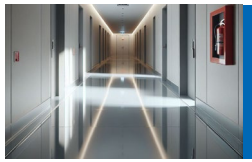
Scenario Execution for Robotics

Frederik Pasch, Research Engineer @ Intel

Testing Approaches for Robotic Systems

Open-Loop Test

Function-Level



Component-Level

Tooling available

- GTest
- Pytest

Closed-Loop Test

Simulation

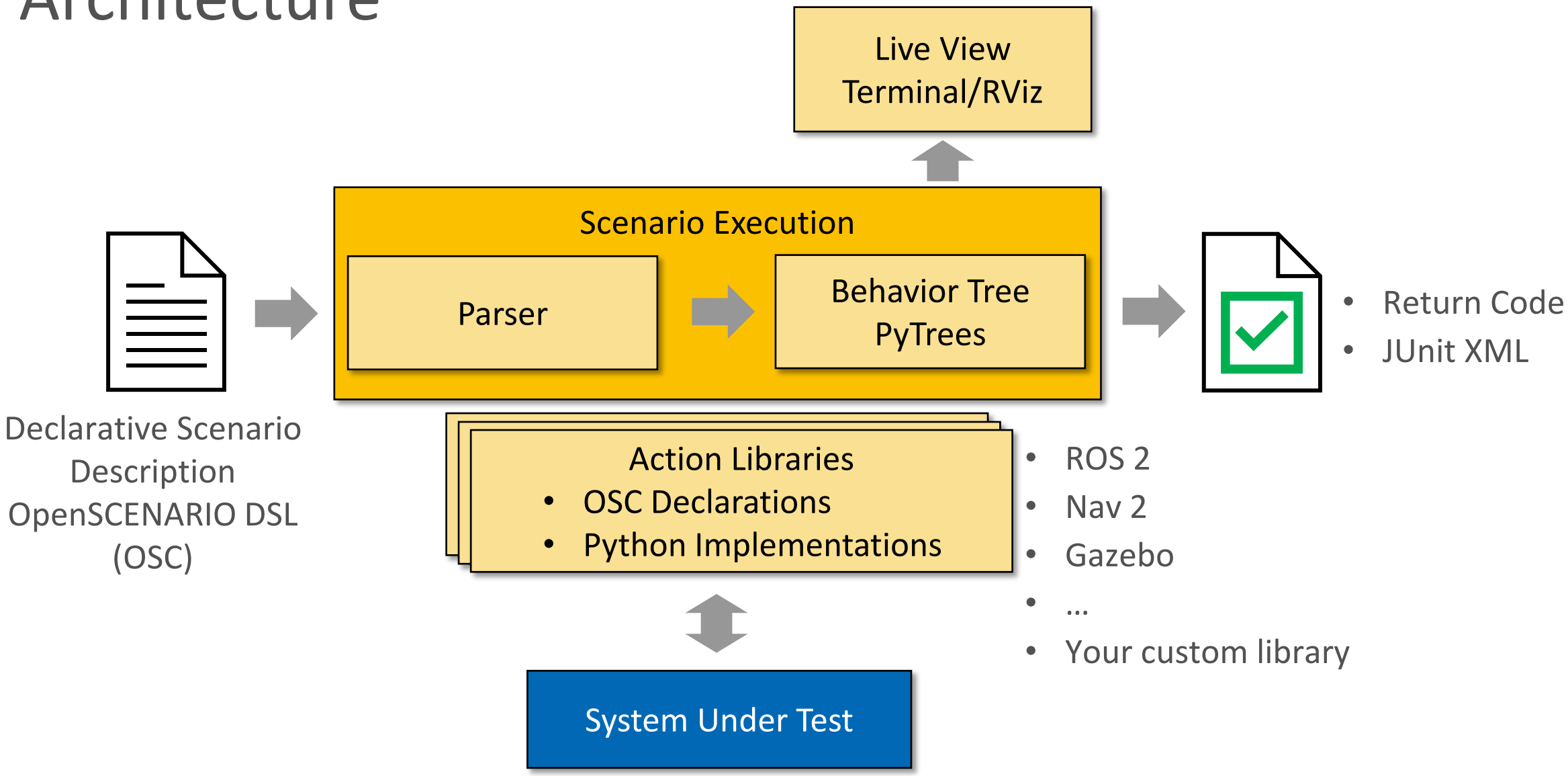
Real Robots

Field

Limited Tooling

- launch_testing
- Scenario Execution

Architecture



