($16.32ms \cdot 6.85$). For the sake of completeness, the table also provides the number of execution time observations used (289), which are those for *class 0* before, and the degree of correlation measured between non-overlapped and overlapped execution time measurements, which is, as expected, very high (0.9).

TABLE I: CleanET applied to filtered data with Overlap as co-variate.

Overlap (r)	6.85 ± 0.78
Constant ($p = 0$)	16.32 ± 1.64
Observations	289
Adjusted R-squared (correlation):	0.90

2) All dilation factors: As a second step, we have applied CleanET to obtain individual dilation factors for 1, 2 and 3 jobs overlapping with the jobs of the task under analysis, as per Section IV-F. Results are shown below in Table II. We observe that results for 1 or 2 jobs overlapping are very similar and ranges (e.g. $\mu \pm \sigma$) overlap almost completely. Statistically, we cannot prove that they are distinguishable and, therefore, we conclude that overlapping with 1 or 2 jobs must be considered together, thus defining that the property in Equation 2 holds if the task overlaps with exactly 1 or 2 jobs. In the case of 3 jobs overlapping, we observe that (1) the value of r_3 is far lower than that for r_1 and r_2, which would mean that overlapping with 3 jobs leads to a lower dilation factor (so a lower execution time increase) than overlapping with 1 or 2 jobs, which is against intuition. However, the real problem with 3 overlaps relates to the fact that, out of the 289 *class 0* measurements, only 9 have 3 jobs overlapping with the task under analysis for some time, which is, in practice, too scarce data to raise any reliable prediction. In fact, the (very high) standard deviation for this case already indicates this behavior indirectly. The number of measurements including data for each overlap case is provided in Table III for completeness. Finally, note that, while the case with no overlap ("Constant" in the tables) has no meaningful change w.r.t. the case where we fit only r, it is not absolutely identical. This relates to the fact that all parameters in Equation 18 are fit together, thus dealing to minor variations when varying the parameters to fit.

TABLE II: Linear model applied on filtered data with Overlap time of one, two and three processes at a time.

Overlap 1 (r_1)	6.97 ± 0.81
Overlap 2 (r_2)	6.55 ± 0.81
Overlap 3 (r_3)	3.23 ± 2.42
Constant ($p = 0$)	16.32 ± 1.65
Observations	289
Adjusted R-squared (correlation):	0.90

3) Statistically significant dilation factors: After concluding that 1 and 2 overlapping measurements are not statistically distinguishable, we apply CleanET again considering only two dilation factors: r_{1-2} for 1 or 2 overlaps, and r_3 for 3 overlaps, as shown in Table IV below. We note that r_{1-2} is not distinguishable in practice with r in Table I when considering just one dilation factor since the amount of data for 3 overlaps is too little to cause meaningful differences. We also note that, by considering 1 and 2 overlaps together instead of separated, the case for 3 overlaps varies drastically, which reflects the weakness of the fit for 3 overlaps due to the too little data available for this case.

TABLE III: Number of *class 0* measurements with part of its execution with 0, 1, 2, 3 jobs overlapping.

Number of overlapping jobs	Measurements	Percentage (w.r.t. 289)
Overlap 0	137	47.4%
Overlap 1	288	99.7%
Overlap 2	218	75.4%
Overlap 3	9	3.1%

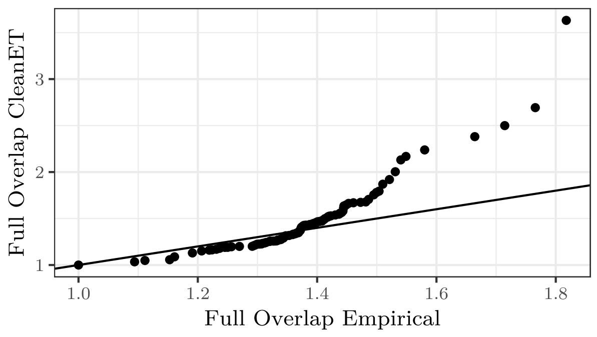

Fig. 5: QQ-plot of full overlap resampled data with CleanET w.r.t. empirical data with full overlap.

