Chasing critical situations in large parameter spaces

Mugur Tatar, QTronic GmbH

Acknowledgements

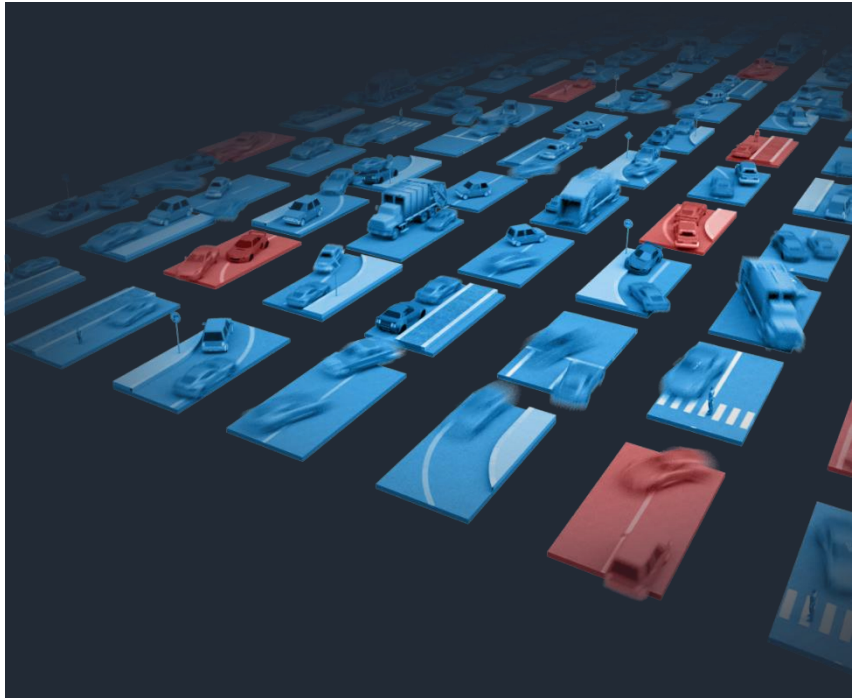
Part of the research on the parameter exploration strategies is founded by the German Federal Ministry for Economic Affairs and Energy (BMWi) as part of the

PEGASUS research project
www.pegasusprojekt.de

We warmly thank all project partners for the many interesting discussions and helpful suggestions.

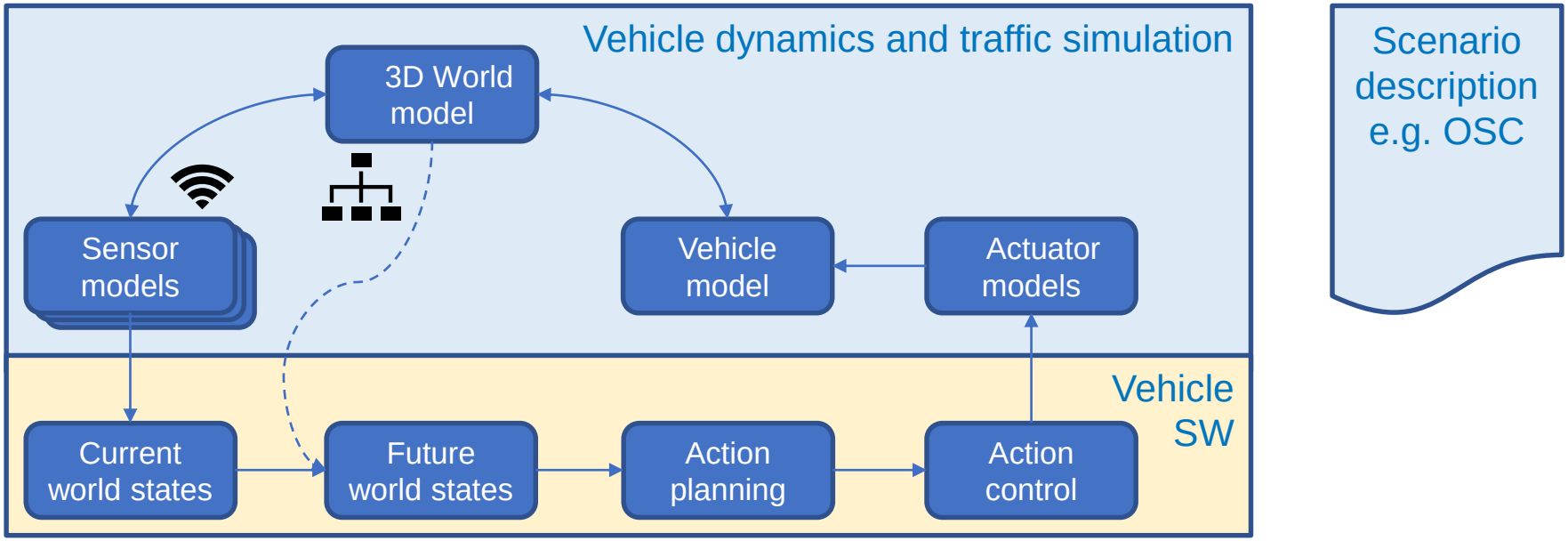
The responsibility for the contents of this presentation is assumed entirely by the author.

