13장. 순찰 (ON PATROL)

동의대학교 컴퓨터소프트웨어공학과
이종민 교수

목차

- 상태 기계
- SMACH를 이용한 상태 기계 정의
- 상태 기계를 이용한 순찰

patrol.py (10장)

```
#!/usr/bin/env python

import rospy
import actionlib
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal

waypoints = [  # <1>
    [(2.1, 2.2, 0.0), (0.0, 0.0, 0.0, 1.0)],
    [(6.5, 4.43, 0.0), (0.0, 0.0, -0.984047240305, 0.177907360295)]
]

def goal_pose(_pose):  # <2>
    _goal_pose = MoveBaseGoal()  # remove the name conflict by preceding '_'
    _goal_pose.target_pose.header.frame_id = 'map'
    _goal_pose.target_pose.pose.position.x = _pose[0][0]
    _goal_pose.target_pose.pose.position.y = _pose[0][1]
    _goal_pose.target_pose.pose.position.z = _pose[0][2]
    _goal_pose.target_pose.pose.orientation.x = _pose[1][0]
    _goal_pose.target_pose.pose.orientation.y = _pose[1][1]
    _goal_pose.target_pose.pose.orientation.z = _pose[1][2]
    _goal_pose.target_pose.pose.orientation.w = _pose[1][3]

    return _goal_pose
```

지점1 (2.1, 2.2)

지점2 (6.5, 4.43)

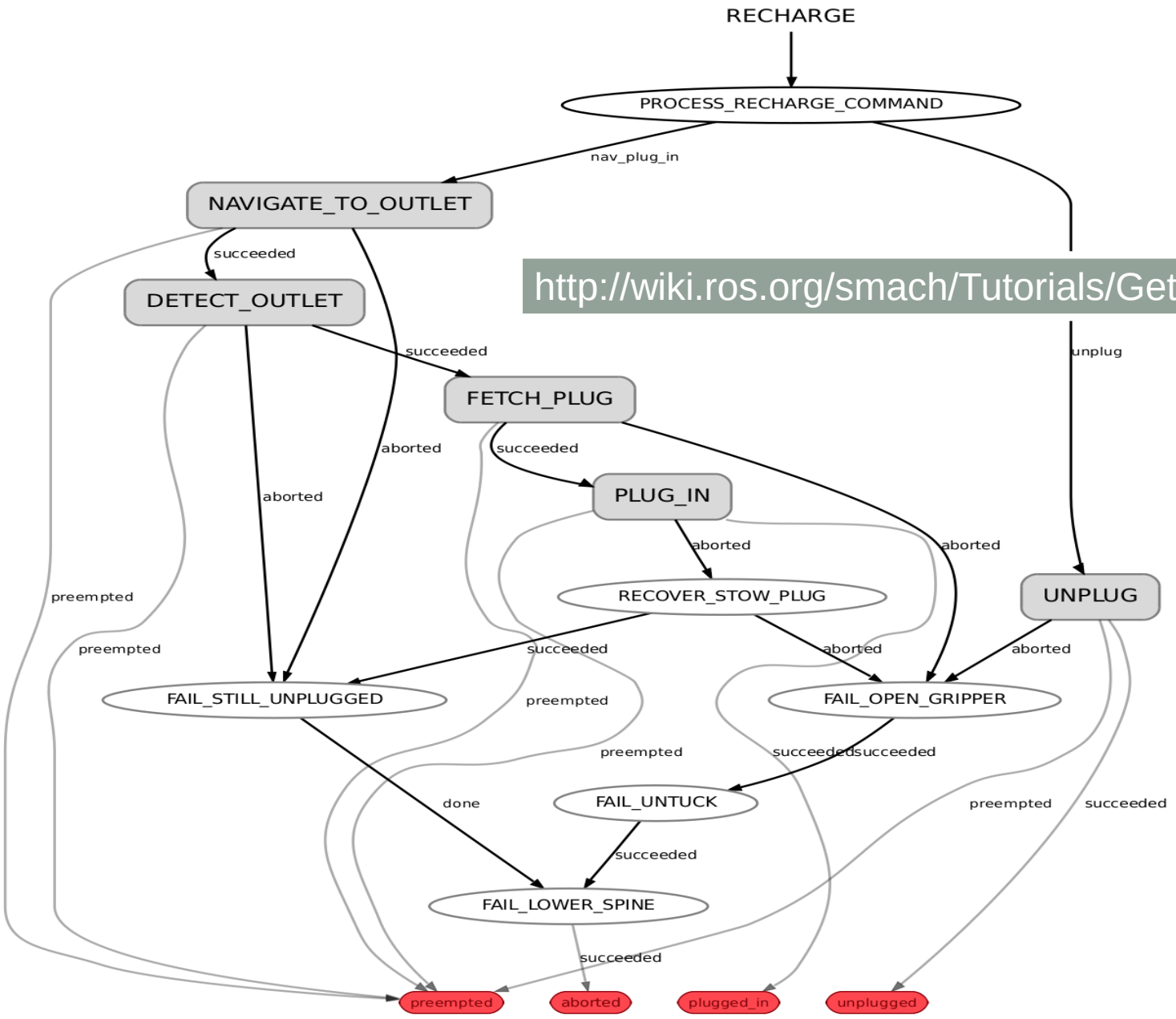
→ 몇 라디언(도)일까?

이동 목표 지점에 대한 MoveBaseGoal 객체 반환 함수 (5장 강의 자료 pp.26~ 참조)

```
26 if __name__ == '__main__':
27     rospy.init_node('patrol')
28
29     client = actionlib.SimpleActionClient('move_base', MoveBaseAction)  # <3>
30     client.wait_for_server()
31
32     while not rospy.is_shutdown():
33         for pose in waypoints:   # <4>
34             goal = goal_pose(pose)
35             client.send_goal(goal)
36             client.wait_for_result()
```

