Theorem Proving and Testing for Autonomous Systems

Kerstin Eder

University of Bristol and
Bristol Robotics Laboratory

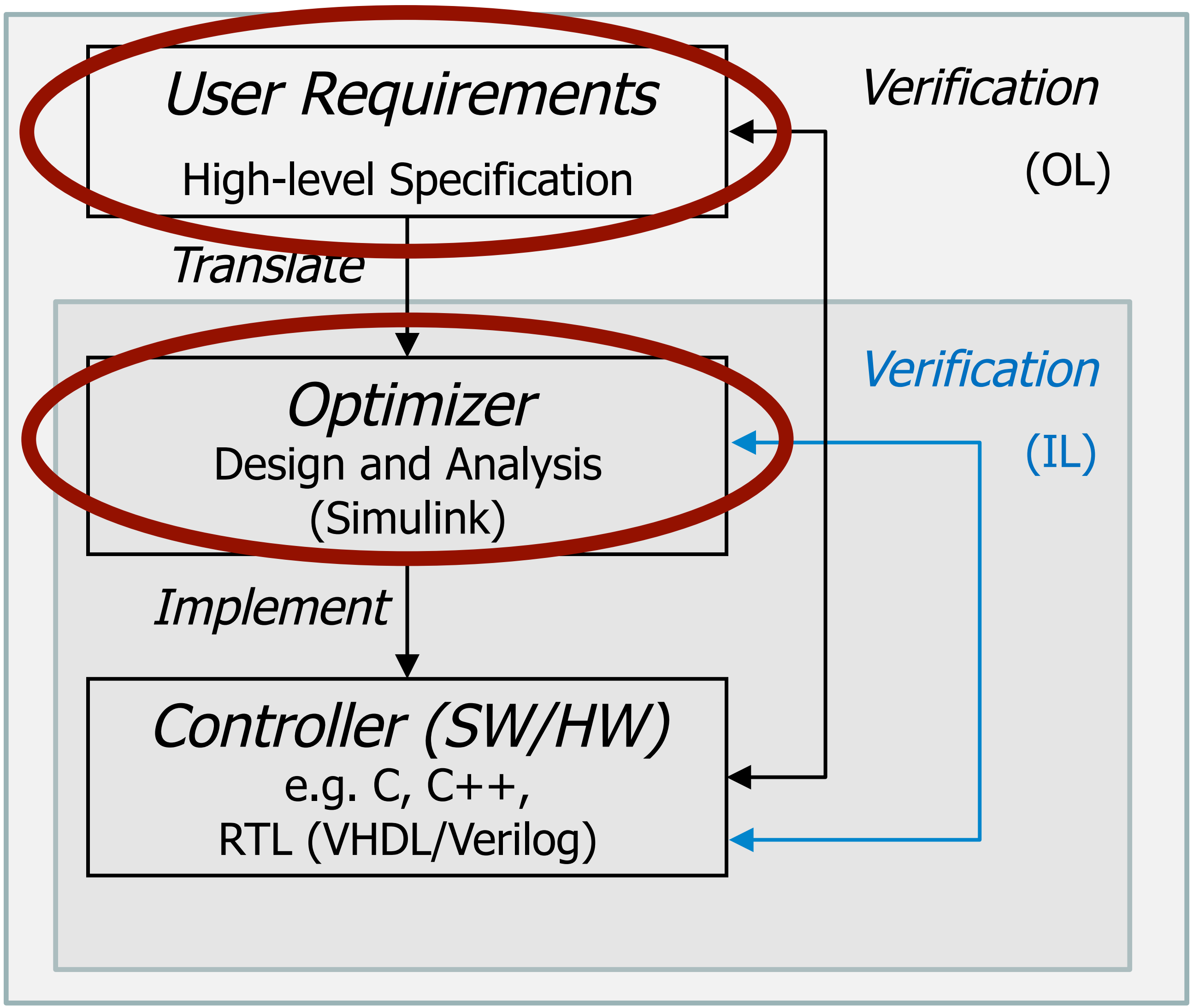
University of BRISTOL

brl Bristol Robotics Laboratory

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

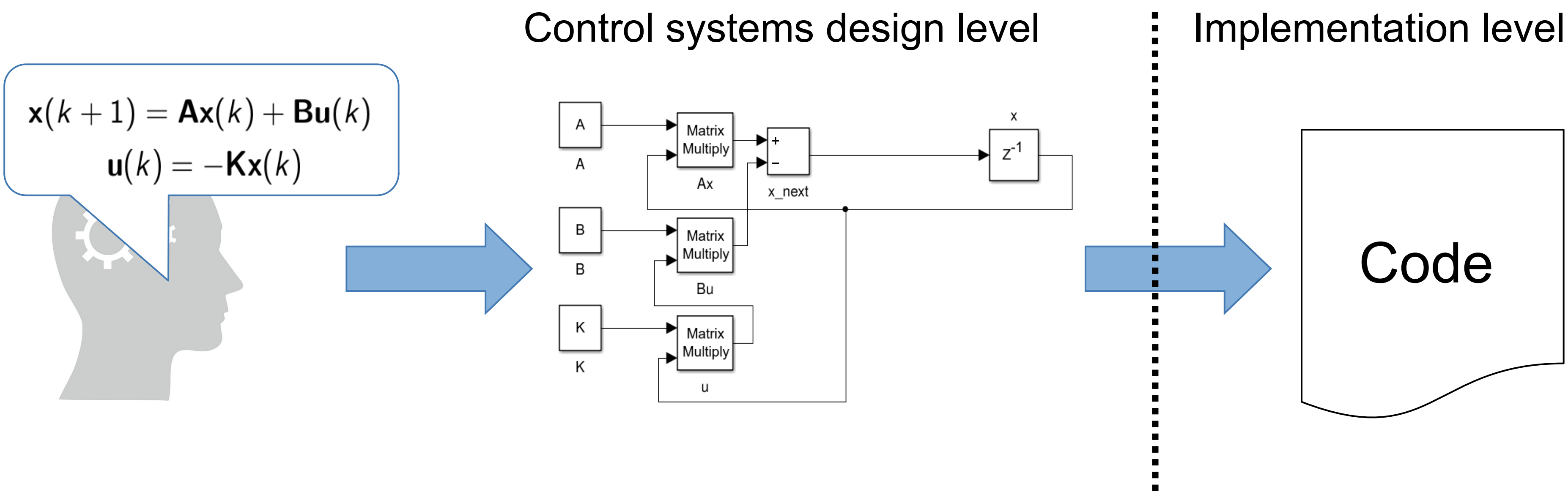


What can be done at the design level?

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. tbc. Jul 2015.
http://arxiv.org/abs/1505.05699

Simulink Diagrams in Control Systems



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

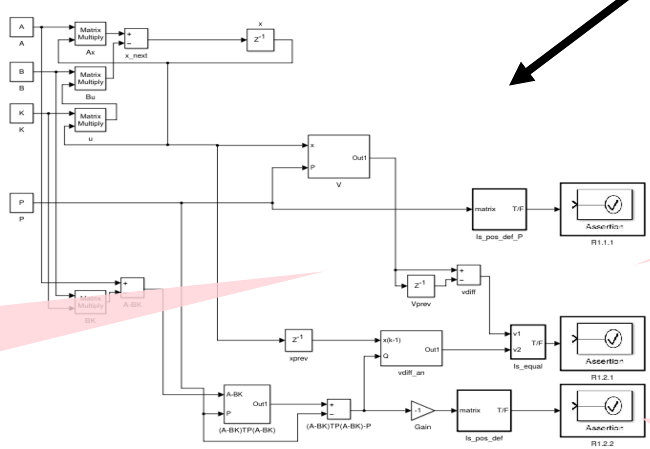
Verifying Stability

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)

