# Gaining Confidence in the Correctness of Robotic and Autonomous Systems


Kerstin Eder

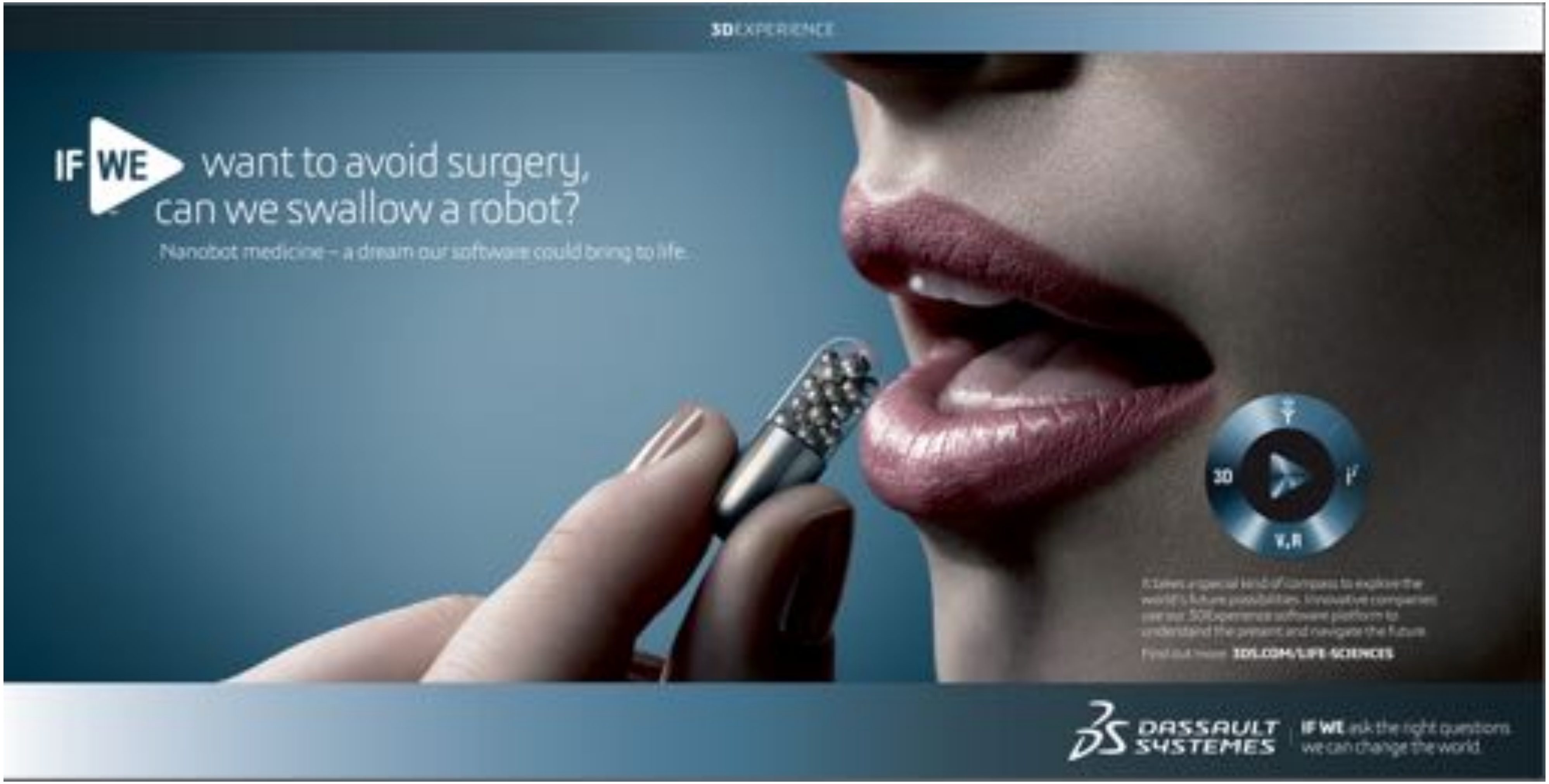
Design Automation and Verification

Trustworthy Systems, University of Bristol

Verification and Validation for Safety in Robots, Bristol Robotics Laboratory


University of Bristol EPSRC Engineering and Physical Sciences Research Council Innovate UK brl Bristol Robotics Laboratory

# Would you swallow a robot?



## Annex A
(informative)

# The Role of Testing in Verification and Validation

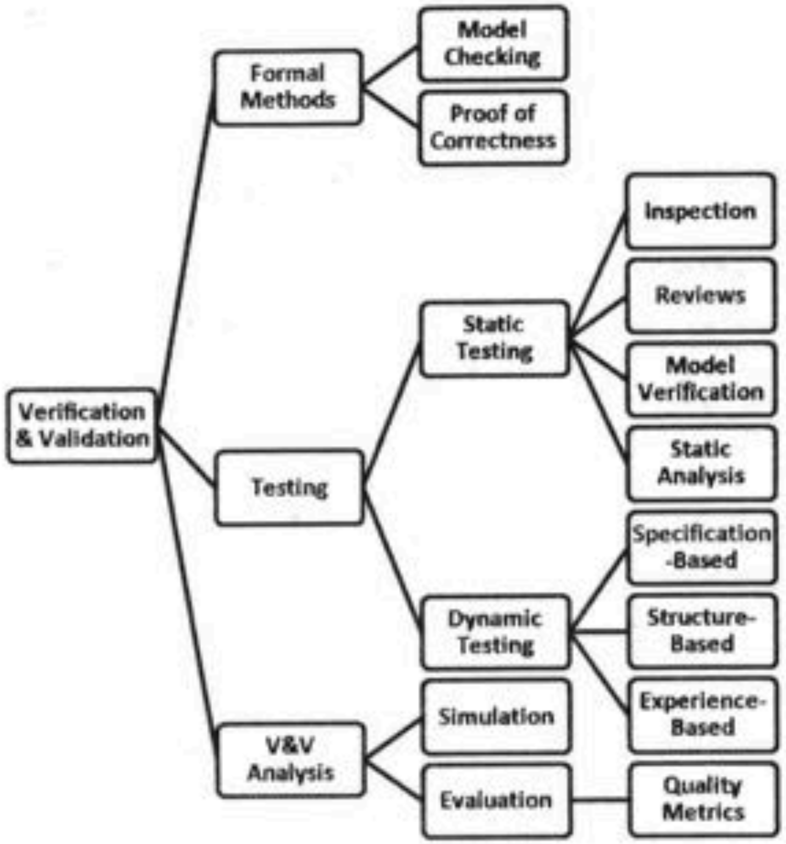



**Figure 11 — Hierarchy of Verification and Validation activities**

Figure 11 defines the complete nature of verification and validation (V&V) activities. V&V can be done on system, hardware, and software products. These activities and planning are defined and refined in IEEE 1012 and ISO/IEC 12207. Much of V&V is accomplished by testing. The ISO/IEC 29119 standard addresses the Dynamic and Static software testing (directly or via reference), thus covering parts of this verification and validation model. ISO/IEC 29119 is not intended to address all the elements of the V&V model, but it is important for a tester to understand where they fit within this model.

**DRAFT INTERNATIONAL STANDARD** ISO/IEC DIS 29119-1

ISO/IEC JTC **1** | Secretariat: **ANSI**

Voting begins on **2012-10-09** | Voting terminates on **2013-01-09**

**Software and systems engineering — Software testing —**

Part 1:
**Concepts and definitions**

*Ingénierie du logiciel et des systèmes — Essais du logiciel —*

*Partie 1: Concepts et définitions*

ICS 35.080

> **To expedite distribution, this document is circulated as received from the committee secretariat. ISO Central Secretariat work of editing and text composition will be undertaken at publication stage.**
>
> **Pour accélérer la distribution, le présent document est distribué tel qu'il est parvenu du secrétariat du comité. Le travail de rédaction et de composition de texte sera effectué au Secrétariat central de l'ISO au stade de publication.**

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

# The Safety Challenge

- Autonomous Systems
- Engineering Challenge
  - Advances in control engineering and ML
  - Focus on “making things work”

Pictures from www.wikipedia.org

# The Safety Challenge

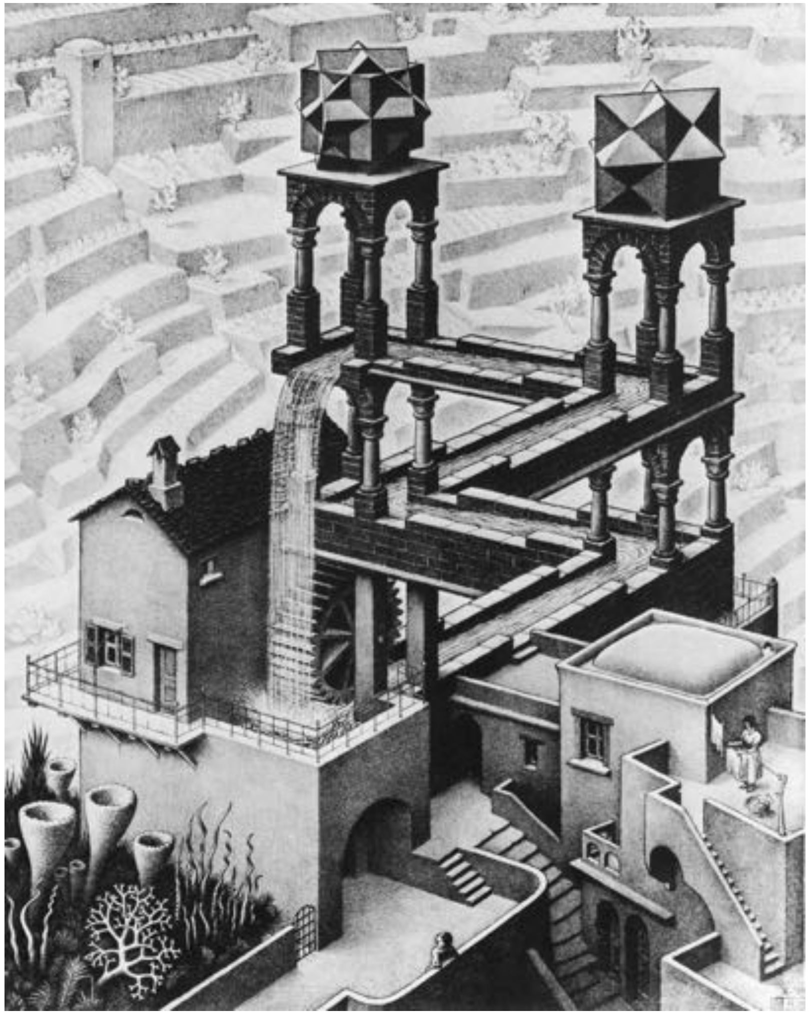
- Autonomous Systems
- Engineering Challenge
  - Advances in control science
  - Focus on “making things work”
- Fundamental concern:
  - Can such systems be trusted?

# Designing Trustworthy Systems

- Create flawless systems.

**AND**

- Design these systems in such a way that the flawlessness can be demonstrated.



"Waterfall" by M.C. Escher.

# EPSRC "Principles of Robotics"

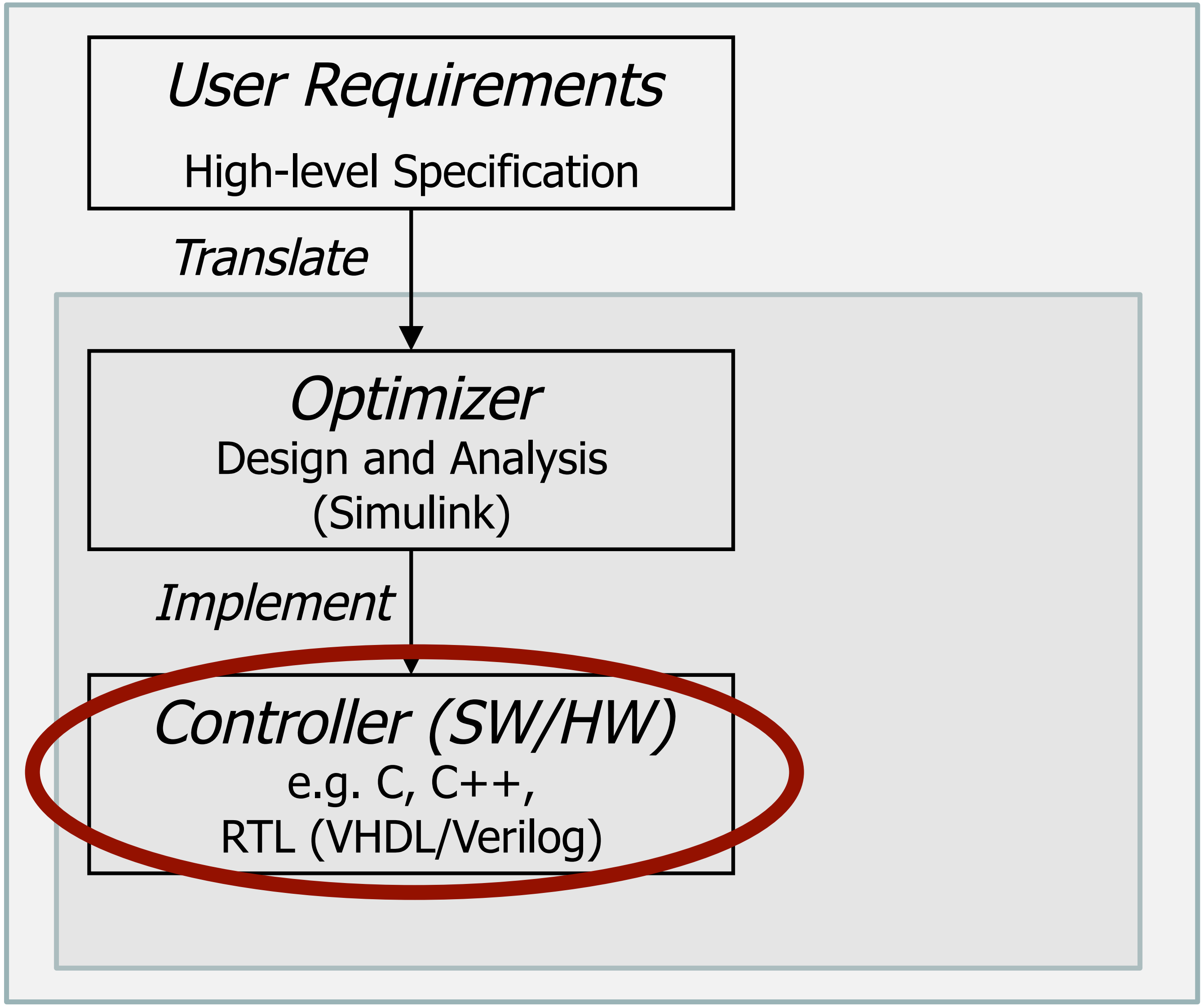
*"Robots are products.*
*They should be* ***designed*** *using* ***processes which assure*** *their* ***safety*** *and security."*

http://www.epsrc.ac.uk/ourportfolio/themes/engineering/activities/Pages/principlesofrobotics.aspx

# Verification and Validation for Safety in Robots

To develop techniques and methodologies that can be used to design autonomous intelligent systems that are verifiably trustworthy.

# Correctness from specification to implementation



# What can be done at the code level?

P. Trojanek and K. Eder.
***Verification and testing of mobile robot navigation algorithms: A case study in SPARK.***
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1489-1494. Sep 2014.
http://dx.doi.org/10.1109/IROS.2014.6942753

# What can go wrong in robot navigation software?

## Generic bugs:

- Array and vector out-of-bounds accesses
- Null pointer dereferencing
- Accesses to uninitialized data

## Domain-specific bugs:

- Integer and floating-point arithmetic errors
- Mathematic functions domain errors
- Dynamic memory allocation errors
- Concurrency bugs and blocking inter-thread communication (non real-time)

# Navigation in SPARK

- Three open-source implementations of navigation algorithms originally in C/C++ (2.7 kSLOC)
  - Vector Field Histogram
  - Nearness Diagram
  - Smooth Nearness-Diagram

| | **Driver** C++ | **Algorithm** C/C++ |
|---|---|---|
| VFH+ | 807 | 782 |
| ND | 828 | 1037 |
| SND | 403 | 941 |
| **Total** | 2038 | 2760 |

# Verification Approach

State of the art verification approaches:

- Model checking: infeasible
- Static analysis of C++: not possible
- Static analysis of C: requires verbose and difficult to maintain annotations

A *Design-for-Verification* approach:

- **SPARK,** a verifiable subset of Ada
  - software reliability a primary goal
  - SPARK specification and tools free for academic use
- Required code modifications:
  - Pre- and post-conditions, loop (in)variants
  - Numeric subtypes (e.g. Positive)
  - Formal data containers

# Navigation in SPARK

- Three open-source implementations of navigation algorithms translated from C/C++ (2.7 kSLOC) to SPARK (3.5 kSLOC)
  - Vector Field Histogram
  - Nearness Diagram
  - Smooth Nearness-Diagram

| | **Driver** | **Algorithm** | |
|---|---|---|---|
| | C++ | C/C++ | Ada |
| VFH+ | 807 | 782 | 918 |
| ND | 828 | 1037 | 1426 |
| SND | 403 | 941 | 1183 |
| **Total** | 2038 | 2760 | 3527 |

- Explicit annotations are less than 5% of the code
- SPARK code is on average 30% longer than C/C++

# Verification Conditions

| | Explicit annotations | | | | | Implicit run-time checks | | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Pre-conditions* | Post-conditions | Loop invariants* | Loop variants | Assertions | Divisions | Integer overflows | Floating-point overflows | Subtype ranges | Array indices | Record discriminants | |
| VFH+ | 46 (3) | 5 | 18 (9) | 0 | 23 | 36 | 36 | 120 | 100 | 102 | 262 | **748** |
| ND | 83 (18) | 10 | 8 (4) | 2 | 3 | 54 | 23 | 254 | 53 | 50 | 0 | **540** |
| SND | 104 (9) | 9 | 14 (7) | 2 | 30 | 29 | 1 | 140 | 22 | 0 | 24 | **375** |

* Separate verification conditions are generated for each call to subprogram with precondition, and similarly for initialization and preservation of each loop invariant; the numbers of explicit annotations are given in brackets.