TABLE IV: Linear model applied on filtered data with Overlap time of one and two processes added, and three processes.

Overlap 1-2 (r_{1-2})	6.88 ± 0.78
Overlap 3 (r_3)	2.75 ± 1.68
Constant ($p = 0$)	16.31 ± 1.64
Observations	289
Adjusted R-squared (correlation):	0.90

Overall, our results show that scenarios with 0, 1 or 2 overlaps can be reliably resampled from the data available. If the case with 3 overlaps is regarded as relevant for operation conditions, then additional input data would be required including many more measurements corresponding to that scenario, so that CleanET could deliver a reliable dilation factor for this case.

B. Validation of the Method

In general, these solutions cannot be validated due to the lack of reference data, which in this case would correspond to measurements with the complete execution time with a constant number of jobs overlapping (e.g. 100% of the execution time overlapping with exactly 1 job). However, in our case, we have measurements with exactly 1 job overlapping during their complete execution. In fact, since 1 and 2 overlappings have been shown to be indistinguishable, and 3 overlappings occur seldom (too occasionally to be statistically significant), we consider as reference measurements those in which some overlapping (with either 1, 2 or 3 jobs) occurs during the whole execution. Hence, we performed the following:

1) Split the set of measurements into 2 categories: one with those measurements with 100% overlap (OVL1 group, 152 values), and those with non-full overlap (OVLmix, the remaining 137 values).
2) Apply CleanET on OVLmix data.

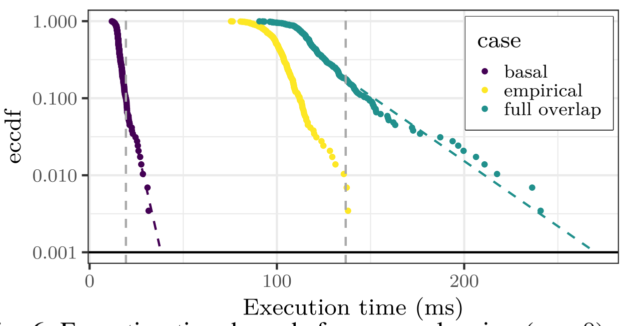

Fig. 6: Execution time bounds for no overlapping ($p = 0$) and full overlapping ($p = 1$).

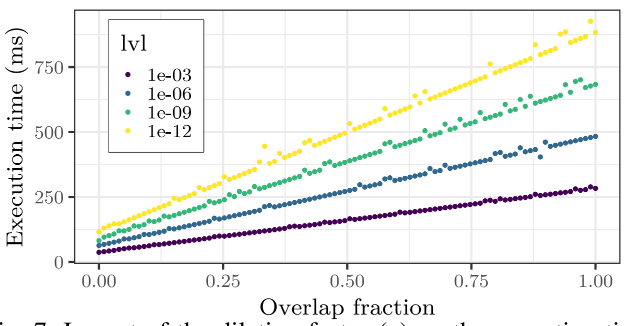

Fig. 7: Impact of the dilation factor (r) on the execution time bounds as we vary the degree of overlapping (p).

3) Compare the results obtained with CleanET resampling data in OVLmix to the case with full overlap, with the actual data with those overlapping characteristics (OVL1 data) with a QQ-plot (see Figure 5).

In the figure, we would like to have a linear relation between the empirical data and the data resampled with CleanET (straight line). However, the fact that r is overestimated leads to non-linearity. Results show that CleanET, using OVLmix data, produces higher values than those observed empirically (OVL1), thus corroborating our expectation of having overestimated execution times for high overlaps due to the overestimation of r. This supports empirically our expectations with a reference data set different to that used by CleanET by partitioning the data into OVL1 and OVLmix groups.

C. *Using CleanET Results*

We have used CleanET to generate, from the actual measurements obtained from the system with arbitrary overlaps, measurements corresponding to overlapping scenarios of interest. In particular, since the purpose of timing validation is assessing that no overrun occurs, we have considered the case with full overlap. For the sake of illustration, we also show results for the case of no-overlap.