Agenda - Touched Topics

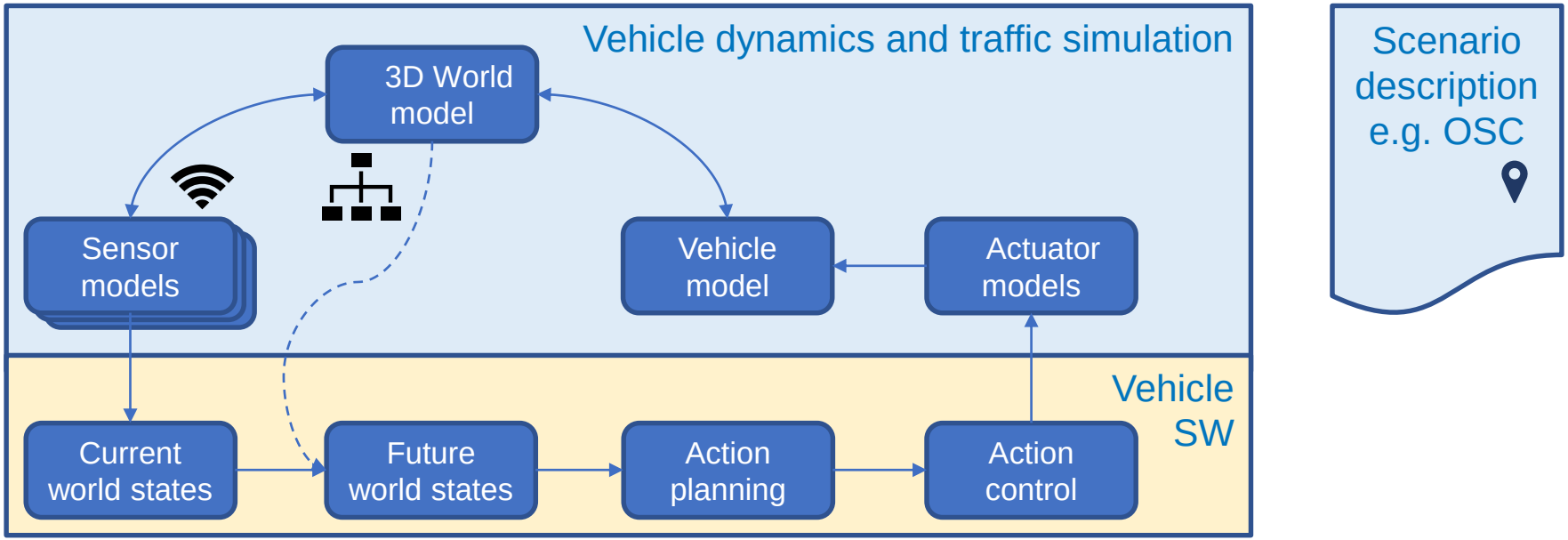


- Context:
  Simulation-based Test of ADAS / AV
- Search Space:
  Static Parameters vs. Dynamic Events
- Test Assessments for
  Correctness & Safety
- Test Objectives
  Worst Cases & System Characterization
- Exploration Strategies: Combinatorial, Stochastic Optimization, Coverage-Driven
- Concluding Remarks &
  Discussion

