A.2 Practice 05 - Services and Actions

A.2.1 Service server

```python
import rclpy
from rclpy.node import Node

from example_interfaces.srv import AddTwoInts


class MinimalService(Node):

    def __init__(self):
        super().__init__('minimal_service')
        self.srv = self.create_service(AddTwoInts, 'firstservice', self.add_two_ints_callback)

    def add_two_ints_callback(self, request, response):
        response.sum = request.a + request.b
        self.get_logger().info('Incoming request\na: %d b: %d' % (request.a, request.b))

        return response


def main(args=None):
    rclpy.init(args=args)

    minimal_service = MinimalService()

    rclpy.spin(minimal_service)

    rclpy.shutdown()


if __name__ == '__main__':
    main()
```

A.2.2 Client

```python
import sys

from example_interfaces.srv import AddTwoInts
import rclpy
from rclpy.node import Node


class MinimalClientAsync(Node):

    def __init__(self):
        super().__init__('minimal_client_async')
        self.cli = self.create_client(AddTwoInts, 'firstservice')
```

```
        while not self.cli.wait_for_service(timeout_sec=1.0):
            self.get_logger().info('service not available, waiting again...')
        self.req = AddTwoInts.Request()

    def send_request(self, a, b):
        self.req.a = a
        self.req.b = b
        self.future = self.cli.call_async(self.req)
        rclpy.spin_until_future_complete(self, self.future)
        return self.future.result()


def main(args=None):
    rclpy.init(args=args)

    minimal_client = MinimalClientAsync()
    response = minimal_client.send_request(int(sys.argv[1]), int(sys.argv[2]))
    minimal_client.get_logger().info(
        'Result of add_two_ints: for %d + %d = %d' %
        (int(sys.argv[1]), int(sys.argv[2]), response.sum))

    minimal_client.destroy_node()
    rclpy.shutdown()


if __name__ == '__main__':
    main()
```

A.2.3 setup.py

```
from setuptools import setup

package_name = 'srvcli'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='mark',
    maintainer_email='m4rk.domonkos@gmail.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
```

```
        'console_scripts': [
            'service = srvcli.serviceserver:main',
            'client = srvcli.serviceclient:main',

        ],
    },
)
```

A.2.4 Action Server

```
import time

import rclpy
from rclpy.action import ActionServer
from rclpy.node import Node

from action_tutorials_interfaces.action import Fibonacci


class FibonacciActionServer(Node):

    def __init__(self):
        super().__init__('fibonacci_action_server')
        self._action_server = ActionServer(
            self,
            Fibonacci,
            'firstaction',
            self.execute_callback)

    def execute_callback(self, goal_handle):
        self.get_logger().info('Executing goal...')

        feedback_msg = Fibonacci.Feedback()
        feedback_msg.partial_sequence = [0, 1]

        for i in range(1, goal_handle.request.order):
            feedback_msg.partial_sequence.append(
                feedback_msg.partial_sequence[i] + feedback_msg.
    partial_sequence[i-1])
            self.get_logger().info('Feedback: {0}'.format(feedback_msg.
    partial_sequence))
            goal_handle.publish_feedback(feedback_msg)
            time.sleep(1)

        goal_handle.succeed()

        result = Fibonacci.Result()
        result.sequence = feedback_msg.partial_sequence
        return result


def main(args=None):
    rclpy.init(args=args)
```

```
42
43      fibonacci_action_server = FibonacciActionServer()
44
45      rclpy.spin(fibonacci_action_server)
46
47
48  if __name__ == '__main__':
49      main()
```

A.2.5 Action Client

```
1   import sys
2   import rclpy
3   from rclpy.action import ActionClient
4   from rclpy.node import Node
5
6   from action_tutorials_interfaces.action import Fibonacci
7
8
9   class FibonacciActionClient(Node):
10
11      def __init__(self):
12          super().__init__('fibonacci_action_client')
13          self._action_client = ActionClient(self, Fibonacci, 'firstaction
        ')
14
15      def send_goal(self, order):
16          goal_msg = Fibonacci.Goal()
17          goal_msg.order = order
18
19          self._action_client.wait_for_server()
20
21          self._send_goal_future = self._action_client.send_goal_async(
        goal_msg, feedback_callback=self.feedback_callback)
22
23          self._send_goal_future.add_done_callback(self.
        goal_response_callback)
24
25      def goal_response_callback(self, future):
26          goal_handle = future.result()
27          if not goal_handle.accepted:
28              self.get_logger().info('Goal rejected :(')
29              return
30
31          self.get_logger().info('Goal accepted :)')
32
33          self._get_result_future = goal_handle.get_result_async()
34          self._get_result_future.add_done_callback(self.
        get_result_callback)
35
36      def get_result_callback(self, future):
37          result = future.result().result
38          self.get_logger().info('Result: {0}'.format(result.sequence))
```

```
        rclpy.shutdown()

    def feedback_callback(self, feedback_msg):
        feedback = feedback_msg.feedback
        self.get_logger().info('Received feedback: {0}'.format(feedback.
    partial_sequence))


def main(args=None):
    rclpy.init(args=args)

    action_client = FibonacciActionClient()

    action_client.send_goal(int(sys.argv[1]))

    rclpy.spin(action_client)


if __name__ == '__main__':
    main()
```

A.2.6 setup.py

```
from setuptools import setup

package_name = 'actionsrvcli'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='mark',
    maintainer_email='m4rk.domonkos@gmail.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'acserver = actionsrvcli.actionserver:main',
            'acclient = actionsrvcli.actionclient:main',

        ],
    },
)
```

