Gaining Confidence in the Correctness of Robotic and Autonomous Systems

Kerstin Eder

Trustworthy Systems Laboratory

Verification and Validation for Safety in Robots, Bristol Robotics Laboratory

University of BRISTOL

EPSRC Engineering and Physical Sciences Research Council

Innovate UK

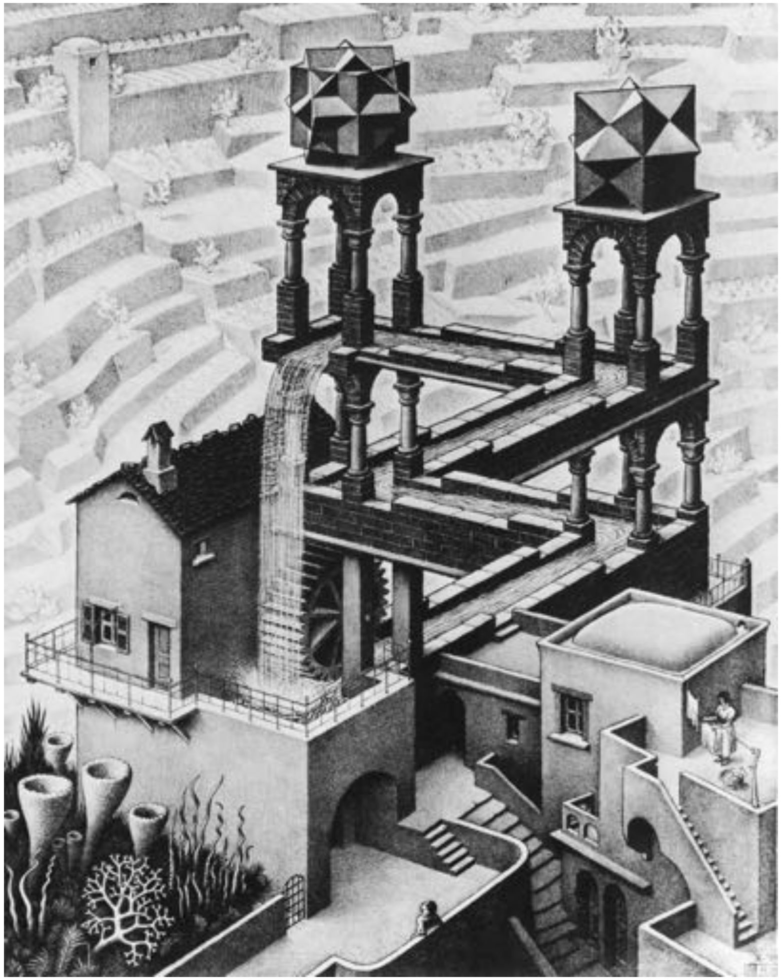
brl Bristol Robotics Laboratory

Designing Trustworthy Systems

- Create *flawless* systems.

AND

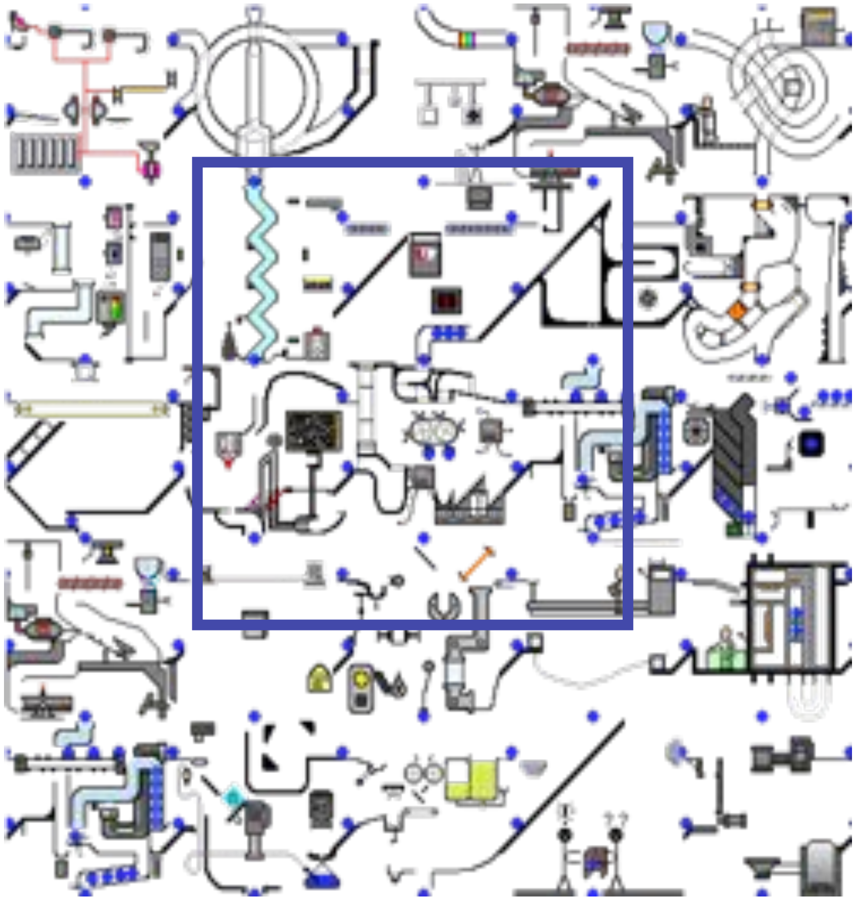
- Design these systems in such a way that the *flawlessness* can be demonstrated.



"Waterfall" by M.C. Escher.

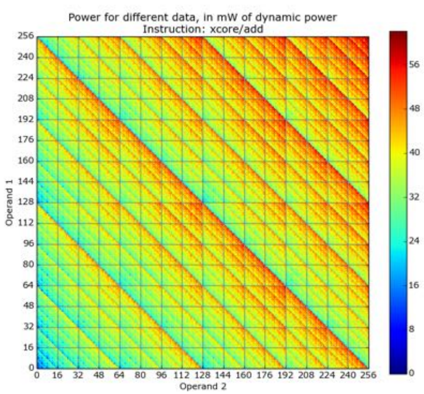
Confidence in a system's trustworthiness can be gained in many different ways, including

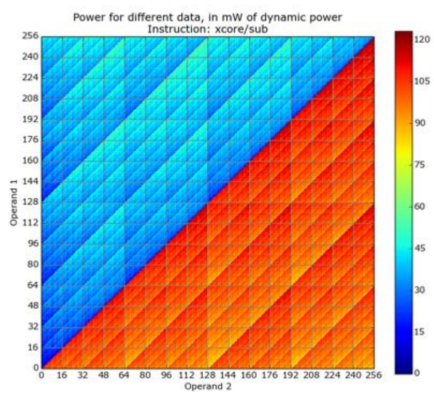
- **by design**, systems that are simple are also understandable;

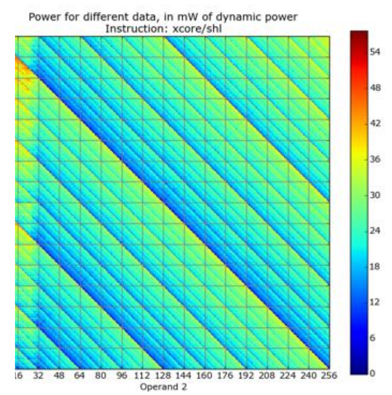


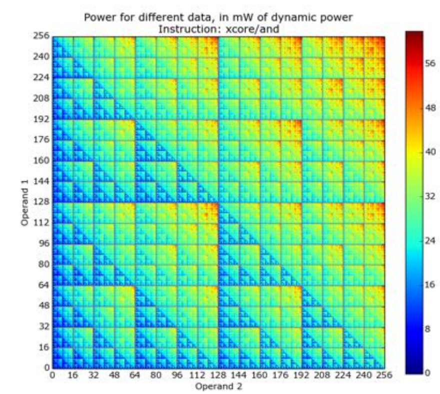
Confidence in a system's trustworthiness can be gained in many different ways, including

- **by design**, systems that are simple are also understandable;
- **through transparency**, systems that allow us an insight into how they make decisions, why they act in a certain way or how they use resources become understandable;





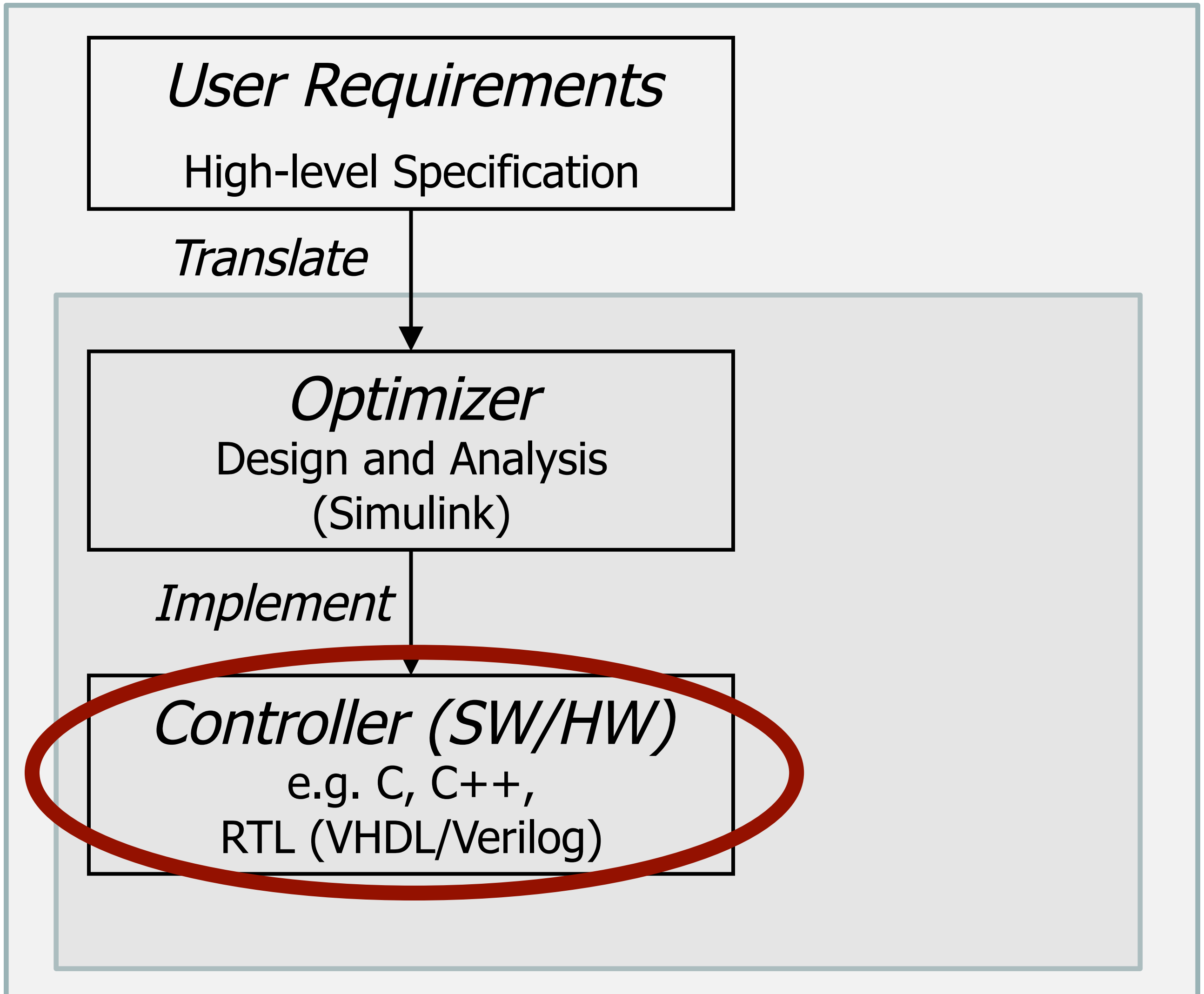




Confidence in a system’s trustworthiness can be gained in many different ways, including

- **by design**, systems that are simple are also understandable;
- **through transparency**, systems that allow us an insight into how they make decisions, why they act in a certain way or how they use resources become understandable; and
- **through verification and validation**, rigorous proof complemented by simulation-based testing using intelligent test generation methods can provide convincing evidence of a system’s trustworthiness.

