An Experience Report on Challenges in Learning the Robot Operating System

Paulo Santos [1,2], Miguel Tavares [1], Ricardo Cordeiro [1], Alcides Fonseca [1], Christopher S. Timperley [2]

[1] LASIGE, Departamento de Informática, Faculdade de Ciências da Universidade de Lisboa

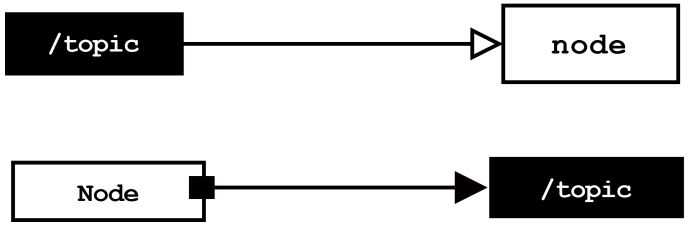
[2] Institute for Software Research, School of Computer Science, Carnegie Mellon University

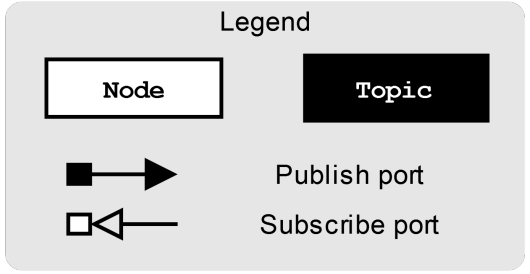
The Robot Operating System (ROS)

ROS

ROS allows developers to reuse existing componentes in their robots

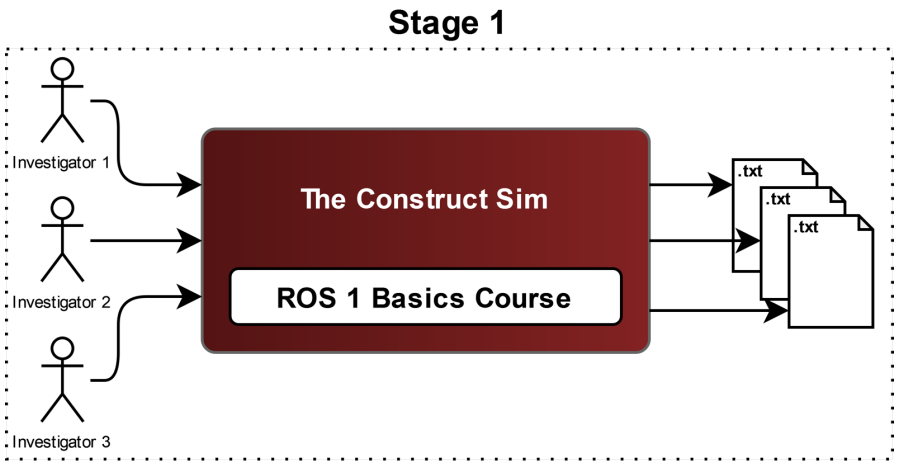
Abstracts the implementation details of several components of their robot, from odometry to route planning.





Understand the **experience** of newcomers when learning the Robot Operating System

The Investigators



Paulo Canelas

Ph.D. Student

No previous experience with robotic systems.

Miguel Tavares

MSc. Student

Experience with Thymio [1].