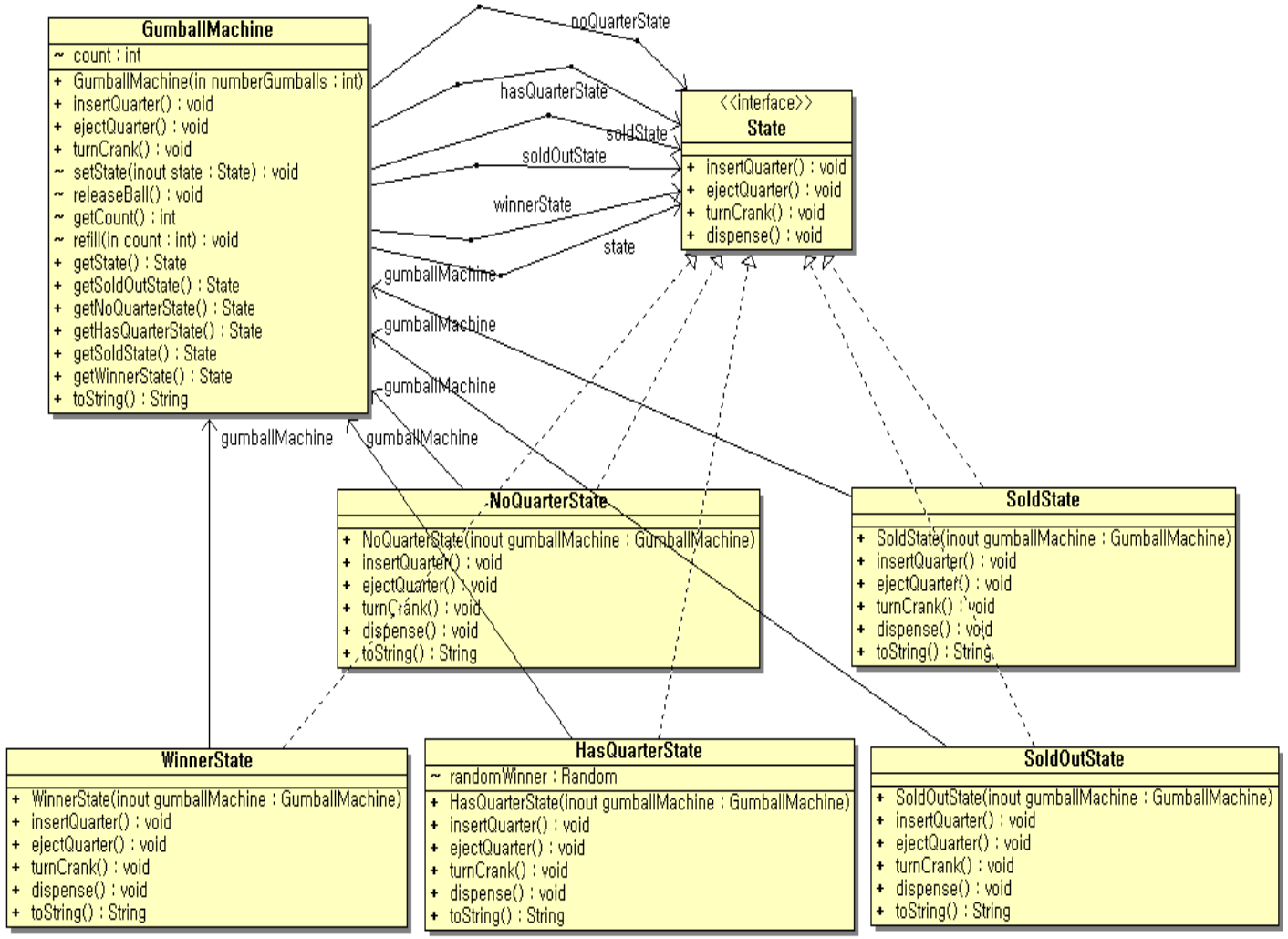
MoveBaseGoal 객체 생성
MoveBaseGoal 전송
이동 결과 대기

상태 기계



http://wiki.ros.org/smach/Tutorials/Getting%20Started

1+1 풍선껌 기계 상태도


GumballMachine
~ count : int
+ GumballMachine(in numberGumballs : int)
+ insertQuarter() : void
+ ejectQuarter() : void
+ turnCrank() : void
~ setState(inout state : State) : void
~ releaseBall() : void
~ getCount() : int
~ refill(in count : int) : void
+ getState() : State
+ getSoldOutState() : State
+ getNoQuarterState() : State
+ getHasQuarterState() : State
+ getSoldState() : State
+ getWinnerState() : State
+ toString() : String
noQuarterState
hasQuarterState
soldState
soldOutState
winnerState
state
gumballMachine
gumballMachine
gumballMachine
gumballMachine
gumballMachine
<<interface>>
State
+ insertQuarter() : void
+ ejectQuarter() : void
+ turnCrank() : void
+ dispense() : void
NoQuarterState
+ NoQuarterState(inout gumballMachine : GumballMachine)
+ insertQuarter() : void
+ ejectQuarter() : void
+ turnCrank() : void
+ dispense() : void
+ toString() : String
SoldState
+ SoldState(inout gumballMachine : GumballMachine)
+ insertQuarter() : void
+ ejectQuarter() : void
+ turnCrank() : void
+ dispense() : void
+ toString() : String
WinnerState
+ WinnerState(inout gumballMachine : GumballMachine)
+ insertQuarter() : void
+ ejectQuarter() : void
+ turnCrank() : void
+ dispense() : void
+ toString() : String
HasQuarterState
~ randomWinner : Random
+ HasQuarterState(inout gumballMachine : GumballMachine)
+ insertQuarter() : void
+ ejectQuarter() : void
+ turnCrank() : void
+ dispense() : void
+ toString() : String
SoldOutState
+ SoldOutState(inout gumballMachine : GumballMachine)
+ insertQuarter() : void
+ ejectQuarter() : void
+ turnCrank() : void
+ dispense() : void
+ toString() : String

smach 패키지

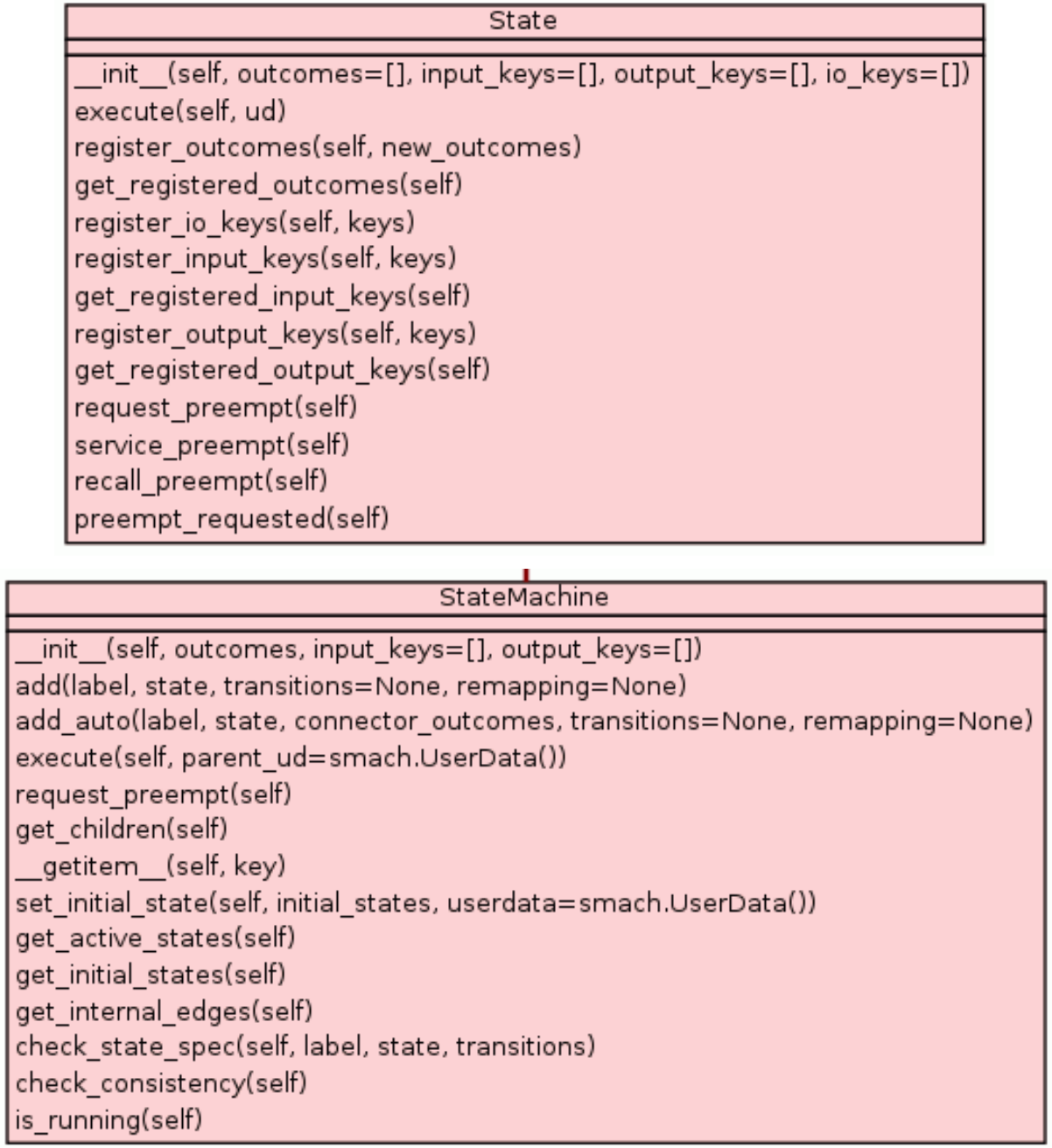
- http://docs.ros.org/groovy/api/smach/html/python/smach-module.html

