Keyword-Driven Simulation Testing for Open-Source Robotics

Niels Hoppe

13/09/2022
Agenda

1) Motivation
   a) Open-source Robotics
   b) Acceptance Testing
   c) Keyword-driven Testing
2) Challenges (and solutions)
   1) Test Adaptation
   2) Time
   3) Space
   4) Continuous signals
3) Conclusion
Motivation: Open-Source Robotics

Robot Operating System (ROS)
- Message-based middleware
- Nodes provide and call services
- Nodes publish and subscribe to topics

Gazebo simulator
- 3D simulator for ROS
- Customizable through plugins
Testing of Trustworthy Systems

Motivation: Acceptance Testing

Our motivation and goals for acceptance testing:
- Lower test levels already covered for ROS (e.g. rostest for integration tests)
- Enable domain experts to write and execute test cases
- Automate test execution in simulation
- Transfer test cases from simulation to real world
Motivation: Acceptance Testing

In an acceptance test we want to ascertain that

- some things happen
  - e.g., objects, in particular robots, reach certain positions and orientations
  - in a specific timeframe
  - in a specific order
- other things DO NOT happen
  - e.g., collisions between objects, in particular robots
- certain properties hold
  - e.g., distances between objects, alignment of objects
  - remain static or
  - follow a specific course
Motivation: Acceptance Testing
Keyword-driven testing (KDT)

- Test cases are sequences of test steps
- Test steps are expressed through keywords
- Good for interactive / event-driven / request-response systems
  - e.g., user interfaces, apps, websites, communication protocols, ...
- Human-readable keywords well understood by domain experts
- Different frameworks exist
  - e.g., Cucumber, Robot, ...
Challenges

When applying KDT to robotics and simulation, we faced challenges with

- Test Adaptation: defining and accessing the test interface
- Time: simulation time vs. wall-clock time
- Space: position and orientation of objects
- Continuous signals: stimuli and observations
Test Adaptation

Custom library for the Robot Framework

- Convenience keywords for ROS
  - Launch ROS launch configurations (roslaunch) and run ROS scripts (rosrun)
  - Read and write ROS parameters (rosparam)
  - Call ROS services (rosservice)

- Gazebo specific keywords
  - Start, pause, reset simulation
  - Spawn, delete, inspect objects

- More to come, available from GitHub:
  - hielsnoppe/robotframework-rosgazebolibrary
Time

There is a disparity between simulation time and wall-clock time
- Expressed as the real-time factor (RTF) in Gazebo
- The RTF fluctuates over time
- RTF often < 1.0 due to low performance / high load
- RTF > 1.0 when simulating at an accelerated tempo (on high performance device)

Possible solutions:
- Check simulation time in a loop (naïve busy waiting)
- Adaptive timeouts and intervals (less busy waiting, but still...)
- Simulator plugin for timeouts and intervals
Time: Simulator Plugin

**Simulator plugin**
- Advertise four ROS services:
  - Set timeout (duration): timeout handle
  - Clear timeout (timeout handle)
  - Set interval (duration): interval handle
  - Clear interval (interval handle)
- Advertise one ROS topic:
  - /timeouts_intervals
- Publish respective handle whenever a timeout or interval is due

**Keyword library**
- Subscribe to topic

**Keywords**
- Wait \{duration\}
  - Call set timeout (duration)
  - Proceed when receiving handle
- Repeat Every \{duration\}
  - Call set interval (duration)
  - Perform action when receiving handle
Check position and orientation of objects

- Absolute (e.g., moving robots)
- Relative to other objects (e.g., distances, alignments, collisions)

How to implement?

- Service to request absolute object positions exists in Gazebo
- `tf2` library calculates relative positions and orientations
- `static_transform_publisher` from `tf2_ros` monitors and publishes relative positions and orientations
- Run such node for every watched relative position and subscribe to topic

Image source: http://wiki.ros.org/tf2
Collisions

Keywords
- Ignore Collision [qualifier]?
- Fail On Collision [qualifier]?
- Log Collision [qualifier]? [As {level}]?
- Expect Collision [qualifier]
- Where [qualifier] is
  - Between {group of links}
  - Involving {group of links}

How to implement?
- Internal topic for collisions exists in Gazebo
- Create Gazebo plugin to publish topic externally
- Subscribe to topic, set listeners for conditions according to keywords
Continuous stimuli and observations

How to express (continuous) change over time in a sequence of keywords?

- Sample stimuli from mathematical functions
- Trace observed properties and evaluate later
- Register watchers on observed properties and react to specific events
Continuous stimuli

Keywords

- Sample \{signal\} From \{function\} At \{interval\}
- Where \{function\} is a function (t: Time) \rightarrow \text{Any, e.g.,}
  - Step: jump to value
  - Impulse: jump to value and back
  - Ramp (linear, sinus): transition to value over time
  - Periodic: modulate signal periodically
  - Custom functions?
- Inspired by MTCD from Model Engineering Solutions

How to implement?

- Custom ROS node or
- parallel thread in test execution
- To be determined!
Continuous observations

Keywords
- Trace \{expression\}: \{watcher handle\}
- Log \{condition\} As \{level\}: \{watcher handle\}
- Fail On \{condition\}: \{watcher handle\}
- Expect \{condition\}: \{watcher handle\}
- Relieve Watcher \{watcher handle\}

Where
- \{expression\} is a function \((s_1: \text{Signal}, \ldots, s_N: \text{Signal}) \rightarrow \text{Any}\)
- \{condition\} is a function \((s_1: \text{Signal}, \ldots, s_N: \text{Signal}) \rightarrow \text{Boolean}\)

How to implement?
- Subscribe to respective topics
- Evaluate expressions and conditions on every update
- Unsubscribe topics when watchers are relieved
Keyword-driven Simulation Testing for Open-Source Robotics

- Basic functionality provided by open-source library
- Some aspects benefit from simulator plugins, e.g.,
  - Simulation time-based timeouts and intervals
  - Spatial relationships via tf2 transforms
- Advanced features still experimental, e.g.,
  - Collision checking
  - Continuous stimuli and observations
Any further questions?

Contact me:
niels.hoppe@fokus.fraunhofer.de