DOI:10.1145/3574133

An attempt to provide a common language between formal models, simulations, real-world (testing) data, and regulatory mechanisms.

BY GEORGIOS BAKIRTZIS, STEVEN CARR, DAVID DANKS, AND UFUK TOPCU

Dynamic Certification for Autonomous Systems

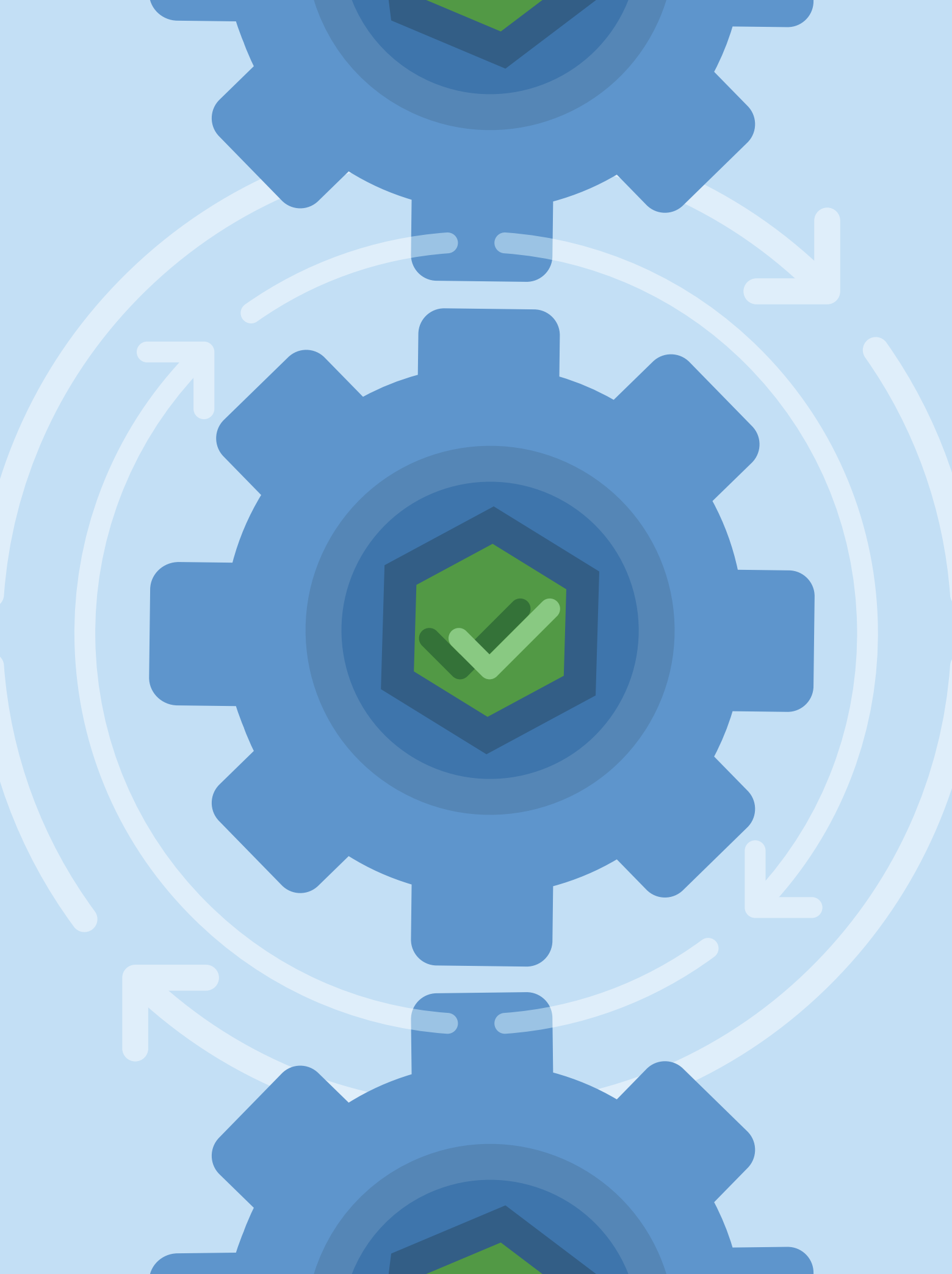
AUTONOMOUS SYSTEMS ARE often deployed in complex socio-technical environments, such as public roads, where they must behave safely and securely. Unlike many traditionally engineered systems, autonomous systems are expected to behave predictably in varying "open-world" environmental contexts that cannot be fully specified formally. As a result, assurance about autonomous systems requires the development of new *certification methods*—codified checks and balances, including regulatory requirements, for deploying systems—and mathematical tools that can dynamically bind the uncertainty engendered by these diverse deployment scenarios. More specifically, autonomous systems increasingly use algorithms trained from data to predict and control behavior in contexts previously not encountered. Using learning is a critical step to engineer autonomy that can successfully operate in *heterogeneous contexts*, but current certification methods must be revised to address the dynamic, adaptive nature of learning. The heterogeneity that any certification framework should address for the design of autonomous systems is twofold. The first relates to the system itself and the heterogeneous components that engender its behavior. The second is the heterogeneity that certifying must address in relation to the complex socio-technical settings in which the system is expected to behave.

