EMQ

AGL All Member Meeting Japan 2026 · Tokyo · May 13–14

From SDV to AIDV: Closing the Gap with LLM Fuzzing and Edge AI Agents

From Deep-State Protocol Security to AI-Native Vehicle Intelligence

SESSION Chapters

① 啓発

Murata's Question

How to design and implement the SDV software-first principles?

② 変化

The Quality-Speed Dilemma

Why traditional testing breaks at SDV iteration

③ 実践

AI-In-The-Loop Fuzzing Framework

Combining Static Documentations with Dynamic AI for SDV

④ 改善

VeloRT + VeloFlux SoDeV Integration

Continuous quality enhancement in the Real-Time SDV loop

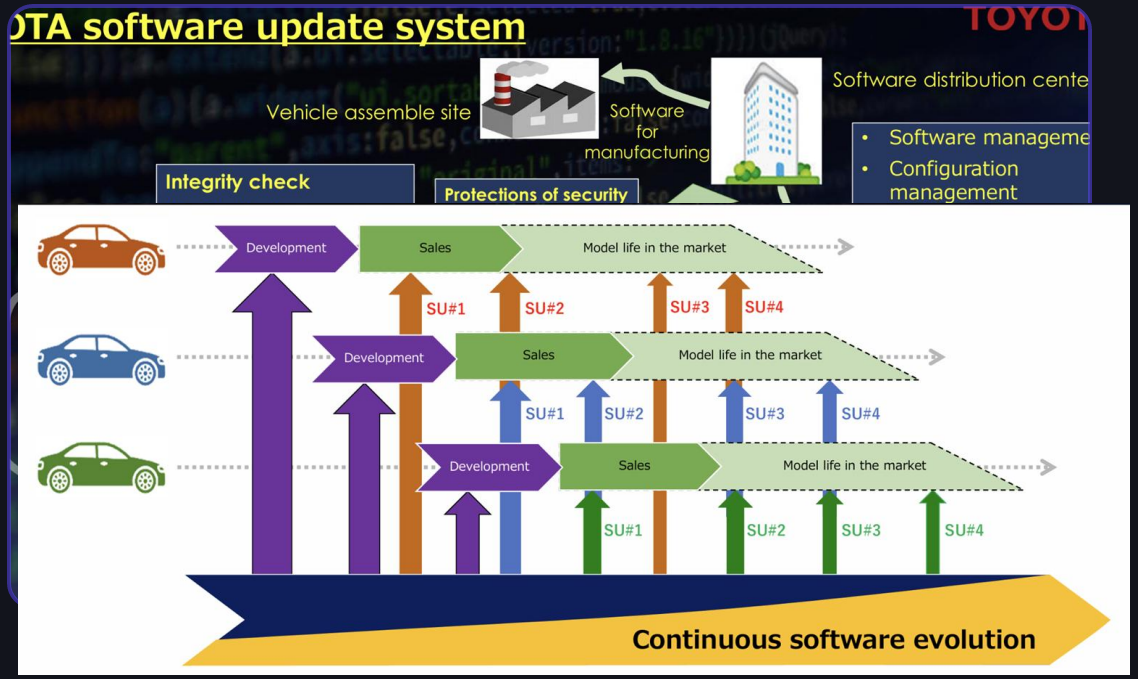
⑤ 共創

From SDV to AIDV

The intelligence loop closes, but the collaboration loop is open.

Speaker: Jaylin Yu · EMQ Technologies · SDV Security & AI Engineering

Where It All Began: Murata's Open-Source Bet



Software-First Architecture

Decouple SW from HW constraints. OTA updates replace ECU replacement cycles.

Continuous Software evolution

Understanding of Vehicle use case by collecting and analyzing vehicle data.

Iteration at Consumer Pace

Fast release cycles — impossible under legacy supplier-certified ECU models.

"Murata's insight opened the door — but Software-First requires a new answer: How do we guarantee quality while moving fast?"

THE QUESTION THAT STARTS THIS SESSION

Kenichi Murata (Toyota Fellow) Asked the Most Important SDV Question

"How do you design SDV software-first — where you continuously update and improve software after SOP — while ensuring the quality that automotive requires, and without blowing up the OEM's R&D cycle?"

