Fuzz-testing with AI methods

Increasing the robustness of software components through automated testing

October, 5th 2021 | Test4Rail | PD Dr.-Ing. habil. Lars Schnieder

Engineering und Software-Entwicklung

Agenda

Engineering und Software-Entwicklung

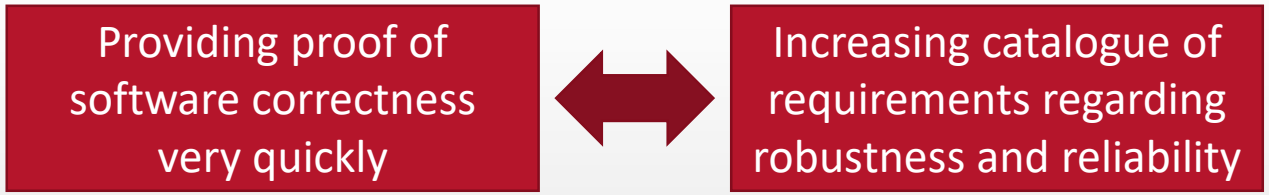
1. Motivation and problem definition
2. Structure of the test method
3. Mutation operators for sequence diagrams
4. Evaluation and results
5. Summary and outlook

Motivation and problem definition

Engineering und Software-Entwicklung

- **Functional Safety** requires software components to be verified, validated and assessed
- In addition, **Cybersecurity** is getting more important for critical infrastructures
  - Networked systems open up new attack vectors

