

Towards Trustworthy Integration of Generative AI in the MBSE Development Lifecycle



Collins Aerospace
An **RTX** Business

Amer Tahat, Isaac Amundson, David Hardin, Darren Cofer
25th Annual High Confidence Software and Systems (HCSS 2025)
May 14, 2025

DARPA PROVERS

Pipelined Reasoning of Verifiers Enabling Robust Systems

Develop automated, scalable **formal methods** tools that are integrated into *traditional* development pipelines

Enable *traditional* product engineers to incrementally produce and maintain **high-assurance** national security systems



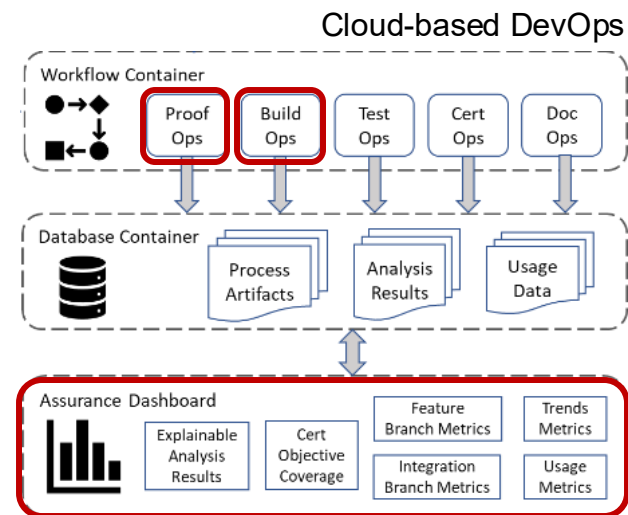
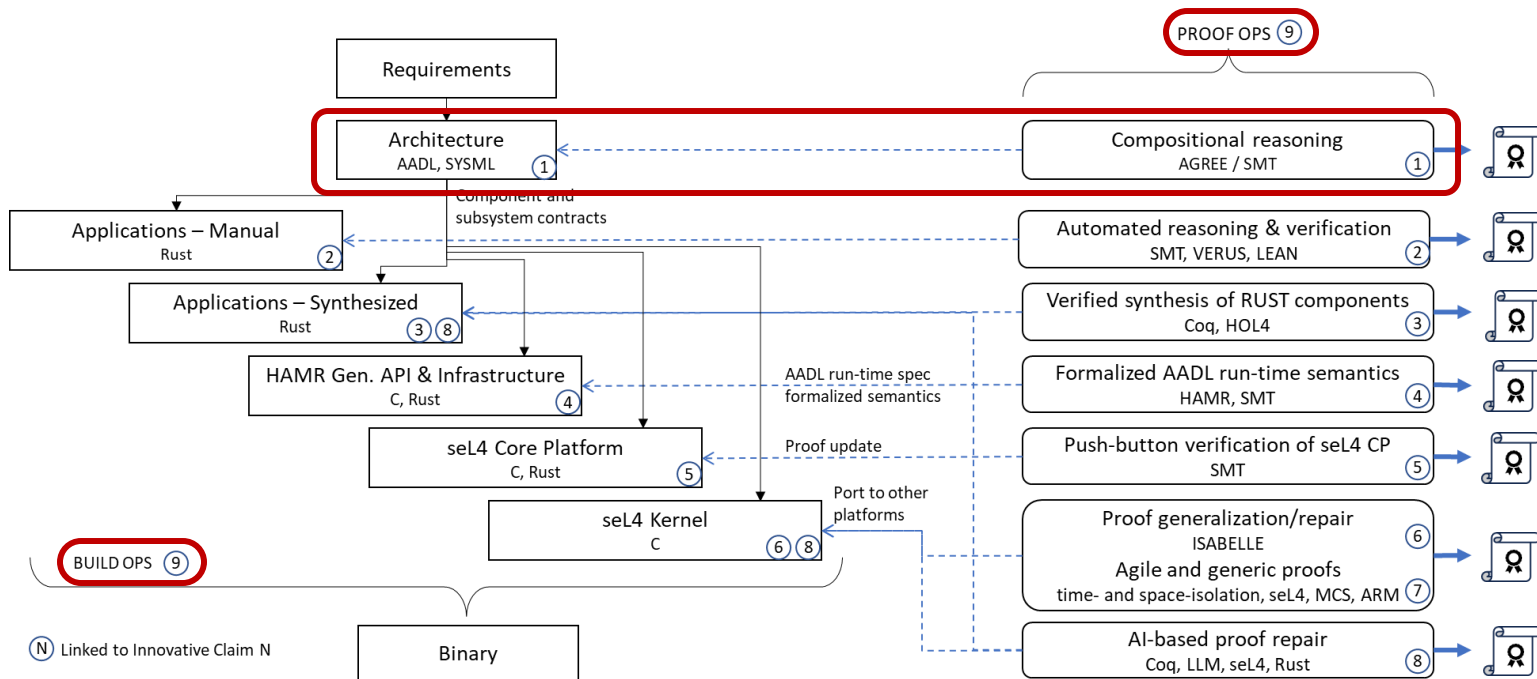
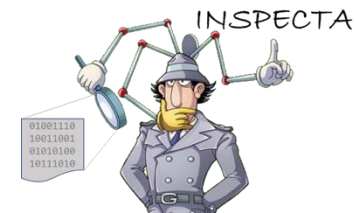
Barriers to formal methods adoption

- Scalability to support real-world systems
- The UI targets formal-methods experts, creating a learning curve for others.
- Lack of commercially licensed / supported tools
- Formal methods skillset required
 - Property specification language
 - Explainable counterexamples

GenAI's Potential: Accelerates system development — but trust, correctness, and consistency must be non-negotiable.