Figure 6 plots the empirical complementary cumulative density function (ECCDF) for the three scenarios: actual data measured (empirical), data obtained with CleanET with no overlap (basal), and data obtained with CleanET with full overlap. Together with the empirical data, we fit an exponential tail to the highest values of the extreme cases (basal and full overlap), which has been recommended as a suitable approach to predict high execution times [22], [23]. For that purpose, we build on peaks-over-threshold methods selecting the threshold as indicated in [33].

Measurements resampled with CleanET allow considering cases that could not be explicitly tested in the system under test. As shown, as expected, the full overlap case leads to execution times higher than those of the empirical (measured) case, whereas the basal case, instead, leads to the lowest execution times one would expect in the real system. Note that, as discussed in Section IV-E, r values are overestimated. Hence, the full overlap case is an overestimation of what would be in practice the behavior of the real system with full overlap, whose timing behavior would be somewhere in the region between the empirical measurements for arbitrary overlaps and the estimated full-overlap measurements obtained with CleanET. Note also that overestimating r makes that, whenever we consider lower overlaps than in practice (e.g. basal case), execution times obtained are in practice *lowerbounds* of the true basal case. In any case, the basal case is irrelevant to validate whether time budgets allocated suffice.

For the sake of completeness, we provide in Figure 7 how execution times expected vary as we vary the degree of overlap (p). We use peak-over-threshold again, and we collect the values obtained at different exceedance levels (lvl), from 10^{-3} to 10^{-12}. As expected, a linear increase of the degree of overlap, leads to nearly-linear increase of the execution times expected, with small disturbances due to the uncertainty associated to threshold selection in the fitting process of a distribution (an exponential distribution in our case). For instance, based on the results obtained, if we consider an exceedance level of 10^{-6}, we would conclude that execution times could be slightly below 500ms (assuming full overlap being the worst case during operation). Thus, validation tests would be passed if such value is lower than the deadline for the task under analysis.

Overall, CleanET allows exploring any arbitrary degree of overlap of interest, thus providing end users with means to validate their system against scenarios that cannot be triggered during system testing, thus relieving end users from having to create additional test cases with the hope of those test cases to produce the execution scenarios that need being considered.

VI. Related Work

The impact of dependencies in WCET estimation has been mostly considered for the particular case where they exist across jobs of a given task. In particular, probabilistic WCET (pWCET) estimation was originally formulated on top of the assumption of independent and identically distributed (i.i.d.) execution time observations for a task [34]. Therefore, solutions for pWCET estimation have often built on top of Extreme Value Theory for i.i.d. processes [35].

However, some researchers noted that execution time dependencies may exist across jobs of a given task, thus jeopardizing i.i.d. properties, so methods considering those dependencies would be more convenient [36], [37]. Amongst those works, Bernat et al. [38] pioneered in the area of execution time dependencies and analyzed them at the scope of program components.

Santinelli et al. [39] considered a particular type of dependencies across jobs – stationary processes – and concluded that they could lead to WCET underestimation if not accounted for carefully. Similarly, Melani et al. [29] showed that appropriate statistical tests (mostly correlation and independence tests) can be used to account for those dependencies satisfactorily. Lima and Bate [40] proposed a solution to mitigate the impact of dependencies across jobs to facilitate WCET estimation. Finally, Abella et al. [41] have recently shown that the source of those dependencies across jobs imposes different constraints on task scheduling.

However, to the best of our knowledge, CleanET is the first solution to assess how the dependencies across different tasks interfere each other's execution time, and how this information can be used to obtain execution time measurements either free of interference or subject to specific levels of interference (e.g. full overlap) so that timing budgets used for scheduling purposes can be validated.

VII. Conclusions

Validating execution time bounds for safety-related real-time systems becomes increasingly difficult due to the increasing complexity of those systems driven by a higher system automation. This is particularly true in automotive systems with the advent of autonomous driving. While end users can retrieve plenty of data of the execution of their systems during validation phases, execution cannot be controlled as desired and hence, execution time measurements are representative for some scenarios that may underrepresent operation conditions.