Simulation-based Testing of ADAS / AV



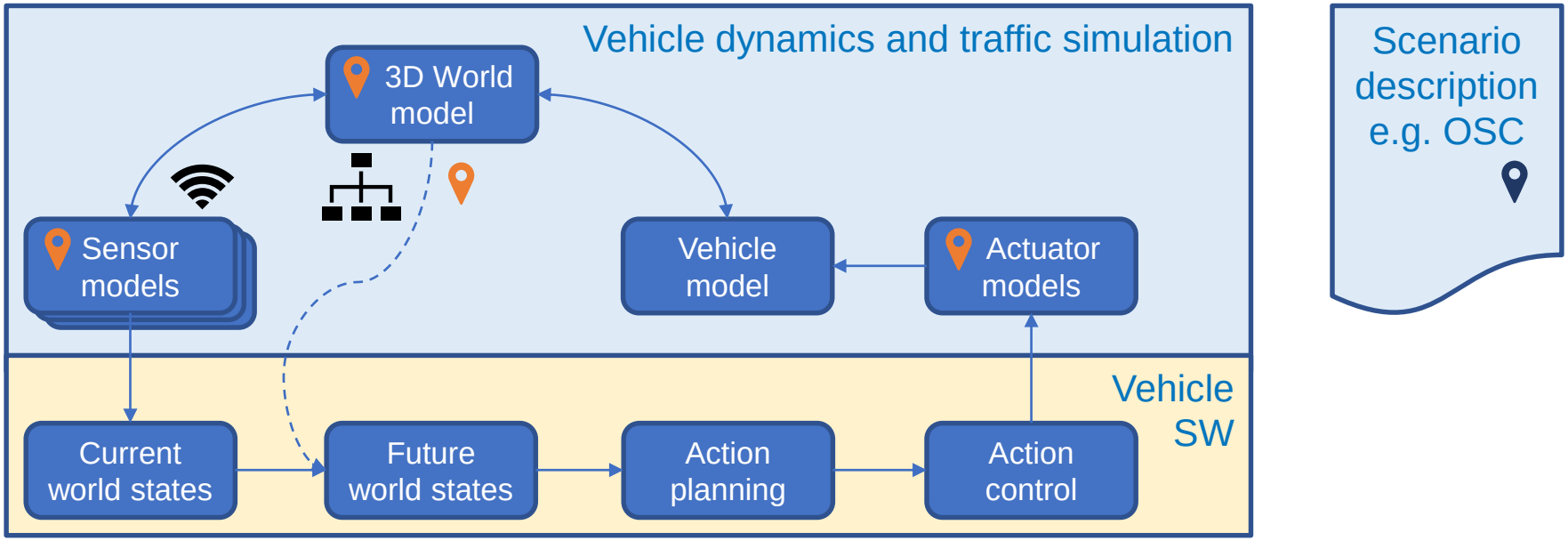
Simulation-based Testing of ADAS / AV



Possible changes by test instrumentation:

- static changes in scenario description: road, traffic objects, behaviors, starting states