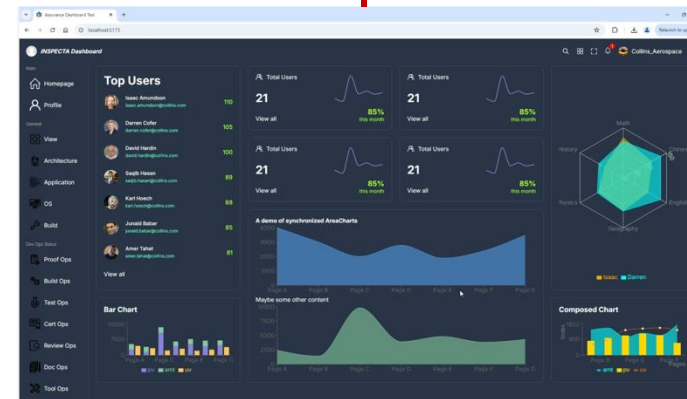
INSPECTA

Industrial-Scale Proof Engineering for Critical Trustworthy Applications



Role of AGREE Copilot (AGREE-Dog):

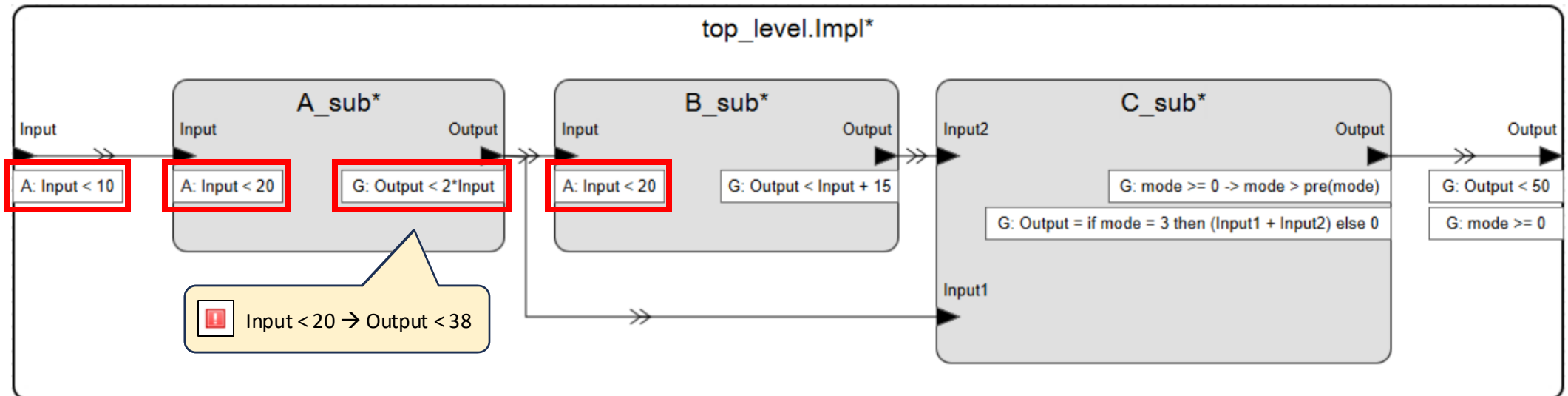
→ Creates a trust fabric for AI-accelerated MBSE engineering.



Compositional Reasoning with AGREE

Assume Guarantee Reasoning Environment

- Assume-Guarantee annex for AADL architecture models
 - Assumptions describe the expectations that a component has on the environment
 - Guarantees describe bounds on the behavior of the component when assumptions are valid
- Compositional analysis to prove correctness of:
 - Component interfaces (component assumptions are satisfied by upstream guarantees)
 - Component implementations (component assumptions and subcomponent guarantees satisfy guarantees)



AGREE Counterexamples

Property	Result
Verification for Car.Car_Impl	1 Invalid, 29 Valid
Contract Guarantees	1 Invalid, 2 Valid
Subcomponent Assumptions	Valid (0s)
[G_car_1] actual speed is less than constant target speed	Invalid (0s)
[G_car_2] acceleration is limited	Valid (0s)
> This component consistent	Valid (0s)
> THROT consistent	Valid (0s)
> CNTRL consistent	Valid (0s)
> AXL consistent	Valid (0s)
> SM consistent	Valid (0s)
> Component composition consistent	Valid (0s)
> Verification for THROT	Valid (0s)
> Verification for CNTRL	Valid (0s)
> Verification for AXL	Valid (0s)
> Verification for SM	Valid (0s)