We propose the *dynamic certification* of autonomous systems—the iterative revision of permissible ⟨use, context⟩ pairs for a system—rather than pre-specified tests that a system must pass to be certified. Dynamic certification offers the ability to learn while certifying, thereby opening additional opportunities to shape the development of autonomous technology. This type of comprehensive, exploratory testing, shaped by insights from deployment,

» key insights

- **Dynamic certification of autonomous systems requires the involvement and distillation of knowledge from diverse stakeholders, from engineers to architects to regulators. It must start at the beginning of the life cycle, where decisions are most effective and cost of design changes are lowest.**
- **Iterative assessment for dynamic certification depends on the use and context in which autonomous systems are expected to deploy and must account for both technical and social requirements.**
- **Multiple rounds of testing and interrogating requirements via formal methods provide insights into the uncertainty present in varying deployment scenarios, where they must perform efficiently and safely.**

IMAGE BY ANDRIJ BORYS ASSOCIATES. USING SHUTTERSTOCK



can enable iterative selection of appropriate contexts of use. More specifically, we propose dynamic certification and modeling involving three testing stages: early-phase testing, transitional testing, and confirmatory testing. Movement between testing stages is not unidirectional; we can shift in any direction depending on our current state of knowledge and intended deployments. We describe these stages in more detail later, but the key is that these stages enable system designers and regulators to learn about and ensure autonomous systems operate within the bounds of acceptable risk.

Our proposal is similar to how the Food and Drug Administration (FDA) tests drugs in stages with increasing scrutiny before the products are approved for public consumption. Rather than a simple yes/no certification, the FDA uses an iterative process of exploratory stages in which pharmaceutical agents are first approved for limited use in restricted contexts under careful oversight. They are only gradually approved for broader use as post-approval monitoring and subsequent studies demonstrate safety and efficacy. Of course, FDA procedures cannot be used directly for dynamic certification of autonomous (software) systems, but they provide an "existence proof" that dynamic certification can work.

Technology creation involves at least two different yet interdependent types of decisions. *Design decisions* determine the structure and intended operation of the autonomous system, including the evaluation functions optimized during development and revision/updates. *Deployment decisions* determine the contexts and uses for the autonomous system, including designating certain situations as "do not use" (or "use only with increased oversight"). In practice, static certification and regulatory systems often focus only on deployment decisions (and take the design decisions and technical specifications as fixed). However, precisely because of the frequent uncertainty about what counts as success for an autonomous system, certification of those systems must also consider design decisions, using technical specifications to predict performance in contexts that are not encountered.

Dynamic certification includes design decisions, particularly in the early stages when changes have the highest impact and lowest cost, often before code or hardware have even been built.[15,35] Mathematical tools from formal methods can thus play an essential role in specifying autonomous systems at different levels of abstraction, even when they have not yet been implemented. Formal methods allow us to specify acceptable risks, identify failures that inform mitigation strategies, and understand and represent the uncertainty associated with deploying autonomous systems in heterogeneous environments. Formal models are also living documents that encode design and deployment decisions made throughout the life cycle of the autonomous system. For example, tracking changes in the specification of requirements throughout the life cycle can offer a good picture of the design problems and solutions at a particular time and how those changes reflect design shifts over time. Successful dynamic certification thus depends on translational research by formal methods, autonomous systems, and robotics communities to establish proper procedures to ensure that deployed systems are unlikely to cause harm.

Dynamic certification relies on an iterative assessment of the risks (and benefits) introduced by deploying autonomous systems for different uses and contexts. Formal methods offer a concrete basis for specification, verification, and synthesis for autonomous systems but do not guide the translation of our desired values and acceptable risk into those formal models. We require frameworks that explicitly allow for ambiguities in specifications and uncertainties and partial decisions in modeling while remaining scalable to practically relevant sizes. More generally, dynamic certification will require an appropriate co-evolution of regulatory and formal frameworks. Having argued for implementing parts of dynamic certification via formal methods, it is crucial to acknowledge other types of analyses that could implement dynamic certification, such as assurance cases,[3] structured interrogation of requirements,[7,28,38] and domain standards.[12] Indeed, these other types of methods and their associated tools and metrics could play valuable roles in the dynamic certification of autonomous systems.

Figure 1. Different ⟨use, context⟩ pairs result in different hazard conditions.

