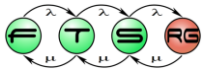
Testing the robustness and safety of context-aware autonomous behaviour

Software Verification Techniques course

István Majzik
Budapest University of Technology and Economics
Dept. of Measurement and Information Systems



Outline

- Motivations and applications
- The testing concept
  - Context modelling
  - Requirements modelling
  - Test strategies
  - Test evaluation
- Testing tool-chain (testing framework)
- Case studies
- Summary

Introduction

- R3-COP and R5-COP projects
  - Development of methods and tools for the efficient verification and testing of dependable and safety-critical systems
- Focus of these projects is testing …
  - Autonomous systems
    - that make decisions to execute missions without direct human control
  - Context-aware systems
    - that use perceived context information to provide relevant services

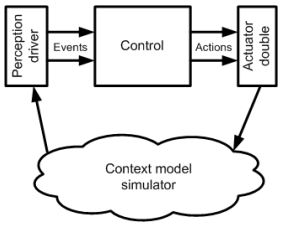
Objectives and applications

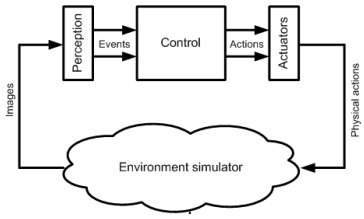
- Systems to be tested in the project
  - Autonomous robots: household and manufacturing robots
  - Autonomous vehicles: LGV, UAV, RUAV
- Objectives of testing
  - Robustness and safety: safe system behaviour in the presence of stressful environmental conditions
  - Context-awareness: dependency of the system behaviour on the evolving state of the complex environment (context)
- Typical safety requirements to be addressed
  - "*In case the robot is in close proximity to living beings it shall send sound or voice alerts*"
  - "*When an obstacle gets to the dangerous area then the vehicle shall stop*"

  → Context-related conditions (initial, interim and final), and corresponding sequence of actions

Test purpose and test environment

- Black-box testing of the behaviour





- Test execution:
  1. Configuration of the (dynamic) context
  2. Starting the robot to execute its mission
  3. Observing context-dependent actions
- Test environment:
  - Simulator
  - Real (physical) environment

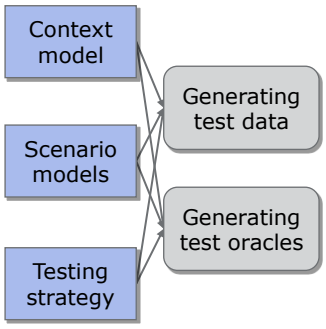
Challenges and solutions

Typical challenges in testing:	Proposed solutions:
Informal requirements specification	Precise requirements specification
➢ Specification of context-aware behaviour is difficult	➢ Modelling contexts, dynamic events and actions
➢ Adaptivity is an issue	➢ Modelling scenarios
Manual preparation of test cases	Systematic, model based test data generation
➢ Based on experience	➢ Based on context model
➢ Testing behaviour in typical situations	➢ Testing behaviour in stressful context
Imprecise test quality metrics	Model based test coverage metrics
➢ Relation to context	➢ Context based coverage
➢ Relation to requirements	➢ Scenario based coverage

The testing concept

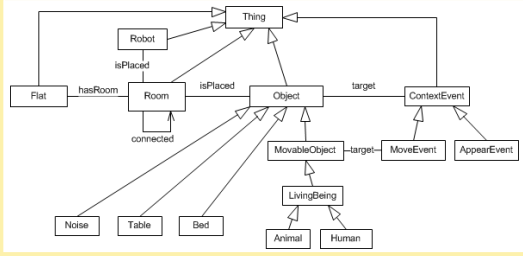
Overview

- Test goals
  - Systematic generation of test data,
    i.e., test contexts that include stressful (unexpected) situations
  - Evaluating the safety of the observed behavior
- What is needed?
  - Description of the environment:
    Context modeling
  - Capturing the test requirements:
    Scenario modeling
  - Systematic generation of test data:
    Testing strategies
- Tools for the tedious steps

