STARTREC: Verification of a safety-critical system for autonomous vehicles

Marwan Wehaiba El Khazen, Slim Ben Amor, Liliana Cucu-Grosjean, Arnaud Dumérat, Xavier Jean, Kossivi Kougblenou, Benjamin Monate

STARTREC: Verification of a safety-critical system for autonomous vehicles

Marwan Wehaiba El Khazen
Inria and Statinf, Paris
marwan.wehaiba-el-khazen@inria.fr

Slim Ben Amor
Statinf, Paris
slim.ben-amor@statinf.fr

Liliana Cucu-Grosjean
Inria and Statinf, Paris
liliana.cucu@inria.fr

Arnaud Dumérat
EasyMile, Toulouse
arnaud.dumerat@easymile.com

Xavier Jean
EasyMile, Toulouse
xavier.jean@easymile.com

Kossivi Kougblenou
Statinf, Paris
kos@statinf.fr

Benjamin Monate
Trust-in-Soft, Paris
benjamin.monate@trust-in-soft.com

***Abstract*—In this paper, we present our on-going work on verification activities of the software used in a safety-critical embedded system dedicated to autonomous vehicles. These activities are focused on the use of formal methods for the verification of functional properties on the embedded code, and statistical methods for the analysis of its Worst-Case Execution Time (WCET). The project's goal is to address some technical barriers of software verification that will impact the safety demonstration of future autonomous driving systems. These barriers are challenging because of the high complexity of an embedded hardware and software, and appeal for methods and tools reaching the highest level of rigorousness.**

***Index Terms*—autonomous vehicles, formal verification, timing verification**

I. Introduction

The safety of autonomous vehicles is one of the numerous challenges this industry will face in the decade to come. In the latest years, the way to tackle this problem has become consensual [10]. An applicant needs first to establish an Operational Design Domain for its use case, summarizing the assumptions of use, hazard analysis and validation strategy at the system scale. This analysis is then refined to the subsystem level, assigning to each component a target integrity level (ASIL) associated to a safety function. The safety demonstration against this function at the appropriate integrity level is conducted under functional safety norms, such as the ISO 26262 [11]. The STARTREC project contributes to this second phase, by addressing the validation of a software component against the ISO 26262.

Even if its functional scope is clearly defined, the verification of a piece of software implementing an autonomous driving function constitutes a major issue with regard to many aspects [2]. It covers topics such as the correctness of algorithms [8], the correct implementation of these algorithms as well as their infrastructure software, being an operating system, a middle ware or third-party libraries, and finally the validation of their real-time properties when deployed on embedded hardware [1].

The component under verification in the STARTREC project is a safety relevant algorithm taking emergency decisions with regard to the environment, the sanity of the autonomous vehicle and its navigation stack. As part of the safe design process, this algorithm is designed to be explainable and verifiable in a white-box approach. It takes as inputs the perception infrastructure including the sensors of the vehicle, and various data produced by the navigation stack. It is interfaced with a safety relevant system capable of executing emergency maneuvers.

The activities planned during the STARTREC project are centered around the three following topics:

- Verify the correctness of the implementation of the algorithm embedded in the system. The correctness of the algorithm itself does not fall in the scope of the project. Related activities are centered around the use of sound static analysis methods and their combination with unit tests and integration tests. Sound methods can prove the absence of some categories of bugs [15], which makes them highly suitable for high integrity systems, but hard to deploy in practice on large pieces of software.
- Evaluate the Worst Case Execution Time (WCET) of the embedded software implementing this algorithm. Related activities are centered around statistical and probabilistic WCET evaluation methods [5].
- Demonstrate the compliance of the tools involved in the verification activities with regard to the ISO 26262 requirements, at the appropriate *Tool Confidence Level* (TCL). This compliance allows the applicant to consider the results provided by the tools as trustworthy.

This paper summarizes the overall problem, details the position of the project's stakeholders on the activities in progress, and presents preliminary results on the probabilistic WCET estimation tool.

II. Problem Statement and Related Work

The STARTREC project addresses the functional verification of the embedded software with the use of sound static analyzers, namely frama-c [14], and TIS-Analyzer [25]. These tools are used to prove (i) that the software is free from runtime errors, and (ii) that it complies with its functional specification. One challenge to address is the seamless integration of these tools with standard verification and validation processes:

radical changes to traditional processes are costly in terms of project management and risky in term of certification [19].