The arrows and shaded regions are possible trajectories: safe (blue) and hazardous (red). Dotted lines indicate connectivity. (a) In suburban areas, the probability of losing connection is low because the signal has minimal attenuation. (b) In a city, buildings are made with concrete and rebar and are laid out densely, increasing the probability of signal attenuation. Therefore, the system is more likely to transition to a hazardous state in the city ⟨use, context⟩ pair. When attempting to deploy in these different scenarios, we must check different safety properties. Dynamic certification requires the development of a framework for designers that easily adapts between the different scenarios.

(a) Suburban context (b) Urban context

Scenario

We motivate and illustrate our proposed framework for dynamic certification using a scenario with two interacting systems: an unmanned aerial vehicle (UAV) and a ground-based delivery robot simultaneously delivering packages (see Figure 1). We require that the UAV only operate while connected to a wireless communication network. If it (likely) loses connection, it must land in place. A hazardous state results if the UAV lands in the same location as the ground-based robot. The UAV designer seeks a high-level risk-mitigation strategy that accounts for the ground-based robot's movement and limits the

probability of transitioning to a hazardous state. This strategy requires specification of acceptable ⟨use,context⟩ pairs—for example, usable trajectories or locations for the UAV. This high-level focus enables key abstractions and simplifications. For instance, the designer can abstract away the low-level UAV controller and assume that it can safely navigate between waypoints. Instead of modeling complex behaviors about the ground-based robot—which may not even be possible if the system is outside of the designer's control—the designer can require that the UAV be robust against random movements of the robot. Though this example is simplified in various ways (for example, only including two robots), those simplifications serve to highlight key conceptual points, including the value of formal methods.

Dynamic Certification

Dynamic certification is built on two fundamental operations: modeling and testing. Modeling allows system engineers to track design choices that otherwise would be difficult to document and adjust when issues arise because of complex interactions between sub-components. Models also enable the engineer to focus on the interfaces between sub-components, often abstracting away the individual sub-components to focus on the behavior of the whole. Importantly, we can use models to understand how the system might succeed or fail before the system is built—that is, for high-impact, low-cost design decisions. In contrast, testing involves the actual implementations, focusing on whether the assumptions of the model and resulting design decisions actually function as expected in the physical world. Conventional static certification struggles when operational (or regulatory) assumptions fail to hold in reality. In contrast, dynamic certification posits that modeling and testing should be intertwined throughout the system life cycle, so our models (and assumptions) can be continually refined as we better understand real-world contexts. Certification of autonomous learning systems requires both elements: testing, since the world can surprise us and the system can change through learning, and modeling, to guide our design and testing decisions through the massive search spaces.

Dynamic certification relies on an iterative assessment of the risks (and benefits) introduced by deploying autonomous systems for different uses and contexts.

Assurance requires specifying when, where, and why an autonomous system is being deployed within a sociotechnical context. But if autonomous systems are expected to learn from their environment and context of operation, then there does not seem to be a stable model for testing. Dynamic certification turns this concern into a virtue: If our base system model contains appropriate parameters, we can iteratively refine and augment this base model through different testing procedures. This virtue and the resulting testing procedures come from the feedback and interaction between stakeholders with different concerns and expertise, making it clear when testing procedures are sufficient and accurate. Therefore, in the long run, we can conduct sufficient testing to have an accurate model that assures stakeholders that systems will operate as expected.

Specification of the base system model for dynamic certification requires (perhaps partial) identification and description of the following four components (inspired by Kimmelman and London).[20]

- **Modules** of the system (primarily software, but potentially hardware) including the function(s) of each module.
- **Contexts** in which the system is expected to be able to operate successfully.
- **Mappings** from Context → Behavior for "successful" performance in various conditions.
- **Variations** in the environment for which the system should be robust.

Given an initial specification of these four elements for an autonomous system, dynamic certification can be divided into three distinct stages (with no requirement for unidirectional progression through these stages). All four components of the base-model specification can be revised or adjusted during each stage. Although discussions of certification often focus on Contexts and Mappings, the inclusion of design decisions in dynamic certification means that other components can also be adjusted—for example, adding Modules to improve performance in given Contexts).

The first stage, *early-phase testing*,

occurs in the development lab or other highly controlled settings. The two main goals of this stage are to verify that the integrated Modules implement the intended Mappings and to develop appropriate base models of the autonomous system for offline testing. The first goal is relatively standard when developing a software system—for example, unit-testing. The second goal, however, is much less common and requires careful consideration of the range of Contexts and Variations that might be encountered in plausible deployment environments. Importantly, all four components of the base system model must be (tentatively) specified in early-phase testing; this stage is not solely technology-focused. Given an initial specification, early-phase testing continues until the software system is suitably verified and its expected performance is sufficiently good in offline testing. In the running scenario, early-phase testing could take the form of building and testing a gridworld that models the high-level decision-making for the UAV. In this stage, the designer would identify anomalous behavior, such as locations that create deadlocks, thereby enabling design decisions to mitigate situations that lead to task degradation.[14]

In the second stage, *transitional testing*, the system is deployed in real-world environments, though with significant oversight and control. The two main goals of this stage are to identify Contexts of real-world failure and to characterize potential environmental Variations. These goals require highly active engagement and interventions; this stage is not simply "deploy and watch" or "compare to prior standards." Rather, transitional testing should involve, for example, focused efforts to place the system into "hard" contexts precisely to improve our understanding of the system. Transitional testing involves careful, systematic efforts to determine the boundaries of appropriate system performance. The information produced by this testing can be iteratively used to change Modules, constrain Contexts, add Mapping complexity, or increase Variation specificity. Transitional testing is exploratory (helping to understand) not merely confirmatory (checking if the system performs as expected).