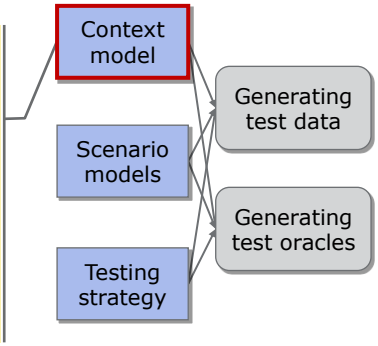


Context modeling

- Domain ontology → Context metamodel
  - Objects with properties, including dynamic objects
  - Relations: concrete or abstract relations
  - Constraints: physical limitations and application-specific constraints
- Action model for control actions

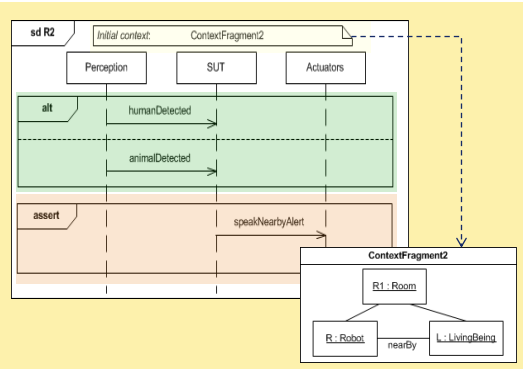


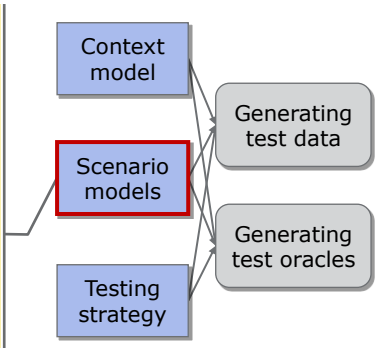
+ Constraints (Object Constraint Language)



Scenario modeling

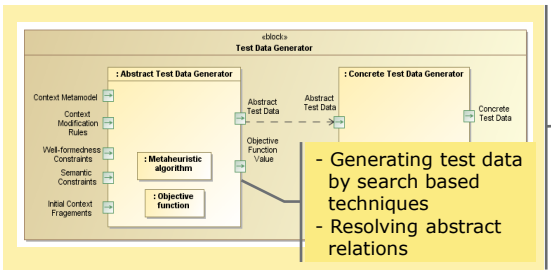
- Safety requirements → Scenario model
  - Initial context fragment: instances of the context metamodel
  - Trigger part: Events (perception) and messages (commands)
  - Assertion part: Expected or forbidden actions and context changes
  - Precise semantics based on MSC and LSC constructs with time

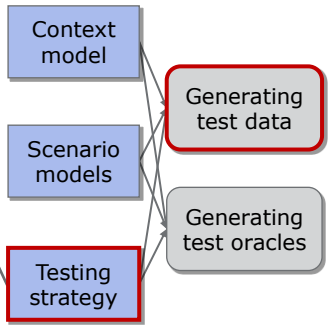




Strategies for testing robustness

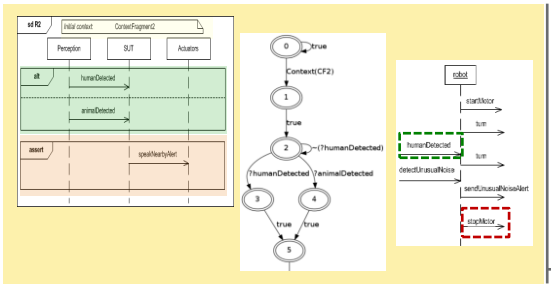
- Unexpected objects → Context coverage
  - Extending the initial context fragment with extra objects
- Complex contexts → Scenario coverage
  - Combining (n-wise) the initial context fragments of scenarios
- Extreme situations → Robustness coverage
  - Mutating initial context fragments: using boundary values of properties, violating application constraints

