Testing the robustness and safety of context-aware autonomous behavior

Research Report

Istvan Majzik
Budapest University of Technology and Economics

64th Meeting of the IFIP 10.4 Working Group on Dependable Computing and Fault Tolerance, Visegrád, Hungary, June 28-30, 2013

Outline

Introduction

- Ongoing research activity
  - Development of methods and tools for the efficient verification and testing of dependable and safety-critical systems
- Focus of this talk 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

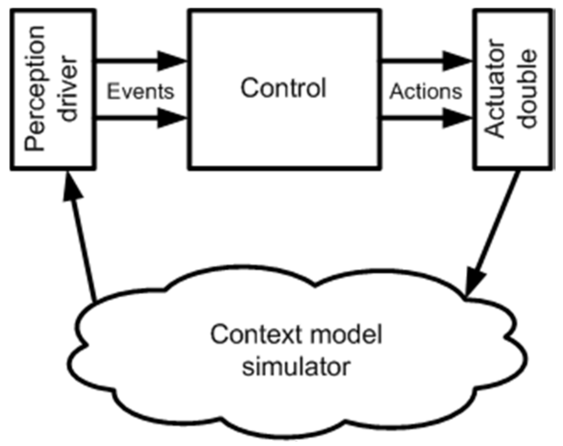
Objectives and applications

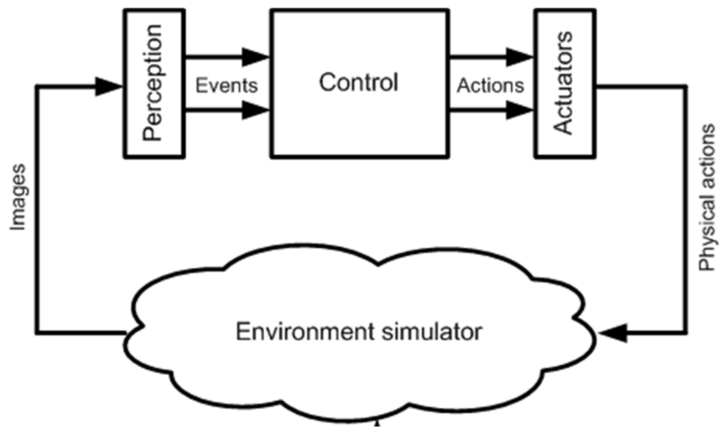
- Objectives of testing
  - Robustness and safety: safe system behavior in the presence of stressful environmental conditions
  - Context-awareness: dependency of the system behavior on the evolving state of the complex environment (context)
- Systems to be tested in the project
  - Autonomous robots: household and manufacturing
  - Autonomous vehicles: LGV, UAV, RUAV
- 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”*

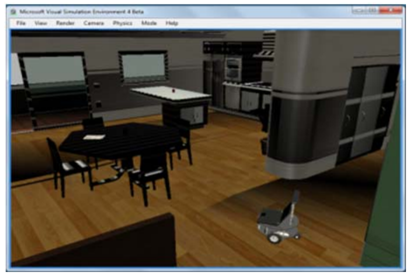
  $\rightarrow$ Context-related conditions (initial, interim and final), and corresponding sequence of actions