Dynamic certification uses testing throughout the life cycle, revealing challenges and trade-offs while design decisions and changes are still possible.

In our running scenario, transitional testing would involve testing (not just modeling) system performance with high-fidelity and hardware-in-the-loop simulations[4,9,34] or in controlled environments—for example, a large industrial park with limited public traffic. This stage intends to gather enough data to modify the formal system model to reflect reality further.

In the third and final stage, *confirmatory testing*, the system is deployed with significant oversight and monitoring, but no further controls beyond those specified in the certification by a set of ⟨use, context⟩ pairs. This stage aims to determine, in real-world settings, both system performance reliability and the extent of system-user value (mis)matches. The latter goal is crucial because many autonomous system "failures" involve a properly functioning system that implements different values than the users expect. The system behaves correctly, but according to a (perhaps implicit) notion of success that is different from that of the human users;[a] that is, the system implements the wrong Mapping. These divergences often appear only once the system is in the hands of untrained users, so confirmatory testing must initially include significant oversight to detect, record, and respond to real-world performance failures and value divergences. This monitoring can be gradually reduced as we learn the exact behavior of the system in relevant real-world contexts—that is, even this stage involves some exploratory testing.[b] In the running scenario, confirmatory testing would involve supervised deployment in a controlled environment, possibly borrowing rules and regulations from the operational design domain.[21] Changes to the system design based on actual operational contexts should reflect the formal model; they must agree. Once testing and modeling agree, the dynamic certification has ensured that the system will behave acceptably and safely.

a Many classic examples of "AI run amok" fall into this category. For example, the paperclip maximizer[6] simply has a different idea of "success" than we do.

b Confirmatory testing is thus quite similar to conformance testing but does not assume that we have a fully specified set of standards and behaviors that are provided in advance.

Current static certification frameworks involve testing only late in the life cycle after a particular system implementation has been built and is often already deployed. They could theoretically play a role beyond setting performance targets, but in practice, they rarely do. In contrast, dynamic certification uses testing throughout the life cycle, revealing challenges and trade-offs while design decisions and changes are still possible. The benefits of life cycle-wide testing require models that can capture the "what", "why", and "how", along with connections to the eventual system design. Formal models play a particularly valuable role in dynamic certification. In particular, formal models can be used early to interrogate our assumptions about the system's requirements rather than only being used late to provide provable guarantees. Formal methods also can give us the tools to add stakeholder-specific semantics to various models of behaviors, requirements, and architectures, thereby providing a common language to reason about the system's design.

Formal Methods for Dynamic Certification

Formal models use the precision of mathematical language to reveal misunderstandings about the system's behavior and requirements.[13,24,29,39] Formal specifications can model complex systems before developing code or synthesizing hardware architectures, allowing systems engineers to interrogate requirements and find clashes and interaction faults early in the system's life cycle. Using formal models, we can architect a system proactively: no system exists yet, so our design-decision effectiveness is highest and the cost of changes lowest since we do not have to bolt modifications onto a preexisting design. Additionally, we can often synthesize behaviors directly from the formal model, which then provides our implementation with guarantees about important properties, such as safety.[33] Finally, formal models can inform testing procedures by simulating different contexts and becoming more comprehensive (and therefore informative) as system data is collected during deployment[11,22]—with the caveat that there will always be a need to interpret those formal results to account for the gap between formal models and reality. It is impossible to make autonomous systems 100% safe 100% of the time, but we posit that formal methods can significantly assist in designing better, safer systems.

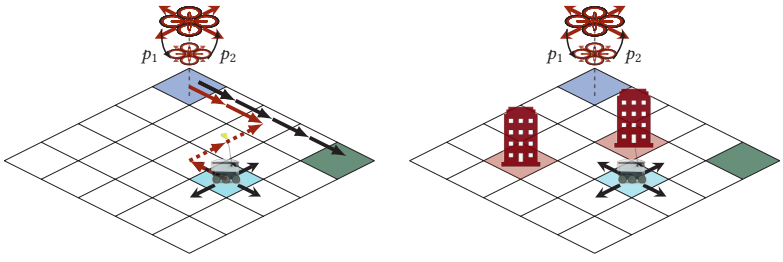
In particular, formal methods can be highly valuable for the dynamic certification of an autonomous system. Formal models can specify behavior and system dynamics that are difficult to implement and test without committing to a specific design. Formal models can therefore be used as an aid to inform what testing ought to take place to ensure that the system will behave as expected. Formal models and specifications can also be readily updated given new data to improve analysis precision as the system is deployed.