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:

- Null pointer dereferencing
- Accesses to uninitialized data
- Array and vector out-of-bounds accesses
- Dynamic memory allocation and blocking inter-thread communication (non real-time)




Domain-specific bugs:

- Integer and floating-point arithmetic errors
- Mathematic functions domain errors


Design for Verification

- **SPARK,** a verifiable subset of Ada
  - Originally developed for high integrity software
- Three open-source implementations of navigation algorithms translated from C/C++ (2.7 kSLOC) to SPARK (3.5 kSLOC)
    - VFH+ (Vector Field Histogram)
    - ND (Nearness Diagram)
    - SND (Smooth Nearness-Diagram) navigation
- Required code modifications:
  - Pre- and post-conditions, loop (in)variants
  - Numeric subtypes (e.g. Positive)
  - Formal data containers

Results

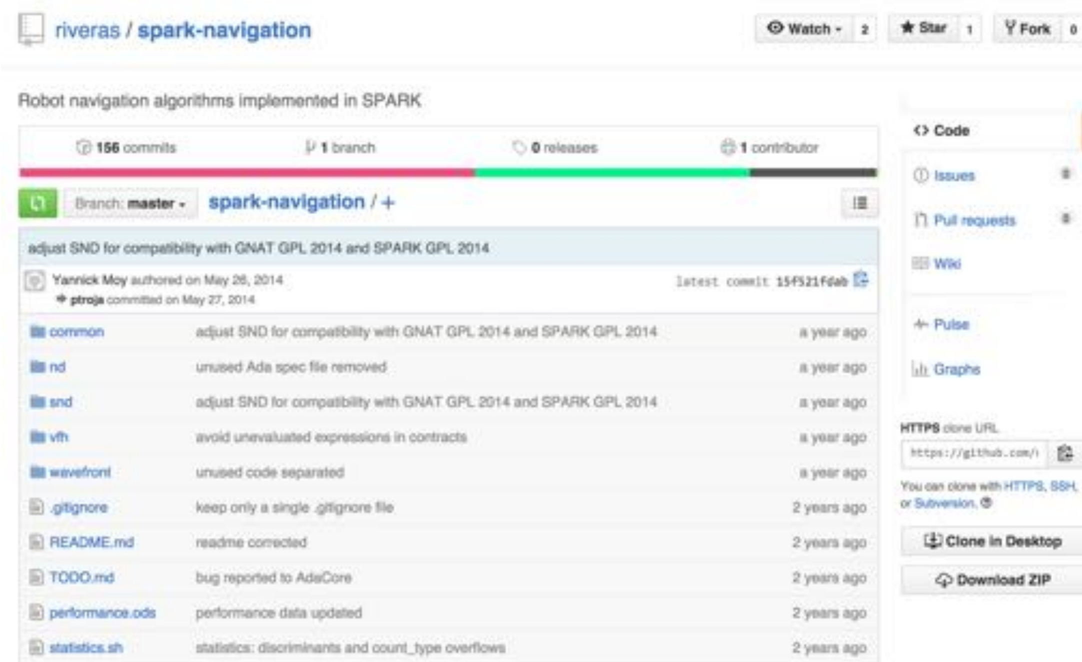
- Several bugs discovered by run-time checks injected by the Ada compiler
  - Fixed code proved to be run-time safe
    - except floating-point over- and underflows
    - These would require the use of complementary techniques.

- Up to 97% of the verification conditions discharged automatically by SMT solvers in less than 10 minutes
- SPARK code is on average 30% longer than C/C++
- No significant difference wrt the performance of the SPARK and C/C++ code

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!

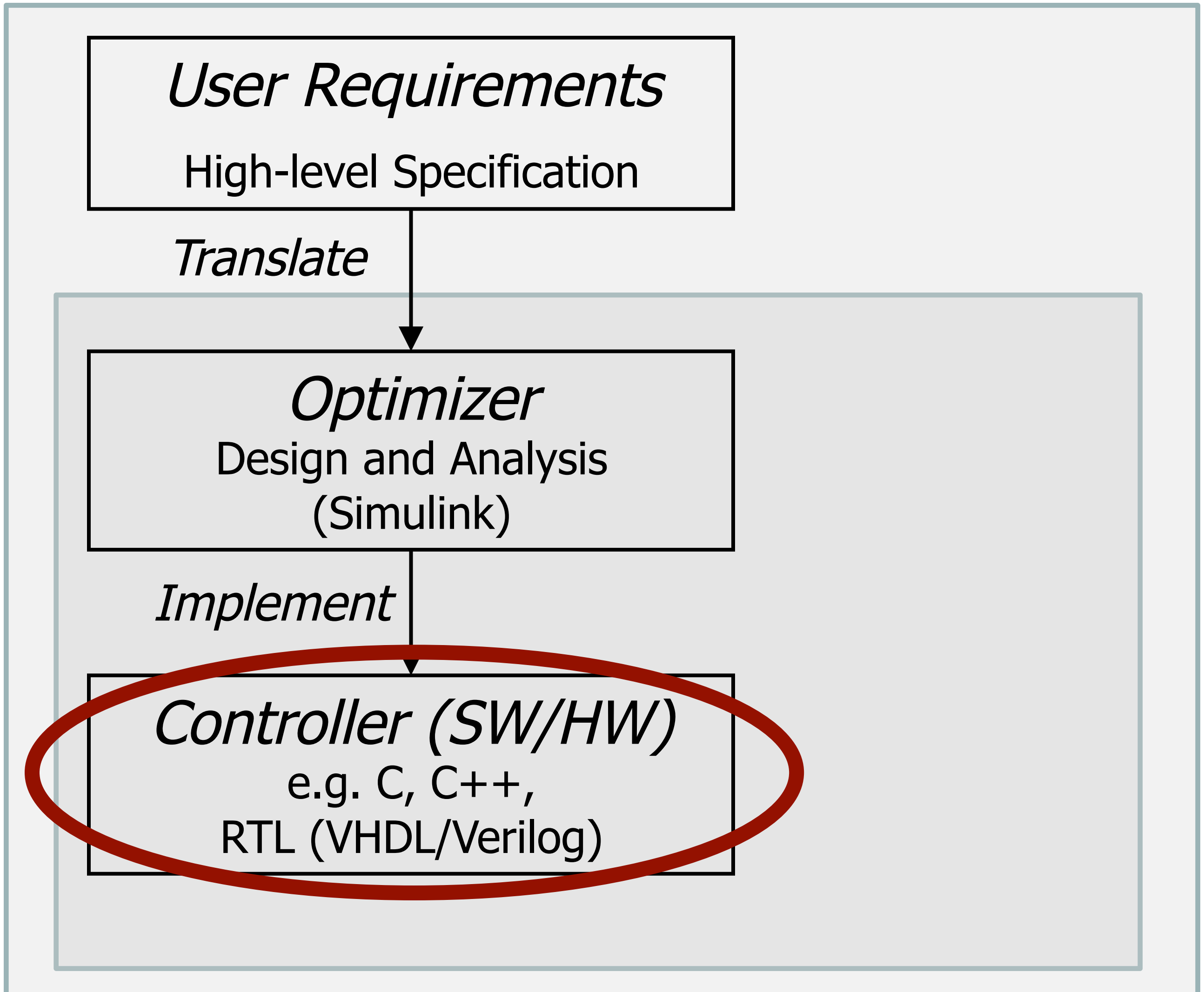
https://rclutz.wordpress.com/2016/09/23/hammer-and-nail/



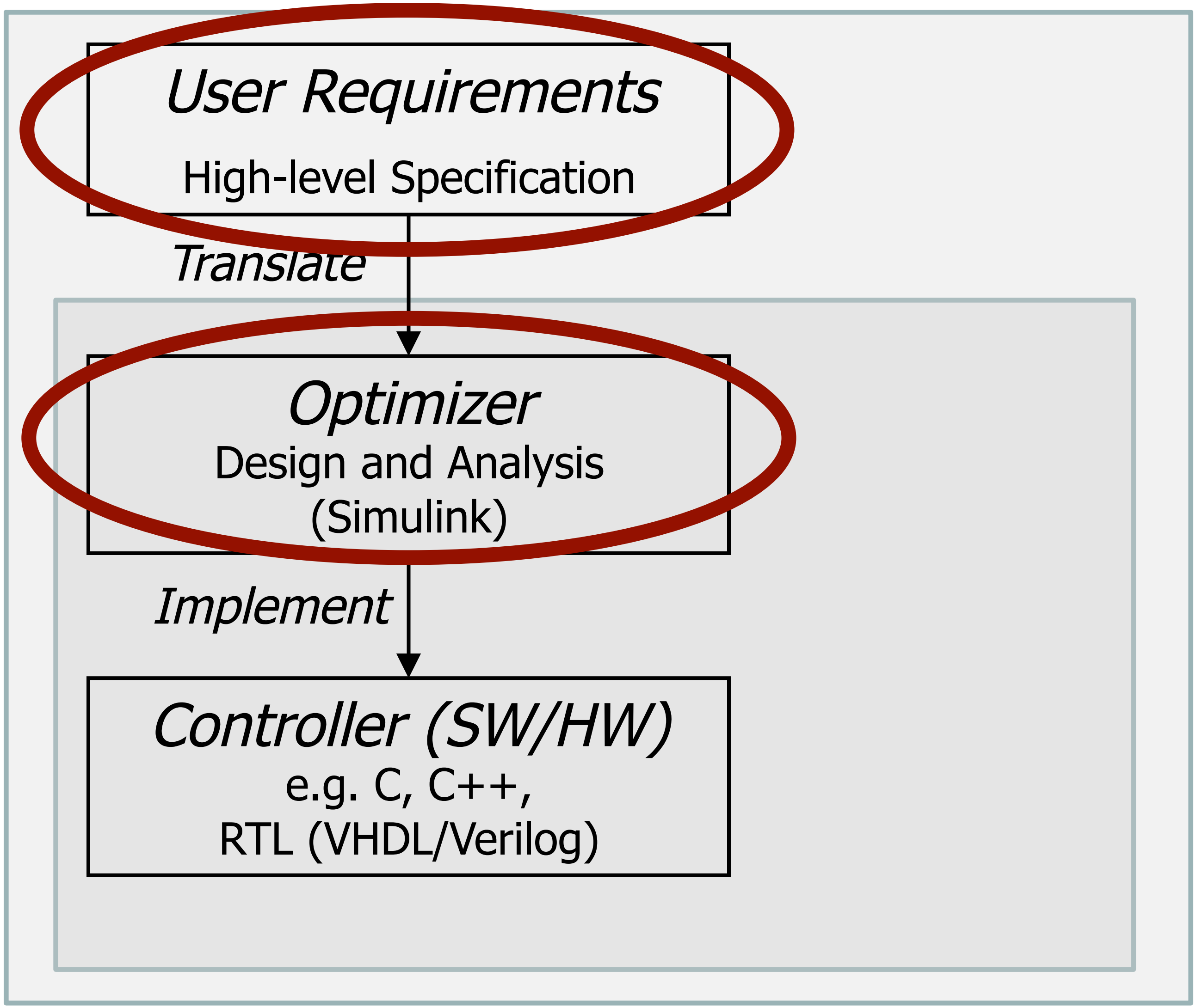
http://github.com/riveras/spark-navigation

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



Correctness from specification to implementation

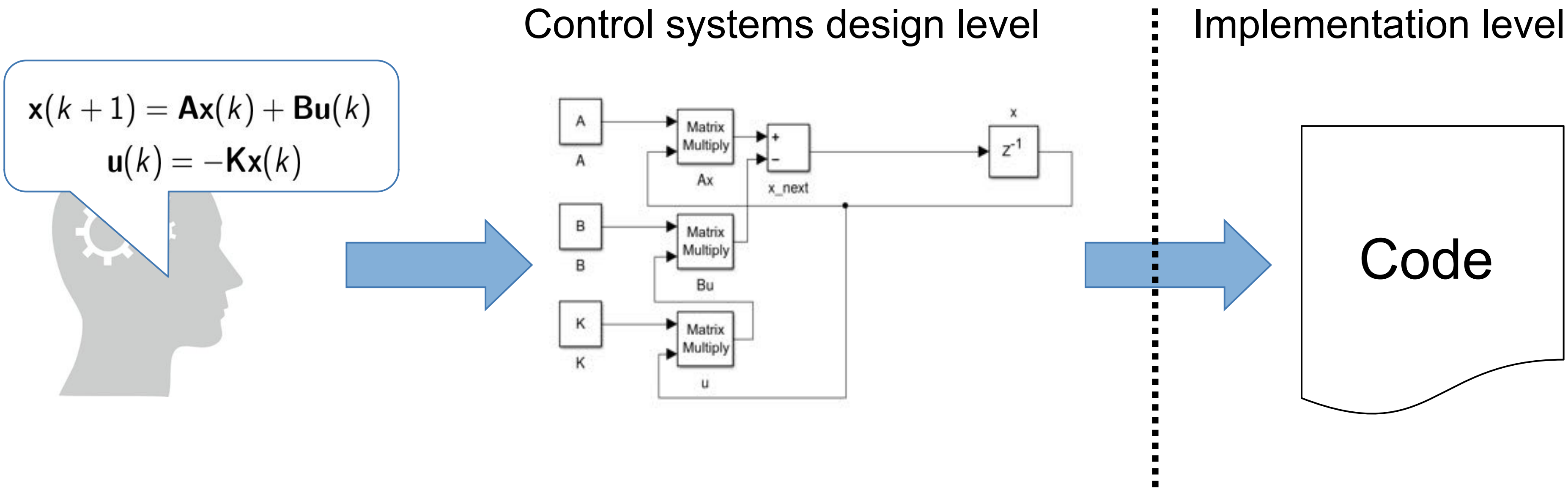


What can be done at the design level?