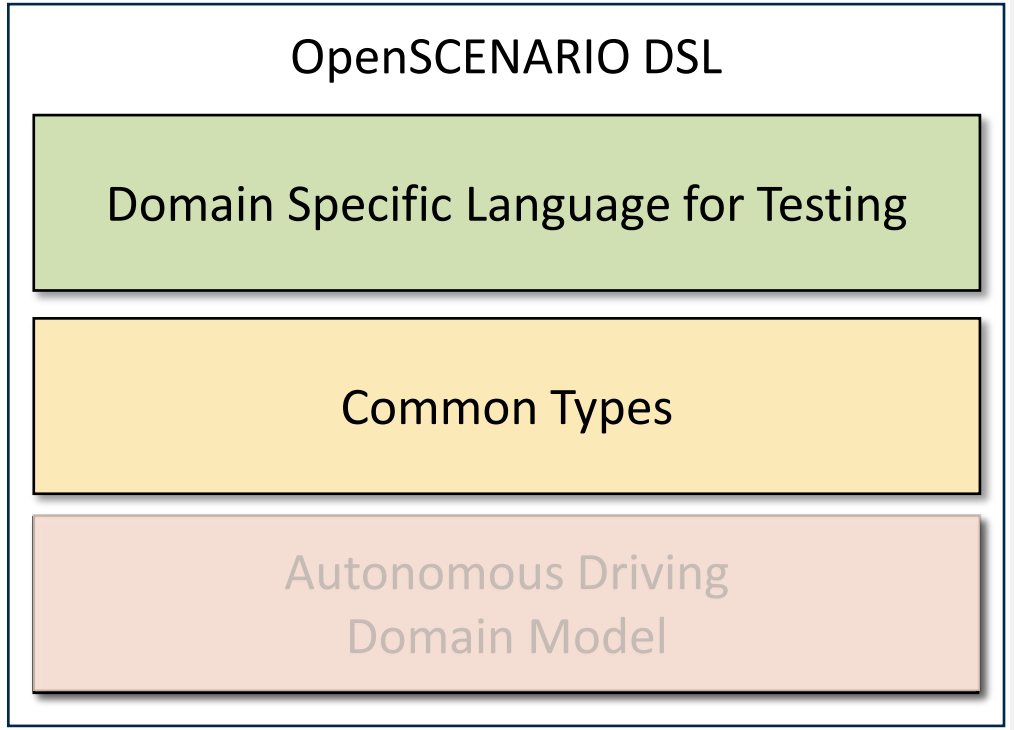
OpenSCENARIO DSL

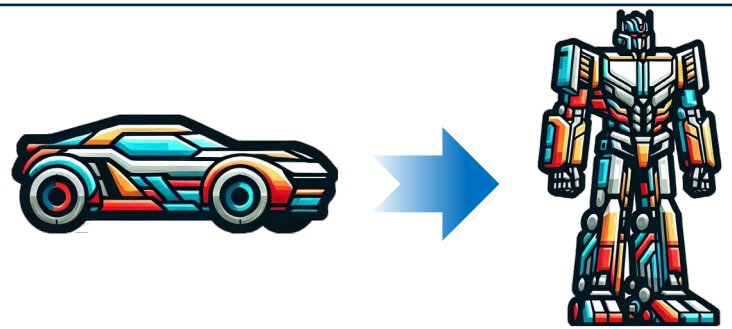
"… designed for verification and validation purposes to test safety and functionality of ~~autonomous vehicles~~ **any robotic system**…"

ASAM Association for Standardization of Automation and Measuring Systems

Key Facts

- Human-readable/writable
- Supports Variation
- Widely Used
- Applicable to any robotics system