— Kenichi Murata, Toyota Fellow · AGL AMM Keynote: How to Design SDV (July 2025)

THE THREE TENSIONS IN MURATA'S QUESTION:

Software-First Speed vs **OEM Certification Rigor**

SDV demands continuous over-the-air delivery. OEM quality assurance was designed for 3–5 year hardware cycles. The same engineering team must now satisfy both.

Post-SOP Evolution vs **Safety & Liability Standards**

Traditional automotive: software is frozen at SOP. SDV: software evolves after SOP — every update is a potential regression. ISO 26262, SOTIF, and UN R155 apply to every OTA.

Supplier Ecosystem vs **Vertical Software Control**

Legacy OEMs rely on 200+ software suppliers. Tesla and BYD own their full stack. Open collaboration (AGL) is the only path for traditional OEMs — but collaboration needs shared quality infrastructure.

Exact same issue we are facing now! Hundreds of Car models. Dozens of Suppliers. Millions of Code integration testing.

THE QUALITY-SPEED DILEMMA

Automotive Testing Reimagined: The Speed-Quality Crisis

72 hrs
Tesla: sense → retrain → OTA deploy cycle

90 days
Traditional OEM with OTA capability

180+ days
Dealer-dependent OEM to achieve 50% fleet fix

25×
Speed gap between Tesla and legacy OEM (NHTSA 2025)

600万+
Lines of code in modern IVI/cockpit software stack

WHERE DOES QUALITY BREAK DOWN IN THE SDV ITERATION LOOP?

01 **Software Complexity Outpaces Manual Testing**

In-vehicle stacks have 600M+ lines of code. Security testing coverage has dropped from 30% to 3-5%. Every OTA release adds new untested paths.

02 **Protocol Depth Exceeds Tooling Depth**

Vehicle protocols and applications have deep state machines. Traditional testing and FTA inject testing data and are rejected by shallow parsers — never reaching the dangerous deep states where real bugs live.

03 **UN R155 & ISO/SAE 21434 & ISO 26262 Compliance Requires Continuous Evidence**

Regulation R155 (mandatory July 2024) requires OEMs to demonstrate security testing for every OTA update under UN R156 SUMS. Manual test evidence takes 3–6 weeks per release. Continuous OTA = continuous compliance gap.

04 **Open Collaboration Shared Quality Gates**

Without automated shared testing, OEMs duplicate expensive test campaigns and miss critical cross-component software interactions. — duplicating cost and missing cross-component interactions.

SDV Protocol Bugs Live Where Traditional Fuzzers Never Go

Message Grammar

Coverage ≠ Application State

Fuzzers that break framing are rejected by the parser immediately — execution stays in the shallow layer. Easy to get 60% code coverage. Impossible to reach deep business logic.

Session State

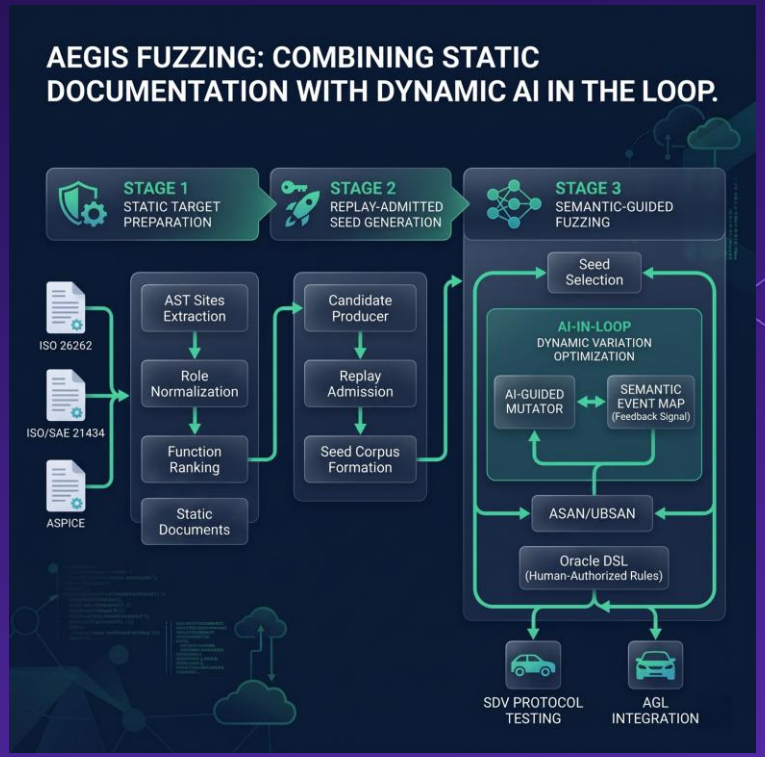
Multi-Message Session Dependencies

State-machine awareness required. A fuzzer without session context injects a SUBSCRIBE before CONNECT and wastes 80% of execution time being rejected at session-check.