- **The challenge:**

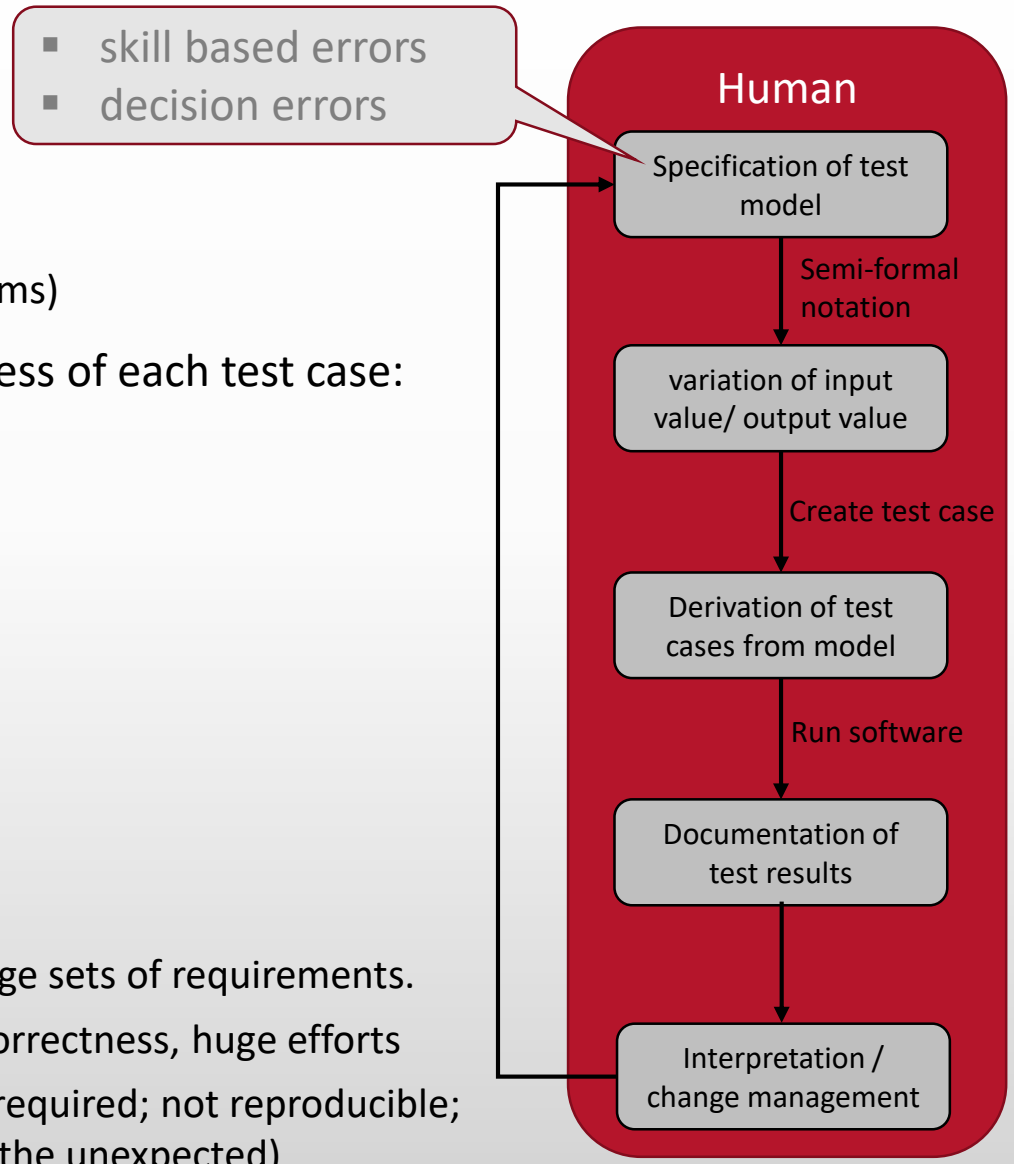


- **Goals:**
  - **Hardening** of systems with regard to cybersecurity
  - **Higher efficiency** of text execution through partial automation
  - Increased **test coverage** by using machine learning

Conventional Testing

Methodical Approach

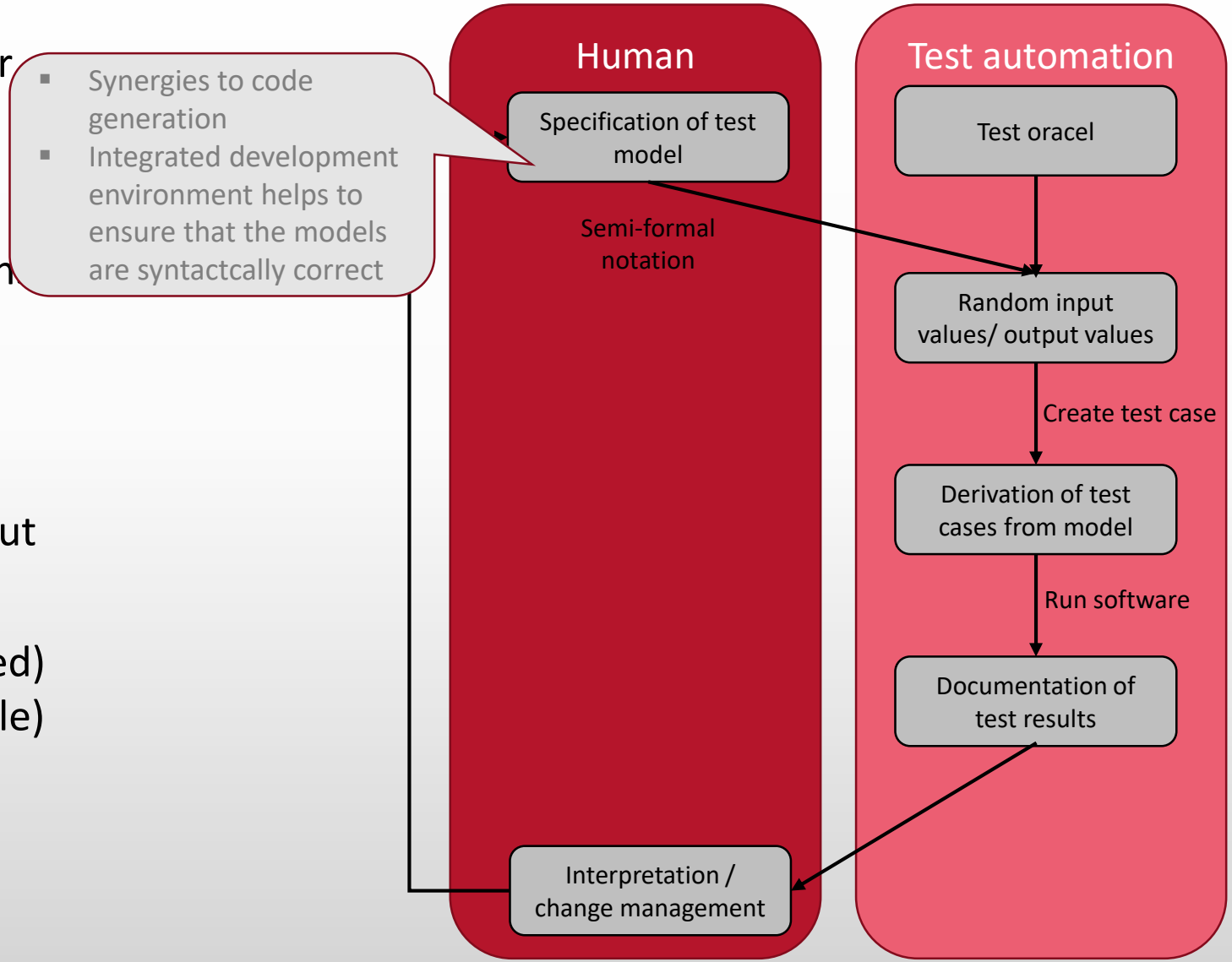
- Test cases can be captured in different notations:
  - Definition of test cases in natural language
  - Definition of test scripts in programming language
  - Definition of test cases in sequence models (e.g. UML behavior diagrams)
- Pattern can be used in test case definition to support completeness of each test case:
  - Input values
  - Test case category (e.g. fault insertion test, requirements based tests)
  - reference to requirements ID (traceability)
  - Test object (e.g. configuration)
  - Start condition
  - Test steps (what to be done by the tester)
  - Expected output values
- Disadvantage of conventional approaches of test execution:
  - Test cases are derived individually -> huge efforts for increasingly large sets of requirements.
  - Test cases are capture manually -> prone to failures, no proof of correctness, huge efforts
  - The tester is the test oracle -> lot of experience (senior staff) required; not reproducible; biased (tester does not expect the unexpected)



