Roadmap Towards the First Practical VML Implementation Discussion Paper for the Technical Working Group for Formal Automotive Systems

Working Draft

July 27, 2026

Abstract

The VML whitepaper establishes a compelling vision: natural language is an engineering-time artifact, while formal machine-interpretable specifications govern runtime vehicle behavior.

However, a significant gap remains between the conceptual language definition and a practical interoperable implementation.

This document identifies the most important unresolved questions, ranks them by implementation relevance, and proposes a roadmap for future standardization work.

The central question is not whether VML is useful, but what additional specifications are required so that two independent organizations can implement VML and obtain compatible runtime behavior.

Contents

1 Current State of the Initiative

The current VML draft successfully defines:

- motivation and architectural role;
- layered language model;
- basic grammar;
- monitor semantics;
- conformance concepts;
- examples of behavioral specifications.

The current documents answer:

> "What should VML represent?"

but do not yet fully answer:

> "What artifacts must exist, what data flows through them, and how is a VML program connected to a real vehicle stack?"

The following sections identify the highest-priority gaps.

2 Priority Ranking of Open Topics

Priority	Topic	Reason
P0	Canonical State Model	No implementation can execute VML without a standardized runtime state representation.
P0	Artifact Model	It is currently unclear what constitutes a VML package or deployment artifact.
P0	Signal Binding Specification	No standard mechanism exists for connecting perception, planning, localization and map data to VML symbols.
P1	Evaluation Semantics	Layer ordering and runtime execution order are not clearly defined.
P1	Planner Integration Contract	Unclear how VML interacts with planners and trajectory generation.
P1	Predicate Classification	No distinction exists between observable facts, derived facts and planner actions.
P2	Legal Rule Libraries	No concept for jurisdiction-specific legal rule sets exists.
P2	Ontology Packaging	Reuse and exchange of ontology definitions remain unspecified.

P3	Advanced Probability Model	Useful but not required for an initial implementation.
P3	Certification Profiles	Important long-term objective but not a blocker for a first prototype.

3 Highest Priority: Canonical State Model

3.1 Problem

The specification states that implementations must map perception, localization, map and planning data into a canonical state representation.

However, the structure of that state remains undefined.

Without a common state model there is no interoperability.

Two implementations may use the same VML file while interpreting symbols differently.

3.2 Questions to Resolve

1. Which primitive types exist?
2. Are coordinates standardized?
3. How are dynamic objects represented?
4. How are lanes represented?
5. How are predicted trajectories represented?
6. How is uncertainty attached to observations?
7. How are timestamps handled?
8. How are multi-rate signals handled?

3.3 Recommended Deliverable

VML Runtime State Model v0.1

The state model should become a normative document.

4 Highest Priority: Artifact Model

4.1 Problem

The specification discusses programs, parsers, monitors, traces, ontologies and verification artifacts.

However, it is unclear what a complete deployable VML package looks like.

4.2 Open Questions

Should a VML deployment contain:

- one file?
- multiple files?
- imports?

- metadata?
- rule libraries?
- ontologies?
- test traces?
- expected verdict streams?

4.3 Proposed Structure

```
vehicle_mission/
    manifest.yml
    mission.vml
    legal_profile/
    ontology/
    bindings/
    traces/
    verdicts/
```

A package model should be standardized early.

5 Signal Binding and Data Mapping

5.1 Problem

The language refers to symbols such as

```
Ego.Velocity
DistanceToPedestrian
TrafficLight.State
```

but their data source is undefined.

5.2 Questions

1. Does VML define signal names?
2. Who owns signal definitions?
3. Are naming conventions mandatory?
4. Can multiple sensor sources provide the same symbol?
5. How is uncertainty attached?
6. How is signal validity represented?

5.3 Deliverable

State Binding Specification.

6 Layer Ordering and Evaluation Semantics

6.1 Problem

The language defines several layers:

- Logic
- Temporal
- Metric
- Probability
- Goal
- Deontic
- Ontology
- Utility

The specification does not explicitly define whether the ordering is:

- pedagogical,
- conceptual,
- dependency-based,
- implementation-based,
- execution-based.