Protocol Transaction

Cross-Message Transaction Semantics

UDS security access: Seed request → Key response must match within timeout window.



Aegis Fuzz'

innovation is the AI-in-the-loop framework that merges static automotive safety documentation (e.g., ISO 26262, FTA) with dynamic deep fuzzing. It uses a Semantic Oracle DSL to transform "paper compliance" into "code-level real-time compliance". This approach bypasses shallow parsers and detects critical non-crash semantic violations traditional tools miss.

Key insight: Bugs that matter — are invisible to any tool that uses response codes as state feedback. You need semantic awareness.

Safety Digital Twin: Compliance-as-Code for SDV

STAGE 1: STATIC SEMANTIC EXTRACTION WORKFLOW FUSING SAFETY ASSETS AND AI

STAGE 1: PARALLEL SITE EXTRACTION (DUAL-TRACK)

Track A (TWFuzz AI Track)

[SOURCE CODE SCAN]
Clang — Use twfuzz_clang_static_extractor.cpp for full scan.

[CLANG AST IDENTIFICATION]
Clang — Identify Dispatch, Guard, Update, Decision sites.

Purpose: Ensure no code details or auxiliary logic are missed.

Track B (Safety Asset Engineering Track)
- ISO 26262
- ASPICE
- FTA

[SAFETY DOC PARSING]
Use script to parse ISO 26262 TSC, ASPICE design docs, FTA fault trees.

[STATE MACHINE EXTRACTION]
Extract defined states and logic points.

Purpose: Extract design-intended Safety-Critical Points.

STAGE 2: SEMANTIC ALIGNMENT & MERGE

Transform "Physical Code" to "Protocol Semantics".

[AUTOMATIC TRACEABILITY & MARKING]
Match Code Sites with Safety Req IDs (Line, Yar, Logic). Mark as 'Verified-Safety-Site'.

[AI FILLING THE GAPS]
LLM role-tags code sites not mentioned in design docs. Mark as 'Exploratory-Site'.

[CONFLICT DETECTION]
Disconnect ⟷ Retry
Compare AI tagging with ISO 26262 constraints. Alert: 'Design-Implementation Mismatch'.

STAGE 3: FTA-WEIGHTED FUNCTION RANKING

Determine Testing Priority

[FUSION STRATEGY]

RANKING — 'ASIL Level Factor' + 'FTA Fault Weight' — Add ASIL factor & FTA weights to original ranking algorithm.

[FORMULA DERIVATION]

Function_Score = (AI_Semantic_Density * 0.4) + (Safety_Criticality_Weight * 0.6)

[RESULTS]
Testing resources倾斜 to 'AI complex' & 'Safety-Critical' functions.

