Towards Interactive, Incremental Programming of ROS Nodes

Sorin Adam, Conpleks Innovation

Ulrik Pagh Schultz, University of Southern Denmark

Robotics Software Development

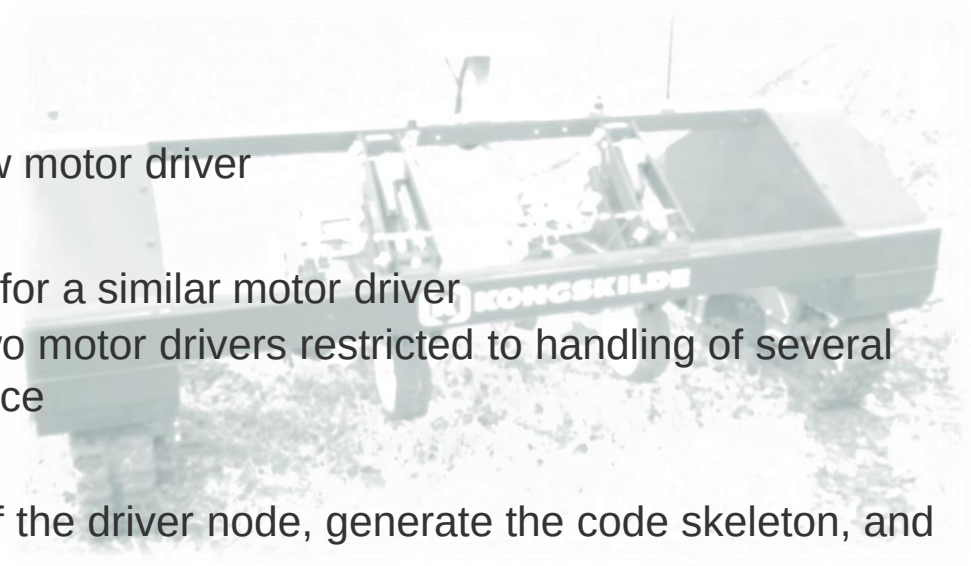
- Ideal tool for robotics software should fit both non-software experts and software experts
- Simplify software development: middleware and frameworks
  - but not enough
- Enhance productivity: MDSD and DSL
  - But often experimentation required
- MDSD: everything modeled
  - Not modeling upfront closer to ROS philosophy

ROS and Robot Software Development

- Component based: ROS nodes written in C++, Python, etc. communicating via topics (publish/subscribe mechanism)
- Steep learning curve
  - Comprehension of ROS concepts and mastering APIs takes time
- Flexible and interactive
  - Used for both single node experiments and complex systems
  - Has command line tools for node creation and interaction, but changes require code recompilation

- Claim: Due to high level of interactivity ROS is preferred by tinkerers

Use case: modifying a Frobomind* node



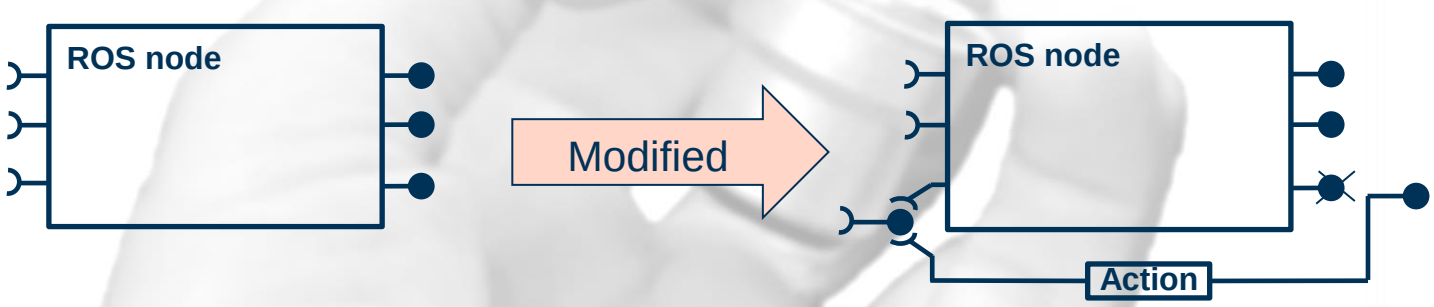
- The problem:
  - A ROS node needed for a new motor driver
- Background:
  - another node already present for a similar motor driver
  - The difference between the two motor drivers restricted to handling of several messages on the serial interface
- Solutions:
  1. MDSD: construct a model of the driver node, generate the code skeleton, and implement the behavior
  2. Refactor: change the code of the existing node to accomodate both drivers by introducing class inheritance
  3. Clone and modify: create a new node by copy-pasting the existing code and modifying it
  4. Our proposal: keep the existing node unchanged and, at runtime, modify only the behavior of a specific topic