Fuzzing

Methodical Approach

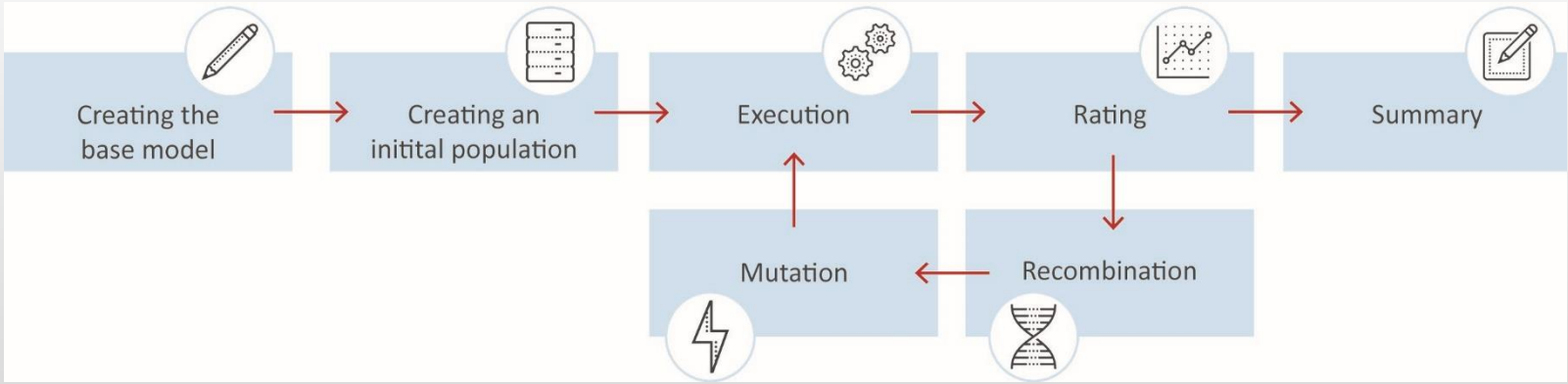
- Fuzzing is an automated technique for test case generation
- Widely used in IT security
- Generates unusual operating situation through random data
- Input interfaces are supplied with random data
- Often reveals faulty plausibility of input data
- The test object is observed (automated) to detect unusual reactions (test oracle)



Application of Artificial Intelligence (AI)

Use of Genetic Algorithms

- Algorithms that emulate evolution
  - **Creation of the base model** = All existing (manually created) test cases
  - **Creation of the initial population** = (initial) set of test cases
  - **Rating (evaluation of fitness function)** = measurement of the quality of the considered test cases
  - **Recombination** = Combination of the best test cases to next generation
  - **Mutation** = Random insertion of changes in test cases
  - **Summary** = Documentation of test results