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 in Control System Design



Important to distinguish design flaws from coding bugs

- Analysis techniques from control systems theory (e.g., stability)
- Serve as requirements/specification
- For (automatic) code generation

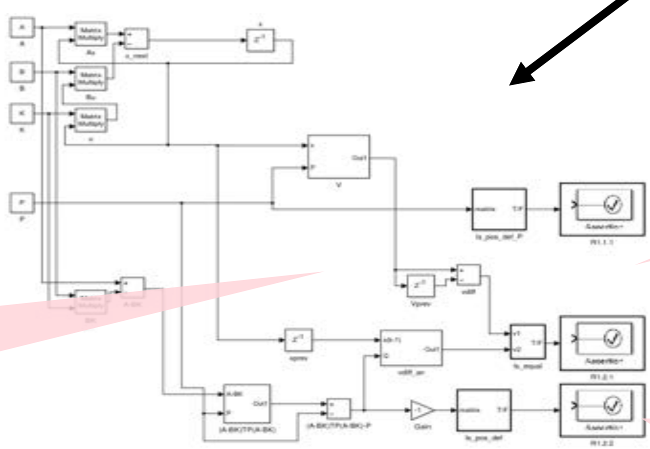
Verifying Stability

Stability

Matrix $\mathbf{P} > 0$
(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)

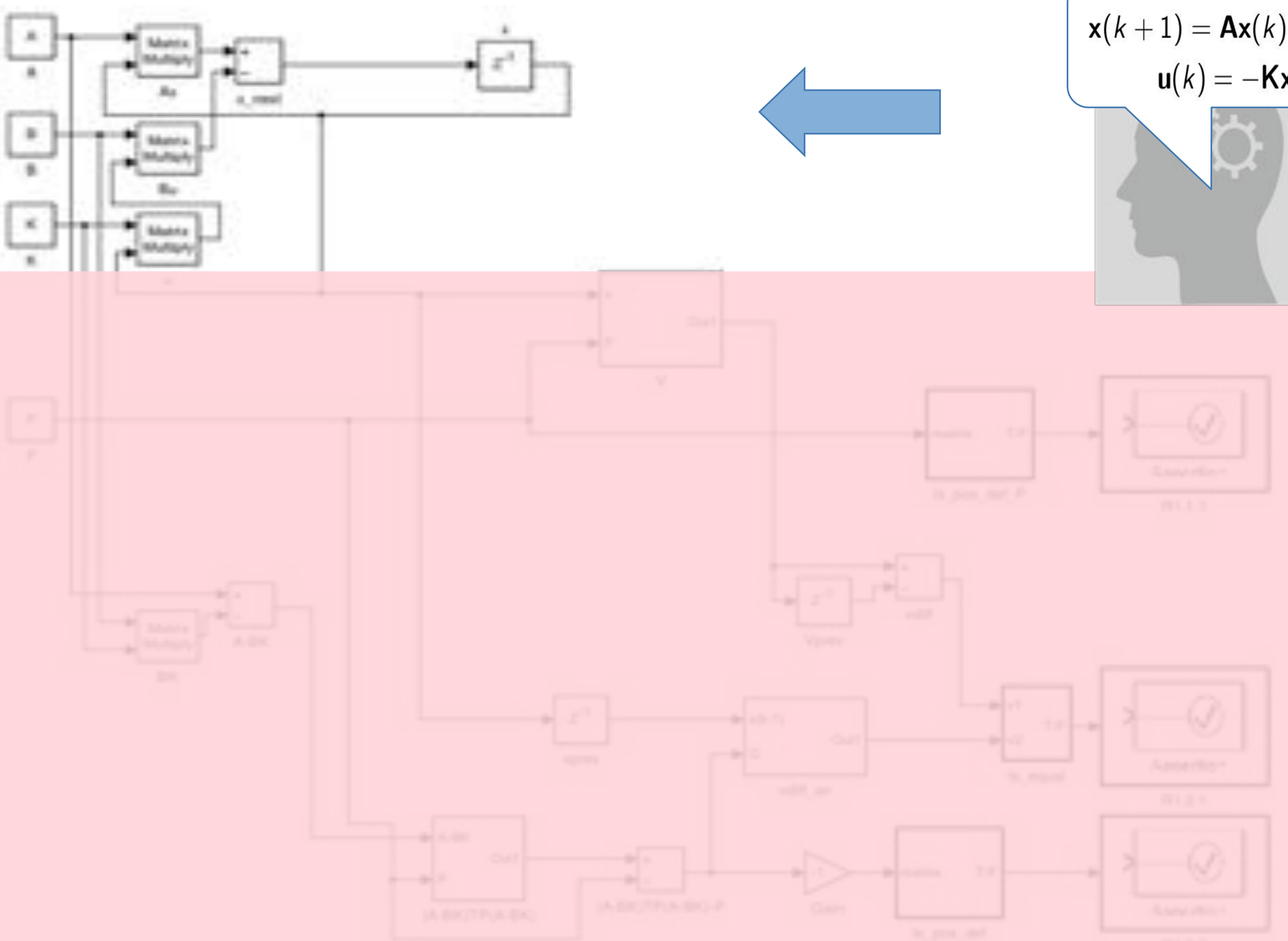


Capture control systems requirements

Add as assertions

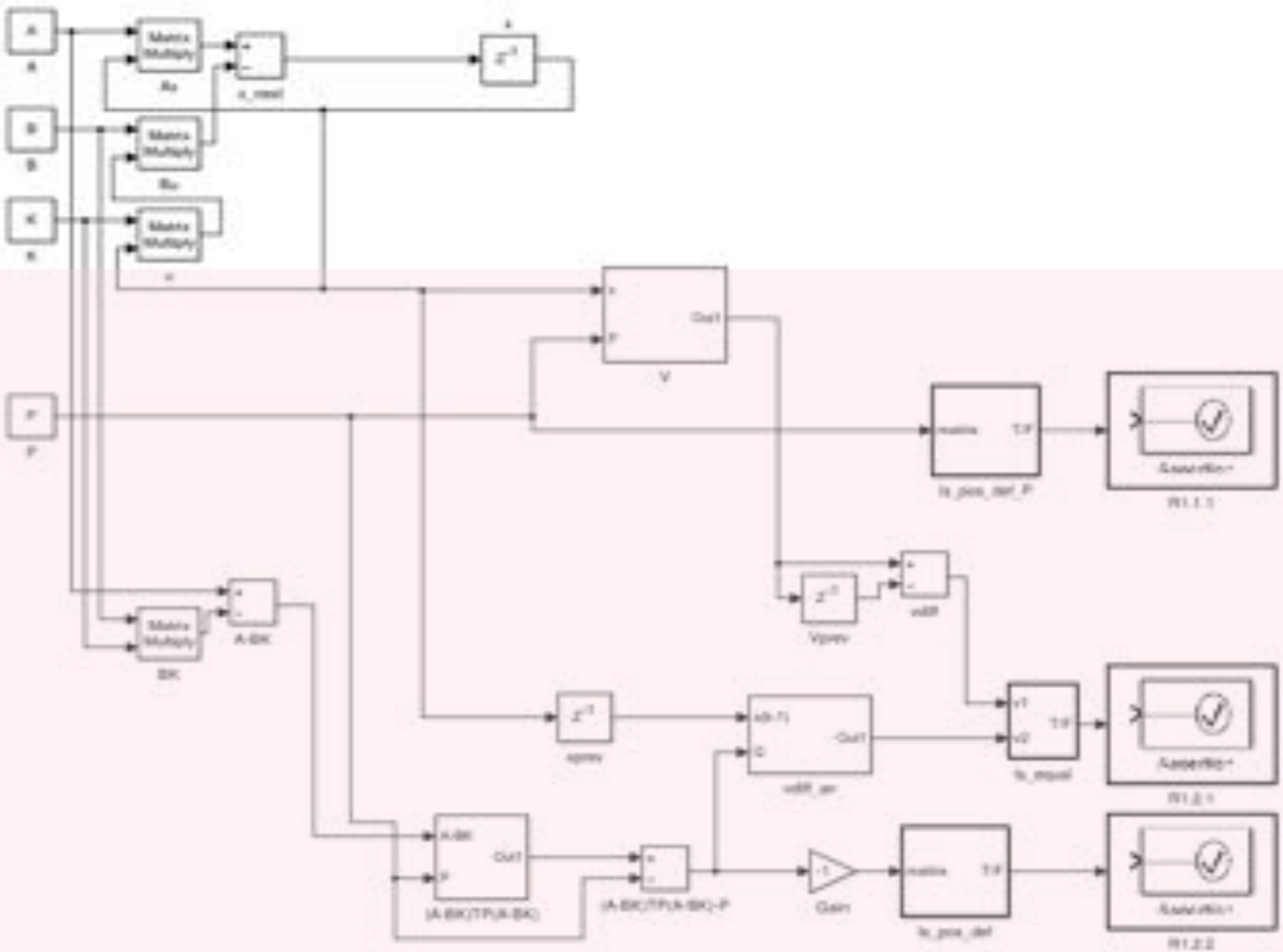
Retain in code implementation

Assertion-Based Verification



$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$$
$$\mathbf{u}(k) = -\mathbf{K}\mathbf{x}(k)$$

Assertion-Based Verification



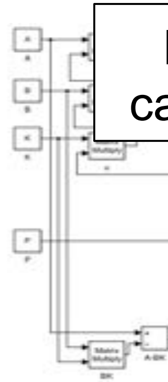
Combining Verification Techniques

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)



Formalize logic theory to capture the Simulink design

Axiom: Bu = B * u
...

...

Goal: vdiff == vdiff_an

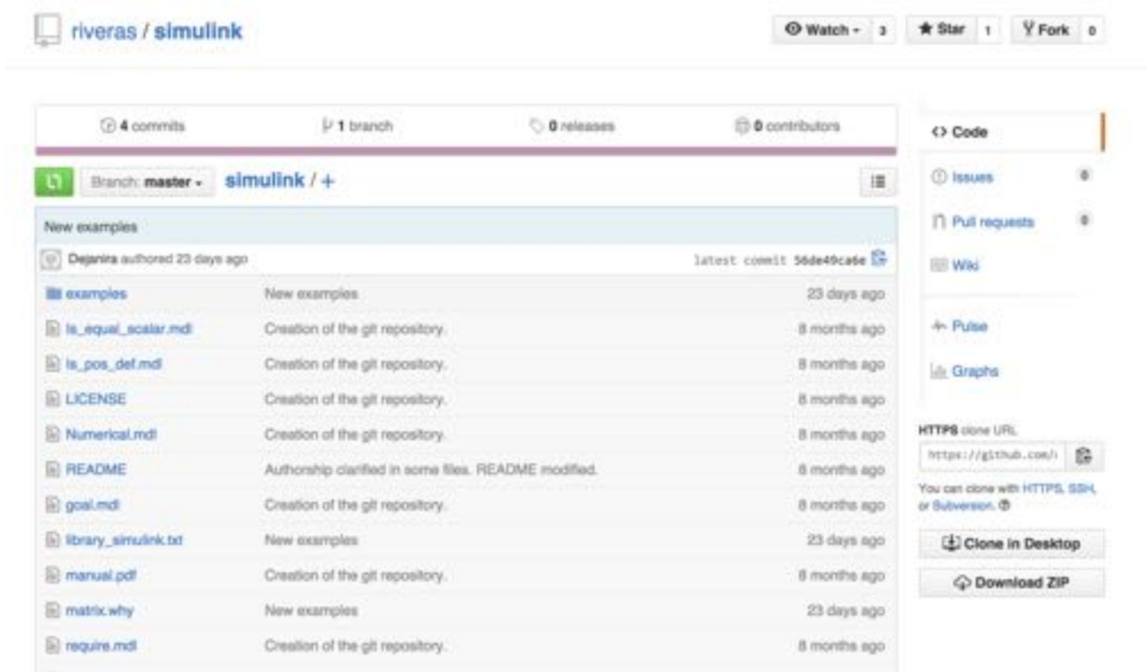
Test in simulation

Automatic theorem proving

Moral

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

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

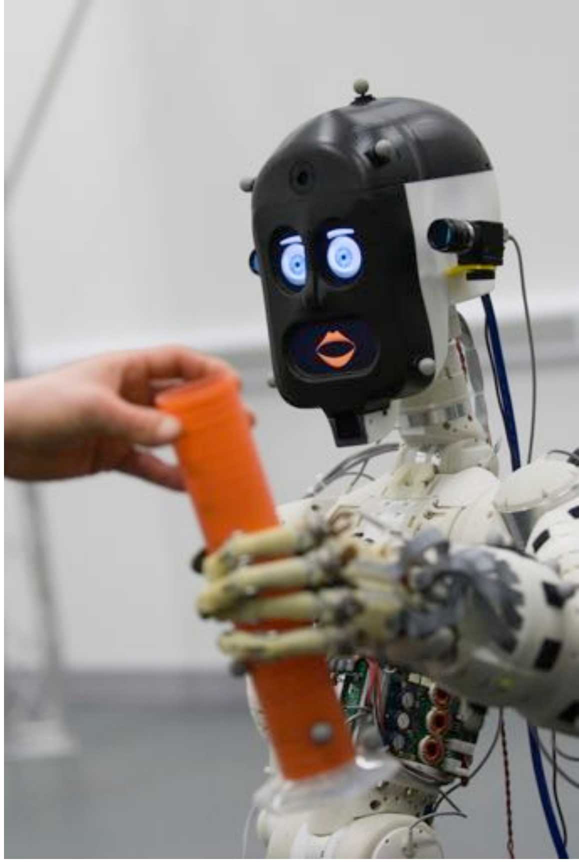


http://github.com/riveras/simulink

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 increase the productivity of simulation-based testing?