Runtime errors cover a large class of software misbehavior. Such errors encountered in embedded software are typically divisions by 0, memory overflows, out-of-bounds accesses, violations of flow integrity, typically dangling pointers or uninitialized variables. The numeric stability of floating point operations is also under consideration. Runtime errors are classically tackled by defensive programming techniques, but this raises a problem of software testing. Indeed, defensive programming introduce a systematic verification of each condition that could trigger a runtime error. This makes the test activity painful, as every branch shall be tested. For each condition a test case shall be specified and developed according to the quality insurance process. The proof of runtime errors absence, presented by Moy et al. [18] as the *"silver level"* of formal verification, allows to reduce the need of defensive programming by removing unnecessary checks at runtime. The method used in the project is abstract interpretation. It can be deployed on large pieces of code with a minimal effort of code annotation, but the analyzer's accuracy trends to decreases while progressing in the software analysis. Therefore, the main challenge is to analyze the algorithm while providing a reasonable set of annotations to keep a sufficient accuracy.

The proof of functional correctness, presented by Moy et al. [18] as the *"gold level"* of formal verification, relies on deductive verification. Frama-c implements this method in the Wp plugin. However, its use demands a high amount of code annotations, and reaches the limits of automatic provers on various aspects, like the handling of floating point numbers or the memory model associated with the representation of pointers. Although the STARTREC project will support the improvement of tools in these areas, proving the functional correctness of the whole software is considered not achievable in the frame of the project. A more pragmatic approach combining unit tests and unit proofs [14] is preferred, with the use of executable annotations.

The second theme of STARTREC is the evaluation of the software's Worst Case Execution Time (WCET). This activity is complicated for the following reasons:

- The software has many inputs impacting the execution time. Typically, sensors generate point clouds whose size and disparity varies with the environment, the meteorological conditions, and the sunlight. The scenario leading to the WCET of the algorithm is far from being reproducible on a real scene, so that the complexity of these inputs remains empirical. It is necessary to understand, build and justify the diversity of the situations encountered in the vehicle's environment to gather a set of recordings sufficient to conduct a meaningful WCET analysis. Note that every situation at a vehicle level does not need to be considered as long as it is demonstrated that they are equivalent to other situations, from a WCET point of view.
- The embedded software is to be deployed on a multi-core processor, able to execute several tasks in parallel. This leads to interference in the underlying hardware, when several cores compete to access shared resources, typically the memories, with complex patterns [3], [21]. At a high-level, this can be observed as a slow-down on every task. Estimating an upper bound for the interference is known to be an open problem in general case, with pathological situations that can be considered as denial-of-service attacks [17], the slow-down factor can be an order of magnitude [20]. We tackle this problem with the following restrictions: (i) the whole stack of embedded software is known, (ii) no piece of software has been designed to maximize the interference, and (iii) the scheduling of tasks might be reconsidered to limit the impact of interference.
- For WCET analyses and interference analyses, identifying the potential sources of interference as well as the size of the benchmarks set is also complex [12]. One typical problem of interference analyses is justifying why their results are trustworthy and provide certification credit.

Statistical methods deployed in the project, as described thereafter, are focused on the detection of inputs that impact the execution time, so that they do not need to be considered independently in the benchmark set. A similar approach is developed for interference analysis on multi-core microprocessors.

Any measurement protocol of the execution time of a program should take into account the fact that the actual WCET might not, or rather almost certainly will not be achieved. Rather, a set of highly unlikely but observed "extreme" worst-case values should be considered to represent the general behavior of all worst-case values. Then, assuming an appropriate theoretical foundation whose hypotheses would be checked, a statistical estimation of the WCET can be produced, along with an appropriately small probability that the actual WCET is higher than that estimation. Extreme Value Theory (EVT) provides such a probabilistic framework, much needed for the estimation of the statistical WCET. Its central result, the extreme value theorem, was given by Gnedenko [9] and describes the possible distributions of the extreme values. With a finite set of observed extreme values, one can then estimate the parameters of the whole distribution of extremes and give probabilistic guarantees pertaining to execution times much larger than the largest observation. The use of probabilistic performance guarantees and of distributions of execution times was originally suggested for scheduling by Tia et al. [22]. Later, these results have been consolidated and expanded to different aspects of real-time systems beyond scheduling by an important thread of results [5]. By using EVT, the authors provide estimations of the distribution bounding a sequence of measurements, when such bound exists. However, previous work [16] underlines that there are two associated problems while applying EVT to the statistical WCET estimation problem. The first one concerns the measurement protocol and whether or not it produces measurements that are representative of the actual distribution of the execution times