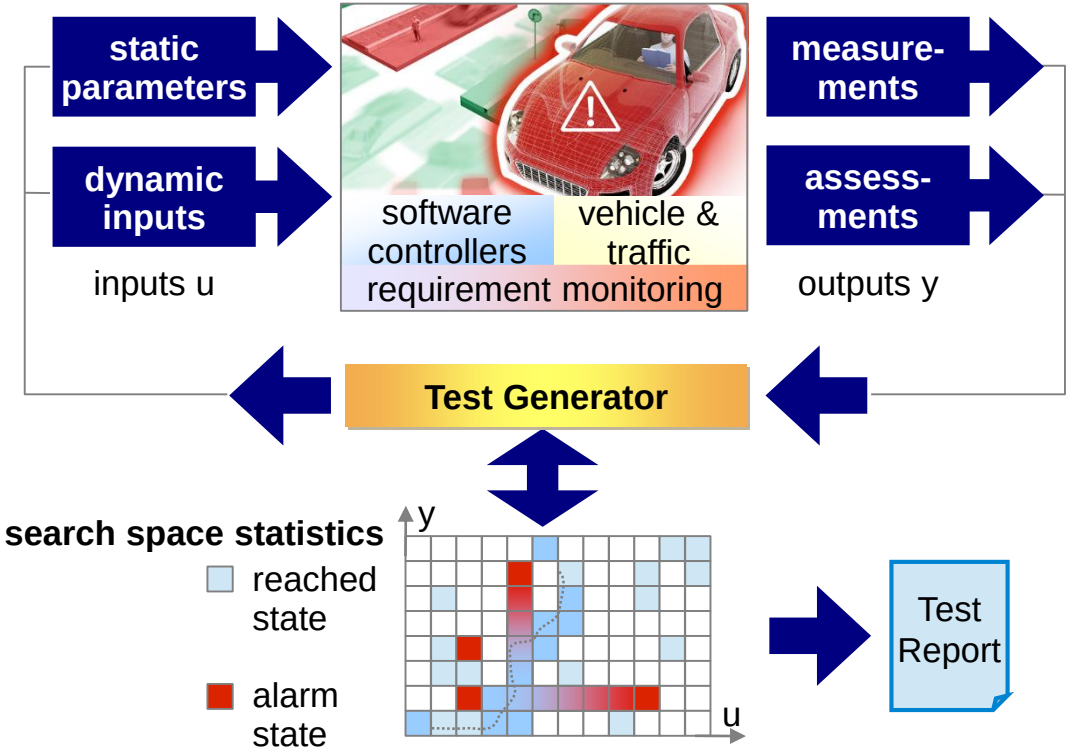
Simulation-based Testing of ADAS / AV



Possible changes by test instrumentation:

- static changes in scenario description: road, traffic objects, behaviors, starting states
- dynamic changes: random disturbances, object insertion, etc.

Simulation-based Explorative Testing - Architecture



- Maintained search space statistics can include
  - distance to violation of requirement
  - coverage information
- Generation strategies
  - search for requirement violations, border cases, worst cases
  - maximization of coverage

Search Space - Static Parameters & Dynamic Events

Commonalities

- Mix of continuous and discrete controllable variables
- Additional logical and algebraic constraints on allowed values, e.g.

```
vStart_ego > vStart_A
```

Static search space

- Predefined parametric scenarios for qualitatively differing classes, such as
  - “lane merge”
  - “obstacle ahead”
- See for instance Pegasus, www.pegasusprojekt.de

Dynamic search space

- In addition to the static case, constrained random changes are allowed, for instance:
  - other traffic objects can perform random manoeuvres, accelerate / break / change position in/across lanes, fault injections
- Main difference to the static search space:
  - number of interventions / scenario not limited

Test Assessments

Functional Requirements

- Requirement-bases test part of ISO26262
- Functional requirements monitored at all times in all scenarios with *requirement watchers*

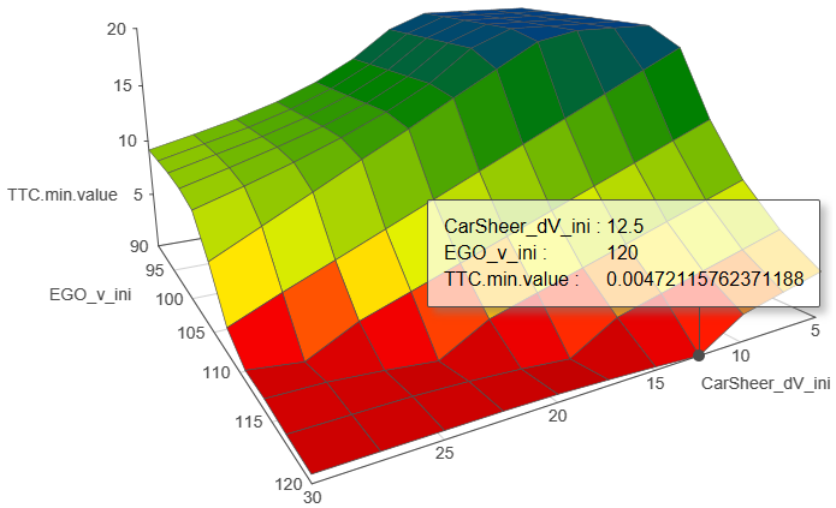
whenever "vehicle ahead signals lane change"
expect "ego-car stops accelerating"
within 2s

- Encoded with, for instance, the TestWeaver Requirement Modelling Language
- Hundreds to thousands of requirements continuously monitored!

