Model-based software development

Lecture XII.

Model-based testing

Dr. Zoltán Micskei,
Dr. István Majzik,
Dr. Oszkár Semeráth

What is model-based testing?

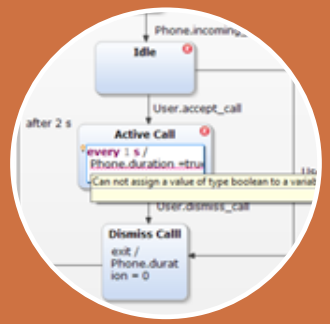
“Testing based on or involving models” [ISTQB]

- Not just test generation
- Not just automatic execution
- Not just for model-driven engineering

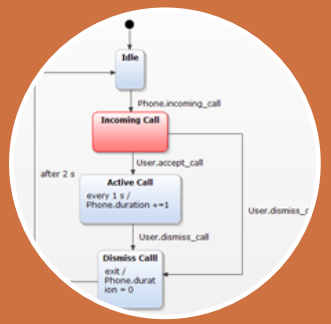
Source of definition: ISTQB. “Foundation Level Certified Model-Based Tester Syllabus”, Version 2015

Landscape of MBT goals

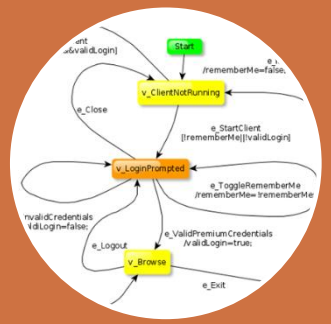
Shared under-standing



Checking specifications



Simulation



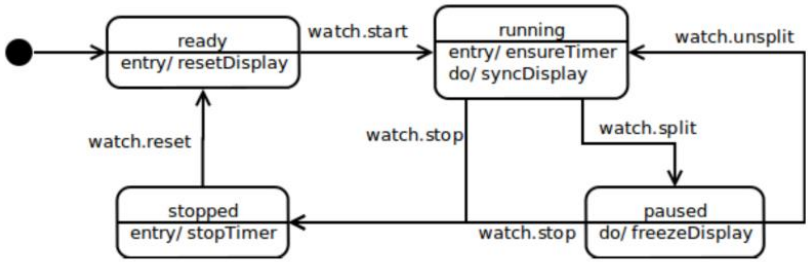
Test data creation

Executable tests

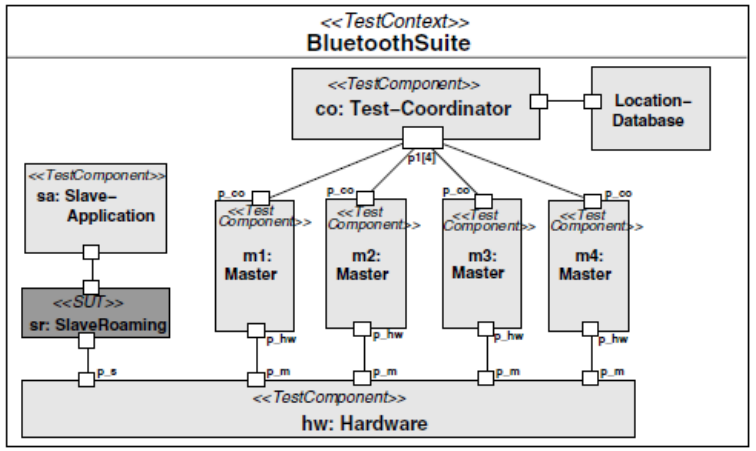
more informal

more formal

Using models in testing (examples)



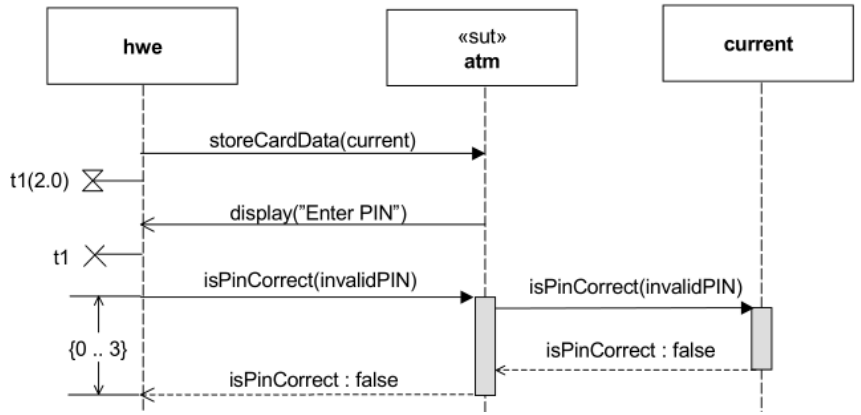
Behavior of SUT



Test configuration

```
timer t;
t.start(5.0);
alt {
 [] i.receive("coffee") {
  Count := Count+1; }
 [] t.timeout { }
}
```

Test sequences



Test sequences

Source: OMG UTP

Benefits of using models

- Close communication with stakeholders
  - > Understanding of domain and requirements
- Early testing: modeling/simulation/generation
- Higher abstraction level (manage complexity)
- Automation (different artefacts)

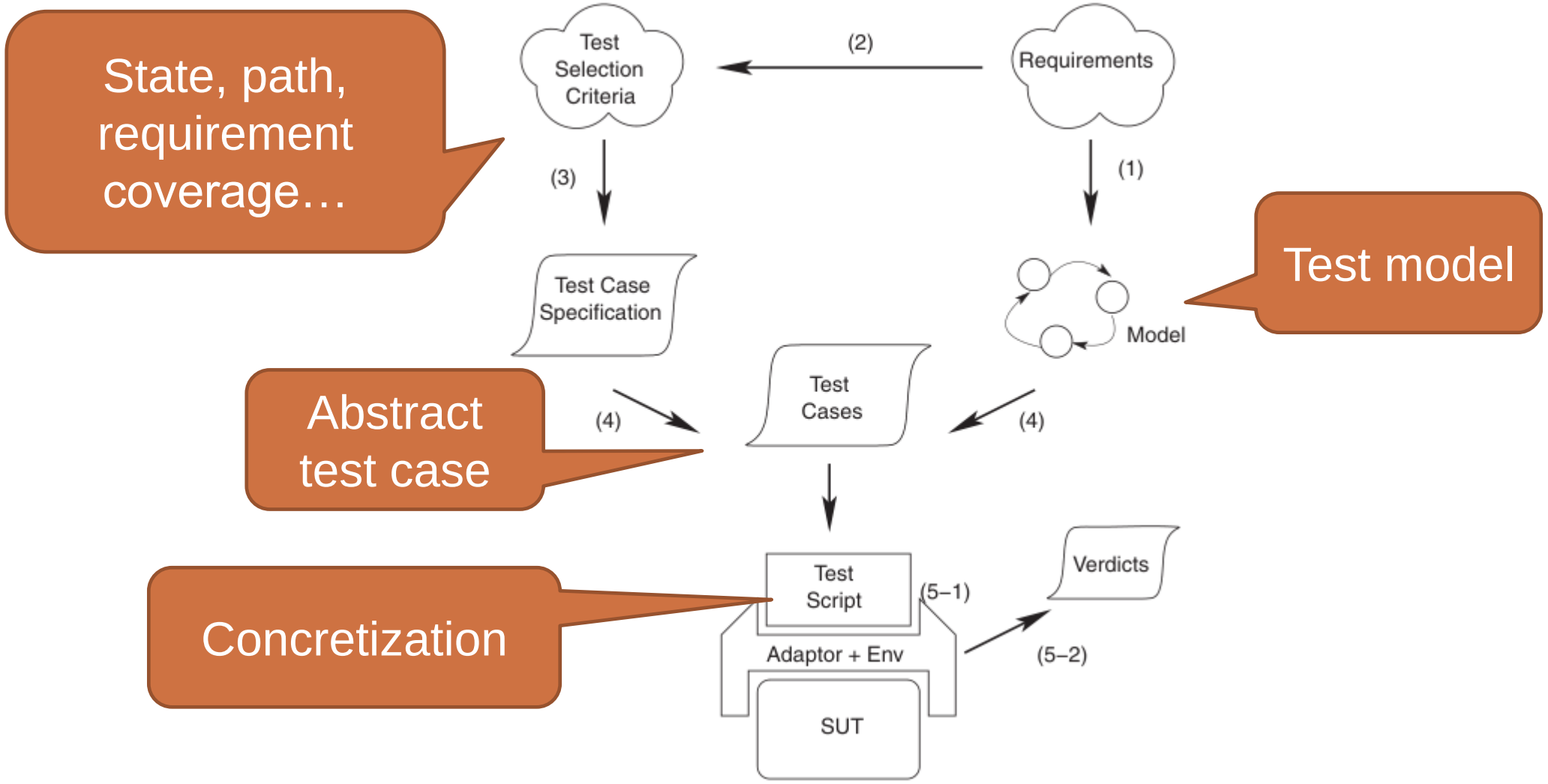
More specific meaning: Test generation

„MBT encompasses the processes and techniques for

- the automatic derivation of abstract test cases from abstract models,
- the generation of concrete tests from abstract tests,
- the manual or automated execution of the resulting concrete test cases”

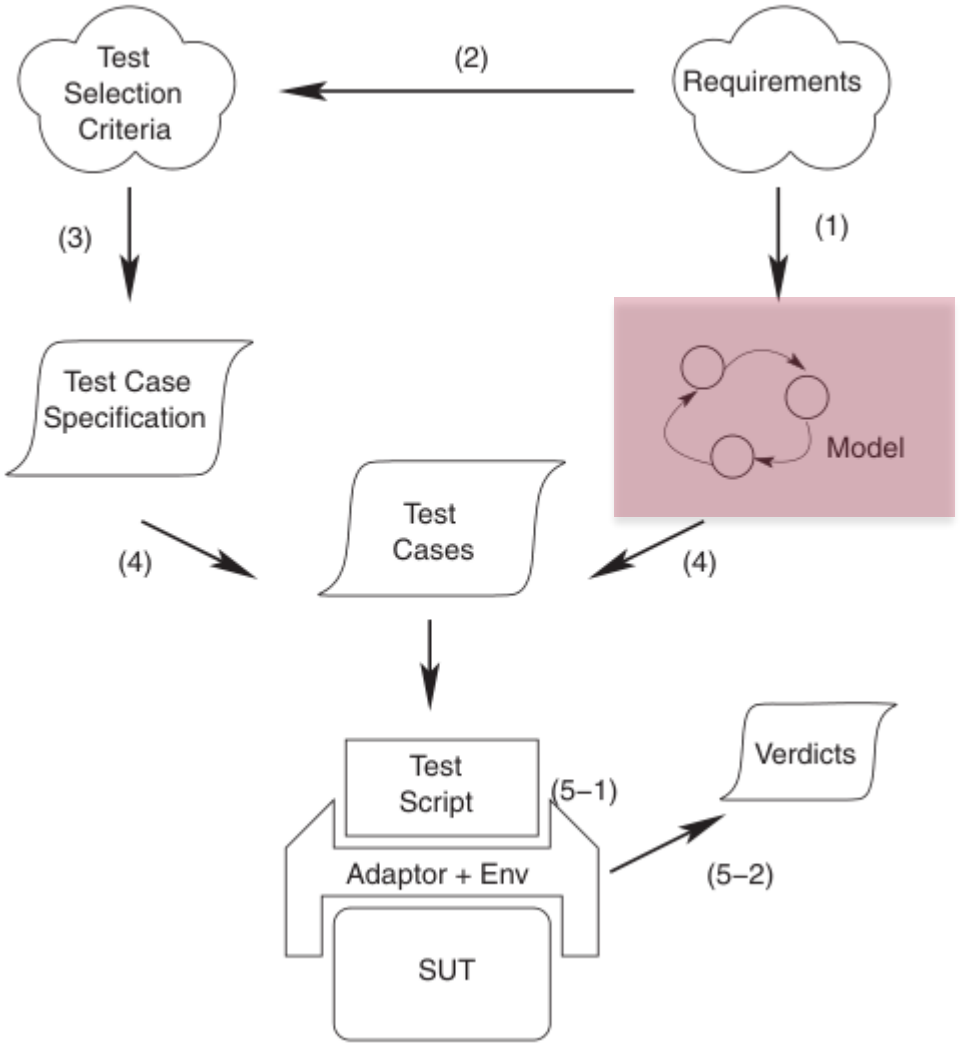
Source: M. Utting, A. Pretschner, B. Legeard. „A taxonomy of model-based testing approaches”, STVR 2012; 22:297–312