Counterexample			
Variables for the selected component implementation			
Variable Name	0	1	2
Inputs:			
{Target_Speed.val}	1499/10	0	0
{Target_Tire_Pitch.val}	-1/10	3/10	-1/10
State:			
{[G_car_1] actual speed is less than constant target speed}	true	true	false
{_TOP.AXL..ASSUME.HIST}	true	true	true
{_TOP.CNTRL..ASSUME.HIST}	true	true	true
{_TOP.SM..ASSUME.HIST}	true	true	true
{_TOP.THROT..ASSUME.HIST}	true	true	true
{const_tar_speed}	true	false	true
Outputs:			
{Actual_Speed.val}	1499/110	1499/121	14990/1331
{Actual_Tire_Pitch.val}	-1/10	1/5	-1/10
{State_Signal.val}	0	0	0

Variables for THROT			
Variable Name	0	1	2
Inputs:			
{THROT.Actuator_Input}	1499/11	-1499/121	-14990/1331
State:			
{THROT..ASSUME.HIST}	true	true	true
Outputs:			
{THROT.Actual.val}	1499/110	1499/121	14990/1331

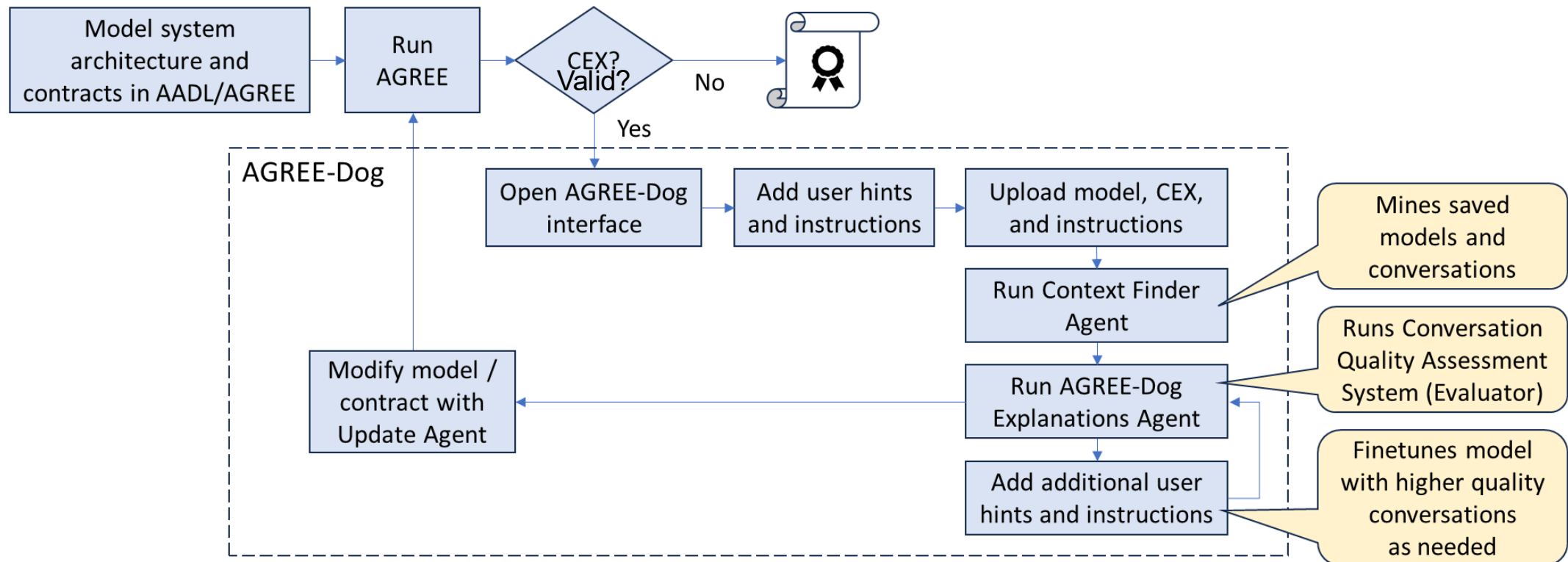
How can we improve AGREE usability by making counterexamples more *actionable*?

Actionable: Determine and address the root cause of the counterexample more *quickly*


Before AgreeDog: costly and error-prone manual interpretation of counterexamples, requirements, logs, and models — sometimes taking **days or weeks** to complete repairs!

Toward Trusted Gen AI in the MBSE Lifecycle

- Early trials of **AGREE-Dog**—have reduced system-level repair cycles from **days/weeks to minutes/seconds**, delivering substantial time savings.
- Powered by GPT-4o, O3, and GPT-4.1 multimodal models.



Toward Trusted Gen AI in the MBSE Lifecycle



AGREE-Dog

Context from files loaded. Add any additional questions or requests here...

Tokens used: 0

Elapsed Time: 00:00:00.00

Copilot Logs

- [20:31:46] INFO: Requirement file provided: ./sys_requirements.csv
- [20:31:46] INFO: Contents of sys_requirement.txt loaded successfully.
- [20:31:46] INFO: Starting the Dash server...
- [20:31:51] INFO: Re-read AADL model file: /home/amertahat/AgreeDog/uploaded_dir/Car/packages/Car.aadl
- [20:31:51] INFO: Re-read requirements file: ./sys_requirements.csv
- [20:31:51] INFO: New counterexample detected: cex10807709049446947555.xls
- [20:31:51] INFO: New counterexample file detected: cex10807709049446947555.xls. Prompt updated with latest files.