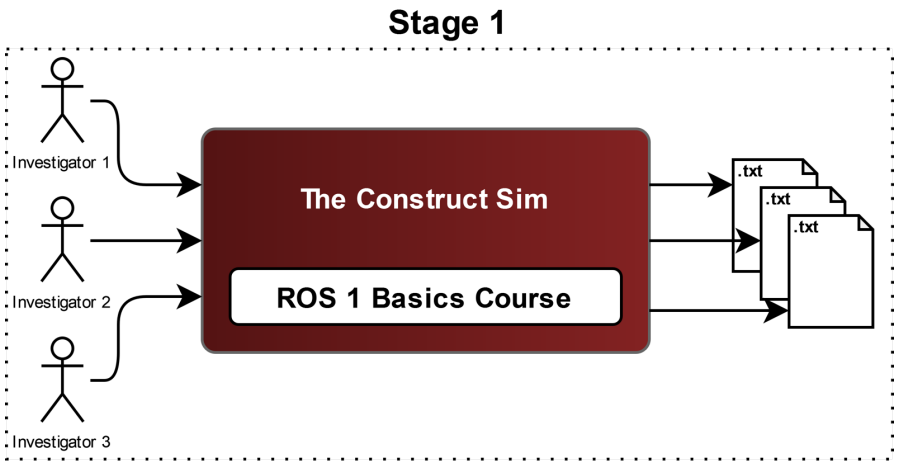
Ricardo Cordeiro

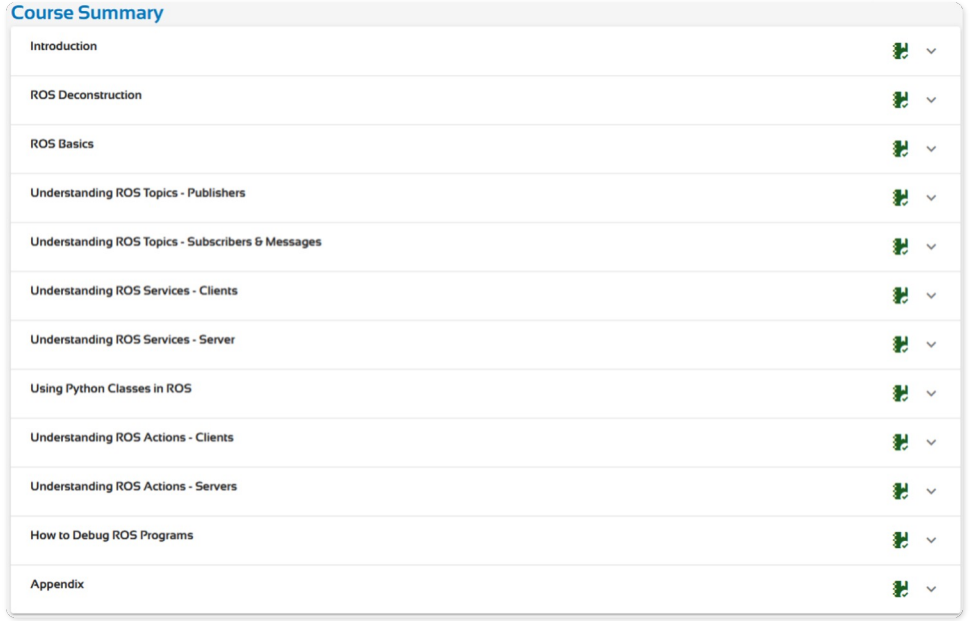
MSc. Student

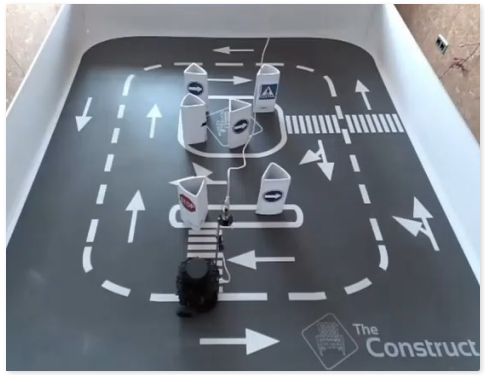
No previous experience with robotic systems.

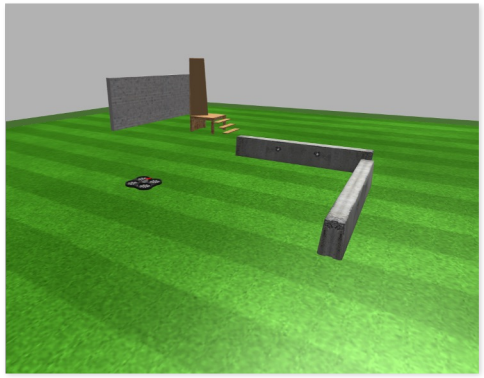
[1] Fanny Riedo. 2015. Thymio a holistic approach to designing accessible educational robots. (2015).

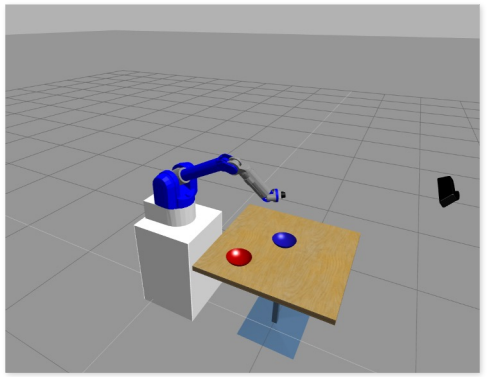
ROS 1 Basics course from The Construct Sim