# Formal Verification Outcome

| | Alt-Ergo 0.96 | Z3 4.3.1 | Alt-Ergo & Z3 combined | **Total** |
|---|---|---|---|---|
| VFH+ | 633<br>11 min | 699<br>37 min | 701<br>48 min | **748** |
| ND | 462<br>17 min | 482<br>21 min | 483<br>41 min | **540** |
| SND | 350<br>29 min | 366<br>6 min | 366<br>36 min | **375** |

Number of discharged verification conditions and the running time of static analysis based on two SMT solvers, Alt-Ergo and Z3

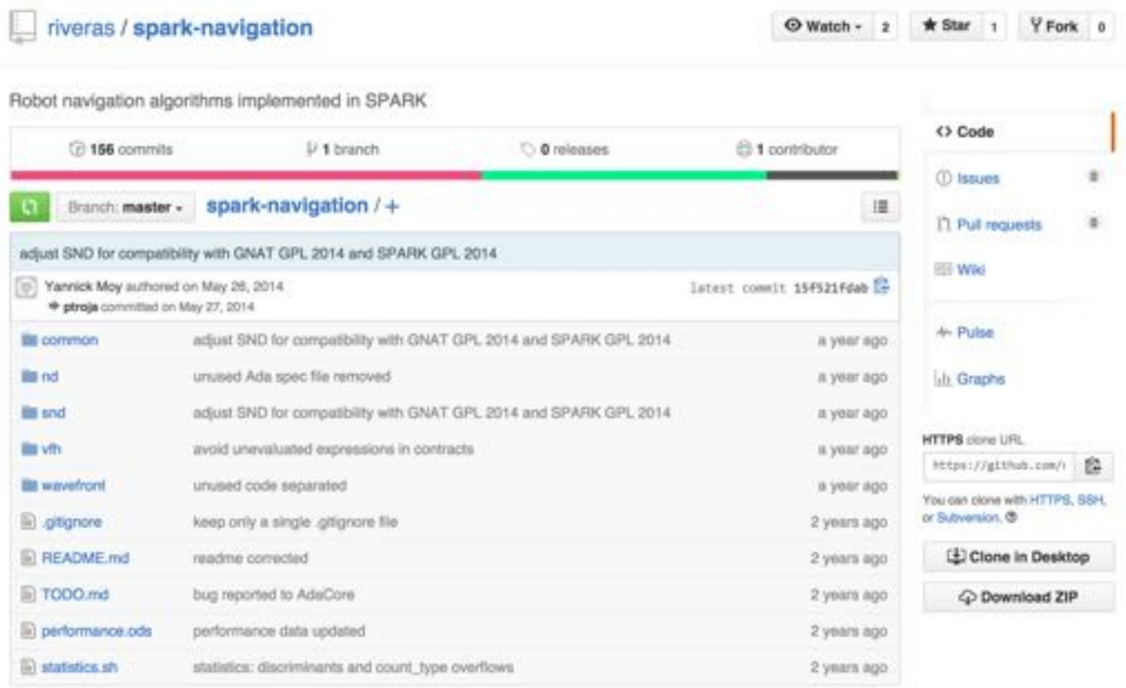
# Results

- Several bugs discovered by run-time checks injected by the Ada compiler
  - Fixed code proved to be run-time safe
    - except floating-point over- and underflows
    - These require the use of complementary techniques, e.g. abstract interpretation.
- Up to 97% of the verification conditions discharged automatically by SMT solvers in less than 10 minutes
- Performance of the SPARK and C/C++ code similar

# Moral

**If you want to make runtime errors an issue of the past, then you must select your tools (programming language and development environment) wisely!**

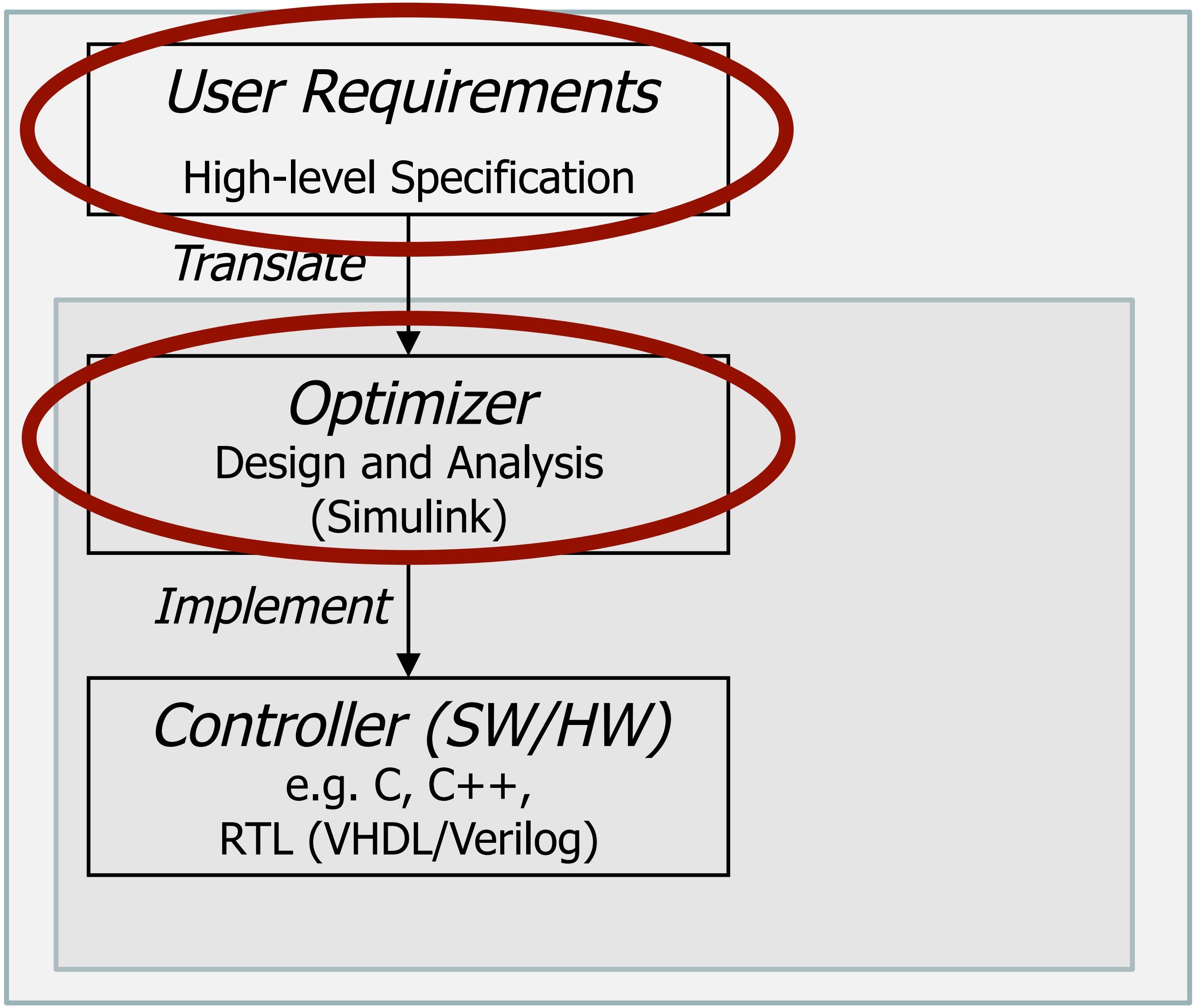

https://rclutz.wordpress.com/2016/09/23/hammer-and-nail/




# http://github.com/riveras/spark-navigation

P. Trojanek and K. Eder.
***Verification and testing of mobile robot navigation algorithms: A case study in SPARK.***
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1489-1494. Sep 2014.
http://dx.doi.org/10.1109/IROS.2014.6942753

# Correctness from Specification to Implementation

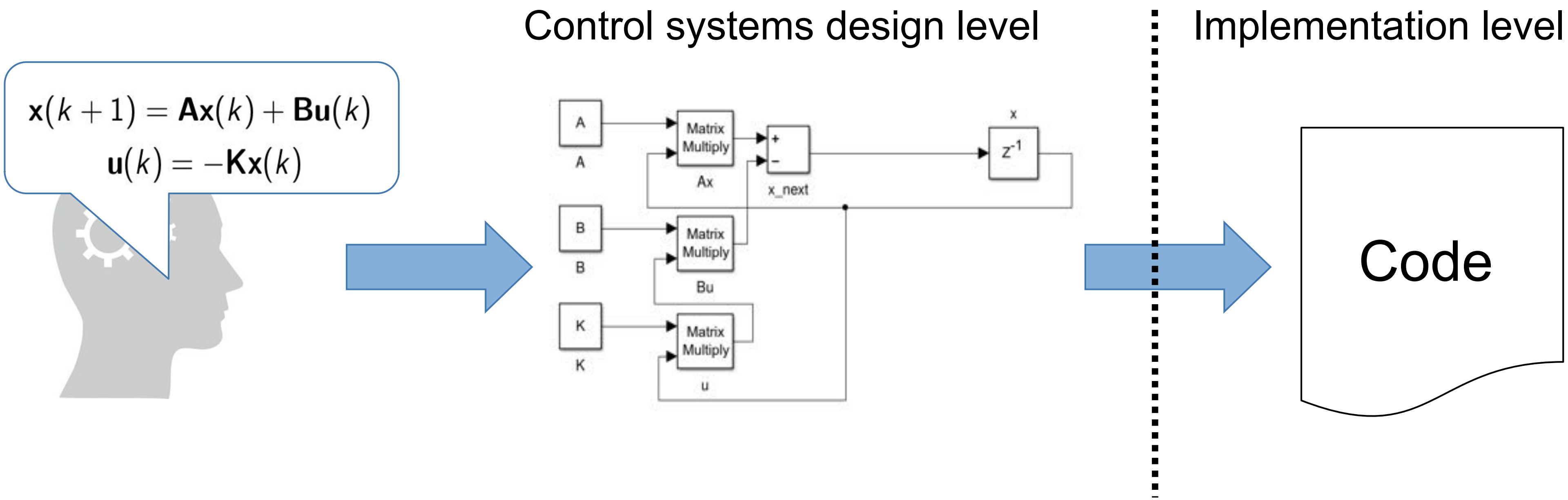


# What can be done at the design level?

D. Araiza Illan, K. Eder, A. Richards.
***Formal Verification of Control Systems' Properties with Theorem Proving.***
International Conference on Control (CONTROL), pp. 244 - 249. IEEE, Jul 2014.
http://dx.doi.org/10.1109/CONTROL.2014.6915147

D. Araiza Illan, K. Eder, A. Richards.
***Verification of Control Systems Implemented in Simulink with Assertion Checks and Theorem Proving: A Case Study*.**
European Control Conference (ECC), pp. 2670 - 2675. Jul 2015.
http://arxiv.org/abs/1505.05699

# Simulink Diagrams in Control Systems




- Simulating the control systems
- Principles of control systems theory (e.g., stability)
- Serve as requirements/specification
- For (automatic) code generation

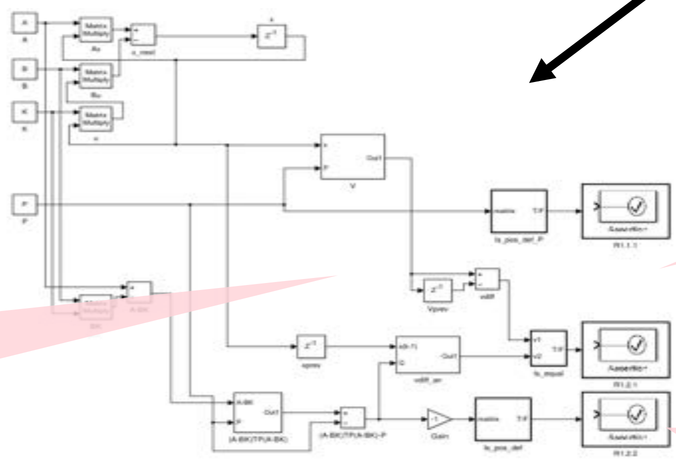
# Verifying Stability

**Stability**

Matrix **P** > 0
(Lyapunov function)

Matrix
$\mathbf{P-(A-BK)^T P(A-BK)} > 0$
(Lyapunov function's difference)

Equivalence
$V(k)-V(k-1) = x(k-1)^T \mathbf{[(A-BK)^T P(A-BK)-P]} x(k-1)$
(Lyapunov's equation application)