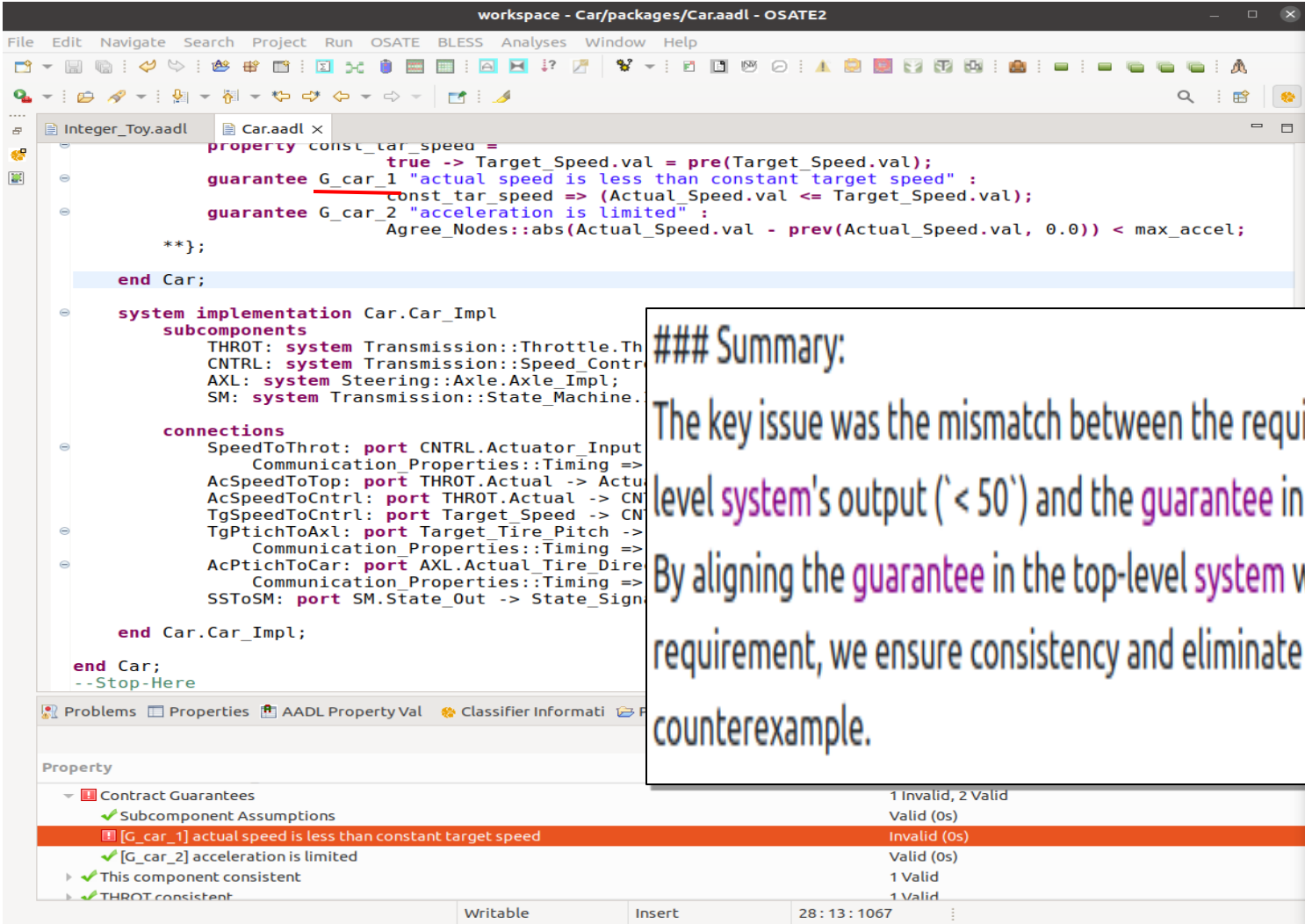
AGREE-Dog

Kind SMTSolvers AI selector
 AgreeDog System Message

Load additional context Use Osate2
 Include import chain Don't include import chain
 GPT-O3 reasoning (128k tk) GPT-4.1 multi-modal (1 m tk) GPT-4o
 GPT-4 (8k tk)
 Full History Last Response
 Enable Git push Disable Git push