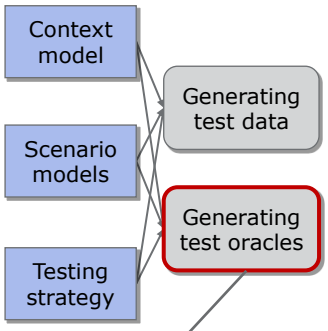




Evaluating test traces

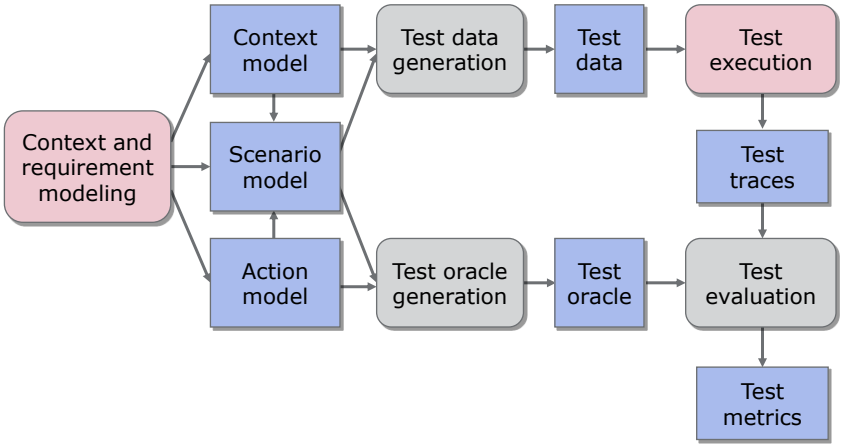
- Evaluating a test trace simultaneously against each requirement scenario, from each relevant step
  - Matching events and context changes (respecting object hierarchy): using efficient graph-based decomposition techniques
  - Automated evaluation of traces: synthesis of observer automata from the scenarios
  - Identifying violated scenarios → metrics





The testing tool-chain

The testing tool-chain



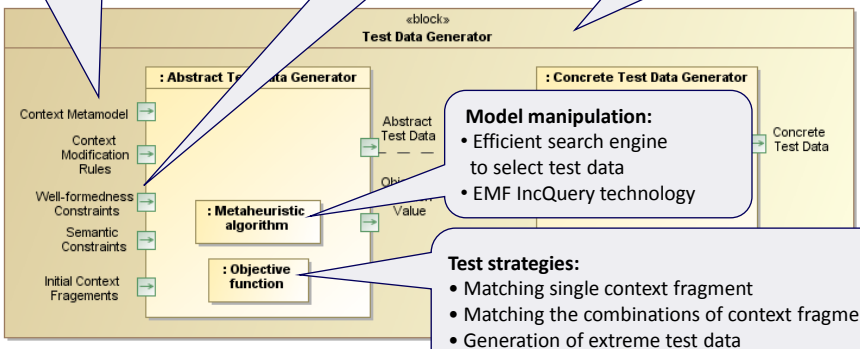
- Lightweight but precise requirement modeling
- Automated tools to support test generation and evaluation