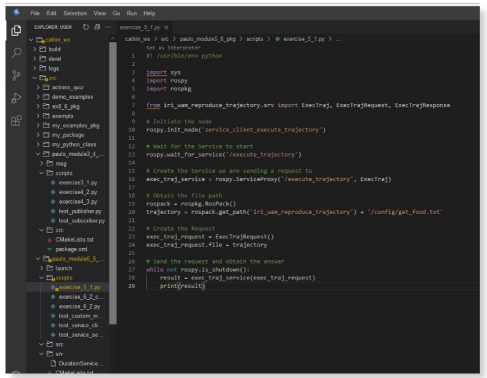




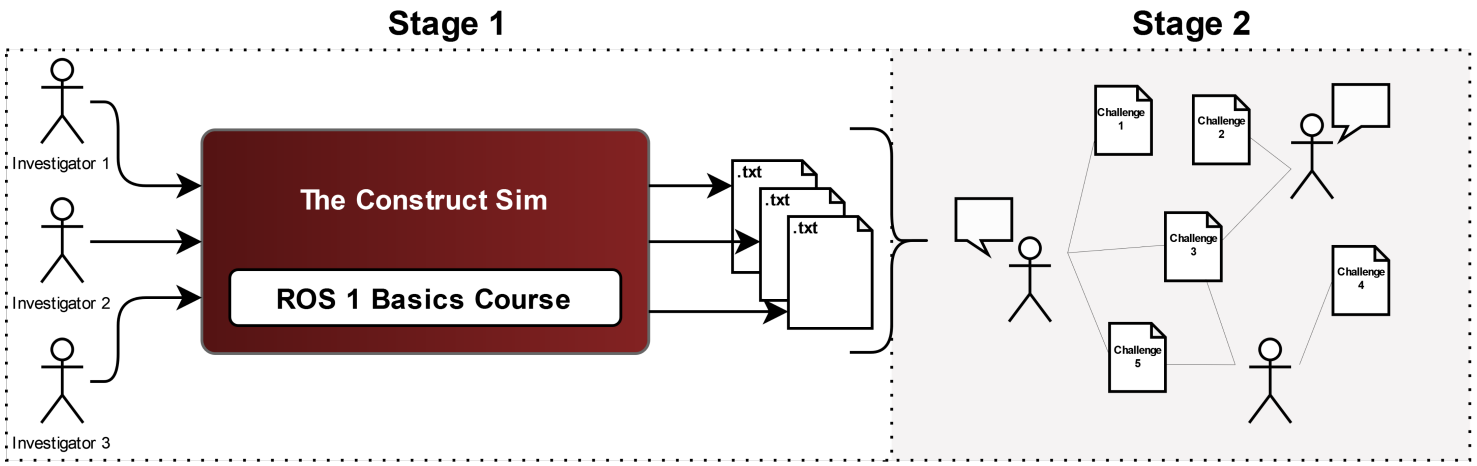






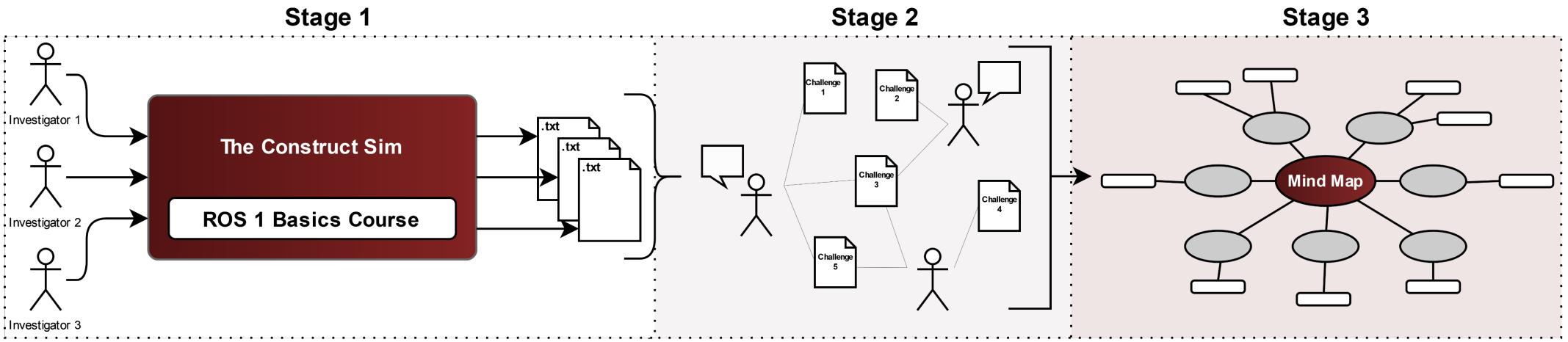


