Explainable Assurance through Compositional Verification with Cognitive Models

Parth Ganeriwala, Candice Chambers, and Siddhartha Bhattacharyya
Department of Electrical Engineering and Computer Science
Florida Institute of Technology
Melbourne, FL
{pganeriwala2022, chambersc2017}@my.fit.edu, sbhattacharyya@fit.edu

Junaid Babar
Applied Research & Technology
Collins Aerospace
Cedar Rapids, IA
junaid.babar@collins.com

***Abstract*—Cyber-physical systems involving human operators are increasingly being deployed in the safety- and security-critical domains. It has become essential to model human operator-system interactions and rigorously analyze whether human operators may be exposed to harm. In these contexts, verifying correctness alone is insufficient; outcomes of the analyses must be accompanied by explanations that are understandable and actionable. We present an approach for architectural modeling and compositional verification that integrates human cognitive models with system models to analyze operator–system interaction. The verification process produces explanation-ready artifacts, such as counterexample traces and proof obligations, that highlight why a property holds or fails and what this means for the human operator. These artifacts provide a bridge between formal verification results and human-understandable explanations, supporting safety-assurance and accountability. We demonstrate the approach through case studies in mixed-reality and increasingly autonomous systems, and show how verification outputs yield understandable and actionable explanations.**

***Index Terms*—Formal Methods, Mixed-Reality Systems, High Assurance.**

I. Introduction

The way in which humans interact with high-assurance systems is evolving. Mission-critical applications are emerging for mixed-reality (MR) and increasingly autonomous systems (IAS) [1], [2] that require formal analysis to assure properties such as security, functional correctness, and the absence of unintended behavior. However, due to the intimate coupling between these systems and their human operators, any formal analysis or explanation of the outcome will be incomplete without a relevant cognitive model of human behavior.

Cognitive models capture human behavioral responses to stimuli. Specifically, they represent human processes related to perception, action, memory, and reasoning [3] and are typically derived from empirical data collected from human subjects research studies. It is envisioned that coding cognitive models in a cognitive architecture is a good software engineering practice, as these architectures are built on general theories of cognition which are the building blocks of cognitive reasoning, as is a compiler for a programming language. A structured, architectural, logical reasoning of cognition, when embedded within cognitive architectures, serves as a structured framework that can be translated into formal verification environments to generate formal analysis outputs (e.g., counterexamples, invariants) with human-relevant explanations. Thus, they not only increase the realism of the analysis but also enable verification results to be communicated in ways that support trust, accountability, and decision-making. Our contributions presented in this paper include:

- A framework for integrating cognitive models into system architectures to yield explanation-ready analysis;
- An open-source Soar annex for the Architecture Analysis and Design Language (AADL), extending models with cognitive reasoning to structure explanatory traces;
- A Soar model analyzer based on nuXmv to verify cognitive guarantees and generate explanatory evidence;
- Demonstrating that the analysis yields actionable explanations through two case studies: MR and IAS.

This effort was sponsored by the Defense Advanced Research Projects Agency (DARPA) under agreement number HR0011-24-9-0439. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

II. Related Work

Compositional reasoning on architecture models has been demonstrated across multiple domains, with AADL [4] and AGREE [5] serving as enablers with their formal semantics and open-source tools. This approach scales system-level V&V by verifying component contracts and composing results—valuable for safety-critical autonomy [6], [7]. At the back end, model checkers such as *nuXmv* (deterministic) and *PRISM* (probabilistic) are widely used to verify temporal properties and quantify behavior under uncertainty [8].

Cognitive architectures provide a complementary lens for human-in-the-loop systems. *Soar* [9] and *ACT-R* [10] encode perception–memory–action–reasoning processes via interpretable structures (e.g., production rules and memory systems), enabling realistic models of operator behavior. Prior work has verified human–machine teaming architectures in AADL/AGREE and analyzed cognitive agents with model checking, including increasingly autonomous systems and MR contexts [11]–[15]. However, cognitive models are often verified separately from the architecture, risking inconsistencies and limiting the interpretability of results for stakeholders.

From an explainability perspective, most formal workflows produce proofs or counterexample traces that are not immediately actionable to human operators or certifiers. Few integrate a cognitive model directly into the architectural formalism and use the verification back end to yield explanation-ready artifacts. Our work addresses these gaps by (i) integrating a Soar cognitive model into AADL, (ii) automatically translating the composite model for verification in nuXmv, and (iii) treating invariants and counterexample traces as explanation-ready evidence. This aligns formal assurance with human-understandable explanations in MR and IAS use cases.