Typical MBT process



Source: M. Utting, A. Pretschner, B. Legeard. „A taxonomy of model-based testing approaches”, STVR 2012; 22:297–312

Typical MBT process



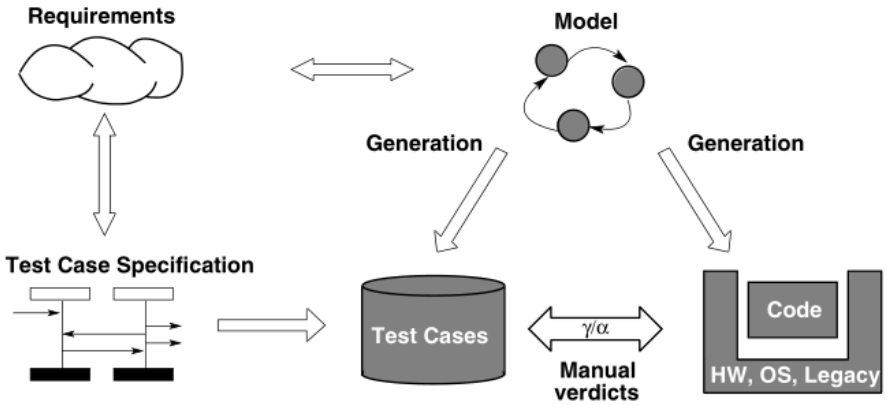
Source: M. Utting, A. Pretschner, B. Legeard. „A taxonomy of model-based testing approaches”, STVR 2012; 22:297–312

Questions for modeling

- What to model?
  - What is the test object?
  - Functionality / performance factors / …
- What abstraction level to use?
  - Too many or too few details
  - Separate models for different test objectives
- What modeling language to use?
  - Structural, behavioral

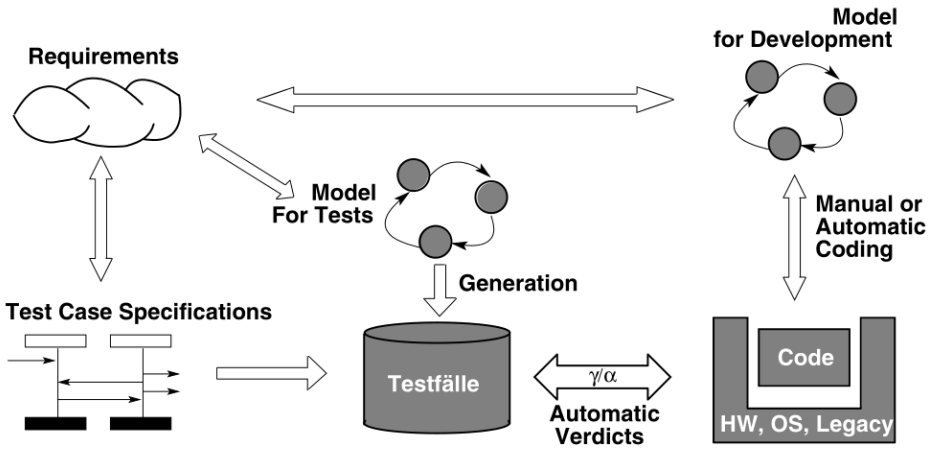
Reuse: Development and Test modeling

What if I have existing design models?



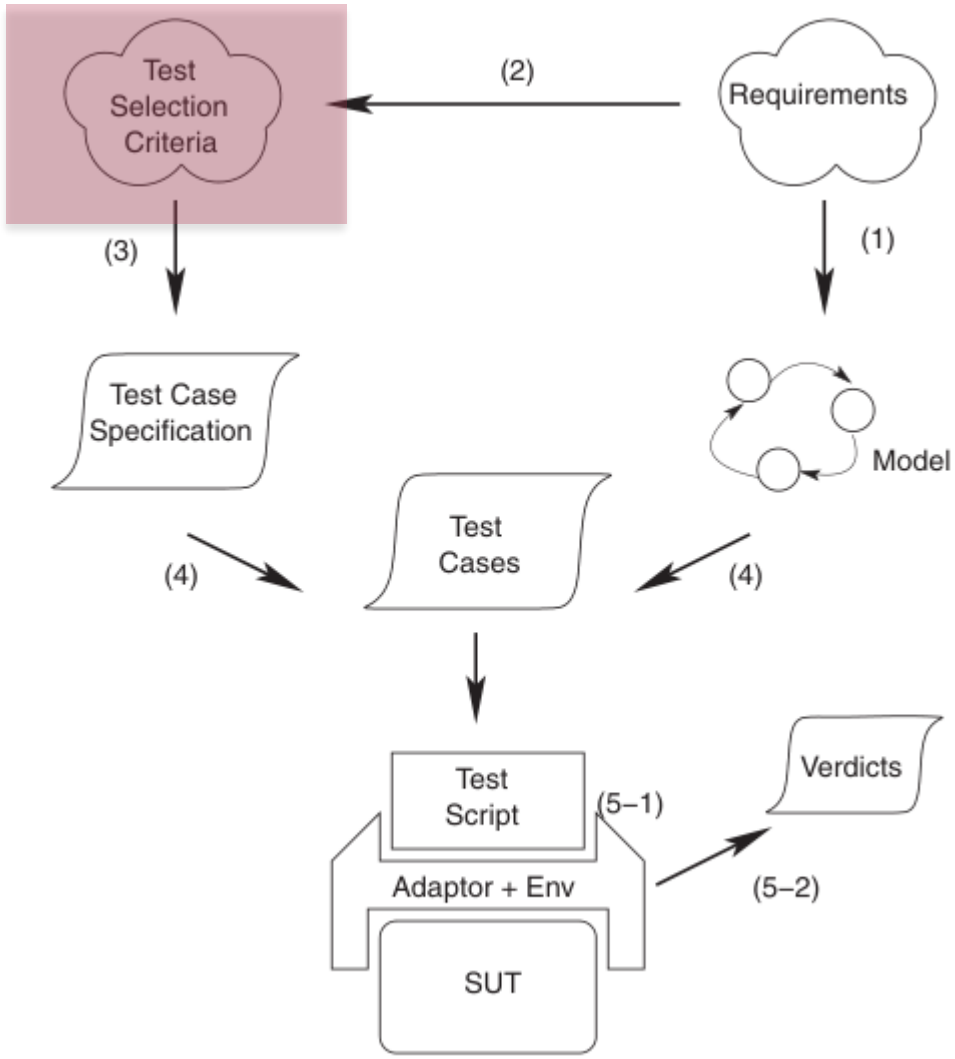
Problem: what do we test here?

Approach: separate dev. and test models



A. Pretschner, J. Philipps. „Methodological Issues in Model-Based Testing”, Model-Based Testing of Reactive Systems, 2005.

Typical MBT process

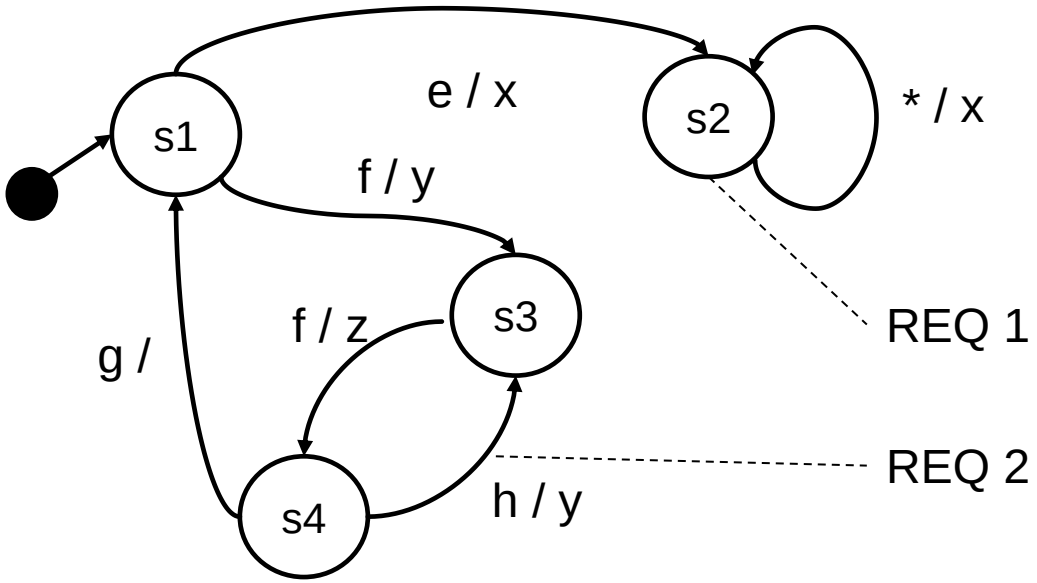


Source: M. Utting, A. Pretschner, B. Legeard. „A taxonomy of model-based testing approaches”, STVR 2012; 22:297–312

Typical test selection criteria

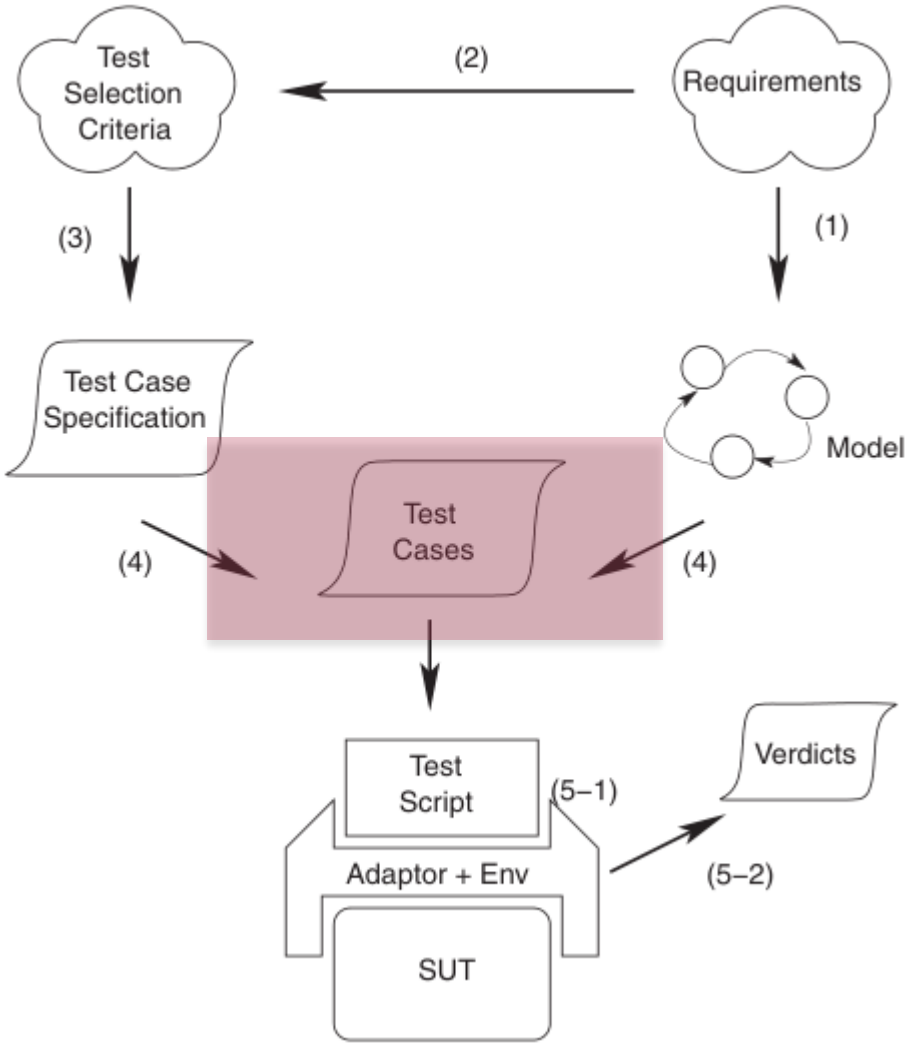
- Coverage-based
  - Requirements linked to the model
  - MBT model elements (state, transition, decision…)
  - Data-related (see spec. test design techniques)
  - Behaviour-related (component interaction)
- Random / stochastic
- Scenario- and pattern-based (use case…)
- Project-driven (risk, effort, resources…)

