CyPhyAssure Spring School

Second Generation Model-based Testing

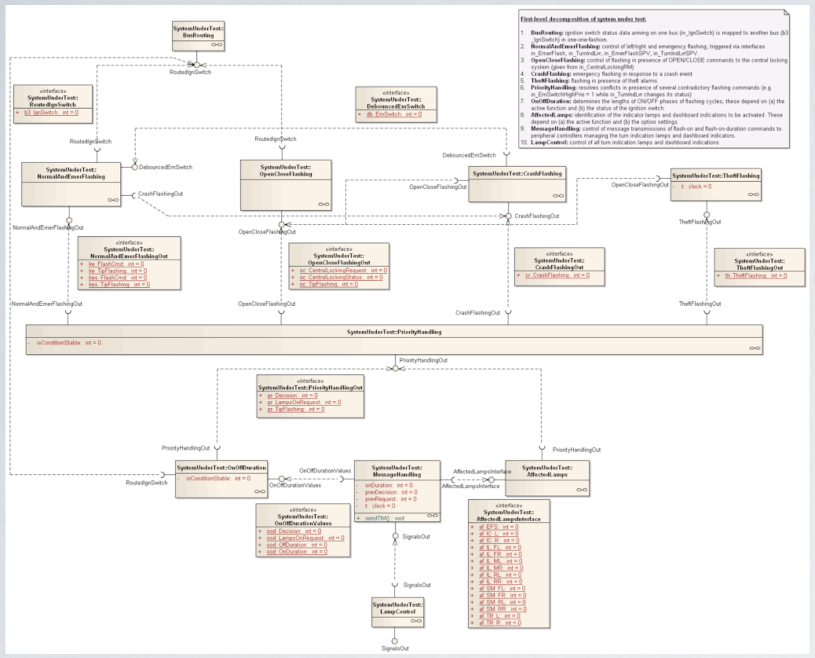
Provably Strong Testing Methods for the Certification of Autonomous Systems
Part I of III –
Motivation and Challenges

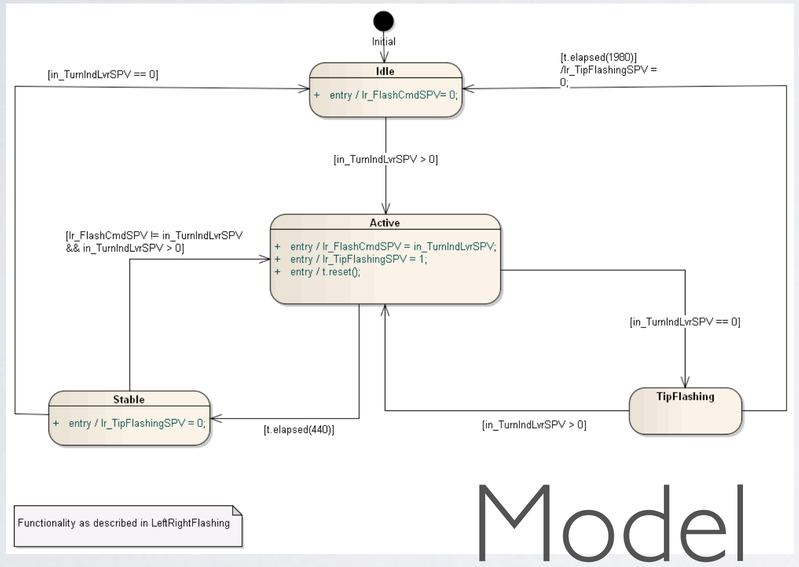
Jan Peleska
University of Bremen and Verified Systems International GmbH
peleska@uni-bremen.de
2019-03-20

Background

Background

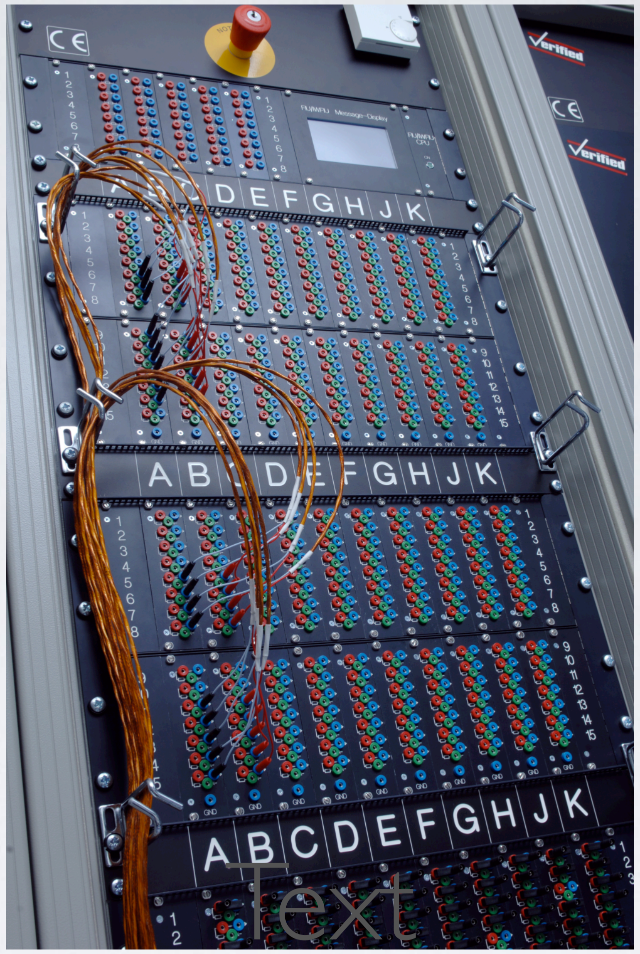
- Jan Peleska's research team at the University of Bremen
  - Formal methods for modelling and model checking
  - Model-based testing (MBT) with strategies providing guaranteed error detection capabilities
- Verified Systems International GmbH
  - Specialised on Verification&Validation (V&V) of safety-critical cyber-physical systems
  - Main customers from avionic domain, railways, automotive