III. Methodology

We present MATRICS (Modeling and Analysis Toolkit for Realizable Intrinsic Cognitive Security), a framework for architecture–cognition modeling and explanation-ready formal assurance (Fig 1). MATRICS is tool-agnostic: it defines clear interfaces between system architectures, cognitive models, and verification back-ends so that (a) different cognitive architectures (e.g., Soar, ACT-R), (b) different architecture notations (e.g., AADL, SysML), and (c) different model checkers (e.g., nuXmv for symbolic/SMT, PRISM for probabilistic dynamics) can be swapped without altering the overall workflow. Here, we instantiate MATRICS using:

- AADL with AGREE contracts for system architecture,
- Soar cognitive architecture implemented as a Soar annex in OSATE (via Xtext) so that cognitive productions can be embedded within AADL components, and
- An automated translator that compiles the AADL + Soar model to nuXmv, checks LTL properties, and extracts explanation-based artifacts.

The Soar annex extends the AADL metamodel with productions as first-class clauses; ports and data features of AADL components as Soar WMEs (working-memory elements). Each backend (nuXmv, PRISM) uses a lightweight emitter that preserves the synchronous transition semantics; unit tests confirm equivalence between the Soar simulation trace and the emitted nuXmv trace. We demonstrate how the resulting artifacts serve as explanations that are understandable, usable, and actionable.

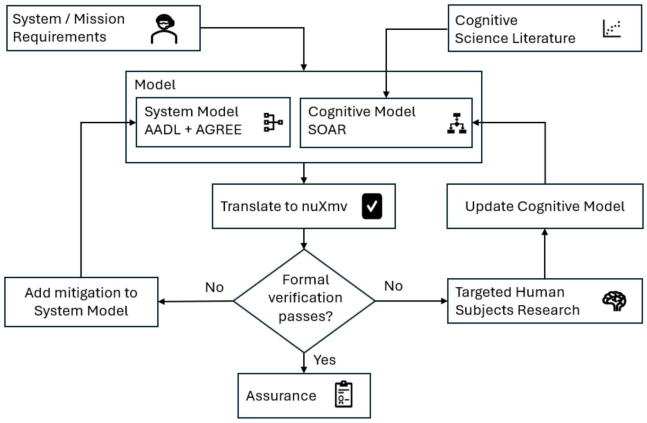

Fig. 1: Workflow for cognitive modeling and analysis.

We model the system architecture and the human cognition that interacts with it. We define a typed interface comprising percepts (what the human/agent observes), human/agent actions, and internal state/memory abstractions. This interface is independent of the application domain (e.g., clinical, aerospace, manufacturing, defense) and of the concrete cognitive architecture. The result is a composite model where human and system behaviors are explicit and analyzable together.

System-level requirements are captured as assume–guarantee contracts at component boundaries (safety, security, functional correctness), while cognitive guarantees describe bounds on human/agent behaviors (e.g., decision preconditions, response windows, attention/recall assumptions). MATRICS encourages expressing properties in a way that anticipates explanation: each formal property is associated with an explanation schema (who/what/why/what-next) that will later be populated from proofs or counterexamples.

The composite model is compiled to a verification-ready intermediate representation of a synchronous transition system (finite or infinite state, depending on data types). Cognitive rules are mapped to guarded transitions; architectural flows and contracts to environment and component constraints. This framework can support multiple back ends: nuXmv for symbolic model checking of LTL/CTL properties on infinite-state synchronous systems [16] or PRISM for probabilistic reasoning when uncertainty or stochastic effects are modeled [8]. This separation keeps the method general, while allowing domain-specific models to select appropriate solvers.

Beyond a pass/fail verdict, MATRICS extracts artifacts that support explanation. Counterexamples produced by the model checker are analyzed and summarized as temporally ordered sequences of architectural events and cognitive states. This process highlights the relevant transitions and identifies the state predicates and parameter valuations needed to reproduce the violation, thus providing a diagnostic account grounded in the model's semantics. For satisfied properties, the framework collects inductive invariants, discharged assumptions, and component-level contract-satisfaction evidence. These proof artifacts document why a property holds under the stated assumptions and attribute the supporting evidence to specific model components and contracts. All extracted artifacts are recorded with traceability links to model elements such as user-inputs, contracts, and production rules, making them suitable for integration into assurance cases and simulation/-training environments for scenario-based what-if analysis.