Safety Metrics

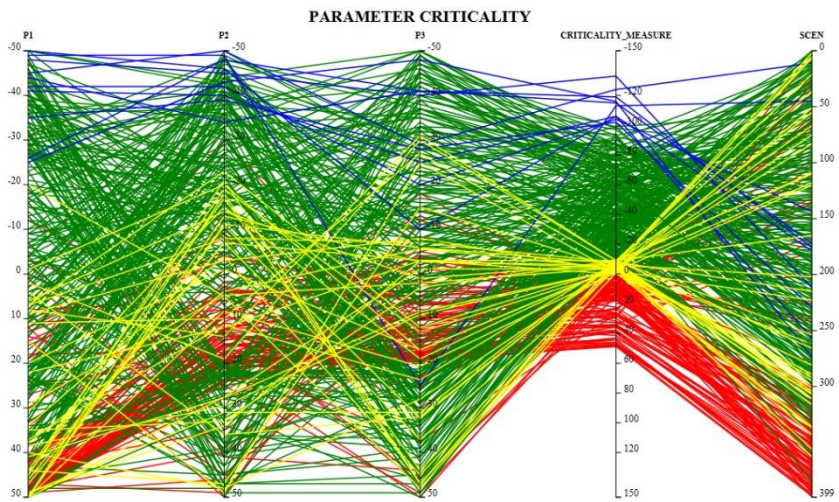
- Deliver a distance to a safety-critical situation, respectively measure the severity
- For instance:
  - Time-To-Collision
  - Collision-Severity
    relative speed at collision time
  - Combinations TTC-CS
- Certain collisions must be accepted, i.e. not caused by the AV and unavoidable
  - **A manual assessment of the identified safety-critical situations might still be required**

Test Objectives



- <u>Search</u> for safety violations / worst case(s)
- <u>Characterize</u> the regions with safety violations, e.g. find their borders
- Deliver coverage reports for one or for a suite of experiments
- Report all functional requirement violations (e.g. breaking driving rules) and safety violations found until a coverage objective is met

Test Coverage - what is / should be “coverage”?



The “coverage” definition should be customizable depending on the test objectives

Possible coverage measures

- Functional requirement coverage
- SW source code coverage
- Operational state coverage for a given scenario or across scenarios
  - Individual indicators vs. cross products
- Input coverage for a given scenario
  - Often based on equivalence classes
  - Individual variables vs. cross products
- Coverage across scenarios / situations
  - Highway / Tunnel / Parking house / etc.

Exploration Strategies - Full Cross Product

- N parameters with S sampling points (equivalence classes) require S^N scenarios
- For S=5 sampling points and assuming 10s / scenario:

S=5	5 parameters	10 parameters	15 parameters
Scenarios	3.125	9.765.625	30.517.578.125
Simulation time	8,7 Hours	3 Years	9677 Years

- Results are easy to comprehend
- “Complete, given the resolution S”
- Only usable for small N, even with very moderate S

Partial Cross Products - K-Combinations

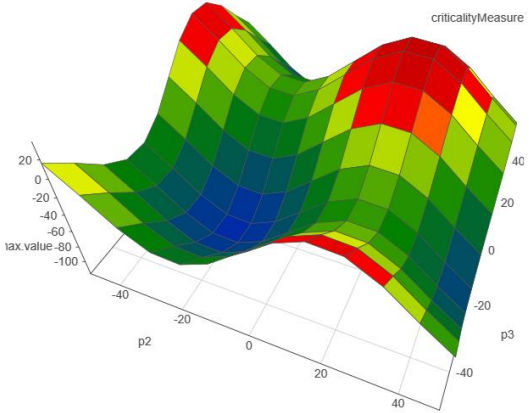
- N parameters with S sampling points (equivalence classes)
- For some/all <u>combinations of K parameters</u> C(N,K), do the <u>K-cross products</u>: C(N,K) x S^K

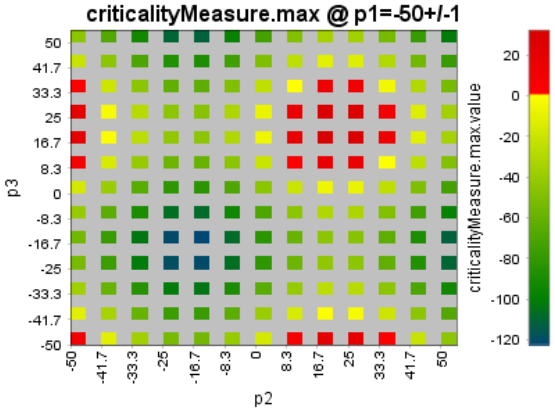
K=2, S=10	5 parameters	10 parameters	15 parameters	20 parameters
Scenarios	1000	4500	10.500	19.000
Simulation time	2,7 hours	12,5 hours	29 hours	2,2 days
K=3, S=5	**5 parameters**	**10 parameters**	**15 parameters**	**20 parameters**
Scenarios	1250	15000	56.875	142.500
Simulation time	3,5 hours	41,6 hours	6,6 days	16,5 days