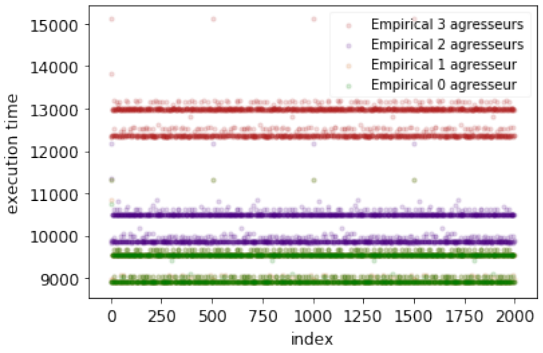

Fig. 1. The execution time sequence of the *minver* program on T1040

of a program. The second problem increases the tightness of the statistical WCET estimation while of course keeping the same level of probabilistic guarantees. Indeed, it may be counter-intuitive, but statistical estimators can be more pessimistic if they lack sufficient information on the sequence of measurements.

III. First results

In this section we provide first results of statistical methods applied to the interference analysis integrated within the WCET estimation.

Within the STARTREC project, we use a new Python library that has been designed by StatInf to fill the gap of EVT-based estimators of the statistical WCET in Python, while increasing their robustness as described in [24] using a combination of existing statistical estimators and novel methods. Moreover, the tightness of the statistical WCET EVT-based estimators has been improved by proposing a new statistical test, the WKS test [23]. The open problem that we will solve within the STARTREC project is the representativity one. We split this problem into three main parts:

- The representativity with respect to the program structure,
- The representativity with respect to the variation of the input variables,
- The representativity with respect to the interference analysis due to the presence of several cores.

The first problem may be solved by a hybrid approach which implies engineering effort putting together existing results, while the second problem has been formalized through a recent preliminary result [4]. In this paper, we deal with the representativity of measurements with respect to the third problem, the interference analysis.

To illustrate our strategy and for the sake of simplicity, we consider the program *minver* of the TACLeBench [7] executed on the QorIQ T1040 which is a Quad-core processor designed by NXP [13]. It belongs to the T1 family of QorIQ using the e5500 Power PC 64-bit single threaded core at 1.4 GHz with a private L1 cache (data and instruction caches) and a private L2 unified cache. It also has a CoreNet Platform Cache (shared L3 cache) associated to a DDR controller. In this paper, we consider e5500 cores and its private caches, as well as on the shared cache. The program *minver* is executed by imposing a specific memory location for the matrix to be inverted, while the adversaries are accessing the same memory location from different cores. The sequence of measured execution times is illustrated in Figure 1, where the horizontal axis corresponds to the order of execution times, while the vertical axis corresponds to the values of the execution times. We may note that such strategy is applied to the Easymile programs and other future publications may include comparison to other existing benchmarks [6], if publicly available.

In Figure 2, we represent the statistical WCET estimation obtained by using two possible EVT estimators, integrated within the StatInf tool and described in [24].

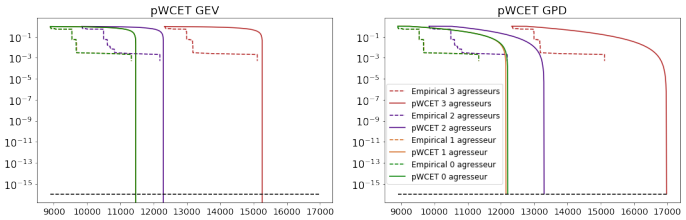

Fig. 2. The statistical WCET estimation using two possible implementations for EVT as implemented by the StatInf tool [24]

There are two possible EVT estimators, GEV and GPD, that first select maxima from an ordered sequence of execution times, and then they estimate the WCET distribution. The GEV estimator uses the block maxima approach by dividing data into several blocks with the same size and select the maximum of each block. While the GPD estimator selects the largest values above a given threshold (see Figure 3).