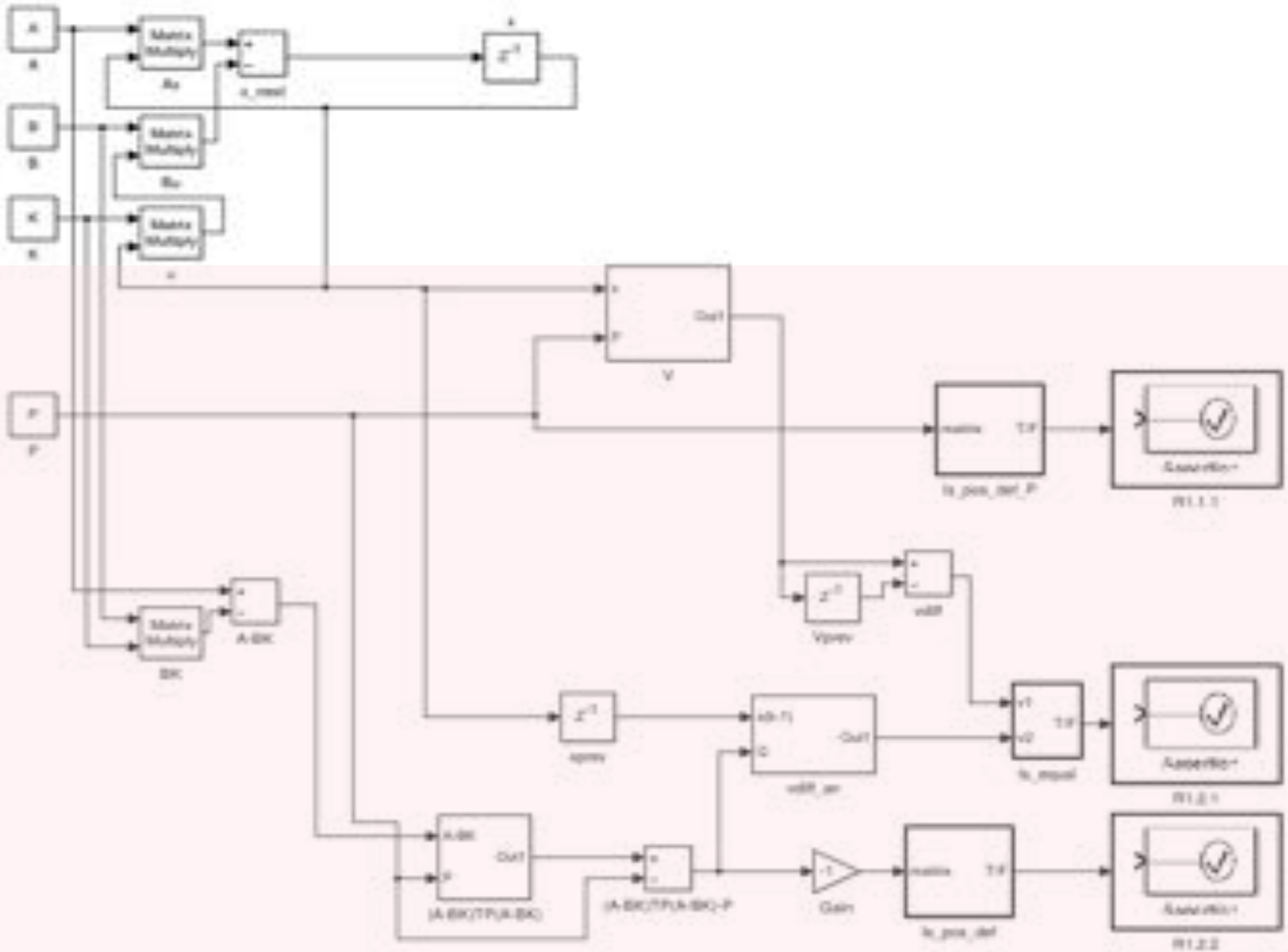


Capture control systems requirements

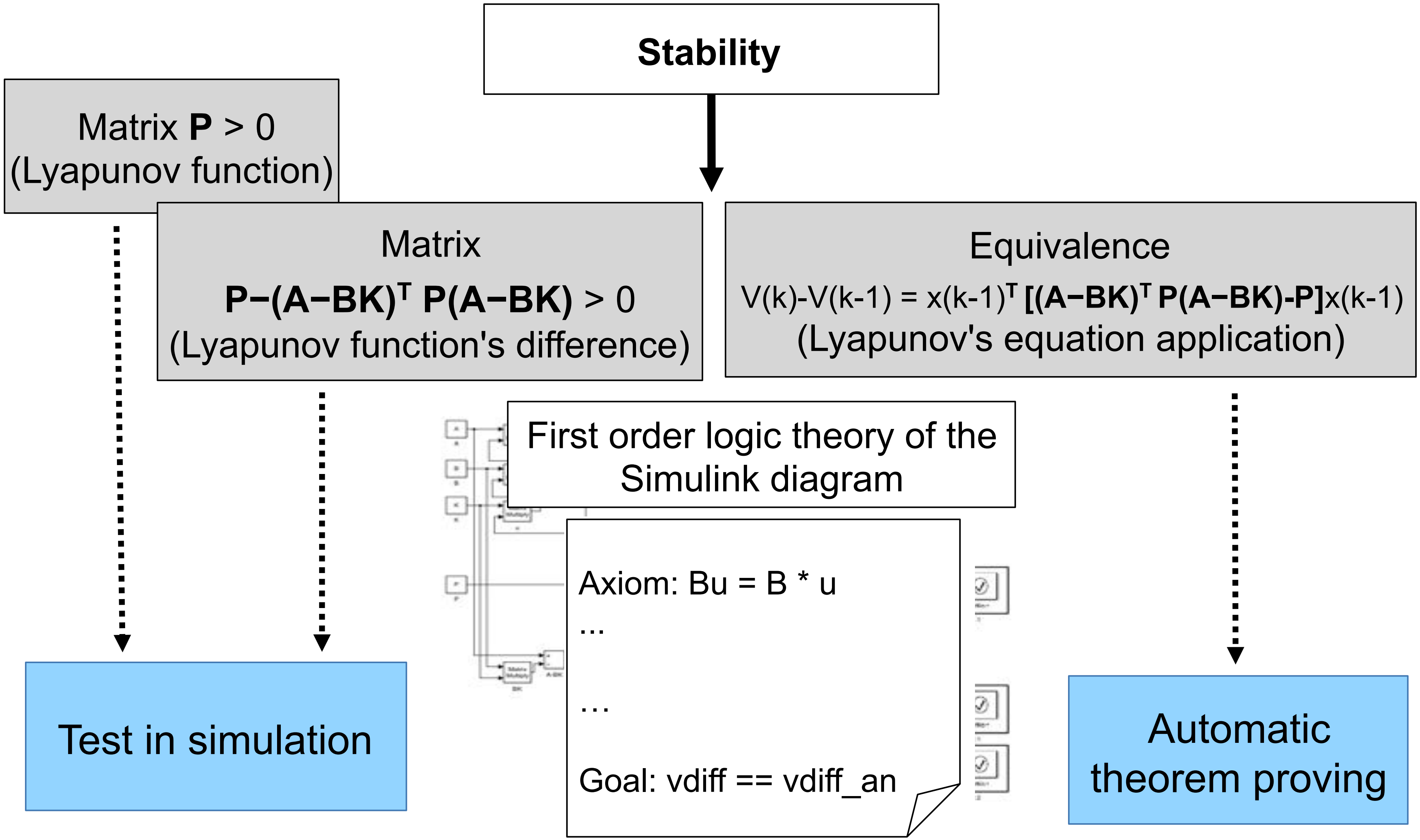
Add as assertions

Retain in code implementation

# Assertion-Based Verification



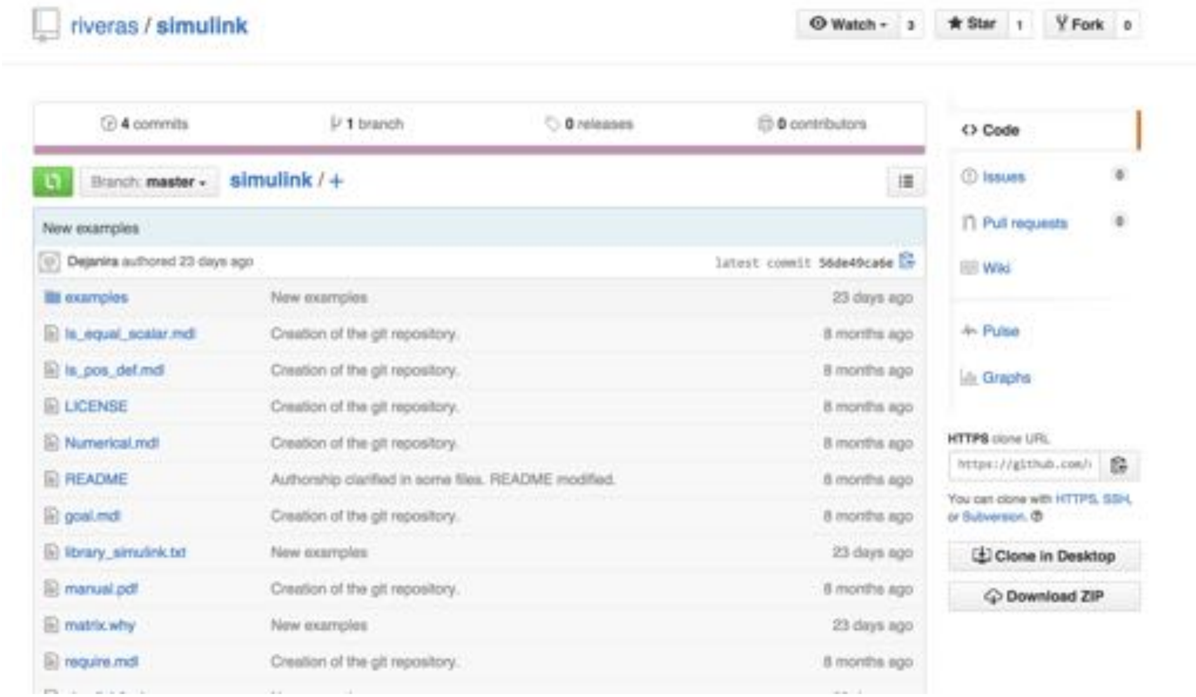
# Combining Verification Techniques



# Moral

**No single technique is adequate to cover a whole design in practice.**

**Combine techniques and learn from areas where verification is more mature.**

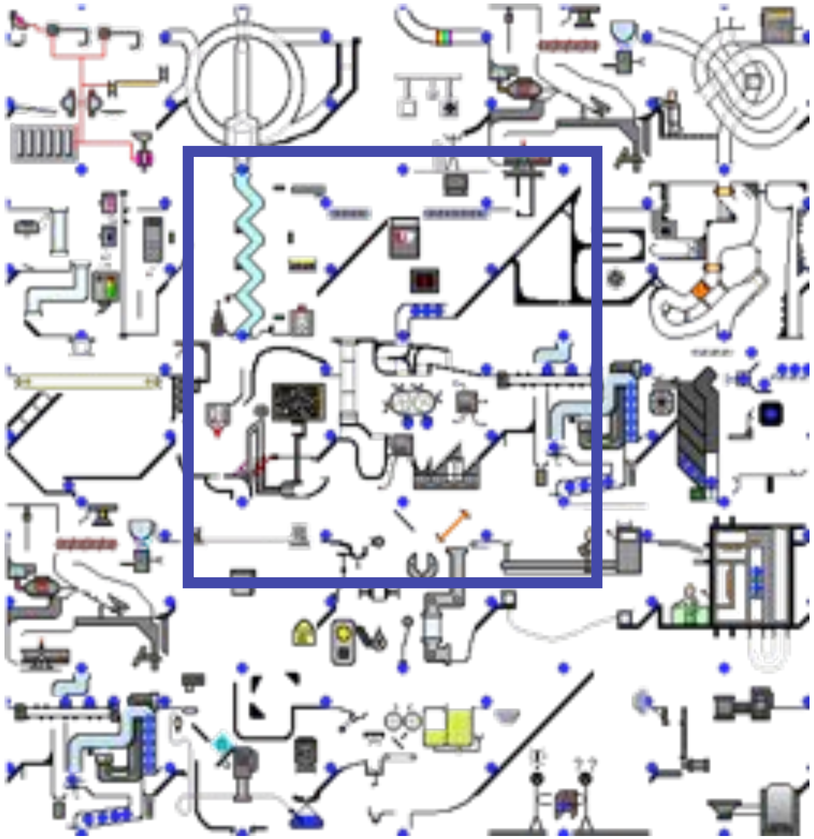



http://github.com/riveras/simulink

D. Araiza Illan, K. Eder, A. Richards.
***Formal Verification of Control Systems' Properties with Theorem Proving.*** International Conference on Control (CONTROL), pp. 244 - 249. IEEE, Jul 2014.
http://dx.doi.org/10.1109/CONTROL.2014.6915147

D. Araiza Illan, K. Eder, A. Richards.
***Verification of Control Systems Implemented in Simulink with Assertion Checks and Theorem Proving: A Case Study*.**
European Control Conference (ECC), pp. 2670 - 2675. Jul 2015.
http://arxiv.org/abs/1505.05699

# What can be done to advance simulation-based testing of RAS?

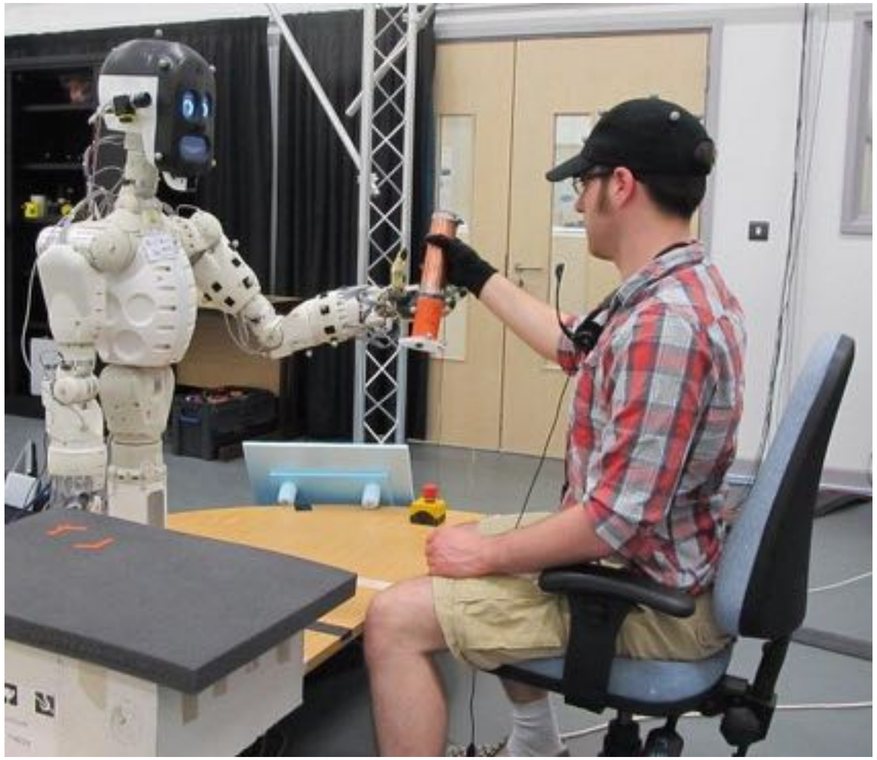



D. Araiza-Illan, D. Western, A. Pipe, and K. Eder, “Coverage-Driven Verification: An Approach to Verify Code for Robots that Directly Interact with Humans,” in Haifa Verification Conference, Haifa, Israel, 2015. http://link.springer.com/chapter/10.1007/978-3-319-26287-1_5

D. Araiza-Illan, D. Western, A. G. Pipe, and K. Eder, “Systematic and Realistic Testing in Simulation of Control Code for Robots in Collaborative Human-Robot Interactions,” in Towards Autonomous Robotic Systems (TAROS), Jun. 2016. http://link.springer.com/chapter/10.1007/978-3-319-40379-3_3

D. Araiza-Illan, A. G. Pipe, and K. Eder, “Intelligent Agent-Based Stimulation for Testing Robotic Software in Human-Robot Interactions,” in Third Workshop on Model-Driven Robot Software Engineering (MORSE), Leipzig, Germany, 2016. https://doi.org/10.1145/3022099.3022101

# Robot to human hand-over task

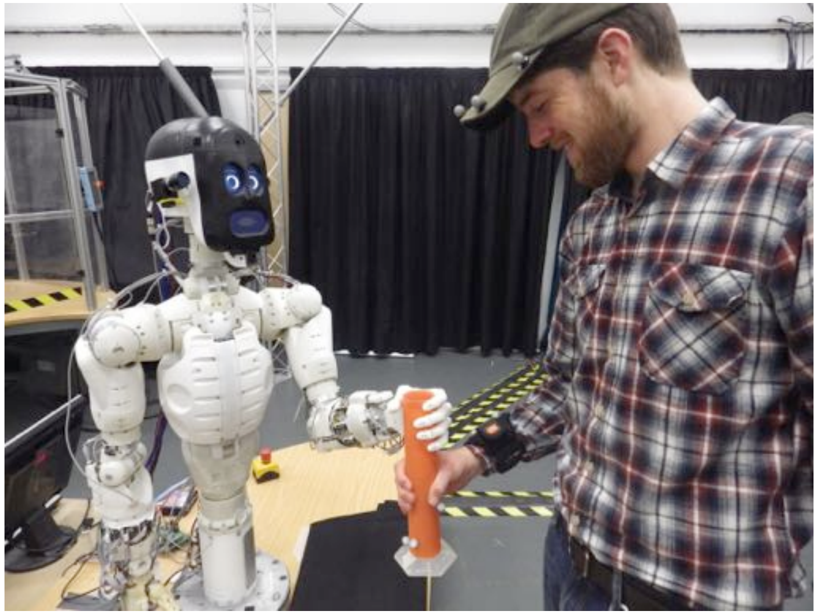


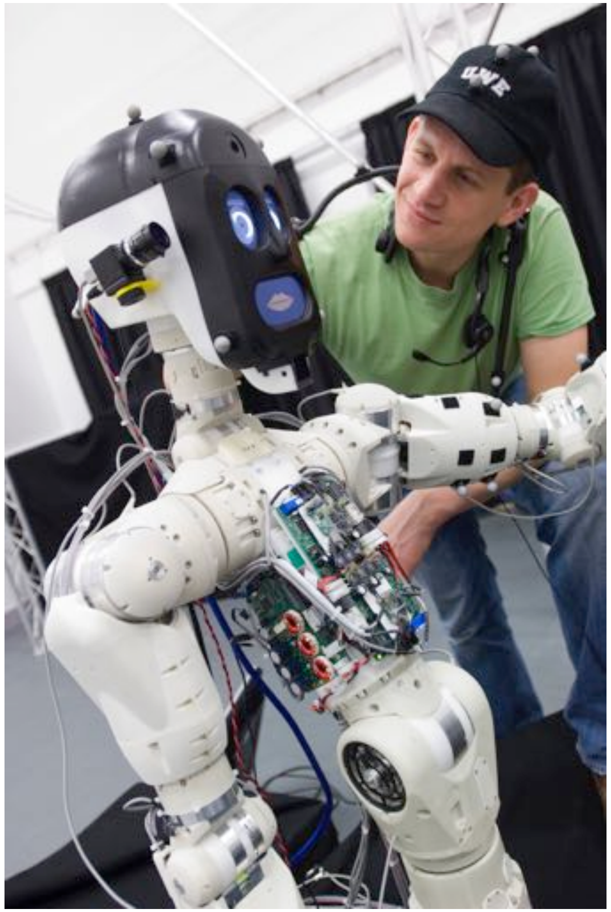
*When should the robot let go, i.e.*
*when is it safe for the robot to let go?*

# We are investigating…

- Testing in simulation
- Coverage-Driven Verification (CDV), a technique well established in microelectronics design verification

… to **verify** code that controls robots in HRI.