STAGE 4: REFINED CALL-CHAIN & PATH SYNTHESIS

Provide Path Templates for Long-Prefix Seeds

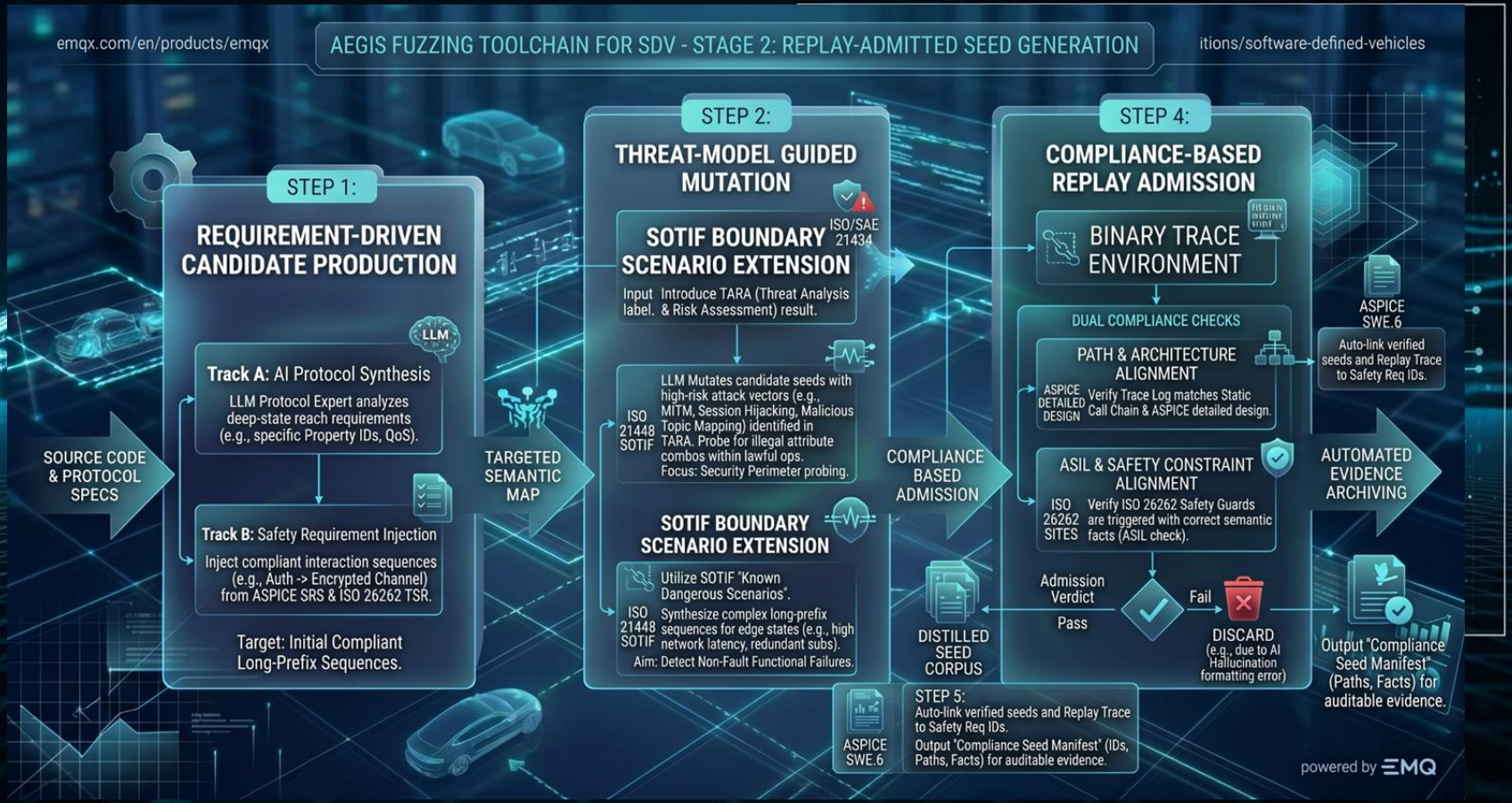
[ENGINEERING CONSTRAINT]
ASPICE
Import 'Golden Paths' (valid software call sequences) from ASPICE.

[AI ENHANCEMENT]
LLM
Inject 'unexpected but legal' perturbations into each node.
LLM

[REFINEMENT & DE-NOISING]
Extract 'Tail Signature', normalize (Bucket), merge similar paths to reduce noise.

[VALIDATION]
REPLAY → REPLAY ADMISSION
All synthesized paths must pass Replay Admission to ensure engineering logic & deep semantic reachability.