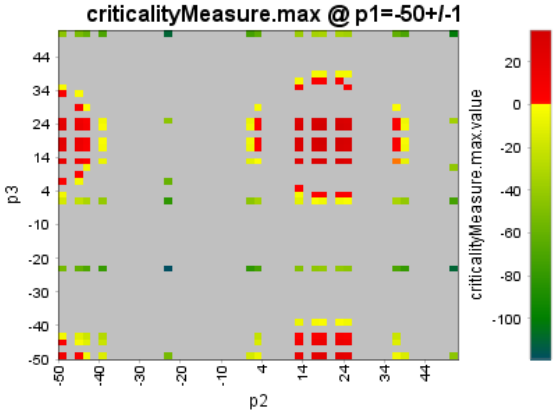
- For small K one can increase N. But when is it “safe” to reduce K?
- See also “Functional Decomposition: an approach to reduce the approval effort for highly automated driving”, Amersbach & Winner, *8. Tagung Fahrerassistenz*, 2017.

K-Combinations Adaptive Sampling

- Use KC with a smaller S and <u>increase the sampling rate around interesting spots</u>
- For instance near (a) safety violation border crosses (b) worst cases



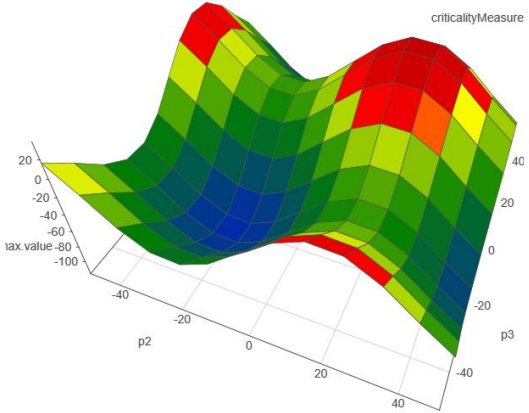


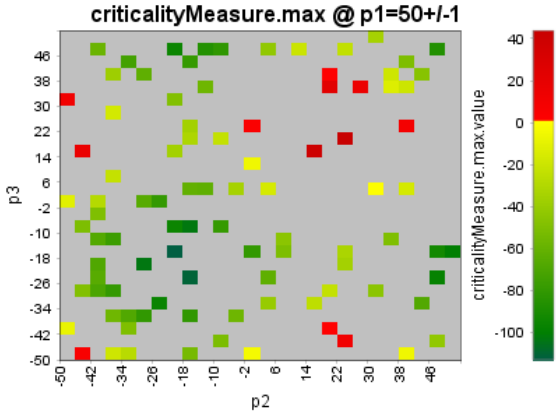


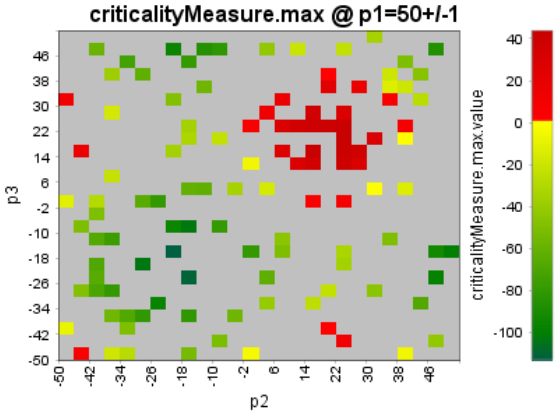
- Can be useful for characterization of areas with requirement violations
- Only applicable for small K
- What values take the unchanged N-K parameters? Better start with a good seed.

Random Generation and Local Optimization

- Figures below illustrate RND and RND+LO
- Local optimization algorithms: hill-climbing (gradient based), simulated annealing