Package smach

source code

Submodules

- smach.concurrence
- smach.container
- smach.exceptions
- smach.iterator
- smach.log
- smach.sequence
- smach.state
- smach.state_machine
- smach.user_data
- smach.util



State 클래스

__init__(self, outcomes=[], input_keys=[], output_keys=[], io_keys=[])
(Constructor)

State constructor

Parameters:

- `outcomes` (array of strings) - Custom outcomes for this state.
- `input_keys` (array of strings) - The userdata keys from which this state might read at runtime.
- `output_keys` (array of strings) - The userdata keys to which this state might write at runtime.
- `io_keys` (array of strings) - The userdata keys to which this state might write or from which it might read at runtime.

Overrides: object.__init__

execute(self, ud)

Called when executing a state. In the base class this raises a NotImplementedError.

Parameters:

- `ud` (`UserData` structure) - Userdata for the scope in which this state is executing

StateMachine 클래스

add(label, state, transitions=None, remapping=None)
Static Method

Add a state to the opened state machine.

Parameters:

- `label` (string) - The label of the state being added.
- `state` - An instance of a class implementing the `State` interface.
- `transitions` - A dictionary mapping state outcomes to other state labels or container outcomes.
- `remapping` - A dictrionary mapping local userdata keys to userdata keys in the container.

execute(self, parent_ud=smach.UserData()) source code

Run the state machine on entry to this state. This will set the "closed" flag and spin up the execute thread. Once this flag has been set, it will prevent more states from being added to the state machine.

Parameters:

- `ud` - Userdata for the scope in which this state is executing

Overrides: **state.State.execute**

smach_ros 패키지

- http://docs.ros.org/groovy/api/smach_ros/html/python/smach_ros-module.html

Package smach_ros

Package smach_ros

source code

Submodules

- smach_ros.action_server_wrapper
- smach_ros.condition_state
- smach_ros.introspection
- smach_ros.monitor_state
- smach_ros.service_state
- smach_ros.simple_action_state
- smach_ros.util

IntrospectionServer 클래스

- 상태 기계 검사 결과는 smach_viewer 패키지의 smach_viewer.py를 통해 확인 가능

Package smach_ros :: Module introspection :: Class IntrospectionServer

Class IntrospectionServer

source code

Server for providing introspection and control for smach.

Instance Methods	
	`__init__(self, server_name, state, path)` Traverse the smach tree starting at root, and construct introspection proxies for getting and setting debug state.
	`start(self)`
	`stop(self)`
	`construct(self, server_name, state, path)` Recursively construct proxies to containers.
	`clear(self)` Clear all proxies in this server.

SimpleActionState 클래스

- Simple action client state.

```
__init__(self, action_name, action_spec, goal=None, goal_key=None, goal_slots=[], goal_cb=None, goal_cb_args=[], goal_cb_kwargs={},
result_key=None, result_slots=[], result_cb=None, result_cb_args=[], result_cb_kwargs={}, input_keys=[], output_keys=[], outcomes=[]
exec_timeout=None, preempt_timeout=rospy.Duration(60.0), server_wait_timeout=rospy.Duration(60.0))
```

Constructor for SimpleActionState action client wrapper.

Parameters:

- `action_name` (string) - The name of the action as it will be broadcast over ros.
- `action_spec` (actionlib action msg) - The type of action to which this client will connect.
- `goal` (actionlib goal msg) - If the goal for this action does not need to be generated at runtime, it can be passed to this state on construction.

smach 관련 패키지

- 관련 패키지

S	Package
■	ros-kinetic-smach
■	ros-kinetic-smach-msgs
■	ros-kinetic-smach-ros
■	ros-kinetic-smach-viewer
□	ros-kinetic-smacha
□	ros-kinetic-smacha-ros

- 설치 방법

```
$ sudo apt install ros-kinetic-smach ros-kinetic-smach-*
```

SMACH 연습 코드: foo_bar.py

```
1 #!/usr/bin/env python
2
3 import rospy
4 import smach
5 import smach_ros
6
7
8 # define state Foo
9 class Foo(smach.State):
10     def __init__(self):
11         smach.State.__init__(self, outcomes=['outcome1','outcome2'])
12         self.counter = 0
13
14     def execute(self, userdata):
15         rospy.loginfo('Executing state FOO')
16         if self.counter < 3:
17             self.counter += 1
18             return 'outcome1'
19         else:
20             return 'outcome2'
```

import time

time.sleep(1)

```
23 # define state Bar
24 class Bar(smach.State):
25     def __init__(self):
26         smach.State.__init__(self, outcomes=['outcome2'])
27
28     def execute(self, userdata):
29         rospy.loginfo('Executing state BAR')
30         return 'outcome2'
31
32
33
34
35 # main
36 def main():
37     rospy.init_node('smach_example_state_machine')
38
39     # Create a SMACH state machine
40     sm = smach.StateMachine(outcomes=['outcome4', 'outcome5'])
41     # sm = smach.StateMachine(outcomes=['outcome4'])
```