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), Dresden, Germany, 2016. http://arxiv.org/abs/1604.05508

Challenges

- Complexity
  - HW
  - SW
- High levels of Concurrency
- Experiments
  - Expensive
  - Unsafe

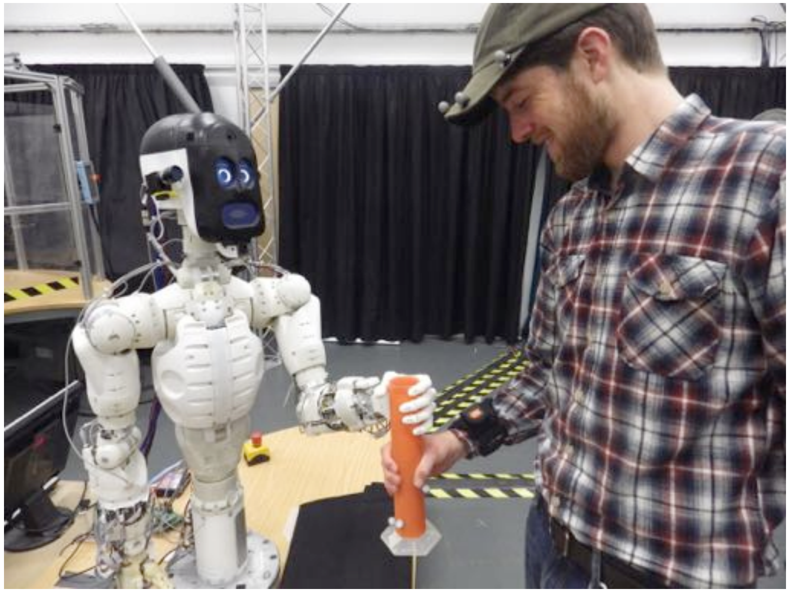
Challenges

- Complexity
  - HW
  - SW
  - People
  - Environment
- High levels of Concurrency
- Experiments
  - Expensive
  - Unsafe

We are investigating…

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

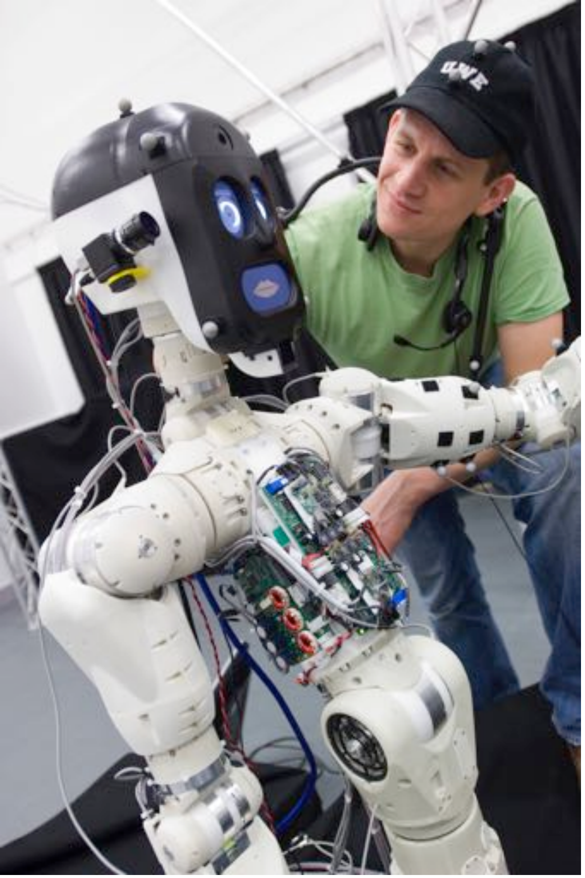
… to **verify** code that controls robots ...



We are investigating…

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

… to **verify** code that controls robots and AVs

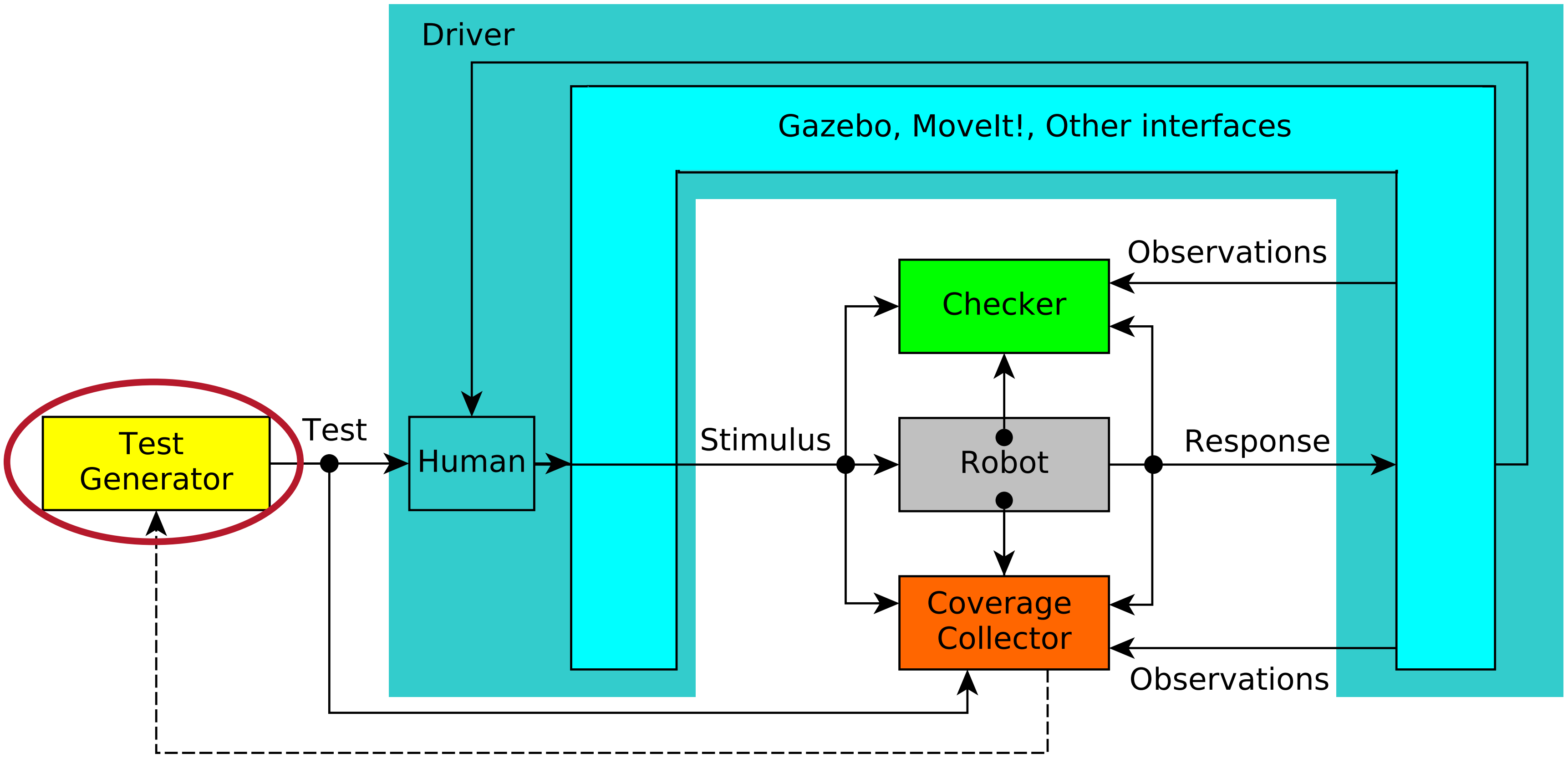


CDV to *automate* simulation-based testing

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)

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)

Test Generator

- Tests must be effective and efficient
- Strategies:
  - Pseudorandom (repeatability)

Test Generator

- Tests must be effective and efficient
- Strategies:
  - Pseudorandom (repeatability)
  - Constrained pseudorandom
  - Model-based to target specific scenarios

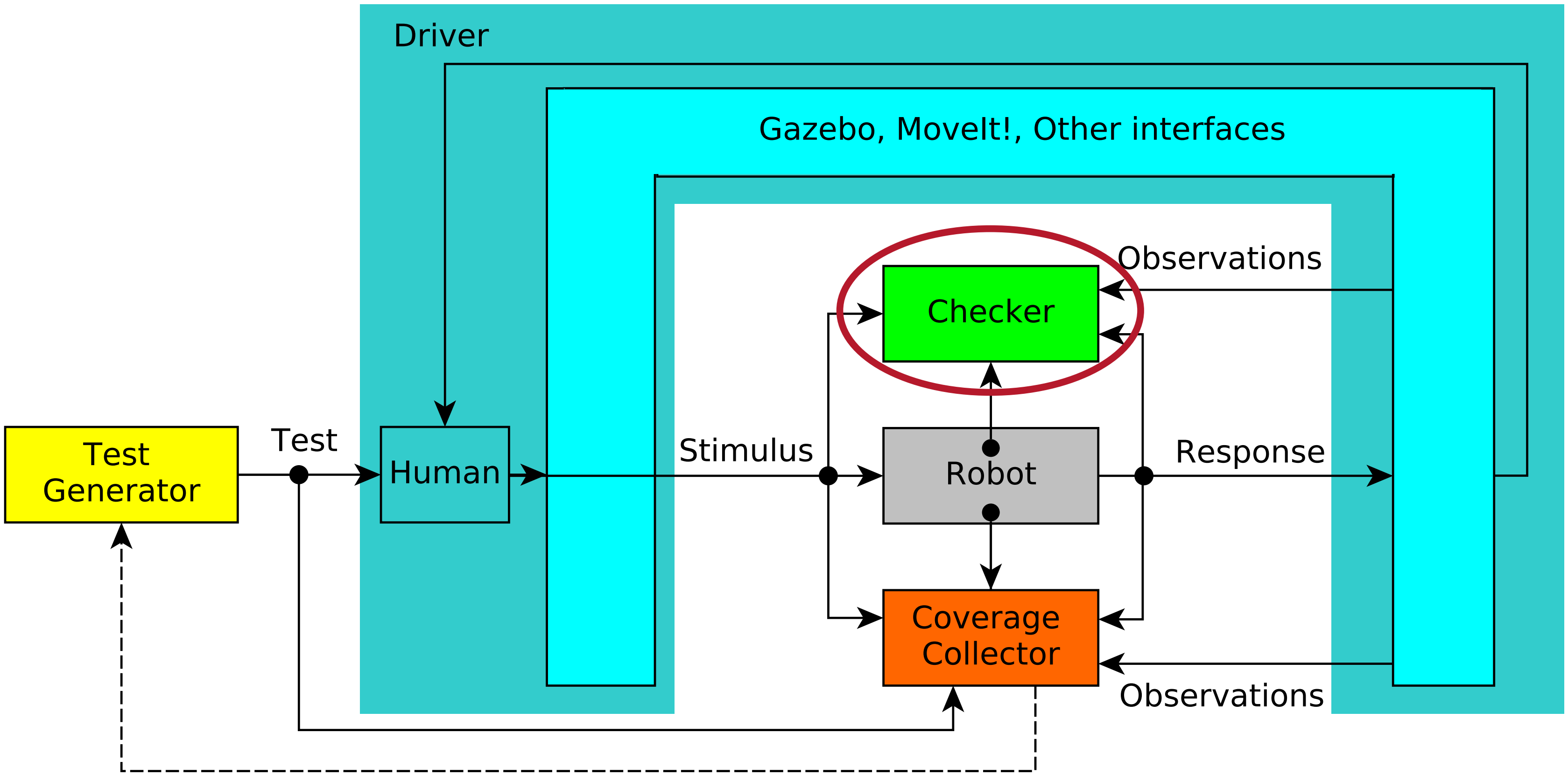
Model-based test generation

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

System + environment

Environment to drive system

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

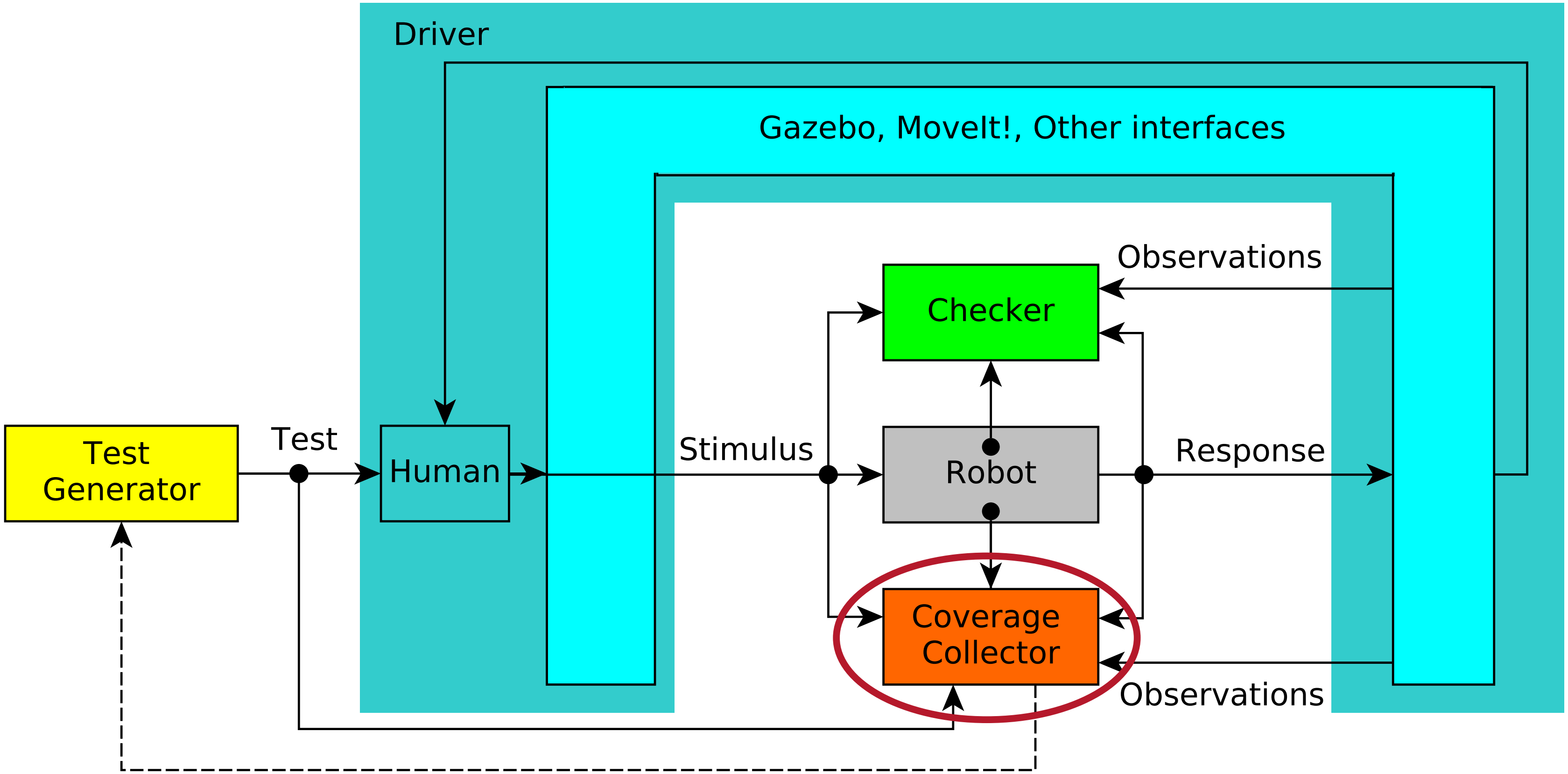
Assertion Checkers

Data Output | Explain | Messages | Notifications

	id [PK] integer	agent_id integer	sim_time double precision	asr_id integer	asr_result boolean
1	1	33	2	1	false
2	2	34	2	1	false
3	3	35	2	1	false
4	4	36	2	1	false
5	5	37	2	1	false
6	6	33	2	2	false
7	7	34	2	2	false
8	8	35	2	2	false
9	9	36	2	2	false
10	10	37	2	2	false