Tool: Test data generation

Input (context) models:
- From Papyrus or RSA
- Mapped internally to EMF models

Constraints:
- Mapped to EMF model patterns
- Matching is checked

Tool framework:
- Eclipse technology
- Prototype implementation

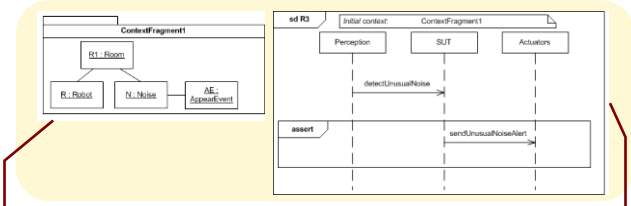


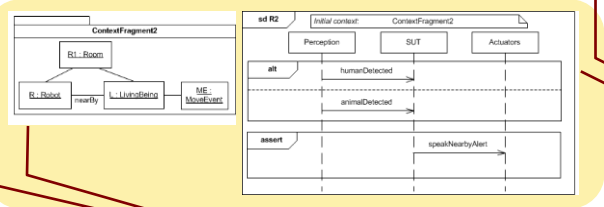
Model manipulation:
- Efficient search engine to select test data
- EMF IncQuery technology

Test strategies:
- Matching single context fragment
- Matching the combinations of context fragments
- Generation of extreme test data

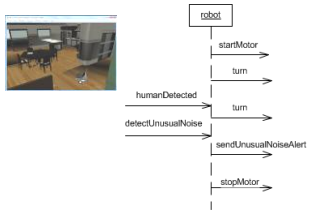
Test generation: Combining initial contexts

Test requirement scenarios:

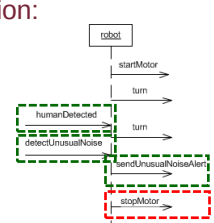




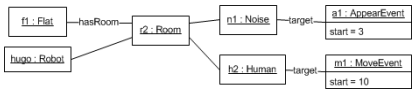
Test trace:



Evaluation:



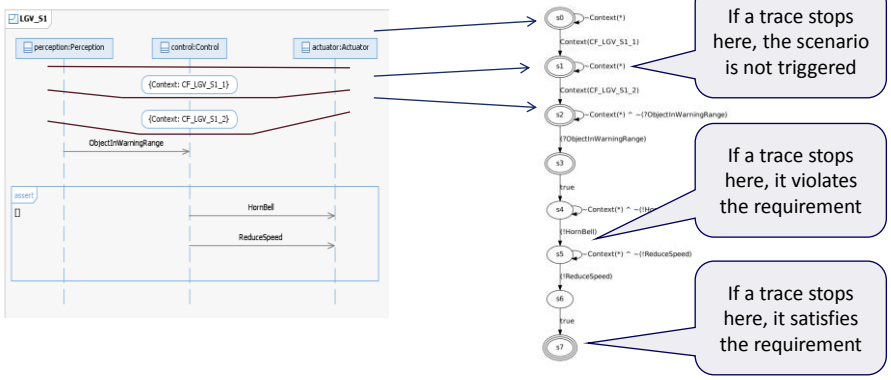
Test data generated:



Test verdict:
R3 passed
R2 failed

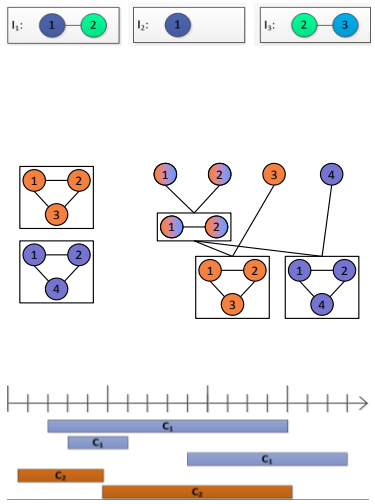
Tool: Test oracle generation

- Goal of test evaluation:
  - Checking which req. scenario was triggered/satisfied/violated
  - Matching of events/actions and context fragments from test traces
- Solution: Building an observer automaton for each scenario