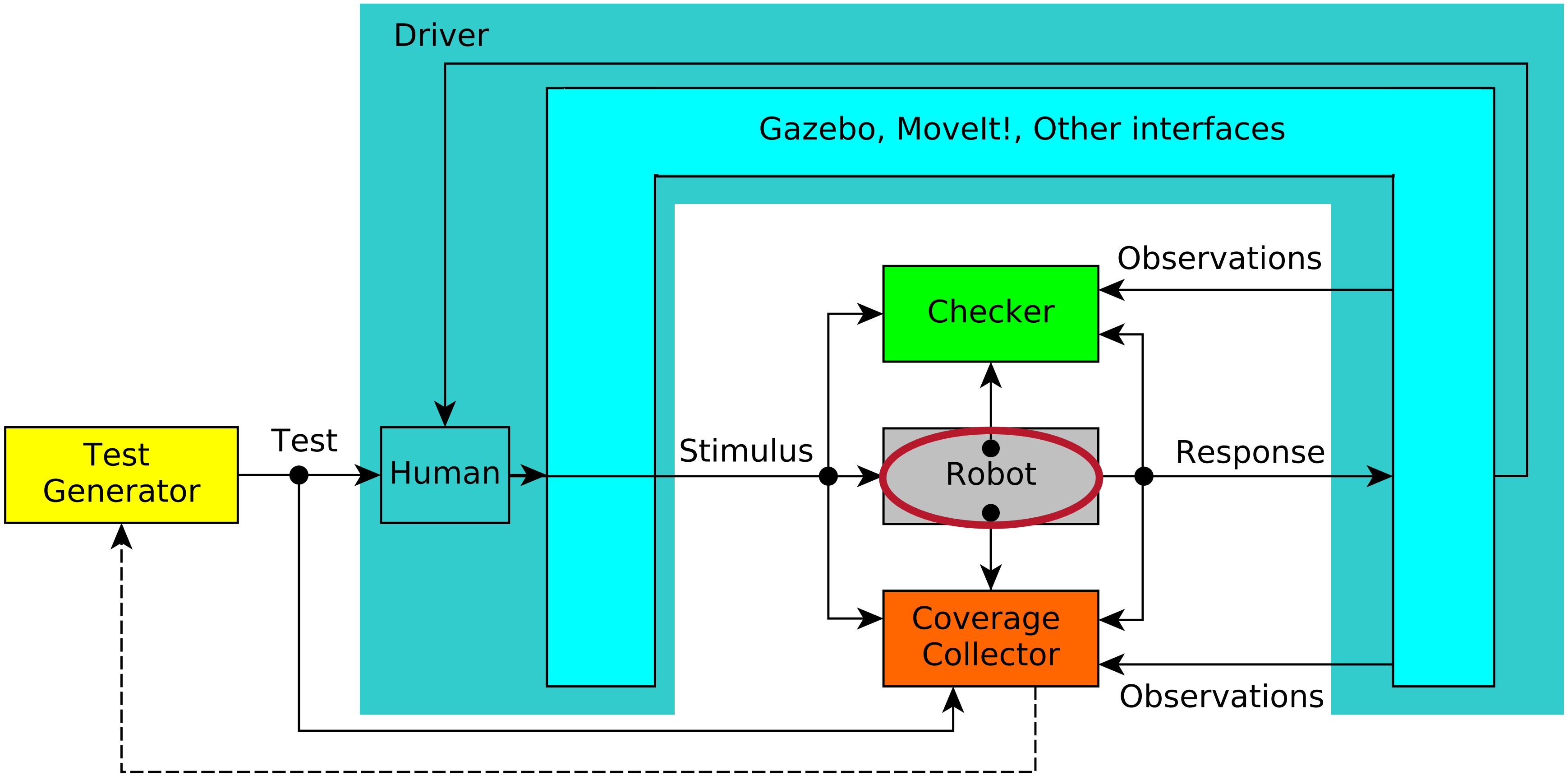


# CDV to *automate* simulation-based testing

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)

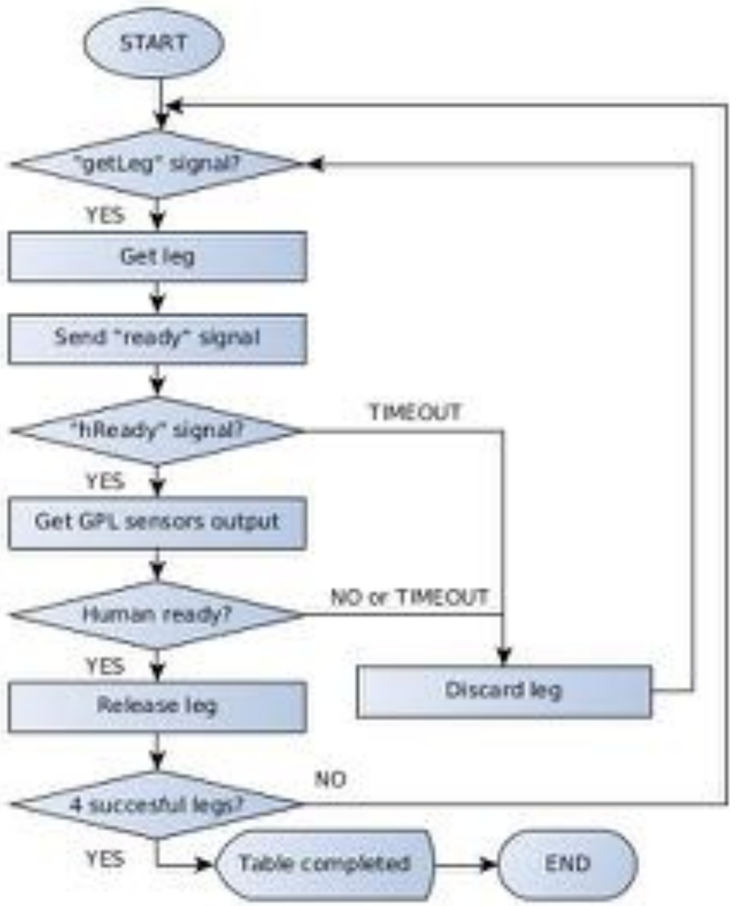
Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.
***Systematic and Realistic Testing in Simulation of Control Code for Robots in Collaborative Human-Robot Interactions.*** 17th Annual Conference Towards Autonomous Robotic Systems (TAROS 2016), pp. 20-32. Lecture Notes in Artificial Intelligence 9716. Springer, June 2016.
(DOI 10.1007/978-3-319-40379-3_3)

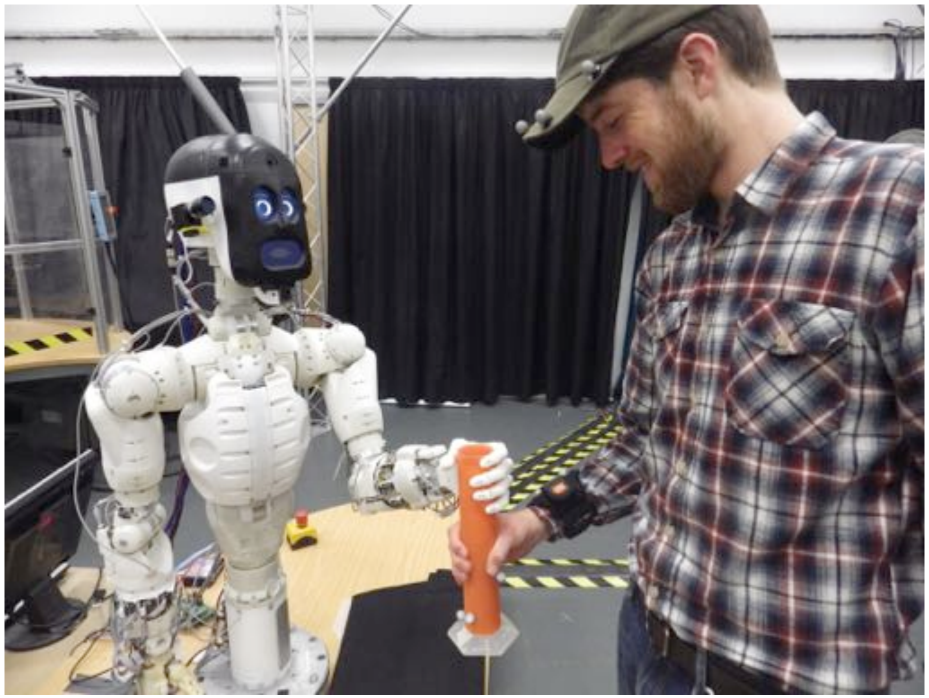
# Simulation-based testing




Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.
***Systematic and Realistic Testing in Simulation of Control Code for Robots in Collaborative Human-Robot Interactions***. 17th Annual Conference Towards Autonomous Robotic Systems (TAROS 2016), pp. 20-32. Lecture Notes in Computer Science 9716. Springer, June 2016. DOI 10.1007/978-3-319-40379-3_3

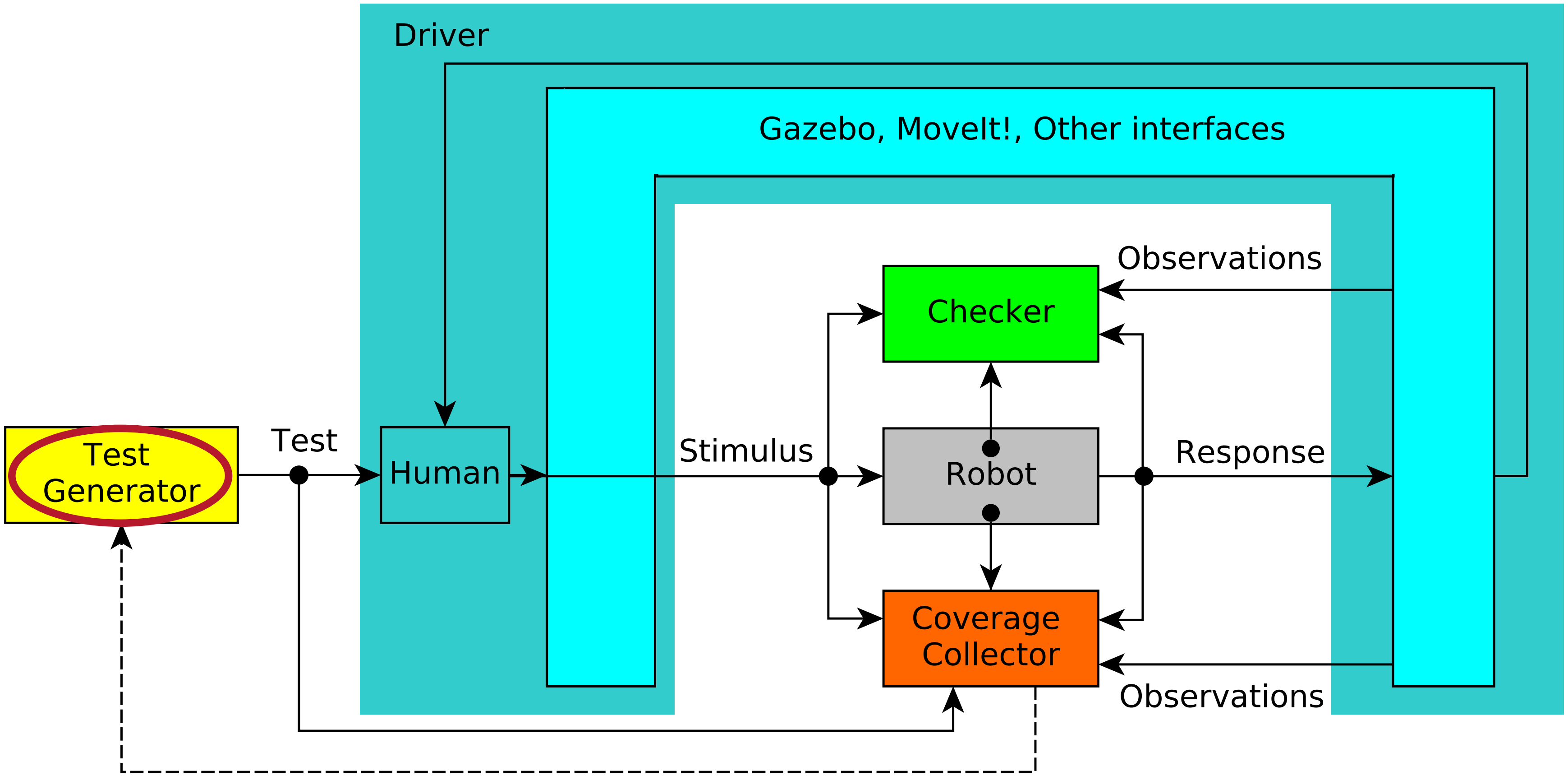
# Robotic code



J. Boren and S. Cousins, "The SMACH High-Level Executive"
IEEE Robotics & Automation Magazine, vol. 17, no. 4, pp. 18–20, 2010.

# Simulation-based testing




Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.
***Systematic and Realistic Testing in Simulation of Control Code for Robots in Collaborative Human-Robot Interactions***. 17th Annual Conference Towards Autonomous Robotic Systems (TAROS 2016), pp. 20-32. Lecture Notes in Computer Science 9716. Springer, June 2016. DOI 10.1007/978-3-319-40379-3_3

# Test Generator

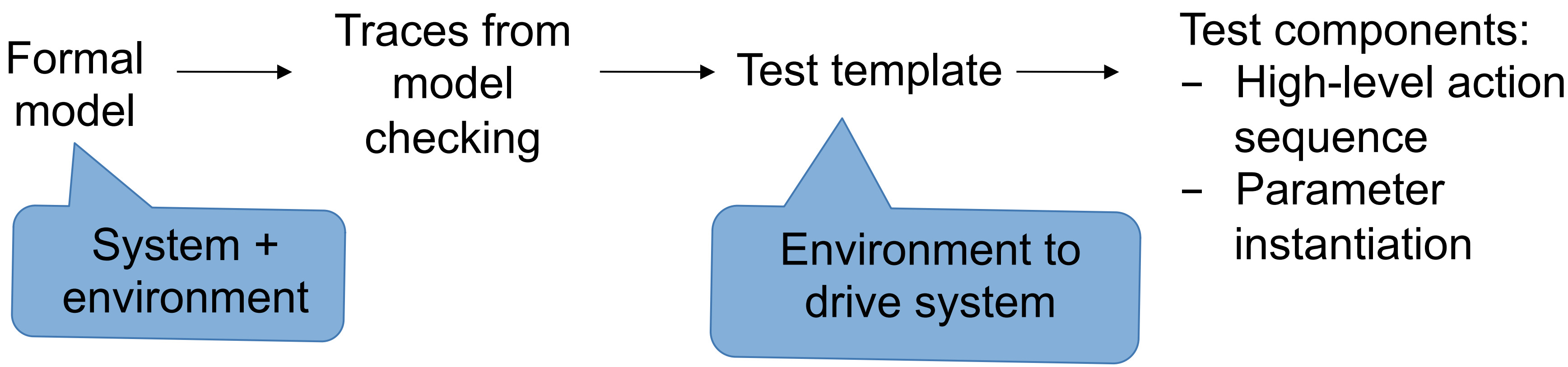
- Effective tests:
  - legal tests
  - meaningful events
  - interesting events
  - while exploring the system
    - typical vs extreme values
- Efficient tests:
  - minimal set of tests (regression)
- Strategies:
  - Pseudorandom (repeatability)