One formal model for autonomous systems that is especially useful for dynamic certification is the Markov decision process (MDP). The MDP models sequential decision making in stochastic systems with nondeterministic choices.[30] MDPs have been useful for modeling high-level decisions in autonomous systems, such as collision-avoidance,[36] surveillance using ground-based robots,[23] and transmission exchange for wireless sensor networks.[2] Analysis with MDPs typically requires that the complete model be known a priori,[18] but there is often significant model uncertainty in early phases since many design and deployment decisions have yet to be made. We can instead use a class of model known as a *parametric* MDP,[16] where parameters model variations in transition probabilities. The parameters may thus represent design choices (for example, requiring a perception Module with a certain error rate or setting specific thresholds for underlying decision-making algorithms); deployment decisions and context characteristics (for example, possible reductions in visibility or likelihoods of interruption of information flow); or modeling uncertainties (for example, unknown characteristics of motion or reaction time under off-nominal conditions). Parametric MDPs have the specificity and flexibility required for a base system model that can be refined and improved through exploratory early-phase testing.

We illustrate the use of parametric MDPs as an early-phase decision-making tool in our running scenario with two autonomous systems: a UAV and a ground-based delivery robot, simultaneously delivering packages (see Figure 2). For dynamic certification, we want to iteratively identify uses and contexts for which the UAV can safely deploy while continually gathering additional data to determine when it can be deployed in more heterogeneous environments. Safe deployment is critical in all phases (not just confirmatory testing) due to the possibility of prob-

Figure 2. Interrogating models containing different *Contexts*.

Using formal methods, we can interrogate models containing different contexts. For example, a UAV (shown in red) deployed in a city (left) is more likely to lose connection and be forced to land (opaque UAV) compared to a similar UAV operating in a suburban area. For the city context, the value for probability p_1 would be larger than the suburban context for the same parameter. This means we must account for and understand how the same system can be less or more likely to encounter hazardous conditions when interacting with ground-based agents (landing in the vicinity of the delivery robot). We include a possible counterexample whereby the UAV drops connection and crash-lands on the delivery robot. In a similar, albeit slightly modified context (right), we can choose to explicitly eliminate the possibility of ground-based interactions by having the UAV enter the ground layer in a prescribed controlled landing zone in the form of the building roofs (shaded red).



(a) Context without landing zones

(b) Context with landing zones

lematic incidents. For example, if the UAV were to hit a delivery robot in early testing, then even if there was no damage to either system, this reportable event might delay the UAV's certification and eventual deployment. Mitigating these issues at design time makes it less likely that such an event would occur and more likely that the system would deploy within schedule.

Suppose we have a list of formal requirements (perhaps translated from human values) for UAV performance. Parametric MDPs provide a useful model to check what probability these properties hold or, perhaps more importantly, do not hold. In this context, the UAV computes a policy that maximizes the probability of satisfying a temporal logic specification. Based on a finite number of samples of the uncertain parameters, each of which induces an MDP, we can estimate the best-case probability that the policy satisfies the required specification by solving a finite-dimensional convex optimization problem.

More specifically, we have three relevant, high-level Modules (Figure 2) that determine the movement of the UAV, the movement of the robot, and the communication status of the UAV. When translating a physical environment such as the predefined scenario (Figure 1) into a formal model, we abstract roadway intersections as states in a gridworld (Figure 2). While gridworlds represent rather simplistic modules, they are quite powerful in demonstrating scalable behavior. Simply, an agent that fails to behave safely in such simple environments is also unlikely to behave safely in the real world.[26] A parametric MDP can model the composition of these three modules into a single socio-technical system. The UAV can land and take off from anywhere in the region. It will lose connection and land-in-place with probability p_1 (opaque UAV in Figure 2) and remain grounded until it reestablishes connection with probability p_2. We formalize the UAV goal of safely delivering the package as the requirement that the UAV behavior maximizes the probability that it delivers a package to the green region while not creating an incident by landing in the same physical location as the delivery robot. We describe such a mission using the temporal logic formula $\varphi = \neg\text{Crash U Goal}$, where Crash is true when a landed UAV shares the same location as the delivery robot. We thus abstract away complex low-level interactions involving landing or taking off in a crowded region; instead, we focus on the human-relevant behavioral understanding and characterization of what might go wrong.

For the range of parameter values, we compute policies for the system using the Storm probabilistic model-checking tool.[10] When synthesizing the optimal policy, that is, the policy that satisfies the expression $p_{max}[\varphi]$, we can also compute the probability that an agent employing this policy will satisfy this mission (Figure 3). These probabilities can then be used to provide crucial guidance in the dynamic certification process.