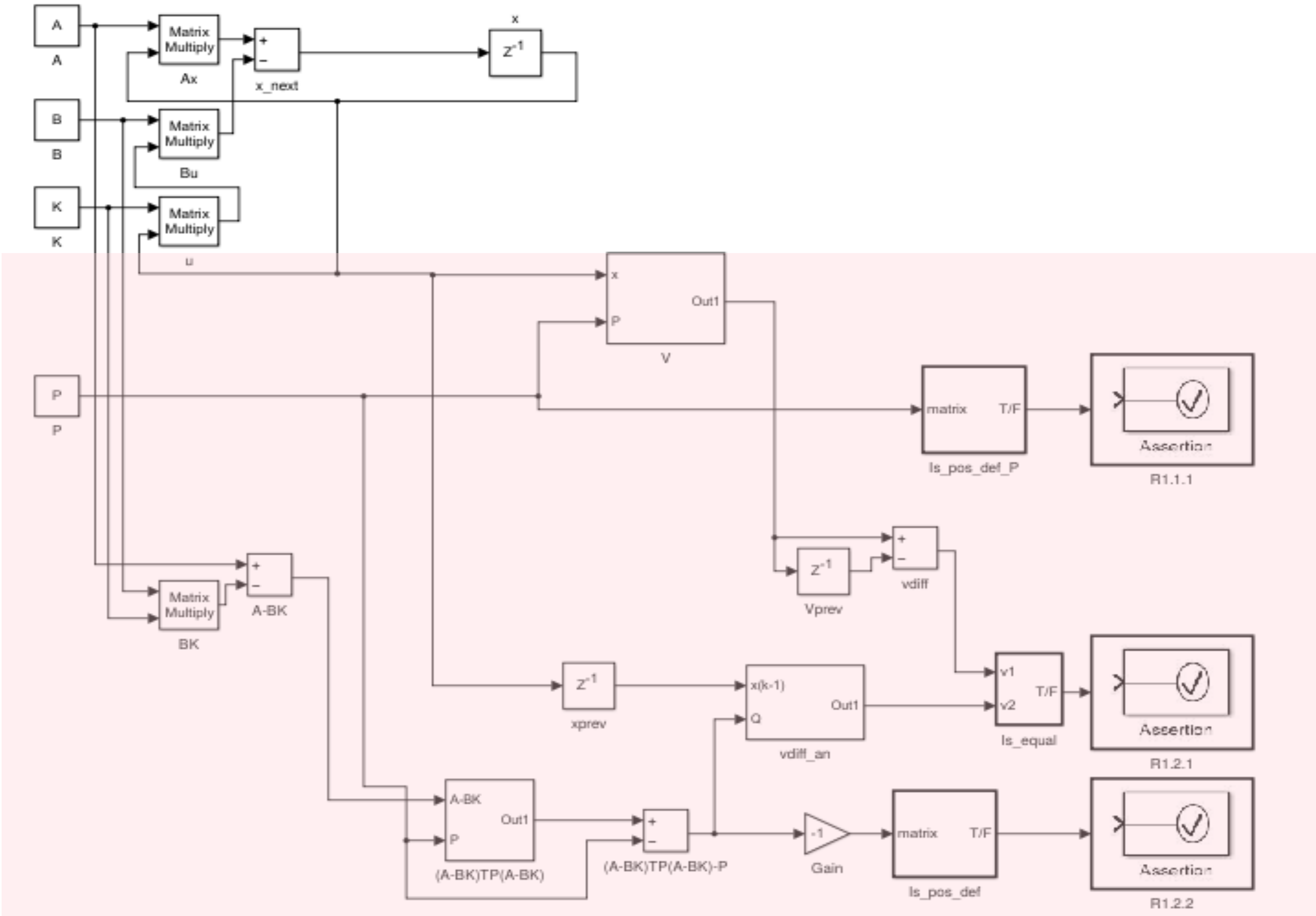


Capture control systems requirements

Add as assertions

Retain in code implementation

Assertion-Based Verification



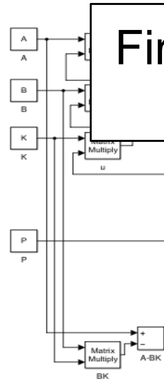
Combining Verification Techniques

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)



First order logic theory of the Simulink diagram

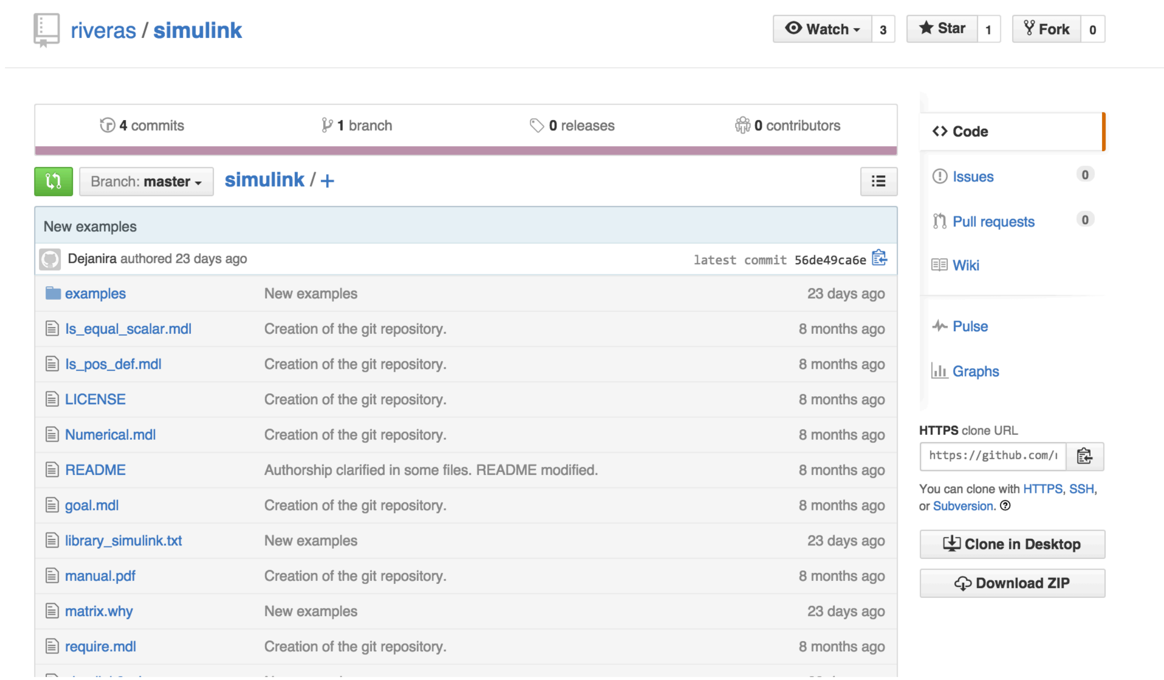
Axiom: Bu = B * u
...

…

Goal: vdiff == vdiff_an

Test in simulation

Automatic theorem proving

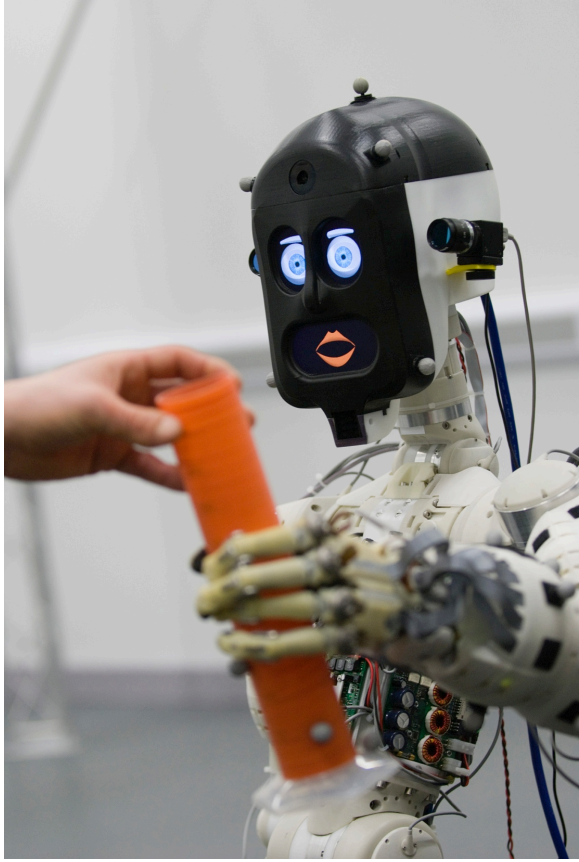


http://github.com/riveras/simulink

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. tbc. Jul 2015.
http://arxiv.org/abs/1505.05699

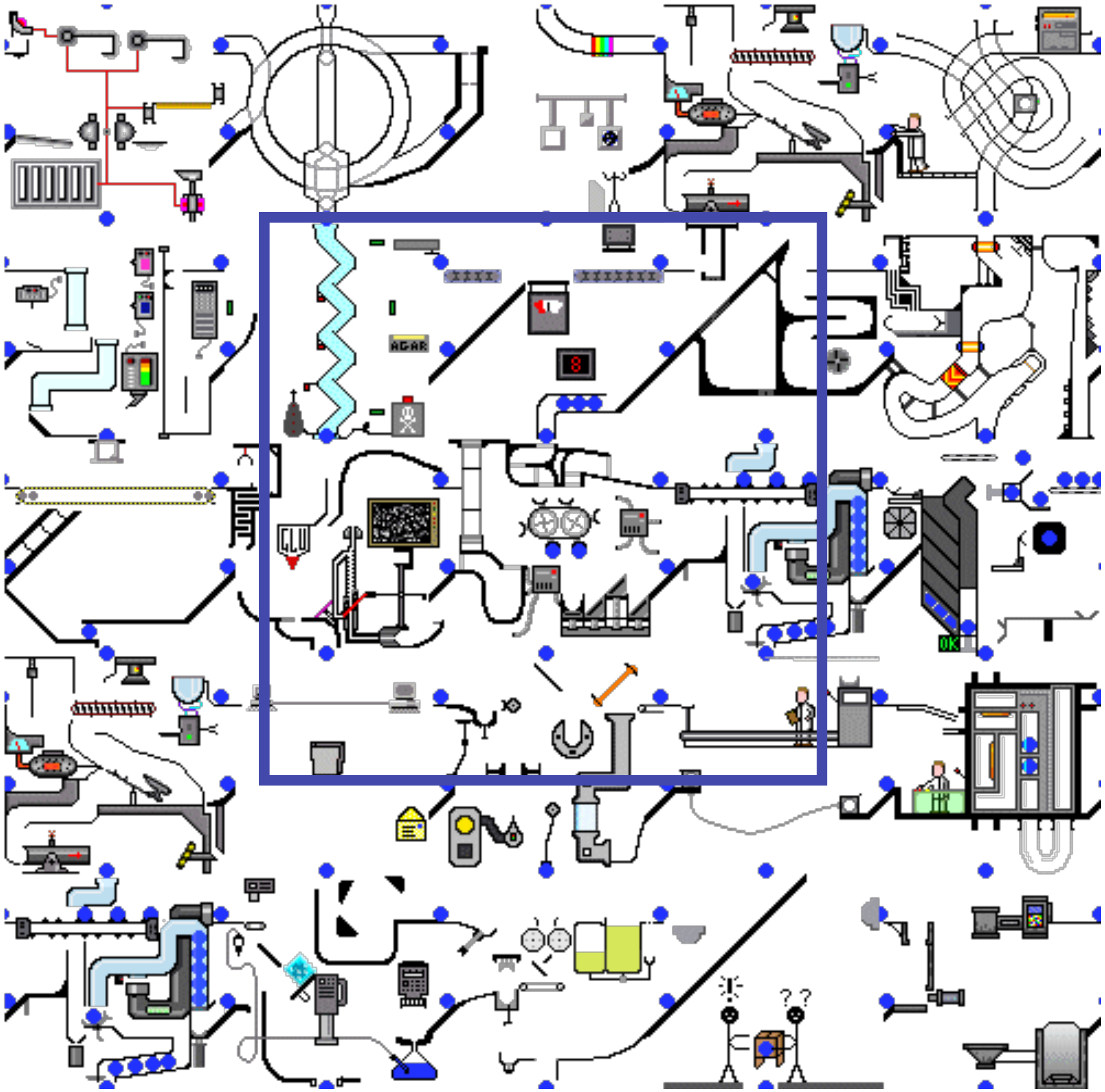
Simulation-based testing
Why and how?