EXAMPLE Test selection for state models



- Select test cases for full
  - > requirement coverage
  - > state coverage
  - > transition coverage

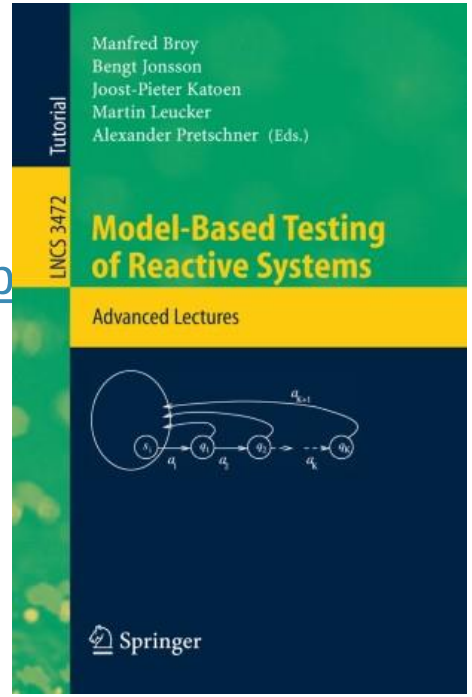
Typical MBT process



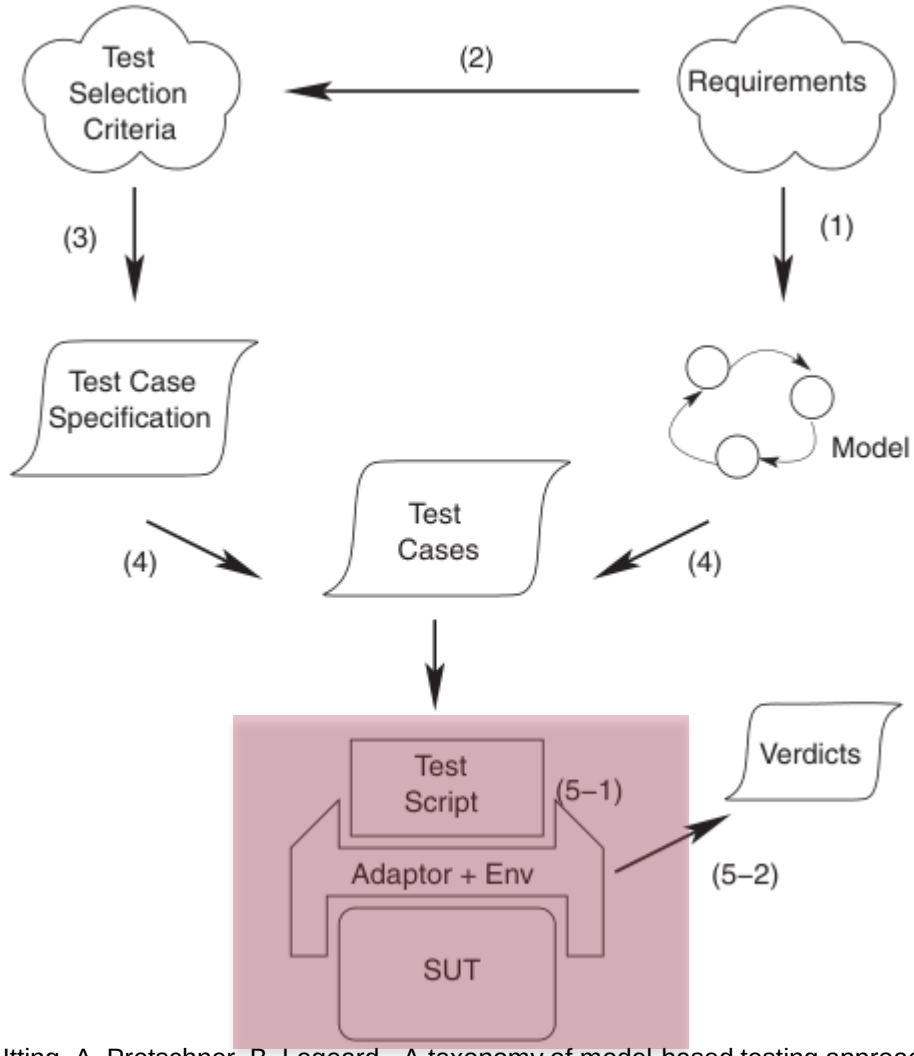
Source: M. Utting, A. Pretschner, B. Legeard. „A taxonomy of model-based testing approaches", STVR 2012; 22:297–312

Test generation methods (sample)

- Direct graph algorithms
  - Transition coverage → "New York Street Sweeper problem"
- Finite State Machine (FSM) testing
  - Homing and synchronizing sequences, (https://web.cecs.pdx.edu/~mperkows/CLASS_573/Kumar_2007/presentation.p
  - State identification and verification,
  - Conformance testing
- Labeled Transition System (LTS) testing
  - Equivalence and preorder relations, ioco
- Using model checkers
- Fault-based (mutation)



Typical MBT process

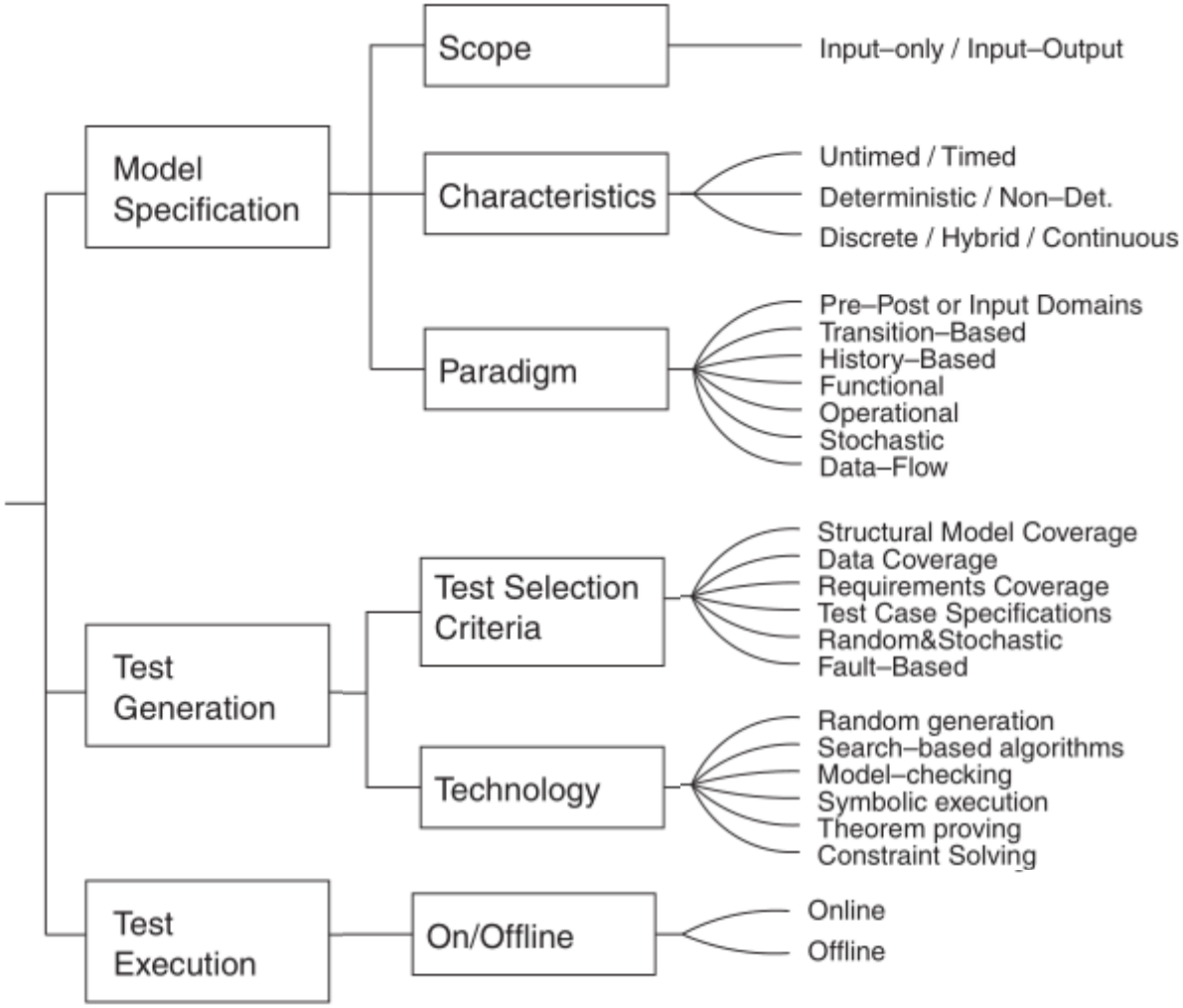


Source: M. Utting, A. Pretschner, B. Legeard. „A taxonomy of model-based testing approaches", STVR 2012; 22:297–312

Abstract and concrete test cases

- Abstract test case
  - Logical predicate instead of values
    (e.g., SLOW/FAST instead of 122.35)
  - High-level events and actions
- Concrete test case
  - Concrete input data, precize enough to test the system
  - Detailed test procedure (manual or automatic)
- Execution:
  - Test environment, in which the concrete test can be executed
  - Wrapper around the SUT

Summary: Taxonomy of MBT approaches



Source: M. Utting, A. Pretschner, B. Legeard. „A taxonomy of model-based testing approaches", STVR 2012; 22:297–312

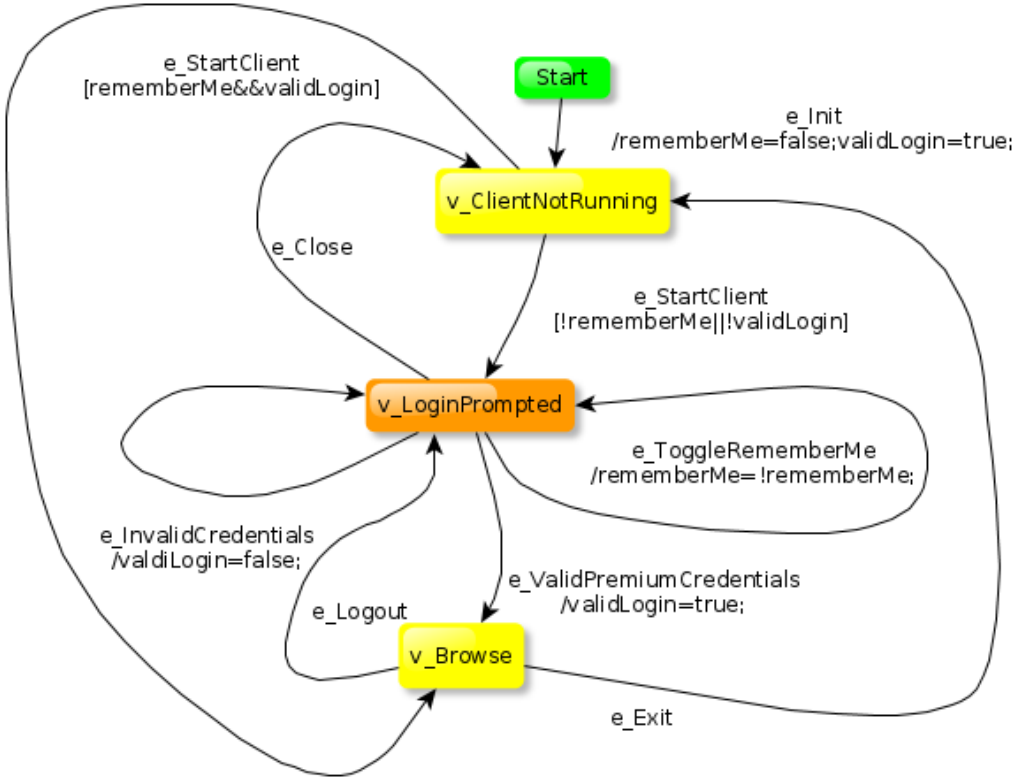
Tools

- Look at some examples!