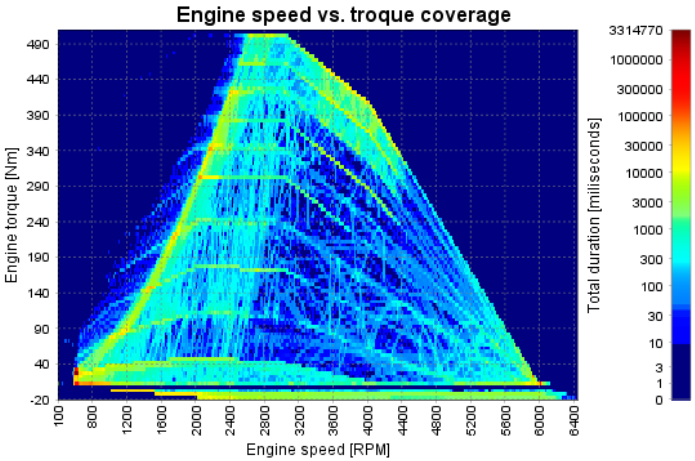


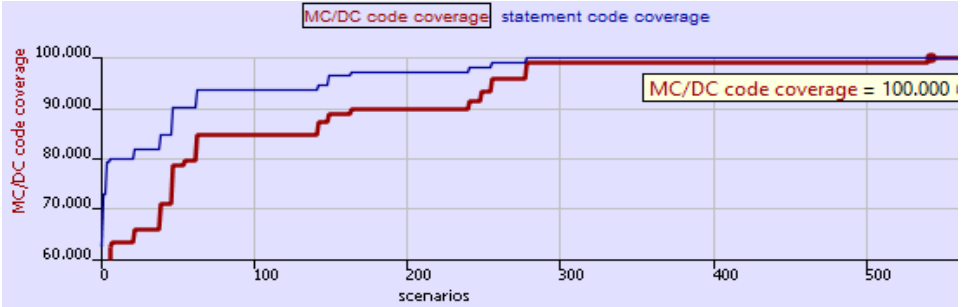


- Good to find hot-spots, but not so good for system characterization
- Restarts with differing seeds may find spots from differing regions with safety violations

Coverage Driven Generation

- Useful for search spaces with dynamic input changes and alternative coverage measures
  - operational state coverage, requirement coverage, code coverage

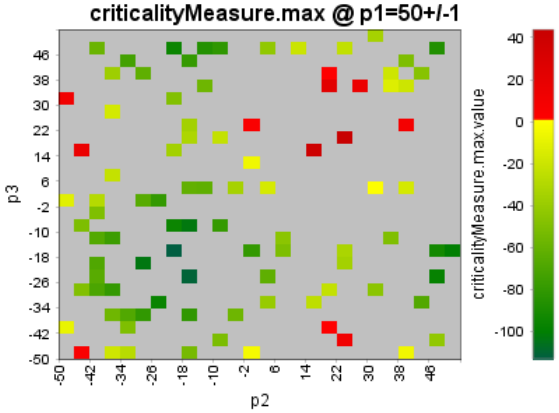


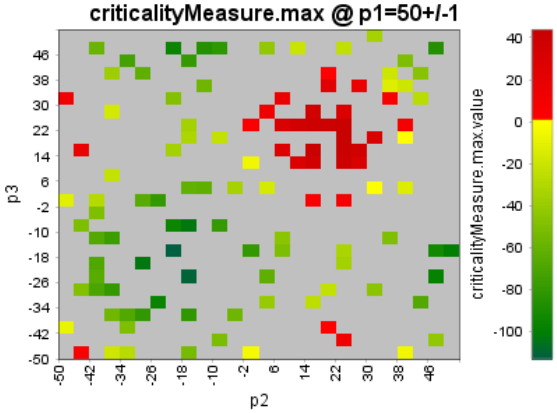


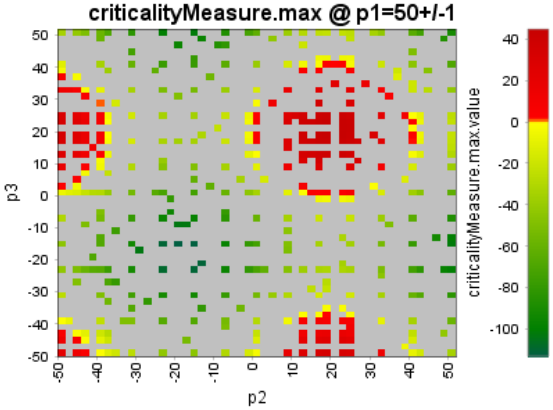
- Applications so far: powertrain and vehicle stability (ESP, ABC)
- Typical problem size, naïve translation to a static parameter space: 10^{300}
  https://youtu.be/EawgRnTAQxE

Multi-Stage Experiments / Strategy Mix

- No best algorithm for all test objectives and larger N
- Iterative approach: **Decompose → Experiment → Learn → Repeat**. For instance:
  - Use RND + LS to find a hot spot
  - Use KC to sample on 2D / 3D projections around the hot spot
  - Repeat or apply KC-AS for better characterization of worst case / safety borders