Collision detected > 8 seconds
with agent_id = 37 (cyclist)

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 Models

- **Code coverage**
- **Structural coverage**
- **Functional coverage**
  - Requirements coverage
    - 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

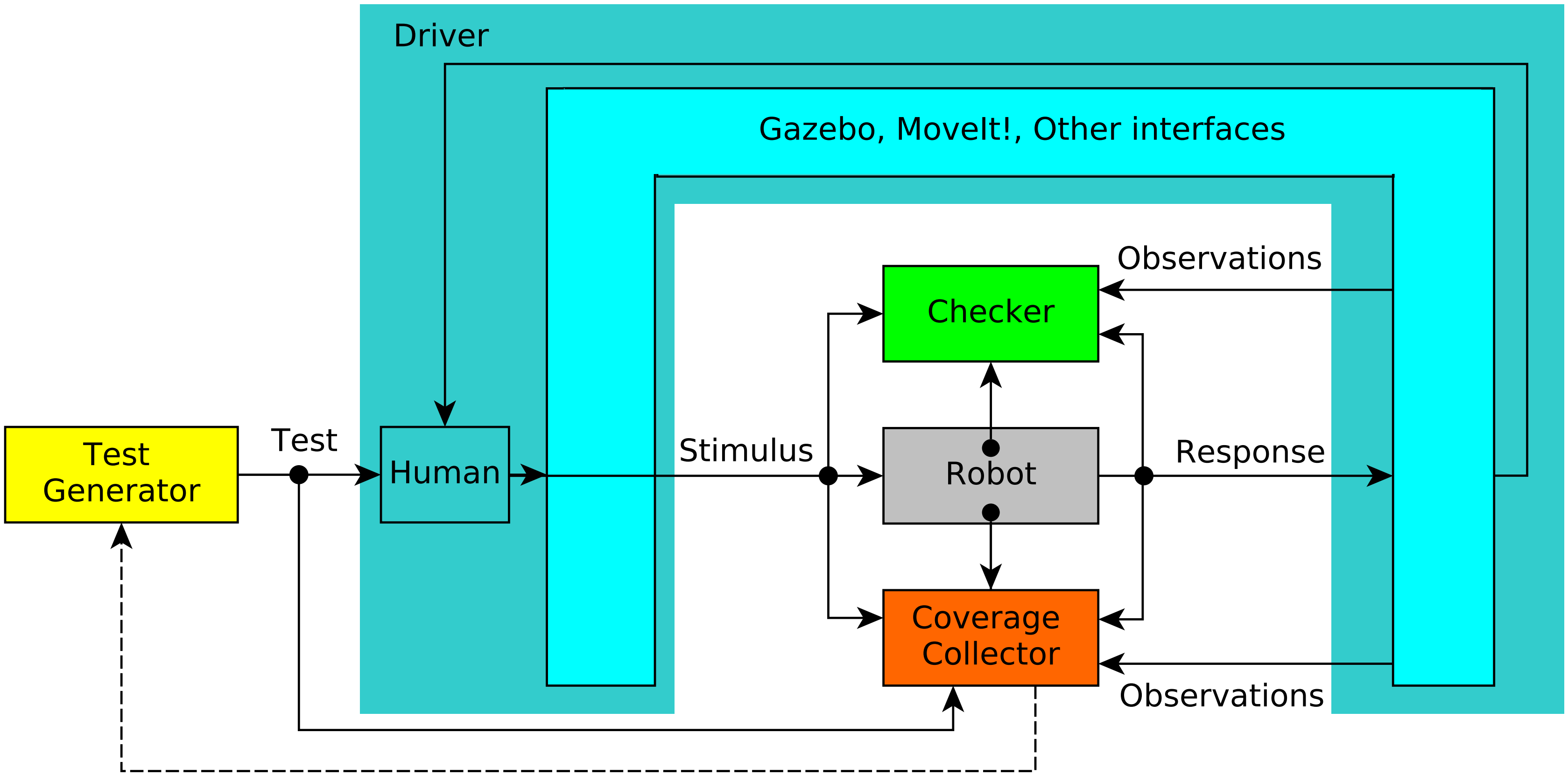
1. If the gaze, pressure and location are sensed as correct, then the object shall be released.
2. If the gaze, pressure or location are sensed as incorrect, then the object shall not be released.
3. The robot shall make a decision before a threshold of time.
4. The robot shall always either time out, 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.

Coverage Models

- **Code coverage**
- **Structural coverage**
- **Functional coverage**
  - Requirements coverage
    - Functional and safety (ISO 13482:2014, ISO 10218-1)
  - Cross-product functional coverage (Situation coverage)

(Gaze, Pressure, Location)	Sense timeout	Release piece	No release	
$(\bar{1},\bar{1},\bar{1})$				
$(\bar{1},\bar{1},1)$				
$(\bar{1},1,\bar{1})$				
$(\bar{1},1,1)$				
$(1,\bar{1},\bar{1})$				
$(1,\bar{1},1)$				
$(1,1,\bar{1})$				
$(1,1,1)$				

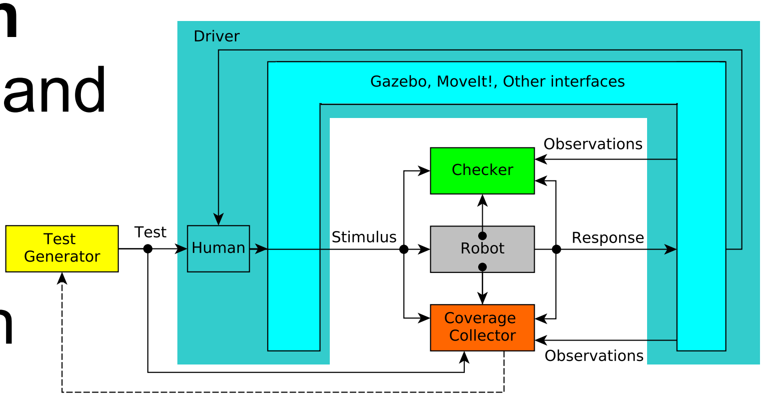
Coverage-Directed Verification

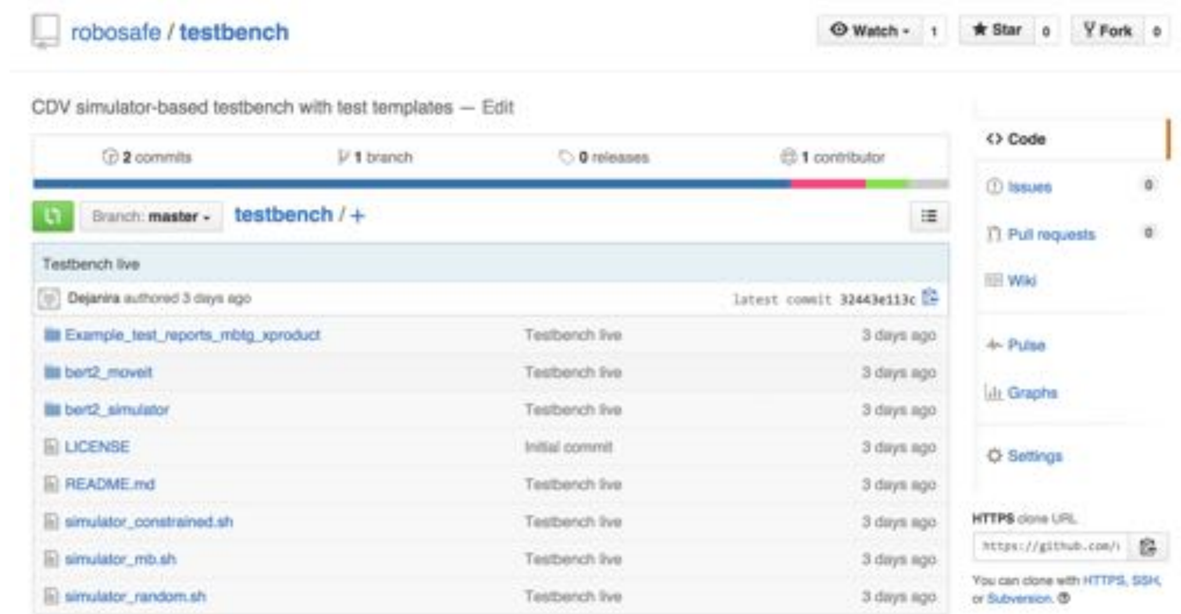


Dejanira Araiza-Illan, David Western, Anthony Pipe and Kerstin Eder.
Systematic and Realistic Testing in Simulation of Control Code for Robots in Collaborative Human-Robot Interactions. 17th Annual Conference Towards Autonomous Robotic Systems (TAROS 2016), pp. 20-32. Lecture Notes in Computer Science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

Coverage-Directed Verification

- **systematic, goal directed** verification method
  - offers a **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 constrained pseudorandom test generation



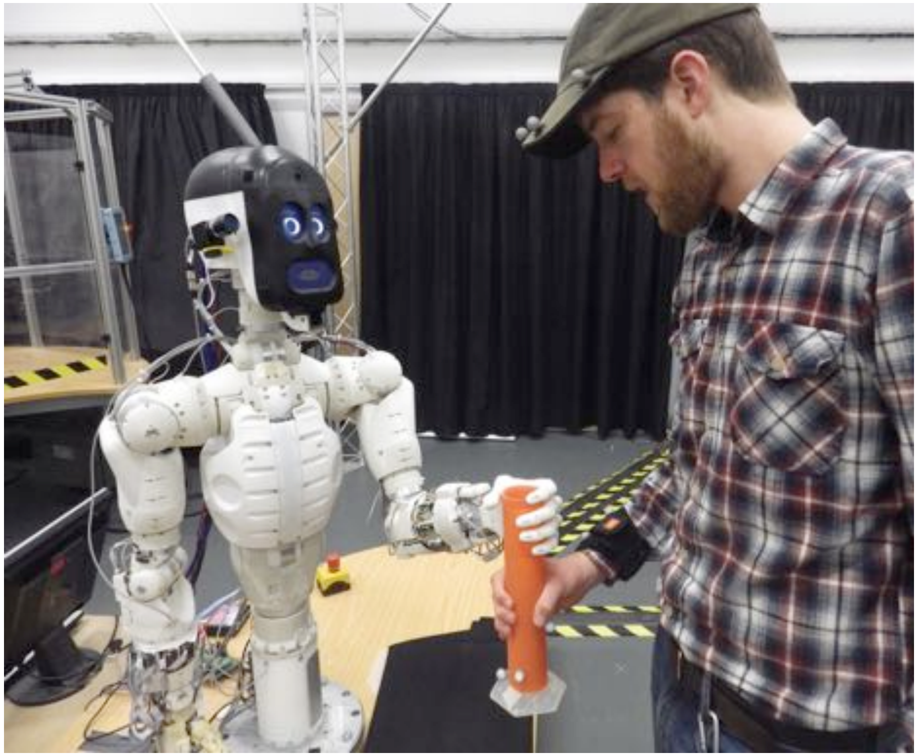


http://github.com/robosafe/testbench

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 Computer Science 9716. Springer, June 2016. (DOI: 10.1007/978-3-319-40379-3_3)

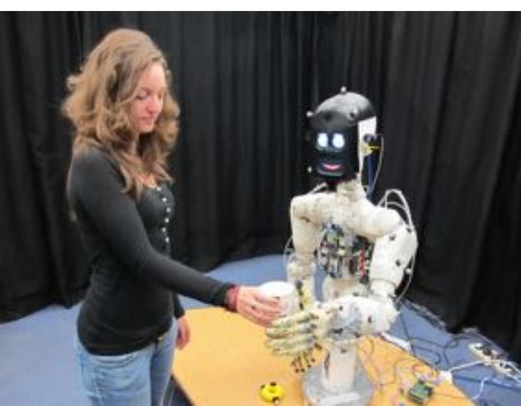
CDV provides *automation*



What about *agency*?