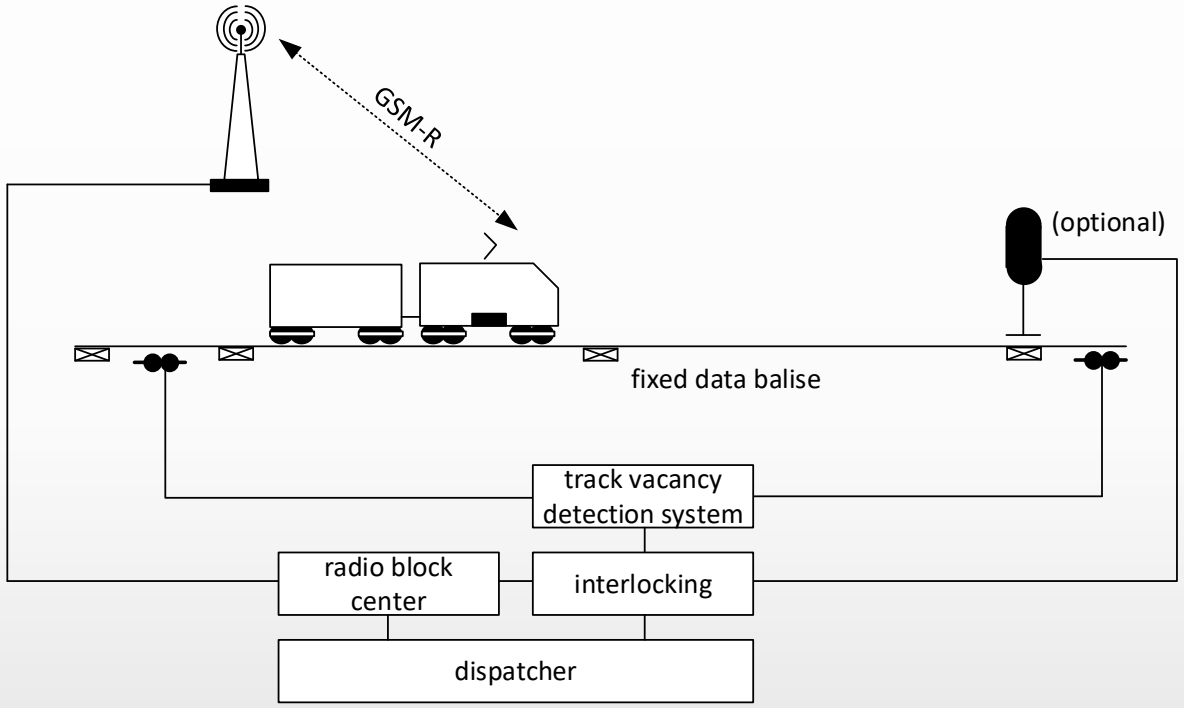


Application of Artificial Intelligence (AI)

System Under Test: Component of a Radio Block Center

Engineering und Software-Entwicklung

- Component has already gone through 10+ manual verifications (with 100% test coverage)
- 700+ manually generated test cases exist
- No new/unknown errors found based on repeated manual verification
- High complexity of system component
  - Communication interfaces to many different entities
    - many vehicles
    - several neighbouring RBCs
  - Concurrrent execution of different message timers
  - Overall exponential growth system states