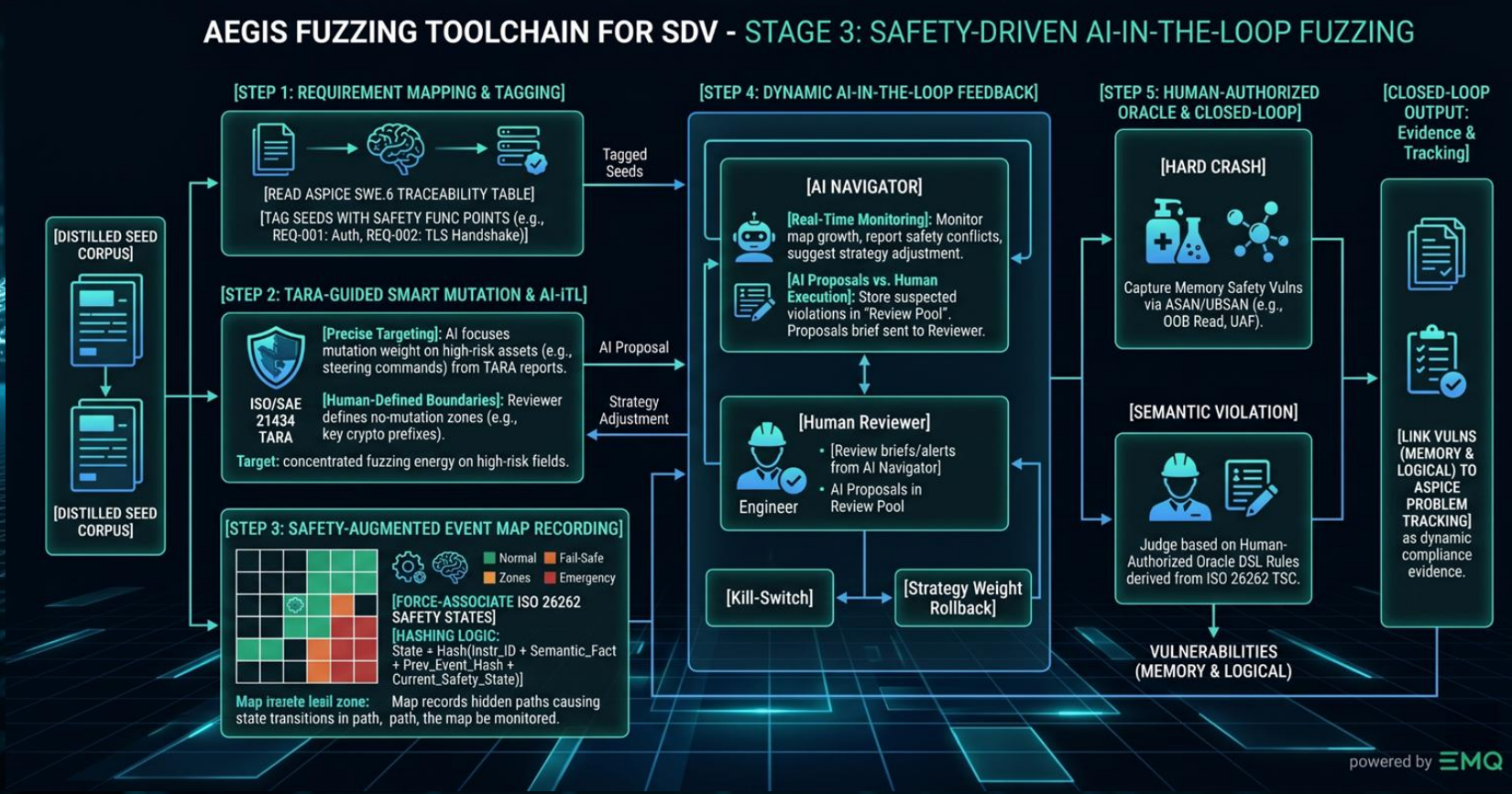
Safety Digital Twin: Compliance-as-Code for SDV



Safety Digital Twin: Compliance-as-Code for SDV

emqx.com/en/products/emqx

emqx.com/en/solutions/software-defined-vehicles



From Semantic-Aware MQTT Fuzzing to Aegis Fuzz

Stage 1
2024

Semantic-Aware MQTT Fuzzing with LLM-Generated Seeds

- Identified state folding as root cause of shallow fuzzing
- LLM generates session-aware seed sequences from call chains
- Semantic event signatures replace response-code state tracking
- AFL++ extended for stateful MQTT broker testing

Limitation: AST + LLM pipeline for seed generation only — feedback loop not closed. LLM not involved in mutation guidance during runtime.

Stage 2
2025

TWFuzz: Hybrid LLM-Guided Input Mutation & Semantic Feedback

- LLM integrated as independent mutation service alongside AFL++
- Master Fuzzer (AFL++) + Helper (LLM) asynchronous co-evolution
- Few-shot prompting vs zero-shot evaluated across 4 open-source LLMs
- DeepSeek-R1 identified as most effective open-source reasoning model

Limitation: Multi-party coordination, but feedback signal still coarse — semantic event map not fully formalized as runtime instrumentation.