Agency for Intelligent Testing

- Robots and autonomous systems in general need to be both powerful and *smart*.
  - AI and learning are increasingly used
- 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

Intentions: chosen plans, according to current beliefs and goals

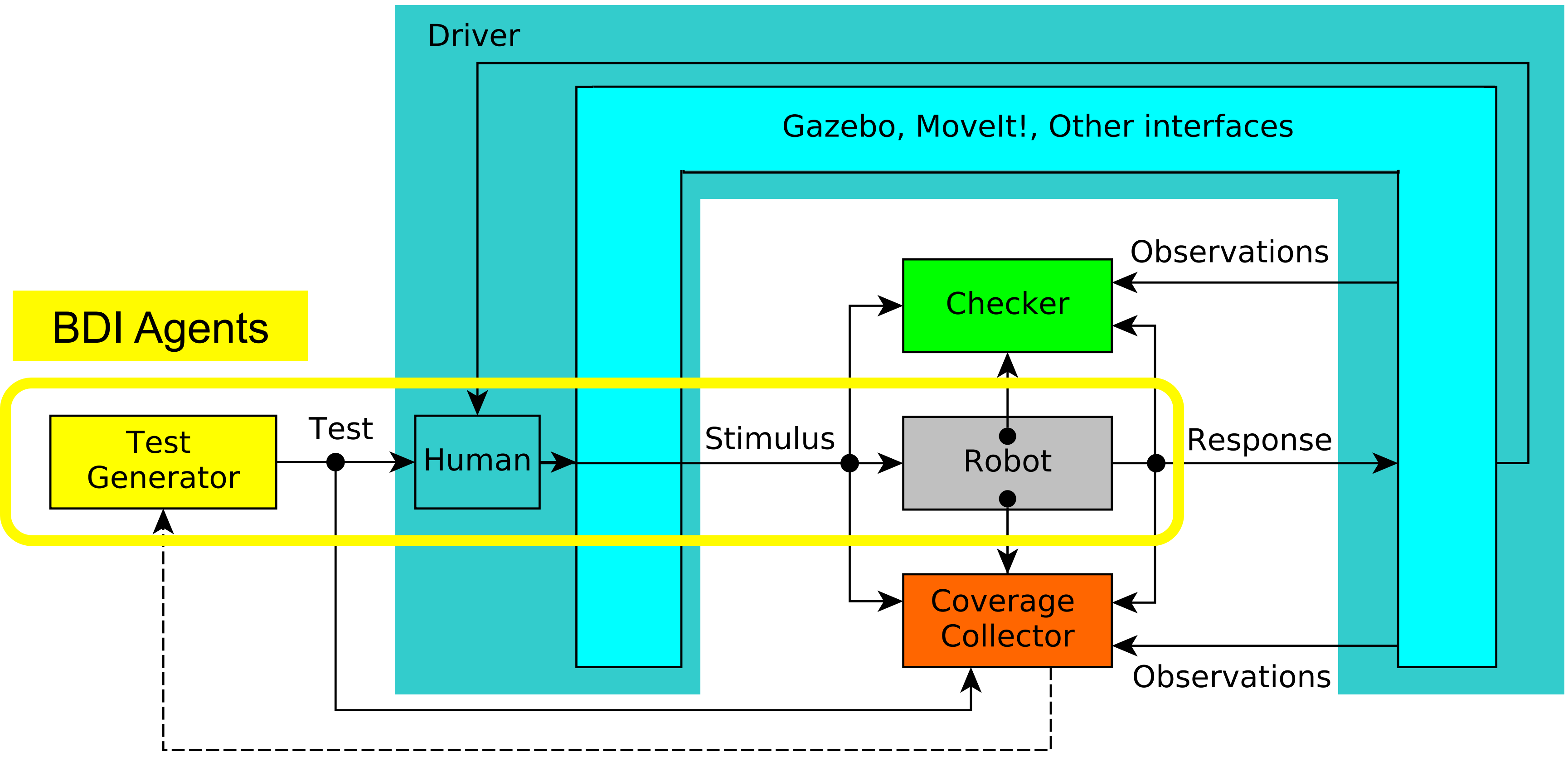
New beliefs

New goals

Guards for plans

From executing plans

CDV testbench components



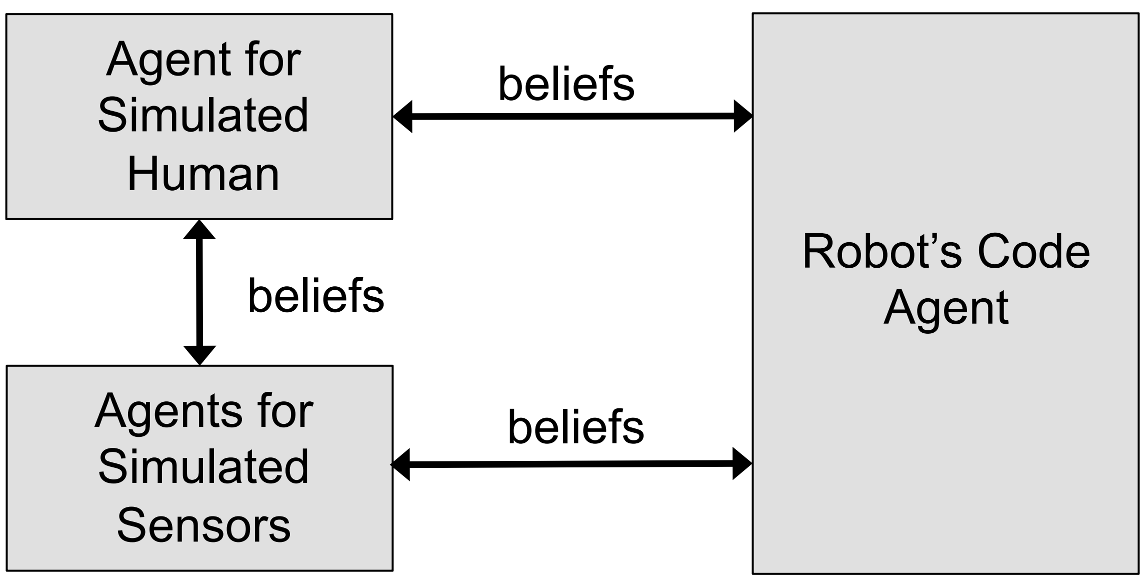
Intelligent testing is harnessing the power of BDI agent models to introduce agency into test environments.

Research Questions

- 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?