Initial
Idle
+ entry / lr_FlashCmdSPV= 0;
[in_TurnIndLvrSPV == 0]
[t.elapsed(1980)] /lr_TipFlashingSPV = 0;
[in_TurnIndLvrSPV > 0]
[lr_FlashCmdSPV != in_TurnIndLvrSPV && in_TurnIndLvrSPV > 0]
Active
+ entry / lr_FlashCmdSPV = in_TurnIndLvrSPV;
+ entry / lr_TipFlashingSPV = 1;
+ entry / t.reset();
[in_TurnIndLvrSPV == 0]
Stable
+ entry / lr_TipFlashingSPV = 0;
[t.elapsed(440)]
[in_TurnIndLvrSPV > 0]
TipFlashing
Functionality as described in LeftRightFlashing

Model

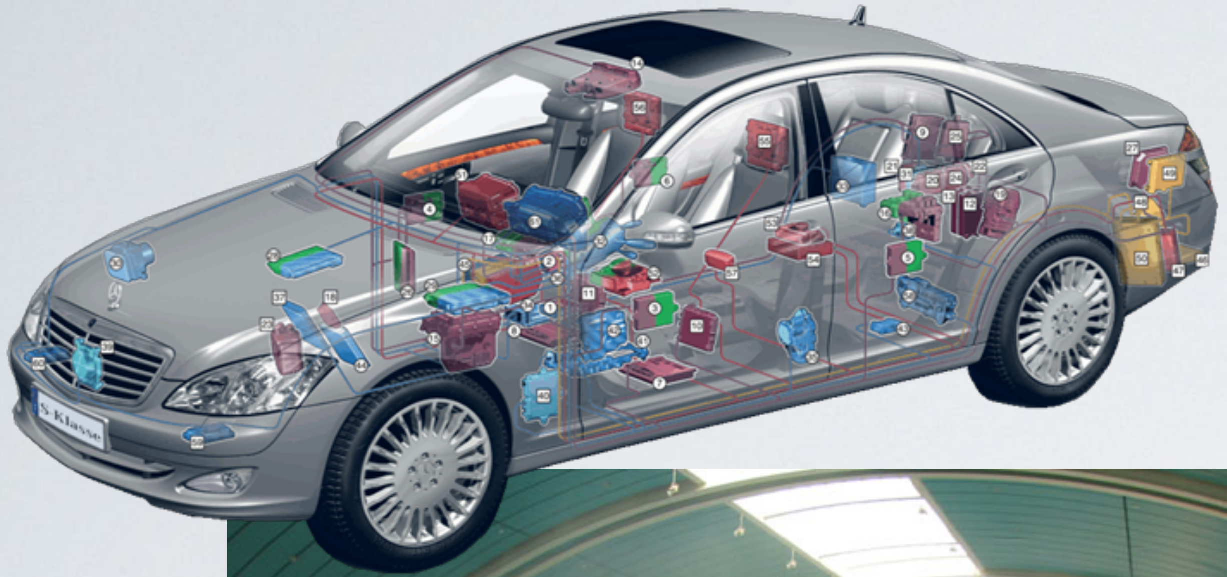


Test Generator
Test Engine

System
Under Test

RESEARCH FOCUS

Verification, validation and test of safety-critical embedded systems – model-based testing — real-time operating systems



APPLICATIONS

Main focus on transportation systems

avionics — railways — automotive — space systems

A Remark on this Spring School

- Quality-related standards (e.g. ISO 9001) require
  - **Quality awareness** – in particular safety/security awareness in the context of safety/security-critical systems
  - **Continual improvement**
- This spring school is an excellent means to support this with respect to the new challenges of autonomous systems

Recall – Some Facts About Testing

Validation vs. Verification

- **Validation**. Determine that the SUT is fit for its intended purpose
- **Verification**. Determine that the SUT conforms to a specification
- This induces 2 categories of testing

Test Strategies

- A **test strategy** is a method to create test cases promising an acceptable **fault coverage** (= error detection capability)
- Test strategies can be derived, for example, from
  - models specifying the expected behaviour of the SUT
  - structural SUT models, including code structure (“programs are models”)
  - interface specifications (e.g. pairwise testing)

Test Cases

- **Definition.** A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

RTCA DO-178B/C

- **Concrete test case.** Inputs, conditions, expected results are explicit values

- **Symbolic (abstract) test case.** Inputs, conditions, expected results are logical formulas, so that every solution is a concrete test case

Test Oracles

- Recall that a **test oracle** is the (preferably automated) testing component deciding whether the SUT reactions to the input stimulations conform to the expected behaviour

