Measurement and success criteria

March 25, 2020
Justyna Zander, *PhD*

ASAM
Association for Standardization of
Automation and Measuring Systems

Measurement and success criteria

Observers shall be the means to set, evaluate and grade the success while designing and executing OpenSCENARIO 2.0 artifacts.

Success shall be measured based on compliance of the scenarios to the test plan, to test objectives, and hence to the expected behavior documented in the test specification.

In addition, **validity** of the scenario shall be evaluated.

Observers

Observers can be seen as part of the scenario. However, it is recommended to abstract them away from the scenario, since they provide additional insights about the design and the execution of a scenario.

Hence, they may provide different results for various test regimes (e.g. re-simulation (replay), simulation, proving ground, on road testing). Often, they do not require a specific scenario to be associated with them.

Observers are divided into the following categories:

- Generic safety checkers
- Context-specific evaluators (e.g. scenario-specific evaluators, driving-mission-specific evaluators, scenario-quality evaluators)
- Coverage measurement models
- Context-specific Key Performance Index (KPI) collectors.

Observer examples
Pseudocode

Example 1:
English specification: *Set an upper bound of 10 seconds for the car to stop.*
Pseudo-code specification: **ensure** car.is_stopped() **within** 10.0 **seconds**

Example 2:
English specification: *The car must achieve all other goals without ever exceeding 60 mph*.
Pseudo-code specification: **ensure always** (**abs**(car.longitudinal_speed()) < 60.0 **mph**)

Example 3:
English specification: *The car must stop in ego lane only after the ego is moving fast enough*.
Pseudo-code specification: **ensure abs**(ego.longitudinal_speed()) > 50.0 **mph sometime before** (car.is_stopped() **and** car.in_ego_lane())

1. Focus on **what** to test
2. Abstract away **how** to do the search (outsource this job to the engine)

Coverage and KPIs

Coverage-driven verification promises to minimize redundant effort by using "coverage" as a guide for directing verification toward untested areas of functionality. Coverage is defined as the percentage of verification objectives that have been met and is used to gauge progress of a verification project toward "*verification closure*".

KPIs serve to record non-coverage metrics and help to assess the **success** of the observers while executing a scenario. KPIs measure this success based on compliance of the scenario execution to the test plan, to test objectives, and hence to the expected behavior documented in the test specification.

KPI examples
Pseudocode

Example 1:

English specification:
Collect time-to-collision KPI for a select scenario (at the end of lane change).

Pseudo-code specification:
collect KPI (time-to-collision) at action / phase

Example 2:

English specification:
Collect KPI for distance between the ego car and the car ahead (throughout duration of the scenario) as a function of time.

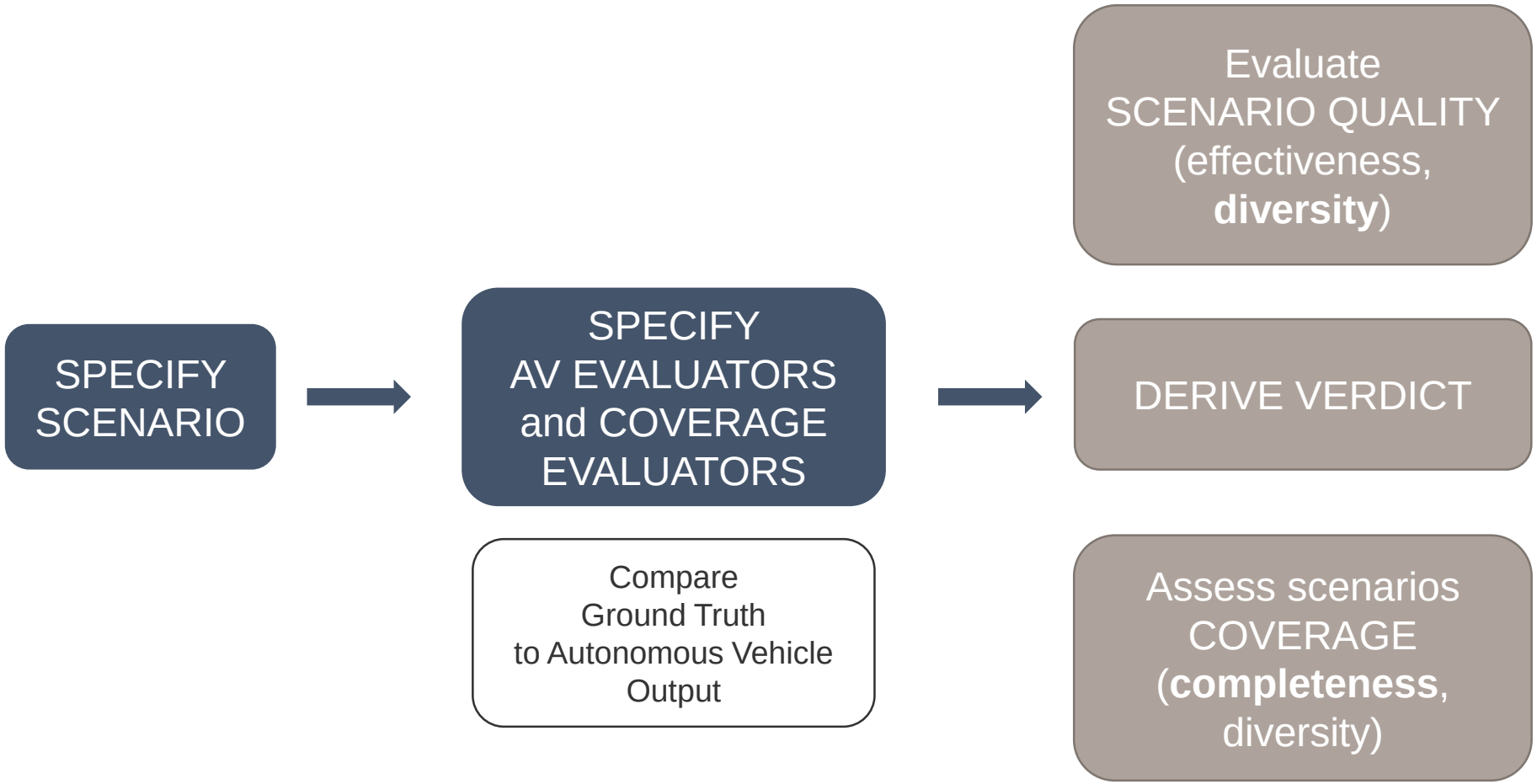
Pseudo-code specification:
collect perception_KPI (distance) at driving