# 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 test generation



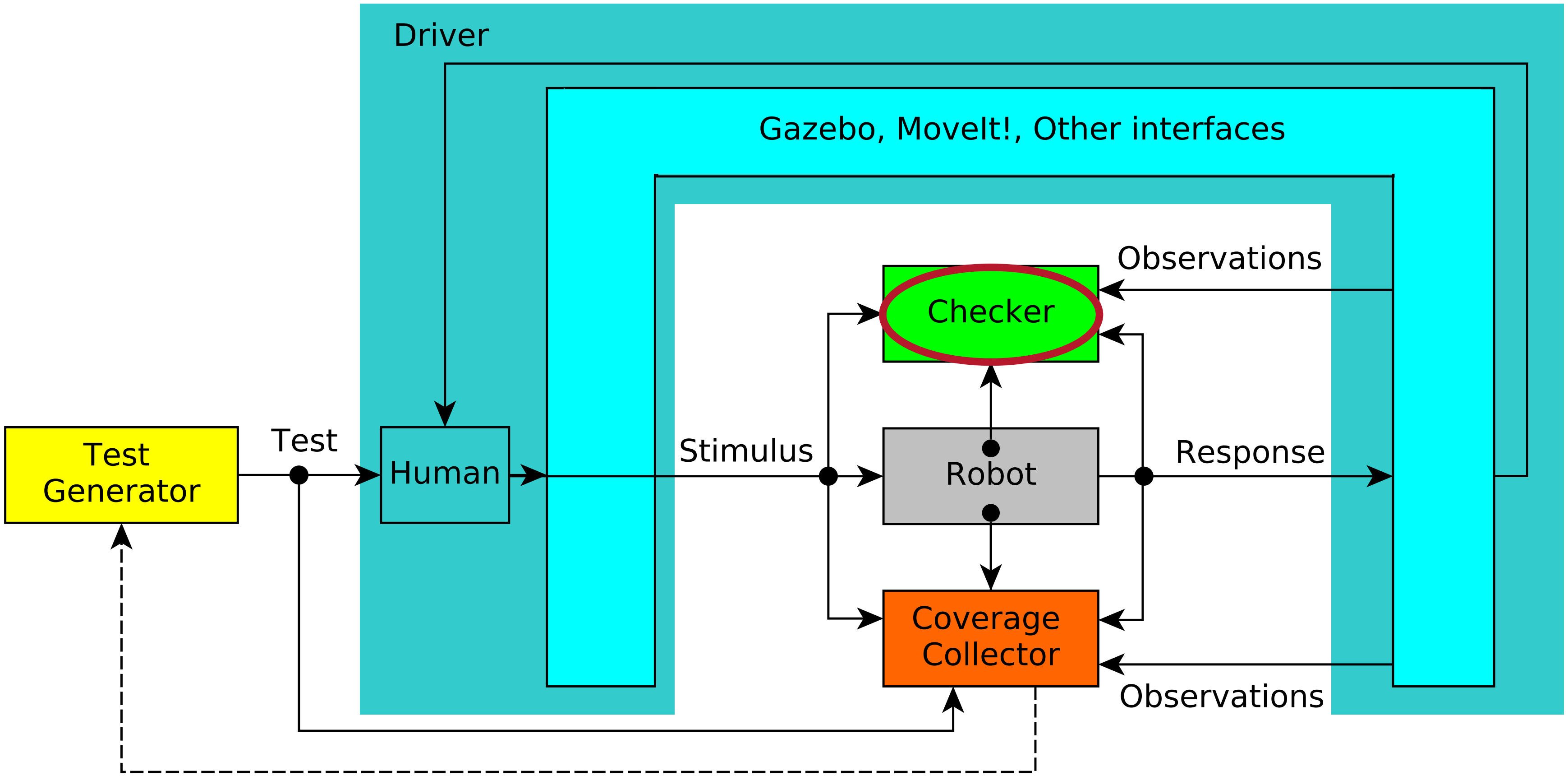
# Model-based test generation


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

System + environment

Environment to drive system

# Simulation-based testing




Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.
***Systematic and Realistic Testing in Simulation of Control Code for Robots in Collaborative Human-Robot Interactions***. 17th Annual Conference Towards Autonomous Robotic Systems (TAROS 2016), pp. 20-32. Lecture Notes in Computer Science 9716. Springer, June 2016. DOI [10.1007/978-3-319-40379-3_3](10.1007/978-3-319-40379-3_3)

# Checker

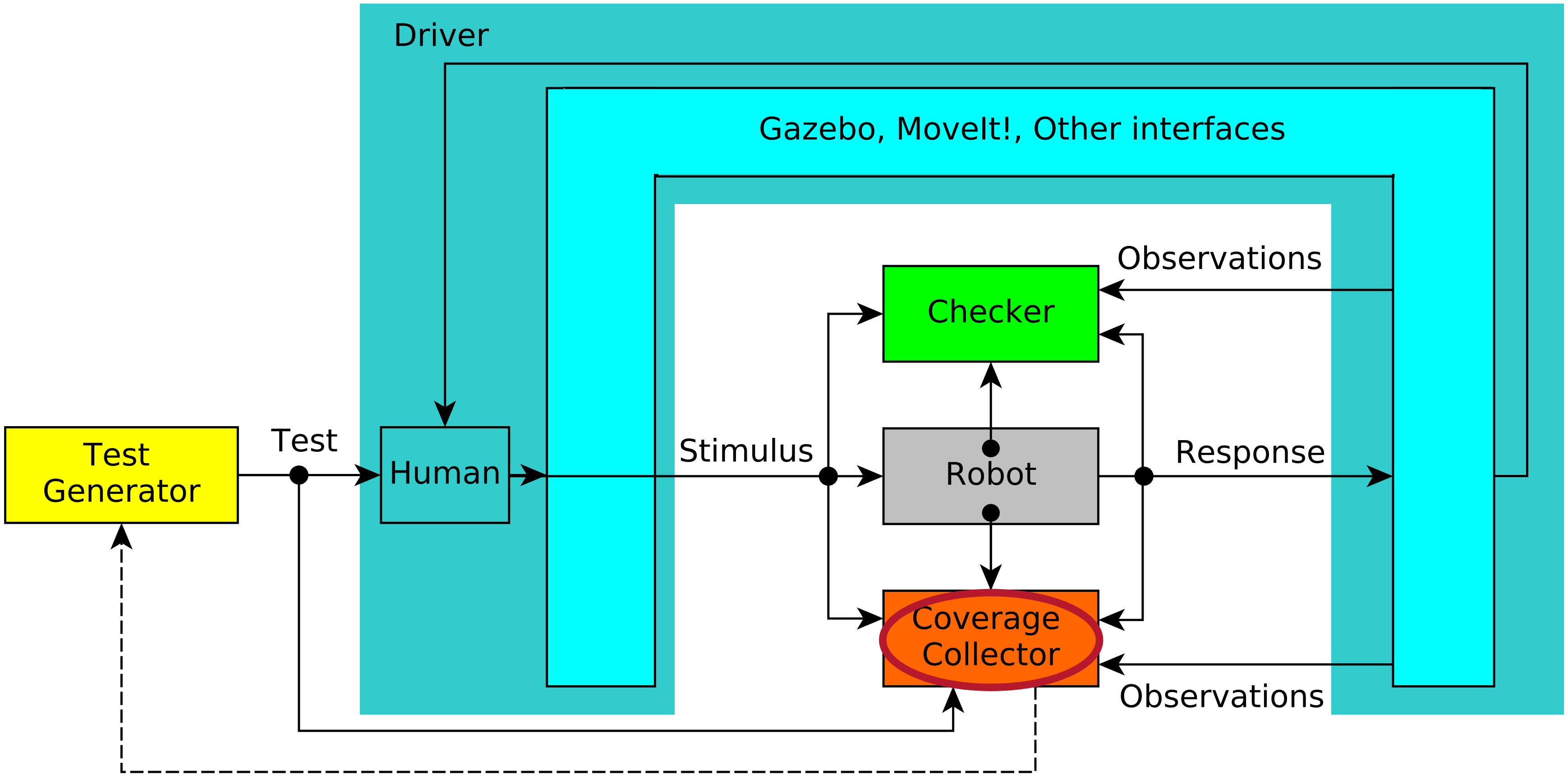
- Encode requirements as assertions:
  - `if [precondition], check [postcondition]`

    *"If the robot decides the human is not ready, then the robot never releases an object".*

  - Implemented as automata for monitoring
- 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)}
```

# Simulation-based testing





Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.
***Systematic and Realistic Testing in Simulation of Control Code for Robots in Collaborative Human-Robot Interactions***. 17th Annual Conference Towards Autonomous Robotic Systems (TAROS 2016), pp. 20-32. Lecture Notes in Computer Science 9716. Springer, June 2016. DOI 10.1007/978-3-319-40379-3_3

# Coverage

- **Code coverage**
- **Structural coverage**, e.g. of the FSM(s)
- **Functional coverage**
  - Requirements coverage
    - Functional and safety (ISO 13482:2014, ISO 10218-1)

Robot to human object
handover scenario

Robot to human object
handover scenario

# Requirements inspired by 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 inspired by 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 inspired by ISO 13482 and ISO 10218

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

- 8a The robot hand speed is always less than 250 mm/s.
- 8b If the robot is within 10 cm of the human, the robot's hand speed is less than 250 mm/s.
- 8c If the robot collides with anything, the robot's hand speed is less than 250 mm/s.
- 8d If the robot collides with the human, the robot's hand speed is less than 250 mm/s.

# Coverage

- **Code coverage**
- **Structural coverage**, e.g. of the FSM(s)
- **Functional coverage**
  - Requirements coverage
    - Functional and safety (ISO 13482:2014, ISO 10218-1)
  - Situation coverage (cross-product coverage) based on gaze, pressure and hand location sensor data

# Coverage

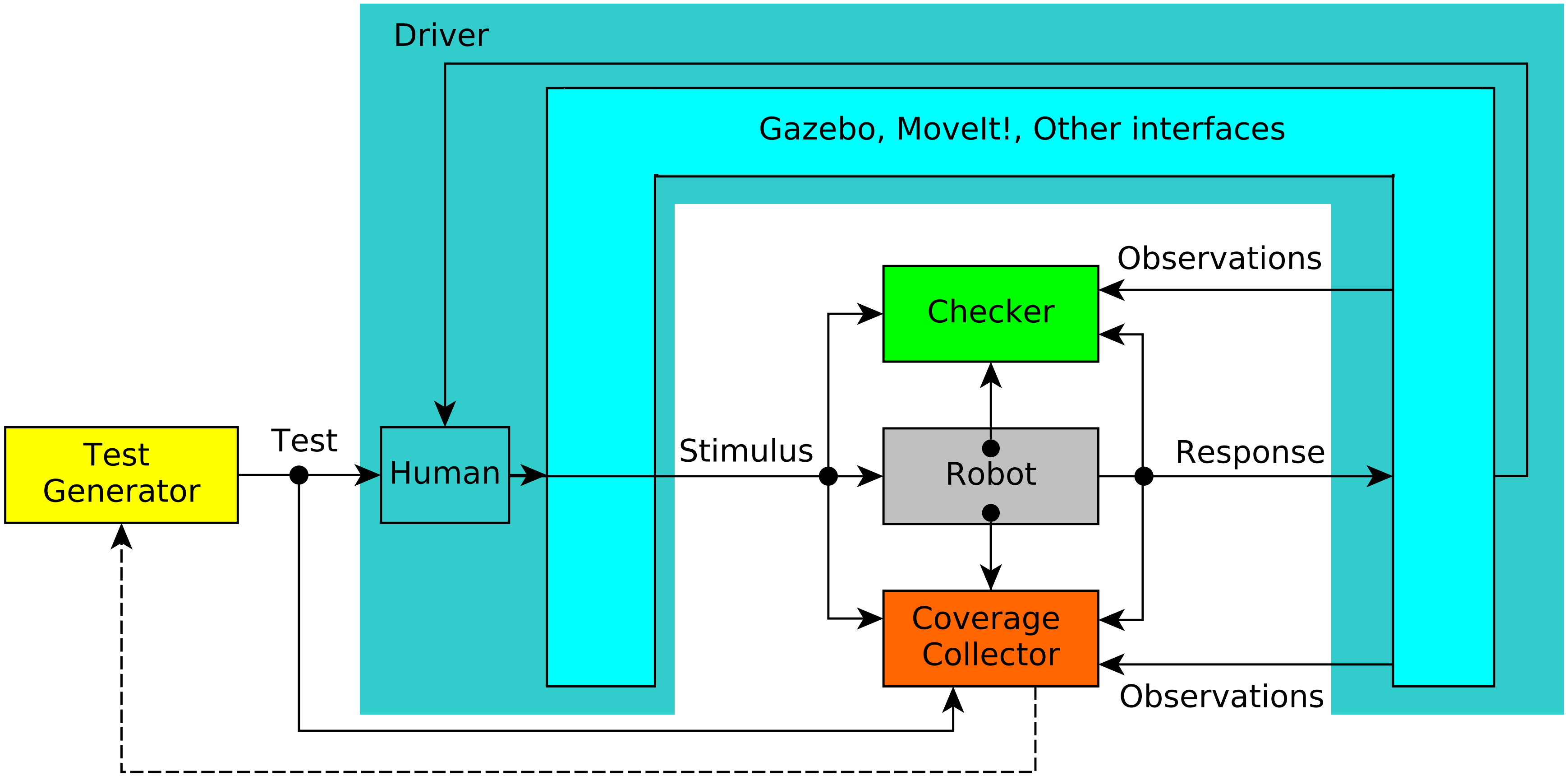
- **Code coverage**
- **Structural coverage**, e.g. of the FSM(s)
- **Functional coverage**
  - Requirements coverage
    - Functional and safety (ISO 13482:2014, ISO 10218-1)
  - Situation coverage (cross-product coverage) based on gaze, pressure and hand location sensor data
  - SOTIF

    (ISO/PAS 21448:2019)

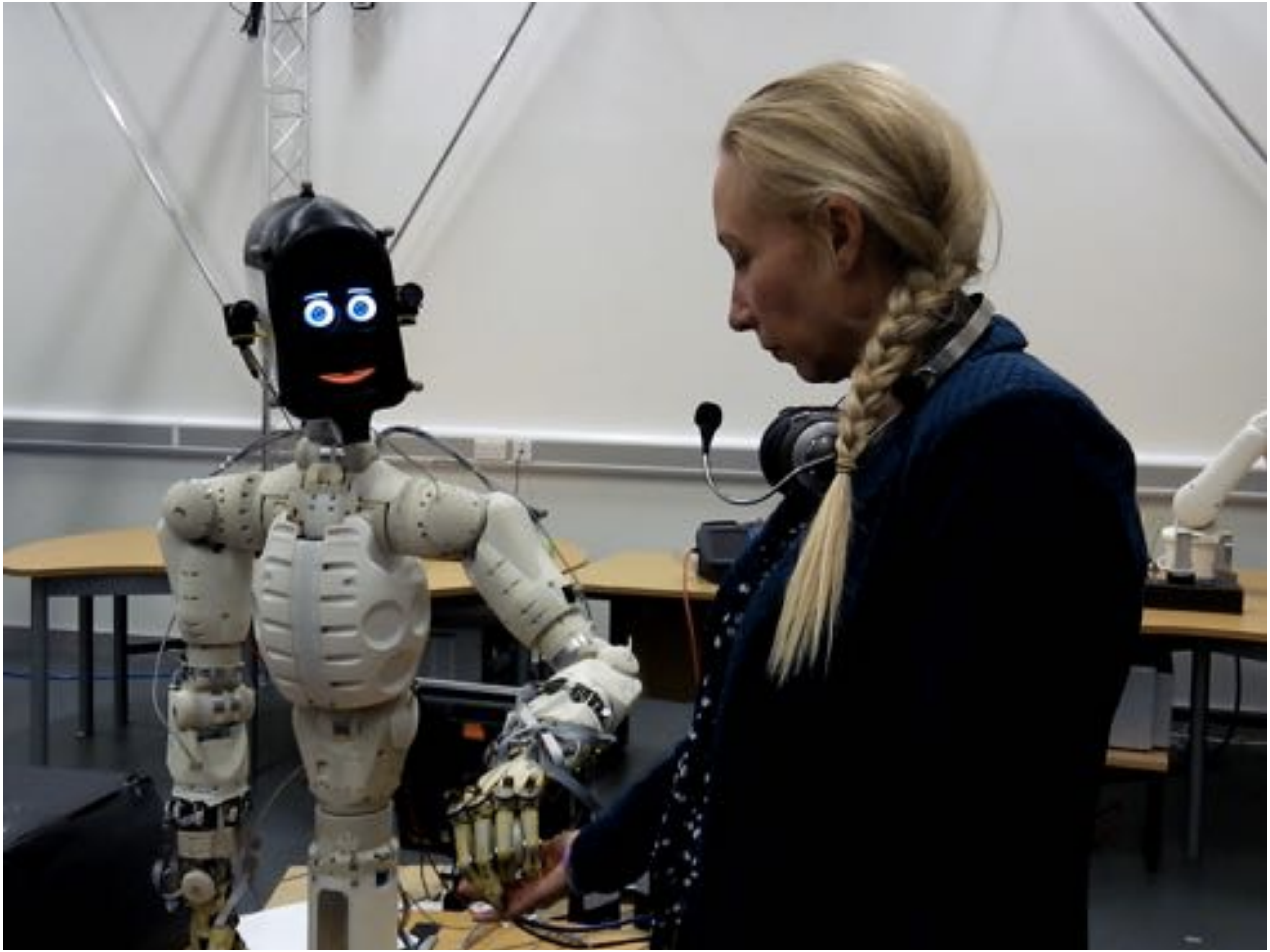
    Road vehicles, Safety of the intended functionality

| (Gaze, Pressure, Location) | Sense timeout | Release piece | No release | | |
|---|---|---|---|---|---|
| $(\bar{1}, \bar{1}, \bar{1})$ | | | | | |
| $(\bar{1}, \bar{1}, 1)$ | | | | | |
| $(\bar{1}, 1, \bar{1})$ | | | | | |
| $(\bar{1}, 1, 1)$ | | | | | |
| $(1, \bar{1}, \bar{1})$ | | | | | |
| $(1, \bar{1}, 1)$ | | | | | |
| $(1, 1, \bar{1})$ | | | | | |
| $(1, 1, 1)$ | | | | | |
| Action | Sense timeout | Release piece | No release | Signal 1 timeout | Signal 2 timeout |
| No activation | | | | | |
| Activation signal 1 | | | | | |
| Not on task | | | | | |

# CDV for Human-Robot Interaction

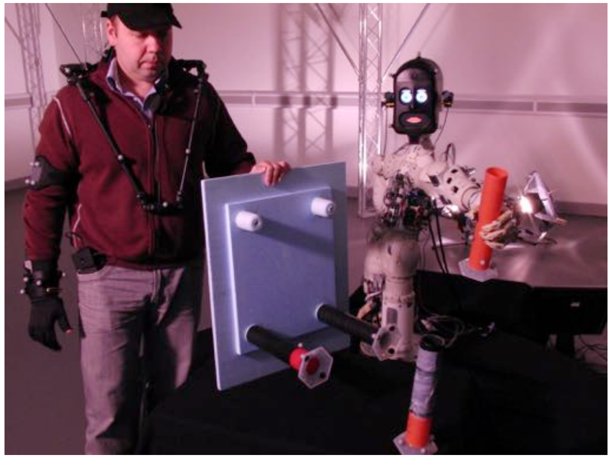


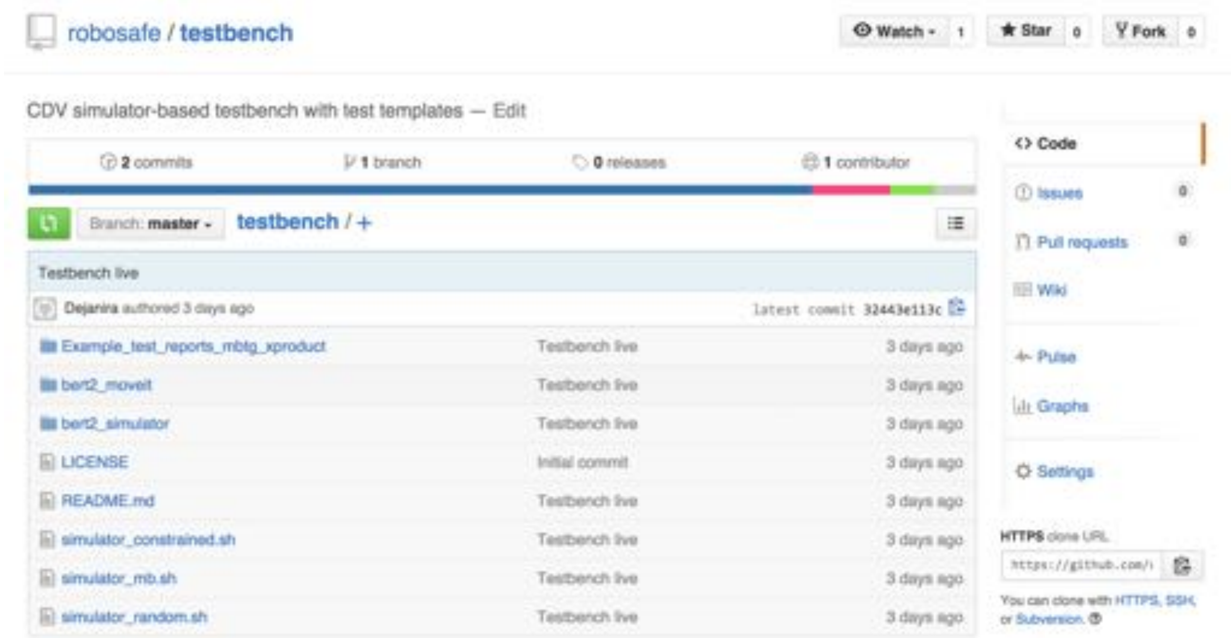

Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.
***Systematic and Realistic Testing in Simulation of Control Code for Robots in Collaborative Human-Robot Interactions***. 17th Annual Conference Towards Autonomous Robotic Systems (TAROS 2016), pp. 20-32. Lecture Notes in Computer Science 9716. Springer, June 2016. DOI [10.1007/978-3-319-40379-3_3](10.1007/978-3-319-40379-3_3)



# 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






# [http://github.com/robosafe/testbench](http://github.com/robosafe/testbench)

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)

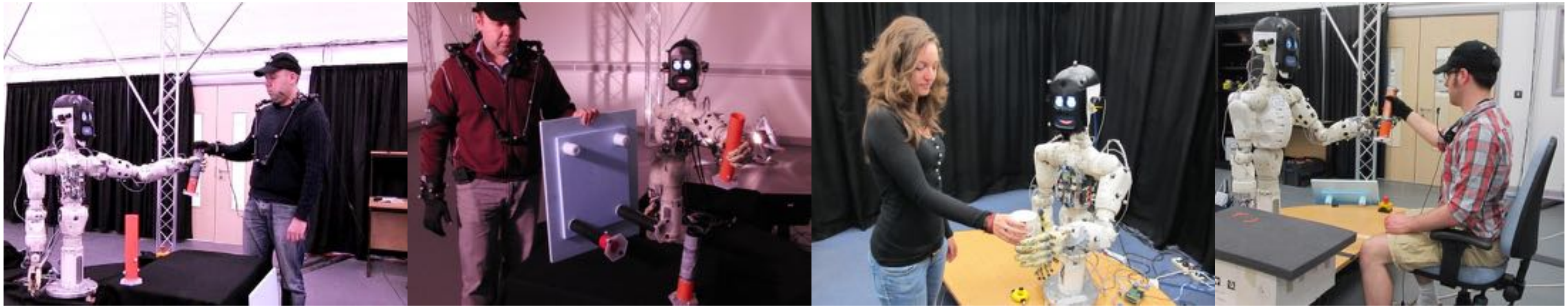
Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.
***Systematic and Realistic Testing in Simulation of Control Code for Robots in Collaborative Human-Robot Interactions***. 17th Annual Conference Towards Autonomous Robotic Systems (TAROS 2016), pp. 20-32. Lecture Notes in Artificial Intelligence 9716. Springer, June 2016. (DOI: 10.1007/978-3-319-40379-3_3)

# CDV provides *automation*

# What about *agency*?

# 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.