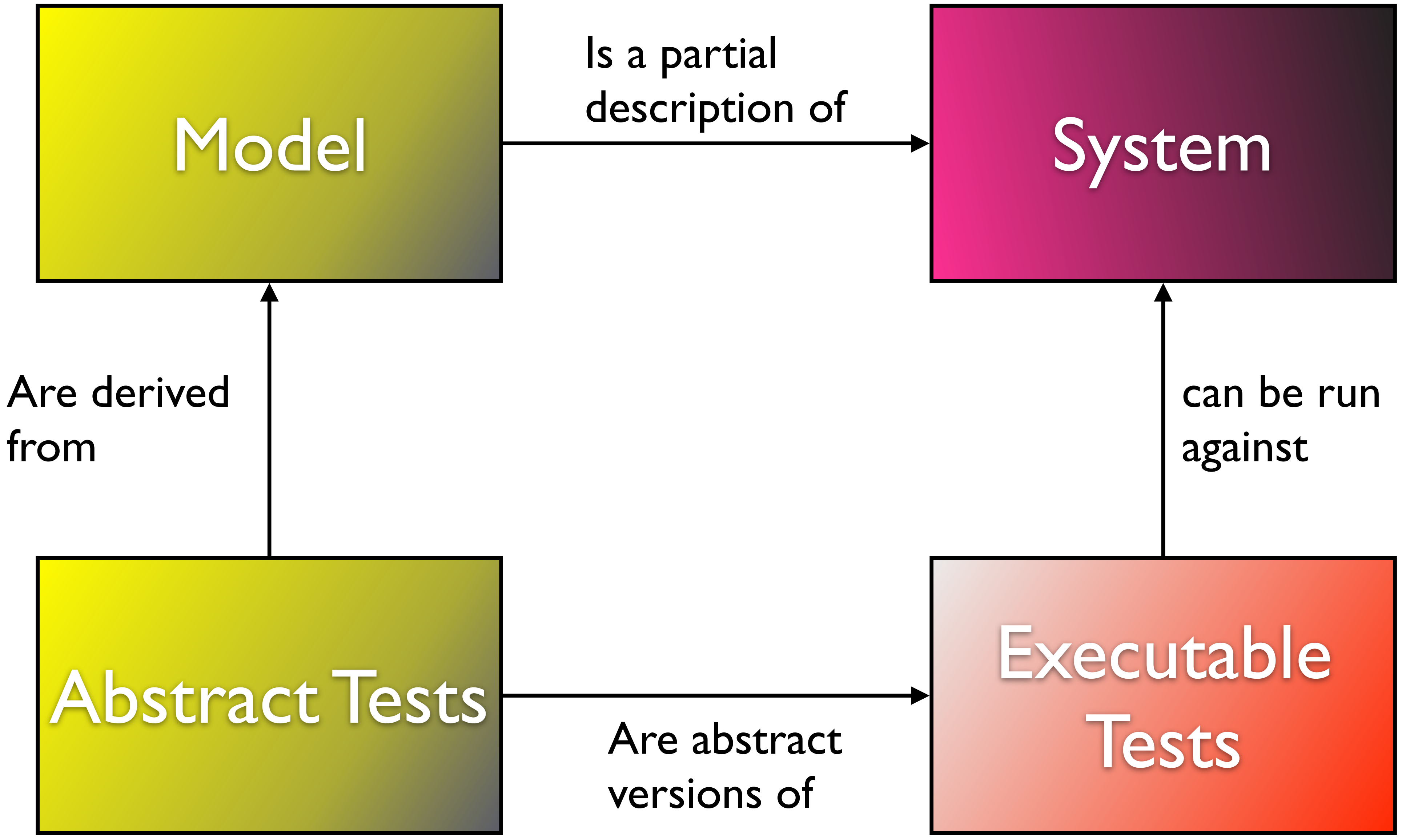
Test Oracles

- Conventional software test oracles – variants
  - Expected result is a concrete value or a sequence thereof – exactly one SUT response is correct
  - Expected result is a logical formula (1st order, trace logic, temporal logic) – several, even infinitely many reactions can be correct, if they fulfil the formula

Test Oracles – Verdicts

- A (conventional) test oracle observes the (timed sequence of) inputs to the SUT and its corresponding outputs and provides a **verdict**
    - **PASS** – the SUT behaviour observed conforms to the expected behaviour
    - **FAIL** – the SUT behaviour observes violates one or more requirements
    - **INCONCLUSIVE** – the test execution could not check the test objectives originally planned

MBT-Paradigm



Testing and Formal Methods

Testing and Formal Methods

- In the context of safety-critical systems
  - Testing is one of the essential means to achieve certification credit
  - It has to be justified that tests have sufficient
    - Requirements coverage
    - Structural coverage
    - Strength to uncover errors

Testing and Formal Methods

- Formal methods are used to
  - devise test strategies with guaranteed test strength
  - prove that a strategy has a certain strength

Fault Models

- A **fault model** consists of
  - Reference model (or other specification)
  - Conformance relation to be fulfilled between reference model and system under test (SUT)
  - Fault domain = collection of models representing the potential (conforming or non-conforming behaviour)

$$\mathscr{F} = (M, \leq, \mathscr{D})$$

Complete Test Suites

- A test suite is **complete** with respect to a given fault model if and only if
  - Every conforming SUT passes all test cases
  - Every non-conforming SUT fails at least one test case, provided that its true behaviour is captured by a model inside the fault domain

Complete Test Suites

- Complete test suite generation methods have been researched since the 1970ies

Chow, T.S.:
Testing software design modeled by finite-state machines.
IEEE Trans. Softw. Eng. 4(3), 178–187 (1978)

- By applying equivalence class generation techniques, complete test suites can be reduced to manageable size, without losing the completeness property

Wen-ling Huang, Jan Peleska:
Complete model-based equivalence class testing for nondeterministic systems.
Formal Asp. Comput. 29(2): 335-364 (2017)

- Experimental results have shown that complete test suites have superior test strength even if the true SUT behaviour lies **outside** the fault domain

Felix Hübner, Wen-ling Huang, Jan Peleska:
Experimental evaluation of a novel equivalence class partition testing strategy.
Software and System Modeling 18(1): 423-443 (2019)