The statistical WCET estimator allows to quantify the impact of programs ”assaulting” the execution of the target program, *minver*. This aggression has been manually built for the particular case of the *minver* program. The purpose of STARTREC is to automate such a process with respect to a given configuration of a processor. Existing work focuses on providing interference analysis results under the hypothesis of worst-case processor configurations. In reality, the user comes with a configuration of the processor that has been tailored to meet other design constraints, such as security concerns, and a reduced configuration space is to be explored.

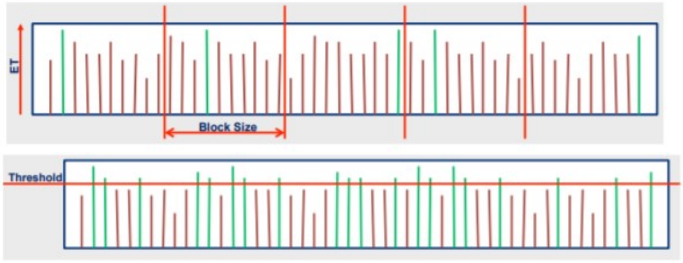

Fig. 3. The two EVT approaches are either picking the appropriate block size or the threshold.

IV. Conclusion and future work

Ongoing work within the STARTREC project shall bring answers concerning future development projects of safety critical software, with an objective to comply with normative activities, ISO 26262 or any other norm dealing with functional safety. The authors believe that the specification of software safety requirements will be enriched by the use of formal specification. The tooling scalability on large examples will determine whether this may be used or not for software verification. Embedded software testing will also benefit from formal specification as it allows the production of testing code for the specified pre-conditions and post-conditions.

The STARTREC project will contribute to tools and methods that extend the existing processes incrementally. The scale-up challenge for the tools and frameworks will take into account the following constraints:

- An industrial process based on Continuous Integration and Continuous Validation,
- Training and mentoring needs for embedded software developers, and auditors,
- All necessary activities to achieve the confidence in the use of software tools and comply with the normative framework, at the appropriate confidence level.

Within the STARTREC project, we also aim at performing Worst Case Execution Time analyses on the software used in the safety critical system, and justify their statistical significance by relying on the Extreme Value Theory. This aspect is challenging because of the complexity of the software and the hardware, leading to computation intensive phases interleaved with numerous data access and bus access patterns that are not humanly addressable. Here the authors claim that the progress of data analysis, combined with a deep knowledge of the underlying hardware, will offer the necessary support for engineers to produce optimized and predictable software with stringent timing constraints.

This approach will leverage the trust that formal verification and statistical WCET analysis brings to safety critical software while limiting the risk of software developments in terms of budget, timeline and certification.

V. Acknowledgment

This research is partially funded by the FR PSPC STARTEC supported by Bpifrance and La Région Occitanie, and the CIFRE StatInf-Inria agreement.

References

[1] Miguel Alcon, Hamid Tabani, Leonidas Kosmidis, Enrico Mezzetti, Jaume Abella, and Francisco J. Cazorla. Timing of autonomous driving software: Problem analysis and prospects for future solutions. In *IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, pages 267–280. IEEE, 2020.

[2] D. Buttle. Real-time in the prime time, keynote talk. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2012.

[3] Cédric Courtaud, Julien Sopena, Gilles Muller, and Daniel Gracia Pérez. Improving prediction accuracy of memory interferences for multicore platforms. In *2019 IEEE Real-Time Systems Symposium (RTSS)*, pages 246–259. IEEE, 2019.

[4] Liliana Cucu-Grosjean, Avner Bar-Hen, Yves Sorel, and Hadrien Clarke. The impact of the period variation on execution time distributions of programs. In *the 27th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Aug 2021.

[5] R. I. Davis and L. Cucu-Grosjean. A survey of probabilistic timing analysis techniques for real-time systems. *LITES*, 6(1):03:1–03:60, 2019.

[6] F. Cazorla et al. https://people.ac.upc.edu/fcazorla/archives/muBT-brochure-jan2017.pdf, 2017.