- **Fast & easy:** simple, open-source tools
- **Full fledged:** commercial tools, full lifecycle support
- **Advanced:** custom modeling languages/tools

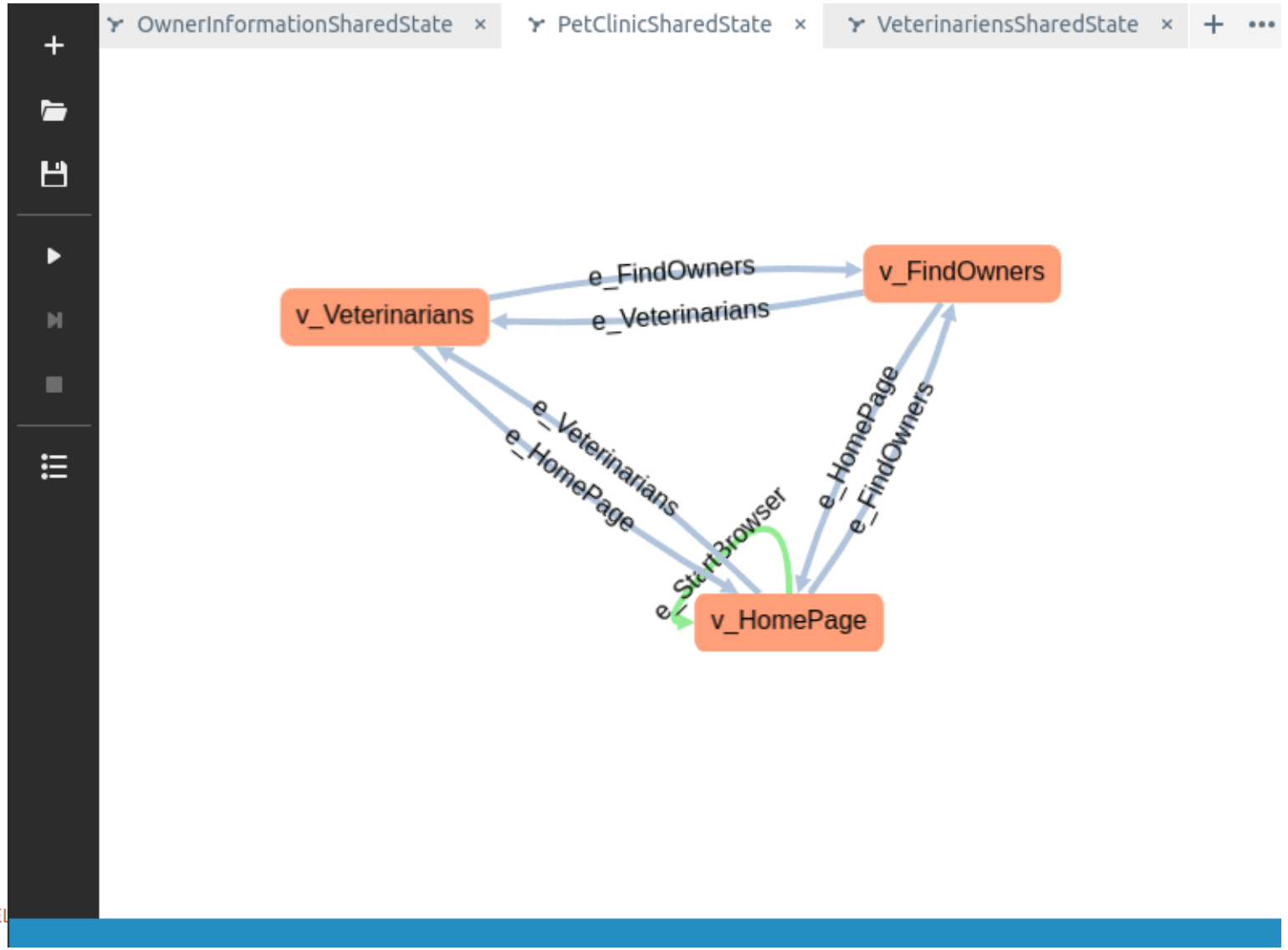
GraphWalker

- FSM model + simple guards
- Coverage:
  - state,
  - transition,
  - time limit (random walk)
- Traversing the graph:
  - random,
  - A*,
  - shortest path
- Generating JUnit test stubs (adapter)