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

D. Araiza Illan, D. Western, A. Pipe, K. Eder.
Model-Based, Coverage-Driven Verification and Validation of Code for Robots in Human-Robot Interactions.
(under review for publication at ICRA 2016)

System Complexity



“Model checking works best for well defined models that are not too huge.

Most of the world is thus not covered.”

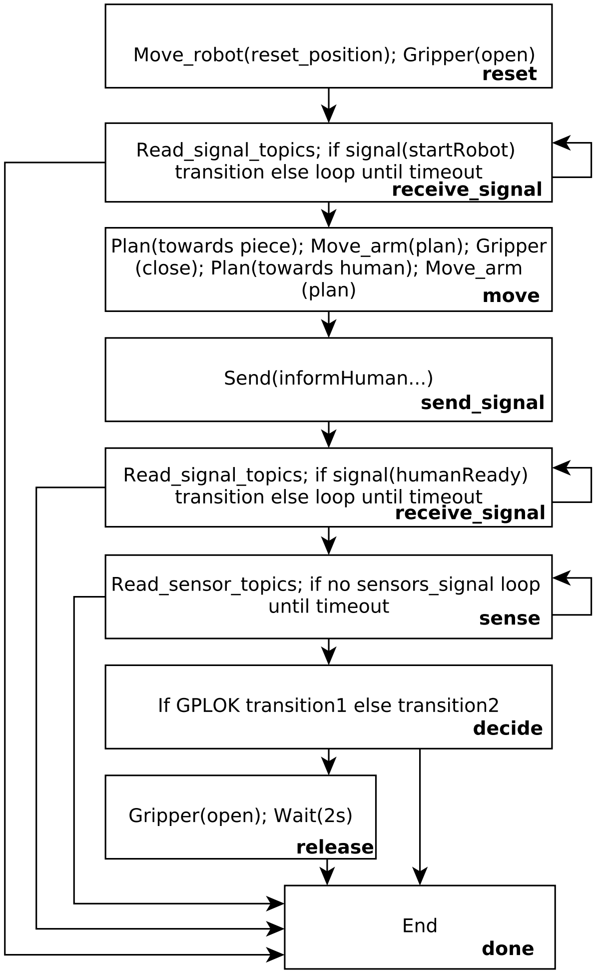
Yaron Kashai,
Fellow at the Systems and Verification R&D Division of Cadence


impossible

Coverage-Driven Verification

SUT

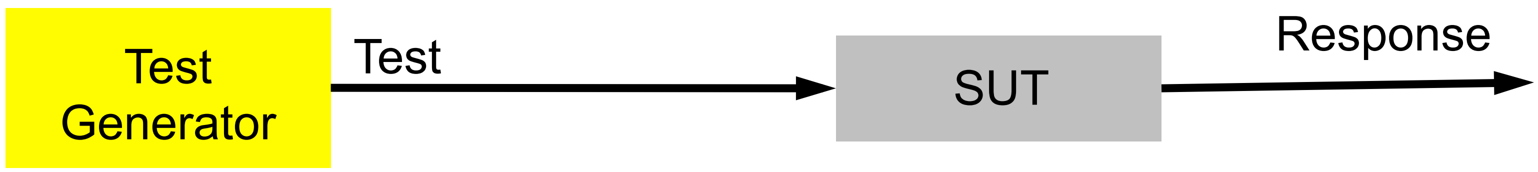
Code Structure



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

Coverage-Driven Verification

Coverage-Driven Verification



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

Test Generator

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

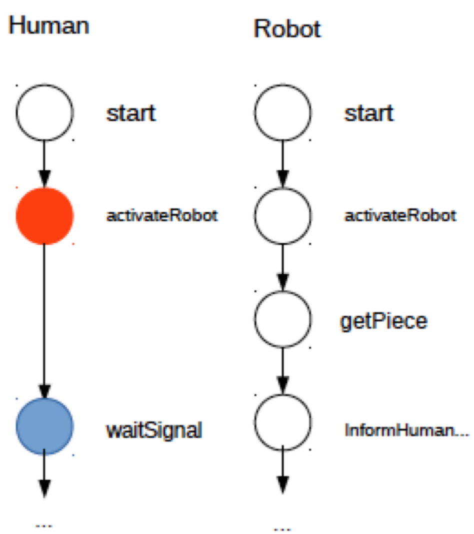
- Efficient tests:
  - minimal set of tests (regression)
- Strategies:
  - Pseudorandom (repeatability)
  - Constrained pseudorandom
  - Model-based to target specific scenarios

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

Formal model



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

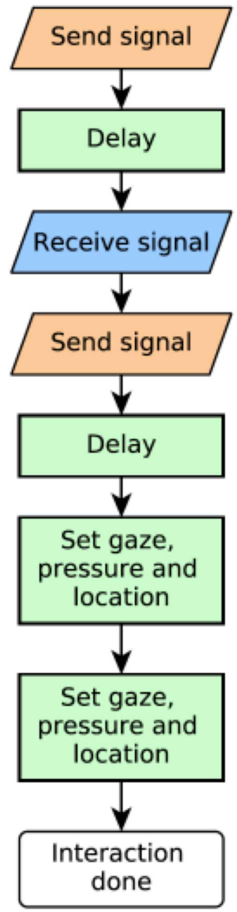
```
send_signal  activateRobot

set_param  time = 40

receive_signal  informHumanOfHandoverStart

send_signal  humanIsReady

set_param  time = 10

set_param  h_onTask = true

set_param  h_gazeOk = true
set_param  h_pressureOk = true
set_param  h_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



Parameter instantiation:

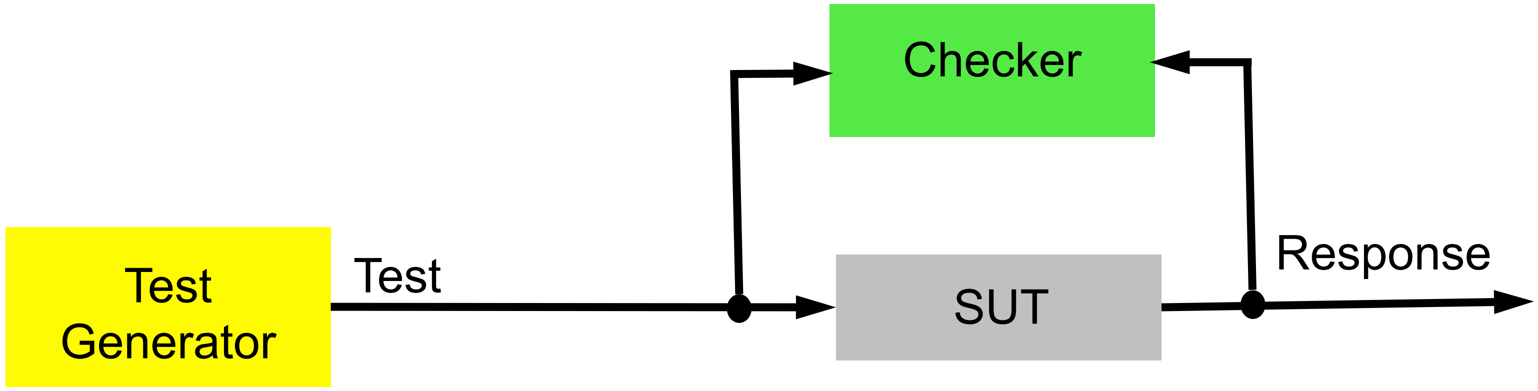
2 s

0.5 s

Gaze: (0.1 m, 0.5 m, 40 $^{\circ}$)
Location: (0.45 m, 0.05 m, 0.73 m)

Gaze: (0.1 m, 0.5 m, 30 $^{\circ}$)
Pressure: (15, 120, 140) to (7, 90, 100)
Location: (0.45 m, 0.05 m, 0.73 m)

Coverage-Driven Verification

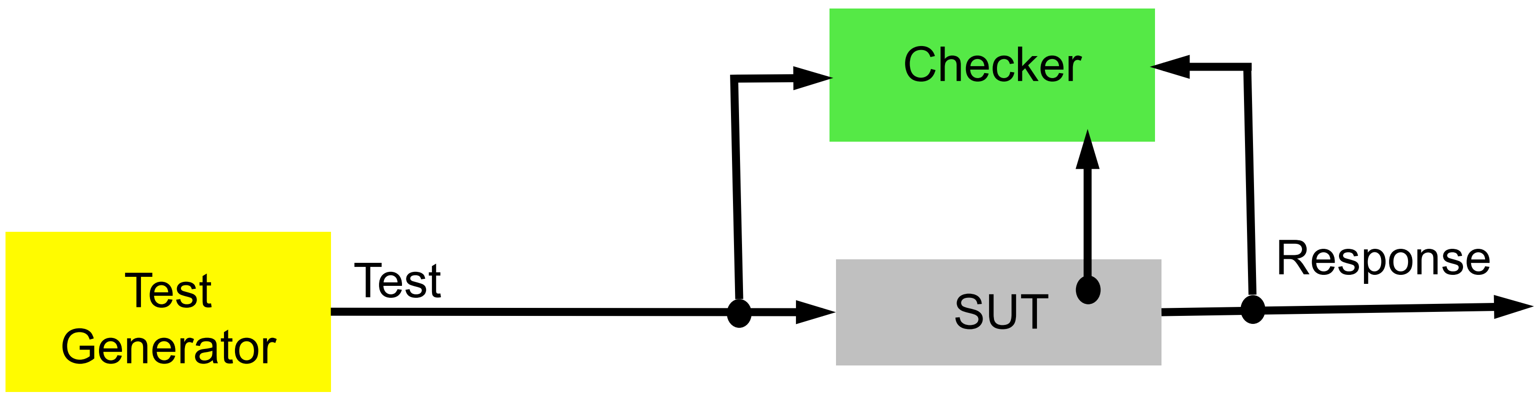


Checker

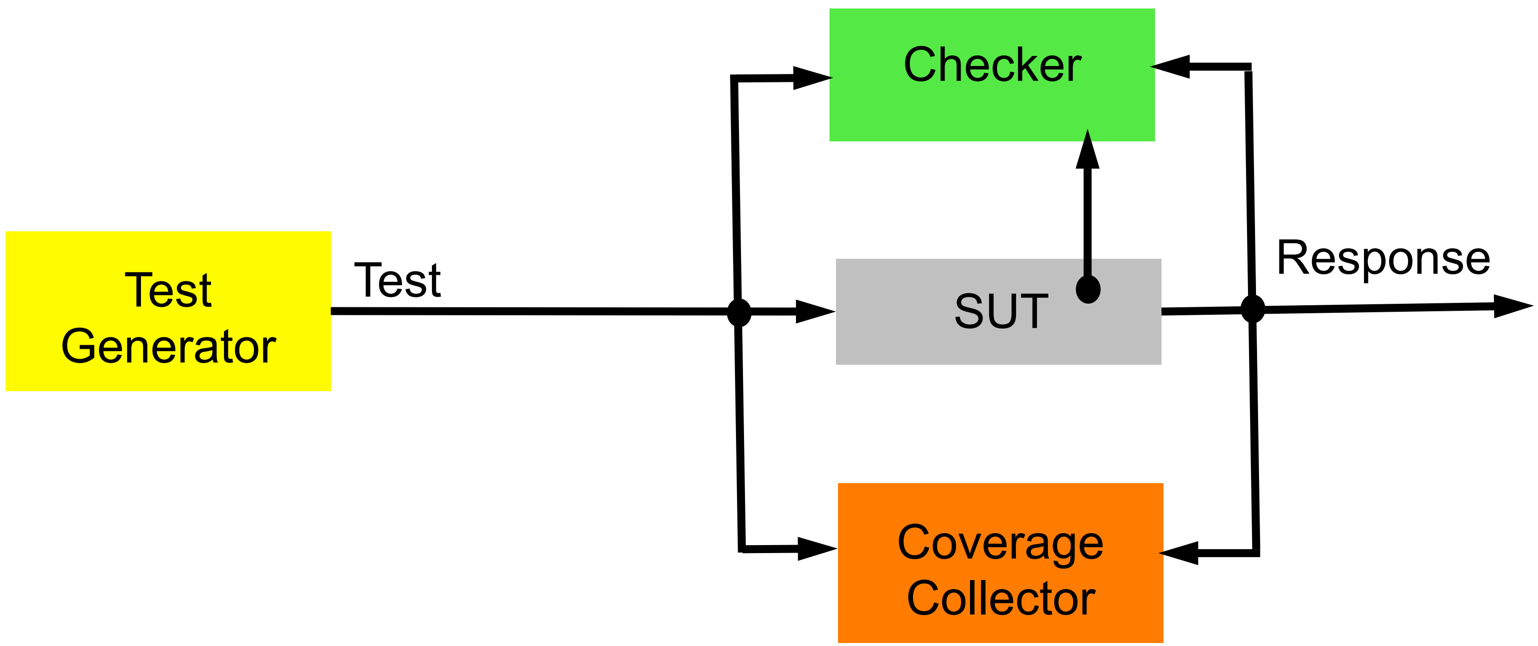
- Requirements as assertions monitors:
  - `if [precondition], check [postcondition]`
  - *“If the robot decides the human is not ready, then the robot never releases an object”.*
  - Implemented as automata
- Continuous monitoring at runtime, self-checking
  - High-level requirements
  - Lower-level requirements depending on the simulation's detail (e.g., path planning, collision avoidance).