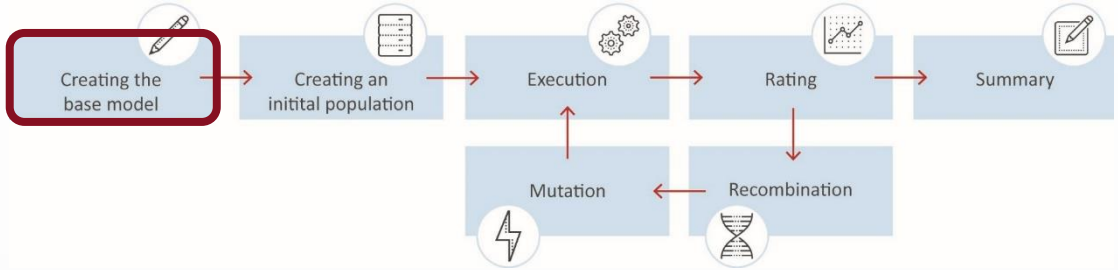


System architecture of ETCS Level 2

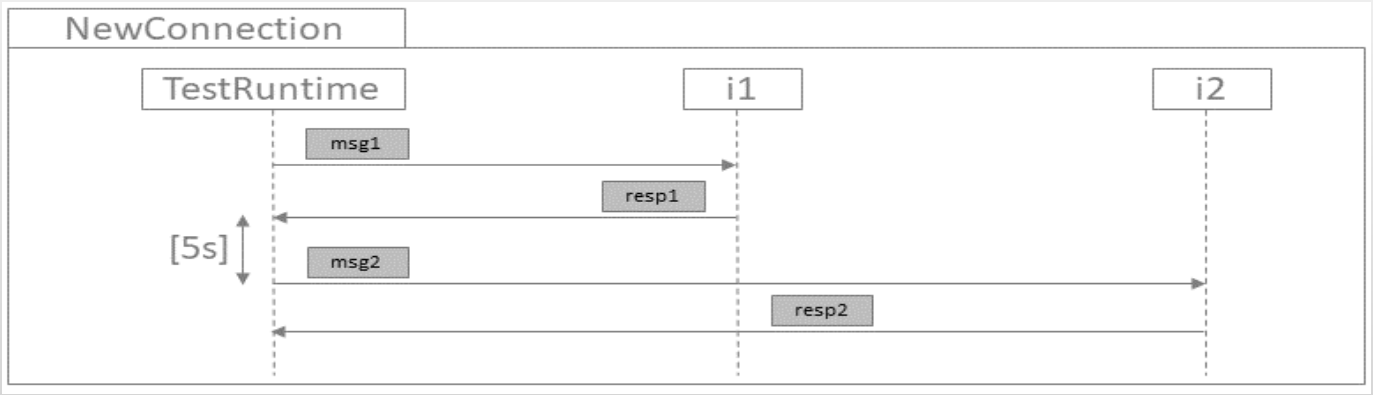
Creation of the base model

Semiformal test sequences as a basis

Engineering und Software-Entwicklung

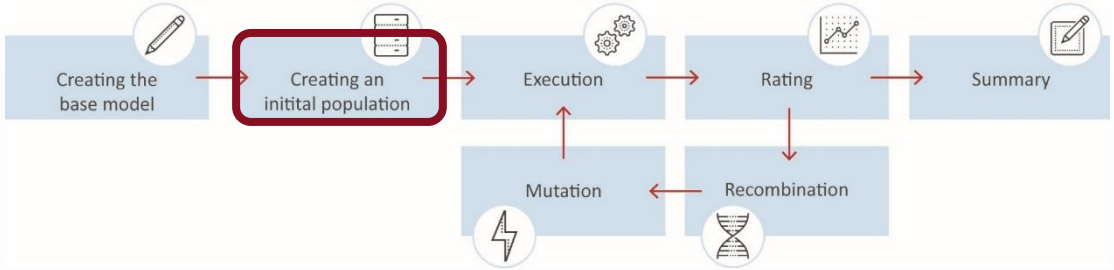


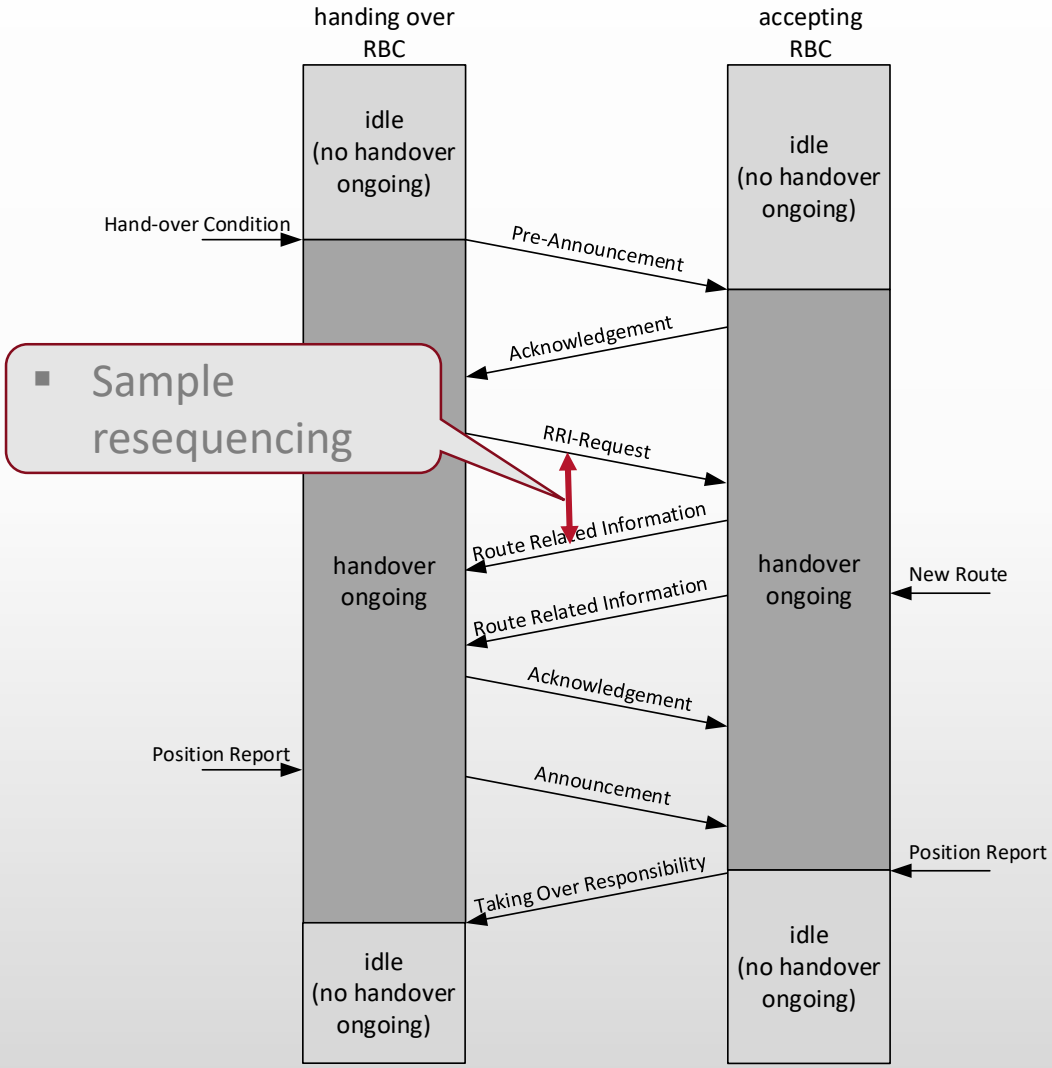
- Test cases are represented as **sequence diagrams**
- The following communication partners exist:
  - **TestRuntime:** runtime environment of the test cases as dummy for the environment of the test object.
  - **Test Object:** All interfaces of the test object to be tested
- Messages:
  - From TestRuntime to an interface: Stimulus, i.e. input data
  - From interface to TestRuntime: expected reaction of the test object
- Dependencies: Temporal conditions (e.g., wait times).