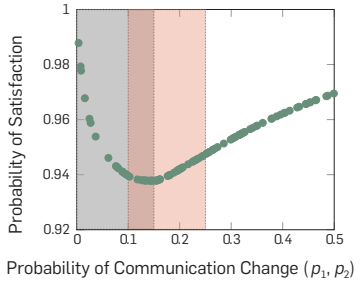
For instance, when beginning the early-phase testing stage, the designer has minimal insight into the values of p_1 or p_2. One possible outcome is that initially, the designer may assume these probabilities correlate with signal strength and are, therefore, equal—that is, $p_1 = p_2$. In such a case, the formal model is a parametric MDP with a single parameter. Under this assumption, we can certify that the agent will successfully perform its mission no worse than ~93% of the time (Figure 3a). However, during transitional or confirmatory testing, we may gather more information about the system and learn that $p_1 \neq p_2$. In light of this new information, we can return to early-phase testing to reconsider the UAV behavior (in this environment) as modeled by a parametric MDP with two parameters. In the process of synthesizing these policies, we can now compute the probabilities of success across values for both parameters (Figure 3b).

The integrated modeling and testing in dynamic certification can lead us to specify a threshold on p_1 or p_2 for safe deployment. We might identify specific, measurable features that define appropriate deployment contexts. For example, we might require that $p_1 \in [0, 0.15]$ (Figure 3, highlighted in gray). Our current design in suburban contexts might satisfy this constraint but require additional changes for urban contexts. We might adjust the design of the UAV (for example, using a more reliable communications device) or instead adjust the context (for example, providing additional signal towers). In either case, we can justifiably determine the systems, uses, and contexts where safe deployment can be assured (to a given probability).

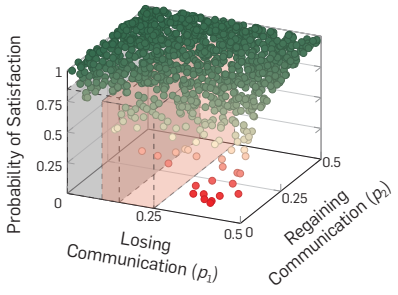
Alternately, an urban context such as Figure 1b could include buildings that provide safe landing zones for the UAV (Figure 2b). In this context, we can compute a policy that ensures success regardless of the values of p_1 and p_2. In other words, the contextual deployment face of safe landing locations in the urban context alleviates the need to test our model for many possible values of p_1 and p_2. Specifically,

Figure 3. Probability a UAV satisfies its mission.

The probability that the UAV satisfies the specification when employing its optimal policy for attempting to deliver a package without incident to the green square in the gridworld model (Figure 2). In the highlighted regions, we show the expected parameter range for p_1 for both suburban (gray) and urban (pink). The samples in green (b) are likely to complete the delivery without incident while the UAVs operating in the environments shown in yellow and red samples are more likely to land-in-place, increasing the probability of an incident occurring.



(a) Assumption that $p_1 = p_2$



(b) Assumption that $p_1 \neq p_2$

the UAV's policy would have it fly between building rooftops only when it can safely cross without collision and loiter at the rooftop otherwise. Of course, such a policy may result in extremely long loitering times while the UAV waits for the delivery robot to move away from the goal region. We could thus make the design decision to include battery charge as an additional parameter in the UAV parametric MDP system model. This design decision could change the acceptable deployment contexts, though the details depend on what was learned through exploratory testing.

Certifying Autonomous Systems in Socio-Technical Contexts

Testing in static certification can be tractable because the target performance is specified ahead of time.[19] In contrast, testing in dynamic certification might appear completely intractable as it depends on the changing system, use, and context. We propose that the integration of modeling and testing can make dynamic certification feasible. A formal model can provide precise, context-sensitive specifications for the system's implementation and inform the types of tests we conduct. This type of dynamic certification will ideally result in believable and defensible guarantees of correct operation.[c] More importantly, this dynamic certification leads to early-phase models that can be used to interrogate required or acceptable behavior, even in the absence of a specific software or hardware implementation. Compared to conventional certification regimes, dynamic certification revises our assumptions and improves decisions or requirements before the system is even built, all with the added benefit of identifying the types of contexts that led to design changes. The effort to understand required assurances can begin while we can still effectively change the design or the broader socio-technical context.

Dynamic certification systematically identifies context-dependent stakeholder values and incorporates them in modeling and testing autonomous systems. We have outlined one way of implementing dynamic certification using formal methods and models. In contrast to the current practice of using formal methods for guarantees once a system is built, we can also use formal methods to model values and restrictions within deployment scenarios and open environments. Formal methods can simulate socio-technical parameters, not just the technological system. The central message of dynamic certification is that we must implement precise feedback loops between formal models, simulation environments, and increasingly open-world deployments, all to ensure that stakeholder values are being protected and advanced. Formal models provide a crucial tool in these loops as they can justify dynamically evolving requirements.