```
assert {robot_3D_space != human_3D_space}
```

Coverage-Driven Verification

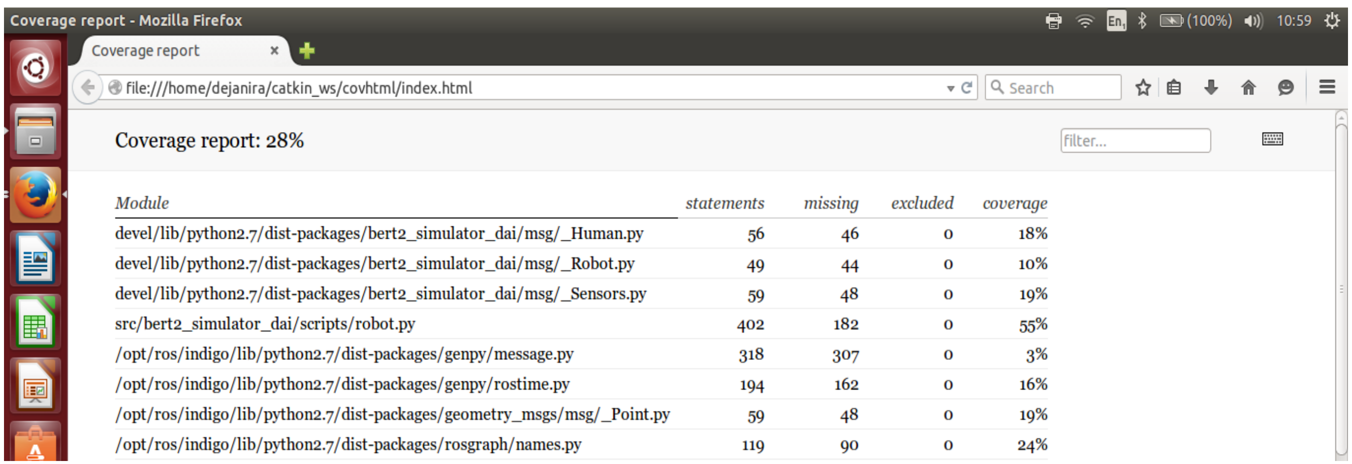


Coverage-Driven Verification



Coverage Collector

- Coverage models:
  - Code coverage from statement to MC/DC
    - e.g., using the 'coverage' modules in Python



Coverage report - Mozilla Firefox

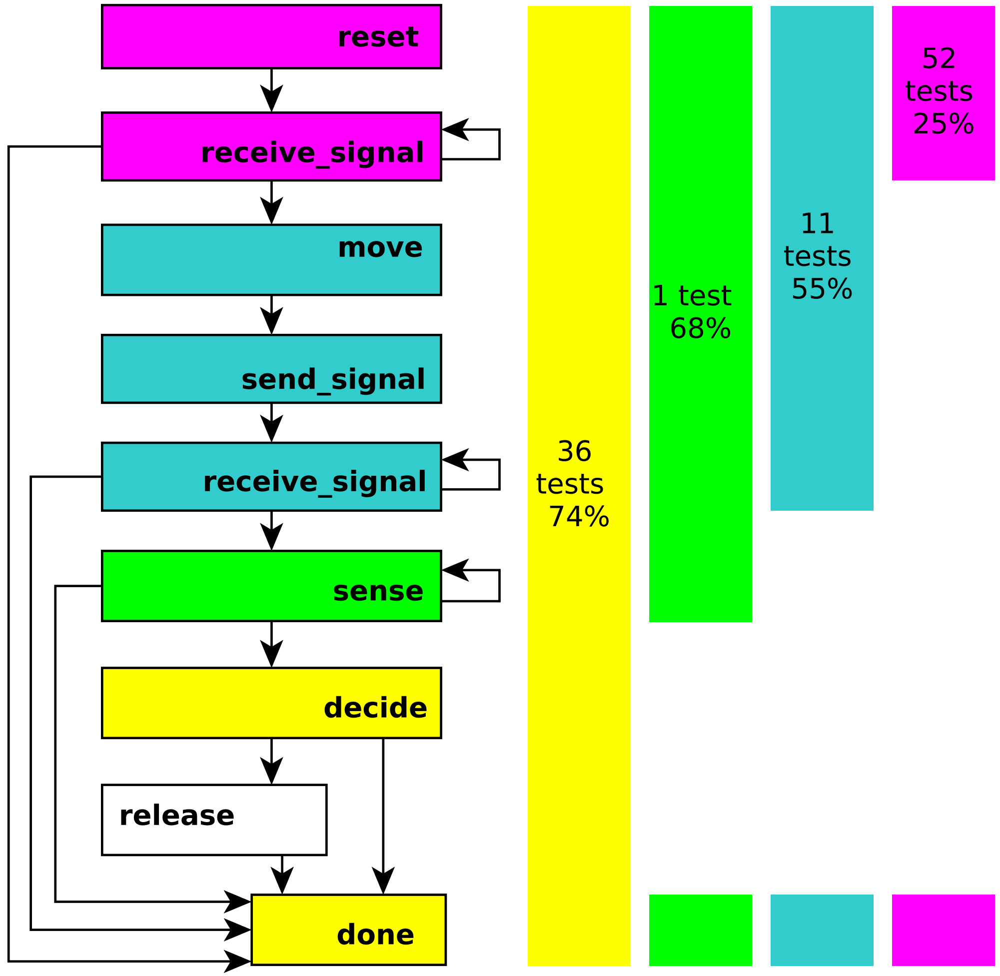
file:///home/dejanira/catkin_ws/covhtml/index.html

Coverage report: 28%

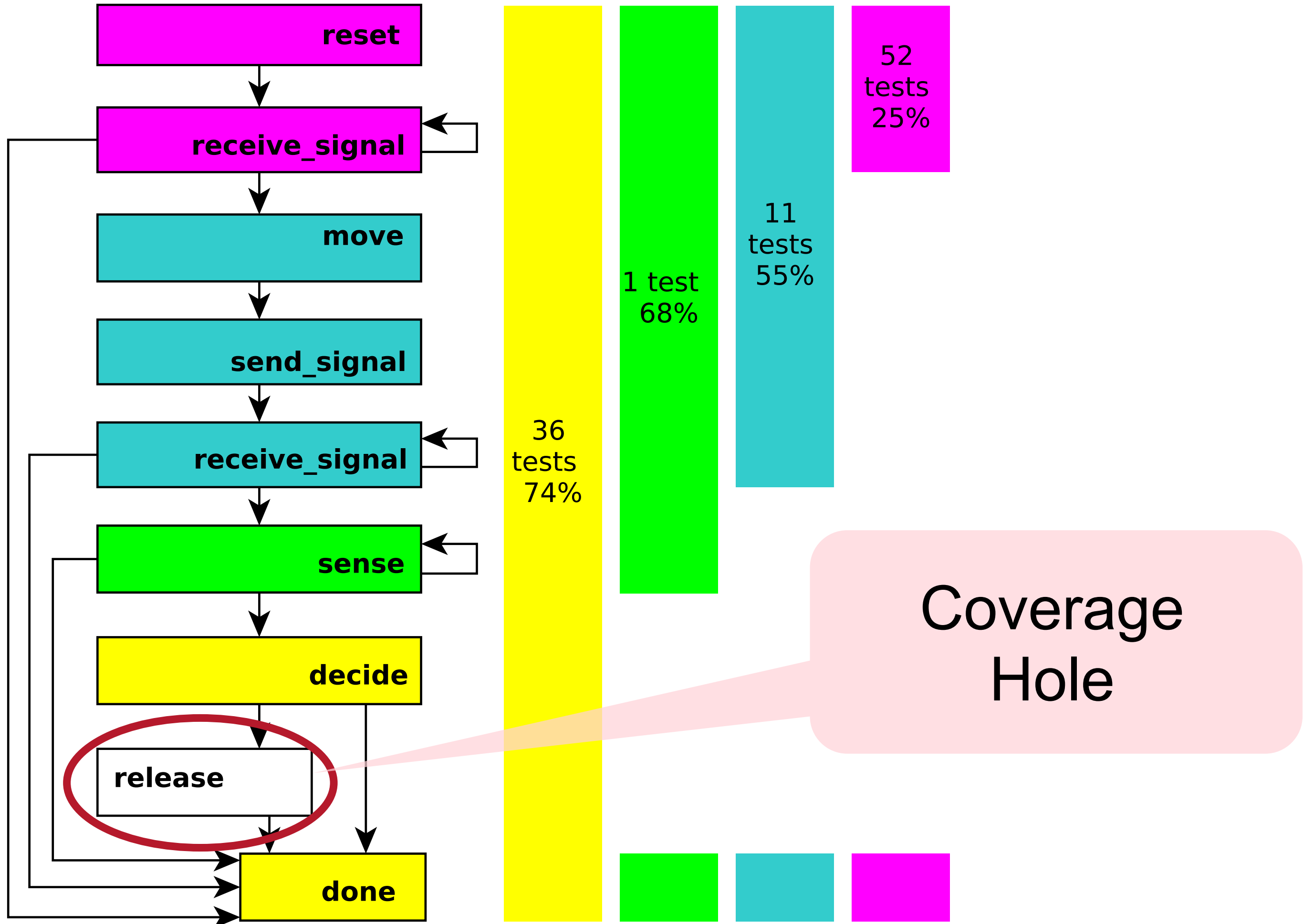
Module	*statements*	*missing*	*excluded*	*coverage*
devel/lib/python2.7/dist-packages/bert2_simulator_dai/msg/_Human.py	56	46	0	18%
devel/lib/python2.7/dist-packages/bert2_simulator_dai/msg/_Robot.py	49	44	0	10%
devel/lib/python2.7/dist-packages/bert2_simulator_dai/msg/_Sensors.py	59	48	0	19%
src/bert2_simulator_dai/scripts/robot.py	402	182	0	55%
/opt/ros/indigo/lib/python2.7/dist-packages/genpy/message.py	318	307	0	3%
/opt/ros/indigo/lib/python2.7/dist-packages/genpy/rostime.py	194	162	0	16%
/opt/ros/indigo/lib/python2.7/dist-packages/geometry_msgs/msg/_Point.py	59	48	0	19%