Advanced settings menu

AGREE-Dog: Counterexample Analysis and Resolution



```

workspace - Car/packages/Car.aadl - OSATE2
File Edit Navigate Search Project Run OSATE BLESS Analyses Window Help
Integer_Toy.aadl Car.aadl x
property const_car_speed =
    true -> Target_Speed.val = pre(Target_Speed.val);
    guarantee G_car_1 "actual speed is less than constant target speed" :
        const_car_speed => (Actual_Speed.val <= Target_Speed.val);
    guarantee G_car_2 "acceleration is limited" :
        Agree_Nodes::abs(Actual_Speed.val - prev(Actual_Speed.val, 0.0)) < max_accel;
**};
end Car;

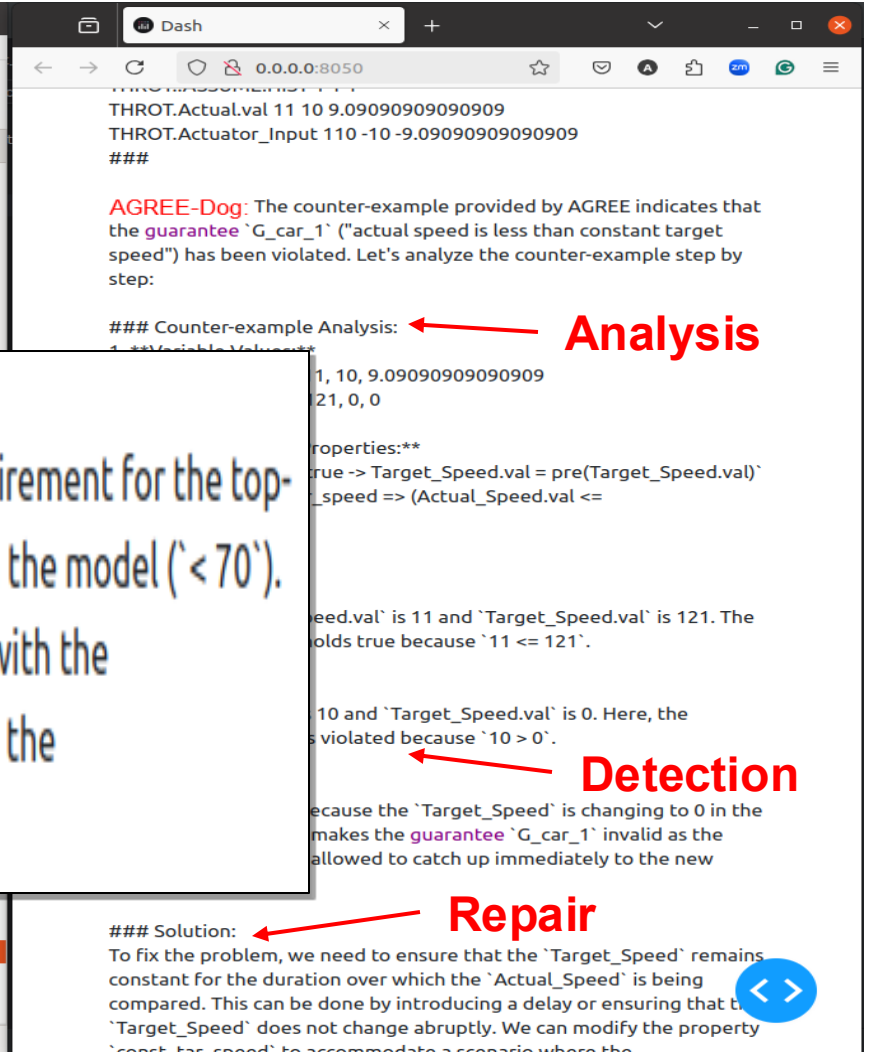
system implementation Car.Car_Impl
subcomponents
    THROT: system Transmission::Throttle.Th
    CNTRL: system Transmission::Speed_Contr
    AXL: system Steering::Axle.Axle_Impl;
    SM: system Transmission::State_Machine.

connections
    SpeedToThrot: port CNTRL.Actuator_Input
        Communication_Properties::Timing =>
    AcSpeedToTop: port THROT.Actual -> Actu
    AcSpeedToCntrl: port THROT.Actual -> CN
    TgSpeedToCntrl: port Target_Speed -> CN
    TgPtichToAxl: port Target_Tire_Pitch ->
        Communication_Properties::Timing =>
    AcPtichToCar: port AXL.Actual_Tire_Dire
        Communication_Properties::Timing =>
    SStoSM: port SM.State_Out -> State_Sign

end Car.Car_Impl;
end Car;
--Stop-Here

Property
Contract Guarantees
    ✓ Subcomponent Assumptions
    ✗ [G_car_1] actual speed is less than constant target speed
    ✓ [G_car_2] acceleration is limited
    ✓ This component consistent
    ✓ THROT.consistent
1 Invalid, 2 Valid
Valid (0s)
Invalid (0s)
Valid (0s)
1 Valid
1 Valid

```



```

Dash
0.0.0.0:8050
THROT.Actual.val 11 10 9.0909090909090909
THROT.Actuator_Input 110 -10 -9.0909090909090909
###
AGREE-Dog: The counter-example provided by AGREE indicates that the guarantee `G_car_1` ("actual speed is less than constant target speed") has been violated. Let's analyze the counter-example step by step:

### Counter-example Analysis:
1. 10, 9.0909090909090909
2. 10, 0
properties:**
true -> Target_Speed.val = pre(Target_Speed.val)
_speed => (Actual_Speed.val <=

peed.val` is 11 and `Target_Speed.val` is 121. The
olds true because `11 <= 121`.

10 and `Target_Speed.val` is 0. Here, the
s violated because `10 > 0`.

because the `Target_Speed` is changing to 0 in the
makes the guarantee `G_car_1` invalid as the
allowed to catch up immediately to the new

### Solution:
To fix the problem, we need to ensure that the `Target_Speed` remains constant for the duration over which the `Actual_Speed` is being compared. This can be done by introducing a delay or ensuring that the `Target_Speed` does not change abruptly. We can modify the property `const_car_speed` to accommodate a scenario where the

```

Analysis (points to Counter-example Analysis)

Detection (points to violation step)

Repair (points to Solution)

Summary:

The key issue was the mismatch between the requirement for the top-level system's output (< 50) and the guarantee in the model (< 70). By aligning the guarantee in the top-level system with the requirement, we ensure consistency and eliminate the counterexample.