Adjudication and Discussion

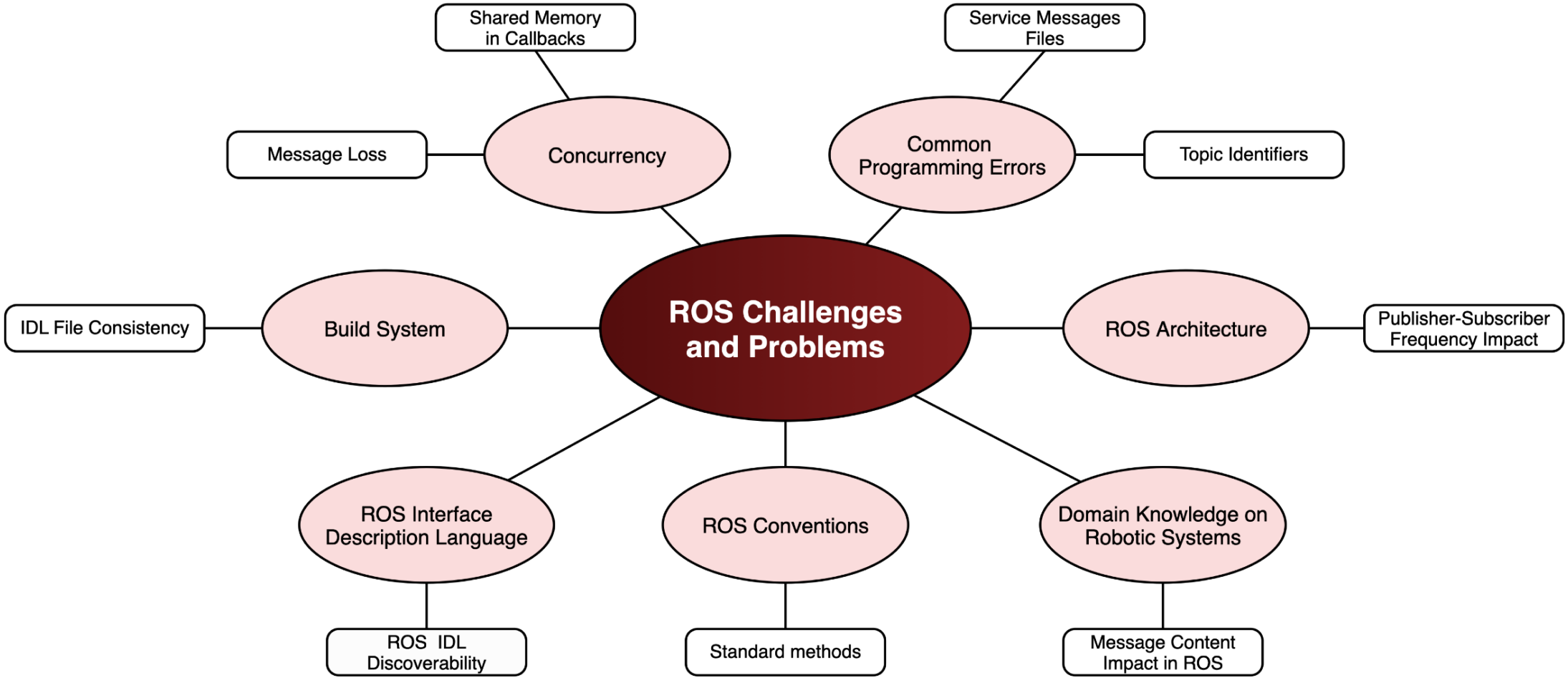


The unorganized notes are categorized and the investigators discuss the shared challenges.