```
// INITIAL BELIEFS
preparing_for_flight.
      inititalising_systems .
     ~hardware_system_passed_test .
     ~has_read_flight_environment_model .
     ~has_read_new_flight_path .
     ~pilot_comms_work .
     ~all_beacon_comms_work .
     ~created_fligh_path_execution_plan .
     ~plan_is_unsafe_for_energy_level_available(Flight) .
     ~announced_text_object .
~ready_for_mission .
~on_ground_before_flight .
~ground_testing
     ~responded_to_take_off_permission .
     ~permission_given_for_take_off .
~flying .
~take_off_testing
     ~there_is_flight_system_weakness_to_report
     ~responded_to_start_mission .
~on_mission .
~people_pause .
~vehicle_pause .
~flying_pause .
~avoiding_behaviour .
~power_return .
~emergency_Landing .
~in_manual_control .
~landed .

//Environment Events and States

     ~people_appearing .
     ~vehicles_appearing .
     ~flying_object_appearing .
     ~weather_too_bad .
     ~visibility_too_bad .
     ~onboard_faults .
     ~command_received .
     ~manual_control_request .
```


```

// EXECUTABLE PLANS
// executable plan :
+ask_permission_to_take_off :
ready_for_mission <-
invoke(comms,runOnce,asking_for_permission,["take off"],[]) .

// executable plan :
+there_is_flight_system_weakness_to_report :
ready_for_mission & ~received_take_off_permission <-
invoke(comms,runOnce,announcing_text_object,["R"],[]) .

// executable plan :
+there_is_flight_system_weakness_to_report :
ready_for_mission  & ~received_take_off_permission <-
invoke(comms,runOnce,announcing_text_object,["R"],[]);
invoke(comms,runOnce,asking_for_permission,["start mission"],[]) .

// executable plan :
+new_commands_has_arrived(Com) :
ready_for_mission  | on_mission <-
invoke(comms,runOnce,interpreting_commands,["Com"],["Txt"]);
// * interpreted commands,  commands_unclear,  did not yet acknowledge all commands  */
-new_commands_has_arrived(Com) .

// executable plan :
+interpreted_commands :
ready_for_mission & not_yet_acknowledge_all_commands <-
invoke(comms,runOnce,announcing_text_object,["Txt"],[]);
// * announced text object, did not hear my text object */
invoke(comms,runOnce,waiting_for_pilot_response,["Txt","~20s"],[]) .
// *  approval timed out,  pilot approved take off, pilot disapproved */

// executable plan :
+pilot disapproved  :
ready_for_mission  <-
invoke(comms,runOnce,announcing_constant_text,["Please repeat your instruction."] []).
```

# Belief-Desire-Intention Agents