Stage 3
2026

Aegis Fuzz: Reaching Deep States for general SDV — Documentation-Guided Fuzzing

- AST-based semantic site extraction → formal semantic fact definitions via LLM
- LLM generates executable long-prefix seeds + dynamic replay admission verification
- Runtime semantic event map: Combining with static documentation engineering. (ISO 26262/ASPICE)
- ASAN/UBSAN oracle + semantic oracle: two-dimensional bug detection
- Discovered state-folding bugs under complex module messaging topology.

Result: First end-to-end framework connecting static program analysis, LLM seed generation, and semantic runtime feedback.

Adapting Aegis Fuzz to Vehicle-Grade Protocols: SOME-IP, UDS, CAN DBC

Why TWFuzz Generalizes to Automotive

All automotive protocols share the same three-layer structure that makes MQTT hard to fuzz:

- **Message Grammar:**

  SOME-IP: framing + service interface header. UDS: service ID + subfunction. CAN DBC: signal encoding.

- **Session State:**

  UDS: security access requires seed-key handshake. SOME-IP: service discovery before method calls.

- **Transaction Semantics:**

  UDS: diagnostic session must match security level. SOME-IP: event subscription before notification.

AST-based semantic site extraction is protocol-agnostic. Only the semantic fact definitions (generated by LLM from spec documents) need to change between protocols.

PROTOCOL ADAPTATION MAP

Protocol	Session State Facts	Transaction Facts
MQTT v5 (VeloRT/SDV-Flow)	CONNECT session + auth	QoS1 PUBLISH/PUBACK packet-ID tracking
SOME-IP (AUTOSAR/ARA)	Service discovery → method call sequence	Event subscription lifecycle, reliable delivery
UDS (ISO 14229) Diagnostic	Session type (0x10) + security access (0x27)	Seed-key pairs, timeout windows, NRC codes
CAN DBC Signal Layer	CAN bus arbitration state, gateway filters	J1939 PGN sequences, diagnostic frames
DoIP (ISO 13400) Diagnostics	TCP activation handshake before UDS routing	Vehicle ID announcement → entity status lifecycle

TWFuzz semantic fact extraction proven on MQTT/NanoMQ. SOME-IP/UDS adaptation in active development, targeting AGL UCB VeloRT communication stack. CAN/DoIP planned for Phase 2.

Aegis Fuzz SDV: AI Fuzzing Meets the Vehicle Ontology

Automotive-Specific Extensions

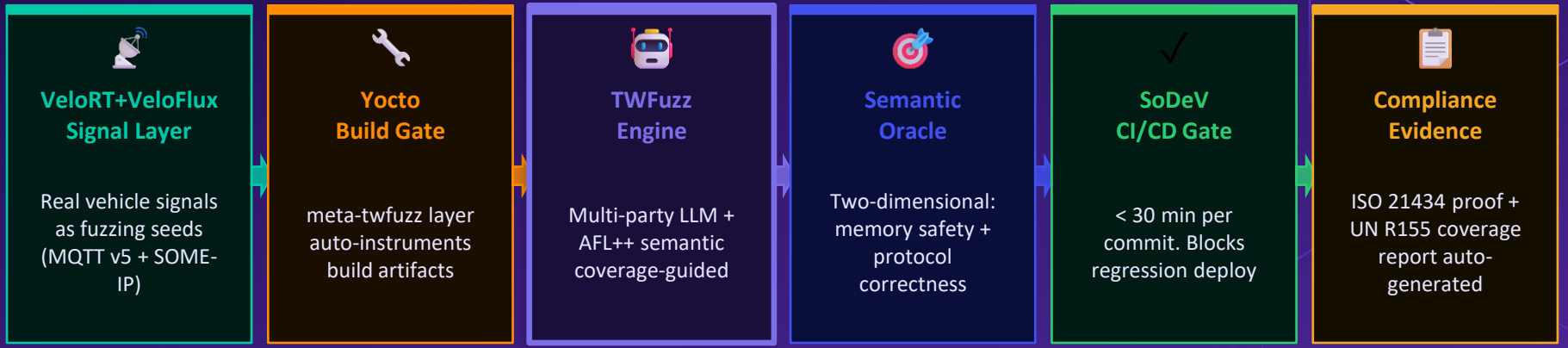
Cross-Domain Oracle
Semantic rules span SOME/IP ↔ CAN ↔ ADAS domain. FTA branch coverage as fuzzing target.

UML/SysML Input
Ingest AUTOSAR component models to map system boundaries.

Real Fleet Seeds
SDV-Flow packet captures → replay-admitted seeds. 12–30× higher fault discovery.