Creation of the Mind Map



We identified seven high-level challenges



Build System

IDL File Consistency (●●○)

○○○
# of investigators who identified the challenge

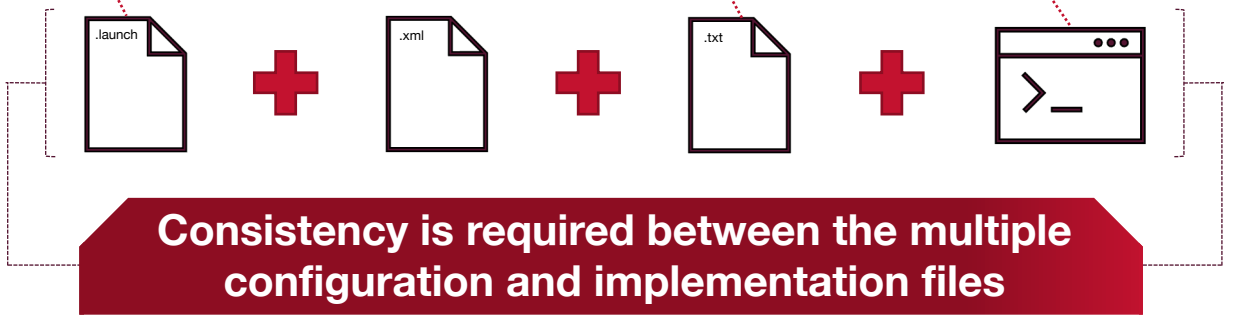
- The process for defining new message formats requires changing code in multiple locations, thus increasing the probability of introducing errors.

- The **lack of sanity** checks by ROS can lead to mismatch between identifiers defined in different files when creating a new node.

```
<launch>
    <node pkg ="module5_6_pkg"
        type="client.py"
        name="service_client"
        output="screen" />
</launch>
```

```
add_service_files(
  FILES
  DurationServiceMessage.srv
)
```

```
import rospy
from module5_6_pkg.srv import DurationServiceMessage,
                              DurationServiceMessageResponse

rospy.init_node('service_client')
service = rospy.Service('/myservice', DurationServiceMessage,
 my_callback)
```



Consistency is required between the multiple configuration and implementation files

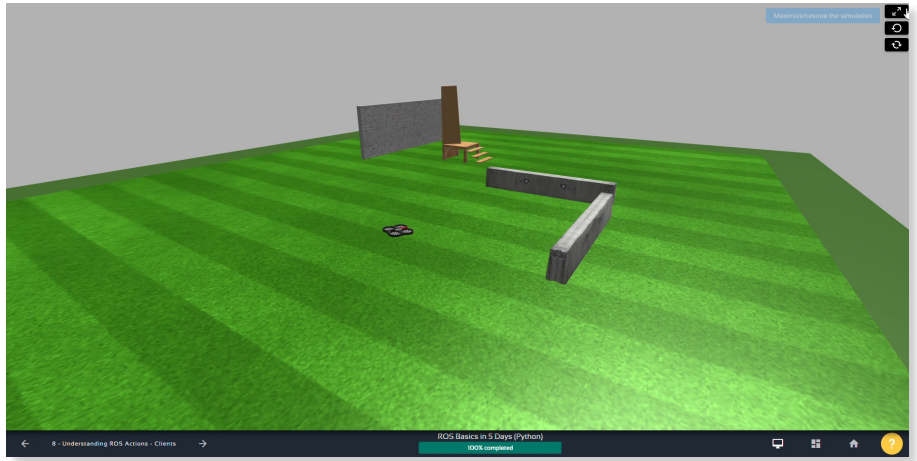
ROS Interface Description Language

ROS IDL Discoverability (●●●)

○○○
# of investigators who identified the challenge

- ROS provides components for different common tasks in robots. Nevertheless, it is challenging to identify the components responsible for newcomers to **identify components responsible** for providing certain information.

- Furthermore, it is not explicit how each message and its parameters impact the execution of the robotic systems due to a **lack of documentation**.



Which topic responsible for the drone position?

```
user:~/catkin_ws$ rostopic list
/camera_info
/clock
/cmd_vel
/drone/down_camera/image_raw
/drone/down_camera/image_raw/compressed
/drone/down_camera/image_raw/compressed/parameter_descriptions
/drone/down_camera/image_raw/compressed/parameter_updates
/drone/down_camera/image_raw/compressedDepth
/drone/down_camera/image_raw/compressedDepth/parameter_descriptions
/drone/down_camera/image_raw/compressedDepth/parameter_updates
/drone/down_camera/image_raw/theora
/drone/down_camera/image_raw/theora/parameter_descriptions
/drone/down_camera/image_raw/theora/parameter_updates
/drone/front_camera/image_raw
/drone/front_camera/image_raw/compressed
/drone/front_camera/image_raw/compressed/parameter_descriptions
/drone/front_camera/image_raw/compressed/parameter_updates
/drone/front_camera/image_raw/compressedDepth
/drone/front_camera/image_raw/compressedDepth/parameter_descriptions
/drone/front_camera/image_raw/compressedDepth/parameter_updates
/drone/front_camera/image_raw/theora
/drone/front_camera/image_raw/theora/parameter_descriptions
/drone/front_camera/image_raw/theora/parameter_updates
/drone/gt_acc
/drone/gt_pose
/drone/gt_vel
/drone/imu
/drone/land
/drone/posctrl
/drone/reset
/drone/sonar
/drone/takeoff
/drone/vel_mode
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/rosout
/rosout_agg
```