Creation of the base model

(initial) set of test cases





- Implemented test cases (later: mutation operators) according to failure taxonomy of EN 50159 for communication:
  - **Omission:** removal of an intended message
  - **Resequencing:** swapping the sequence
  - **Corruption:** changing the data within a message
  - **Insertion:** Adding an additional (unexpected) message
  - **Repetition:** sending the previous message again
  - **Delay:** change of waiting time for following message

Guide word (according to IEC 61882:2016)	Explanation
No	Deviation from intended behavior
More	Quantitative growth, too much
Less	Quantitative decrease, too little
As well as	Additional events to the intended behavior
Part of	intended behavior only incompletely achieved
Reverse	Opposite of the intended behavior
. . .	. . .

Creation of the base model

(initial) set of test cases

Engineering und Software-Entwicklung

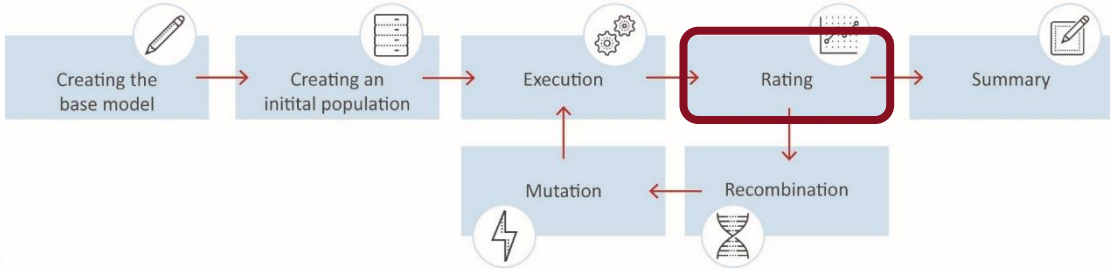
- Derivation of safety mechanisms according to UNISIG

Function	ETCS mode	Failure mode	Failure effects			failure cause	ETCS external protection / mitigation barriers	severity	Internal barriers
			local	intermediate	Final				
Taking Over Responsibility (Accepting RBC informs Handing Over RBC that it has taken Over)	Any	CORRUPTION (Incorrect Take Over Information transmitted to handing over RBC)	Handing Over RBC commands a non intended ETCS onboard (wrong ETCS onboard ID) to terminate the communication, While maintaining the communication to the intended ETCS onboard.	Non intended ETCS on-board is not connected to an RBC anymore. Handing Over RBC receives a position report from intended ETCS on-board that the min safe rear end has crossed the border, and sends a disconnection order to this ETCS on-board	No MA revocations can be transmitted to the non intended on-board	▪ RBC Failure ▪ RBC Communication failure		▪ catastrophic	Usage of the EuroRadio MAC for revelation of corruption in communication. Message is discarded as safety reaction

Unisig Subset 078 - Failure Modes and Effects Analysis for the Interface to/from an Adjacent RBC - in Application Level 2. Version 2.4.0 (2009-04-07)