/opt/ros/indigo/lib/python2.7/dist-packages/rosgraph/names.py	119	90	0	24%

  - Structural coverage
    - e.g., FSM coverage

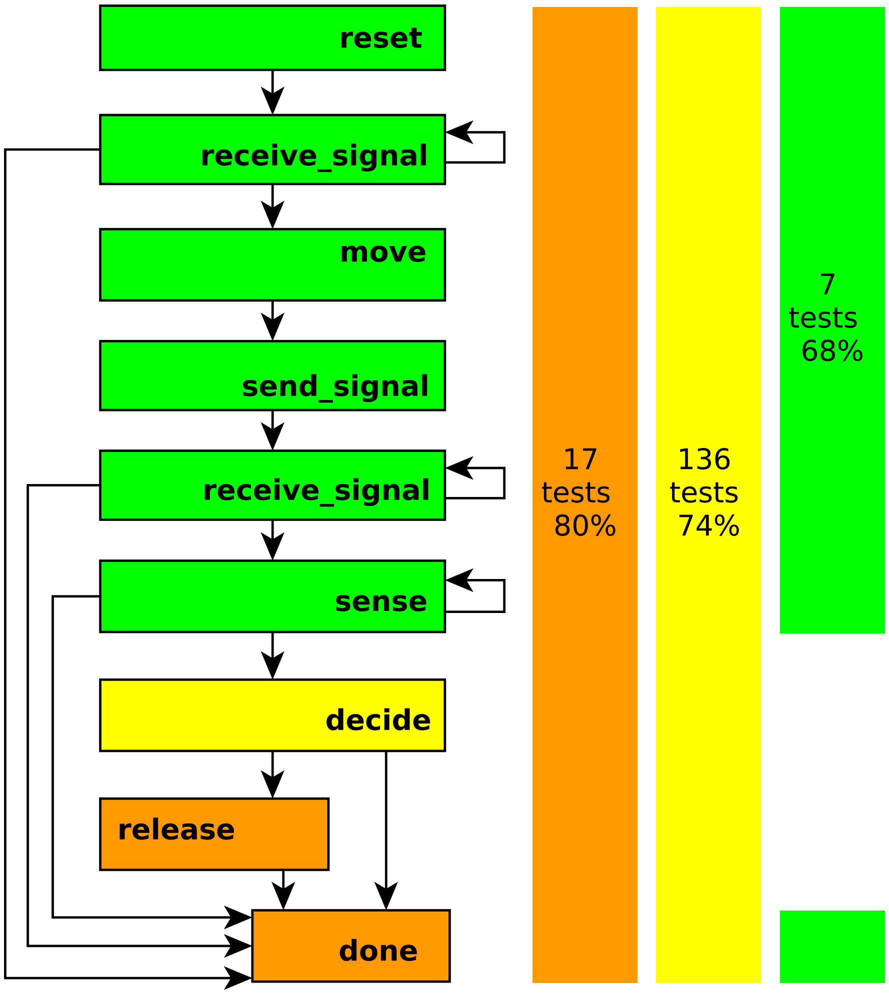
Coverage of 100 pseudornd Tests



Coverage of 100 pseudornd Tests



Coverage of 160 MB Tests



Functional Coverage

- Requirements coverage
- "Cross-product" coverage

[O Lachish, E Marcus, S Ur and A Ziv. Hole Analysis for Functional Coverage Data. Design Automation Conference (DAC), June 10-14, 2002, New Orleans, Louisiana, USA.]

A cross-product coverage model is composed of the following parts:

1. A semantic **description** of the model (story)
2. A list of the **attributes** mentioned in the story
3. A set of all the **possible values** for each attribute (the attribute value **domains**)
4. A list of **restrictions** on the legal combinations in the cross-product of attribute values

A **functional coverage space** is defined as the Cartesian product over the attribute value domains.

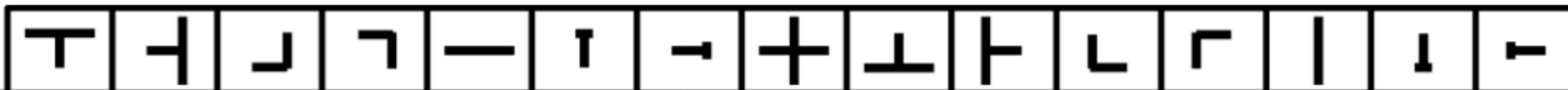
Cross-Product Models in *e*

Verification Languages, such as *e*, support cross-product coverage models natively.