Common Programming Errors

Topic Identifiers (●●●)

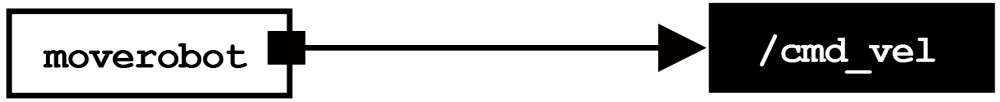
of investigators who identified the challenge

- In ROS, to publish or subscribe to information one needs to provide the topic name as a string.

- One of the most common error is the **mistyping of topic names**. Since no verification is done, the system compiles and runs but does not behave as intended.

```
rospy.init_node('moverobot')
pub = rospy.Publisher('\cmd_vel', Twist, queue_size=1)
```


```
rospy.init_node('moverobot')
pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
```



ROS Conventions

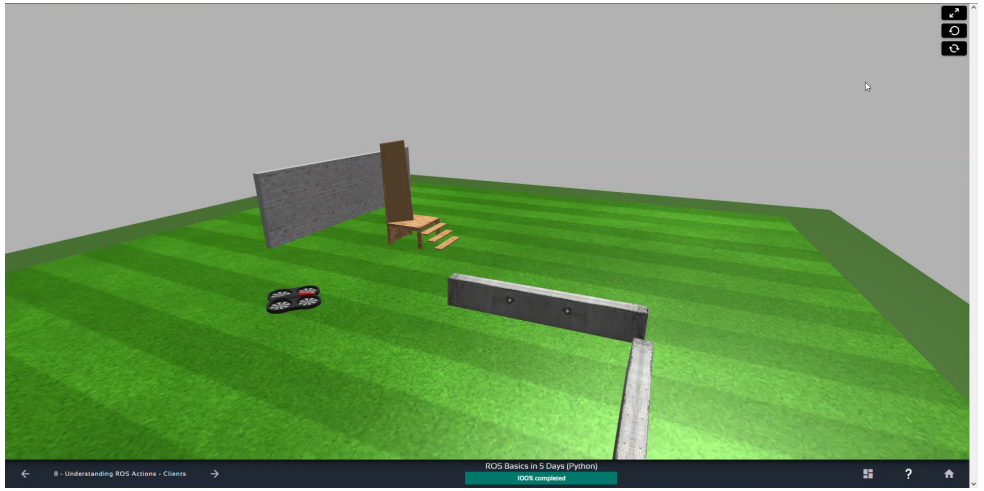
Standard Methods (●●●)

of investigators who identified the challenge

- The investigators found it common not to follow expected good practices. In ROS, the lack of good practices can lead to an unintended behaviour of the system.

- One example is forgetting to implement callbacks and hook methods, typically associated with the good functioning of the robotic system. However, there is **no warning or clear message** identifying this issue in ROS.

```
pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
position = Twist()

def shutdown_publish():
    global pub, position

    position.linear = Vector3(0, 0, 0)
    position.angular = Vector3(0, 0, 0)

    pub.publish(position)

rospy.on_shutdown(shutdown_publish)
```



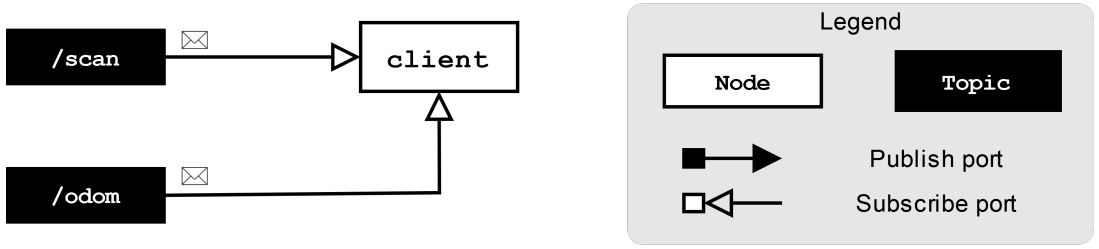
Concurrency

Shared Memory in Callbacks (●○○)

○○○
# of investigators who identified the challenge

- The investigators found concurrency related issues were not properly addressed by the ROS API nor The Construct Sim.

