The role of testing in verification and certification

Kerstin Eder
Design Automation and Verification, Microelectronics
[and Trustworthy Systems Laboratory]
Verification and Validation for Safety in Robots, Bristol Robotics Laboratory
What can be done to increase the productivity of simulation-based testing?


HRI Verification Challenges

- System complexity
  - HW
  - SW
  - People
- Concurrency
- Experiments in labs
  - Expensive
  - Unsafe
We are investigating...

- Testing in simulation
- Techniques well established in microelectronics design verification
  - Coverage-Driven Verification

... to verify code that controls robots in HRI.
Agency for Intelligent Testing

- Robotic assistants need to be both powerful and *smart*.
  - AI and learning are increasingly used in robotics
- We need *intelligent* testing.
  - No matter how clever your robot, the testing environment needs to reflect the *agency* your robot will meet in its target environment.
CDV to *automate* simulation-based testing
Why and how?

Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.
*Coverage-Driven Verification — An Approach to Verify Code for Robots that Directly Interact with Humans*. In Hardware and Software: Verification and Testing, pp. 69-84. Lecture Notes in Computer Science 9434. Springer, November 2015. (DOI [10.1007/978-3-319-26287-1_5](https://doi.org/10.1007/978-3-319-26287-1_5))

Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.
Coverage-Driven Verification
Robotic Code

J. Boren and S. Cousins, “The SMACH High-Level Executive”
Coverage-Driven Verification
Coverage-Driven Verification
Effective tests:
- legal tests
- meaningful events
- interesting events
- while exploring the system
  - typical vs extreme values

Efficient tests:
- minimal set of tests (regression)

Strategies:
- Pseudorandom (repeatability)
- Effective tests:
  - legal tests
  - meaningful events
  - interesting events
  - while exploring the system
    - typical vs extreme values

- Efficient tests:
  - minimal set of tests (regression)

- Strategies:
  - Pseudorandom (repeatability)
  - Constrained pseudorandom
Test Generator

- **Effective tests:**
  - legal tests
  - meaningful events
  - interesting events
  - while exploring the system
    - typical vs extreme values

- **Efficient tests:**
  - minimal set of tests (regression)

- **Strategies:**
  - Pseudorandom (repeatability)
  - Constrained pseudorandom
  - Model-based to target specific scenarios
Model-based Test Generation

Example trace:

State: robot.start, human.start
Transitions:
human to human.activateRobot
robot to robot.activateRobot

State: robot.activateRobot, human.activateRobot, time++=40
Transitions:
robot to robot.getPiece

State: robot.getPiece, human.activateRobot
Transitions:
human to human.waitSignal
robot to robot.informHuman...

State: robot.informHuman..., human.waitSignal...

High-level stimulus:

send_signal activateRobot
set_param time = 40
receive_signal informHumanOfHandoverStart
send_signal humanIsReady
set_param time = 10
set_param b_onTask = true
set_param b_gazeOk = true
set_param b_pressureOk = true
set_param b_locationOk = true
Model-based Test Generation

High-level stimulus

```
send_signal activateRobot
set_param time = 40
receive_signal informHumanOfHandoverStart
send_signal humanIsReady
set_param time = 10.
set_param h_onTask = true
set_param h_gazeOk = true
set_param h_pressureOk = true
set_param h_locationOk = true
```

"Human" actions in ROS

```
Send signal
  Delay
  Receive signal
  Send signal
  Delay
  Set gaze, pressure and location
  Set gaze, pressure and location
  Interaction done
```

Parameter instantiation:
- Time: 2 s
- Gaze: (0.1 m, 0.5 m, 40°)
  Location: (0.45 m, 0.05 m, 0.73 m)
- Gaze: (0.1 m, 0.5 m, 30°)
  Pressure: (15, 120, 140) to (7, 90, 100)
  Location: (0.45 m, 0.05 m, 0.73 m)
Model-based test generation

Formal model → Traces from model checking → Test template → Test components:
- High-level actions
- Parameter instantiation

System + environment

Environment to drive system
Coverage-Driven Verification
Requirements as assertion monitors:
- if [precondition], check [postcondition]
  
  "If the robot decides the human is not ready, then the robot never releases an object".