```
(ADD, 00000000)
(ADD, 00000001)
(ADD, 00000010)
(ADD, 00000011)
…
(XOR, 11111110)
(XOR, 11111111)
```

```
struct instruction {
   opcode: [NOP, ADD, SUB, AND, XOR];
   operand1 : byte;
   event stimulus;
   cover stimulus is {
      item opcode;
      item operand1;
      cross opcode, operand1
         using ignore = (opcode == NOP);
   };
};
```

Situation Coverage



Car															
Bike															
HGV															
Ped															

Situation coverage – a coverage criterion for testing autonomous robots

Rob Alexander*, Heather Hawkins*, Drew Rae [†]

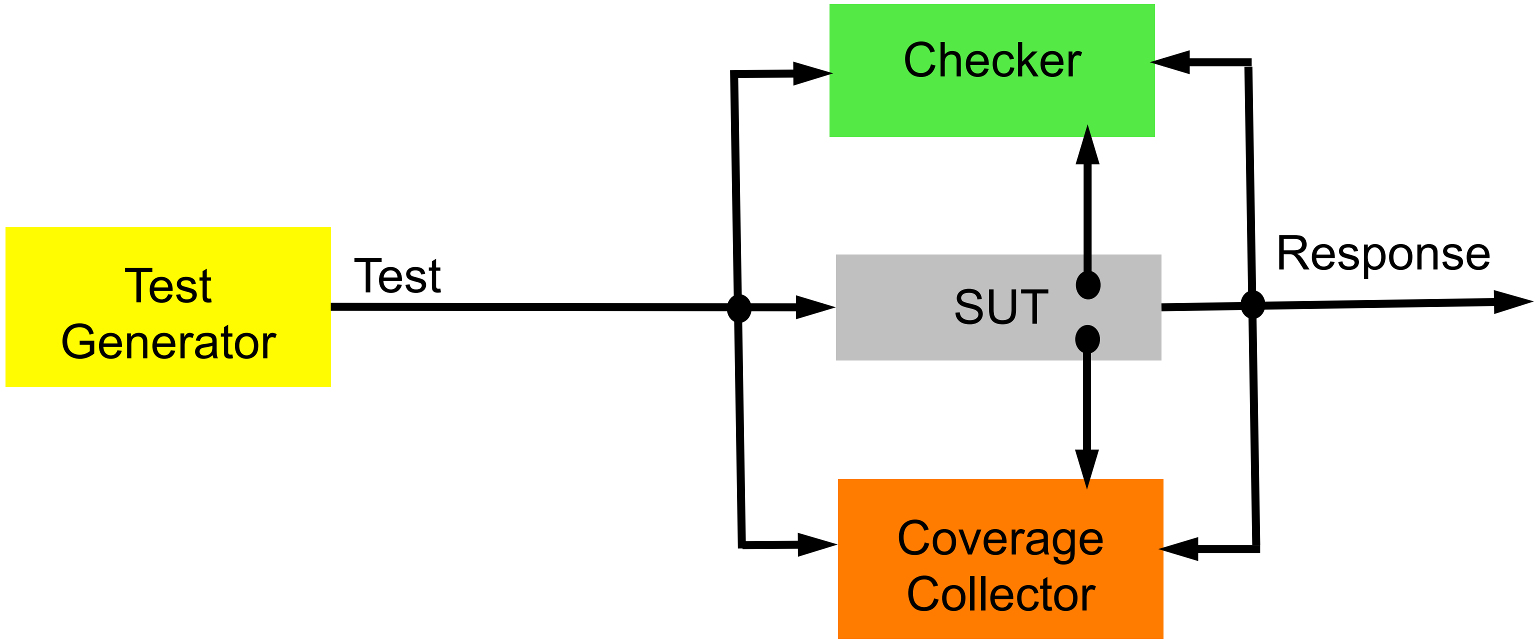
* *University of York, York, United Kingdom*

[†] *Griffith University, Brisbane, Australia*

rob.alexander@york.ac.uk

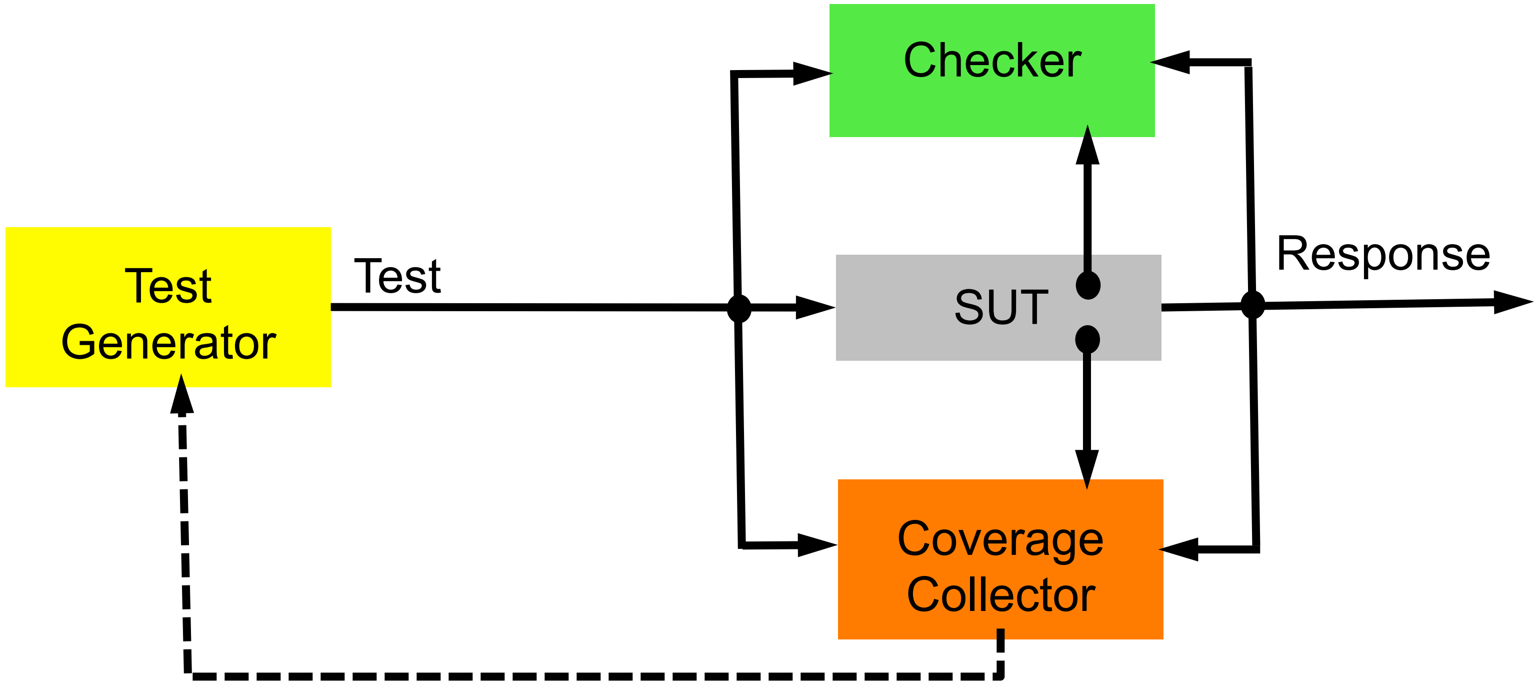
Coverage-Driven Verification

Coverage analysis enables feedback to test generation