- Race conditions can lead to an unintended behaviour of the robotic system. A possible solution is the introduction of concurrency safety procedures (e.g., mutex). However, the use of a mutex may change the frequency at which the callback operates.




```
sensor = list()

# Callback for scan topic
def scan(scan_msg):
  sensor = scan_msg.ranges

# Callback for odometry topic
def odom(msg):
  position : Pose = msg.pose.pose.position
  if sensor[90] < 1.0:
    print(f"Robot close to wall: {sensor[90]}")
    print(f"Robot Position: {position}")

sub1 = rospy.Subscriber('/odom', Odometry, odom)
sub2 = rospy.Subscriber('/scan', LaserScan, scan)
rospy.spin()
```

Concurrency

Message Loss (●●○)

of investigators who identified the challenge

- A common problem faced by the investigators is the loss of messages when a node publishes to a topic only once before the subscriber is listening, leading the robot to an idle state.
- When a node that uses actions or services is launched and the corresponding server is not ready, the published messages are silently lost.
- ROS allows the persistence of the last published message to a topic by “latching” the connection.

If the connection is not *latched*, the order in which the subscriber and publisher are initiated matters.

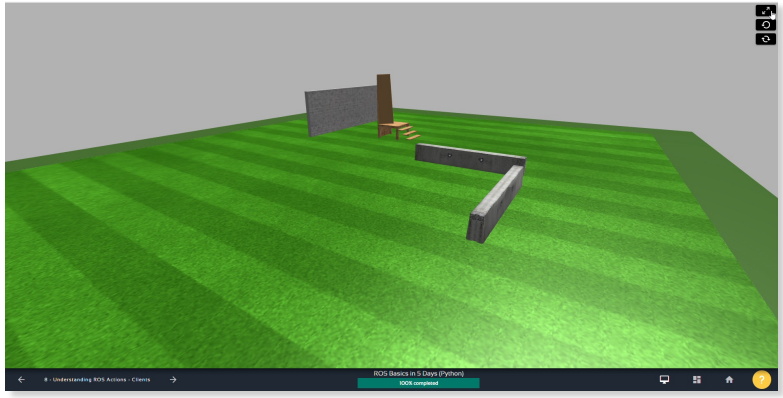
Domain Knowledge on Robotic Systems

Message Content Impact in ROS (●●●)

○○○
# of investigators who identified the challenge

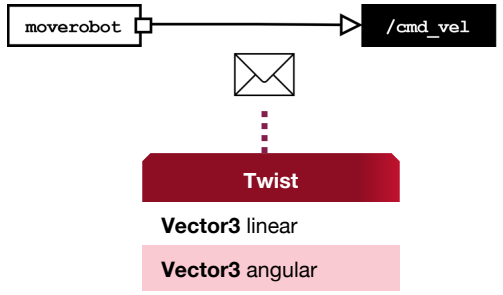
How to estimate and understand the impact of the message content with the real-world behavior

- For instance, how does the velocity value published affects the real speed of the robot.



- When trying to smoothly land the drone, considering the messages publishing frequency and their content is not enough to achieve this objective.

- The abstraction model of ROS hides the dependency on the domain knowledge and the implementation details, hindering the connection between high-level code and its impact in the simulation.



ROS Architecture

Publisher-Subscriber Frequency Impact (●●●)

○○○
# of investigators who identified the challenge

- This challenge appears in the **definition of the publishing rate** and the **adequate queue size**. Both ROS publisher and subscriber place their messages on a bounded queue at a specific publication rate.

- A component may need to perform an action each millisecond, but the information provider only emits updated data each second. ROS developers can use both components without considering the mismatch in the assumed and provided frequencies.

Challenge 1: **What is the proper queue size?**

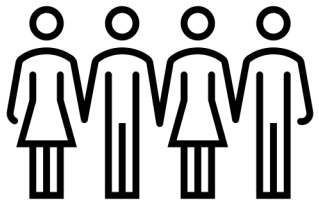
Challenge 2: **Considering the queue size, what is the proper publishing rate?**

- Dependency between the queue size and the publishing rate.
- The wrong configuration combination can lead to unintended robot behavior due to the loss of messages.

```
# Create the publisher
pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)

# Create the message
message = Twist()
message.linear = Vector3(0.5, 0, 0)

# Define the rate
rate = rospy.Rate(10)

# Publish the speed at fixed rate of 10 Hz
while not rospy.is_shutdown():
    message.linear.x += 0.01
    pub.publish(message)
    rate.sleep()
```

What next?