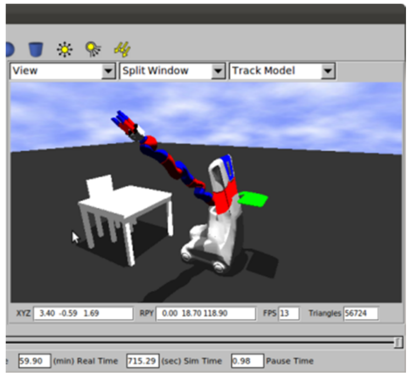
Test purpose and challenges

- Black-box testing of the control functions







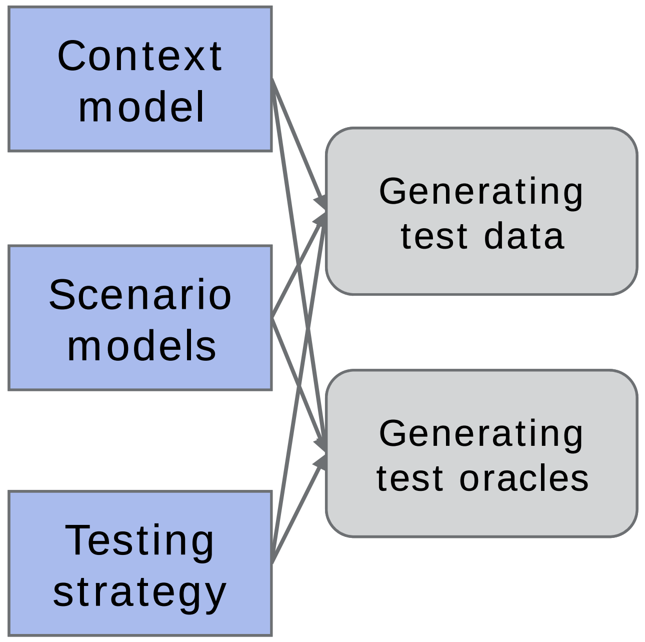


  - Simulator: Context can be configured, and actions can be observed during a mission

- Challenges
  - Complex context, large number of possible situations
  - Informal requirements → Precise representation
  - Ad-hoc test data → Systematic robustness testing
  - Lack of test quality metrics → Coverage metrics