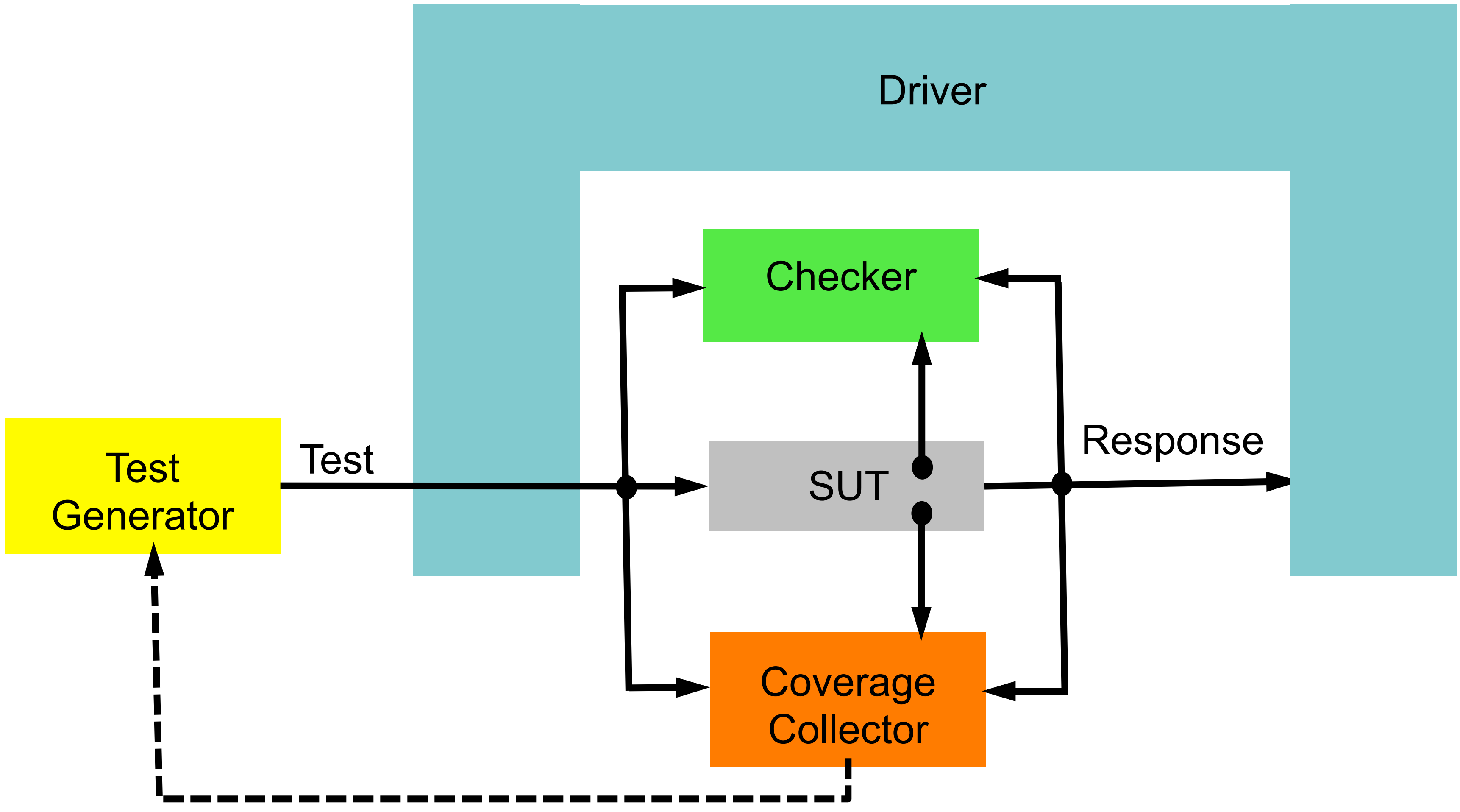


Coverage-Driven Verification

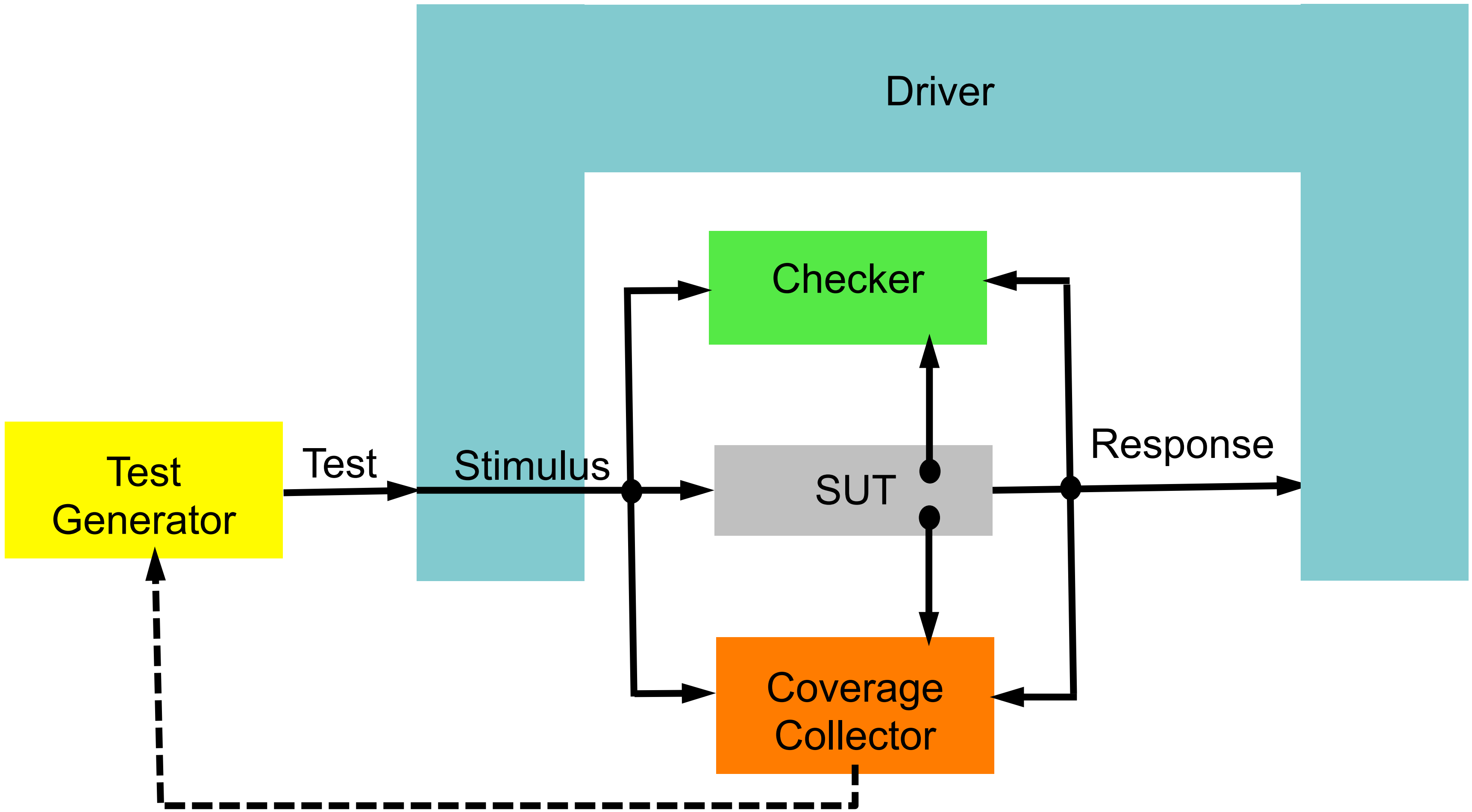
Coverage analysis enables feedback to test generation



Stimulating the SUT

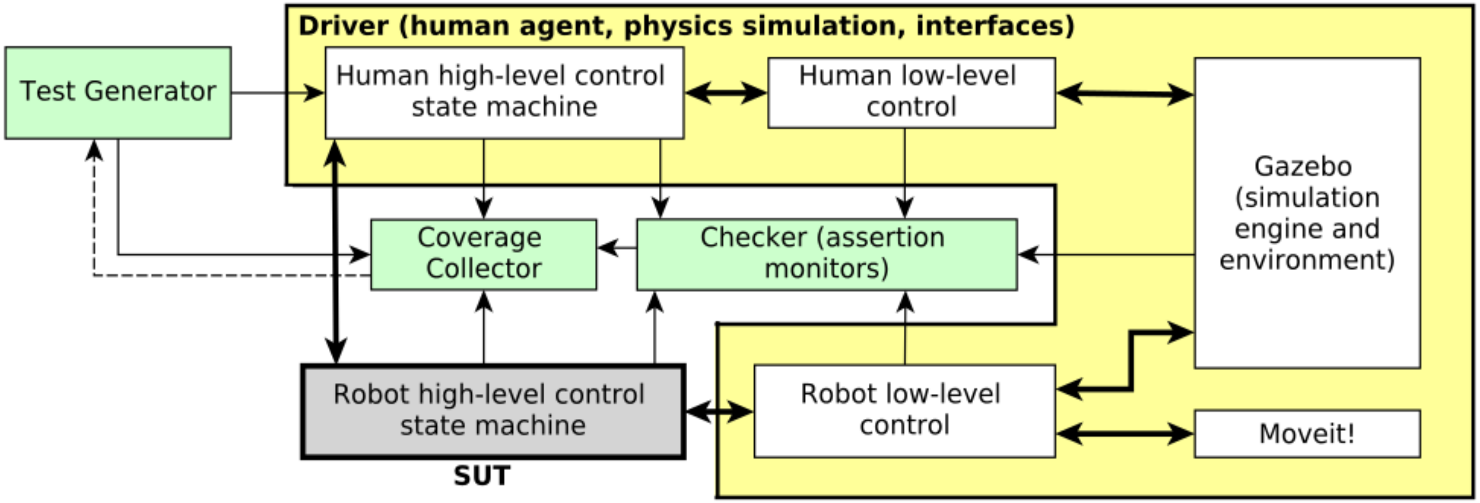


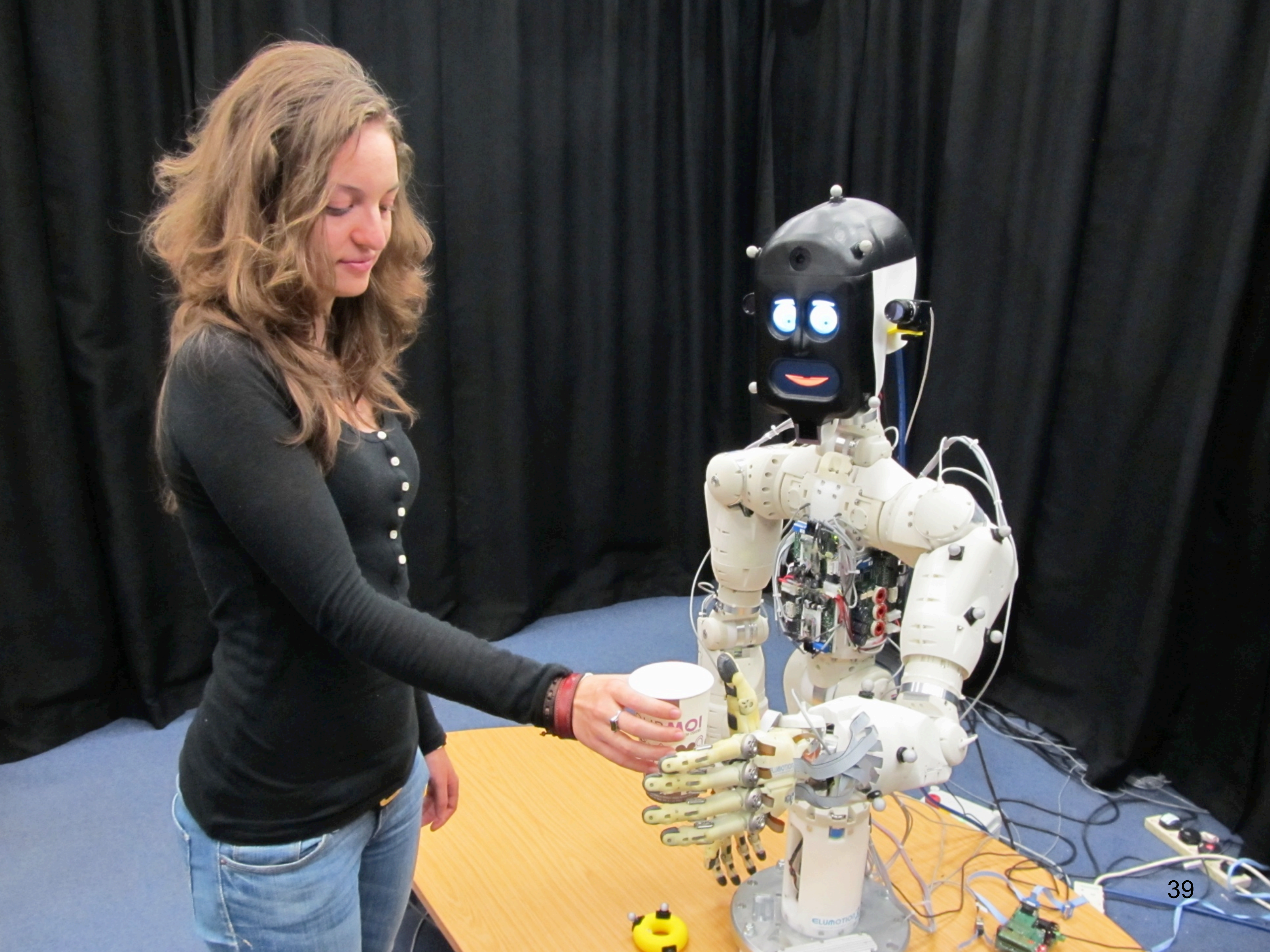
Stimulating the SUT



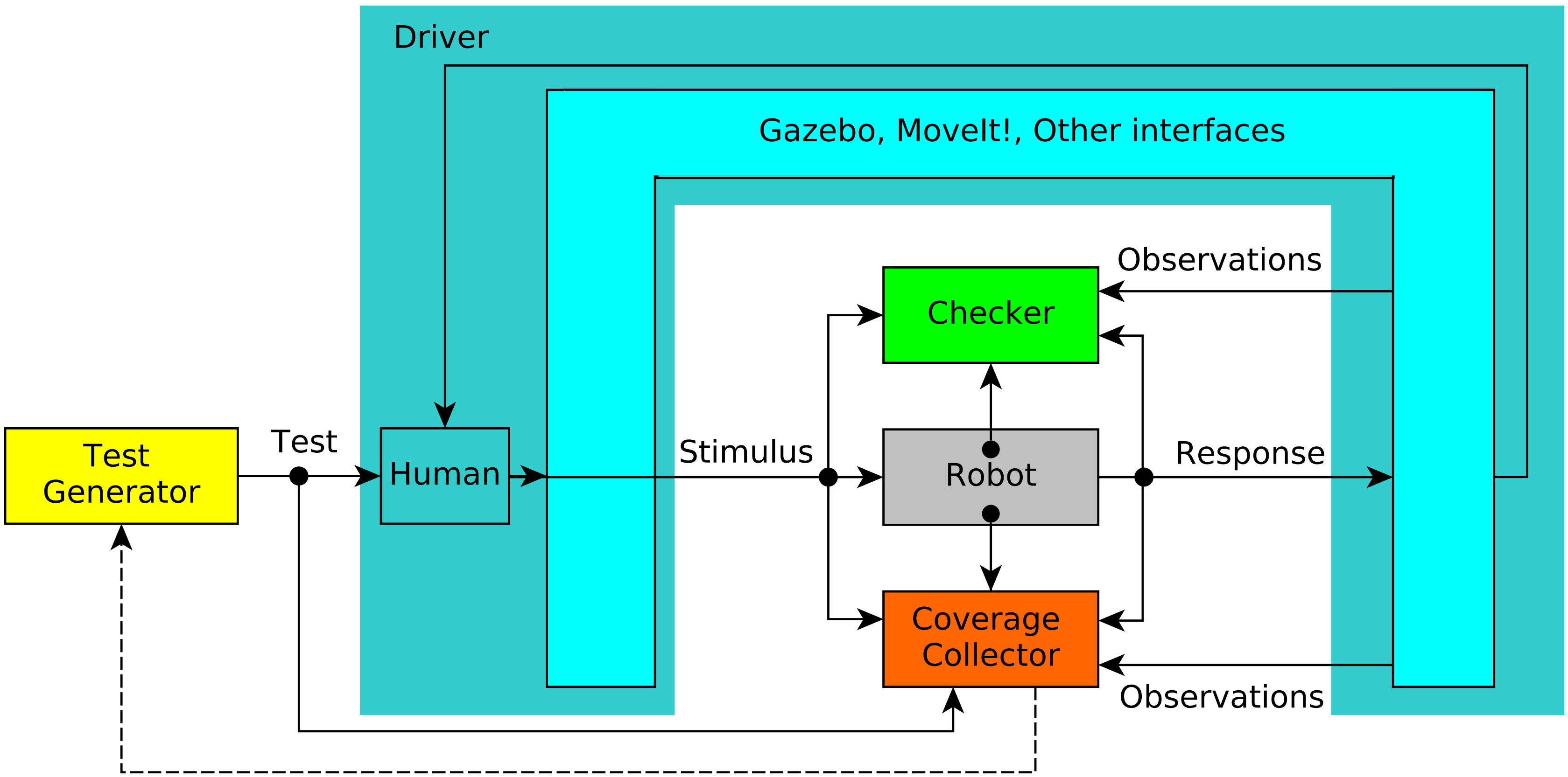
Driver

- Environmental components (models) interacting with the system's control software
- Examples: humans, actuators (Gazebo), communication signals, sensors





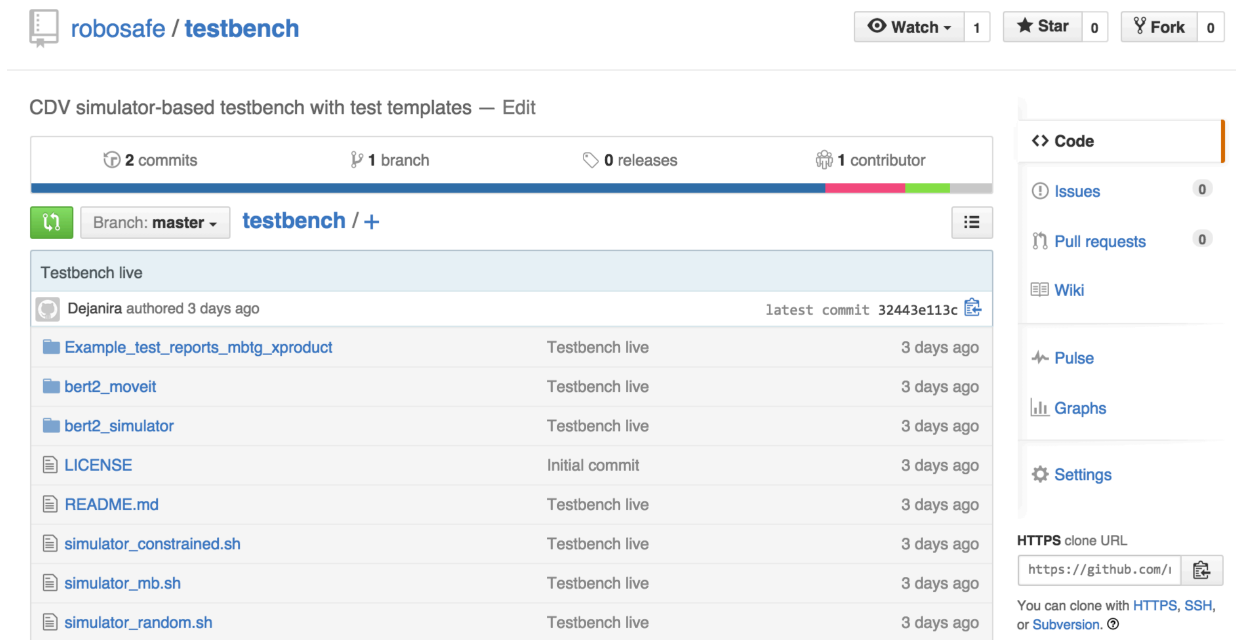
CDV for Human-Robot Interaction



D. Araiza Illan, D. Western, A. Pipe, K. Eder. **Model-Based, Coverage-Driven Verification and Validation of Code for Robots in Human-Robot Interactions.** (under review for publication at ICRA 2016)

Coverage-Directed Verification

- systematic, goal directed simulation-based V&V
- 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