A.2.7 CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5)
project(first_interfaces)

# Default to C99
if(NOT CMAKE_C_STANDARD)
  set(CMAKE_C_STANDARD 99)
endif()

# Default to C++14
if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

# find dependencies
find_package(ament_cmake REQUIRED)
find_package(geometry_msgs REQUIRED)
find_package(rosidl_default_generators REQUIRED)
rosidl_generate_interfaces(${PROJECT_NAME}
  "msg/Num.msg"
  "msg/Sphere.msg"
  "srv/AddThreeInts.srv"
  DEPENDENCIES geometry_msgs # Add packages that above messages depend
   on, in this case geometry_msgs for Sphere.msg
)
# uncomment the following section in order to fill in
# further dependencies manually.
# find_package(<dependency> REQUIRED)

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  # the following line skips the linter which checks for copyrights
  # uncomment the line when a copyright and license is not present in
   all source files
  #set(ament_cmake_copyright_FOUND TRUE)
  # the following line skips cpplint (only works in a git repo)
  # uncomment the line when this package is not in a git repo
  #set(ament_cmake_cpplint_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()

ament_package()
```

A.2.8 testpub.py

```
import rclpy
from rclpy.node import Node

```

```
from first_interfaces.msg import Num     # CHANGED


class MinimalPublisher(Node):

    def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(Num, 'secondtopic', 10)
       # CHANGED
        timer_period = 0.5
        self.timer = self.create_timer(timer_period, self.timer_callback
   )
        self.i = 0

    def timer_callback(self):
        msg = Num()                                             # CHANGED
        msg.num = self.i                                        # CHANGED
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%d"' % msg.num)  # CHANGED
        self.i += 1


def main(args=None):
    rclpy.init(args=args)
    minimal_publisher = MinimalPublisher()
    rclpy.spin(minimal_publisher)
    minimal_publisher.destroy_node()
    rclpy.shutdown()


if __name__ == '__main__':
    main()
```

A.2.9 testsub.py

```
import rclpy
from rclpy.node import Node

from first_interfaces.msg import Num        # CHANGED


class MinimalSubscriber(Node):

    def __init__(self):
        super().__init__('minimal_subscriber')
        self.subscription = self.create_subscription(
            Num,                                                # CHANGED
            'secondtopic',
            self.listener_callback,
            10)
        self.subscription

    def listener_callback(self, msg):
```

```
            self.get_logger().info('I heard: "%d"' % msg.num) # CHANGED


def main(args=None):
    rclpy.init(args=args)

    minimal_subscriber = MinimalSubscriber()

    rclpy.spin(minimal_subscriber)

    minimal_subscriber.destroy_node()
    rclpy.shutdown()


if __name__ == '__main__':
    main()
```

A.2.10 testsrv.py

```
from first_interfaces.srv import AddThreeInts      # CHANGED

import rclpy
from rclpy.node import Node


class MinimalService(Node):

    def __init__(self):
        super().__init__('minimal_service')
        self.srv = self.create_service(AddThreeInts, 'second_service',
    self.add_three_ints_callback)         # CHANGED

    def add_three_ints_callback(self, request, response):
        response.sum = request.a + request.b + request.c
                                          # CHANGED
        self.get_logger().info('Incoming request\na: %d b: %d c: %d' % (
    request.a, request.b, request.c)) # CHANGED

        return response

def main(args=None):
    rclpy.init(args=args)
    minimal_service = MinimalService()
    rclpy.spin(minimal_service)
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

A.2.11 testcli.py

```
from first_interfaces.srv import AddThreeInts       # CHANGED
import sys
import rclpy
from rclpy.node import Node


class MinimalClientAsync(Node):

    def __init__(self):
        super().__init__('minimal_client_async')
        self.cli = self.create_client(AddThreeInts, 'second_service')
       # CHANGED
        while not self.cli.wait_for_service(timeout_sec=1.0):
            self.get_logger().info('service not available, waiting again
   ...')
        self.req = AddThreeInts.Request()
       # CHANGED

    def send_request(self):
        self.req.a = int(sys.argv[1])
        self.req.b = int(sys.argv[2])
        self.req.c = int(sys.argv[3])                   # CHANGED
        self.future = self.cli.call_async(self.req)


def main(args=None):
    rclpy.init(args=args)

    minimal_client = MinimalClientAsync()
    minimal_client.send_request()

    while rclpy.ok():
        rclpy.spin_once(minimal_client)
        if minimal_client.future.done():
            try:
                response = minimal_client.future.result()
            except Exception as e:
                minimal_client.get_logger().info(
                    'Service call failed %r' % (e,))
            else:
                minimal_client.get_logger().info(
                    'Result of add_three_ints: for %d + %d + %d = %d' %
                                # CHANGE
                    (minimal_client.req.a, minimal_client.req.b,
   minimal_client.req.c, response.sum)) # CHANGE
                break

    minimal_client.destroy_node()
    rclpy.shutdown()


if __name__ == '__main__':
    main()
```

A.2.12 setup.py - for tests

```
from setuptools import setup

package_name = 'interface_tests'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='mark',
    maintainer_email='m4rk.domonkos@gmail.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'msgtestpub=interface_tests.testpub:main',
            'msgtestsub=interface_tests.testsub:main',
            'srvtestsrv=interface_tests.testsrv:main',
            'srvtestcli=interface_tests.testcli:main',

        ],
    },
)
```