Case Study.
Urban Mobility

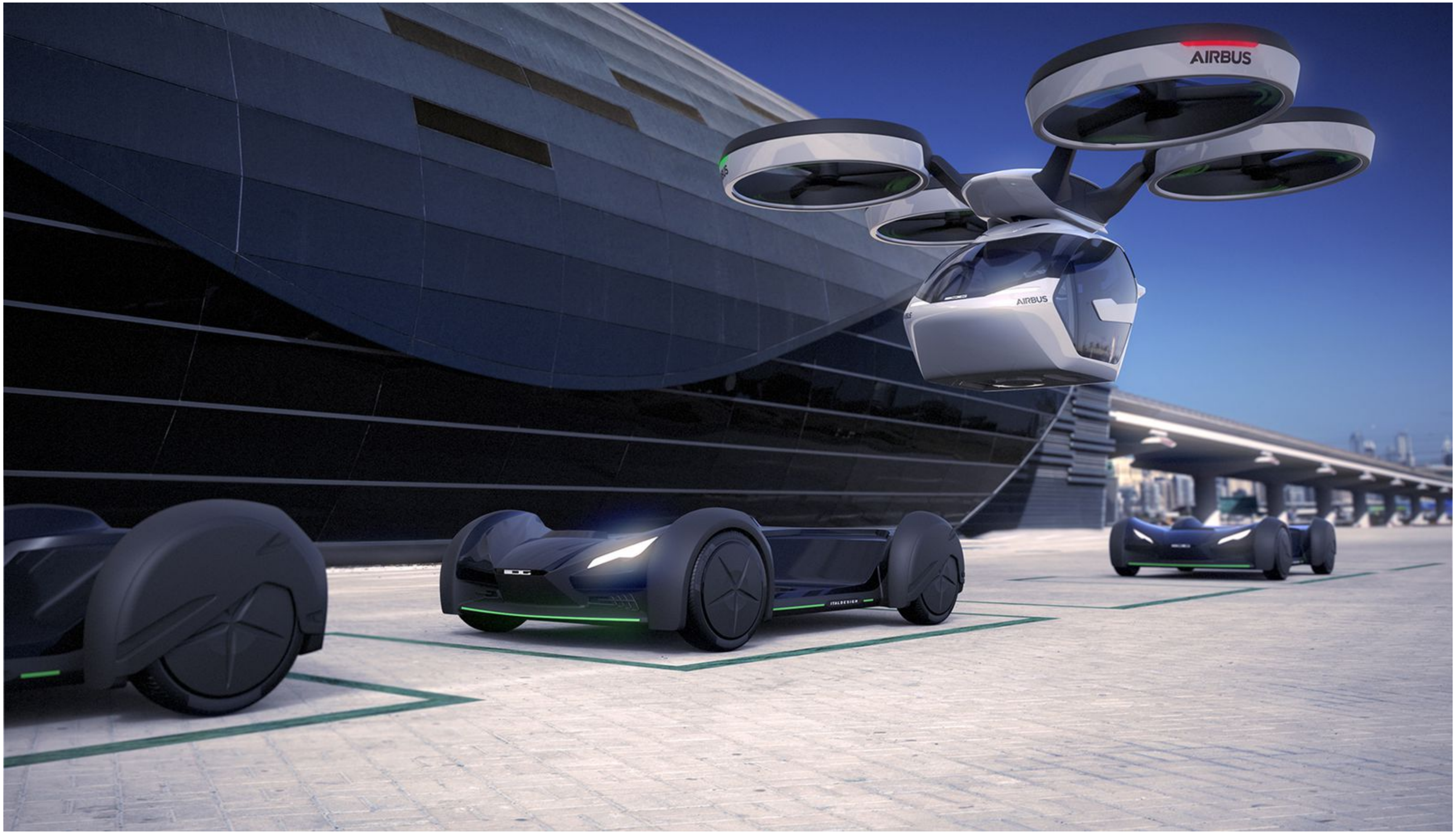
Airbus - italdesign Pop.Up

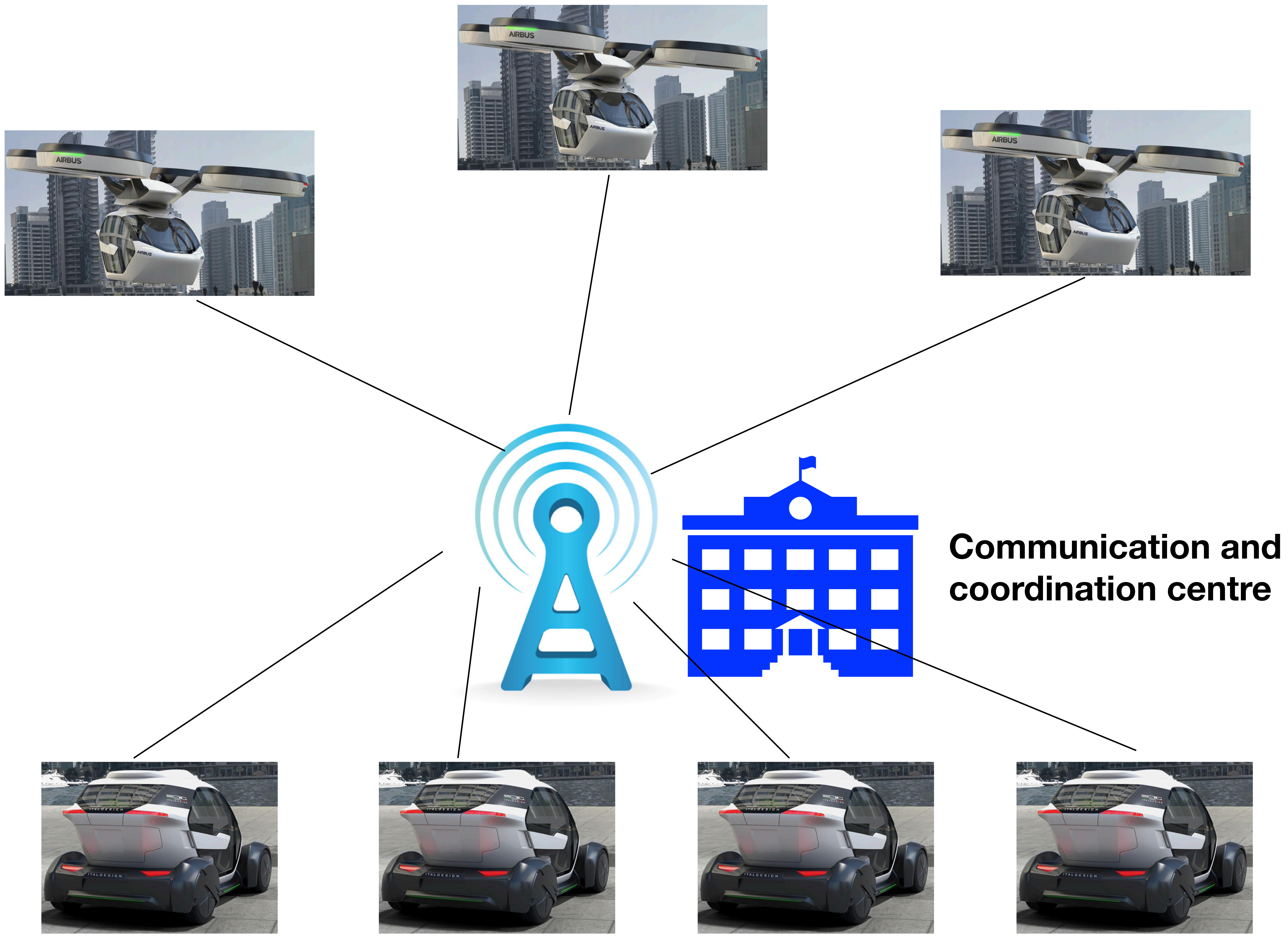
Pop.Up System consists of a **three layers concep**t:

- an **Artificial Intelligence platform** that, based on its user knowledge, manages the travel complexity offering alternative usage scenarios and assuring a seamless travel experience;
- a vehicle shaped as **a passenger capsule** designed **to be coupled with** two different and independent electric propelled modules, **the ground module and the air module**. Other public means of transportation (e.g. trains or hyperloops) could also integrate the Pop.Up capsule;
- an **interface module** that dialogues with users in a fully virtual environment.

Source: https://www.italdesign.it/project/popup/

AIRBUS
AIRBUS
ITALDESIGN
ITALDESIGN
ITALDESIGN


AIRBUS
AIRBUS
ITALDESIGN


AIRBUS
ITALDESIGN
Communication and coordination centre

Autonomous mobile systems – certification-related challenges with impact on testing

Emergent behaviour

- Control centre and drones **cooperate** to optimise the transportation of waiting passenger capsules to their destination

- This also applies to cooperating walking/driving/flying robots

- The resulting behaviour depends on
  - the actual configuration
  - the contract negotiations between participants
  - the learned behaviour of each participant

Emergent behaviour

- Autonomous cars have a slightly less complex emergent behaviour
  - They only cooperate in so far as to avoid collisions
  - This does not require contract negotiations
- Consider, however, trucks driving in line using **electronic towbar (= platooning)** technology
  - In the future, contracts negotiations may become necessary
  - For example, leading truck should be the fastest

Platooning Trucks

Here, all members have the same type – no contract negotiations required

Evolution of behaviour

- Autonomous systems may change their behaviour over time, due to
  - Dynamic configuration changes
    - **Example.** A new drone arrives to speed up the transportation of waiting passenger capsules
  - Learning how to optimise behaviours
    - **Example.** A drone learns how to optimise the flight trajectory when approaching a specific geographical point where a passenger capsule should be picked up

Learning by Local Experience vs. Learning from Global Information

- Global information from
  - Other sub-systems of the mission
  - Internet instructions