ISO 26262 Oracle DSL
Safety violations expressed as executable oracle rules. Integration with Policy Guard.

Closing Murata's Quality Loop: TWFuzz as the Security CI Gate in AGL SoDeV



Answers Murata's Question

Continuous automated security testing (< 30 min/commit) ensures every SoDeV OTA preserves quality. Engineers focus on new features — not regression hunting. This IS the quality-speed bridge.

AGL Community Benefit

TWFuzz as an AGL community project creates shared security CI infrastructure. Honda, Toyota, and Panasonic all benefit from the same protocol fuzz harness for SOME-IP/UDS — reducing cost 3–5× vs bespoke OEM campaigns.

UN R155 Compliance Automation

Each CI run generates an audit-ready evidence package: coverage report, vulnerability list, protocol state coverage, CVSS scores. UN R156 SUMS OTA compliance becomes a by-product of the development loop.

VeloRT+VeloFlux: Real Fleet Data as Fuzzing Seeds

The vehicle's own telemetry becomes the most powerful fuzzing corpus imaginable

1. Real Data Seed

- VeloRT captures Signals & events
  (sensor, timing, state)
- VeloFlux pre-labels
  event class + anomaly
- VeloRT transmits QoS
  arrives annotated at cloud
- Simulation Seed
  fingerprint → parameters

2. Cloud SIM Expansion

- AI agent injection
  into virtual ECU
- Boundary variation
  ±20°C, ±30% load, jitter
- Scenario Generation
  500–5,000 per field event
- Cost Optimization
  ~¥0.15 / simulation run

3. Aegis Fuzz Synthesis

- Failure Boundaries
  Identify reproducibility conditions
- Evidence Package
  Auto-generates FTA-ready data
- OTA Triggers
  Confidence-based queuing
- Decision Review
  Engineers review decisions

Siemens PAVE360 & Ansys TwinAI benchmark: real-field-seeded simulation achieves **12–30× higher** fault discovery vs. synthetic-only generation

DATA TRANSPARENCY

The Non-Negotiable Foundation of SDV Safety

Why Silos Are Killing Automotive Safety

Fragmented defect knowledge

Each OEM rediscovers the same protocol-level bugs independently. NanoMQ deadlock-class issues likely exist in multiple production vehicles today — unreported.

Safety evidence opacity

When auditors ask for proof that MQTT reconnect storms were tested, the honest answer is: we have no systematic evidence. ASPICE paperwork ≠ empirical safety proof.

AI training data desert

LLMs and fuzz engines improve with data. Hoarding defect datasets means the community's AI tools are trained on toy examples, not production failure modes.

Competitive advantage is misallocated

Safety is not a competitive differentiator — it is a social contract. Racing to build redundant private testing infrastructure is waste. Sharing the foundation to compete on features is rational.

What AGL Members Could Share

- → Anonymized defect databases
- → Protocol spec ABNF grammars
- → FTA templates per subsystem
- → Fuzzing coverage metrics
- → Regression test corpora

The Flywheel Effect

- ↻ More shared data → better LLM seeds
- ↻ Better seeds → more bugs found across members
- ↻ More bugs found → better safety evidence
- ↻ Better evidence → faster ASPICE certification
- ↻ Faster certification → faster SDV iteration

We Are Calling on Every AGL Member in This Room

The vehicle that runs the AGL SoDeV stack today will make safety-critical decisions autonomously within 24 months. The semantic bugs that will cause the first incident are already in the codebase. They are just waiting for the right input sequence.

① Join Our Wrk

Propose a new AGL Expert Group to work with ELISA for AI-driven safety verification. Toyota, Panasonic, Honda, DENSO already have the production data and failure-mode knowledge we need.

② Contribute Oracle Rules

Your safety engineers know your failure modes. Encode one FTA branch as an Aegis Fuzz oracle rule — contribute it to the shared library. One rule protects every AGL member.

③ Share Anonymised Field Seeds

Structured eKuiper event traces from your production fleet — anonymised, consent-compliant — fed to Aegis Fuzz dramatically improve fault discovery for everyone.

④ Require It of Your Suppliers

Make Aegis Fuzz oracle compliance a procurement requirement for Tier-1 components integrated into AGL SoDeV. One certification standard for all OEMs.

More community eyes mean safer roads. Let's build this together — starting today.