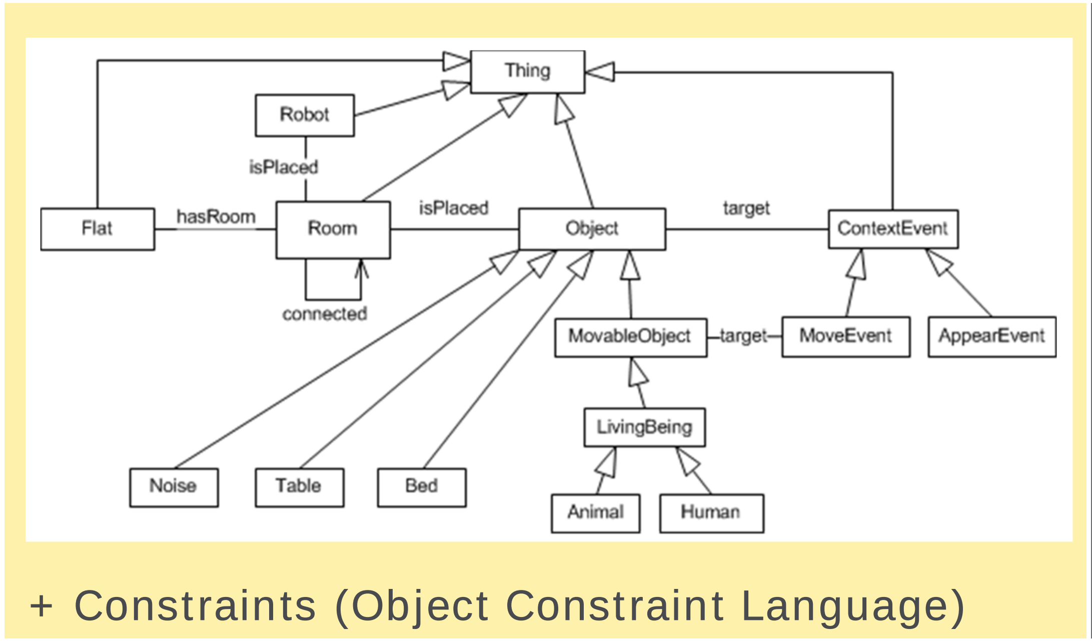
Testing concept

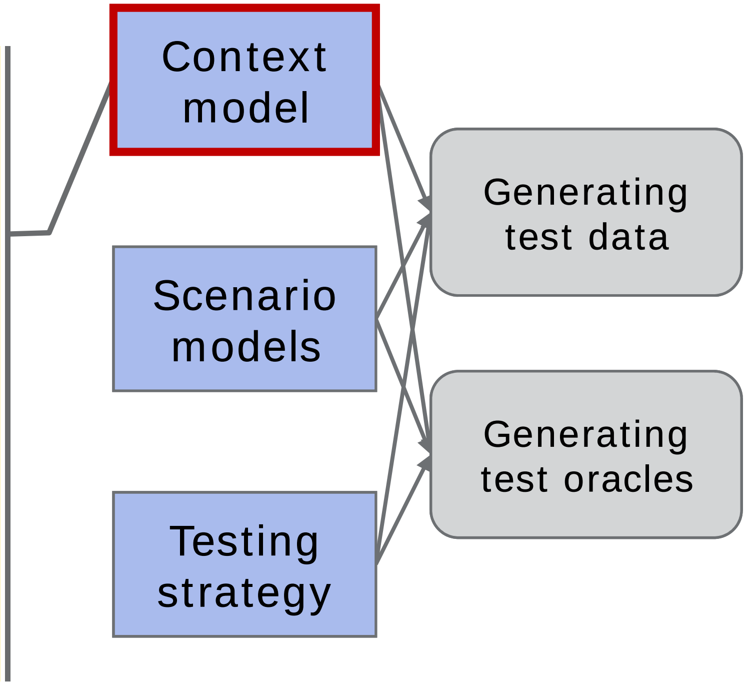
- Test goals
  - Systematic generation of test data,
    i.e., test contexts that include stressful (unexpected) situations
  - Evaluating the safety of the observed behavior
- What is needed?
  - Description of the environment:
    Context modeling
  - Capturing the test requirements:
    Scenario modeling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



Context modeling

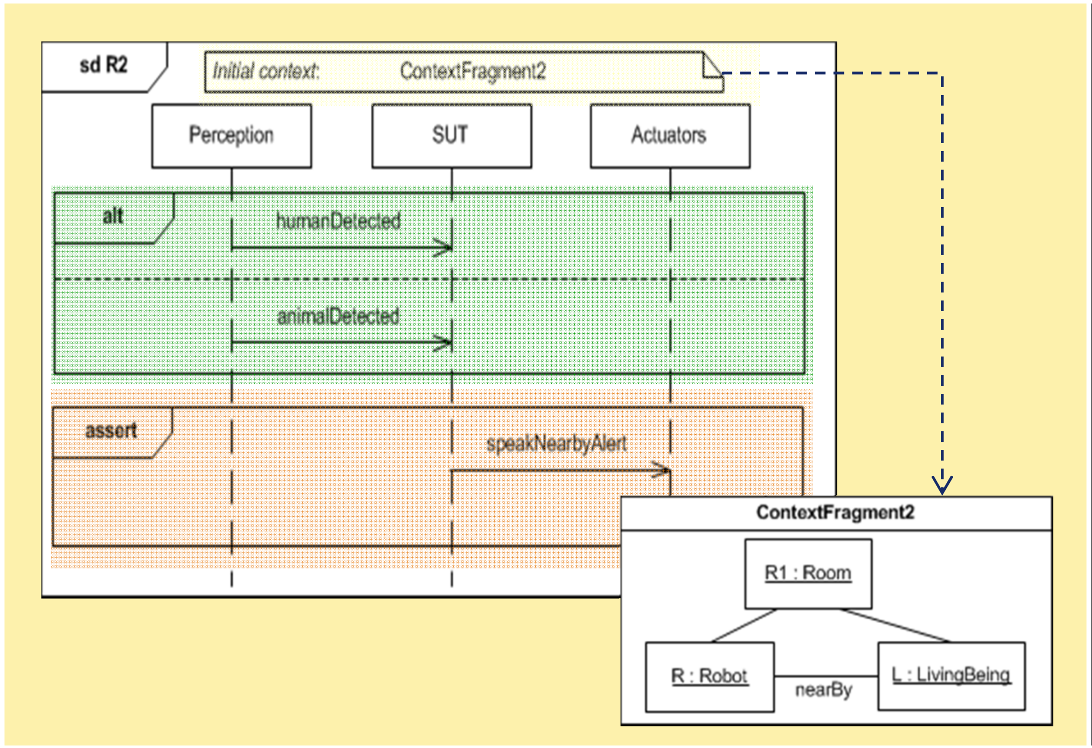
- Domain ontology → Context metamodel
  - Objects with properties, including dynamic objects
  - Relations: concrete or abstract relations
  - Constraints: physical limitations and application-specific constraints
- Action model for control actions

