# Probabilistic Verification to Support Next-Generation Certification

Sam Procter
*Software Engineering Institute*
*Carnegie Mellon University*
Pittsburgh, Pennsylvania, United States of America
sprocter@sei.cmu.edu

Dionisio de Niz
*Software Engineering Institute*
*Carnegie Mellon University*
Pittsburgh, Pennsylvania, United States of America
dionisio@sei.cmu.edu

***Abstract*—The development of critical systems is heavily reliant on verifying system behavior through test, simulation, and other informal methods. Certification standards for those systems, though varying somewhat by domain, are largely process-focused. We describe a project that will evaluate the use of quantitative verification techniques which use the outputs of these informal methods (e.g., execution logs or test results) to establish bounded confidence intervals that can be used in support of new, property-focused certification approaches.**

## I. Overview

The failure of *critical systems* will lead to injury or death of humans, mission failure, or catastrophic financial loss. As a result, most critical systems are certified for, e.g., safety, security, or airworthiness before being deployed and used. Many standards for certification place a heavy focus on a critical system's development *process* as opposed to the *properties* of the engineered system itself. Though this approach has its benefits, there are drawbacks as well [1]. These drawbacks are compounded by a) the size and complexity of modern critical systems and b) the accelerated pace of technological change.

In the past few years, new statistical and formal techniques that rely on observations of system behavior have been developed that can provide a quantitative evaluation of properties of interest [2]. We are developing a new approach which relies on probabilistic model checking in order to prove certification properties. These properties would support a new, property-focused certification style while remaining connected to the well-established testing practices of traditional system development.

## II. Testing and Verification

The assurance and evaluation of critical systems is heavily reliant on testing as opposed to more formal methods of verification such as model checking or theorem proving. This testing takes a variety of forms, from low-level unit tests up through system-integration and acceptance tests. System evaluation may also include simulations of subcomponents or the entire system in various operating conditions. There are a number of reasons for the predominance of testing, and we are not advocating against them or their utility. However, there are drawbacks to relying nearly-exclusively on tests for verification, namely, that they do not provide the guarantees that formal methods do. More specifically, the confidence that testing provides is unquantifiable: a passed test (or even suite of tests) does not mean that a system has no bugs.

### A. Exhaustive Verification: Scaling and Overly-strong Guarantees

Formal verification techniques provide a high level of confidence that verified claims are true; this is of obvious benefit for critical systems. The types of claims that can be verified vary widely, and include ensuring the correctness of output, the timeliness of that output, that the physical behavior reliant upon that output is bounded within safe limits, and that the output will remain secure and correct in the presence of an adversary. Many formal techniques do not scale to larger inputs, though, and so can be difficult or impossible to use on real-world systems. Additionally, the high level of precision of formal methods are not always necessary since many system requirements are not absolute. Many systems can tolerate occasional deadline misses or small deviations from correct outputs as long as they are infrequent.

## III. Certification of Critical Systems

The goals and details of certification standards vary by system domain, but many consider the quality of the process used to create a system to be a proxy for the quality of the engineered system. Consider, for example, DO-178C, a standard used in the United States, Canada, and Europe for avionics software [3]. It places a heavy emphasis on the process used to create the software (i.e., the development lifecycle) and ensuring traceability between software-development artifacts (e.g., between source code and low-level requirements, or high-level requirements and test cases). DO-178C and related standards have their benefits: they have enabled an impressive aviation safety record, and their focus on process also means that the impact on a system's development schedule is well-understood and predictable.

### A. Pace of Technological Change

The rate at which technology changes poses a challenge to certification approaches like DO-178C which require supplements to address new technologies and approaches. Consider, for example, DO-331 and DO-333: they are supplements to

DO-178C for the use of model-based development and formal methods, respectively. A more flexible approach, grounded in argumentation about the system itself and the properties it must exhibit for safe functioning, would enable more rapid use of current and future technologies. This is of particular importance given the pace at which artificial intelligence is poised to remake software development. Indeed, we agree with Chelini et al. who write that "the approach adopted in DO-178C to address new technologies through specific supplements does not appear sustainable because the time to develop additional supplements may not be timely with the introduction of new technologies." [1]

### B. The Overarching Properties

An attempt to improve and clarify certification approaches like DO-178C, termed the *Overarching Properties* (OPs) began in a FAA working group in late 2015 [1], [4]. The three OPs, which "are intended to define a sufficient set of properties for making approval decisions" are Intent (the defined behavior is correct), Correctness (the implementation is correct), and Innocuity (parts of the implementation that are not required are not unacceptable) [1]. The OPs make explicit several implicit aspects of existing certification approaches, and shift the focus to properties of the system itself rather than the system's development process [5].