Hello World Scenario

Import Library

Scenario Declaration

Composite

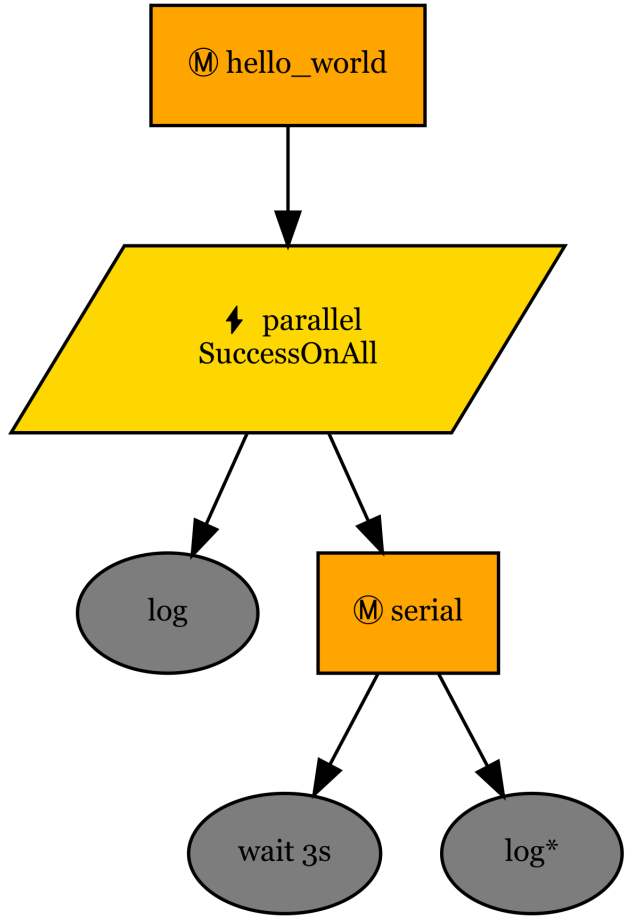
```
import osc.helpers

# my first scenario
scenario hello_world:
    var msg: string = "Hello World!"
    do parallel:
        log(msg)
        serial:
            wait elapsed(3s)
            log("Have a nice day!")
```

Comment

Variable Declaration

Action Invocation

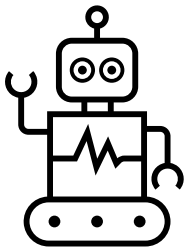
Wait Directive

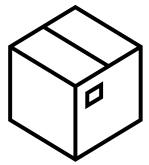


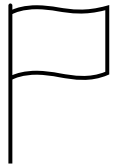

```
$
```

Navigation Scenario

- Start Simulation/Nav2
- Initialize Nav2
- Request to navigate to goal
- If robot reaches position:
  - Spawn Obstacle into Path
- Success Criteria:
  - Robot does not collide with obstacle
  - Goal is reached







```
import osc.helpers
import osc.ros
import osc.gazebo
import osc.nav2

scenario example_simulation:
    robot: differential_drive_robot
    box: osc_actor
    do serial:
        ros_launch('tb4_sim_scenario', 'sim_nav_scenario_launch.py') with:
            running_is_success()
        robot.init_nav2(pose_3d(position_3d(x: 0.0m, y: 0.0m)))
        parallel:
            behavior_under_test: serial:
                robot.nav_to_pose(pose_3d(position_3d(x: 3.0m, y: -3.0m)))
                emit end
            error_injection: serial:
                robot.tf_close_to(
                    reference_point: position_3d(x: 1.5m, y: -1.5m),
                    threshold: 0.4m,
                    robot_frame_id: 'turtlebot4_base_link_gt')
                box.spawn(
                    spawn_pose: pose_3d(
                        position: position_3d(x: 2.0m, y: -2.0m, z: 0.1m),
                        orientation: orientation_3d(yaw: 0.0rad)),
                    model: 'example_simulation://models/box.sdf')
            failure_check: serial:
                wait_for_data("/bumper_contact", "ros_gz_interfaces.msg.Contacts")
                emit fail
```

Scenario Parameter Variation

Potential Scenario Count

Abstract Scenario

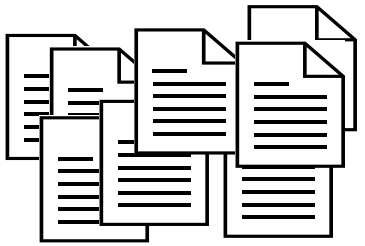
A navigating robot is able to localize itself using erroneous lidar data