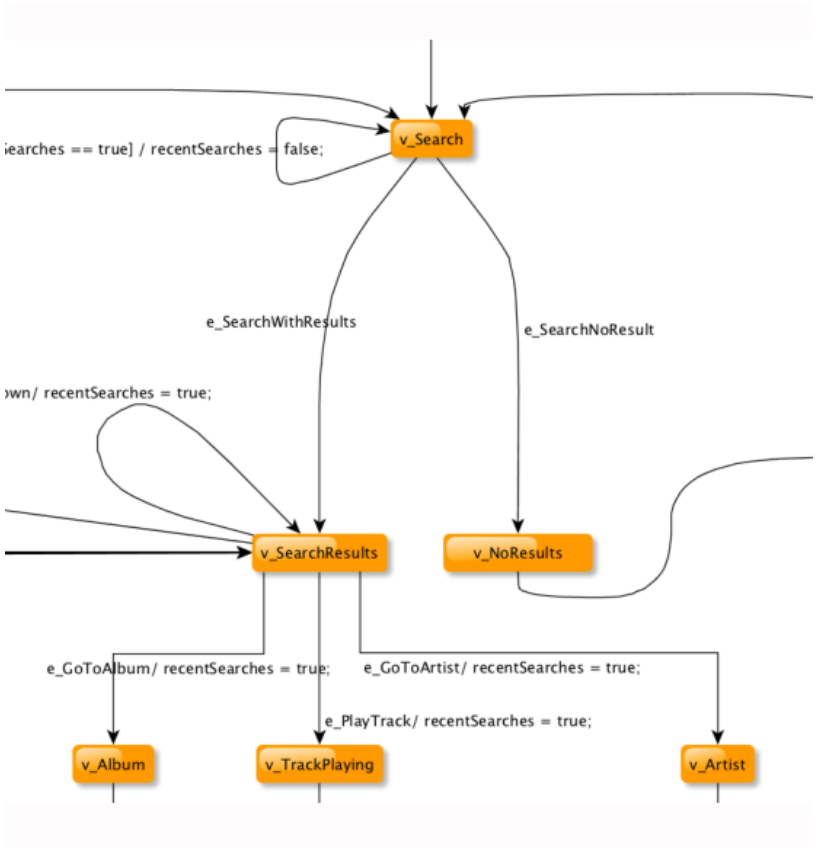
Source: GraphWalker

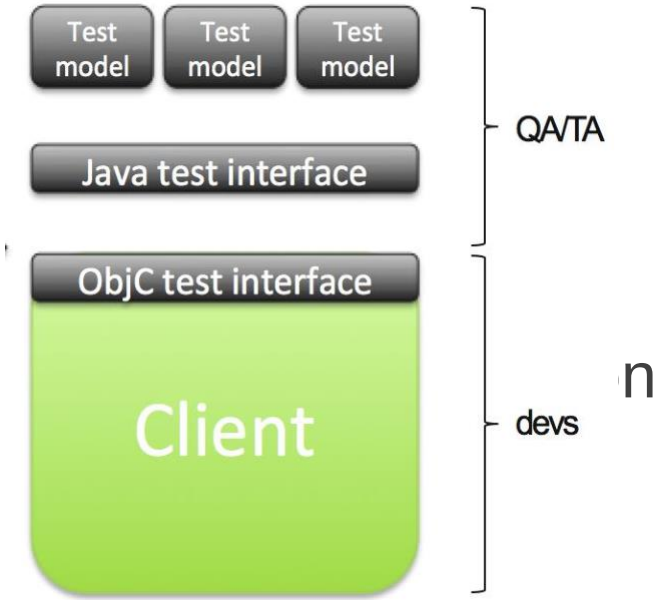
Online execution



Case study: Spotify

Model + GraphWalker

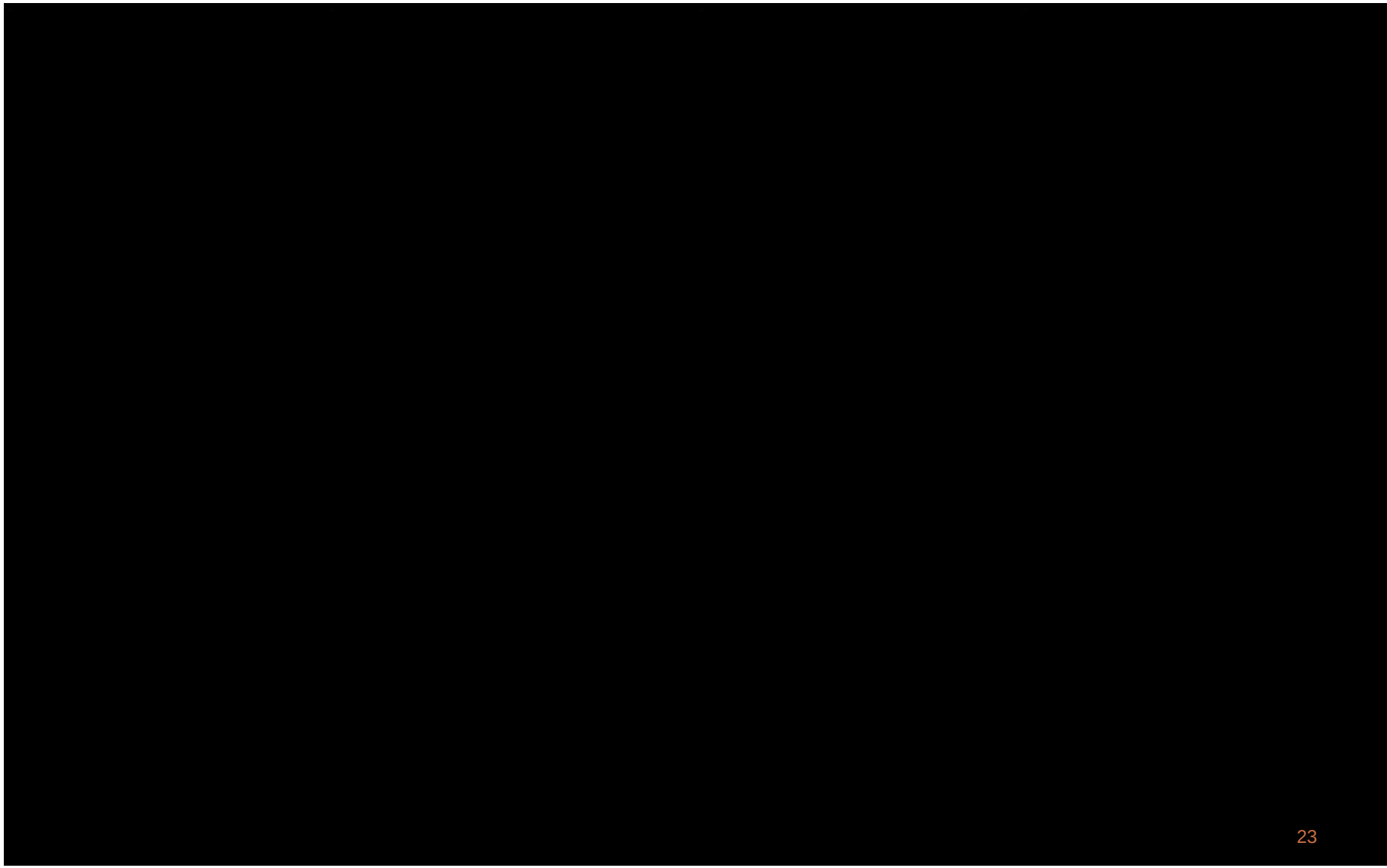




Test automation and Model-Based Testing in agile dev cycle @ Spotify, UCAAT 2013

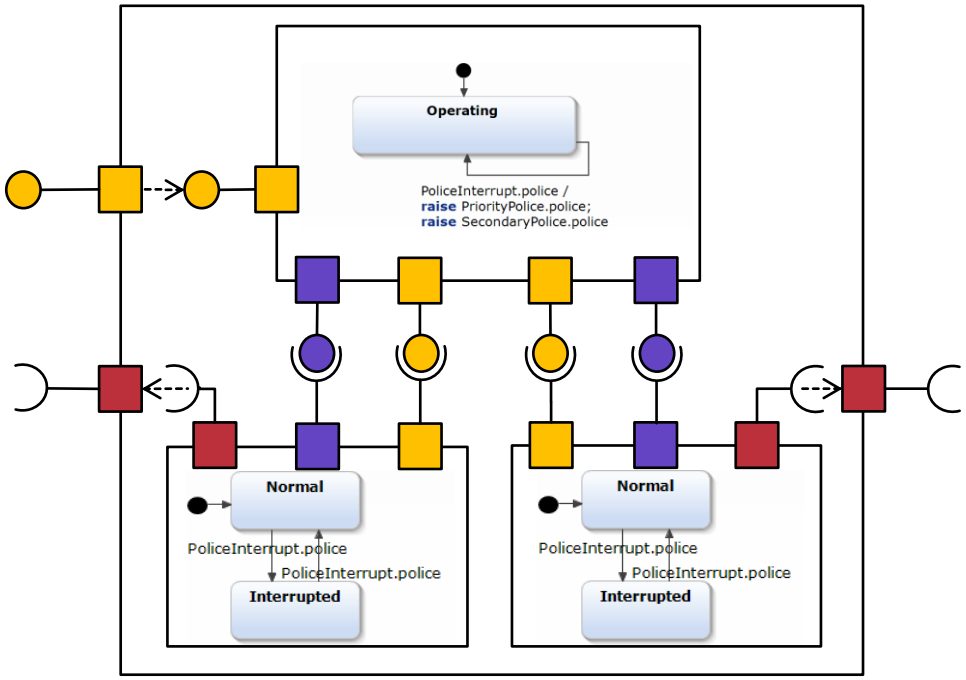
Case Study: web testing

https://www.youtube.com/watch?v=y251HEaTnus



Example: Gamma framework

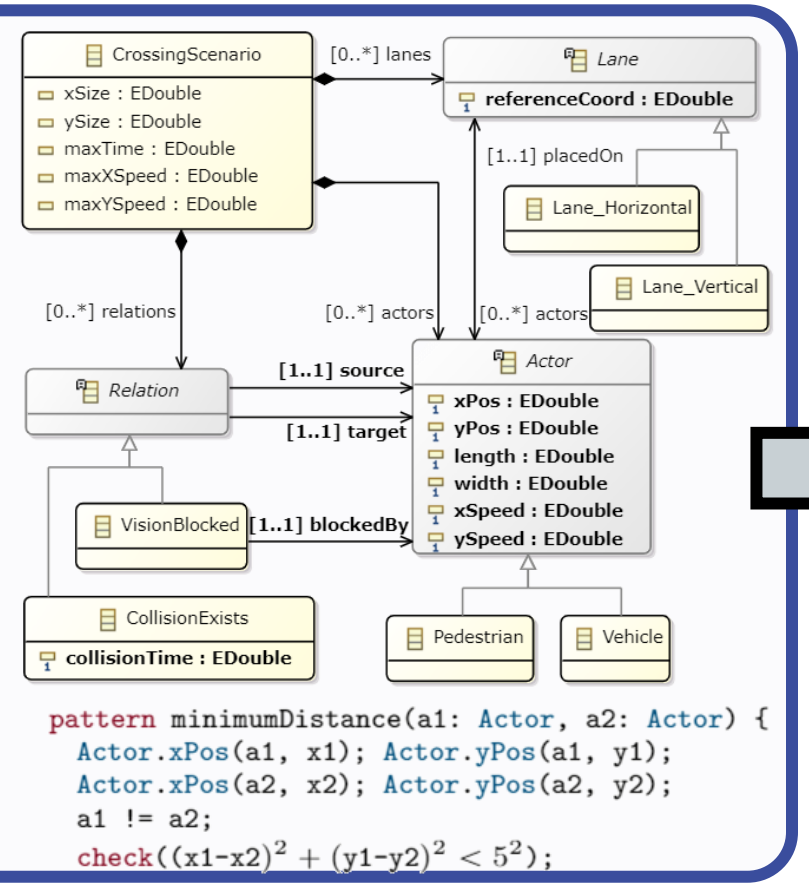
- UML/SysML-based *statecharts* (*GSL*) + *topology* (SysML *ibd*) *descriptions* (*GCL*)
- Test coverage criteria
  - State, transition, transition-pair
  - Interaction
  - Dataflow
- Traversing the model using *model checkers*
  - UPPAAL
  - Theta
- Generating abstract test cases (GTL) and concrete JUnit tests
  - Reflective Java API



https://github.com/ftsrg/gamma

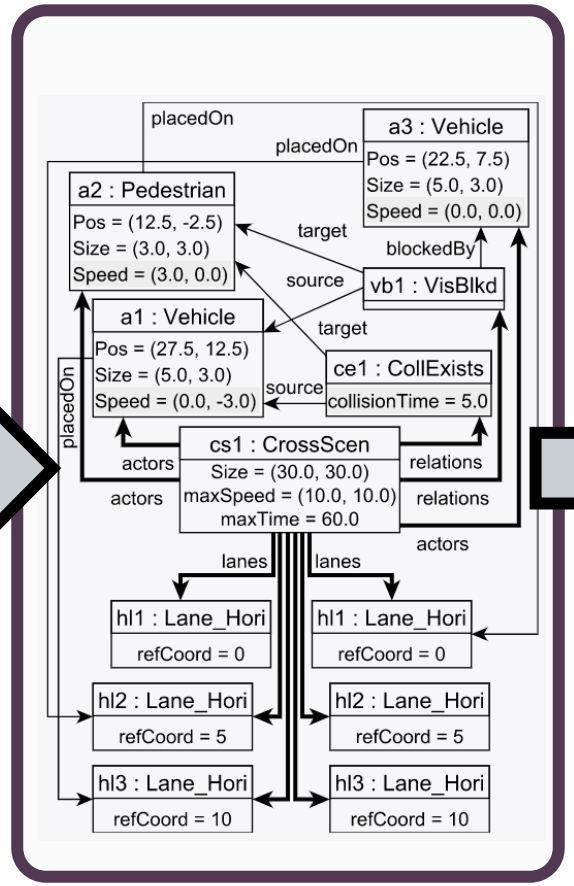
Case study: AV testing

Modeling Dangerous Situations

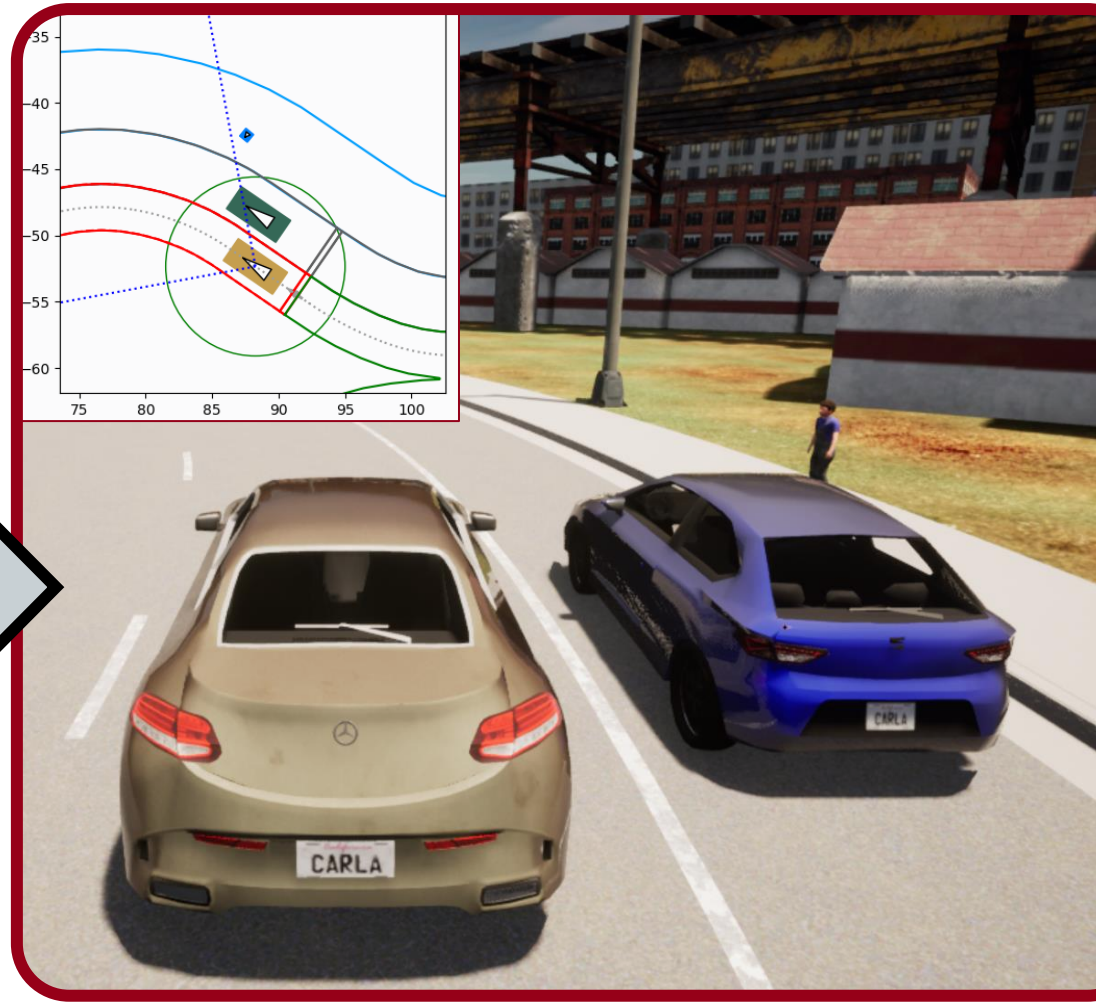



```
pattern minimumDistance(a1: Actor, a2: Actor) {
  Actor.xPos(a1, x1); Actor.yPos(a1, y1);
  Actor.xPos(a2, x2); Actor.yPos(a2, y2);
  a1 != a2;
  check((x1-x2)^2 + (y1-y2)^2 < 5^2);
}
```