- Implemented as automata

Continuous monitoring at runtime, self-checking

- High-level requirements
- Lower-level requirements depending on the simulation's detail (e.g., path planning, collision avoidance).

`assert {! (robot_3D_position == human_3D_position)}`
Coverage-Driven Verification
Coverage-Driven Verification
Coverage Collector

- Coverage models:
  - Code coverage
  - Structural coverage
  - Functional coverage
    - Requirements coverage
HRI Handover Scenario

Requirements:
- Functional and safety (ISO 13482:2014, ISO 10218-1)
Requirements based on ISO 13482 and ISO 10218

1. If the gaze, pressure and location are sensed as correct, then the object shall be released.
2. If the gaze, pressure or location are sensed as incorrect, then the object shall not be released.
3. The robot shall make a decision before a threshold of time.
4. The robot shall always either time out, decide to release the object, or decide not to release the object.
5. The robot shall not close the gripper when the human is too close.
6. The robot shall start in restricted speed and force.
7. The robot shall not collide with itself at high speeds.
8. The robot shall operate within allowable maximum values to avoid dangerous unintentional collisions with humans and other safety-related objects.
Requirements based on ISO 13482 and ISO 10218

1. If the gaze, pressure and location are sensed as correct, then the object shall be released.
2. If the gaze, pressure or location are sensed as incorrect, then the object shall not be released.
3. The robot shall make a decision before a threshold of time.
4. The robot shall always either time out, decide to release the object, or decide not to release the object.
5. The robot shall not close the gripper when the human is too close.
6. The robot shall start in restricted speed and force.
7. The robot shall not collide with itself at high speeds.
8. The robot shall operate within allowable maximum values to avoid dangerous unintentional collisions with humans and other safety-related objects.
Requirements based on ISO 13482 and ISO 10218

Considering a speed threshold of 250 mm/s (from ISO 10218-1), last requirement implemented as:

- The robot hand speed is always less than 250 mm/s.
- If the robot is within 10 cm of the human, the robot’s hand speed is less than 250 mm/s.
- If the robot collides with anything, the robot’s hand speed is less than 250 mm/s.
- If the robot collides with the human, the robot’s hand speed is less than 250 mm/s.
Coverage Collector

- Coverage models:
  - Code coverage
  - Structural coverage
  - Functional coverage
    - Requirements coverage
    - Cross-product functional coverage
      - Cartesian product of environment actions, sensor states and robot actions

Situation coverage – a coverage criterion for testing autonomous robots

Rob Alexander*, Heather Hawkins*, Drew Rae

* University of York, York, United Kingdom
† Griffith University, Brisbane, Australia

rob.alexander@york.ac.uk
## Functional Coverage

<table>
<thead>
<tr>
<th>(Gaze, Pressure, Location)</th>
<th>Sense timeout</th>
<th>Release piece</th>
<th>No release</th>
<th>Signal 1 timeout</th>
<th>Signal 2 timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I, I, I)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(I, I, I)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(I, I, 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(I, 1, I)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(I, 1, 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1, I, I)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1, I, 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1, 1, I)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1, 1, 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Sense timeout</th>
<th>Release piece</th>
<th>No release</th>
<th>Signal 1 timeout</th>
<th>Signal 2 timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>No activation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activation signal 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not on task</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
HRI Handover Scenario

Coverage models:
- Code statement (robot high-level control)
- Requirements in the form of Assertions
- Cross-product functional coverage
Coverage Results
Code Coverage Results

Pseudorandom
Constrained
Model-based

Coverage Hole
Assertion Coverage Results

- 100 pseudorandomly generated tests
- 100 constrained pseudorandomly generated tests
- 4 model-based tests
Functional Coverage Results

- 100 pseudorandomly generated tests
- 160 model-based tests
- 180 model-based constrained tests
- 440 tests in total
Coverage analysis enables feedback to test generation
CDV for Human-Robot Interaction

Coverage-Directed Verification

- **systematic, goal directed** verification method
  - high level of automation
  - capable of exploring systems of realistic detail under a broad range of environment conditions

- **focus on test generation and coverage**
  - constraining test generation requires significant engineering skill and SUT knowledge
  - **model-based test generation** allows targeting requirements and cross-product coverage more effectively than pseudorandom test generation
Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.