6.2 Question

Can two implementations evaluate the same VML program and guarantee identical results?

6.3 Recommendation

Define a normative evaluation pipeline.
Example:

1. ontology expansion
2. symbol resolution
3. metric evaluation
4. probabilistic evaluation
5. logical evaluation
6. temporal progression
7. deontic activation
8. goal updates
9. utility ranking

7 Planner Integration Contract

7.1 Problem

The architecture references:

VML Evaluator → Constraint Filter → Trajectory Optimizer

but the interface remains unspecified.

7.2 Questions

1. Does VML generate planner constraints?
2. Does VML reject trajectories?
3. Does VML assign costs?
4. Does VML activate fallback maneuvers?
5. Can VML request replanning?
6. Can VML force emergency actions?

7.3 Recommendation

Define a planner-facing API.

8 Predicate Classification

8.1 Problem

VML currently treats all predicates similarly.
In practice there are different categories.

8.2 Proposed Classification

Observed Predicate Directly measurable.

Derived Predicate Computed from state.

Probabilistic Predicate Associated with confidence.

Planner Intent Predicate Represents planner decisions.

Action Predicate Represents requested behavior.

Legal Predicate Represents regulatory abstractions.

8.3 Example

`TrafficLight.State = RED`

Observed Predicate.

`IntersectionClear`

Derived Predicate.

`ExecuteMinimalRiskManeuver`

Action Predicate.

9 Jurisdiction and Legal Profiles

9.1 Motivation

A vehicle driving in Germany, Japan, California and Saudi Arabia may follow different rules.
VML therefore requires a mechanism for legal portability.

9.2 Proposal

Introduce legal profiles.

```
IMPORT LEGAL_PROFILE DE_STVO_2026;
```

or

```
legal_profile:
    jurisdiction: DE
    regulation: STVO
    version: 2026
```

9.3 Open Questions

- How are legal updates versioned?
- How is provenance maintained?
- How are conflicts resolved?

10 End-to-End Example Requirement

One of the most important missing artifacts is a complete runnable example.

10.1 Suggested Reference Scenario

Driving from Ludwigsburg to Stuttgart.
Required artifacts:

1. Mission specification
2. Route definition
3. State schema
4. Legal rule library
5. Ontology package
6. Example traffic traces
7. Expected monitor outputs
8. Reference implementation

10.2 Success Criterion

Two independent implementations must generate identical verdict streams.

11 Reference Implementation Roadmap

Phase 1: Minimal Executable Core

- parser
- AST
- state schema
- monitor
- trace execution
- verdict generation

Phase 2: Automotive Pilot

- lane keeping
- speed limits
- traffic lights
- pedestrian protection
- route missions

Phase 3: Ontology and Reuse

- inheritance
- rule libraries
- legal profiles
- package management

Phase 4: Advanced AI Integration

- semantic compilers
- probabilistic entities
- risk models
- planner interaction

12 Recommended Working Group Priorities

The working group should focus on the following order:

1. Canonical Runtime State Model
2. Artifact and Package Model
3. Signal Binding Specification
4. Evaluation Semantics

5. Planner Integration Contract
6. Predicate Classification
7. End-to-End Reference Scenario
8. Legal Profiles
9. Ontology Exchange
10. Advanced Probability Extensions
11. Certification Profiles

The first seven topics are likely prerequisites for meaningful interoperability between independent VML implementations.

13 Conclusion

The current VML initiative has successfully established a strong conceptual foundation.

The next phase must shift focus from language concepts toward implementation contracts.

The highest value will likely come from defining:

1. the canonical runtime state,
2. deployment artifacts,
3. data binding semantics,
4. executable evaluation semantics,
5. planner interfaces.

Once these foundations exist, VML can evolve from a formal language proposal into a practical interoperable ecosystem. The primary objective should therefore be not additional language features but a precise definition of how a VML specification interacts with real automotive software systems.