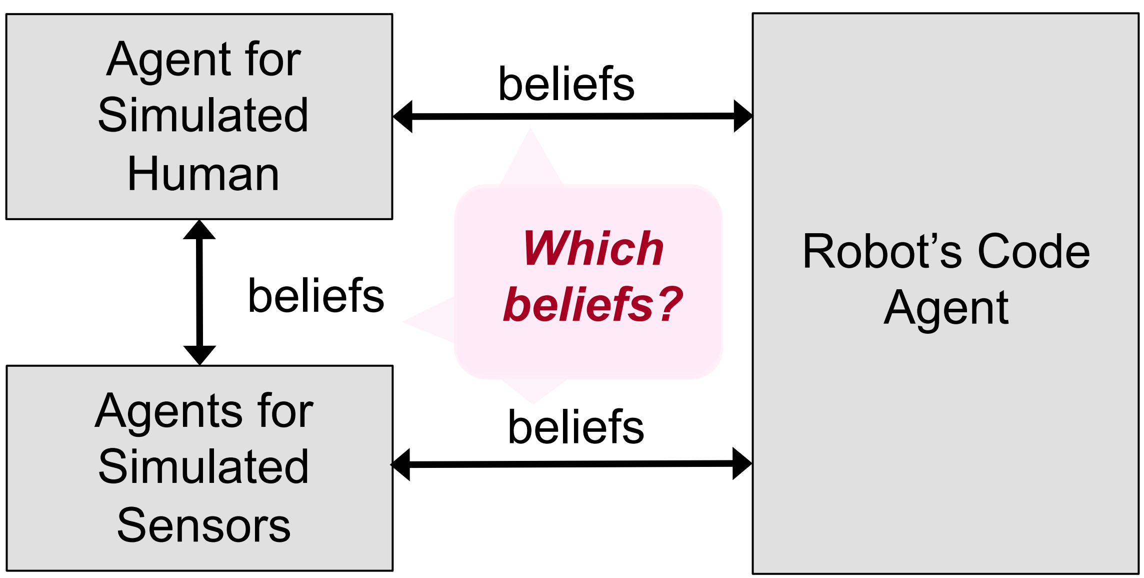
Interacting Agents

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



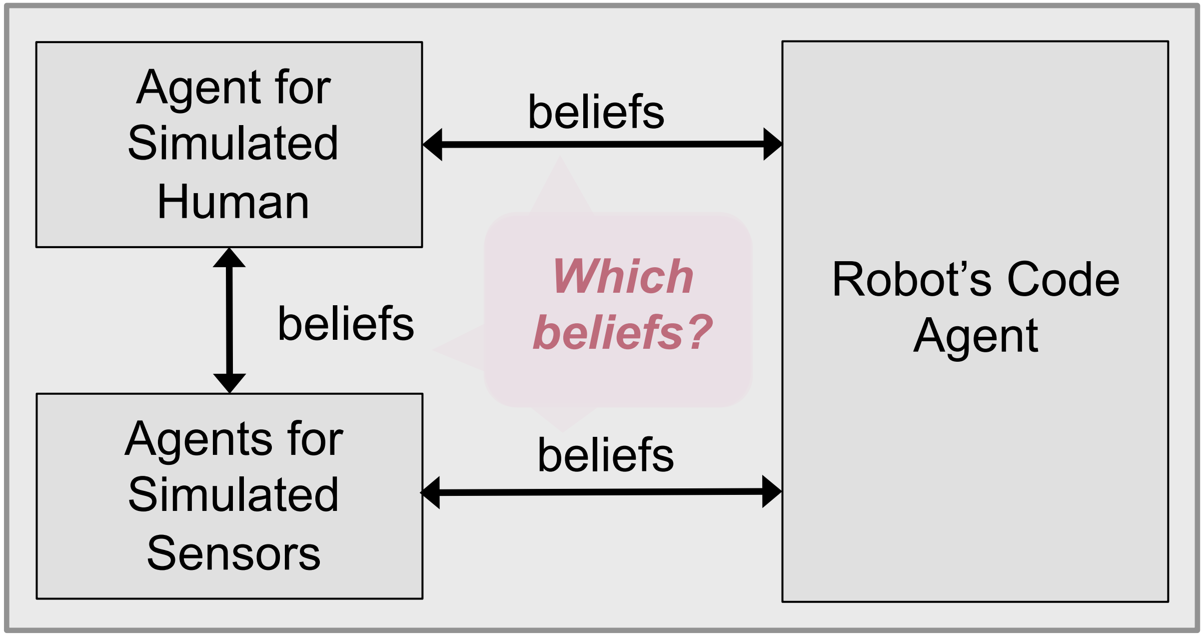
Interacting Agents

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



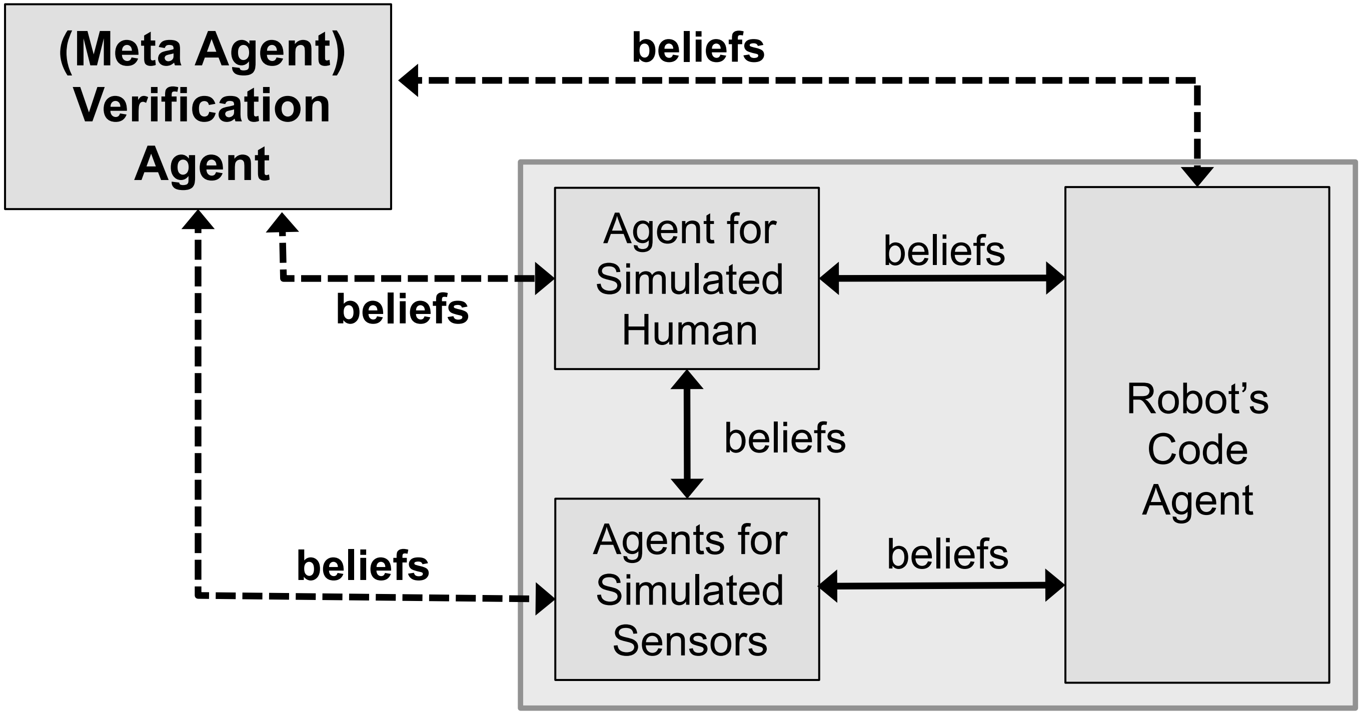
Interacting Agents

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

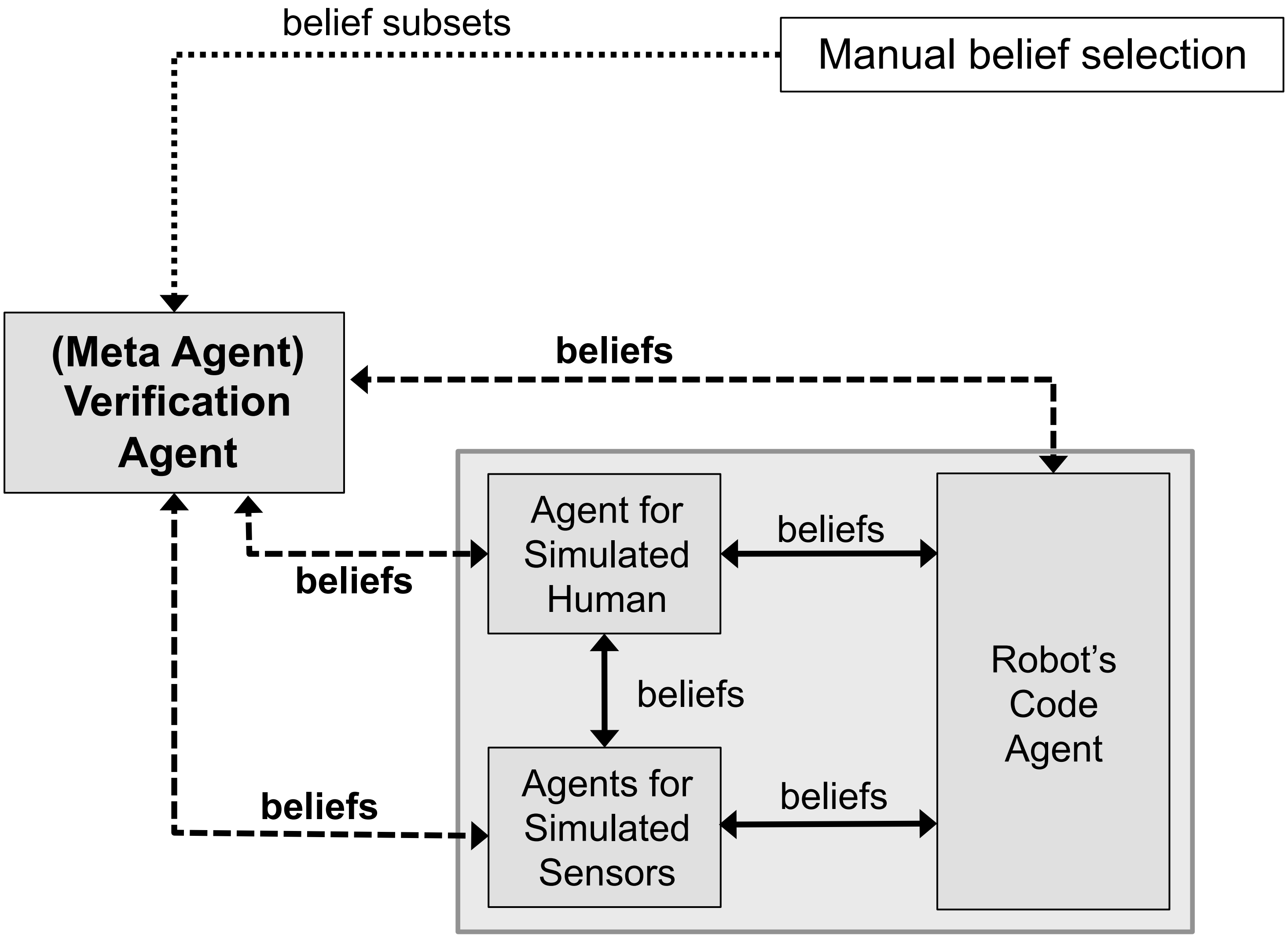


Verification Agents

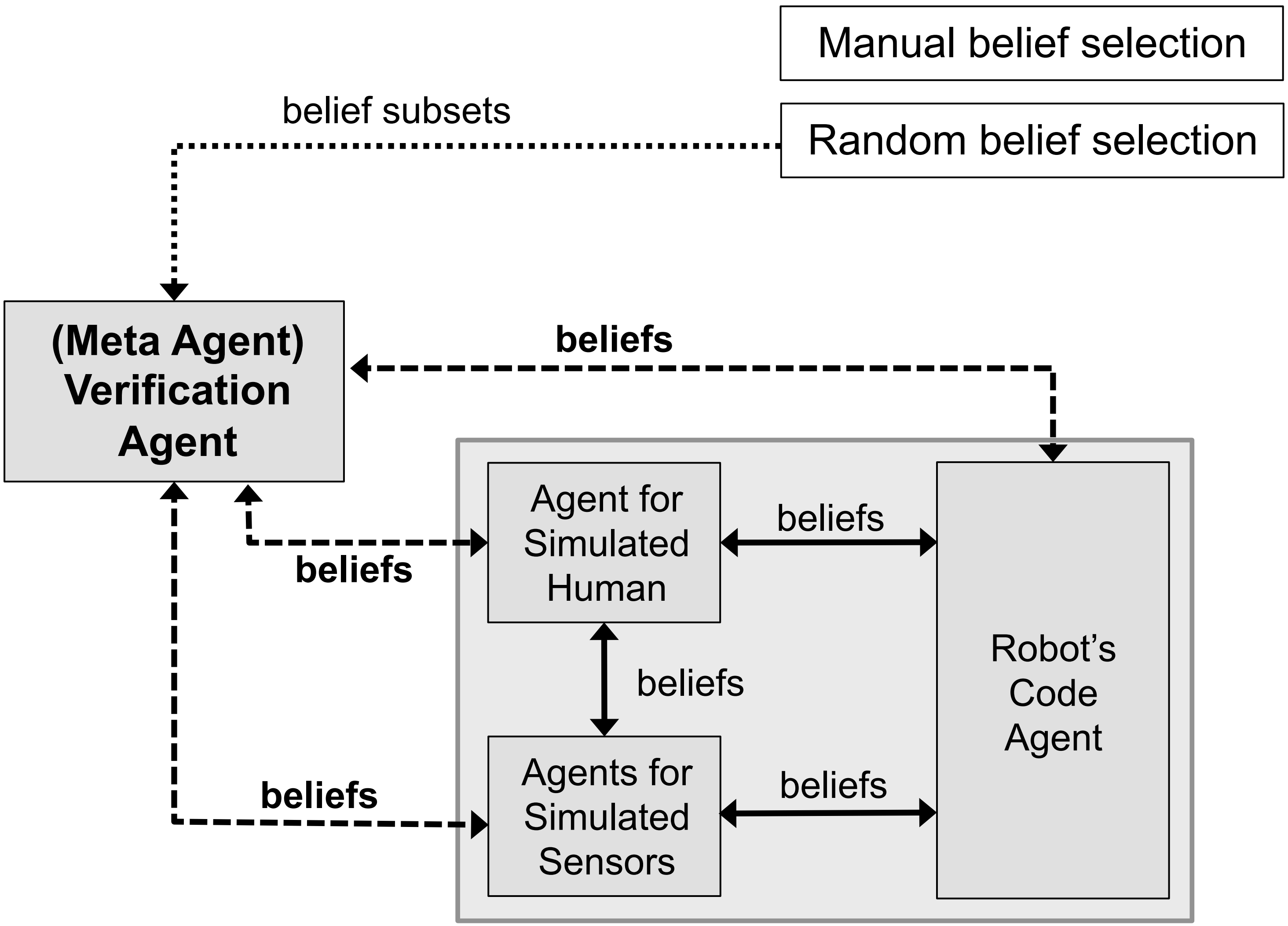
- Meta agents can influence beliefs
- This allows biasing/directing the interactions



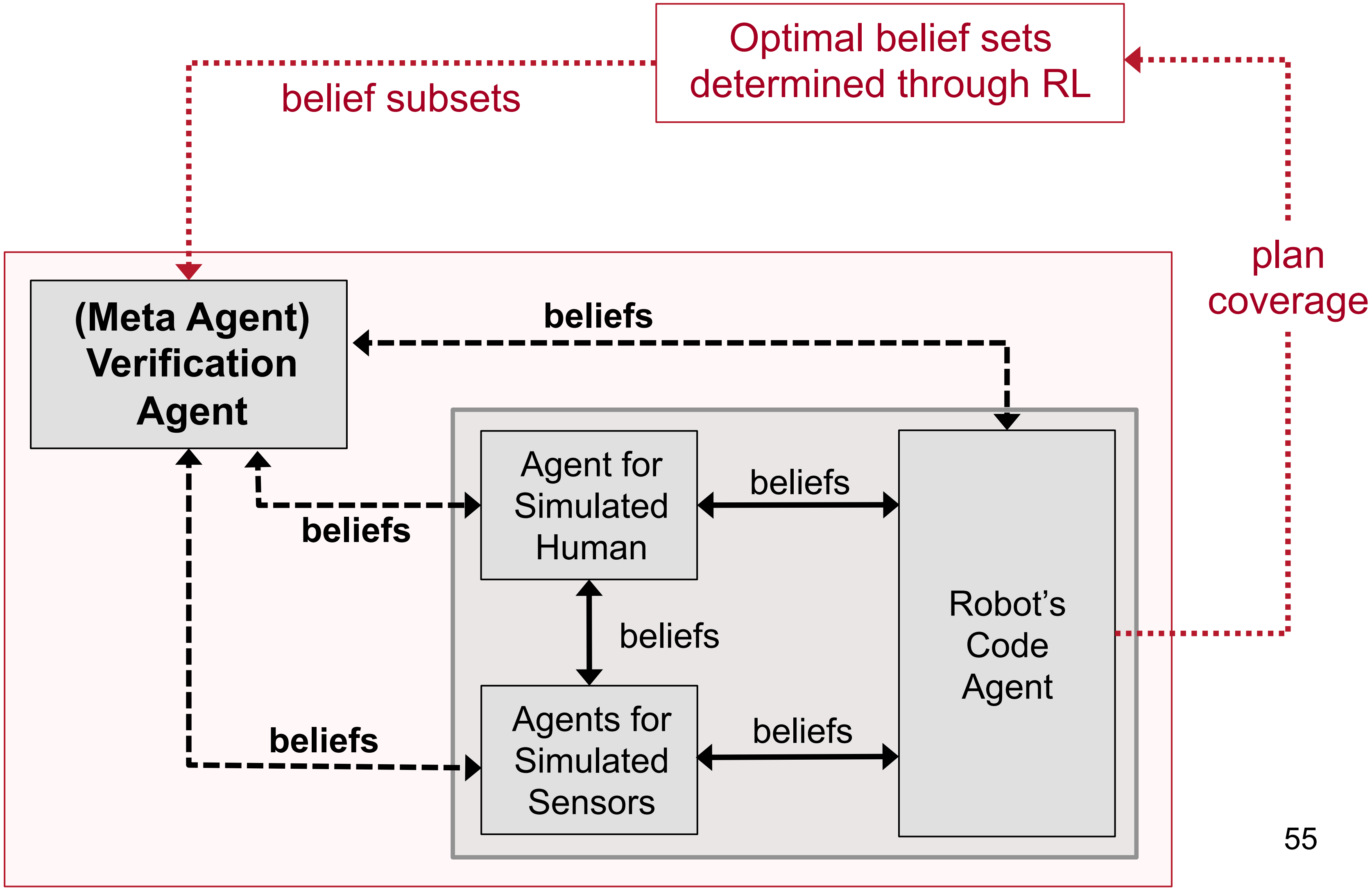
Which beliefs are effective?

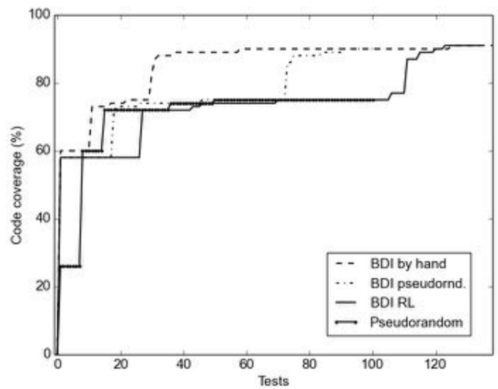


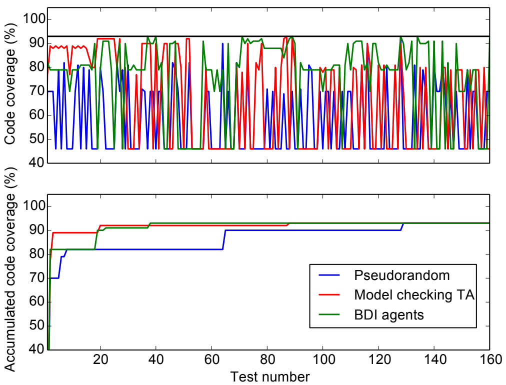
Which beliefs are effective?



Which beliefs are effective?