Rating

Evaluation of fitness function

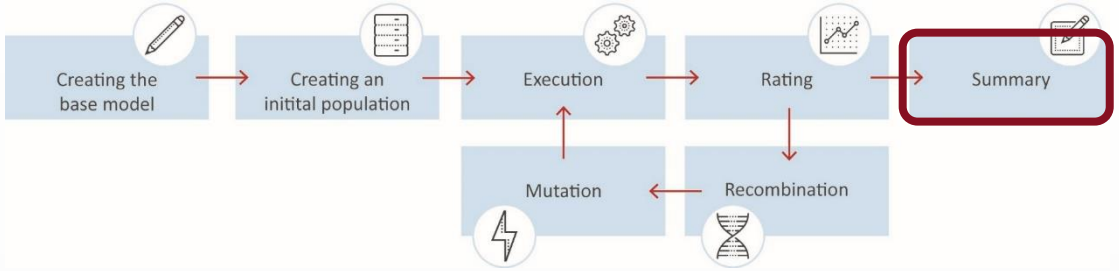


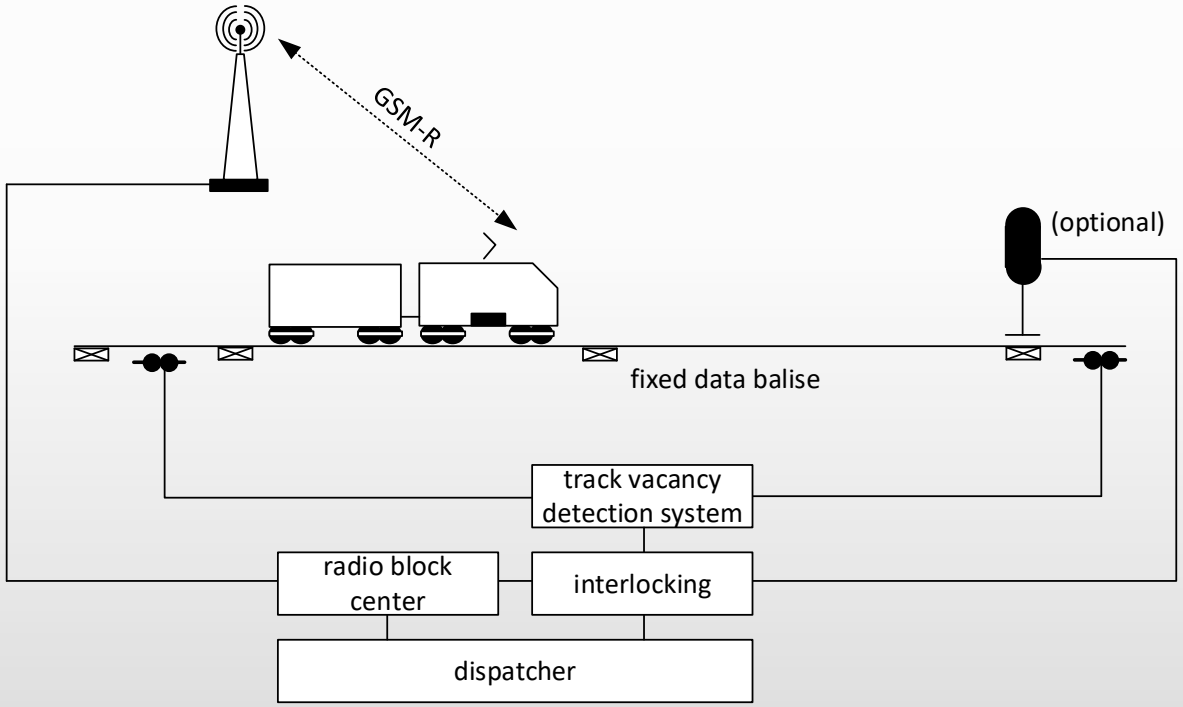
- The evaluation of the quality of an executed test case is the basis for its existence in the next generation
- Factors for goodness (sorted according to importance):
  - **Triggered untreated crashes**: Does the test object crash without handling the fault?
  - **New detected output behavior**:
    - Diagnostic messages example: All sequences of diagnostic outputs issued by the test object are saved
    - If a new sequence is recorded, new behavior has been detected.
  - **New code coverage**:
    - The total lines traversed in all generations are highlighted
    - If a test case runs through a line that has never been run before, new behavior has been detected
  - **Absolute code coverage**:
    - The number of **lines of code** passed is measured
    - A higher number of lines traversed increases the number of bugs found
- A weighted sum of the factors results in the evaluation of a test case

Summary

Documentation of trest results

- After AI-inspection:
  - No untreated crashes found
  - New detected output behavior (unexpected systems reactions found which lead to a safe state -> impact on availability)
  - Code coverage (new/absolute -> see next slide)
- 5h execution time for full test coverage (quality gain with limited amount of effort)