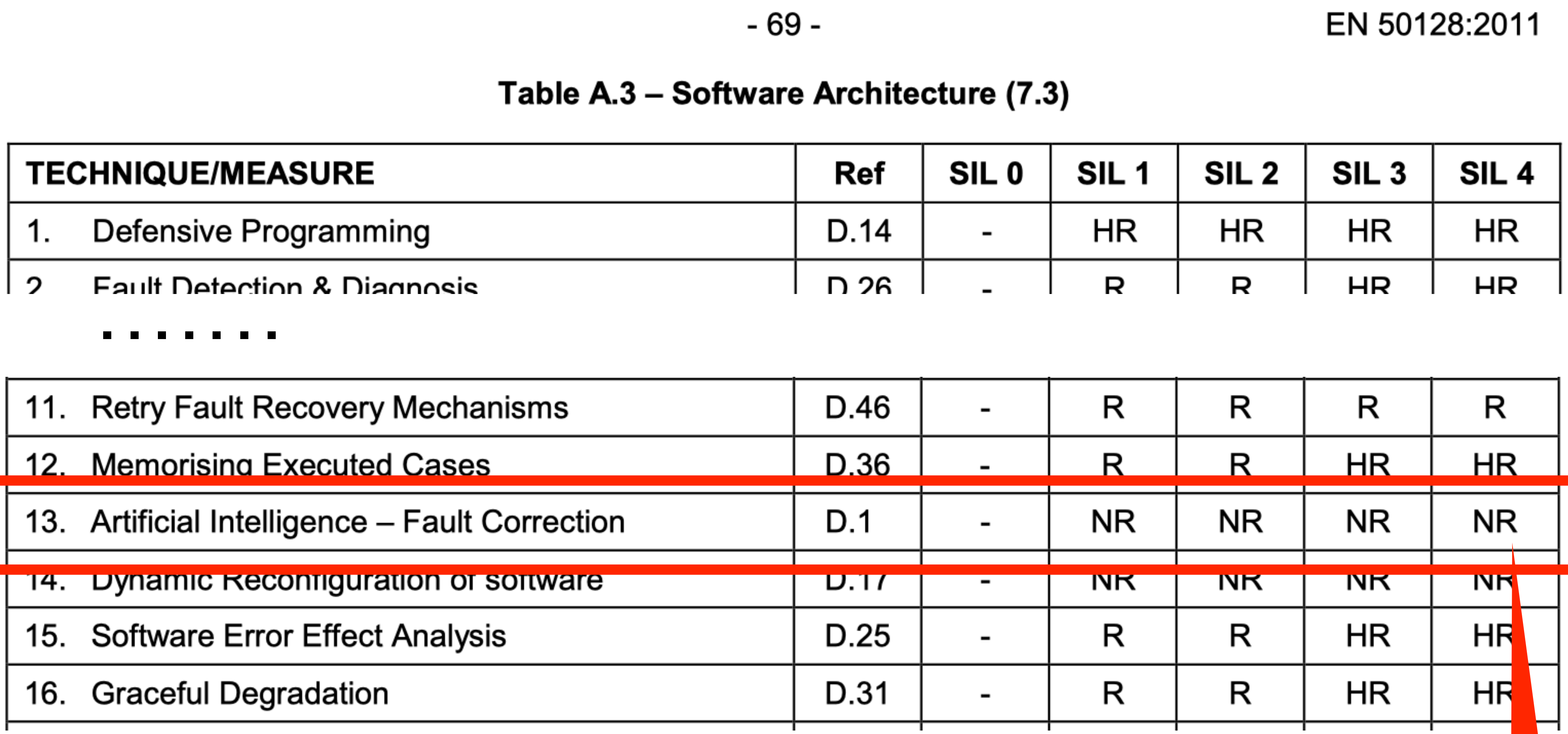
Combined Safety&Security Analysis

- Autonomous collaborative cyber-physical systems depend so strongly on distributed communication technologies that safe operation can only be guaranteed if attacks by malicious agents can be repelled

Certification Challenges

- Safe behaviour of a cyber-physical system depends on mutual guarantees between all sub-systems involved
- Safe behaviour depends on the specific mission and its configuration
- System behaviour may change during operation
- Not all emergent behaviours can be anticipated
- Current standards are not prepared for the application of Artificial Intelligence in safety-critical systems

Current standards are not prepared for the application of Artificial Intelligence in safety-critical systems



- 69 - EN 50128:2011

Table A.3 – Software Architecture (7.3)

TECHNIQUE/MEASURE	Ref	SIL 0	SIL 1	SIL 2	SIL 3	SIL 4
1. Defensive Programming	D.14	-	HR	HR	HR	HR
2. Fault Detection & Diagnosis	D.26	-	R	R	HR	HR
....						
11. Retry Fault Recovery Mechanisms	D.46	-	R	R	R	R
12. Memorising Executed Cases	D.36	-	R	R	HR	HR
13. Artificial Intelligence – Fault Correction	D.1	-	NR	NR	NR	NR
14. Dynamic Reconfiguration of software	D.17	-	NR	NR	NR	NR
15. Software Error Effect Analysis	D.25	-	R	R	HR	HR
16. Graceful Degradation	D.31	-	R	R	HR	HR