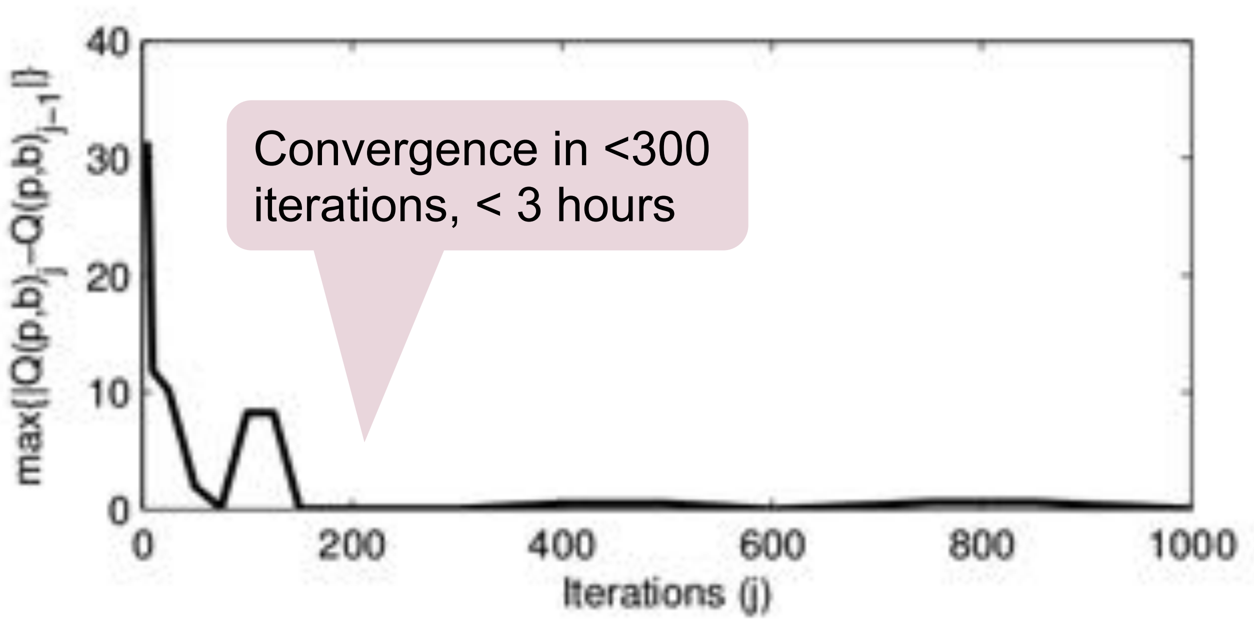
Results

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)

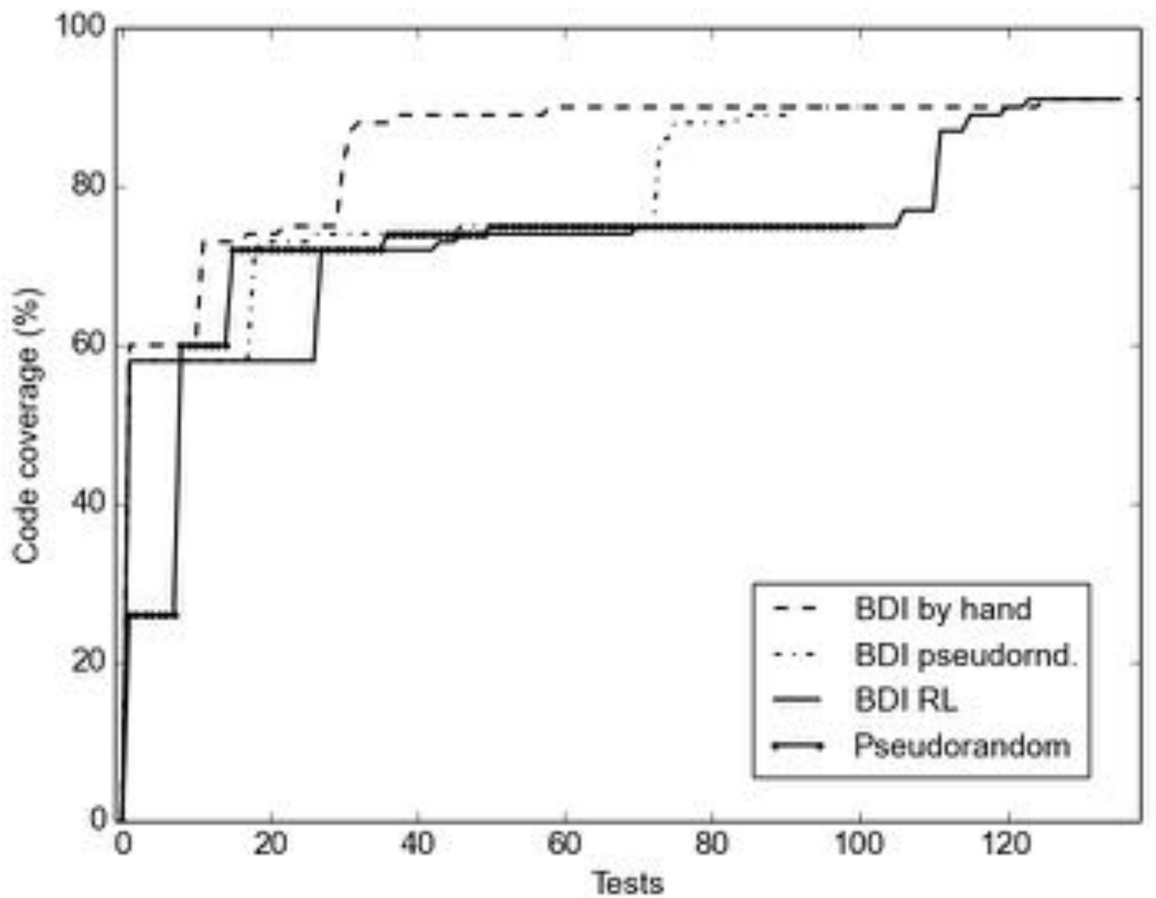
D. Araiza-Illan, A.G. Pipe, K. Eder
Model-based Test Generation for Robotic Software: Automata versus Belief-Desire-Intention Agents. (under review, preprint available at arXiv:1609.08439)

The cost of learning belief sets

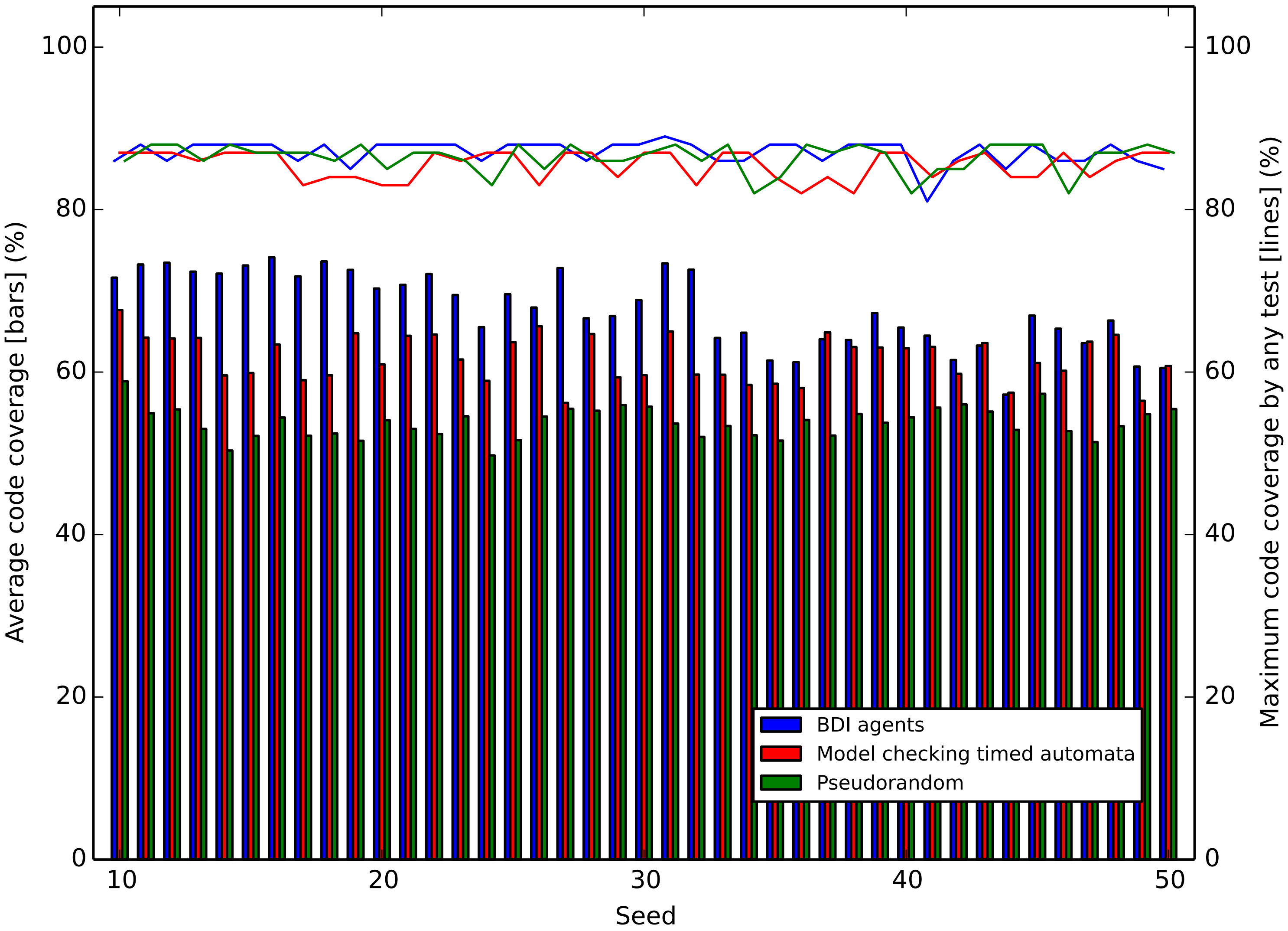


The cost of learning a good belief set needs to be considered when assessing the different BDI-based test generation approaches.

Code Coverage Results



BDI-agents vs timed automata



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	$\approx$ 10.5 hrs	$\approx$ 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	$\approx$ 5.5 hrs	$\approx$ 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

- **Traces of interactions** between BDI agents 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.
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)

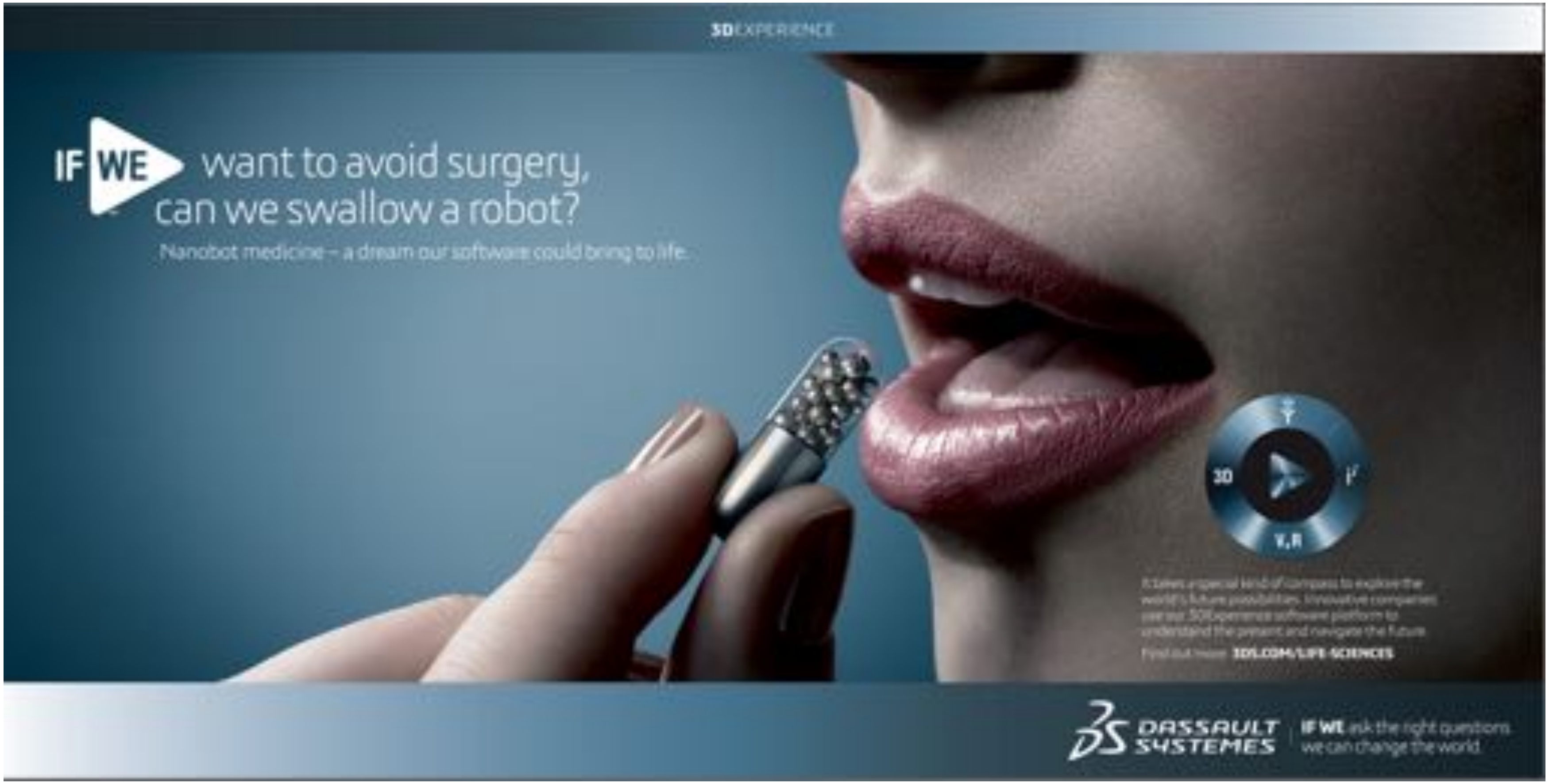
D. Araiza-Illan, A.G. Pipe, K. Eder.
Intelligent Agent-Based Stimulation for Testing Robotic Software in Human-Robot Interactions. (Proceedings of MORSE 2016, ACM, July 2016)
DOI: **10.1145/3022099.3022101** (arXiv:1604.05508)

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
  - *Specification* is essential to perform V&V
    - flexible, vague and probabilistic*
- Automation, automation, automation
- Innovation
  - Creative combinations of techniques
  - Be more clever, *use the power of AI for V&V*

* 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)



Kerstin.Eder@bristol.ac.uk

Special thanks to Greg Chance, Abanoub Ghobrial, Séverin Lemaignan, 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.