This paper proposes CleanET, a novel solution to obtain measurements representative of any relevant scenario from the set of measurements obtained during analysis. In particular, CleanET builds upon estimating the impact of execution time interference across tasks to allow resampling data into any degree of interference (i.e. execution overlap) of interest. CleanET is a purely measurement-based solution, thus in line with the requirements of automotive end users, that facilitates testing execution scenarios that cannot be practically produced by testing engineers. Our results on a commercial AD framework, Apollo, corroborate the advantages of CleanET for timing validation.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (MINECO) under grant TIN2015-65316-P, the SuPerCom European Research Council (ERC) project under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 772773), and the HiPEAC Network of Excellence. MINECO partially supported Jaume Abella under Ramon y Cajal postdoctoral fellowship (RYC-2013-14717).

References

[1] R. Wilhelm et al., "The worst-case execution-time problem&mdash;overview of methods and survey of tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, pp. 36:1–36:53, May 2008.

[2] J. Abella, C. Hernandez, E. Quiñones, F. J. Cazorla, P. R. Conmy, M. Azkarate-askasua, J. Perez, E. Mezzetti, and T. Vardanega, "Wcet analysis methods: Pitfalls and challenges on their trustworthiness," in *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, June 2015, pp. 1–10.

[3] M. D. Natale, J. Abella, J. Reineke, A. Hamann, and G. Farrall, "Predictable system timing – probab(ilistical)ly?" in *DAC (panel in automotive track)*, 2016.

[4] J. Nowotsch, M. Paulitsch, D. Bühler, H. Theiling, S. Wegener, and M. Schmidt, "Multi-core interference-sensitive WCET analysis leveraging runtime resource capacity enforcement," in *26th Euromicro Conference on Real-Time Systems*, July 2014, pp. 109–118.

[5] H. Kim, D. de Niz, B. Andersson, M. Klein, O. Mutlu, and R. Rajkumar, "Bounding memory interference delay in cots-based multi-core systems," in *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April, pp. 145–154.

[6] J. N. Jan and M. Paulitsch, "Leveraging multi-core computing architectures in avionics," in *EDCC*, 2012.

[7] RapiTime, *www.rapitasystems.com*, 2008.

[8] "Apollo, an open autonomous driving platform." http://apollo.auto/, 2018.

[9] "Udacity. An Open Source Self-Driving Car." https://github.com/udacity/self-driving-car/, 2017.

[10] "Autoware. An open autonomous driving platform." https://github.com/CPFL/Autoware/, 2016.

[11] International Organization for Standardization, *ISO/DIS 26262. Road Vehicles – Functional Safety*, 2009.

[12] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha, "Memguard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms," in *19th IEEE Real-Time and Embedded Technology and Applications Symposium, (RTAS)*, April 2013, pp. 55–64.

[13] R. Pellizzoni, B. D. Bui, M. Caccamo, and L. Sha, "Coscheduling of CPU and I/O Transactions in COTS-Based Embedded Systems," in *Real-Time Systems Symposium (RTSS)*, Nov 2008, pp. 221–231.

[14] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo, and R. Kegley, "A predictable execution model for COTS-based embedded systems," in *17th IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2011, pp. 269–279.

[15] A. Alhammad, S. Wasly, and R. Pellizzoni, "Memory efficient global scheduling of real-time tasks," in *21st IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2015, pp. 285–296.

[16] M. Becker, D. Dasari, B. Nicolic, B. Akesson, V. Nelis, and T. Nolte, "Contention-free execution of automotive applications on a clustered many-core platform," in *28th Euromicro Conference on Real-Time Systems (ECRTS)*, July 2016, pp. 14–24.

[17] A. Biondi and M. D. Natale, "Achieving predictable multicore execution of automotive applications using the LET paradigm," in *24th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2018.

[18] A. Schranzhofer, R. Pellizzoni, J. J. Chen, L. Thiele, and M. Caccamo, "Worst-case response time analysis of resource access models in multi-core systems," in *Design Automation Conference*, June 2010, pp. 332–337.

[19] D. Dasari and V. Nelis, "An Analysis of the Impact of Bus Contention on the WCET in Multicores," in *IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, ser. HPCC '12, 2012, pp. 1450–1457.