**Coverage-Driven Verification — An Approach to Verify Code for Robots that Directly Interact with Humans.** In Hardware and Software: Verification and Testing, pp. 69-84. Lecture Notes in Computer Science 9434. Springer, November 2015. (DOI: 10.1007/978-3-319-26287-1_5)

Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.

CDV provides *automation*

What about *agency*?
10// INITIAL BELIEFS
12// preparing_for_flight
13// initialising_systems
14// hardware_system_passed_test
15// has_read_flight_environment_model
16// has_read_new_flight_path
17// pilot_comms_work
18// all_beacon_comms_work
19// created_flight_path_execution_plan
20// plan_is_unsafe_for_energy_level_available(Flight)
21// announced_text_object
22// ready_for_mission
23// on_ground_before_flight
24// ground_testing
25// responded_to_take_off_permission
26// permission_given_for_take_off
27// flying
28// take_off_testing
29// there_is_flight_system弱点_to_report
30// responded_to_start_mission
31// on_mission
32// people_pause
33// vehicle_pause
34// flying_pause
35// avoiding Behaviour
36// power_return
37// emergency_Landing
38// in_manual_control
39// landed
40
41// Environment Events and States
42
43// people_appearing
44// vehicles_appearing
45// flying_object_appearing
46// weather_too_bad
47// visibility_too_bad
48// on_board_faults
49// command_received
50// manual_control_request
51
52// EXECUTABLE PLANS
53// executable_plan: 
54// network_priority_text.once: take_off:
55// ready_for_mission :=
56// invoke_comms_rnd.once, asking_for_permission, "take off", []
57
58// executable_plan:
59// there_is_flight_system弱点_to_report:
60// ready_for_mission & received_take_off_permission :=
61// invoke_comms_rnd.once, announcing_text_object, "M1", []
62
63// executable_plan:
64// there_is_flight_system弱点_to_report:
65// ready_for_mission & received_take_off_permission :=
66// invoke_comms_rnd.once, announcing_text_object, "M2", []
67
68// executable_plan:
69// new_comands.has_arrived(Com): 
70// ready_for_mission & on mission :=
71// invoke_comms_rnd.once, interpreting_comands, Com
72// interpret_comands, command_uncler, did not yet acknowledge all commands w/ 
73
74// executable_plan:
75// set_new_command:
76// ready_for_mission & not yet acknowledge all_comands :=
77// invoke_comms_rnd.once, announcing_text_object, "Mx", []
78// announce_text_object, did not hear my text object w/ 
79// announce_text_object, waiting_for_public_response, "Mx", "2048", []
80
81// executable_plan:
82// pilot_disapproved:
83// ready_for_mission :=
84// invoke_comms_rnd.once, announcing_constant_text, \"Please repeat your instructions\", []
85
Belief-Desire-Intention Agents

Desires: goals to fulfil

Beliefs: knowledge about the world

Intentions: chosen plans, according to current beliefs and goals

Guards for plans

New goals

New beliefs

From executing plans
Intelligent testing is harnessing the power of BDI models to introduce agency into test environments.
Research Questions

- Are Belief-Desire-Intention agents suitable to model HRI?
- How can we exploit BDI agent models for test generation?
- Can machine learning be used to automate test generation in this setting?
- How do BDI agent models compare to automata-based techniques for model-based test generation?
Interacting Agents

- BDI can model agency in HRI
  - Interactions between agents create realistic action sequences that serve as test patterns
Interacting Agents

- BDI can model agency in HRI
  - Interactions between agents create realistic action sequences that serve as test patterns
Verification Agents

- Meta agents can influence beliefs
- This allows biasing/directing the interactions
Which beliefs are effective?