Toward Trusted Gen AI in the MBSE Lifecycle

```

ree_Nodes.aad
0 as: - Reads project + NL sys requirements +
1 CEX + agree.log
2
3
4
5 as:
6 - Explains violation in natural language
7
8
9
10 gu: - Suggests repairs
11
12
13 gu:
14 - 1-Click application → full AGREE re-check
15 → Git push
16
17 **)
18
19 end Car;
20
21 system implementation Car.Car_Impl
22 subcomponents
23 THROT: system Transmission::Throttle.Throttle_Impl;
24 CNTRL: system Transmission::Speed_Control.Speed_Control_Impl;
25 AXL: system Steering::Axle.Axle_Impl;
26 SM: system Transmission::State_Machine.State_Machine_Impl;
    
```

Property	Result
Contract Guarantees	3 Valid
This component consistent	1 Valid
THROT consistent	1 Valid
CNTRL consistent	1 Valid
AXL consistent	1 Valid
SM consistent	1 Valid
Component composition consistent	1 Valid
Verification for THROT	4 Valid
Verification for CNTRL	4 Valid
Verification for AXL	4 Valid
Verification for SM	9 Valid

AGREE-Dog

Context from files loaded. Add any additional questions or requests here...

Submit with Context Save Feedback Loop Insert

AGREE-Dog: Happy to hear everything verifies cleanly! 🐶
 If you need more help tweaking contracts or exploring additional properties, just let

(Woof! 🐶)

Tokens used: 4756

Elapsed Time: 0:00:04.37, Total API Call Time: 32.29 seconds

[19:53:14] INFO: Successfully updated /home/amertahat/AgreeDog/uploaded_dir/Car/updated file. If you're not happy with these changes, please revert via your IDE's Git

Copilot Logs

- [19:52:28] INFO: Combined global context prompt with user's additional input
- [19:53:14] INFO: Successfully updated /home/amertahat/AgreeDog/uploaded_dir/Car/updated file. If you're not happy with these changes, please revert via your IDE's Git
- [19:53:38] INFO: Start file '/home/amertahat/AgreeDog/uploaded_dir/Car/package
- [19:53:38] INFO: Read agree.log data
- [19:53:38] INFO: Great, all AGREE results are valid and no counterexamples were
- [19:54:27] INFO: Combined global context prompt with user's additional input