Instantiating situations



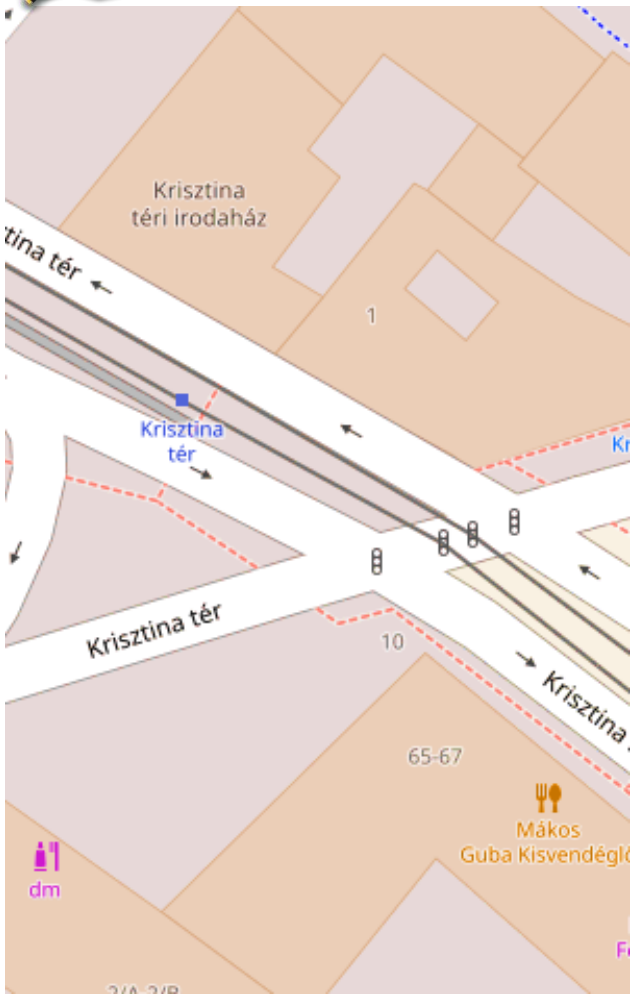
Running the scenarios



Real-world location

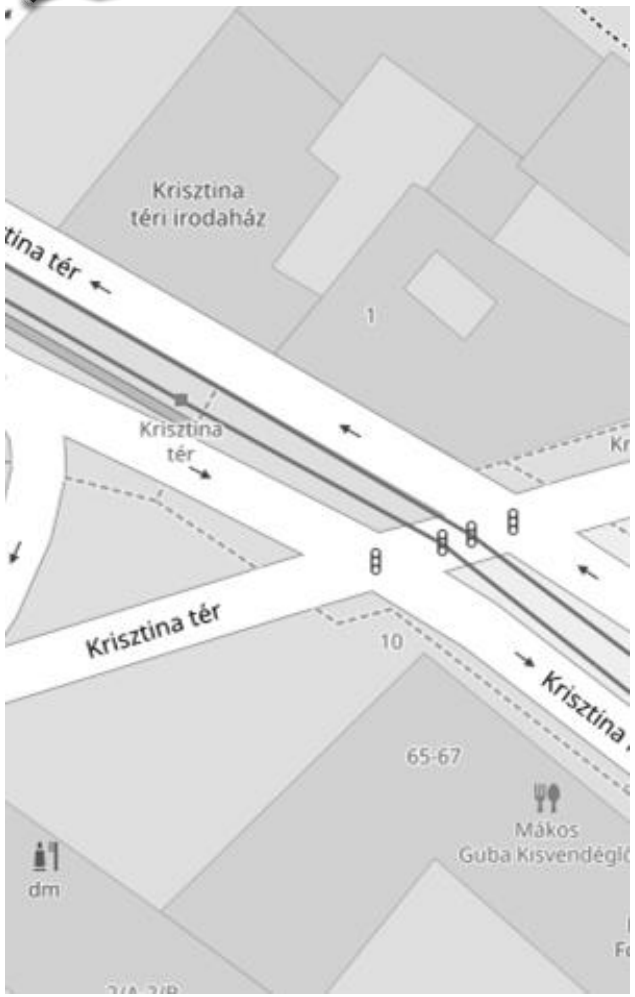
Real-world location

Map Import OpenStreetMap



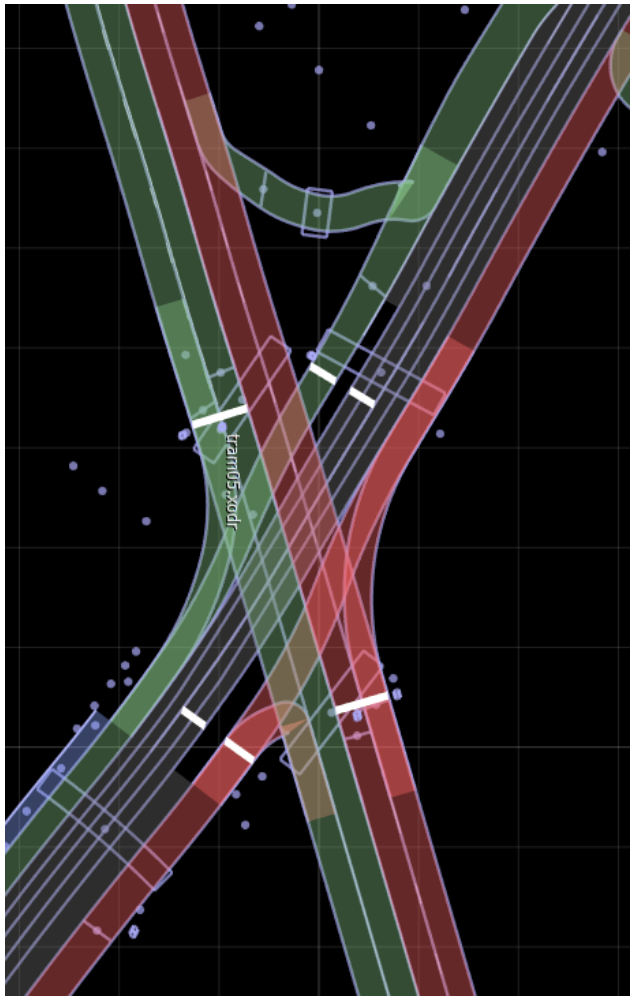
Real-world location

Map Import OpenStreetMap



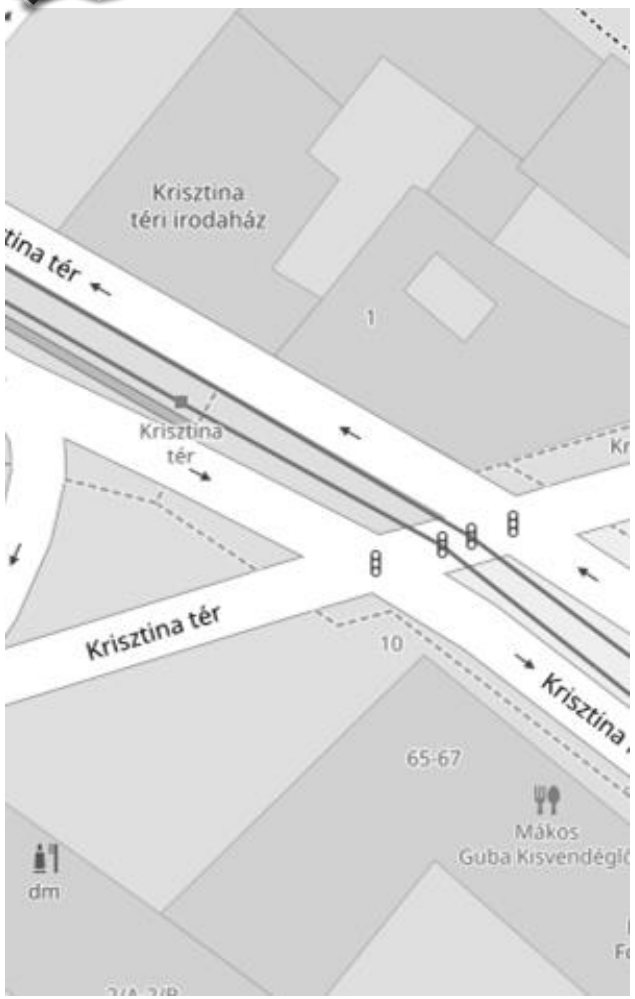
Road topology import

OpenDrive



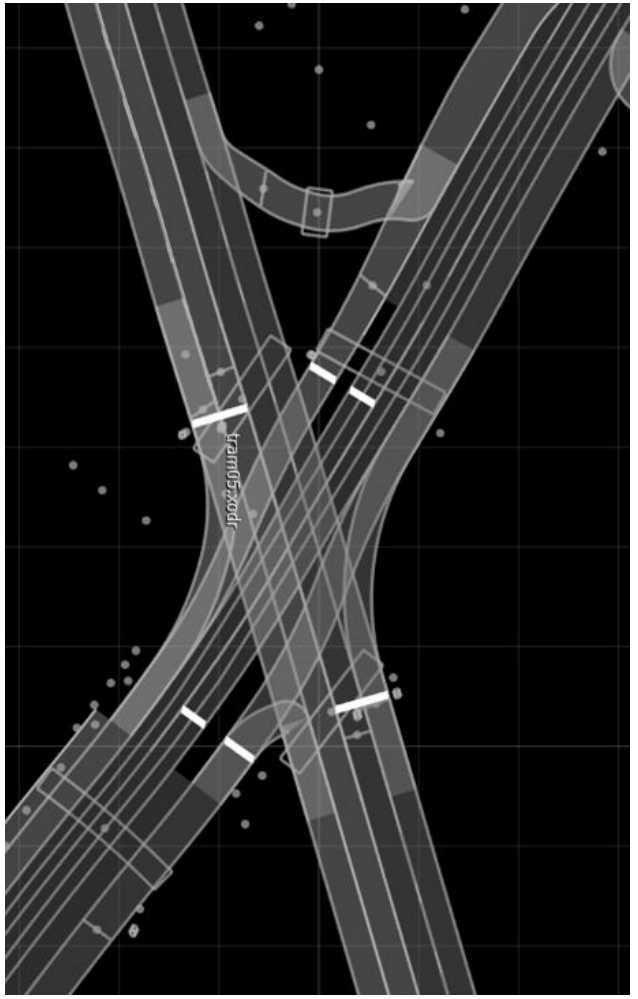
Real-world location

Map Import OpenStreetMap



Road topology import

OpenDRIVE

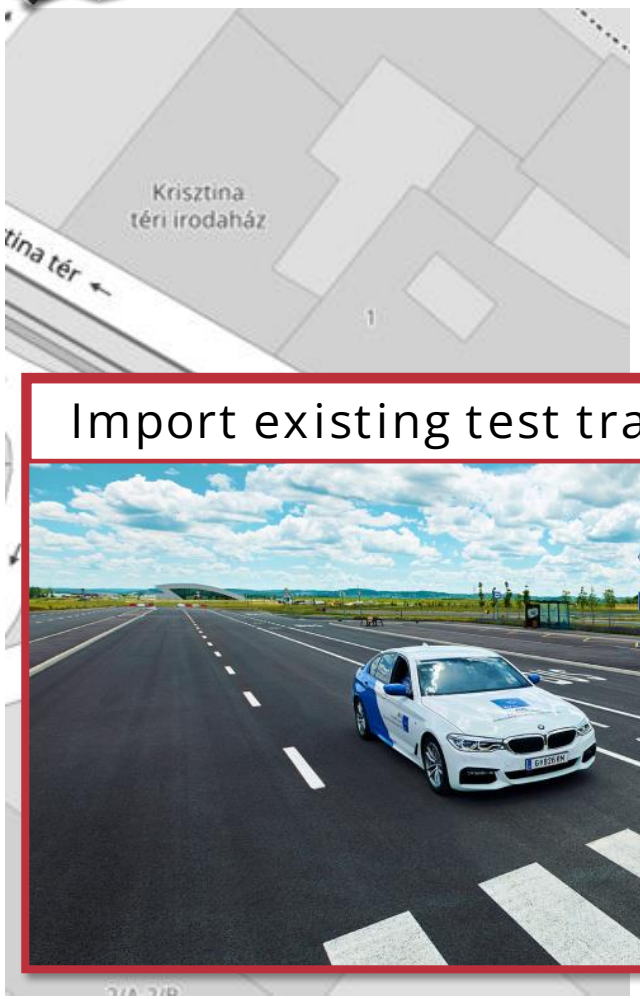


Adding buildings, signs Google Maps, Roadrunner