∞

+

Logical Scenario

Lidar Data Errors:
- Noise Std.Dev.: 0 to 0.7
- Data Loss: 0 to 70%



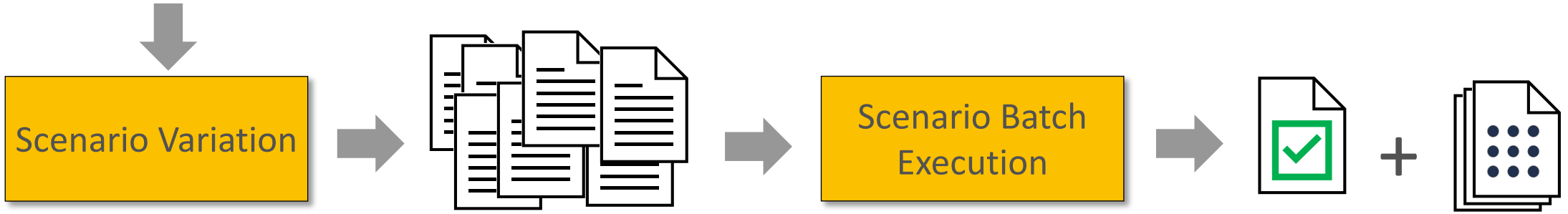
+

Concrete Scenario

Lidar Data Errors:
- Noise Std.Dev.: 0.4
- Data Loss: 10%

Scenario Parameter Variation Example

```
scenario laserscan_fault_injection:
    robot: differential_drive_robot
    do serial:
        robot.init_nav2(pose_3d(position_3d(x: 0.0m, y: 0.0m)))
        bag_record(['/tf', '/tf_static'], use_sim_time: true)
        set_node_parameter('laserscan_modification', 'gaussian_noise_std_deviation') with:
            keep(it.parameter_value in ['0.0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7'])
        set_node_parameter('laserscan_modification','random_drop_rate') with:
            keep(it.parameter_value in ['0.0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7'])
        robot.nav_to_pose(pose_3d(position_3d(x: 3.0m, y: -3.0m)))
        robot.nav_to_pose(pose_3d(position_3d(x: 0.0m, y: -0.0m)))
```



8x8 Concrete Scenarios

Overall Test Result

64 ROS bags

Scenario Execution for Robotics

OpenSCENARIO DSL

Action Libraries

Behavior Tree

ROS2 realization

Easy CI integration

Modular implementation, easy to extend

Compatible with simulations and physical robots

Questions?

GitHub

github.com/IntelLabs/scenario_execution

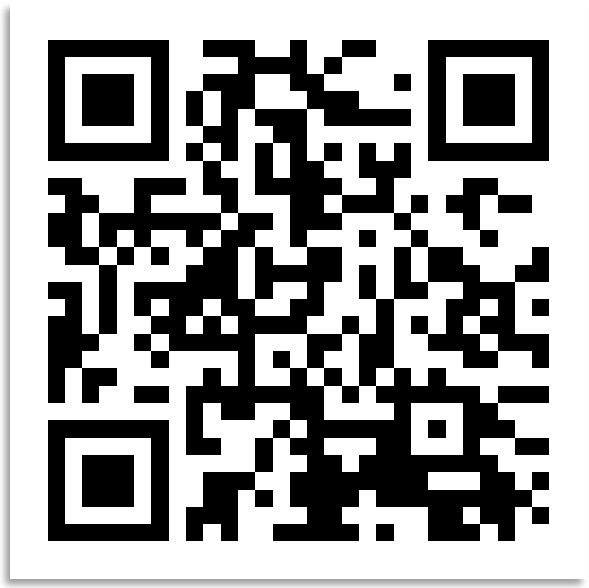
LinkedIn Profiles

Frederik Pasch

Florian Mirus


intel

Try it out!