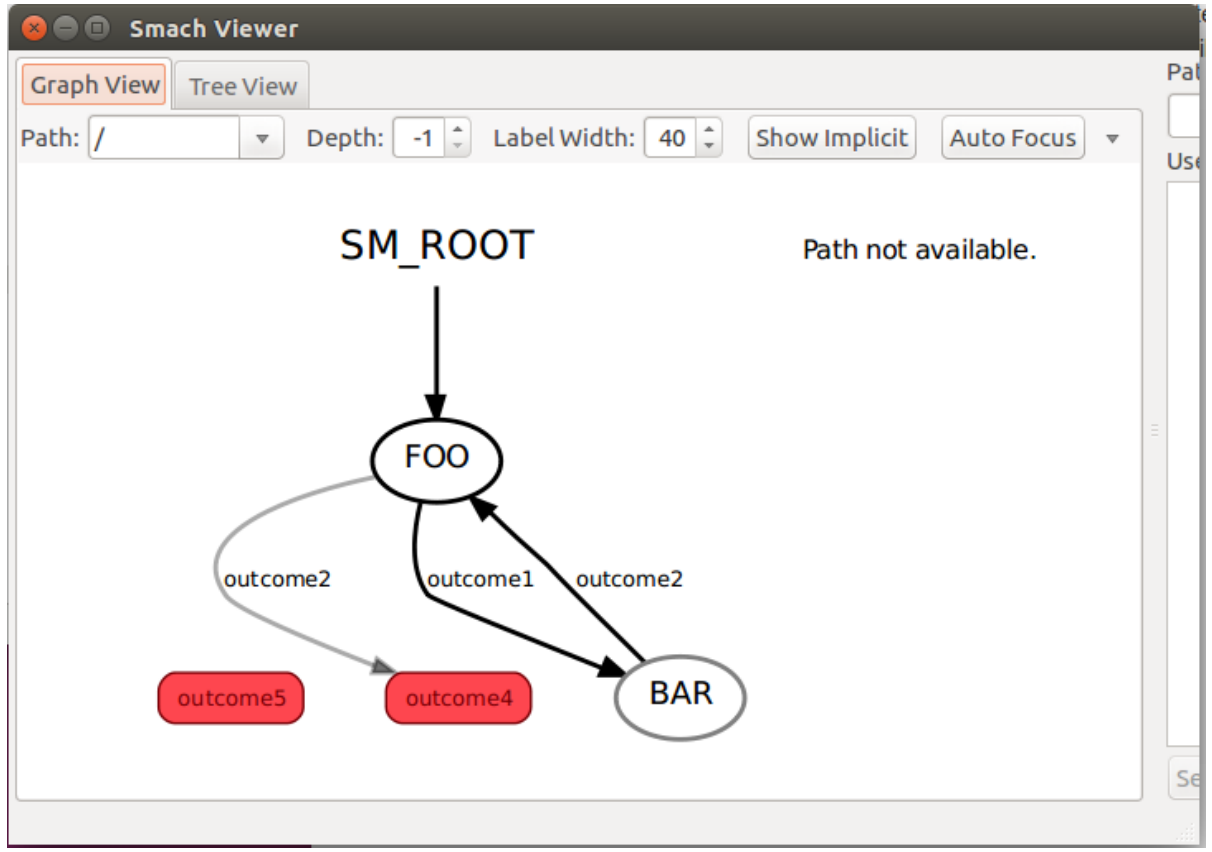
```
43     # Open the container
44     with sm:
45         # Add states to the container
46         smach.StateMachine.add('FOO', Foo(),
47                                transitions={'outcome1':'BAR',
48                                             'outcome2':'outcome4'})
49         smach.StateMachine.add('BAR', Bar(),
50                                transitions={'outcome2':'FOO'})
51
52     # LJM - add an introspection server - optional
53     sis = smach_ros.IntrospectionServer('FOO_BAR_SIS', sm, '/SM_ROOT')
54     sis.start()
55
56     # Execute SMACH plan
57     outcome = sm.execute()
58
59     # LJM - add an introspection server - optional
60     rospy.spin()
61     sis.stop()
62
63
64 if __name__ == '__main__':
65     main()
```

smach_viewer용 코드

smach_viewer용 코드

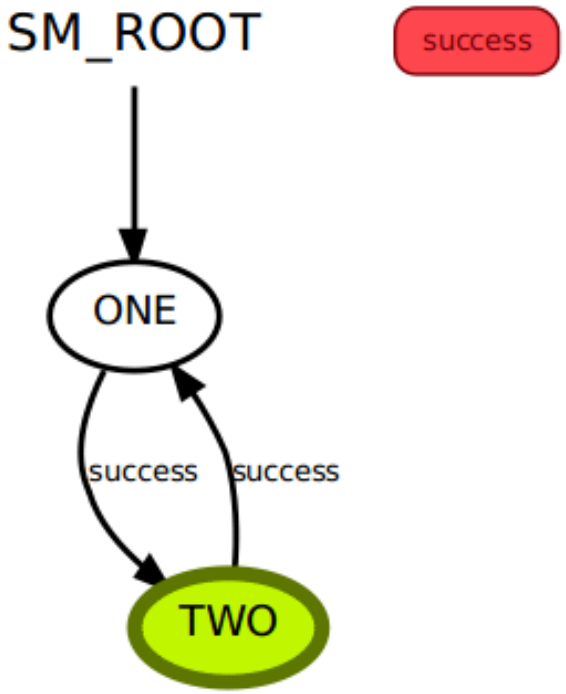
상태 다이어그램

$ rosrun smach_viewer smach_viewer.py



simple_fsm.py (pp.222-223)

```python
#!/usr/bin/env python

import rospy
from smach import State, StateMachine
from time import sleep
import smach_ros  # LJM: add for introspection - optional


class One(State):
    def __init__(self):
        State.__init__(self, outcomes=['success'])

    def execute(self, userdata):
        print 'one'
        sleep(1)
        return 'success'


class Two(State):
    def __init__(self):
        State.__init__(self, outcomes=['success'])

    def execute(self, userdata):
        print 'two'
        sleep(1)
        return 'success'
```



```
29 if __name__ == '__main__':
30     rospy.init_node('simple_fsm')  # LJM - add for introspection - optional
31
32     sm = StateMachine(outcomes=['success'])
33     with sm:
34         StateMachine.add('ONE', One(), transitions={'success': 'TWO'})
35         StateMachine.add('TWO', Two(), transitions={'success': 'ONE'})
36
37     # LJM - add for introspection - optional
38     sis = smach_ros.IntrospectionServer('simple_fms_SIS', sm, '/SM_ROOT')
39     sis.start()
40
41     sm.execute()
42
43     # LJM - add for introspection - optional
44     rospy.spin()
45     sis.stop()
```

실행 방법

[Terminal 01]

```
$ rosrun deu_ros simple_fsm.py
```

[Terminal 02]

```
$ rosrun smach_viewer smach_viewer.py
```

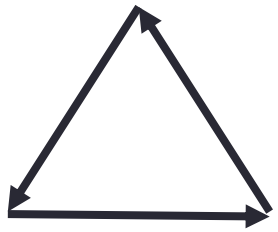
실행 결과

```
roscore http://localhost:11311/
jongmin@ubunt... | jongmin@ubunt... | jongmin@ubunt... | roscore http://lo...
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:34117/
ros_comm version 1.12.14


SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES

auto-starting new master
process[master]: started with pid [2771]
ROS_MASTER_URI=http://localhost:11311/

setting /run_id to 53ce153a-07b4-11ea-8693-000c29bd89fa
process[rosout-1]: started with pid [2784]
started core service [/rosout]
```

shapes.py (pp.226-227)

```python
#!/usr/bin/env python


import rospy
from smach import State, StateMachine

from time import sleep
import smach_ros


class Drive(State):
    def __init__(self, distance):
        State.__init__(self, outcomes=['success'])
        self.distance = distance

    def execute(self, userdata):
        print 'Driving', self.distance
        sleep(1)
        return 'success'
```