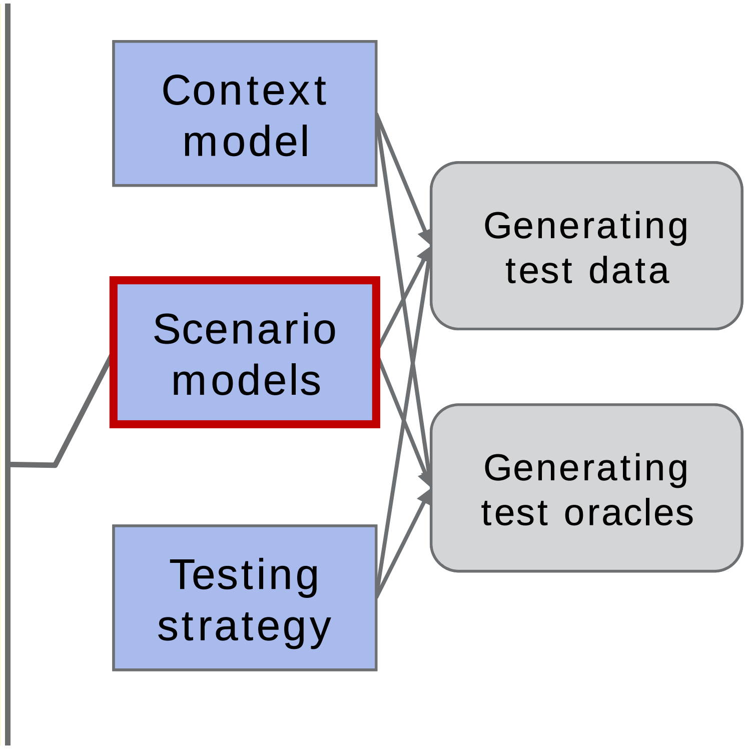




Scenario modeling

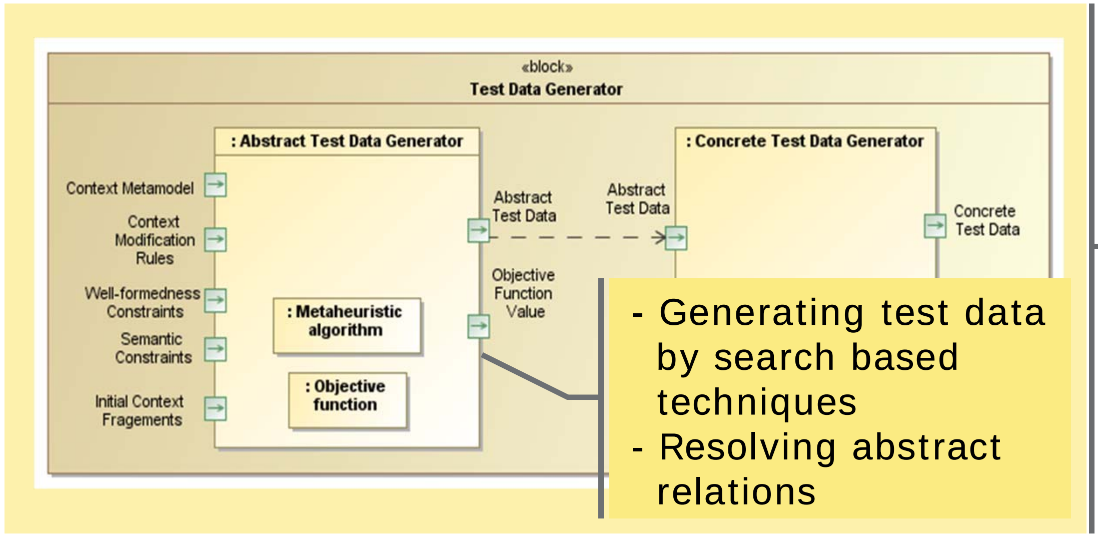
- Safety requirements → Scenario model
  - Initial context fragment: instances of the context metamodel
  - Condition part: Events (perception) and messages (commands)
  - Assertion part: Expected or forbidden actions and context changes
  - Precise semantics based on MSC and LSC constructs with time