Scalability is not an issue within dynamic certification because we expect the formal model to provide partial proof of safe deployment.

A common concern when implementing verification and certification using formal methods is scalability. However, scalability is not an issue within dynamic certification because we expect the formal model to provide partial proof of safe deployment. Dynamic certification augments formal models with testing for this reason. Indeed, the feedback between models, testing, and stakeholder values minimizes scalability issues found in most static certification contexts. The latter must test all behaviors of systems, while the former can focus on the behaviors that lead to losses (violation of stakeholder values).[27] Formal methods thus do not need to scale indefinitely;[d] however, that does not mean we could instead require regulators and designers to tame some of the environmental complexity or limit the required autonomy. Alternately, one could require only that the formal model demonstrates resilience in the sense of a return to normalcy after an uncontrolled action. This narrower requirement can vastly reduce the scenarios for formal modeling and assur-

c We cannot require infallible guarantees, as they may be based on incorrect or imperfect assumptions. No certification process can be perfect, but dynamic certification has the benefit of continued testing to detect incorrect (formal) models.

d This does not mean that bottlenecks cannot be hit with partial models, but partial modeling gives us the means to say something meaningful about the relationship between what the system ought (not) to do (that is, its requirements) and what the exhibited behavior actually is.

ance,[17] as others can specify what is required for "normal behavior."

Dynamic certification differs from conventional certification not because it proposes stages and feedback loops—already present in static certification—but based on the types of testing (exploratory, not just confirmatory) and specification (partial, rather than complete) in every stage. The more precise data we can capture with models and tools, the better-informed stakeholders will be to ensure the operational needs of the system. Toward this goal, research must be conducted at the intersection of robotics, control, learning, safety, security, resilience, testing, and formal methods. For example, roboticists must include realistic dynamical models for surrounding information that can be given by learning;[32] learning must be interpretable based on test vectors;[31] control must account for clashing safety requirements based on dynamics;[28] and safety,[25] security,[37] and resilience[8] must be given formal interpretations based on realism but allow partial modeling, precisely to account for the uncertainty arising from coupled learning systems. Two recent improvements that will assist with developing dynamic certification are compositional verification, which relates different model types,[5] and more operational data—for example, high-definition maps for streets in major cities.[1]

Dynamic certification is an approach for autonomous systems that attempts to provide a common language between formal models, simulations, real-world (testing) data, and regulatory mechanisms. Dynamic certification requires advances in formalism compatibility and co-design, the development of high-fidelity simulation tools that can input information from formal models, expansive context-aware testing vectors, and legal codification of acceptable stages of deployment. In light of these multidisciplinary aspects, it is unsurprising that dynamic certification has been a relatively under-explored approach. However, dynamic certification promises better-designed, safer, and more secure autonomous systems, providing assurance of correct behavior and increased deployment of those systems. The effort to advance dynamic certification can provide significant benefits.

At the same time, AI presents additional challenges for the dynamic certification of autonomous systems. First, the distributed nature of much AI and robotic development can lead to significant communication barriers between different stakeholders during the requirements elicitation stage, and research is needed to develop, test, and validate structured approaches for requirement and value elicitation. Second, modular and scalable methods and tools are needed to characterize precisely—whether through formal methods or otherwise—the connections between requirements and system (mis)behavior, particularly given the inevitable uncertainties with AI-enabled systems. Third, higher-fidelity causal models could improve counterfactual reasoning in the design and certification of autonomous systems, as the certification processes could then incorporate additional feedback loops that identify counterexamples in data collection, provide diagnostic capabilities, and clarify assumptions used to evaluate performance of the autonomous system in uncertain, open-world environments. C