[20] L. Santinelli, F. Guet, and J. Morio, "Revising measurement-based probabilistic timing analysis," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2017, pp. 199–208.

[21] F. Guet, L. Santinelli, and J. Morio, "On the Reliability of the Probabilistic Worst-Case Execution Time Estimates," in *Embedded Real-time Software and Systems (ERTS^2) Conference*, 2016.

[22] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quiñones, and F. J. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs," in *2012 24th Euromicro Conference on Real-Time Systems*, July 2012, pp. 91–101.

[23] J. Abella, M. Padilla, J. D. Castillo, and F. Cazorla, "Measurement-based worst-case execution time estimation using the coefficient of variation," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 4, Jun. 2017.

[24] E. Díaz, E. Mezzetti, L. Kosmidis, J. Abella, and F. J. Cazorla, "Modelling multicore contention on the aurixtm tc27x," in *Proceedings of the 55th Annual Design Automation Conference (DAC)*, 2018.

[25] J. Jalle, M. Fernandez, J. Abella, J. Andersson, M. Patte, L. Fossati, M. Zulianello, and F. J. Cazorla, "Bounding resource-contention interference in the next-generation multipurpose processor (ngmp)," in *Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS^2)*, 2016.

[26] SAE International, *SAE J3016. Levels of Driving Automation*, 2014.

[27] D. Shapiro, "Introducing xavier, the nvidia ai supercomputer for the future of autonomous transportation," *NVIDIA blog*, 2016. [Online]. Available: https://blogs.nvidia.com/blog/2016/09/28/xavier/

[28] D. Ziegenbein and A. Hamann, "Timing-aware control software design for automotive systems," in *Proceedings of the 52Nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: ACM, 2015, pp. 56:1–56:6.

[29] A. Melani, E. Noulard, and L. Santinelli, "Learning from probabilities: Dependences within real-time systems," in *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, Sept 2013, pp. 1–8.

[30] K. Berezovskyi, F. Guet, L. Santinelli, K. Bletsas, and E. Tovar, "Measurement-based probabilistic timing analysis for graphics processor units," in *Proceedings of the 29th International Conference on Architecture of Computing Systems – ARCS 2016 - Volume 9637*. New York, NY, USA: Springer-Verlag New York, Inc., 2016, pp. 223–236.

[31] L. Santinelli, F. Guet, and J. Morio, "Revising measurement-based probabilistic timing analysis," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2017, pp. 199–208.

[32] X. Yan and X. G. Su, *Linear Regression Analysis: Theory and Computing*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2009, ch. 2.

[33] J. C. et al., "Methods to distinguish between polynomial and exponential tails," *Scandinavian Journal of Statistics*, vol. 41, no. 2, pp. 382–393, 2014. [Online]. Available: http://dx.doi.org/10.1111/sjos.12037

[34] S. Edgar and A. Burns, "Statistical analysis of wcet for scheduling," in *Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001) (Cat. No.01PR1420)*, Dec 2001, pp. 215–224.

[35] R. Fisher and L. Tippett, "Limiting forms of the frequency distribution of the largest or smallest member of a sample," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 24, no. 2, 1928.

[36] S. Coles, *An Introduction to Statistical Modeling of Extreme Values*. Springer, 2001.

[37] S. Kotz and S. Nadarajah, *Extreme value distributions: theory and applications*. World Scientific, 2000.

[38] G. Bernat, A. Colin, and S. M. Petters, "WCET analysis of probabilistic hard real-time system," in *Real-Time Systems Symposium RTSS*, 2002.

[39] L. Santinelli, J. Morio, G. Dufour, and D. Jacquemart, "On the Sustainability of the Extreme Value Theory for WCET Estimation," in *14th International Workshop on Worst-Case Execution Time Analysis*, vol. 39, 2014.

[40] G. Lima and I. Bate, "Valid application of evt in timing analysis by randomising execution time measurements," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2017, pp. 187–198.

[41] J. Abella, E. Mezzetti, and F. Cazorla, "On assessing the viability of probabilistic scheduling with dependent tasks," in *Symposium On Applied Computing (SAC)*, 2019.