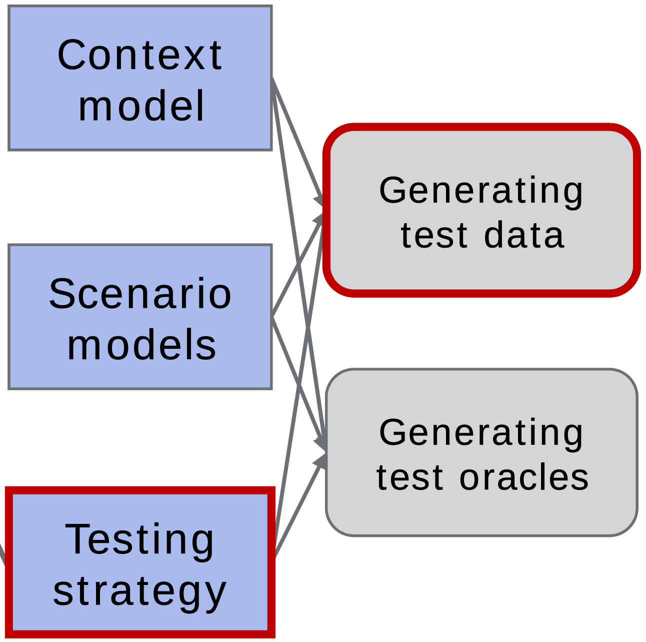


Strategies for testing robustness

- 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

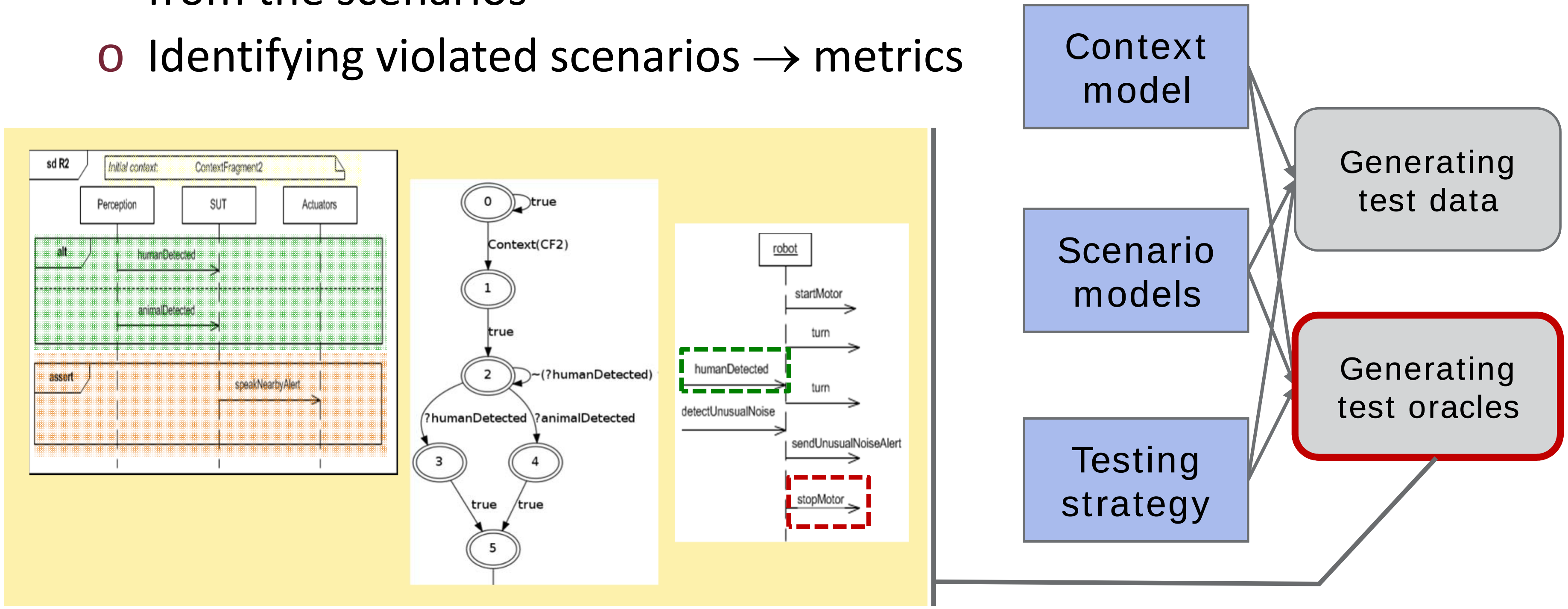

«block»
Test Data Generator
: Abstract Test Data Generator
: Concrete Test Data Generator
Context Metamodel
Context Modification Rules
Well-formedness Constraints
Semantic Constraints
Initial Context Fragements
: Metaheuristic algorithm
: Objective function
Abstract Test Data
Abstract Test Data
Objective Function Value
Concrete Test Data