Enter commit message

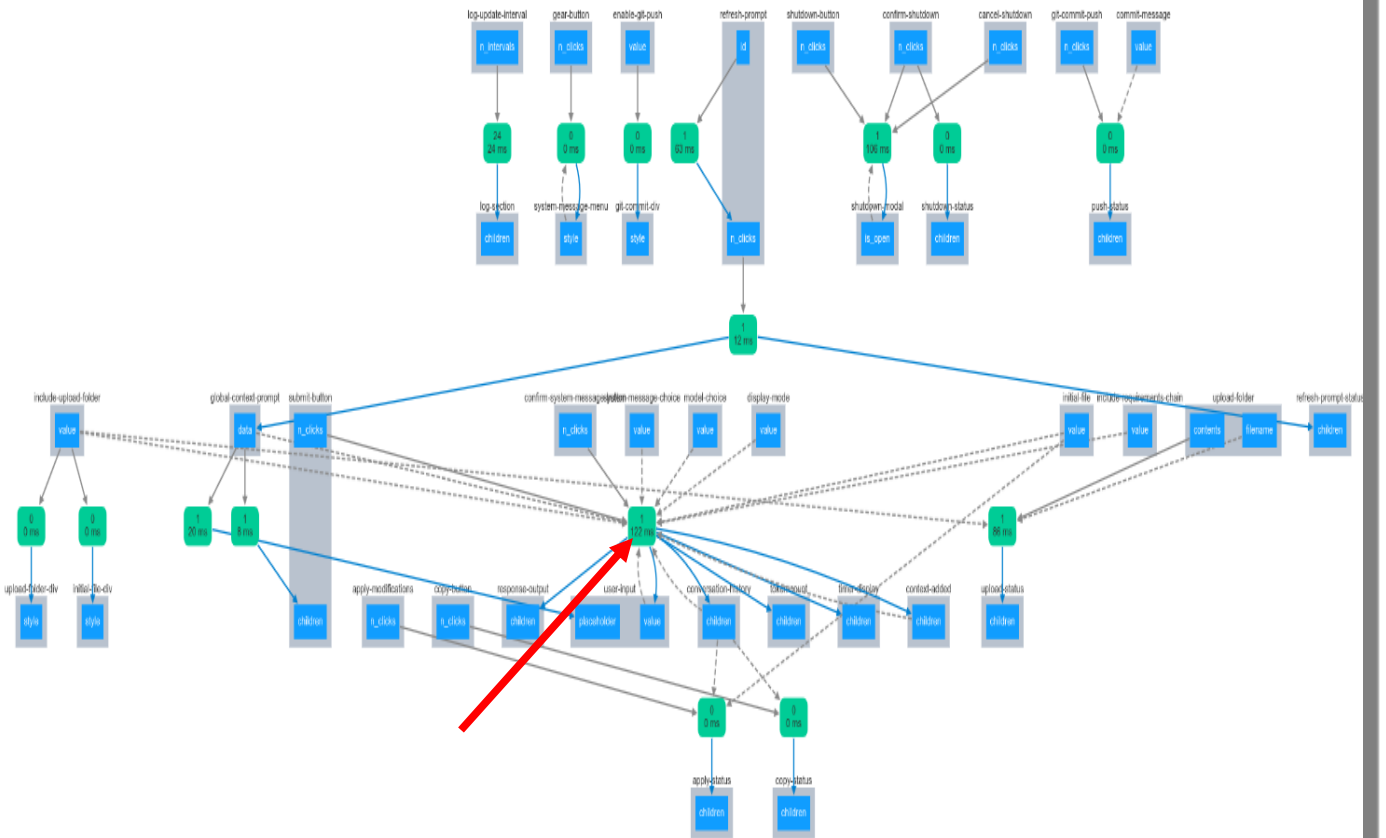
Git Commit and Push

AgreeDog System Call-Graph Complexity

Traditional Workflow: Tens of interdependent calls & manual back-and-forth

AGREE-Dog Approach:

- ✓ Collapses complex interactions into a single recommendation step
 - ✓ Minimizes manual navigation of call-graphs and dependencies
- Dramatic reduction in user effort via push-button UI.



AGREE-Dog significantly automates **16 DevOp and ProofOp** → drastic user effort reduction

Ongoing Experimental Evaluations:

- **Test bed:** We created 13 test scenarios for an autonomous-vehicle (Car) AADL model—each scenario comprising 400 LoC models, <100 LoC CEX, <100 LoC sys requirements, and agree Logs. Total ~**10,000** LoC
- × **Fault Injection:** We injected faults into three key components (Top-level Control System, Steering, and Transmission) and ran a full **AGREE-Dog** repair “conversation” for each.

Key Impact & Scalability

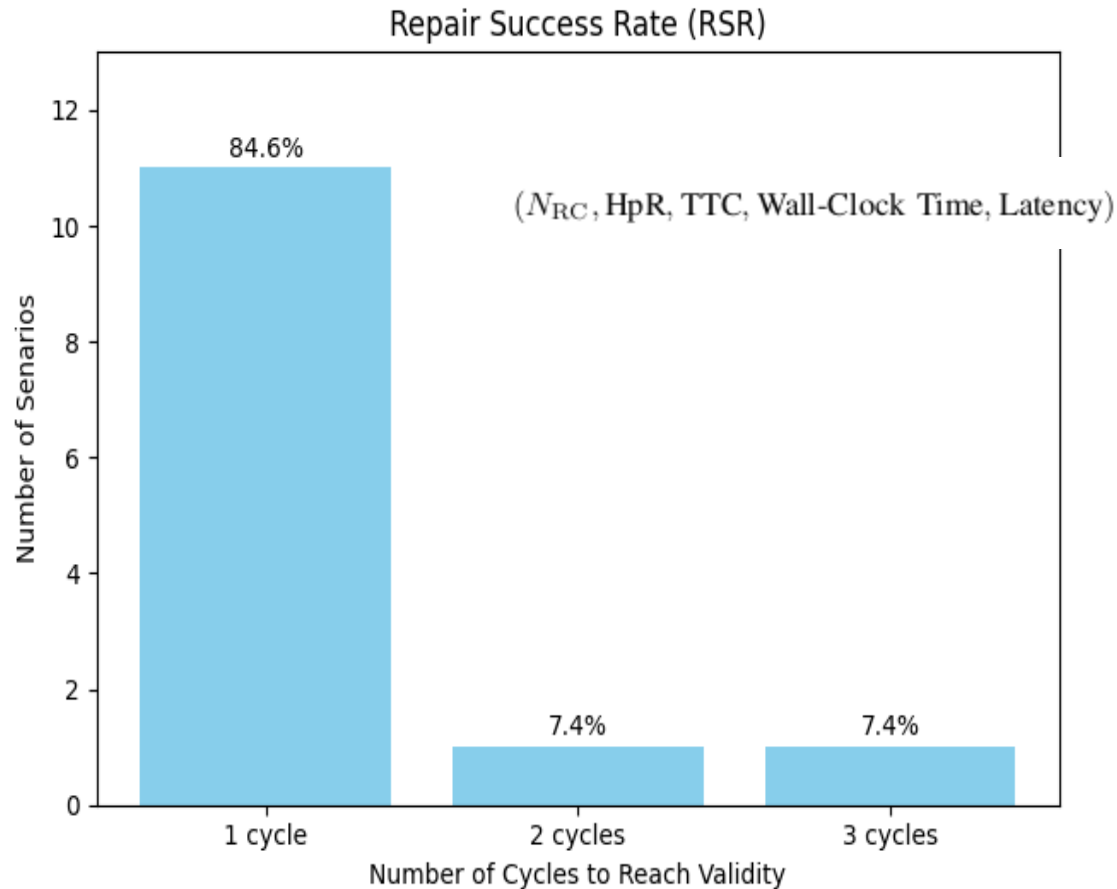


TABLE I: **Structural Metrics Summary.** Token values rounded to the nearest hundred.