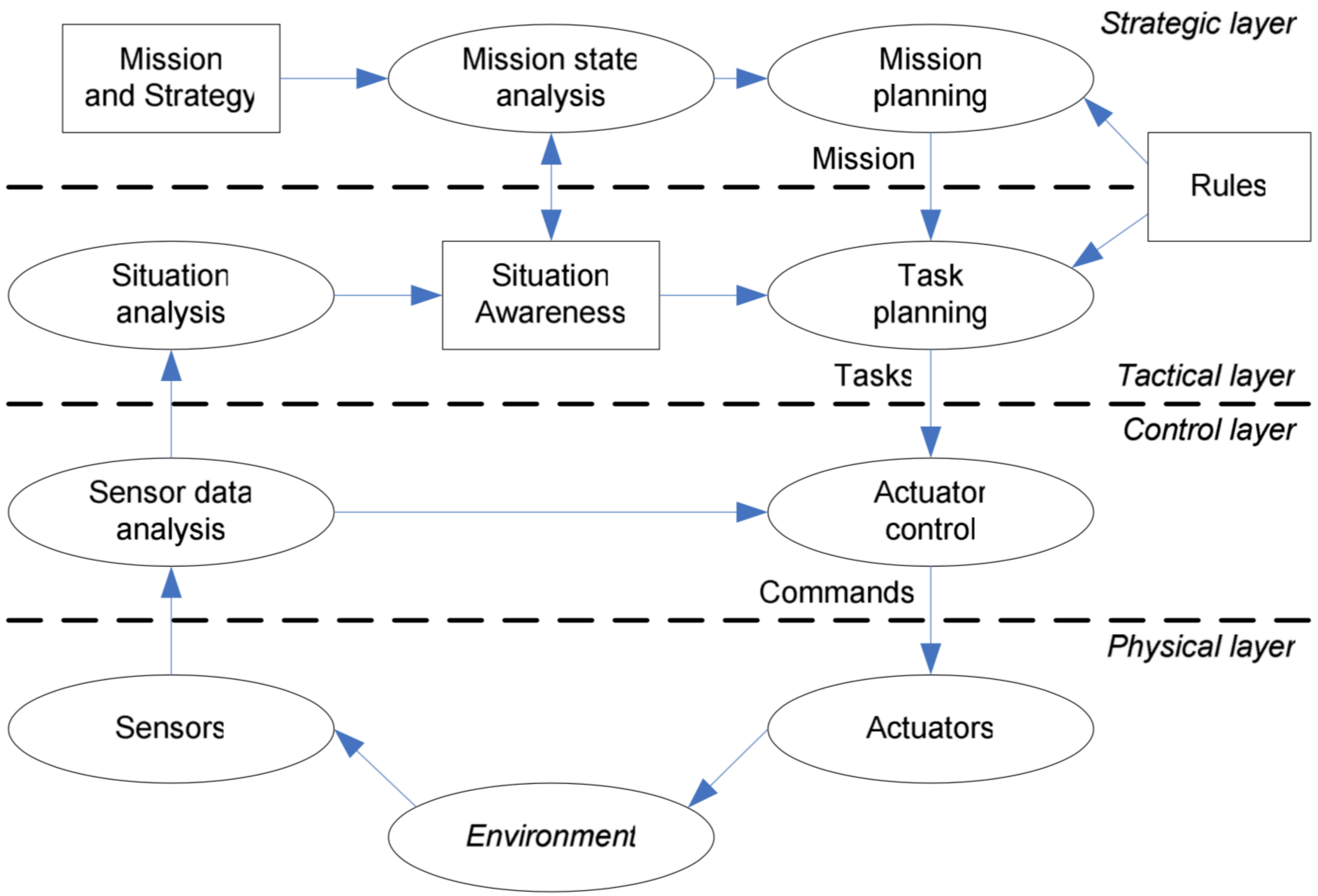
NR = NOT RECOMMENDED

Specific test-related challenges

Test Strategies – Challenges

- The effectiveness of test strategies depends (among other things) on the development techniques used
- But autonomous systems development uses novel techniques such as
  - Strategic mission planning
  - Situation awareness and dynamic risk assessment
- **As a consequence, proven strategies for testing conventional systems are not necessarily effective for autonomous systems**

Generic Architecture for Autonomous Systems



Wardziński A. (2008) Safety Assurance Strategies for Autonomous Vehicles. In: Harrison M.D., Sujan MA. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2008. Lecture Notes in Computer Science, vol 5219. Springer, Berlin, Heidelberg

Test Case Generation – Challenges

- **Too many test cases** required to create them manually
- **No complete reference model** available for MBT, so model-based test generation does not necessarily lead to all relevant test cases
- Test models need comprehensive **environment representation**
- Some validation tests may need to be designed/executed during runtime – **runtime acceptance testing**:
    - Validation depends on contracts between configuration of constituent systems
    - Validation depends on mission details specified for the actual task at hand

Test Oracles – Challenges

- For autonomous systems, test oracles need to cope with

  1. Behaviour that is under-specified

  2. Behaviour that is only acceptable if its risk level is acceptable

  3. Behaviour that is not deterministic, but follows some (sometimes even unknown) probability distribution or probabilistic reference model

Test Oracles – Challenges

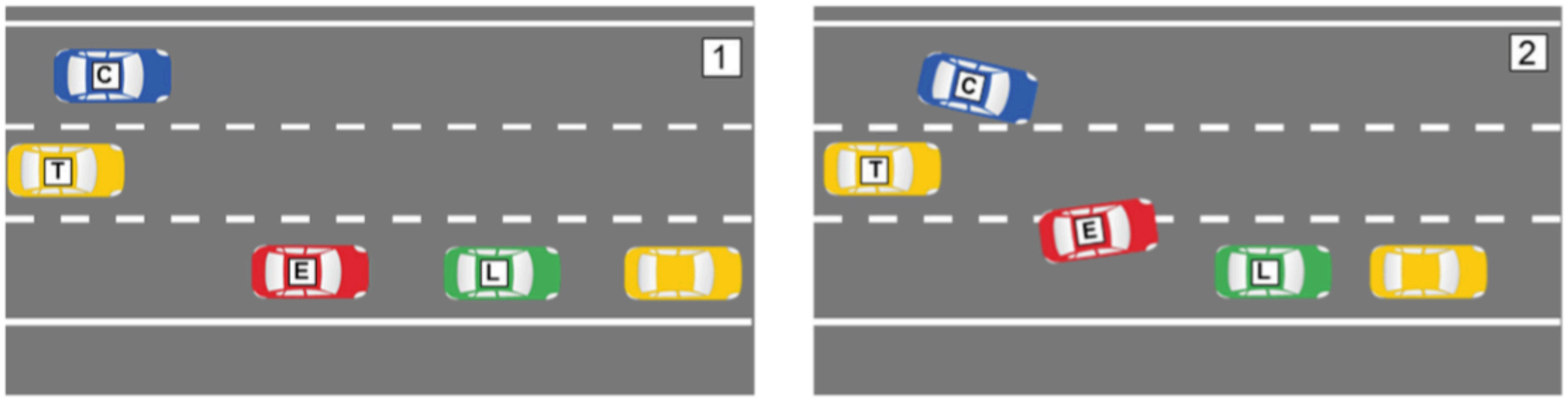
- **Example 1. Under-specified behaviour**
  - A robot arm handing a drinking cup to a disabled patient can solve this mission by infinitely many trajectories for the cup
  - This type of problem has led to layered architectures in robotics control software
    - Strategic Layer for defining and controlling the high-level mission (“lift cup from table to patient’s mouth”)
    - Control layer for executing concrete movements in space-time (“find trajectory for cup to reach patient’s mouth without collisions with any obstacles”)