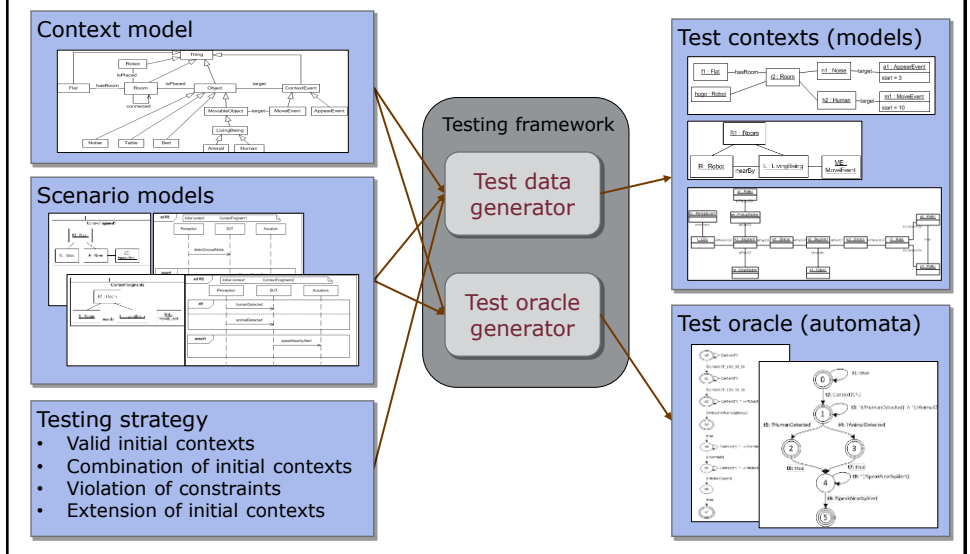
Test oracle: Checking context fragments

- Checking sequences of changing contexts (dynamic objects)
  - Graph based representation and matching of graph sequences
- All requirements (contexts) are checked simultaneously
  - Decomposition of context fragments to utilize common parts that could be checked once
- Context fragments (from requirement scenarios) are checked for matching at each step of the trace
  - Concurrent threads for evaluation of matching contexts

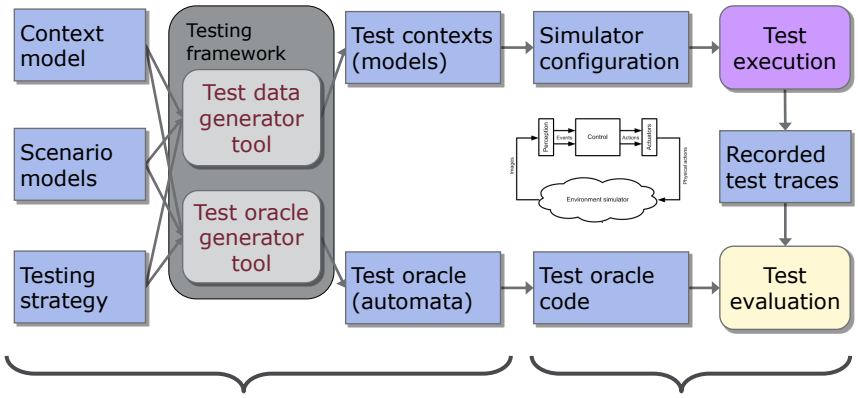


The testing framework in practice

Inputs and outputs of generator tools



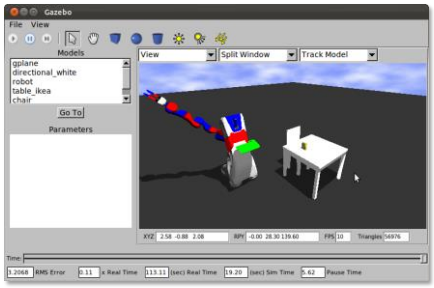
Behaviour testing – Use in demonstrators