Metric	Result
System Validity	100%
Repair Success Rate (Fig. 5)	11/13 (84.6%) in 1 cycle; 1/13 in 2 cycles; 1/13 in 3 cycles
Human Input Ratio (HpR)	< 0.1% of total tokens
AGREE-Dog Input	> 99.9% of total tokens
Token Use (per test suite)	4.8k, 5.5k, 22k tokens

TABLE II: **Temporal Metrics Summary.** Time values rounded to the nearest second.

Metric	Result
Wall-Clock Time (WCT)	Mean: 2:09 min; Median: 1:39 min
Latency (per cycle)	Mean: 22 s; Range: 4–33 s

- ✓ **Key Observation:** Human input dropped below 0.1% of the workflow—yet we still achieved 100% system-wide validity, demonstrating a dramatic reduction in manual effort.

Challenges

- Contextual constraints
 - GenAI struggles with **accurate** AGREE model explanations due to **limited** context, causing hallucinations
 - AGREE-Dog Retrieval-Augmented Generation (RAG) System **adjusts context dynamically**, reducing hallucinations and improving explanation accuracy.
- Model repair
 - Repairs can generate multiple solutions, some of which violate system requirements
 - AGREE-Dog **deploys Agree.log validity analysis** and follows predefined requirements to ensure repairs respect system constraints. (**MMU is useful**)
- Evaluation
 - Manual evaluation limits scalability for large models
 - A Conversational Quality Assessment System (CQAS) automates performance tracking

Current Limitations

- Manual Solver Selection
 - Users must still choose between JKind, SMT solvers.
 - We are developing a **multi-agent router** that intelligently selects solvers based on context, performance, and explainability.
- Model repair
 - We are testing multiple-root causes for cex generation, across more complex systems.
 - The AGREE-Dog more automated navigation between system files.
 - Judge Worker orchestration: feasibility study.
- RL Integration In Progress
 - Repair adaptation via reinforcement learning is in early stage.
 - We are incorporating conversation history into our recommendation system and fine-tuning datasets.

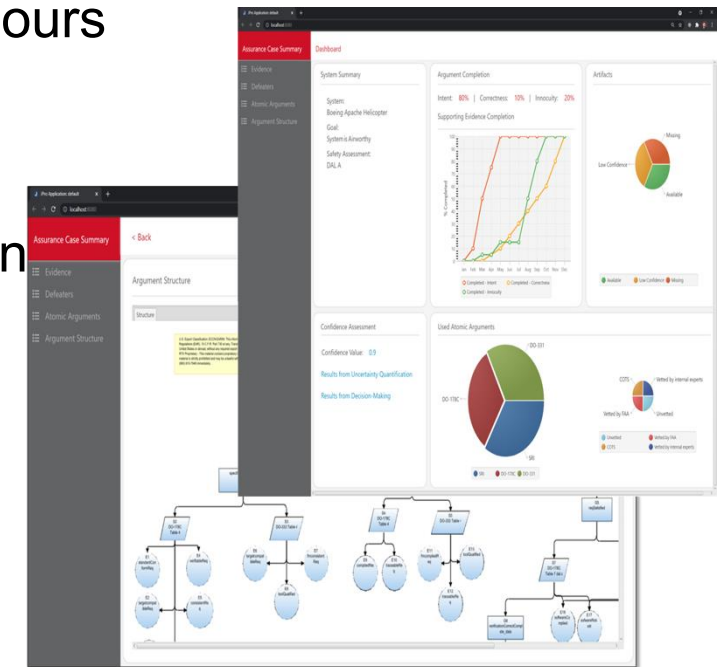
Conclusions

- **Summary**

- AGREE-Dog: Push-Button • System-Wide Formally Valid Repairs • Explainable
- Traceable • Rapid • Human-in-the-Loop
- Validation cycles reduced from days/weeks to minutes/hours

- **Up Next:**

- Federated/**Organizational agents** & solver orchestration
- CQAS (evaluation metrics)
- SysMLv2 migration: Embed AGREE-Dog like Copilot into next-gen modeling environments
- Assurance patterns dashboard



Resources

- AGREE-Dog Video Demo:

 <https://www.youtube.com/watch?v=2cKRm9fuifs>

- INSPECTA project page:

 <https://loonwerks.com/projects/inspecta.html>

- AGREE-Dog source code:

 <https://github.com/loonwerks/AgreeDog>

- AGREE-Dog-Plugin source code:

 <https://github.com/loonwerks/AgreeDog-plugin.git>

Thank You!

This work was funded by DARPA contract FA8750-24-9-1000.

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.