References

1. ARGO AI. *Argoverse: Public Datasets Supported by Detailed Maps to Test, Experiment, and Teach Self-Driving Vehicles How to Understand the World around Them.* (2022); https://tinyurl.com/2n52u8fn.
2. Alsheikh, M.A. et al. Markov decision processes with applications in wireless sensor networks: A survey. *IEEE Communication Surveys and Tutorials.* (2015).
3. Asaadi, E. et al. Dynamic assurance cases: A pathway to trusted autonomy. *Computer* (2020).
4. Bacic, M. On Hardware-in-the-loop simulation. In *Proceedings of the 44th IEEE Conf. on Decision and Control IEEE*, (2005).
5. Bakirtzis, G. et al. Compositional thinking in cyberphysical systems theory. *Computer* (2021).
6. Bostrom, N. Ethical issues in advanced artificial intelligence. *Rev. of Contemporary Philosophy* (2006).
7. Bourbouh, H. et al. Integrating formal verification and assurance: An inspection rover case study. In *Proceedings of the 13th Intern. Symp. on NASA Formal Methods, Lecture Notes in Computer Science*. Springer (2021).
8. Bouvier, J. et al. Quantitative resilience of linear driftless systems. In *Proceedings of the Conf. on Control and its Applications*, SIAM (2021).
9. Curiel-Ramirez, L.A. et al. Hardware in the loop framework proposal for a semi-autonomous car architecture in a closed route environment. *Intern. J. on Interactive Design and Manufacturing* (2019).
10. Dehnert, C. et al. A storm is coming: A modern probabilistic model checker. In *Intern. Conf. on Computer Aided Verification*. Springer (2017), 592-600.
11. Fan, C. Formal methods for safe autonomy: Data-driven verification, synthesis, and applications. *Ph.D. Dissertation*. University of Illinois at Urbana-Champaign (2019).
12. Farrell, M. et al. Evolution of the IEEE P7009 standard: Towards fail-safe design of autonomous systems. In *Proceedings of the IEEE Intern. Symp. on Software Reliability Engineering* (2021).
13. Fisher, M. et al. Verifying autonomous systems. *Communications of the ACM* (2013).
14. Fleming, C.H. et al. Cyberphysical security through resiliency: A systems-centric approach. *Computer* (2021).
15. Frola, F.R. and Miller, C.O. System safety in aircraft acquisition. *Logistics Management Institute* (1984); https://perma.cc/QYV7-C5BY.
16. Hahn, E.M. et al. Synthesis for PCTL in parametric Markov decision processes. In *Proceedings of the 3rd NASA Intern. Symp. on Formal Methods, Lecture Notes in Computer Science*. Springer (2011).
17. Hosseini, S. et al. A review of definitions and measures of system resilience. *Reliability Engineering System Safety* (2016).
18. Junges, S. et al. *Parameter Synthesis for Markov Models* (2019); arXiv:1903.07993 [cs.LO].
19. Kaner, C. What is a good test case? *Conf. on Software Testing Analysis Rev. East* (2003).
20. Kimmelman, J. and London, A.J. The structure of clinical translation: Efficiency, information, and ethics. *Hastings Center Report* (2015).
21. Koopman, P. and Fratrik, F. How many operational design domains, objects, and events? In *Proceedings on the 33rd Conf. on Artificial Intelligence, Workshop on Artificial Intelligence Safety* (2019).
22. Kress-Gazit, H. et al. Formalizing and guaranteeing human-robot interaction. *Communications of the ACM.* (2021).
23. Lahijanian, M. et al. Temporal logic motion planning and control with probabilistic satisfaction guarantees. *IEEE Transactions on Robotics* (2012).
24. Lamport, L. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers.* Addison-Wesley (2002).
25. Lecomte, T. The bourgeois gentleman engineering and formal methods. In *Intern. Symp. on Formal Methods*. Springer (2019).
26. Leike, J. et al. *AI Safety Gridworlds* (2017); arXiv preprint arXiv:1711.09883.
27. Leveson, N.G. *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press (2011).
28. Leveson, N.G. et al. Requirements specification for process-control systems. *IEEE Transactions on Software Engineering* (1994).
29. Luckcuck, M. et al. Formal specification and verification of autonomous robotic systems: A survey. *ACM Comput. Surv.* (2019).
30. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley Sons (2014).
31. Sekhon, J. and Fleming, C. Towards improved testing for deep learning. In *Proceedings of the 41st Intern. Conf. on Software Engineering: New Ideas and Emerging Results*. IEEE/ACM (2019).
32. Sekhon, J. and Fleming, C. SCAN: A spatial context attentive network for joint multi-agent intent prediction. In *Proceedings of the 35th AAAI Conf. on Artificial Intelligence*. AAAI Press (2021).
33. Seshia, S.A. et al. Formal methods for semi-autonomous driving. In *Proceedings of the 52nd Annual Design Automation Conf.* ACM (2015).
34. Shah, S. et al. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Proceedings of the 11th Intern. Conf. on Field and Service Robotics.* Springer (2017).
35. Strafaci, A. What does BIM mean for civil engineers. *CE News Transportation* (2008).
36. Temizer, S. et al. Collision avoidance for unmanned aircraft using markov decision processes. In *Proceedings of the AIAA Guidance, Navigation, and Control Conf.* (2010).
37. Voas, J. and Schaffer, K. Whatever happened to formal methods for security? *Computer* (2016).
38. Webster, M. et al. A corroborative approach to verification and validation of human-robot teams. *Intern. J. Robotics Res* (2020).
39. Wing, J.M. A specifier's introduction to formal methods. *Computer* (1990).

Georgios Bakirtzis (bakirtzis@utexas.edu) is Peter O'Donnell Jr. postdoctoral fellow at The University of Texas at Austin, USA.

Steven Carr is a postdoctoral fellow at The University of Texas at Austin, USA.

David Danks is a professor of Data Science and Philosophy at the University of California, San Diego, USA.

Ufuk Topcu is associate professor of Aerospace Engineering and Engineering Mechanics at The University of Texas at Austin, USA.