- Generating test data by search based techniques
- Resolving abstract relations



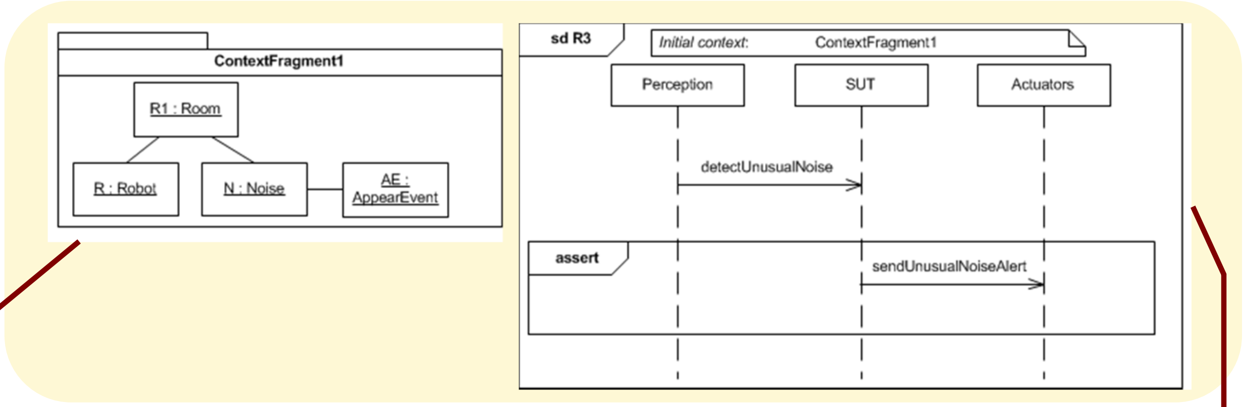
Evaluating test traces

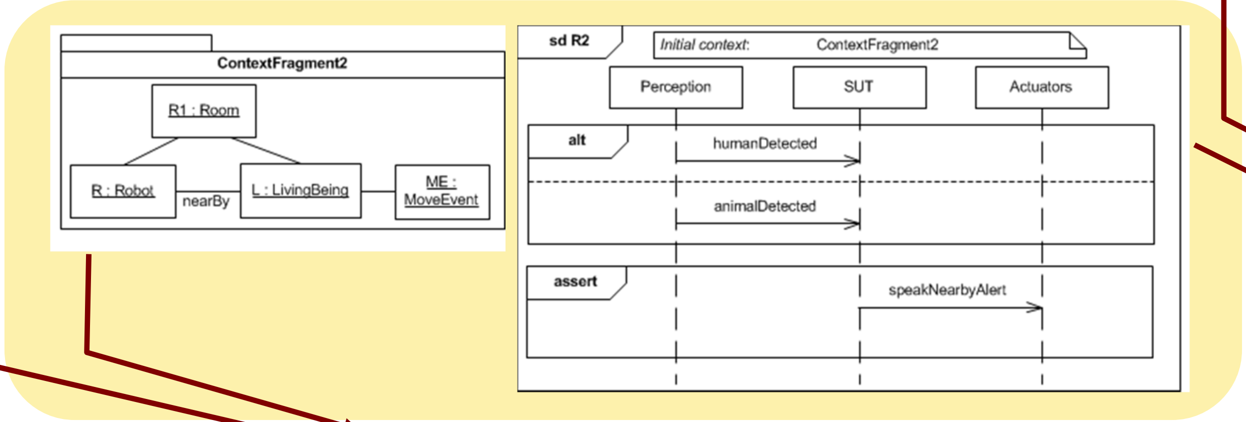
- Evaluating a test trace simultaneously against each requirement scenario, from each relevant step
  - Matching events and context changes (respecting object hierarchy): using efficient graph-based decomposition techniques
  - Automated evaluation of traces: synthesis of observer automata from the scenarios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



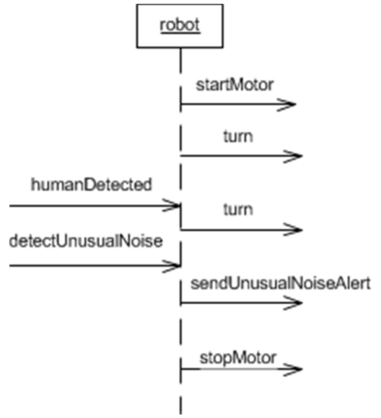
Example: Combining initial contexts

Test requirement scenarios:

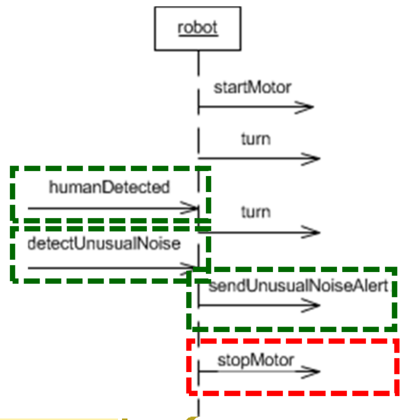




Test trace:

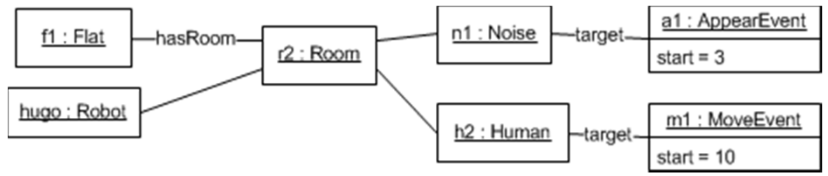


Evaluation:

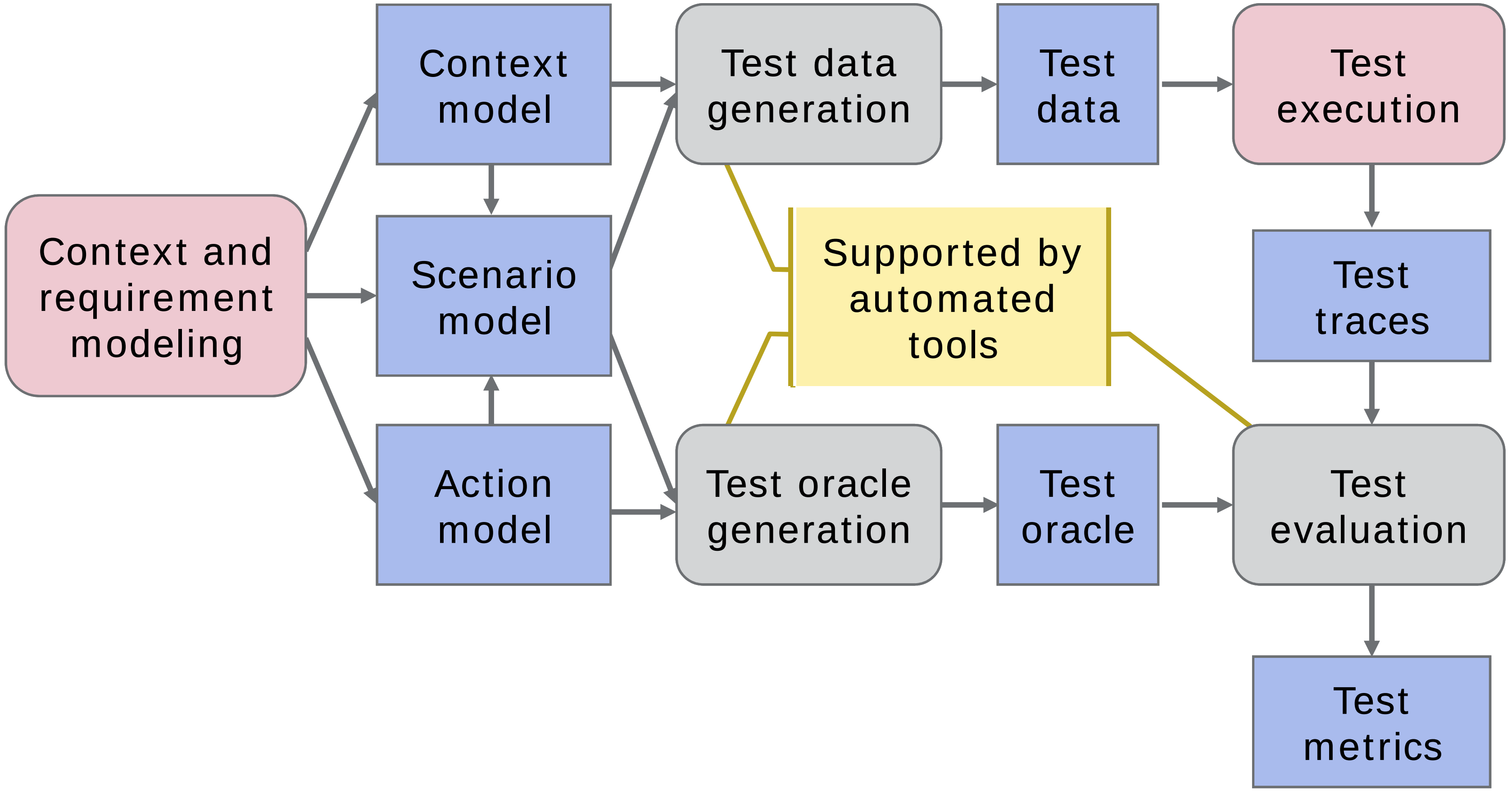


Test verdict:
R3 passed
R2 failed

Test data generated:



The testing method



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

Summary

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