Using assume-guarantee reasoning with AGREE, leaf-level results (incl. cognitive guarantees) are propagated up the hierarchy to justify system-level claims. MATRICS maintains traceability from each system claim to (i) the components that contribute evidence, and (ii) the cognitive decisions relevant to the claim. Finally, MATRICS enables explanations based on contracts satisfied/violated, counterexamples and traces.

Section IV applies our prototype instantiation of MATRICS to (i) mixed reality and (ii) increasingly autonomous systems; the same workflow can be applied unchanged to other applications by swapping domain libraries (percepts/actions) and

properties, or by selecting a different verification backend where uncertainty is central.

We treat explanations as the byproduct of aligning verification outcomes with the model's interface requirements—i.e., the typed ports, modes, and assume–guarantee contracts that define each component boundary. For any proof or counterexample, we trace which interface elements it touches and report (i) the requirement(s) that were satisfied or violated, (ii) the specific interface signals and modes involved, and (iii) the minimal condition that triggers remediation (what to change at the interface to restore the requirement). This keeps explanations actionable (what next), auditable (bound to contracts), and shared across stakeholders (operators see signals/modes; certifiers see contracts/assumptions).

IV. Use Cases

We present two use cases to demonstrate the application of MATRICS to formally verify system models that include cognitive behavioral specifications.

A. Cybersickness Protection in Mixed-Reality Systems

For this use case, we wish to develop a helmet-mounted display (HMD) application that can be used by a border guard in an observation tower to identify people carrying suspicious packages. A preliminary cybersecurity assessment reveals that if an adversary could induce nausea (e.g., by taxing the HMD processor to increase latency and decrease frame rate), the guard would be forced to remove the HMD, thereby decreasing the likelihood of detecting malicious agents. We therefore design our HMD system to be robust to such physiological attacks. In order to prove guarantees of cybersickness protection in our design, we must analyze our HMD system model in combination with a model of cybersickness onset to determine whether the system is composed in such a way that inducing cybersickness could be possible. Note that for this use case we only model aspects of the HMD and human cognition that are relevant to the analysis.

Our AADL model included HMD and Operator components annotated with AGREE contracts (Fig 2). The HMD Display contract guarantees a frame rate in the range [0, 60], while the Operator contract assumes that frame rates above 40 fps prevent cybersickness. This composition immediately exposes two violations: (i) AGREE cannot verify that 40 fps is sufficient to avoid cybersickness, and (ii) the display does not guarantee that fps ≥ 40. To address (i), we incorporated a Soar cognitive model capturing cybersickness triggers, which was then translated to nuXmv for analysis. This allowed AGREE to discharge the operator contract using the formal analysis results from Soar, i.e., when fps ≥ 40, no counterexample is produced. The resulting artifact is directly usable: "maintaining frame rates ≥ 40 fps ensures that the operator remains safe", and provides an explanation for the real-time system-behavior to the human operator.

1) Property Verification: Once the models were generated in nuXmv, nominal behavior (i.e., if $40 \leq$ fps ≤ 60, the Operator should not experience incapacitating cybersickness) was verified before off-nominal behavior was verified. This is captured as the following nuXmv specification: `LTLSPEC G (cl != incapacitated-level-III)` which verifies and serves as the explanation for nominal behavior.

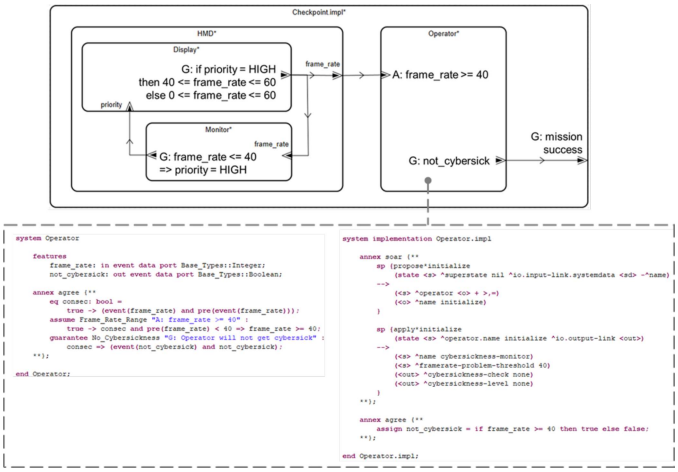

Fig. 2: Architectural Model with a frame rate monitor along with the Soar and AGREE annexes for the Operator.