*Frobomind - a software framework for field robots

Our solution

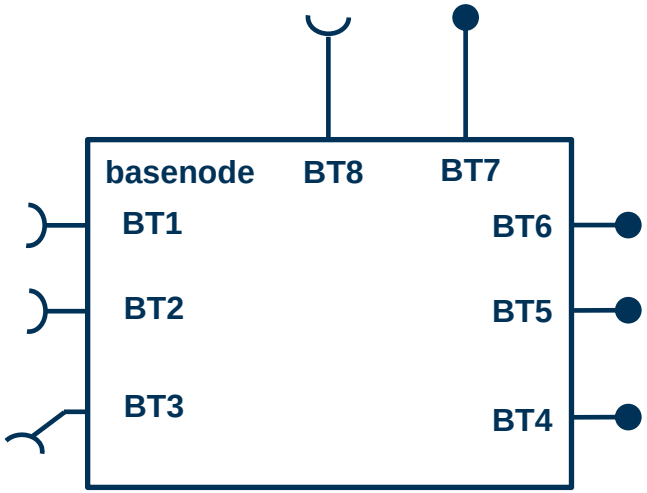
- An internal DSL to facilitate *interactively* writing and changing a ROS node
  - Interactivity provided by Python command line interface

- What we want:



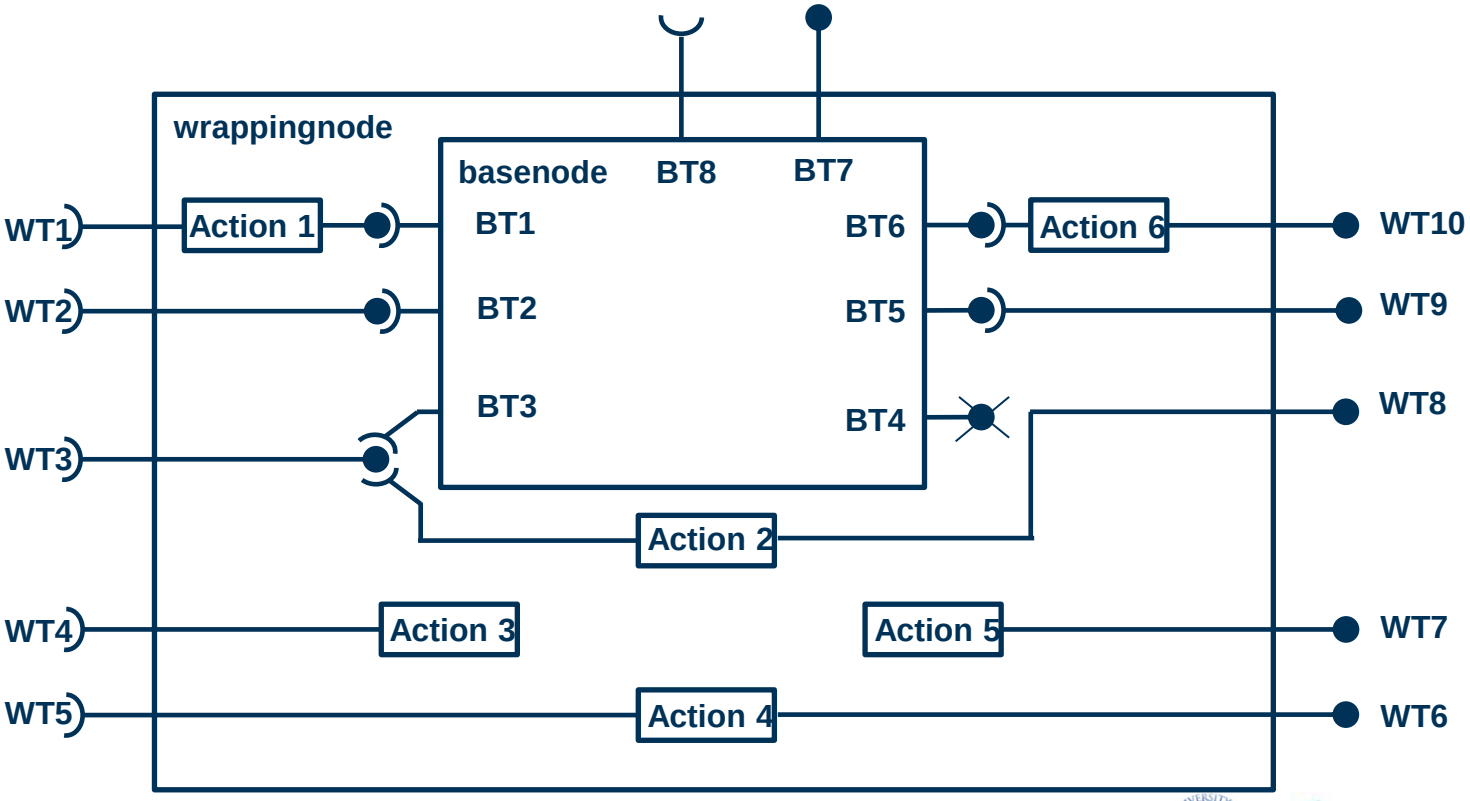
General concept: component wrapping

- Component wrapping is a standard technique in CBSD
  - but wrapping ROS nodes interactivelly from Python is not
- Starting from a basenode



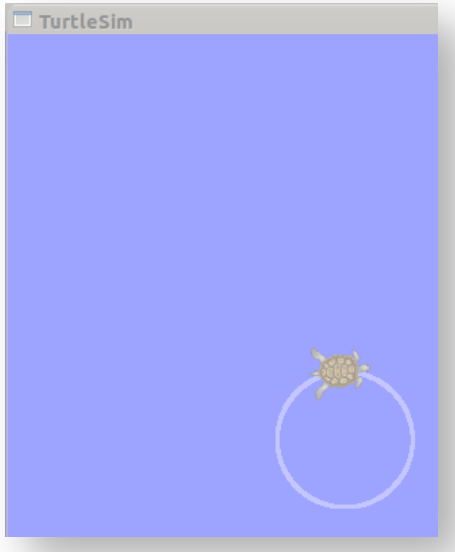
General concept: component wrapping

- Component wrapping is a standard technique in CBSD
  - but wrapping ROS nodes interactivelly from Python is not
- Starting from a basenode
  - and rename topics though a wrappingnode