Checking for errors

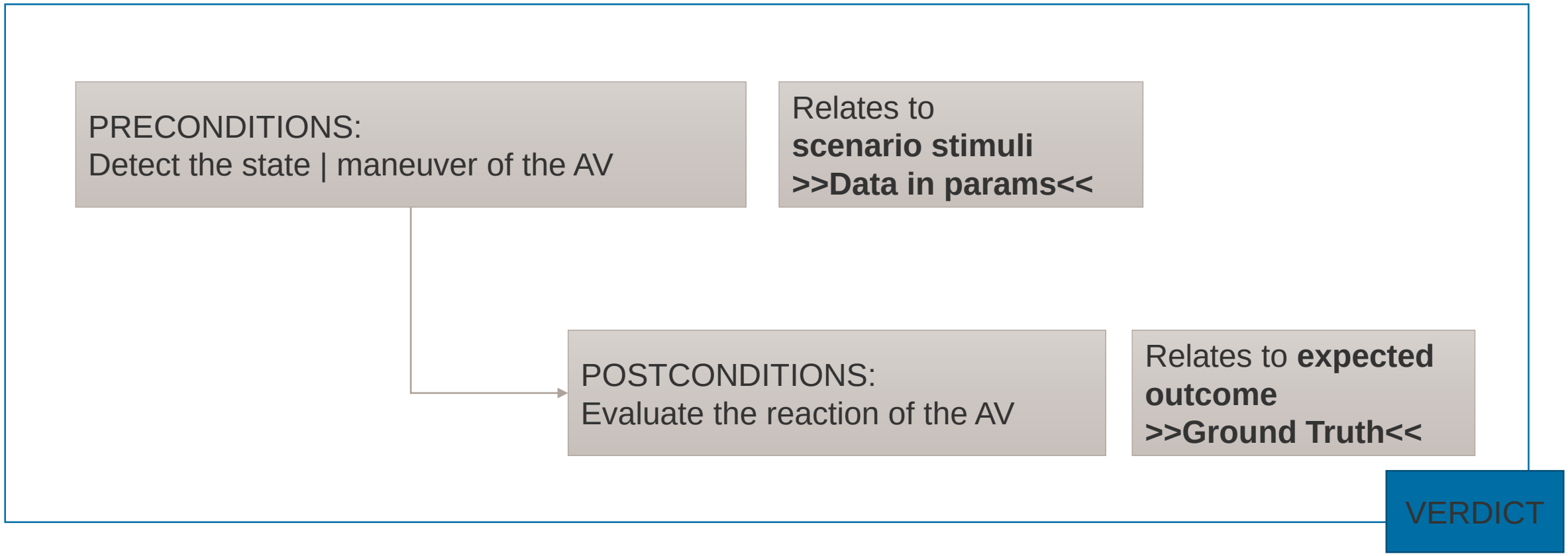
- A **DUT error** means the DUT (ego car) did something wrong.
- The language should provide a way to indicate this, invoking a zero-time scenario, e.g., command in an exemplary pseudo-code such as: dut.error().

- A **scenario failure** means that the scenario did not happen according to its definition.
- This should be indicated by calling the zero-time scenario automatically by the execution engine.

Methodology FLOWCHART



Evaluator


PRECONDITIONS:
Detect the state | maneuver of the AV
Relates to
scenario stimuli
>>Data in params<<
POSTCONDITIONS:
Evaluate the reaction of the AV
Relates to **expected outcome**
>>Ground Truth<<
VERDICT