VECTORZERO

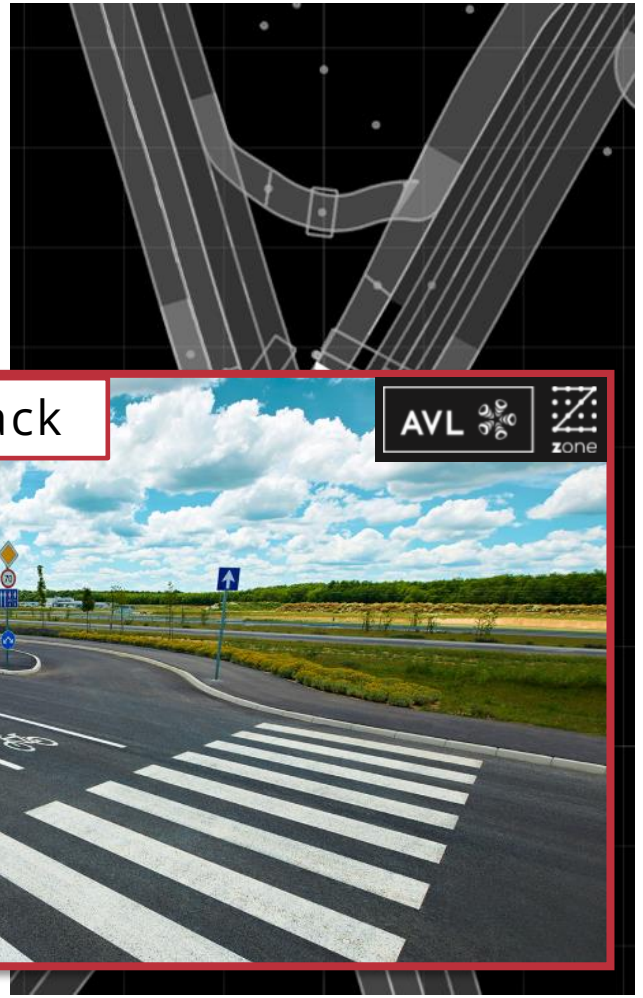
Real-world location

Map Import
OpenStreetMap



Road topology import

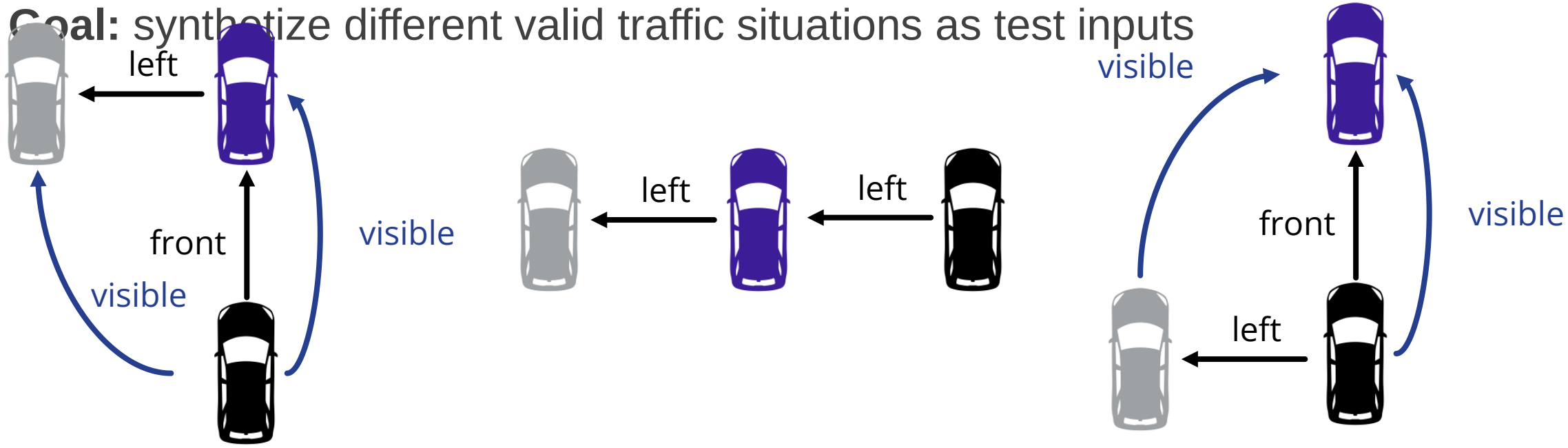
OpenDrive



Adding buildings, signs
Google Maps, Roadrunner

Import existing test track

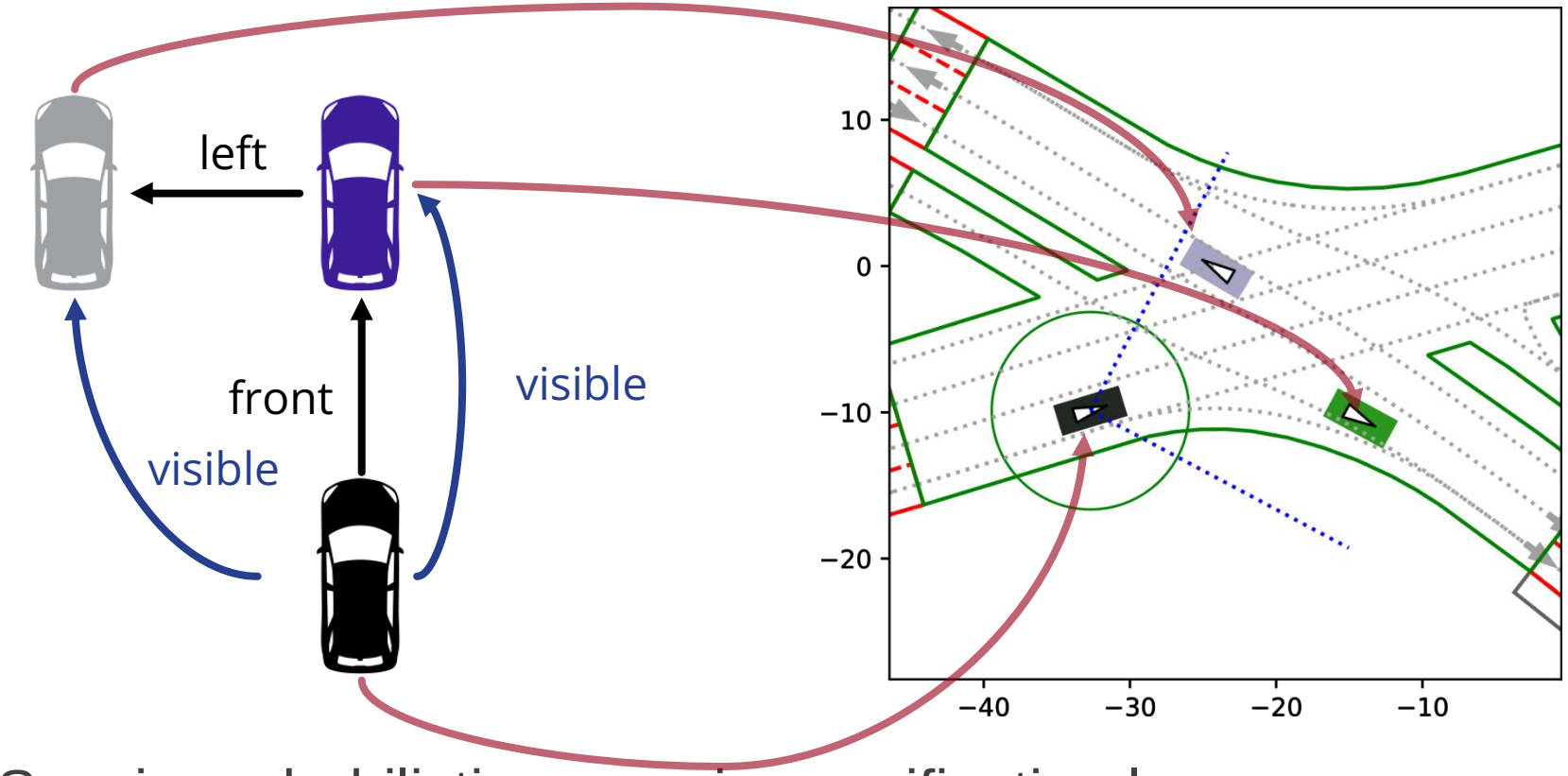
- **Goal:** synthetize different valid traffic situations as test inputs



Structurally different situations ⇒ Semantically different scenes

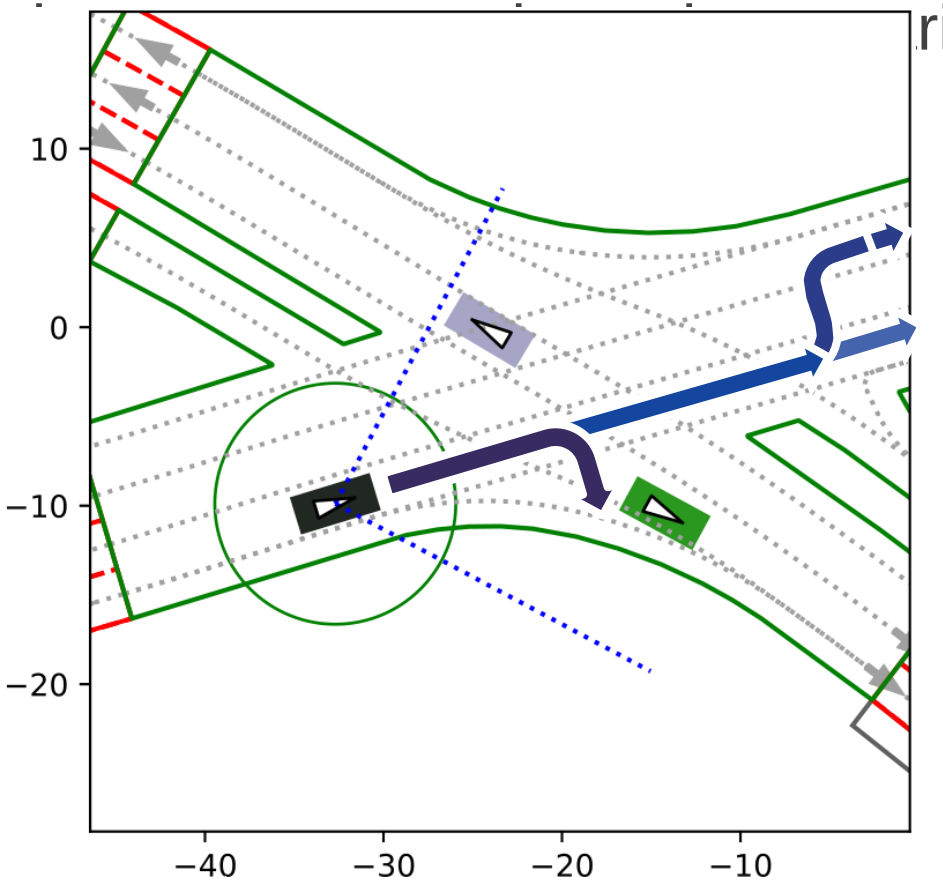
- **Solution:** graph generation with VIATRA Solver / Refinery