We specify three temporal property classes. Reachability requires the system to remain within valid states at all times, expressed as `LTLSPEC G (state_name ≠ nil ∧ state_name ≠ dm)`, meaning the state name is never `nil` nor `demographic_monitor` simultaneously. Safety prevents unsafe or unintended states and enforces constraints to avoid invalid cybersickness levels unless specific violations occur; formally, `LTLSPEC G (¬ p)` where the bad state (p) is ((op ≠ apply-latency-violated ∧ cl = incapacitated-level-III)), with (op = state_operator_name) and (cl = state_io_cybersickness_level). Liveness ensures expected transitions under normal conditions and constrains cybersickness evolution: `LTLSPEC (cl ≠ level3) U (op = apply-latency-violated)` and `LTLSPEC G (cl = level3 → X G (cl = level2))`.

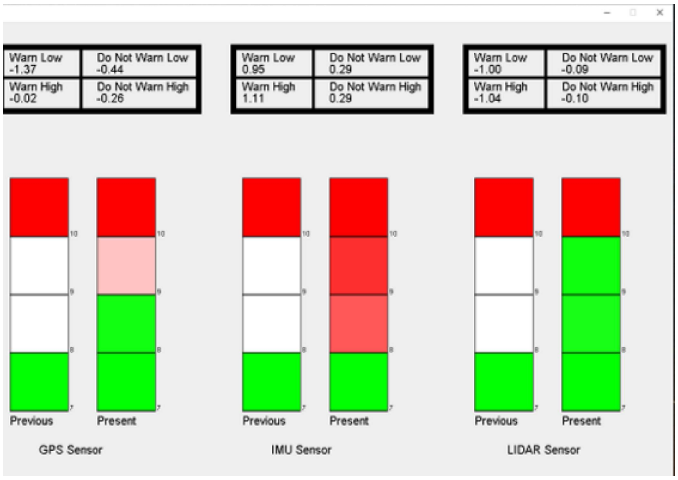

Fig. 3: Explainable interface for Reinforcement Learning

When the model checker finds that $\mathbf{G}(\texttt{fps} \geq 40 \rightarrow \texttt{cl} \neq \texttt{incapacitated-level-III})$ holds, we report it at the interface: `fps` (display output port), `cl` (operator state output), and the *display-in-nominal* contract. If a counterexample arises, the explanation names the interface elements, the current mode, and the condition that fixes the violation (e.g., "raise `fps` to $\geq$40 for 5 steps"). Thus the *why* and *what-next* are expressed only in terms of ports, modes, and contracts.

B. IAS with Autonomous Pilot Operator

The Autonomous Pilot Operator (APO) is an integral component of the Assured Human-Machine Interface for Increasingly Autonomous Systems (AHMIIAS) framework designed to ensure adaptive decision-making and safety assurance in aviation environments [11], [13]–[15]. The APO serves as an intelligent cognitive agent that facilitates decision making based on multi-sensor input, fault detection, and dynamic role allocation between human pilots and increasingly autonomous systems (IAS). It also learns pilot behavior with exploration using reinforcement learning. This is explained on an interface designed with bar graphs that show that the cognitive agent learned to warn (red) or not to warn (green) color when sensor error is in the range 7-9. The integration of the cognitive agent with Xplane provides the explainability required to visually display what the cognitive agent has learned (Fig 3).

Using AADL with AGREE contracts, we specified assumptions on sensor reliability and guarantees for safe throttle control and flight mode transitions. The APO's cognitive logic was encoded in Soar and integrated into the architecture via our Soar annex. The composite model was translated into nuXmv, where we specified temporal properties and extracted explanation artifacts to ensure that the APO's learned decisions aligned with safety constraints.

1) Property Verification: To verify these properties, we specify key system behaviors using temporal logic and categorize them as discussed in the MR property verification.

Similar to the MR properties, we formalize properties in the three different categories. Reachability for instance, `LTLSPEC F` (s_op_name = transition), indicating that the operator state (state_operator_name) will at some point make a transition. Safety properties such as enforcing state_io_throttle ≤ 1.0 so the throttle never exceeds its maximum. Liveness; for example, `LTLSPEC F` ($s\_throttle < 1.00 \rightarrow$ state_fm = horizontal), meaning that whenever the throttle (state_io_throttle) is below 1.00, the flight mode (state_flight-mode) will eventually be `horizontal`.

For throttle safety $\mathbf{G}(\texttt{state\_io\_throttle} \leq 1.0)$ and workload-aware warnings $\mathbf{G}(\texttt{warn} \leftrightarrow \texttt{err} \in [7, 9])$, explanations reference interface signals (`state_io_throttle`, `warn`, sensor error), the active flight mode, and the component contract that bound them. Proofs cite which contracts and assumptions discharge the requirement; violations present the interface condition that corrects behavior.