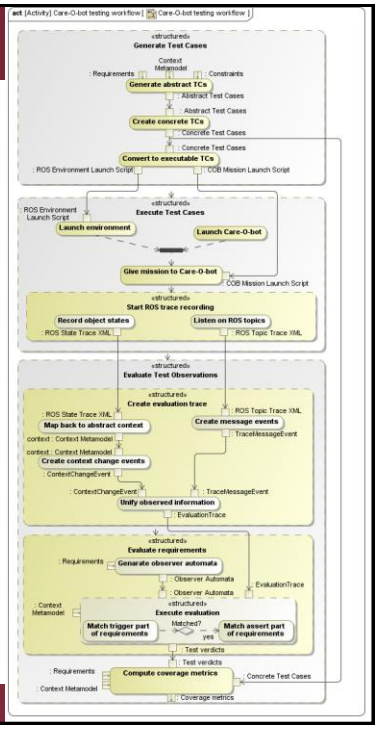


- Demonstrated for LGV (forklift) by generating test context models
- Demonstrated for home robots (Care-O-bot) by interfacing with ROS+Gazebo simulator

Simulator based testing

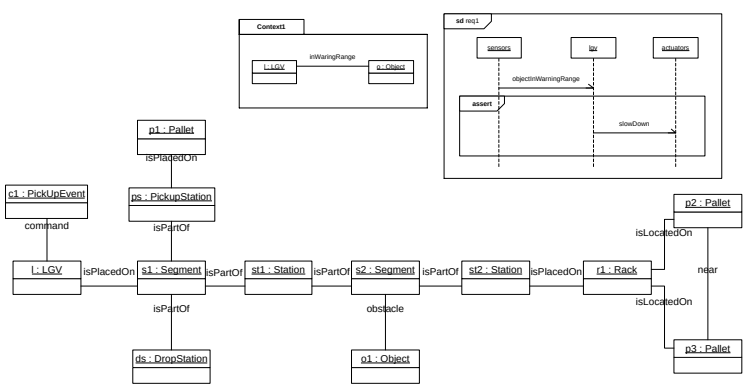
- FHG Care-o-Bot: Simulator based testing
- Interfacing the testing framework with the ROS+Gazebo simulator
  - Mapping the generated test context model to simulator configuration
  - Mapping events and context changes recorded in the simulator to the input of the test oracle





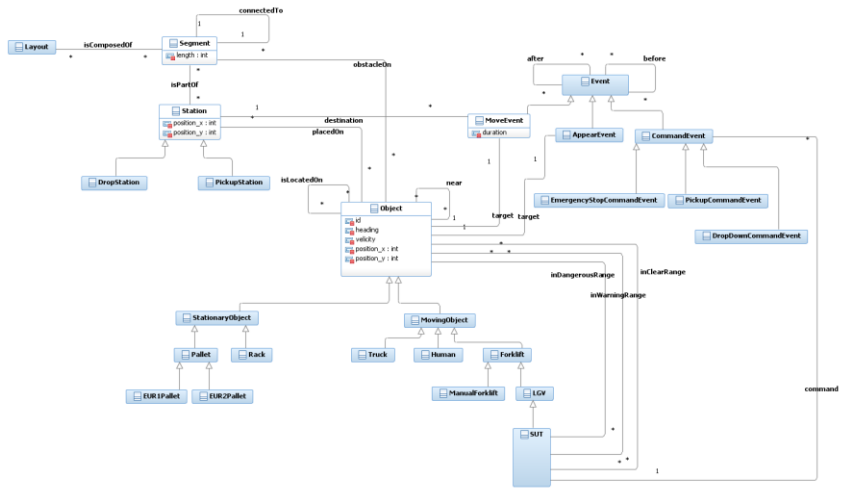
Testing in real environment

- E80 Forklift: Testing collision avoidance
  - Context and requirements modelling
  - Generating abstract test context (object model)
  - Mapping abstract test context to real configuration



Application: Testing an LGV 1/3

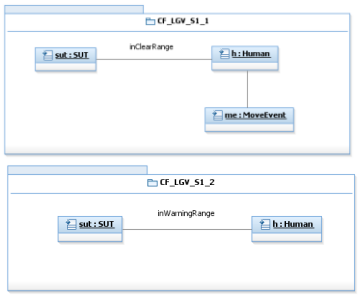
- Context modelling: Capturing environment objects, relations, constraints