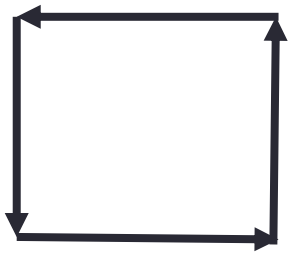
```
22 class Turn(State):
23     def __init__(self, angle):
24         State.__init__(self, outcomes=['success'])
25         self.angle = angle
26
27     def execute(self, userdata):
28         print 'Turning', self.angle
29         sleep(1)
30         return 'success'
31
32
33 if __name__ == '__main__':
34     triangle = StateMachine(outcomes=['success'])
35     with triangle:
36         StateMachine.add('SIDE1', Drive(1), transitions={'success': 'TURN1'})
37         StateMachine.add('TURN1', Turn(120), transitions={'success': 'SIDE2'})
38         StateMachine.add('SIDE2', Drive(1), transitions={'success': 'TURN2'})
39         StateMachine.add('TURN2', Turn(120), transitions={'success': 'SIDE3'})
40         StateMachine.add('SIDE3', Drive(1), transitions={'success': 'success'})
```



하위 상태 기계 triangle 정의

```
42    square = StateMachine(outcomes=['success'])
43    with square:
44        StateMachine.add('SIDE1', Drive(1), transitions={'success': 'TURN1'})
45        StateMachine.add('TURN1', Turn(90), transitions={'success': 'SIDE2'})
46        StateMachine.add('SIDE2', Drive(1), transitions={'success': 'TURN2'})
47        StateMachine.add('TURN2', Turn(90), transitions={'success': 'SIDE3'})
48        StateMachine.add('SIDE3', Drive(1), transitions={'success': 'TURN3'})
49        StateMachine.add('TURN3', Turn(90), transitions={'success': 'SIDE4'})
50        StateMachine.add('SIDE4', Drive(1), transitions={'success': 'success'})
51
52    shapes = StateMachine(outcomes=['success'])
53    with shapes:
54        StateMachine.add('TRIANGLE', triangle, transitions={'success': 'SQUARE'})
55        StateMachine.add('SQUARE', square, transitions={'success': 'success'})
56
57    # LJM - add for introspection - optional
58    rospy.init_node('shapes')
59    sis = smach_ros.IntrospectionServer('shapes', shapes, '/SM_ROOT')
60    sis.start()
61
62    shapes.execute()
63
64    # LJM - add for introspection - optional
65    rospy.spin()
66    sis.stop()
```

하위 상태 기계 square 정의



상태 기계 shapes 정의

실행 방법

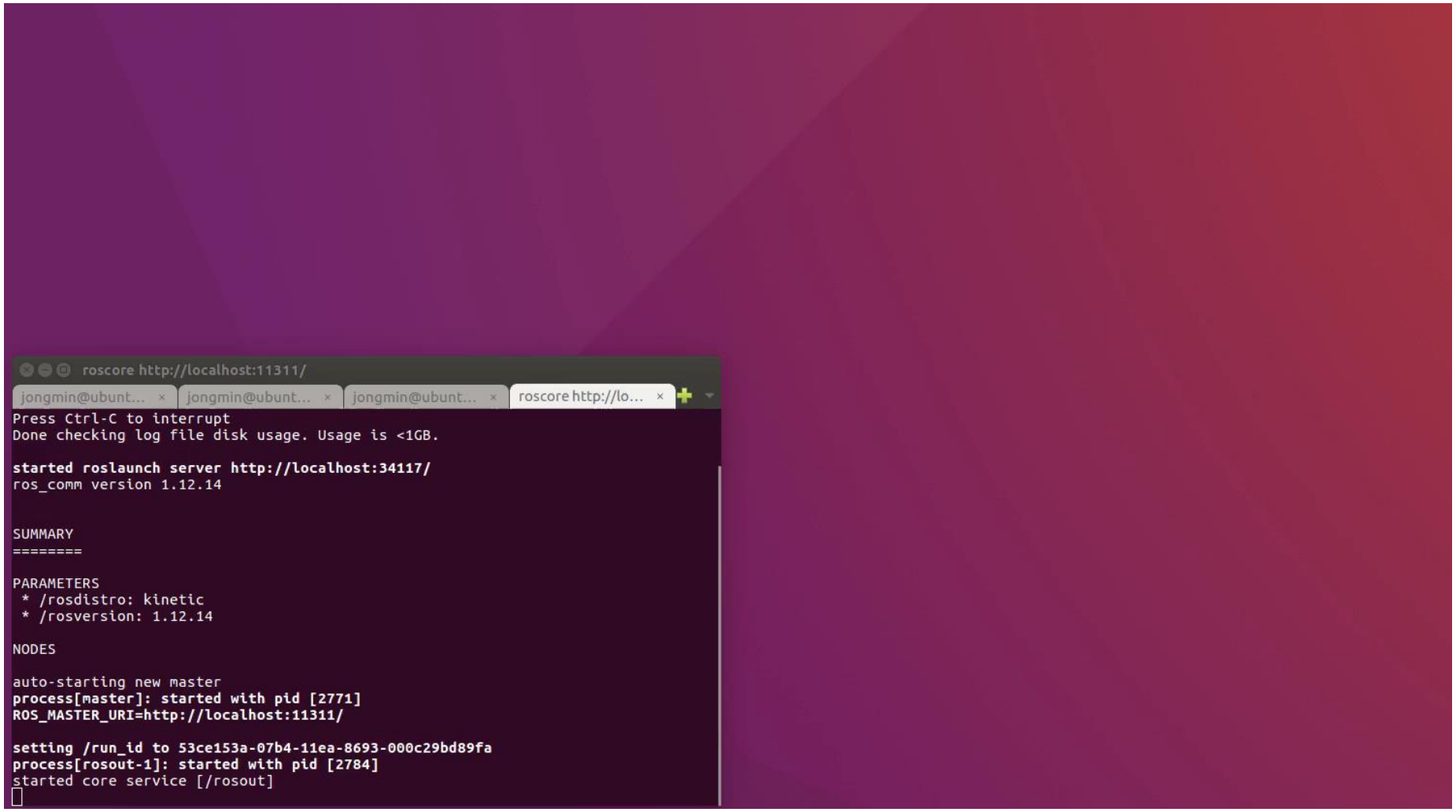
[Terminal 01]

```
$ rosrun deu_ros shapes.py
```

[Terminal 02]

```
$ rosrun smach_viewer smach_viewer.py
```

실행 결과


roscore http://localhost:11311/
jongmin@ubunt...
jongmin@ubunt...
jongmin@ubunt...
roscore http://lo...
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
started roslaunch server http://localhost:34117/
ros_comm version 1.12.14
SUMMARY
========
PARAMETERS
* /rosdistro: kinetic
* /rosversion: 1.12.14
NODES
auto-starting new master
process[master]: started with pid [2771]
ROS_MASTER_URI=http://localhost:11311/
setting /run_id to 53ce153a-07b4-11ea-8693-000c29bd89fa
process[rosout-1]: started with pid [2784]
started core service [/rosout]

shapes2.py (pp.229-230)

```
1 #!/usr/bin/env python
2
3
4 import rospy
5 from smach import State, StateMachine
6
7 from time import sleep
8 import smach_ros
9
10
11 class Drive(State):
12     def __init__(self, distance):
13         State.__init__(self, outcomes=['success'])
14         self.distance = distance
15
16     def execute(self, userdata):
17         print 'Driving', self.distance
18         sleep(1)
19         return 'success'
```