[7] Heiko Falk, Sebastian Altmeyer, Peter Hellinckx, Björn Lisper, Wolfgang Puffitsch, Christine Rochange, Martin Schoeberl, Rasmus Bo Sørensen, Peter Wägemann, and Simon Wegener. TACLeBench: A benchmark collection to support worst-case execution time research. In *the 16th International Workshop on Worst-Case Execution Time Analysis (WCET 2016)*, volume 55, pages 2:1–2:10, 2016.

[8] W.H. Freeman and Company, editors. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

[9] B.V. Gnedenko. Sur la distribution limite du terme maximum d'une seris aleatoire. *Annals of Mathematics*, 44:423–453, 1943.

[10] Road vehicles — safety of the intended functionality. Standard, International Organization for Standardization, Geneva, CH, 2019.

[11] Road vehicles — functional safety. Standard, International Organization for Standardization, Geneva, CH, 2018.

[12] Xavier Jean, Laurence Mutuel, and Vincent Brindejonc. Assurance methods for cots multi-cores in avionics. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–7, 2016.

[13] Kossivi Kougblenou, Rihab Bennour, Adriana Gogonel, and Liliana Cucu-Grosjean. Work-in-progress: Towards representative measurement protocols. In *the 41st IEEE Real-Time Systems Symposium (RTSS)*, pages 419–422, 2020.

[14] Viet Hoang Le, Loic Correnson, Julien Signoles, and Virginie WIELS. Verification Coverage for Combining Test and Proof. In *12th International Conference on Tests and Proofs*, Toulouse, France, June 2018.

[15] Benjamin Livshits, Manu Sridharan, Yannis Smaragdakis, Ondřej Lhoták, J Nelson Amaral, Bor-Yuh Evan Chang, Samuel Z Guyer, Uday P Khedker, Anders Møller, and Dimitrios Vardoulakis. In defense of soundiness: A manifesto. *Communications of the ACM*, 58(2):44–46, 2015.

[16] Cristian Maxim, Adriana Gogonel, Irina Mariuca Asavoae, Mihail Asavoae, and Liliana Cucu-Grosjean. Reproducibility and representativity: mandatory properties for the compositionality of measurement-based WCET estimation approaches. *SIGBED Rev.*, 14(3):24–31, 2017.

[17] Thomas Moscibroda and Onur Mutlu. Memory performance attacks: Denial of memory service in multi-core systems. In *USENIX Security Symposium*, 2007.

[18] Yannick Moy. Climbing the software assurance ladder - practical formal verification for reliable software. *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.*, 76, 2018.

[19] Yannick Moy, Emmanuel Ledinot, Hervé Delseny, Virginie Wiels, and Benjamin Monate. Testing or formal verification: Do-178c alternatives and industrial experience. *IEEE Software*, 30(3):50–57, 2013.

[20] Jan Nowotsch, Michael Paulitsch, Daniel Bühler, Henrik Theiling, Simon Wegener, and Michael Schmidt. Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement. In *2014 26th Euromicro Conference on Real-Time Systems*, pages 109–118. IEEE, 2014.

[21] Ahsan Saeed, Daniel Mueller-Gritschneder, Falk Rehm, Arne Hamann, Dirk Ziegenbein, Ulf Schlichtmann, and Andreas Gerstlauer. Learning based memory interference prediction for co-running applications on multi-cores. In *2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD)*, pages 1–6, 2021.

[22] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J.W.-S. Liu. Probabilistic performance guarantee for real-time tasks with varying computation times. In *Real-Time Technology and Applications Symposium*, pages 164–173, 1995.

[23] Marwan Wehaiba El Khazen, Liliana Cucu-Grosjean, Adriana Gogonel, Hadrien Clarke, and Yves Sorel. WKS test, a local unsupervised statistical algorithm for the detection of transitions in timing analysis. In *the 27th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, August 2021.

[24] Marwan Wehaiba El Khazen, Adriana Gogonel, and Liliana Cucu-Grosjean. Work in Progress: Lessons learnt from creating Extreme Value Libraries in Python. In *the 41st IEEE Real Time Systems Symposium (RTSS)*, Dallas / Virtual, United States, December 2020.

[25] Jakub Zwolakowski. Trust-in-soft ebook, 2020.