[Diagram with labels: (Meta Agent) Verification Agent, Agent for Simulated Human, Agents for Simulated Sensors, Robot's Code Agent, belief subsets, Manual belief selection]
Which beliefs are effective?

Manual belief selection
Random belief selection

(Meta Agent)
Verification
Agent

Robot's
Code
Agent

Agent for Simulated Human

Agents for Simulated Sensors

belief subsets

beliefs

belief subsets

beliefs

belief subsets

beliefs
Which beliefs are effective?

Optimal belief sets determined through RL

Plan coverage

Belief subsets

(Meta Agent) Verification Agent

Agent for Simulated Human

Agents for Simulated Sensors

Robot's Code Agent
Results

How effective are BDI agents for test generation?
How do they compare to model checking timed automata?


D. Araiza-Illan, A.G. Pipe, K. Eder
The cost of learning a good belief set needs to be considered when assessing the different BDI-based test generation approaches.
Code Coverage Results
BDI-agents vs timed automata

Effectiveness:
- high-coverage tests are generated quickly
### BDI-agents vs timed automata

<table>
<thead>
<tr>
<th></th>
<th>Model checking TA</th>
<th>BDI agents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cooperative Manufacturing Assistant</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model’s lines of code</td>
<td>725</td>
<td>348</td>
</tr>
<tr>
<td>No. states (transitions) or plans</td>
<td>53 (72)</td>
<td>79</td>
</tr>
<tr>
<td>Modelling time</td>
<td>(\approx 10.5) hrs</td>
<td>(\approx 6) hrs</td>
</tr>
<tr>
<td>Model explor. time (min/test)</td>
<td>0.001 s</td>
<td>5 s</td>
</tr>
<tr>
<td>Model explor. time (max/test)</td>
<td>33.36 s</td>
<td>5 s</td>
</tr>
<tr>
<td><strong>Home Care Assistant</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model’s lines of code</td>
<td>722</td>
<td>131</td>
</tr>
<tr>
<td>No. states (transitions) or plans</td>
<td>42 (67)</td>
<td>35</td>
</tr>
<tr>
<td>Modelling time</td>
<td>(\approx 5.5) hrs</td>
<td>(\approx 3) hrs</td>
</tr>
<tr>
<td>Model explor. time (min/test)</td>
<td>0.001 s</td>
<td>1 s</td>
</tr>
<tr>
<td>Model explor. time (max/test)</td>
<td>2.775 s</td>
<td>1 s</td>
</tr>
</tbody>
</table>
Back to our Research Questions

- **Belief-Desire-Intention agents** are suitable to model HRI
- **Traces of interactions** between BDI agent models provide **test templates**
- **Machine learning** (RL) can be used to automate the selection of belief sets so that test generation can be biased towards maximizing coverage
- Compared to traditional model-based test generation (model checking timed automata), BDI models are:
  - more intuitive to write, they naturally express agency,
  - smaller in terms of model size,
  - more predictable to explore and
  - equal if not better wrt coverage.
http://github.com/robosafe

D. Araiza Illan, D. Western, A. Pipe, K. Eder.  

D. Araiza Illan, D. Western, A. Pipe, K. Eder.  


D. Araiza-Illan, A.G. Pipe, K. Eder  
Thank you

Kerstin.Eder@bristol.ac.uk

Special thanks to Dejanira Araiza Illan, Jeremy Morse, David Western, Arthur Richards, Jonathan Lawry, Trevor Martin, Piotr Trojanek, Yoav Hollander, Yaron Kashai, Mike Bartley, Tony Pipe and Chris Melhuish for their collaboration, contributions, inspiration and the many productive discussions we have had.