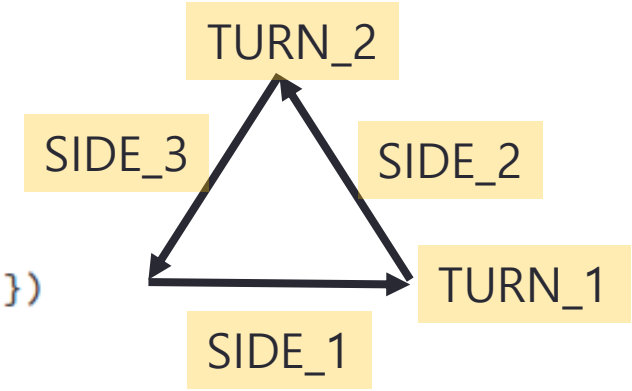
```
22 class Turn(State):
23     def __init__(self, angle):
24         State.__init__(self, outcomes=['success'])
25         self.angle = angle
26
27     def execute(self, userdata):
28         print 'Turning', self.angle
29         sleep(1)
30         return 'success'
31
32
33 def polygon(sides):
34     polygon = StateMachine(outcomes=['success'])
35     with polygon:
36         # Add all but the final side
37         for i in xrange(sides - 1):
38             StateMachine.add('SIDE_{0}'.format(i + 1),
39                              Drive(1),
40                              transitions={'success': 'TURN_{0}'.format(i + 1)})
41
42         # Add all the turns
43         for i in xrange(sides - 1):
44             StateMachine.add('TURN_{0}'.format(i + 1),
45                              Turn(360.0 / sides),
46                              transitions={'success': 'SIDE_{0}'.format(i + 2)})
47
48         # Add the final side
49         StateMachine.add('SIDE_{0}'.format(sides),
50                          Drive(1),
51                          transitions={'success': 'success'})
52     return polygon
```

polygon(3)

'SIDE_1', Drive(1), transitions={'success': 'TURN_1'}
'SIDE_2', Drive(1), transitions={'success': 'TURN_2'}

'TURN_1', Turn(120), transitions={'success': 'SIDE_2'}
'TURN_2', Turn(120), transitions={'success': 'SIDE_3'}

'SIDE_3', Drive(1), transitions={'success': 'success'}



```
55 if __name__ == '__main__':
56     triangle = polygon(3)
57     square = polygon(4)
58
59     shapes = StateMachine(outcomes=['success'])
60     with shapes:
61         StateMachine.add('TRIANGLE', triangle, transitions={'success': 'SQUARE'})
62         StateMachine.add('SQUARE', square, transitions={'success': 'success'})
63
64     # LJM - add for introspection - optional
65     rospy.init_node('shapes2')
66     sis = smach_ros.IntrospectionServer('shapes2', shapes, '/SM_ROOT')
67     sis.start()
68
69     shapes.execute()
70
71     # LJM - add for introspection - optional
72     rospy.spin()
73     sis.stop()
```

실행 방법

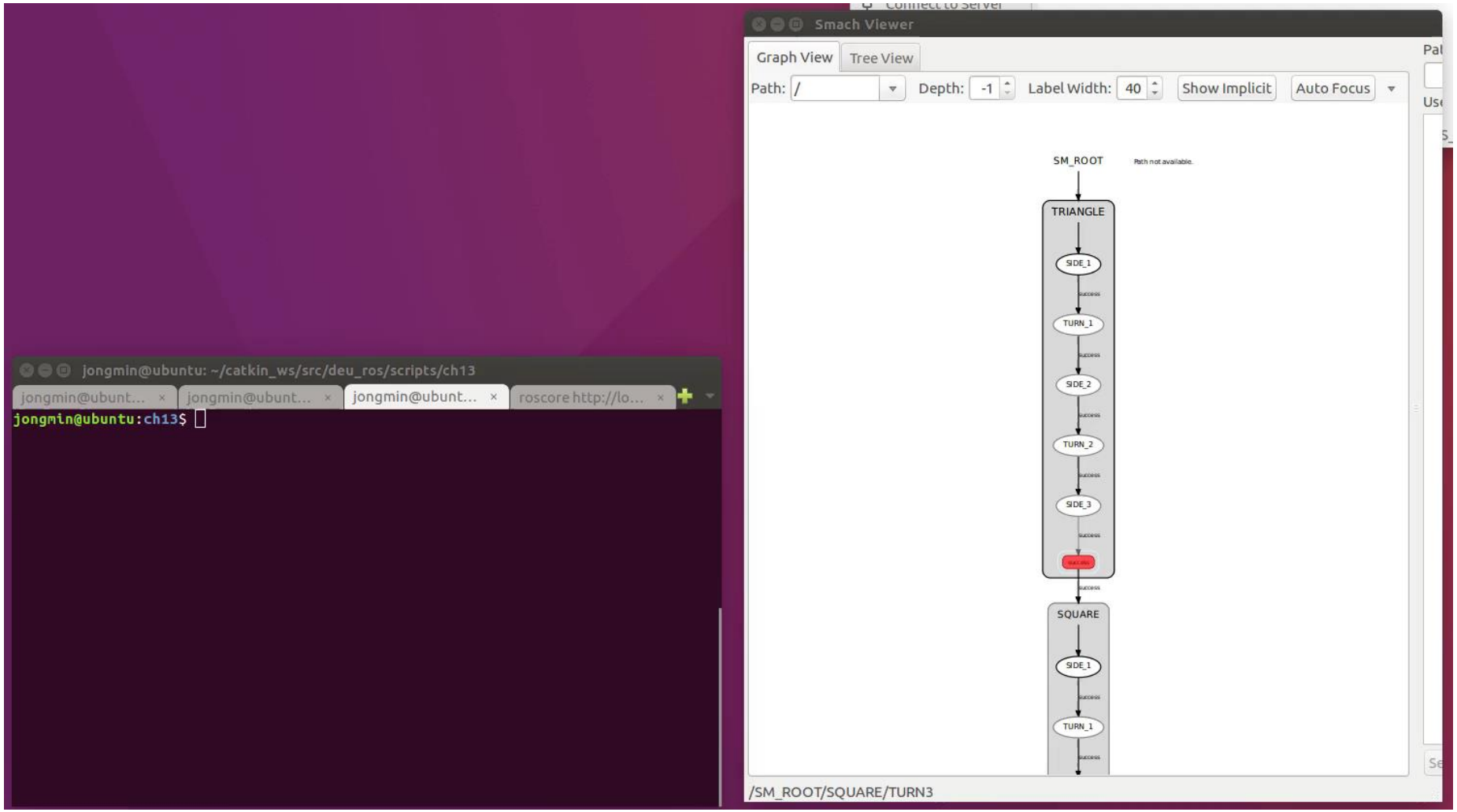
[Terminal 01]

```
$ rosrun deu_ros shapes2.py
```