Usability Studies



- Help design more in-depth usability studies with larger groups.
- Difficulty of applying good practices in ROS and its impact on the robot's behavior.

Documentation Improvement

- Encourage the improvement of the documentation:
  - Component's interface;
  - Indented communication model;
  - Frequency;
  - Bounds on messages values.

Introduction and Improvement of Verification Techniques

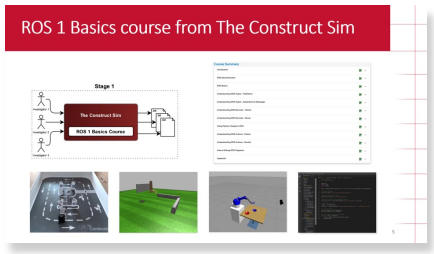
ROSDiscover HAROS

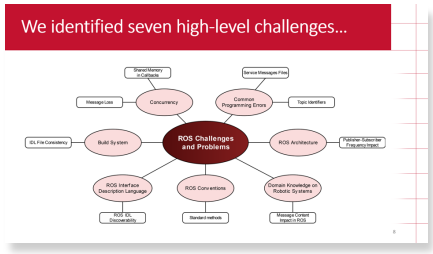
Architectural Robot and System Verification

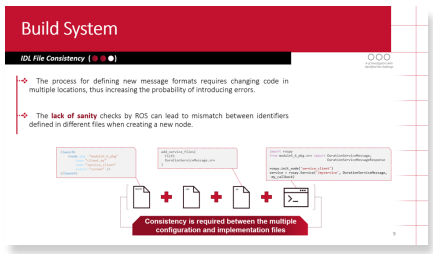
- Analysis of the architecture of the robot and systems configuration files to provide novice and expert users the information needed to correct existing problems.
- Introduction of specification techniques of the systems architecture by the user and the formal static verification [4].

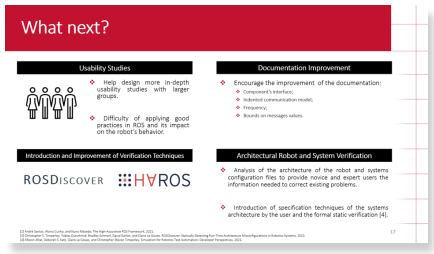
[2] André Santos, Alcino Cunha, and Nuno Macedo. The High-Assurance ROS Framework. 2021.
[3] Christopher S. Timperley, Tobias Dürschmid, Bradley Schmerl, David Garlan, and Claire Le Goues. ROSDiscover: Statically Detecting Run-Time Architecture Misconfigurations in Robotics Systems. 2022.
[4] Afsoon Afzal, Deborah S. Katz, Claire Le Goues, and Christopher Steven Timperley. Simulation for Robotics Test Automation: Developer Perspectives. 2021.

Understand the **experience** of newcomers when learning the Robot Operating System









This work was supported by *Fundação para a Ciência e Tecnologia* (FCT) in the LASIGE Research Unit under the ref. (UIDB/00408/2020 and UIDP/00408/2020), and the CMU-Portugal Dual Degree PhD Program (SFRH/BD/151469/2021), by the CMU-Portugal project CAMELOT, (POCI-01-0247-FEDER-045915), the RAP project under the reference (EXPL/CCI-COM/1306/2021), and the U.S. Air Force Research Laboratory (\OSR-4066).

The authors are grateful for their support. Any opinions, findings, or recommendations expressed are those of the authors and do not necessarily reflect those of the US Government.

LASIGE driven by excellence · Ciências ULisboa · Carnegie Mellon Portugal · ISR institute for SOFTWARE RESEARCH · Carnegie Mellon University School of Computer Science · FCT Fundação para a Ciência e a Tecnologia

An Experience Report on Challenges in Learning the Robot Operating System

LASIGE driven by excellence
Ciências ULisboa
Carnegie Mellon Portugal
isr institute for SOFTWARE RESEARCH
Carnegie Mellon University School of Computer Science
FCT Fundação para a Ciência e a Tecnologia