```
//Initial beliefs
//Initial goals
!reset.
//Plans
+!reset :  true <- add_time(20);.print("Robot is resetting");!waiting.
+!waiting : not leg <- .print("Waiting"); !waiting.
+!waiting : leg <- add_time(40);.print("You asked for leg");-leg[source(human)];!grabLeg.
...
```

Desires: goals to fulfil

Beliefs: knowledge about the world

Guards for plans

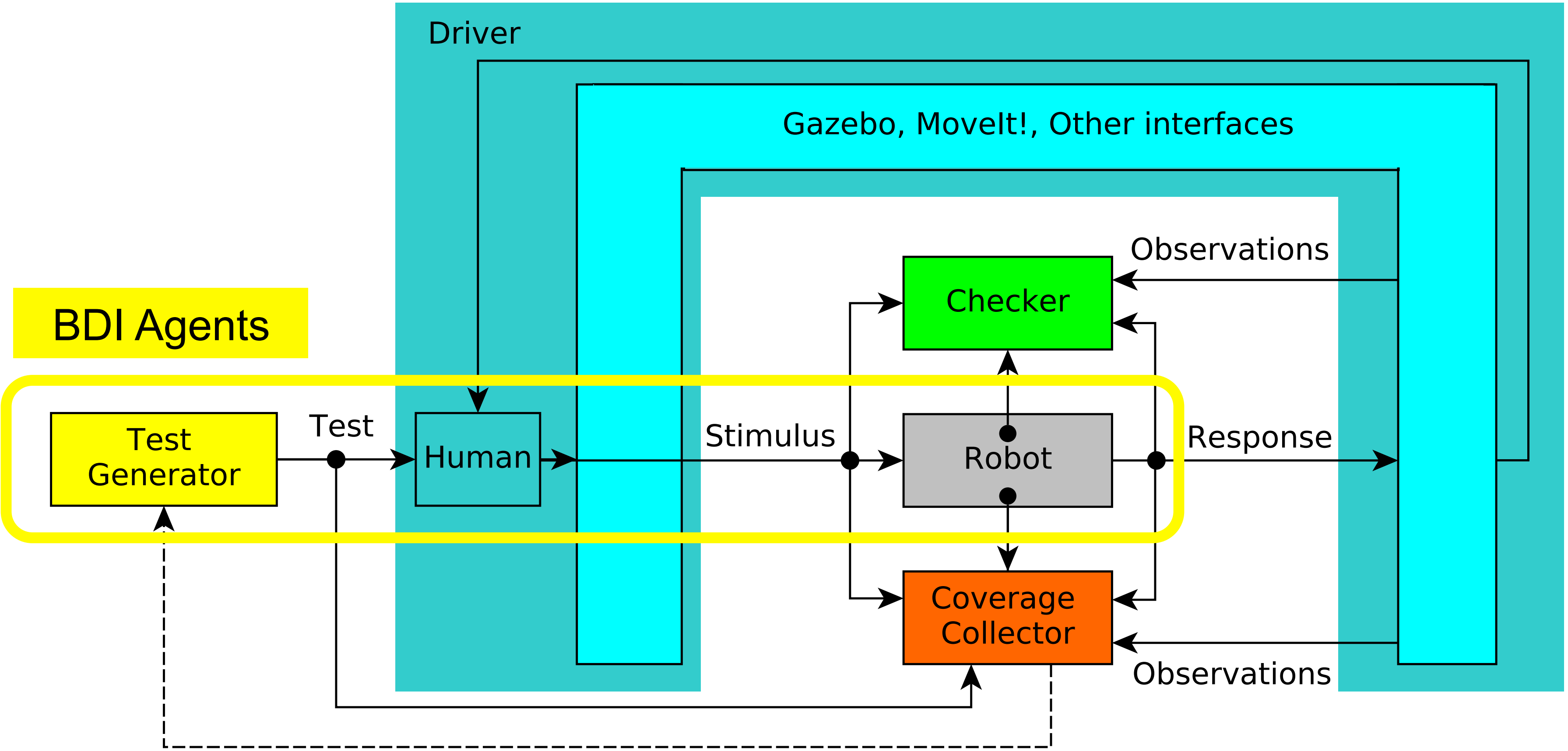
Intentions: chosen plans, according to current beliefs and goals

New beliefs

New goals

From executing plans

# CDV testbench components



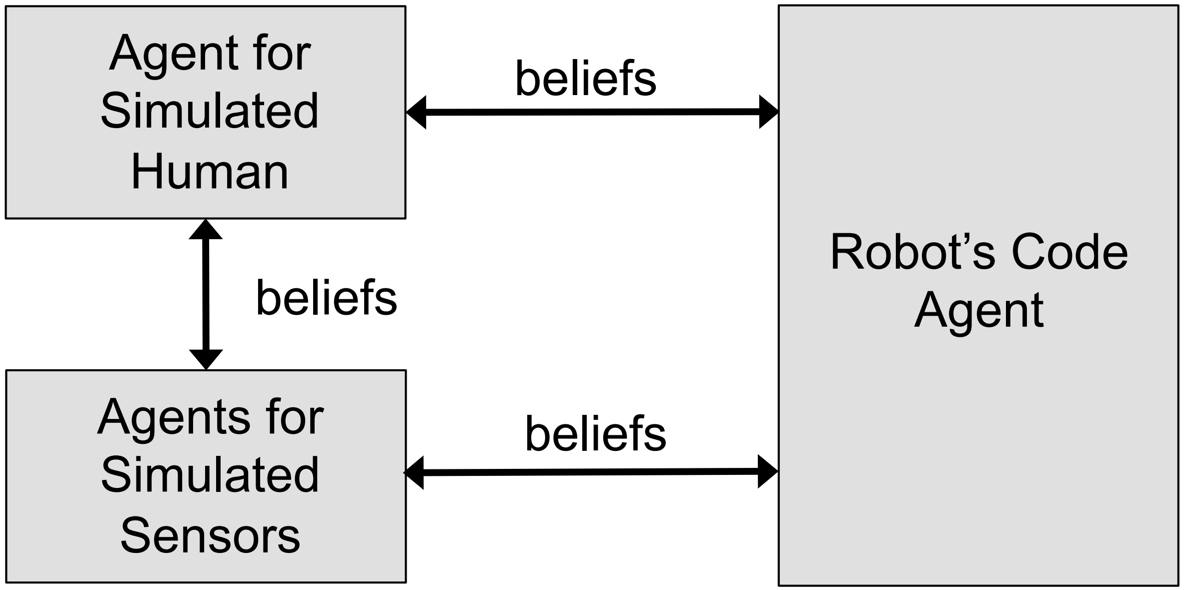

Intelligent testing is harnessing the power of BDI agent models to introduce agency into test environments.

# 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?

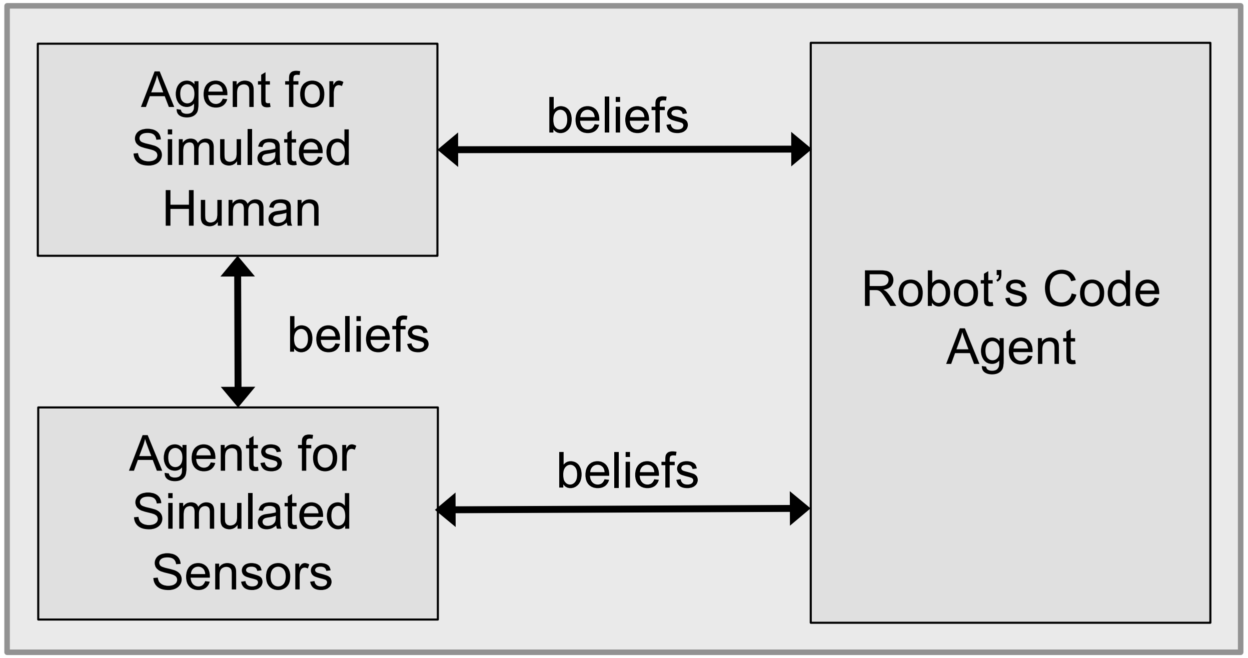
# Interacting Agents

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



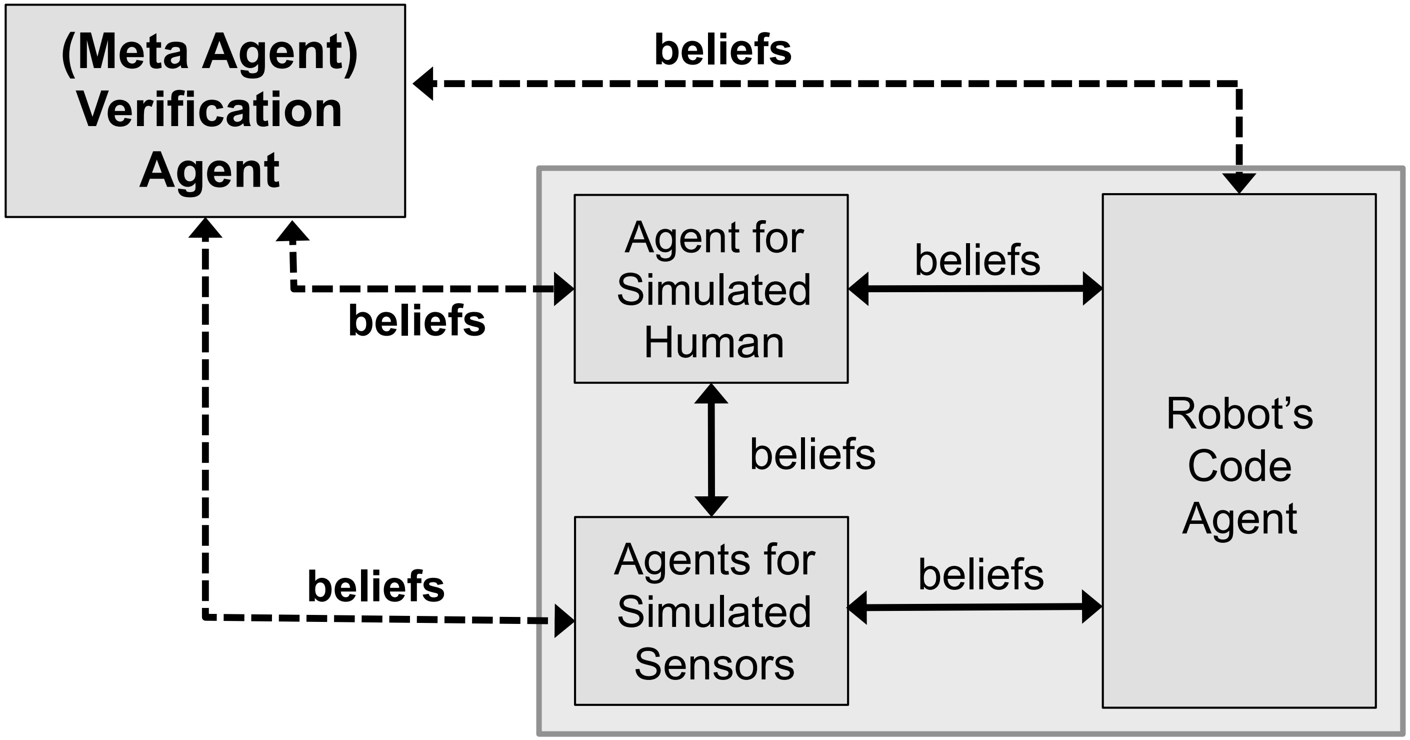
# Interacting Agents

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

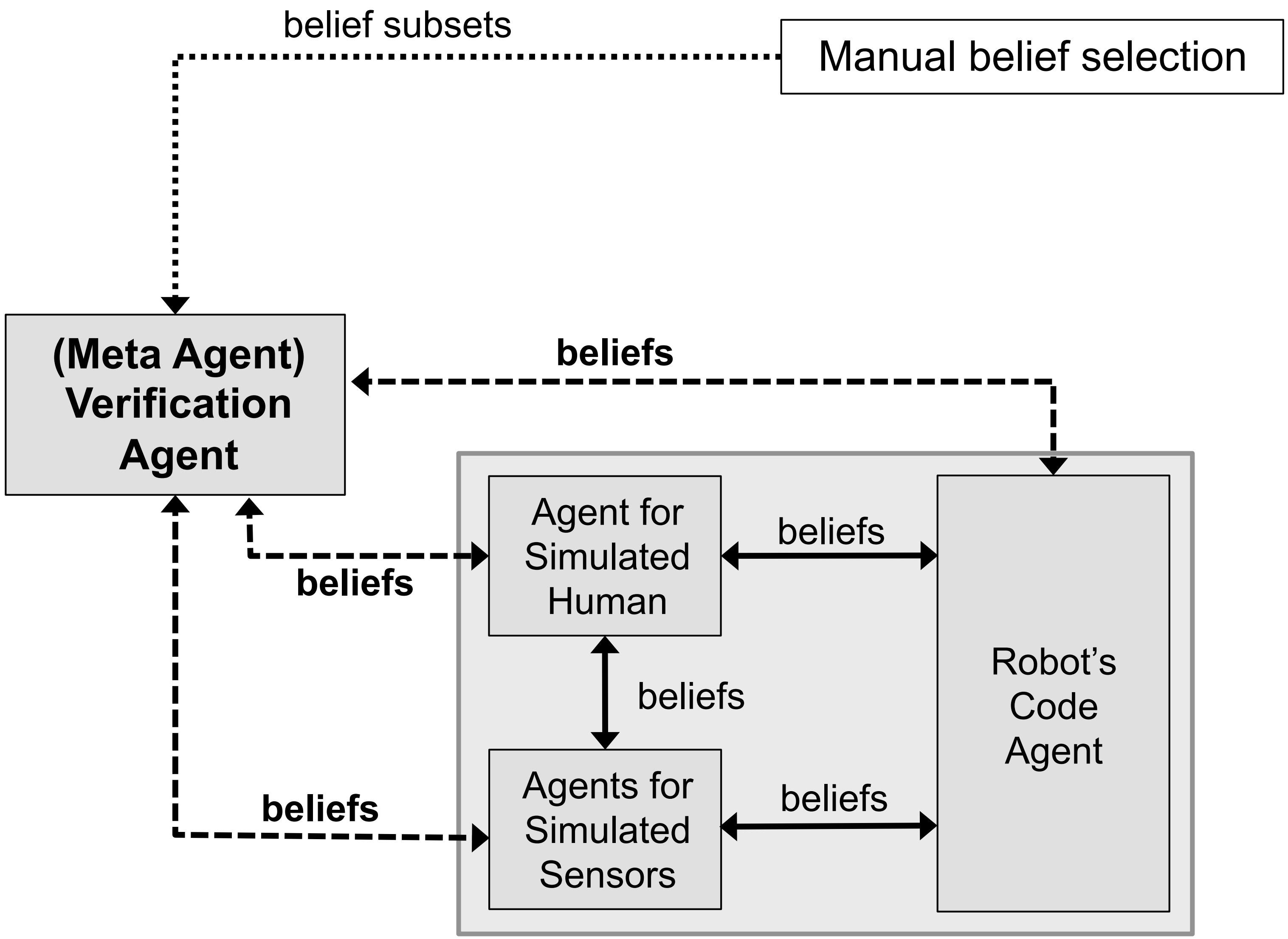


# Verification Agents

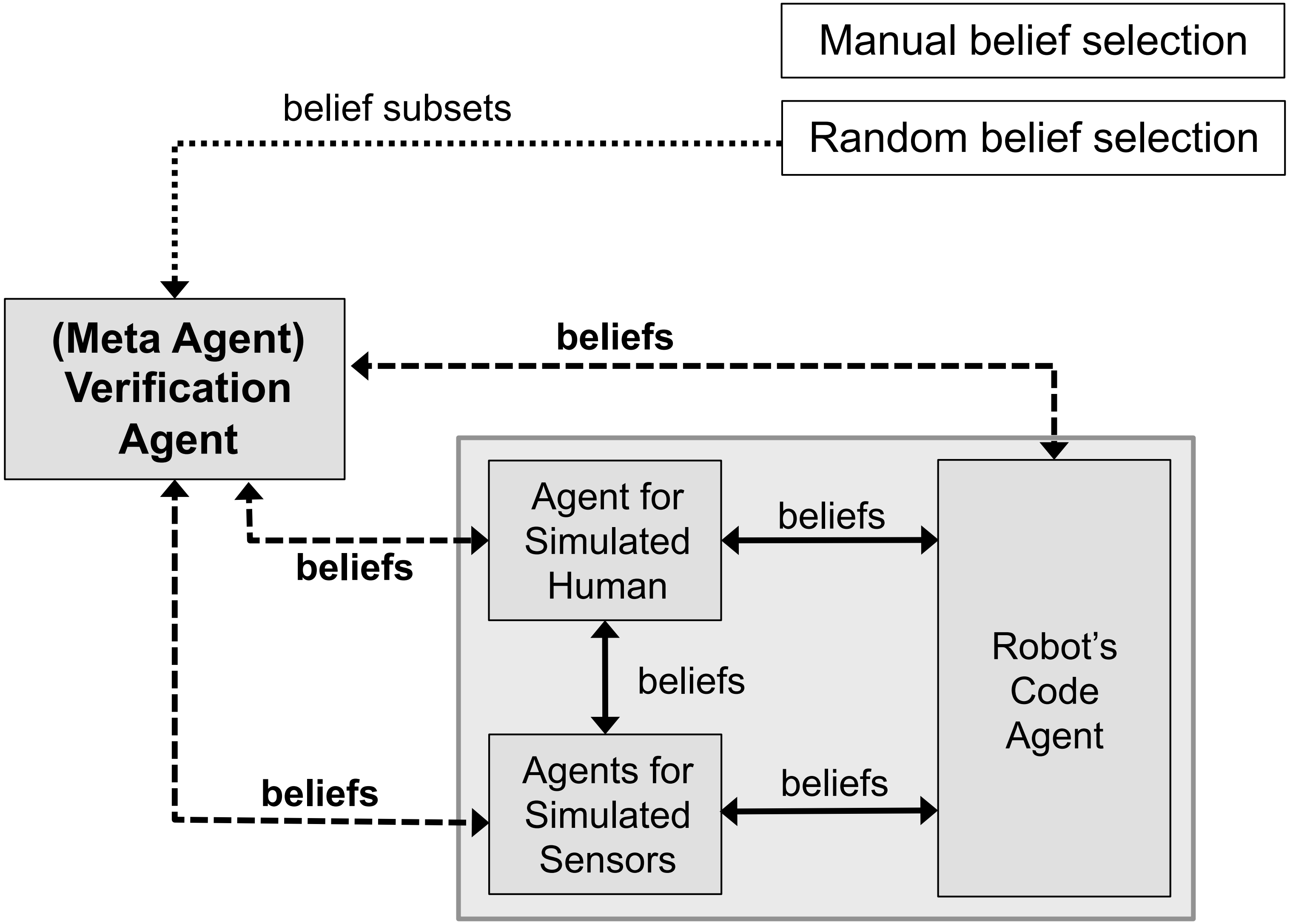
- Meta agents can influence beliefs
- This allows biasing/directing the interactions



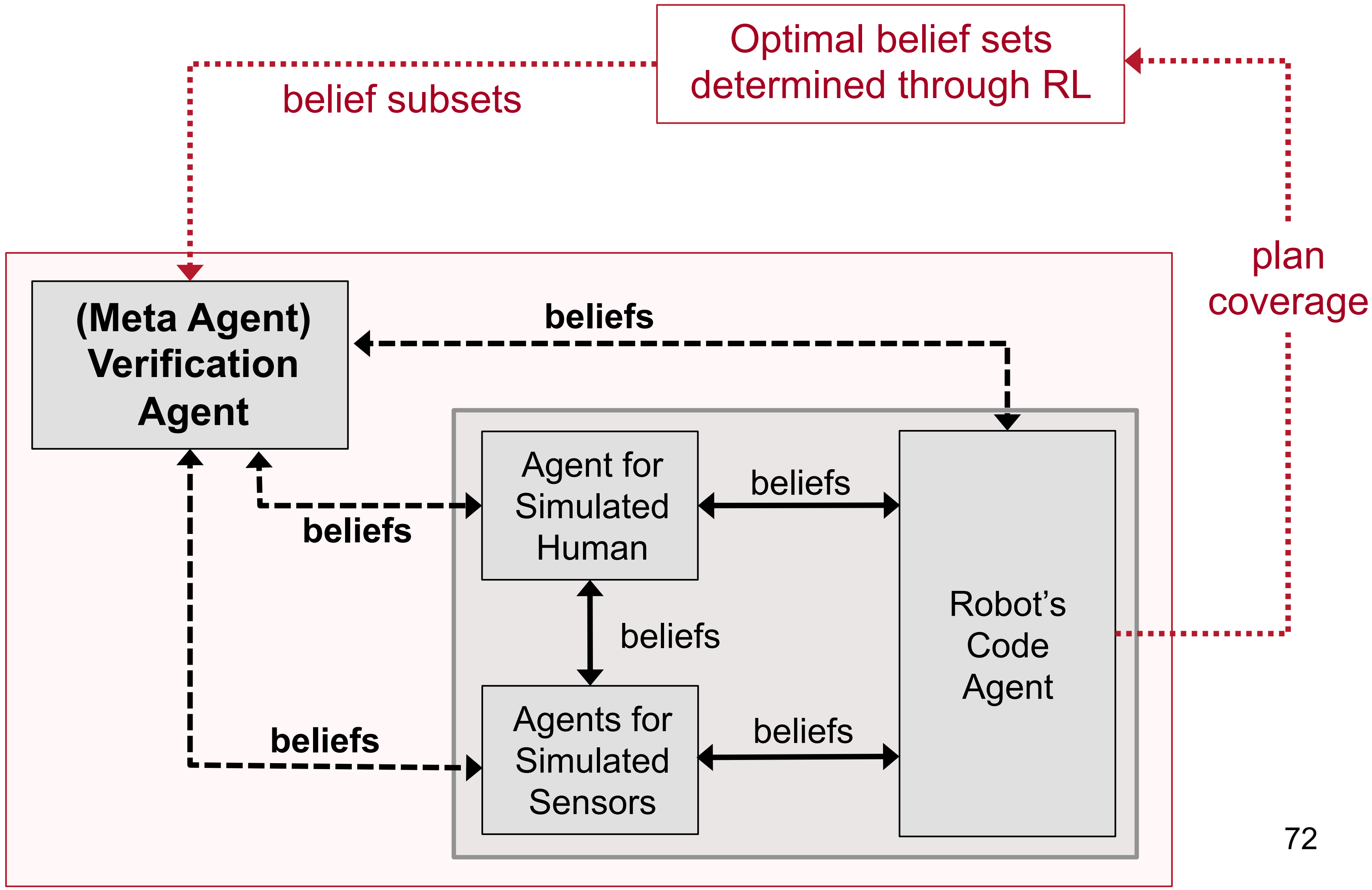
# Which beliefs are effective?

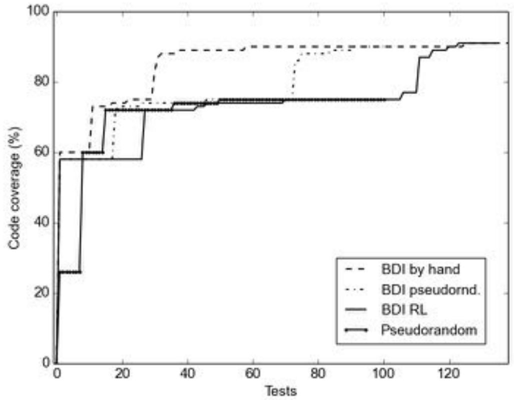


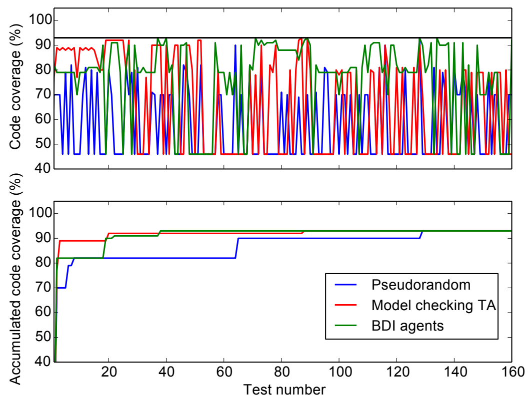
# Which beliefs are effective?



# Which beliefs are effective?





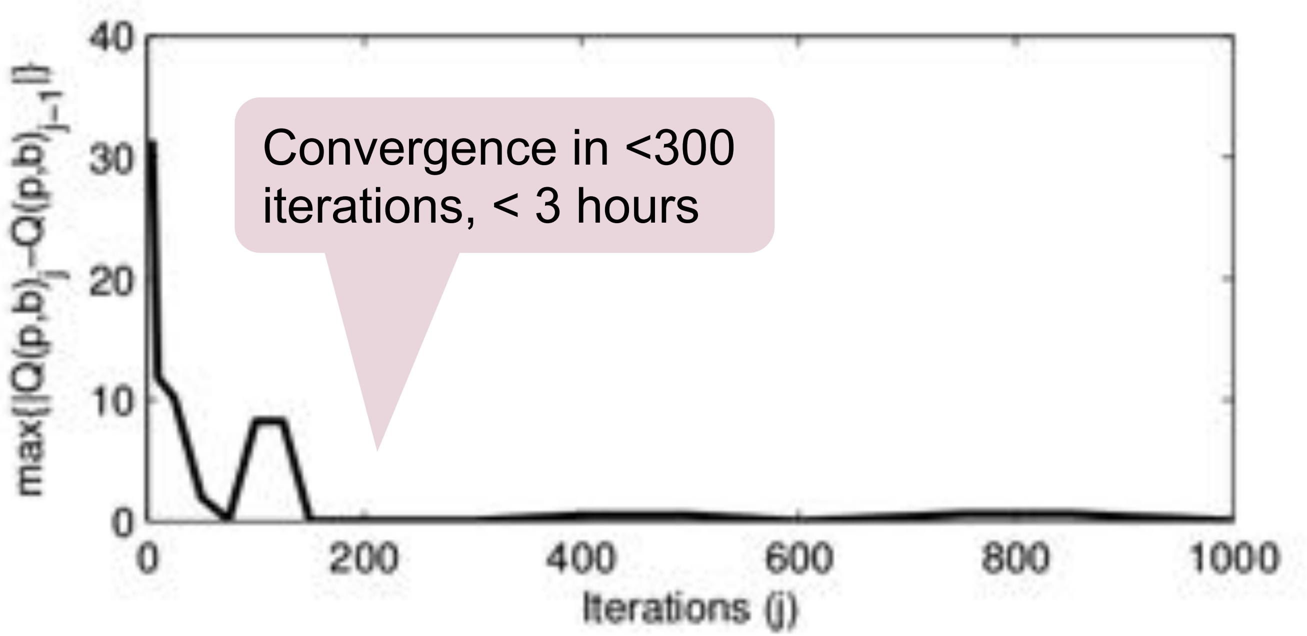

# Results

## 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. **Intelligent Agent-Based Stimulation for Testing Robotic Software in Human-Robot Interactions.** (Proceedings of MORSE 2016, ACM, July 2016) DOI: 10.1145/3022099.3022101 (arXiv:1604.05508)

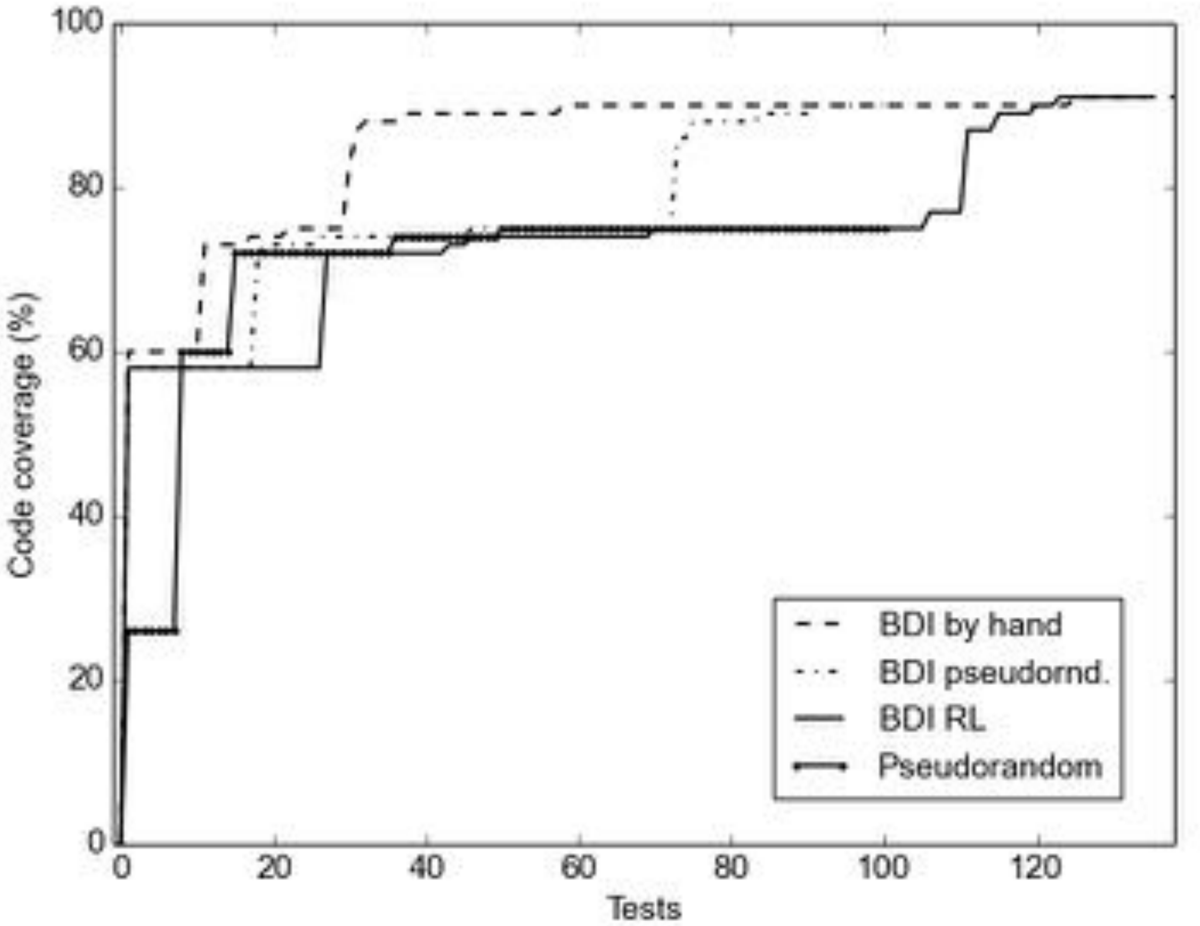
D. Araiza-Illan, A.G. Pipe, K. Eder
**Model-based Test Generation for Robotic Software: Automata versus Belief-Desire-Intention Agents.** (under review, preprint available at arXiv:1609.08439)

# The cost of learning belief sets

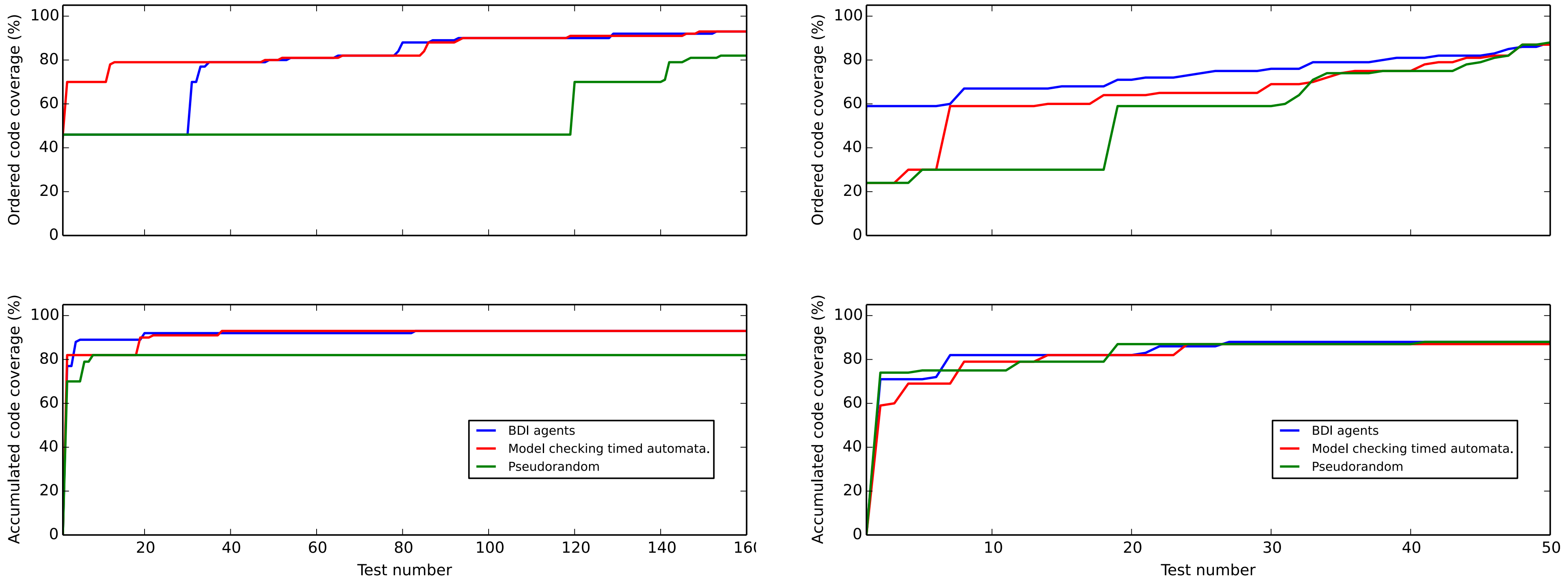



The cost of learning a good belief set needs to be considered when assessing the different BDI-based test generation approaches.

# Code Coverage Results



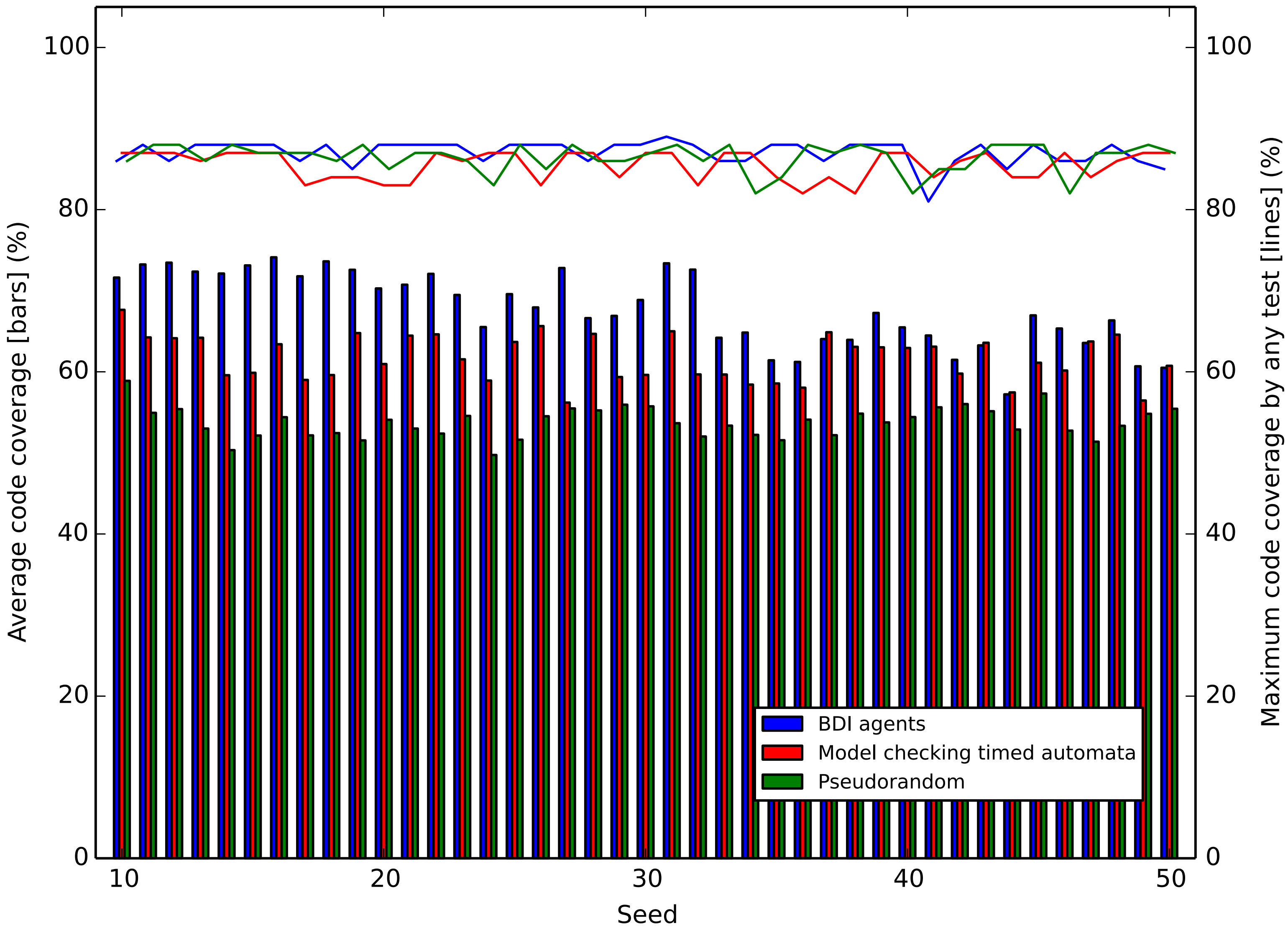
# BDI-agents vs timed automata




# Effectiveness:

- high-coverage tests are generated quickly

# BDI-agents vs timed automata



# BDI-agents vs timed automata

| | Model checking timed automata | BDI agents |
|---|---|---|
| Cooperative Manufacturing Assistant | | |
| Model's lines of code | 725 | 348 |
| Number of states (transitions) or plans | 53 (72) | 79 |