[Terminal 02]

```
$ rosrun smach_viewer smach_viewer.py
```

실행 결과

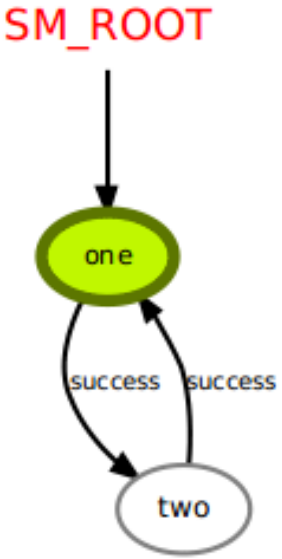

jongmin@ubuntu: ~/catkin_ws/src/deu_ros/scripts/ch13
jongmin@ubunt...
jongmin@ubunt...
jongmin@ubunt...
roscore http://lo...
jongmin@ubuntu:ch13$
Smach Viewer
Graph View
Tree View
Path: /
Depth: -1
Label Width: 40
Show Implicit
Auto Focus
SM_ROOT
Path not available.
TRIANGLE
SIDE_1
TURN_1
SIDE_2
TURN_2
SIDE_3
SQUARE
SIDE_1
TURN_1
/SM_ROOT/SQUARE/TURN3

patrol_fsm.py (pp.231-232)

```python
#!/usr/bin/env python

import actionlib
import rospy
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal
from smach import State, StateMachine
import smach_ros


waypoints = [
    ['one', (2.1, 2.2), (0.0, 0.0, 0.0, 1.0)],
    ['two', (6.5, 4.43), (0.0, 0.0, -0.984047240305, 0.177907360295)]
]


class Waypoint(State):
    def __init__(self, position, orientation):
        State.__init__(self, outcomes=['success'])

        # Get an action client
        self.client = actionlib.SimpleActionClient('move_base', MoveBaseAction)
        self.client.wait_for_server()

        # Define the goal
        self.goal = MoveBaseGoal()
        self.goal.target_pose.header.frame_id = 'map'
        self.goal.target_pose.pose.position.x = position[0]
        self.goal.target_pose.pose.position.y = position[1]
        self.goal.target_pose.pose.position.z = 0.0
        self.goal.target_pose.pose.orientation.x = orientation[0]
        self.goal.target_pose.pose.orientation.y = orientation[1]
        self.goal.target_pose.pose.orientation.z = orientation[2]
        self.goal.target_pose.pose.orientation.w = orientation[3]
```

```
35     def execute(self, userdata):
36         self.client.send_goal(self.goal)
37         self.client.wait_for_result()
38         return 'success'
39
40
41 if __name__ == '__main__':
42     rospy.init_node('patrol_fsm')
43
44     patrol = StateMachine('success')
45     with patrol:
46         for i, w in enumerate(waypoints):
47             StateMachine.add(w[0],
48                              Waypoint(w[1], w[2]),
49                              transitions={'success':
50                                  waypoints[(i+1) % len(waypoints)][0]})
51
52     # LJM - add for introspection - optional
53     sis = smach_ros.IntrospectionServer('patrol_fsm', patrol, '/SM_ROOT')
54     sis.start()
55
56     patrol.execute()
57
58     # LJM - add for introspection - optional
59     rospy.spin()
60     sis.stop()
```

'one', Waypoint(...), transitions={'success': 'two'}
'two', Waypoint(...), transitions={'success': 'one'}



실행 방법

[Terminal 01]

```
$ roslaunch turtlebot_stage turtlebot_in_stage.launch
```

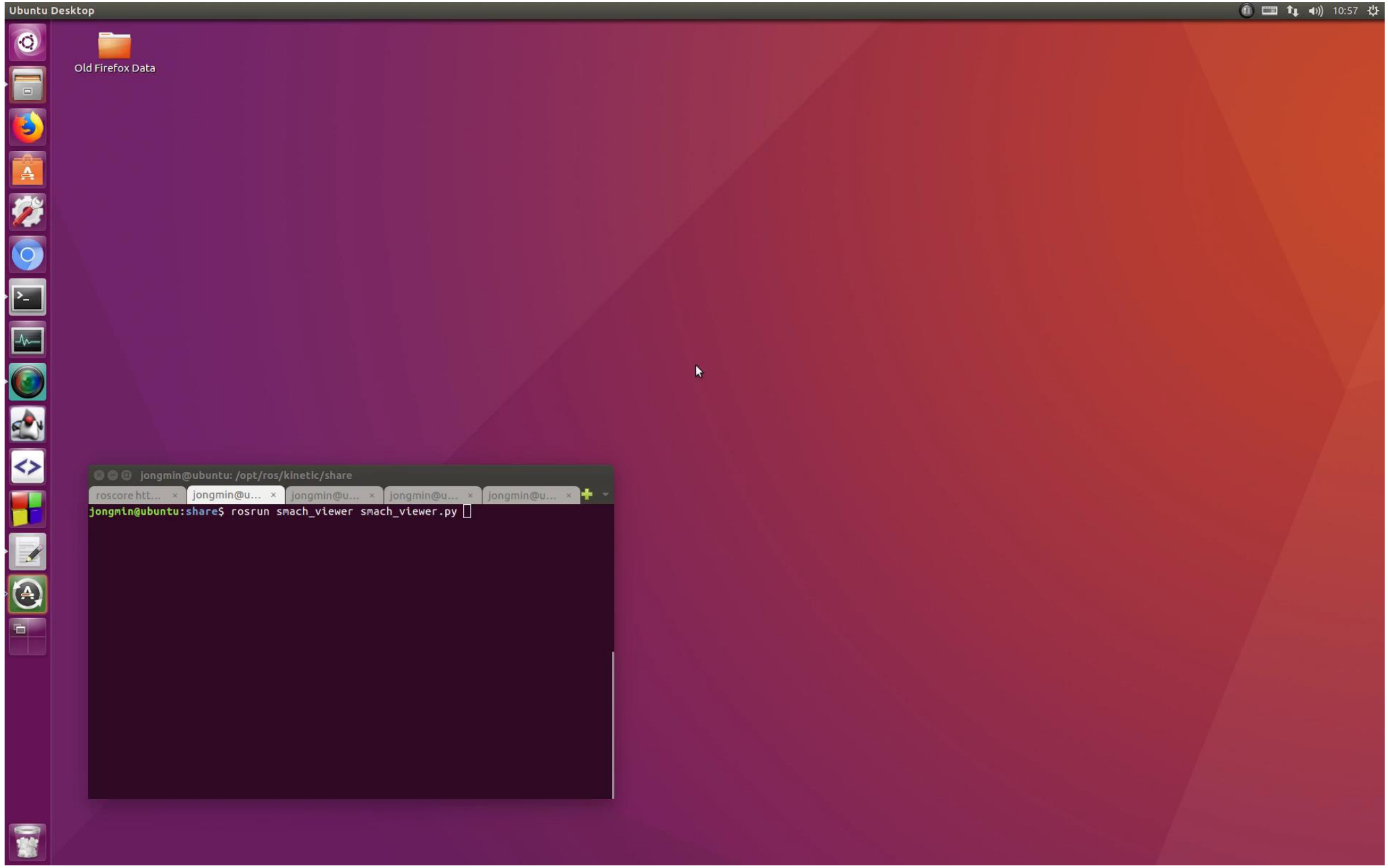
[Terminal 02]

```
$ rosrun deu_ros patrol_fsm.py
```