## IV. Probabilistic Verification for Certification

A particular challenge inherent to property-based certification approaches is that the time required for certification is much less predictable than with process-based certification. This is because it is challenging to predict how long it will take to demonstrate that a system exhibits a certain property, e.g., meets its timing requirements. What's more, the arguments used to establish that a system exhibits the OPs may be challenging to construct using only test or simulation results. We propose the use of probabilistic verification techniques which take as input testing outputs (or simulation data / execution logs) and produce as output bounded confidence intervals on system properties. We note that there is support from some of the creators of the OPs for such an approach: probabilistic verification is identified as a means of demonstrating that a system has a particular property in [5].

To this end, we are encoding certain airworthiness properties in formal language so that they can be used with the *Formal verificAtion with Confidence inTervals* (FACT) approach [2]. This usage, which would be part of a larger property-based certification process like the OPs, would rely on observations of system behavior derived from traditional system evaluation activities like testing or simulation. As Figure 1 shows, the FACT approach requires three inputs in addition to testing / simulation data:

- A *state formula* $\Phi = \mathcal{P}_{\bowtie p}[\Psi]$*:* Essentially an encoding of the system property of interest (e.g., an aircraft's autopilot will not cause it to become unstable) using Probabilistic Computation Tree Logic (PTCL) [6].
- A *parametric Markov chain* $\mathcal{M}$*:* Developed along with the state formula $\Phi$, the events of this Markov chain are abstractions, relative to the property of interest, of the system's states and their transitions.
- A *target error level* $\alpha$*:* $1 - \alpha$ is the user's desired error level for the property of interest.
- *State transition observations* $\mathcal{O}$*:* These observations can be extracted from a variety of system tests, execution or simulation logs, or other activities that exercise the system. Part of the contribution of our work is deriving this automatically from the sorts of system verification activities being done in traditional certification like DO-178C.

| | Inputs | Process | Goals | Outcomes |
|---|---|---|---|---|
| DO-178C | Test Plan | Conduct tests following plan | Modified condition/ decision coverage | Major bugs found |
| FACT | $\Phi$, $\mathcal{M}$, and $\alpha$ | Conduct tests to collect $\mathcal{O}$ | Confidence interval | $(1-\alpha)$ confidence interval for $\mathcal{P}_{=?}[\Psi]$ |

Fig. 1. A high-level comparison of the testing approach described by DO-178C [3] the FACT approach from [2].

The state formula and parametric Markov chain will require expertise with the technique to create, but once specified, will be re-usable by non-experts, i.e., once a system requirement has been encoded in the formalism used by the FACT approach, domain (rather than formal verification) experts will be able to use software tooling to compute the confidence that the system will meet the requirement. Our goal with this effort is that once a property has been specified, all that is needed to generate bounded confidence intervals on the property are the sorts of tests being done in traditional development processes.

This effort, if successful, will provide some of the benefits of formal, quantitative verification for system development. It will support property-based certification approaches, like the Overarching Properties, while also reducing the schedule impact typically associated with formal verification. This reduction (in terms of time required to conduct the verification activities) comes about because our approach uses data being generated by existing system evaluation activities, e.g., testing, simulation, and other execution logs. By re-using this data instead of requiring new, additional activities, system developers can get the benefits of formal verification with less effort and expense.

## Acknowledgment



This material is based upon work supported by the Department of War under Air Force Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of

[1]These definitions are paraphrased, see complete definitions and explanations in [4]

the Software Engineering Institute, a federally funded research and development center.



DM26-0214

## REFERENCES

[1] J. Chelini, J. L. Camus, C. Comar, D. Brown, A.-P. Porte, M. de Almeida, and H. Delseny, "Avionics Certification: Back to Fundamentals with Overarching Properties," in *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, Toulouse, France, January 2018. [Online]. Available: https://hal.science/hal-02156109

[2] R. Calinescu, C. Ghezzi, K. Johnson, M. Pezzé, Y. Rafiq, and G. Tamburrelli, "Formal Verification With Confidence Intervals to Establish Quality of Service Properties of Software Systems," *IEEE Transactions on Reliability*, vol. 65, no. 1, pp. 107–125, March 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7177126

[3] S. C. . (SC-205), "Software considerations in airborne systems and equipment certification (rtca do-178c)," RTCA, Inc., 1150 18th Street, NW, Suite 910, Washington, DC 20036-3816 USA, Tech. Rep. DO-178C, December 2011.

[4] C. M. Holloway, "Understanding the Overarching Properties," National Aeronautics and Space Administration, Tech. Rep. NF1676L-33745, July 2019. [Online]. Available: https://ntrs.nasa.gov/citations/20190029284

[5] K. S. Wasson and C. M. Holloway, "An Introduction to Constructing and Assessing Overarching Properties Related Arguments (OPRAs): Version 1.0," National Aeronautics and Space Administration, Tech. Rep., Jan. 2022. [Online]. Available: https://ntrs.nasa.gov/citations/20210025425

[6] F. Ciesinski and M. Größer, "On Probabilistic Computation Tree Logic," in *Validation of Stochastic Systems: A Guide to Current Research*, C. Baier, B. R. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, Eds. Berlin, Heidelberg: Springer, 2004, pp. 147–188. [Online]. Available: https://doi.org/10.1007/978-3-540-24611-4_5