- **Goal:** allocate the situation on the map

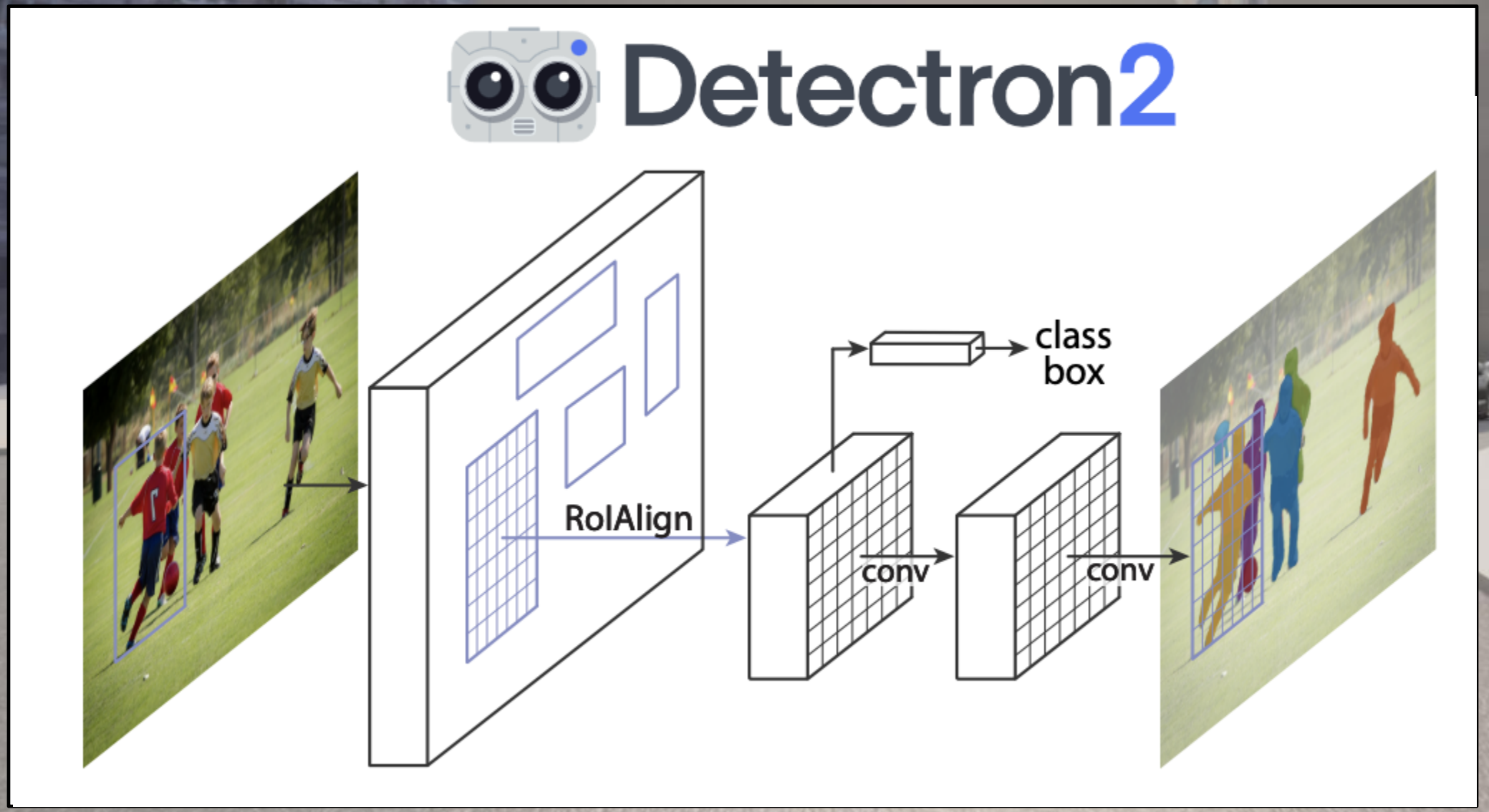


- **Solution:** Scenic probabilistic scenario specification language

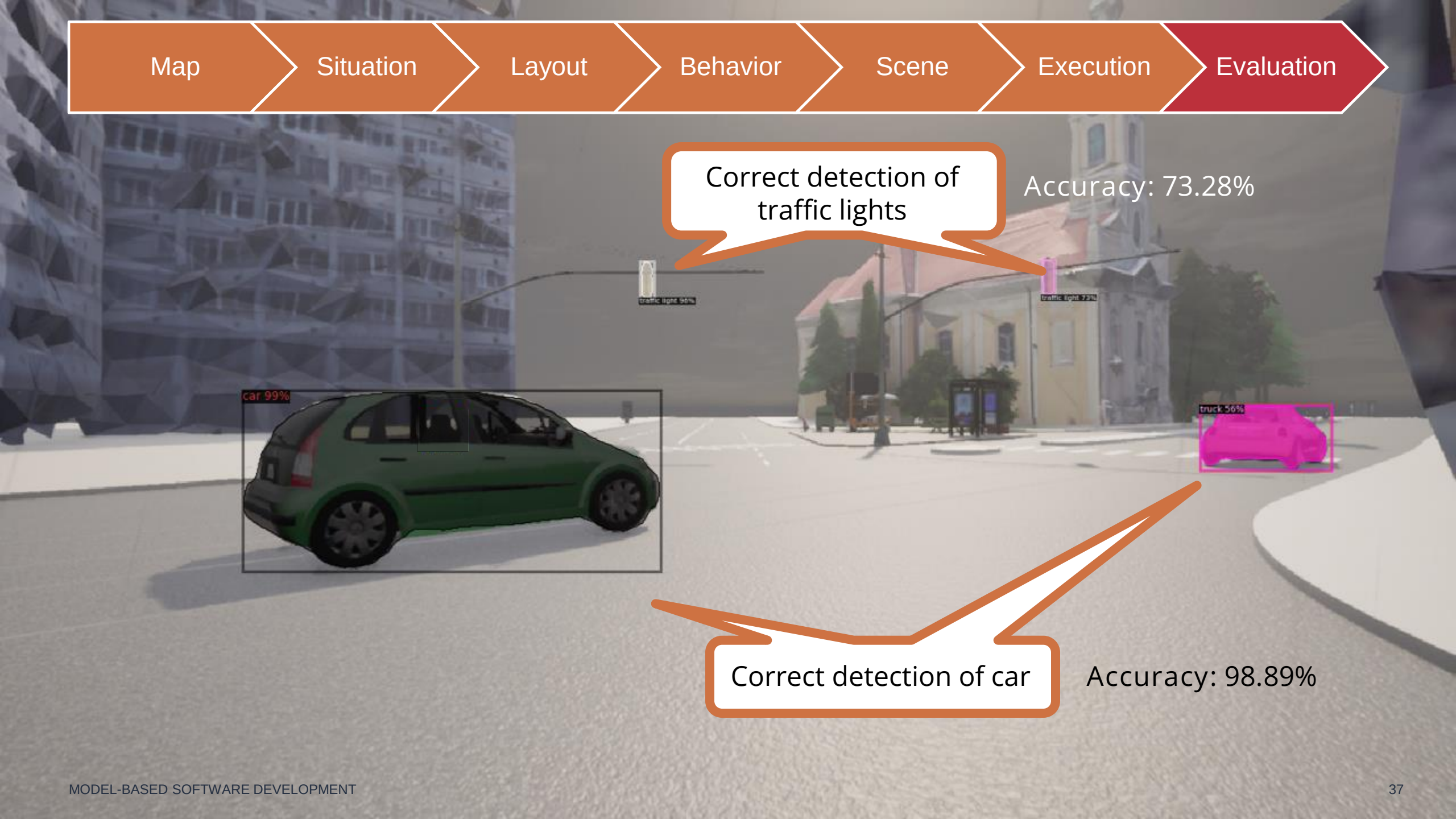
- **Goal:** assign tasks to rios




CARLA

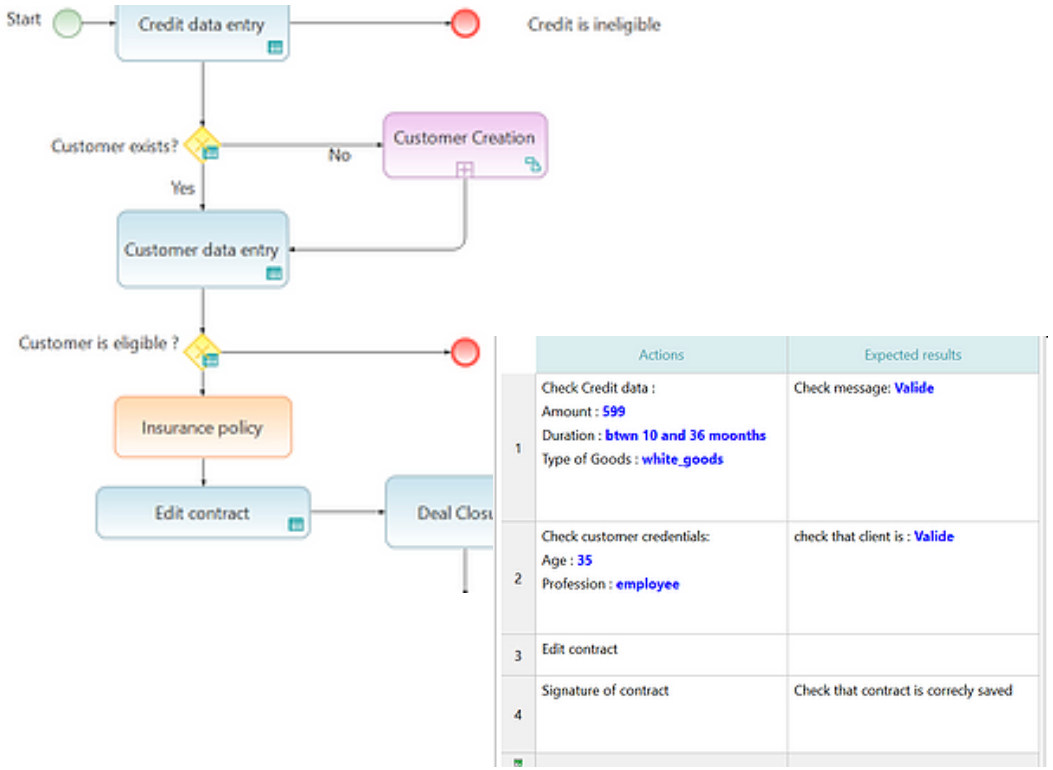


He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.


Correct detection of traffic lights
Accuracy: 73.28%
car 99%
truck 56%
Correct detection of car
Accuracy: 98.89%

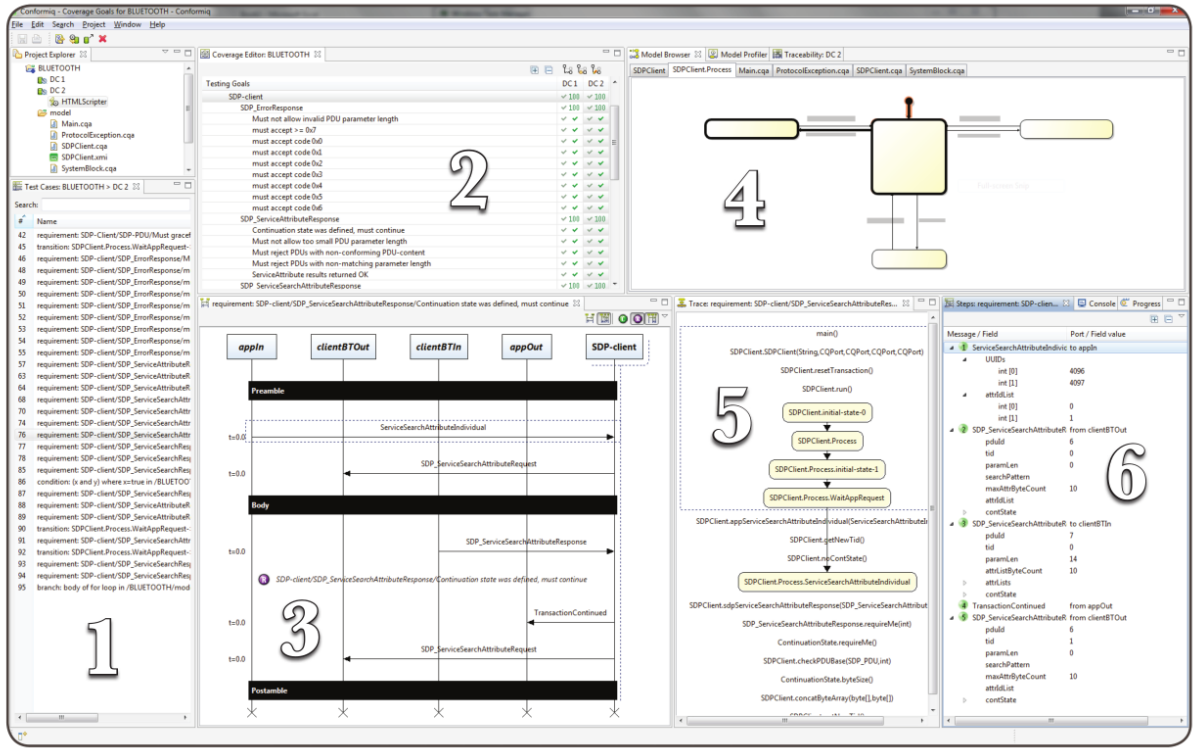
Industrial examples

Smartesting Yest: workflow



	Actions	Expected results
1	Check Credit data : Amount : 599 Duration : btwn 10 and 36 moonths Type of Goods : white_goods	Check message: Valide
2	Check customer credentials: Age : 35 Profession : employee	check that client is : Valide
3	Edit contract	
4	Signature of contract	Check that contract is correcly saved

Conformiq: state machines



Conformiq Designer IDE for automatic test case generation

Source: Conformiq. „Testing Bluetooth Protocol Stacks with Computer-Generated Tests". Technology brief. 2010

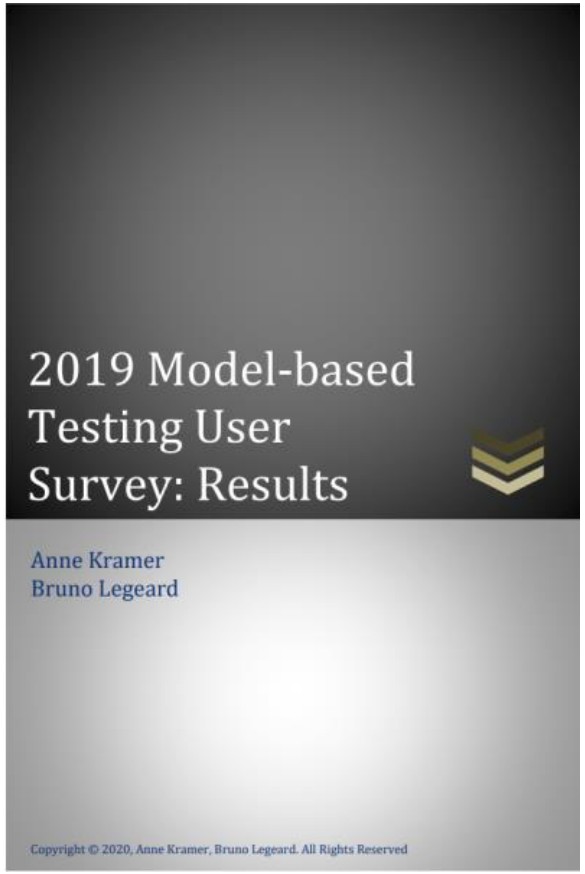
List of tools:

http://mit.bme.hu/~micskeiz/pages/modelbased_testing.html

MBT User Survey

Answer Options	2019
Acceptance testing	51,7%
System testing	79,3%
Integration testing	51,7%
Component (or unit) testing	10,3%

Answer Options	2019
Test cases (for manual test execution)	66,7%
Test scripts (for automated test execution)	70,8%
Test data	12,5%
Other artifacts (documentation, test suites,...)	20,8%



- “approx. 80h needed to become proficient”
- MBT is effective
- Lots of other details!

Source: https://www.cftl.fr/wp-content/uploads/2020/02/2019-MBT-User-Survey-Results.pdf

Thank you for your attention