Test Oracles – Challenges

- **Example 2. Behaviour that is only acceptable if its risk level is acceptable**

  - An autonomous car avoiding collision with another car during conflicting lane changes by accelerating during the lane change – instead of aborting the lane change

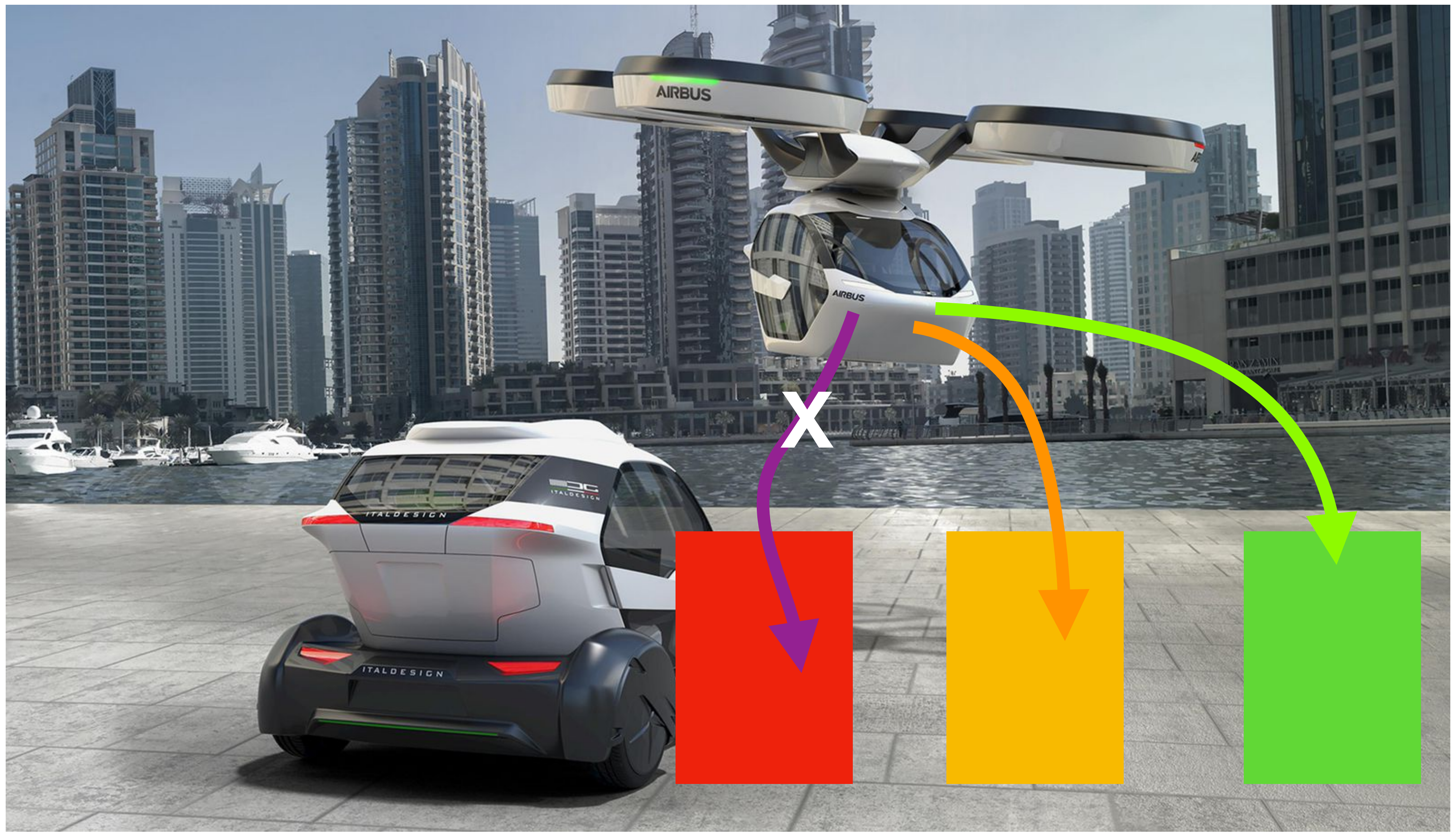
  - Test fails due to intolerable risk taken by autonomous car E



Hardi Hungar: Scenario-Based Validation of Automated Driving Systems. ISoLA (3) 2018: 449-460

Test Oracles – Challenges

- **Example 3. Behaviour that is not deterministic, but follows some probabilistic reference model**
  - A drone that chooses landing trajectories that are distributed around an optimal trajectory with acceptable variance



Test Oracles – Challenges

- Test oracles for autonomous systems will become a combination of
  - conventional oracles for control systems and
  - statistical testing of hypotheses
- For the statistical testing, the number of test executions (for the same test case) needs to be much higher than for deterministic or nondeterministic systems without statistical distribution requirements

Test Oracles

- For autonomous safety-critical systems (as in our case study) test oracles have extended verdicts
  - **(definitely) FAIL** – violation of a non-probabilistic requirement – the mission objectives could not be achieved
  - **FAIL due to unacceptable risk level** – though the mission objectives could be achieved
  - **PASS with acceptable risk level** – the mission objectives could be achieved
  - **(definitely) PASS** – conformance to a non-probabilistic requirement
  - **INCONCLUSIVE**

PLEASE ATTEND THE 2 SESSIONS ON NOVEL TESTING SOLUTIONS TO BE PRESENTED ON THURSDAY!