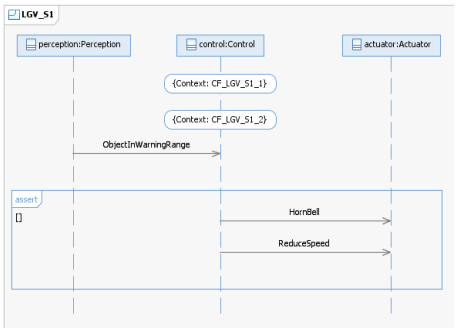
Application: Testing an LGV 2/3

- Scenario modelling: Capturing the situations to be avoided or reached
  - Example: The robot shall horn the bell and reduce its speed when a human comes to the warning area

Context fragments:



Scenario:

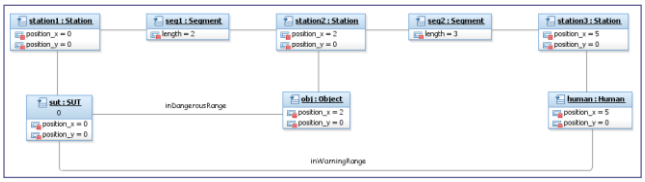


Application: Testing an LGV 3/3

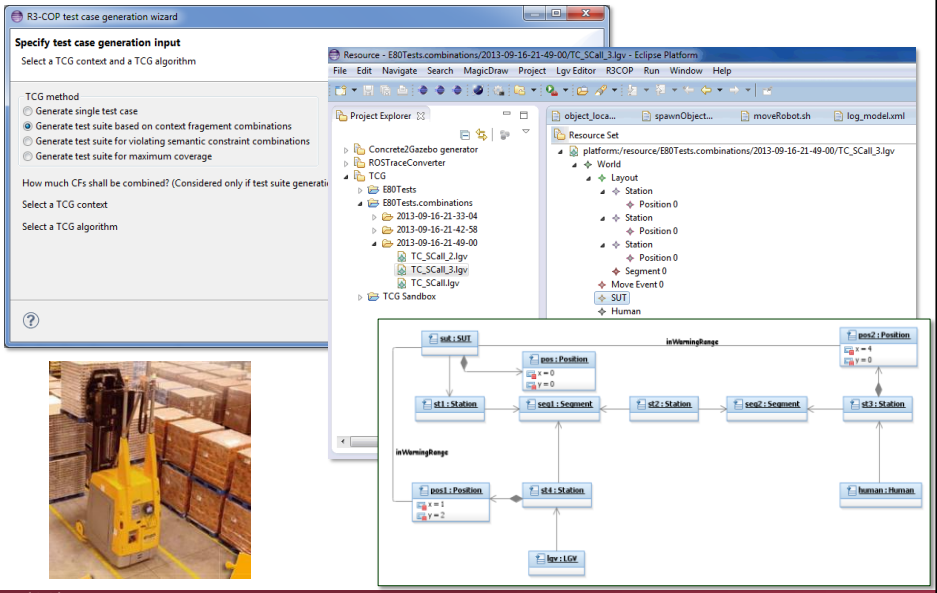
- Test generation: Coverage of individual and combined scenarios + mapping abstract test cases to concrete
  - Example: Obstacle in the dangerous area, human in the warning area (S1+S3)

← Abstract test data

Concrete test data →



Tools, models and test execution environment



Summary

- Model based robustness testing approach
  - Context modelling: “What is possible?”
  - Scenario modelling: “What is required?”
  - Initial context fragment: “What is relevant?”
  - Testing strategy: “What is stressful?”
- Developing methods and tools
  - Context and requirements modelling
  - Generating test data for testing robustness and safety of the context-aware behaviour
  - Generating test oracles for test evaluation
- Applications and validation
  - Household robot (ROS based simulator)
  - Laser guided forklift (real configurations)