| Modelling time | ≈ 10.5 hrs | ≈ 6 hrs |
| Model exploration time (min/test) | 0.001 s | 5 s |
| Model exploration time (max/test) | 33.36 s | 5 s |
| Home Care Assistant | | |
| Model's lines of code | 722 | 131 |
| Number of states (transitions) or plans | 42 (67) | 35 |
| Modelling time | ≈ 5.5 hrs | ≈ 3 hrs |
| Model exploration time (min/test) | 0.001 s | 1 s |
| Model exploration time (max/test) | 2.775 s | 1 s |

# 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**.

robosafe

http://github.com/robosafe

D. Araiza Illan, D. Western, A. Pipe, K. Eder.
Coverage-Driven Verification - An approach to verify code for robots that directly interact with humans. (Proceedings of HVC 2015, Springer, November 2015)

D. Araiza Illan, D. Western, A. Pipe, K. Eder.
Systematic and Realistic Testing in Simulation of Control Code for Robots in Collaborative Human-Robot Interactions.
(Proceedings of TAROS 2016, Springer, June 2016)

D. Araiza-Illan, A.G. Pipe, K. Eder.
**Intelligent Agent-Based Stimulation for Testing Robotic Software in Human-Robot Interactions.** (Proceedings of MORSE 2016, ACM, July 2016)
DOI: **10.1145/3022099.3022101** (arXiv:1604.05508)

D. Araiza-Illan, A.G. Pipe, K. Eder
**Model-based Test Generation for Robotic Software: Automata versus Belief-Desire-Intention Agents.** (under review, preprint available at **arXiv:1609.08439**)

# Challenges for RAS V&V

- Specification: vague and probabilistic

  J. Morse, D. Araiza-Illan, J. Lawry, A. Richards, K. Eder
  **A Fuzzy Approach to Qualification in Design Exploration for Autonomous Robots and Systems.** https://arxiv.org/abs/1606.01077
  (Proceedings of IEEE International Conference on Fuzzy Systems Fuzz-IEEE 2017)

- Automation, automation, automation
- Combination of techniques
- More AI for V&V, ... **we* need to be more clever*
  - Intelligent agent-based test generation:
    - a step towards online testing of learning machines
    - testing games between verification agents and robots

# Thank you

Kerstin.Eder@bristol.ac.uk

*Special thanks to Dejanira Araiza Illan, Jeremy Morse, David Western, Greg Chance, Abanoub Ghobrial, Arthur Richards, Jonathan Lawry, Trevor Martin, Clare Dixon, Michael Fisher, Matt Webster, Kerstin Dautenhahn, Maha Salem, Piotr Trojanek, Yoav Hollander, Yaron Kashai, Mike Bartley,* Séverin Lemaignan, *Tony Pipe and Chris Melhuish for their collaboration, contributions, inspiration and the many productive discussions we have had.*

University of BRISTOL

EPSRC
Engineering and Physical Sciences Research Council

Innovate UK

brl
Bristol Robotics Laboratory