The two case studies illustrate how MATRICS can be applied across distinct domains with different operational challenges, and demonstrate its generalizability.

V. Conclusion

We have presented MATRICS, a framework that integrates cognitive models with system architectures to enable formal analysis and the extraction of explanation-ready artifacts. Our explanations are generated by mapping proofs and counterexamples to ports, modes, and contracts, so operators and certifiers see the same evidence through the system's interface. Further research is needed to establish automated validation techniques that account for system and cognitive behaviors and developing modeling techniques that capture cognitive details succinctly while preserving interpretability, and scaling verification to account for probabilistic factors.

References

[1] I. M. Gregory and et al., "Intelligent contingency management for urban air mobility." in *Dynamic Data Driven Applications Systems. DDDAS 2020. Lecture Notes in Computer Science*, 2020.

[2] M. Shakeri, A. Sadeghi, and S.-M. Choi, "Augmented reality-based border management," *Virtual Reality*, vol. 26, pp. 1–21, 01 2022.

[3] ISO/IEC JTC 1, *Information technology — Vocabulary*. International Standards Organization, May 2015, vol. 1.

[4] P. H. Feiler and D. P. Gluch, *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language*, 1st ed. Addison-Wesley Professional, 2012.

[5] M. W. Whalen, A. Gacek, D. Cofer, A. Murugesan, M. P. E. Heimdahl, and S. Rayadurgam, "Your "what" is my "how": Iteration and hierarchy in system design," *IEEE Software*, vol. 30, no. 2, pp. 54–60, 2013.

[6] J. B. Michael, R. Riehle, and M.-T. Shing, "The verification and validation of software architecture for systems of systems," in *2009 IEEE International Conference on System of Systems Engineering (SoSE)*. IEEE, 2009, pp. 1–6.

[7] M. O'Connor, S. Tangirala, R. Kumar, S. Bhattacharyya, S. Sznaier, and L. Holloway, "A bottom-up approach to verification of hybrid model-based hierarchical controllers with application to underwater vehicles," in *2006 American Control Conference*, 2006, pp. 6 pp.–.

[8] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of Probabilistic Real-Time Systems," in *Computer Aided Verification*, G. Gopalakrishnan and S. Qadeer, Eds. Berlin, Heidelberg: Springer, 2011, pp. 585–591.

[9] J. E. Laird, A. Newell, and P. S. Rosenbloom, "Soar: An architecture for general intelligence," *Artificial Intelligence*, vol. 33, no. 1, pp. 1–64, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370287900506

[10] J. R. Anderson, *The adaptive character of thought*. Psychology Press, 2013.

[11] S. Bhattacharyya, J. Davis, A. Gupta, N. Narayan, and M. Matessa, "Assuring increasingly autonomous systems in human-machine teams: An urban air mobility case study," vol. 348, pp. 150–166, oct 2021. [Online]. Available: https://doi.org/10.4204%2Feptcs.348.11

[12] N. Neogi, S. Bhattacharyya, D. Griessler, H. Kiran, and M. Carvalho, "Assuring intelligent systems: Contingency management for uas," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 6028–6038, 2021.

[13] N. Narayan, P. Ganeriwala, R. M. Jones, M. Matessa, S. Bhattacharyya, J. Davis, H. Purohit, and S. F. Rollini, "Assuring learning-enabled increasingly autonomous systems*," in *2023 IEEE International Systems Conference (SysCon)*, 2023, pp. 1–7.

[14] P. Ganeriwala, M. Matessa, S. Bhattacharyya, R. M. Jones, J. Davis, P. Kaur, S. F. Rollini, and N. Neogi, "Design and validation of learning aware hmi for learning-enabled increasingly autonomous systems," in *2025 IEEE International systems Conference (SysCon)*, 2025, pp. 1–8.

[15] P. Ganeriwala, N. Narayan, R. M. Jones, M. Matessa, S. Bhattacharyya, J. Davis, S. F. Rollini, H. Purohit, and N. Neogi, "Systems engineering with architecture modeling, formal verification, and human interactions for learning-enabled autonomous agent," *Systems Engineering*, 2025.

[16] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "Nusmv: A new symbolic model verifier," in *Computer Aided Verification: 11th International Conference, CAV'99 Trento, Italy, July 6–10, 1999 Proceedings 11*. Springer, 1999, pp. 495–499.