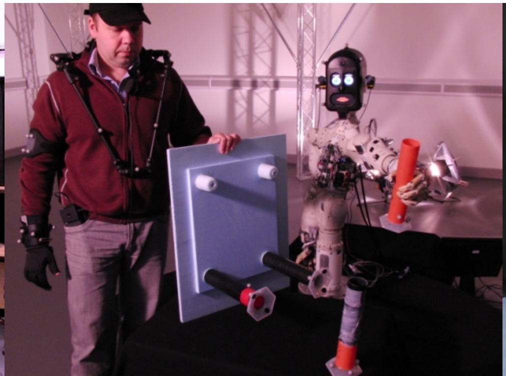
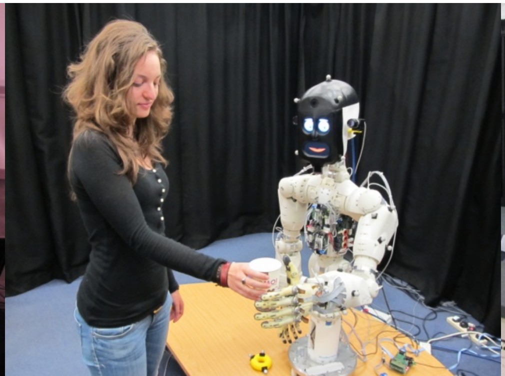
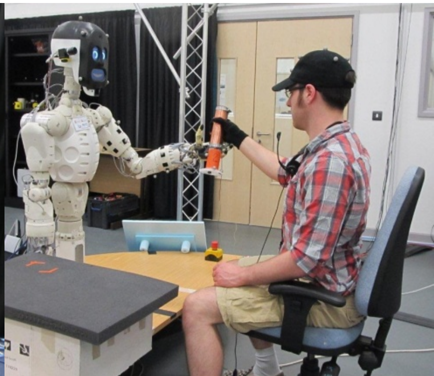
http://github.com/robosafe/testbench

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

D. Araiza Illan, D. Western, A. Pipe, K. Eder.
Model-Based, Coverage-Driven Verification and Validation of Code for Robots in Human-Robot Interactions.
(under review for publication at ICRA 2016)

Summary

- No single technique is adequate for an entire design/system in practice.
- Verification techniques can be combined.
- Learn from areas where verification techniques are mature.
- We need to design *for* verification.

Thank you

Engineering and Physical Sciences Research Council

Any questions?

Kerstin.Eder@bristol.ac.uk

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

University of BRISTOL

Bristol Robotics Laboratory