GitHub

https://github.com/IntelLabs/scenario_execution

Documentation

https://intellabs.github.io/scenario_execution/

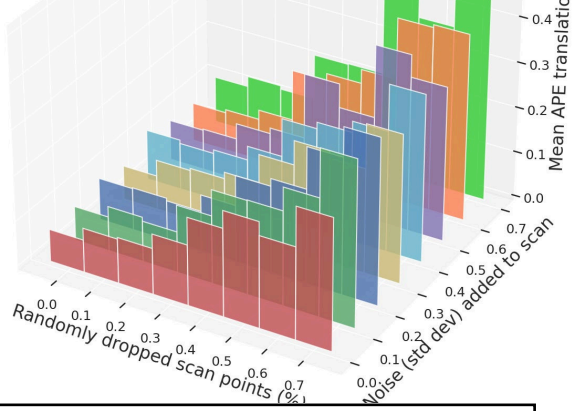
Paper

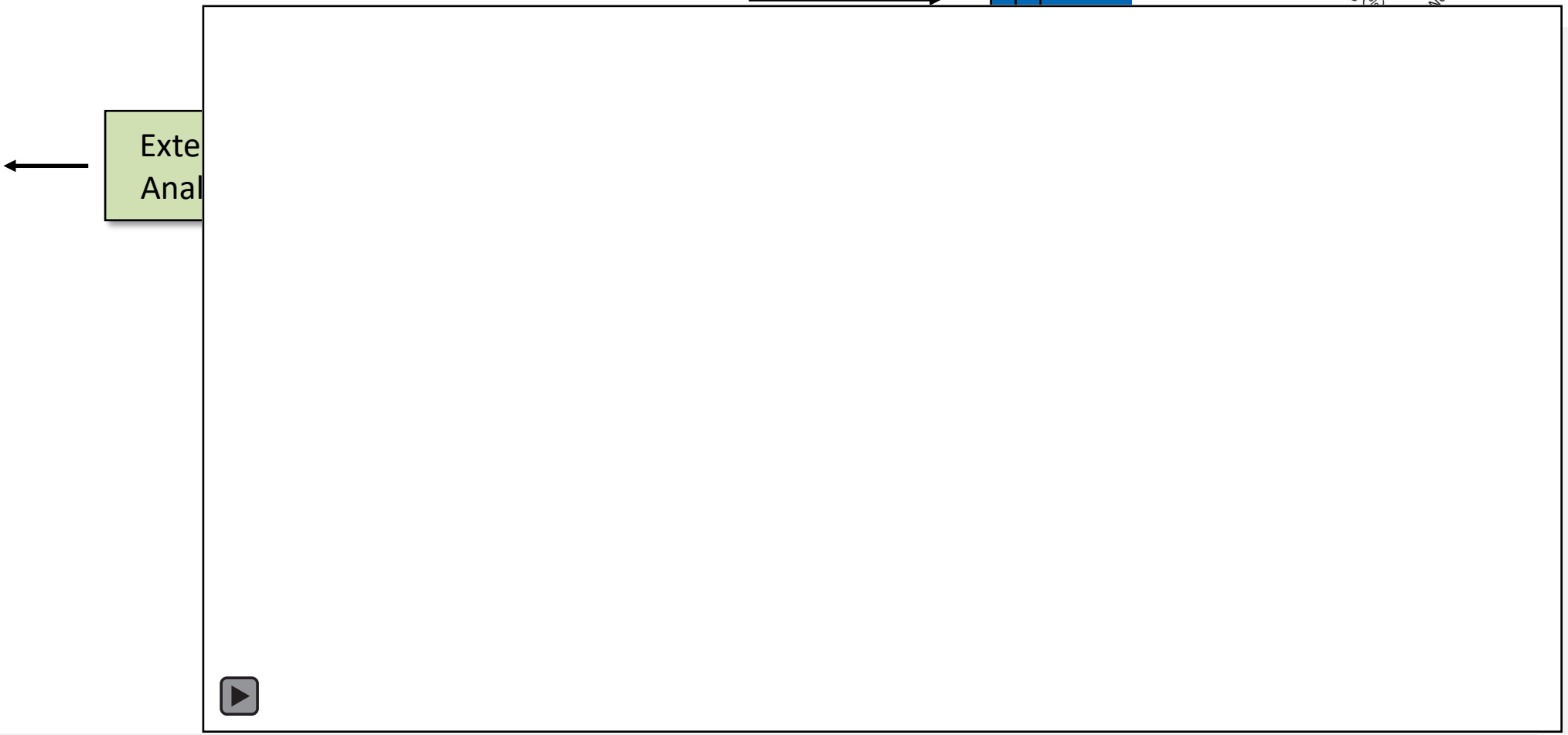
http://www.arxiv.org/abs/2409.07080

Action Libraries

ROS 2 	Navigation 2 	Gazebo 
MoveIt 2 	X11 	OS 
Docker 	Kubernetes 	PyBullet 
Helpers 	To be continued…	

A Parameter Variation Scenario





OpenSCENARIO DSL

"… designed for verification and validation purposes to test safety and functionality of ~~autonomous vehicles~~ **any robotic system**…"

ASAM Association for Standardization of Automation and Measuring Systems

Key Facts

- Human-readable/writable
- Supports Variation
- Widely Used
- Applicable to any robotics system