rospec: A Domain-Specific Language for ROS-based Robot Software

Paulo Canelas[†,‡], Bradley Schmerl[‡], Alcides Fonseca[†], and Christopher S. Timperley[‡]

[†] LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
[‡] Software and Societal Systems Department, Carnegie Mellon University, US

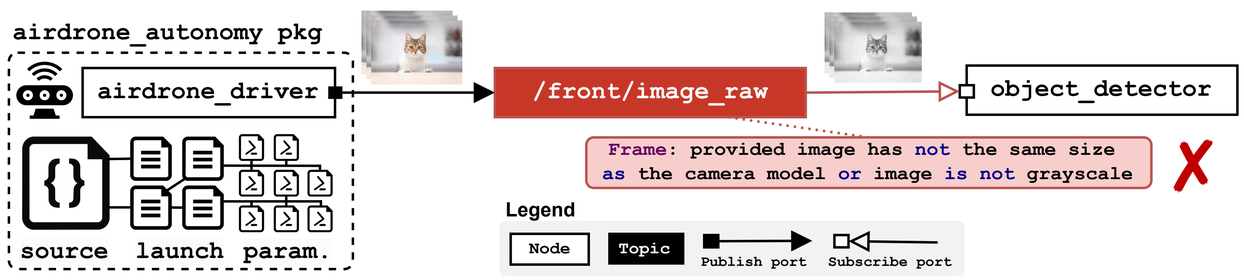
reliable software systems

Motivation

The **Robot Operating System (ROS)** is the de facto open-source framework for building complex robot software offering reusable and configurable off-the-shelf components. ROS components often **lack proper documentation**, making system configuration challenging and forcing developers to rely on **unverified assumptions**, leading to errors.

In this work, we propose a ROS-tailored **domain-specific language** to specify component **configuration** and **integration** using domain concepts.

Misconfigurations arise from mismatched expectations between components, potentially causing dangerous robot behaviors in physical environments.



By specifying properties over component configurations and their integration, we can detect misconfigurations prior to execution

AMCL Component Specification

```
type alias AfterHumbleVersion: Enum[…, Humble, Iron, Jazzy, Kilted] where {_ >= Humble};
type alias LaserModelType: Enum[Beam, LikelihoodField, LikelihoodFieldProb];

type alias Meter: float32;

message alias RestrictedLoadMap: nav2_msgs/LoadMap {
    request  field map_url: string;
    response field map: nav2_msgs/OccupancyGrid;
    response field result: uint8 where {(_ >= 0 and _ <= 3) or _ == 255};
}

node type amcl_type {
```

Type aliases act as documentation providing semantic information about the purpose of each configuration

Restrictions over types → Z3 Solver = (SAT, SAT, UNSAT, UNSAT)

(0, 255, 4, -1)

Typing / Bounds / Presence

```
    context distribution: AfterHumbleVersion;

    param laser_model_type: LaserModelType;

    optional param z_hit: double = 0.5;
    optional param z_max: double = 0.05;
    optional param z_rand: double = 0.5;
    optional param z_short: double = 0.005;
    optional param always_reset_initial_pose: bool = false;
```

QoS / Cardinality / Connection

```
    @qos{sensor_qos}
    publishes to particle_cloud: nav2_msgs/ParticleCloud;

    @qos{default_qos}
    subscribes to /initialpose: geometry_msgs/PoseWithCovarianceStamped
                                    where {count(publishers(_)) == 1};

    provides service set_initial_pose: nav2_msgs/SetInitialPose;
```

Dependency / Conditionals

```
} where {
    laser_model_type == Beam -> z_hit + z_max + z_rand + z_short == 1;
    laser_model_type == LikelihoodField -> z_hit + z_rand == 1;
    always_reset_initial_pose -> exists(initial_pose);
}
```

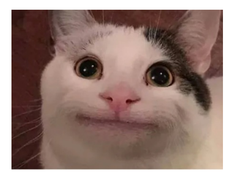
Case Study System Specification

```
system {

✗   node instance amcl: amcl_type {

        context distribution = Jazzy;

        param laser_model_type = Beam;

        param z_hit = 0.5;
        param z_max = 0.0;
        param z_rand = 0.5;

        param always_reset_initial_pose = false;
    }
}
```

Contextual information allows developers to specify deployment-specific requirements and dependencies

Sensor Quality of Service (QoS)

```
policy instance sensor_qos: qos {
    setting history = KeepAll;
    setting depth = 1;
    setting reliability = BestEffort;
}
```

@color_format{RGB8} +   @color_format{Grayscale}

Case Study

We evaluated rospec on **Neobotix MP-400** robot navigating a warehouse environment through specification and configuration of configurations across **18 components** including navigation, localization, and obstacle avoidance.

Evaluation

Analyzed **182 questions** from Stack Exchange, from which,

61	23	28 + 39	31
Detectable	Documentation	Missing Info / Out of Scope	Not Supported

rospec verifier

D-System

$$\Gamma \vdash \overline{D} \dashv \Gamma' \qquad \frac{\overline{s \mapsto \textbf{publishes to}(x, T_1) \in \Gamma' \wedge \Gamma' \vdash T_1 <: T_2}^{s \mapsto \textbf{subscribes to}(x,\ T_2) \in \Gamma'} \quad \overline{\textbf{check\_qos}(q1,\ q2)}^{s \mapsto \textbf{publishes to}(x) \textbf{ with } qos(q1) \in \Gamma',\ s' \mapsto \textbf{subscribes to}(x) \textbf{ with } qos(q2) \in \Gamma'} \quad \overline{s \mapsto \textbf{broadcasts}(x_1, x_2) \in \Gamma'}^{s \mapsto \textbf{listens}(x_1, x_2) \in \Gamma'} \quad \overline{x_1 = x_2}^{s \mapsto \textbf{broadcasts}(x_1, x_2) \in \Gamma',\ s' \mapsto \textbf{broadcasts}(x_2, x_3) \in \Gamma'}}{\Gamma \vdash \textbf{system}\ \{\ \overline{D}\ \} \dashv \Gamma'}$$

ERROR: Dependency laser_model_type == Beam -> z_hit + z_max + z_rand + z_short == 1 in amcl is not respected.

ERROR: Publisher not found for the subscriber amcl to the topic /initialpose.

Acknowledgements:

This work was supported by Fundação para a Ciência e Tecnologia (FCT) in the LASIGE Research Unit under the ref. (UID/00408/2025 and EXPL/CCI-COM/1306/2021), and the CMU Portugal Dual PhD program (SFRH/BD/151469/2021).