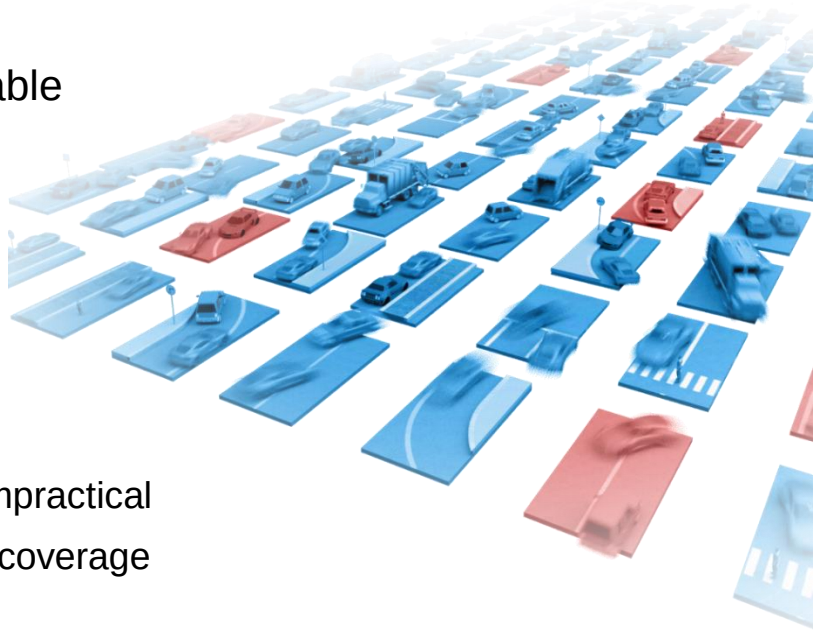




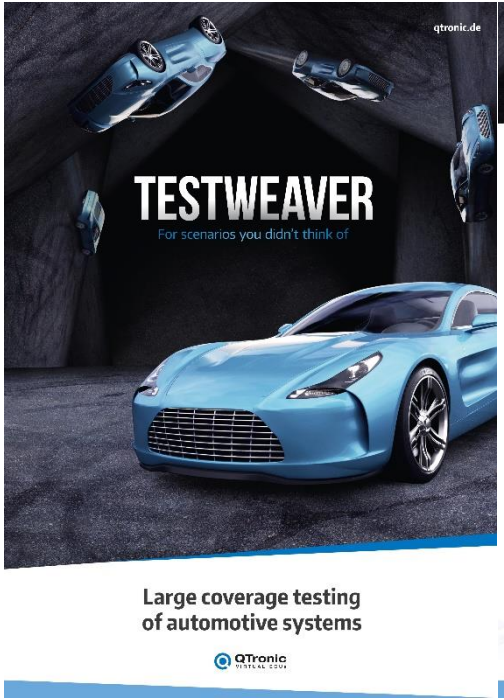


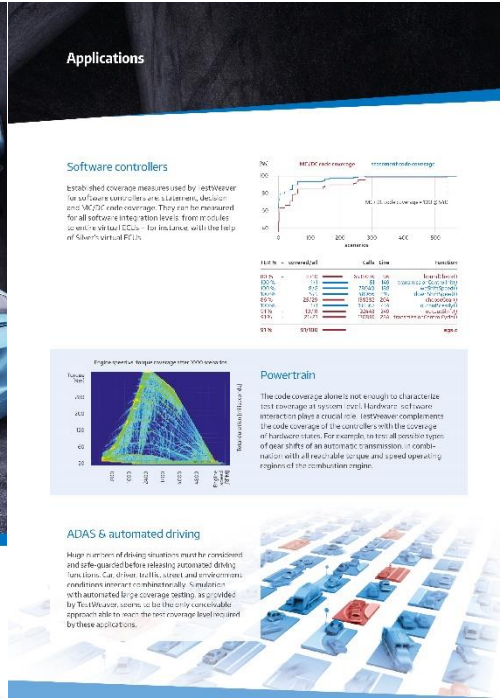
Future Work and Conclusion

- Problem dimensionality is a huge challenge
- No satisfying solutions that are generally applicable
- Future work:
  - Dimensionality reduction: sensitivity analysis, feature extraction, principal component analysis
  - Use of surrogate (learned) models for speed-up and characterization
- Nevertheless:
  - Focusing on the combinatorial coverage (S^N) is impractical
  - Need to come up with other <u>useful and accepted</u> coverage measures



Testing with TestWeaver





All presented exploration strategies available in

- **TestWeaver 3.6**
- release expected later this year

Thank you for the attention.

mugur.tatar(at)qtronic.de

See you in the AV Expo @ **AV10002**