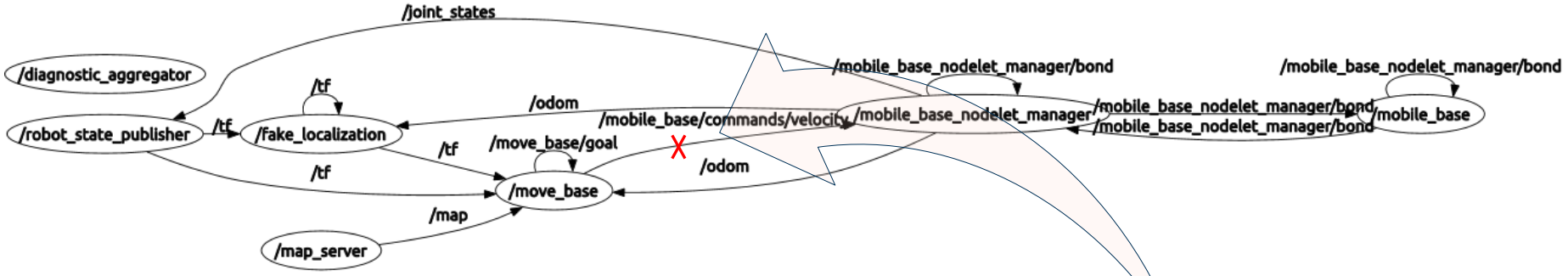


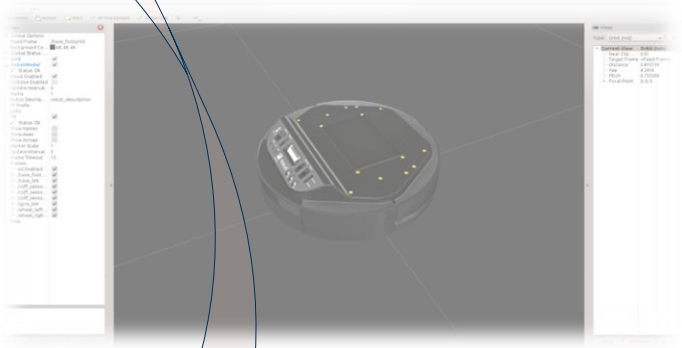
Example 1: New ROS node creation

```
from geometry_msgs.msg import Twist
from turtlesim.msg import Pose

def showPose(data):
    print("Pose: {}".format(data))

nd = rosNode("turtle_control_node")

{
nd.new.subscribe( topic = "/turtle1/pose", handler = showPose, msgType = Pose)
          .publish( topic = "/turtle1/cmd_vel", msgType = Twist)
}

nd.create()

def onTimer(event):
    msg = Twist()
    msg.linear.x = 2.0
    msg.angular.z = 1.8
    nd.write("/turtle1/cmd_vel", msg )

rospy.Timer( rospy.Duration(1), onTimer)
```



Example 2: ROS node wrapping





```
subnode = rosNode("experimental_move_base")

subnode.baseNode("move_base").basePackage("move_base")
{
 subnode.reuse
    .publish( topic = "cmd_vel", type = Twist)
    .publish( topic = "move_base/current_goal", type = PoseStamped)
    .publish( topic = "move_base/goal", type = MoveBaseActionGoal)
    .subscribe( topic = "tf_static", type = TFMessage)
    .subscribe( topic = "move_base_simple/goal", type = PoseStamped)
    .subscribe( topic = "tf", type = TFMessage)
}
{
subnode.new
        .subscribe( topic = "cmd_vel", handler = relayVelocity, type = Twist)
        .publish( topic = "mobile_base/commands/velocity", type = Twist)
}
```

Future work

- Alternative syntax for DSL:
  - node.**reuse.publish** (**topic** = "cmd_vel", **type** = Twist)
  - node.**reuse.publish**.cmd_vel(Twist)
- Wrapping
  - Code execution speed penalty measurements and improvements
  - To be extended to other middlewares and other programming languages
- MDSD: component model integration with other tools
- Extend its usage with safety enhancements
  - See ”*Towards Rule-based Dynamic Safety Monitoring for Mobile Robots”* poster presentation Wednesday, Oct. 22 14:10-14:40

Summary

- Use Python's command line as an interactive interface together with a ROS targeted DSL
  - Roboticists with other expertise than software could use the DSL without focusing on the ROS details
  - Software experts get a tool to speed up the experimental phase by making interactive changes to the ROS nodes

About Conpleks Innovation

Conpleks Innovation is based in Struer, Denmark, and operates worldwide. Our customer base consists of both small domestic companies and large international corporations.

Depending on the nature of the task, we can involve an entire network of competent hardware, software and mechanical design professionals.
This way, we can assign appropriate resources, find the best solutions and create a successful product.

Contact us

You are always welcome to contact us and discuss how we can help you to develop your idea – partly or fully – depending on your needs.

CONPLEKS
INNOVATION

Fælledvej 17 · 7600 Struer · Denmark · Email: contact@conpleks.com · www.conpleks.com