System architecture of ETCS Level 2

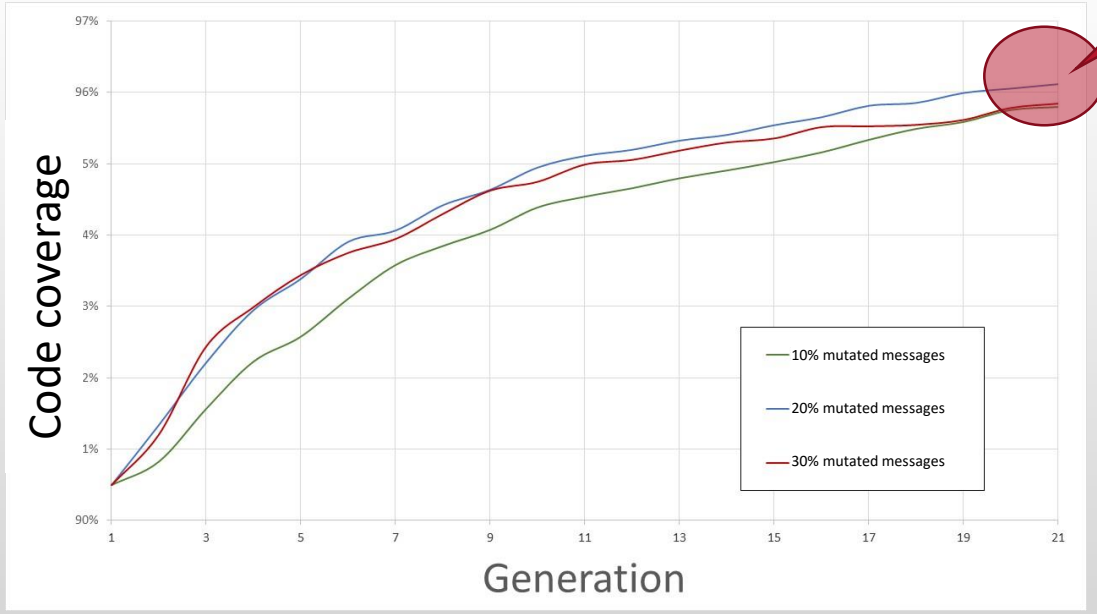
Summary

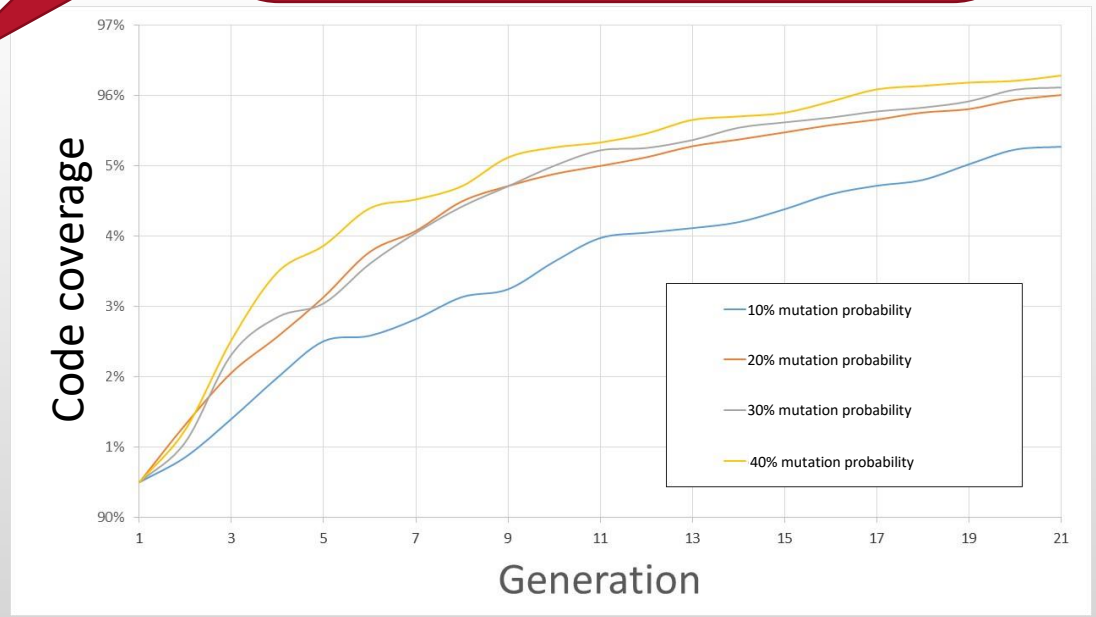
Evaluation of test cases

Engineering und Software-Entwicklung

- Measurement of the influence of different parameters on the quality of the algorithm
- Measurement of code coverage (for simplified representation)
- Basis: subset of existing test cases
- Mutation probability and proportion of mutated mess[illegible] hav

- Fuzz testing is an additional criterion to increase system robustness.
- Fuzz testing is **necessary,** but **not sufficient** (still manual testing required, but on a lower level)





Summary and outlook

Engineering und Software-Entwicklung

Summary

- Basic suitability of genetic algorithms for increasing test coverage demonstrated
- Choice of parameters is a crucial challenge
- Automated generation of new test sequences possible and reasonable

Outlook

- Improvement of the fitness function
- Extension by assurances for improved target-actual comparison
- Addition of further mutation operators for more targeted generation of new sequences

Get in contact!

PD Dr.-Ing. habil. Lars Schnieder

Chief Operation Officer

Mail: lars.schnieder@ese.de

Phone: +49 (0)170 7712863

Engineering und Software-Entwicklung