[Terminal 03]

```
$ rosrun smach_viewer smach_viewer.py
```

실


Ubuntu Desktop
10:57
Old Firefox Data
jongmin@ubuntu: /opt/ros/kinetic/share
roscore htt...
jongmin@u...
jongmin@u...
jongmin@u...
jongmin@u...
jongmin@ubuntu:share$ rosrun smach_viewer smach_viewer.py

better_patrol_fsm2.py (pp.233-234)

```python
#!/usr/bin/env python

import rospy
from smach import StateMachine # <1>
from smach_ros import SimpleActionState # <2>
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal
import smach_ros


waypoints = [
    ['one', (2.1, 2.2), (0.0, 0.0, 0.0, 1.0)],
    ['two', (6.5, 4.43), (0.0, 0.0, -0.984047240305, 0.177907360295)]
]


def get_goal_pose(_pose):
    """
    function of patrol.py in ch.10

    :param _pose: a tuple of position, orientation
    :return: Goal Pose of MoveBaseGoal message type
    """
    _goal_pose = MoveBaseGoal() # remove the name conflict by preceding '_'
    _goal_pose.target_pose.header.frame_id = 'map'
    _goal_pose.target_pose.pose.position.x = _pose[0][0]
    _goal_pose.target_pose.pose.position.y = _pose[0][1]
    _goal_pose.target_pose.pose.position.z = 0.0
    _goal_pose.target_pose.pose.orientation.x = _pose[1][0]
    _goal_pose.target_pose.pose.orientation.y = _pose[1][1]
    _goal_pose.target_pose.pose.orientation.z = _pose[1][2]
    _goal_pose.target_pose.pose.orientation.w = _pose[1][3]
    return _goal_pose
```

```
35 if __name__ == '__main__':
36     rospy.init_node('better_patrol_fsm2')
37
38     patrol = StateMachine(['succeeded', 'aborted', 'preempted'])
39     with patrol:
40         for i,w in enumerate(waypoints):
41             goal_pose = get_goal_pose((w[1], w[2]))
42
43             StateMachine.add(w[0],
44                              SimpleActionState('move_base',
45                                                MoveBaseAction,
46                                                goal=goal_pose),
47                              transitions={'succeeded':waypoints[(i + 1) % \
48                                len(waypoints)][0]})
49
50     # LJM - add for introspection - optional
51     sis = smach_ros.IntrospectionServer('patrol_sis', patrol, '/SM_ROOT')
52     sis.start()
53
54     patrol.execute()
55
56     # LJM - add for introspection - optional
57     rospy.spin()
58     sis.stop()
```

실행 방법

[Terminal 01]

```
$ roslaunch turtlebot_stage turtlebot_in_stage.launch
```

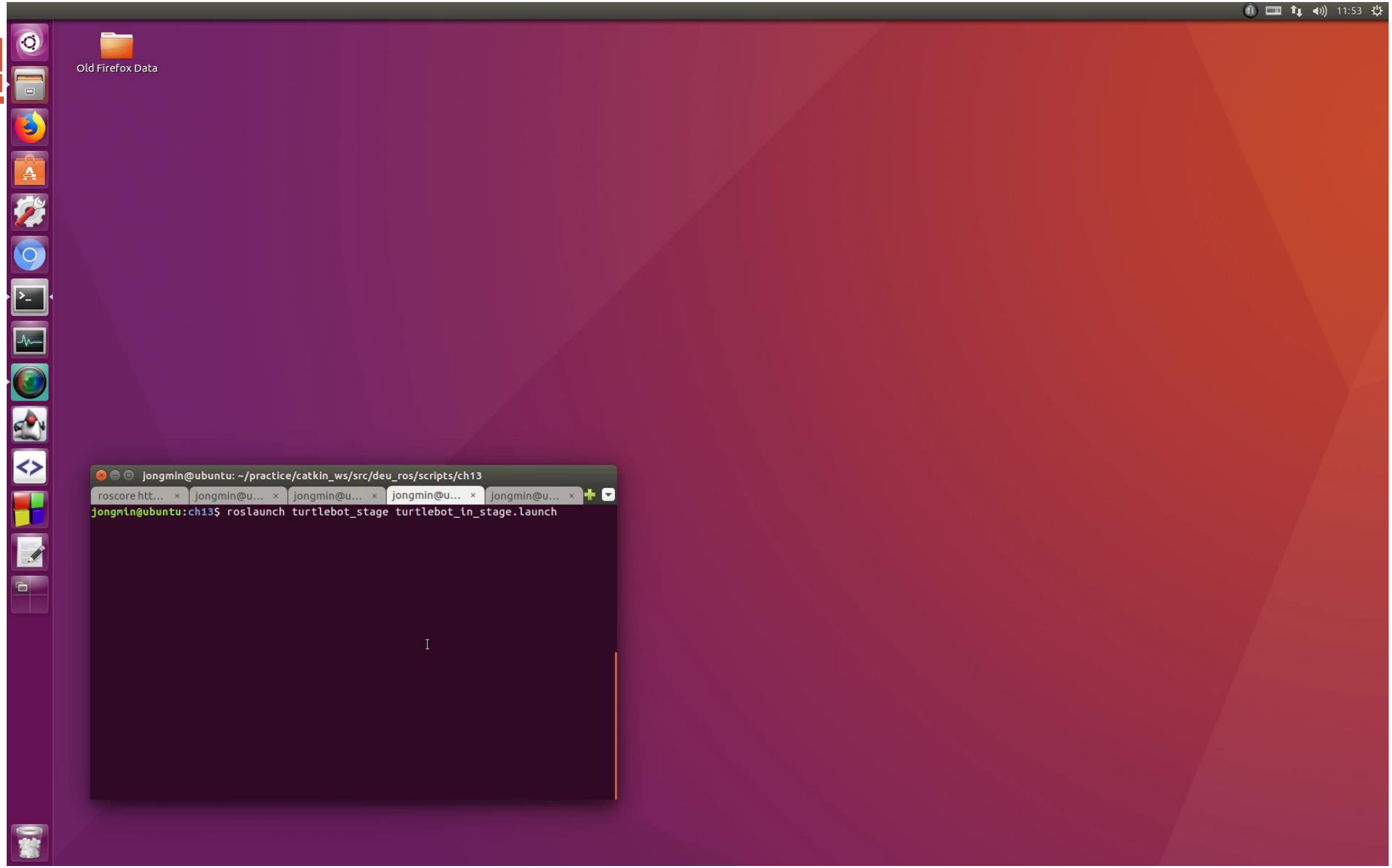
[Terminal 02]

```
$ rosrun deu_ros better_patrol_fsm2.py
```

[Terminal 03]

```
$ rosrun smach_viewer smach_viewer.py
```

실


Old Firefox Data
jongmin@ubuntu: ~/practice/catkin_ws/src/deu_ros/scripts/ch13
roscore htt...
jongmin@u...
jongmin@u...
jongmin@u...
jongmin@u...
jongmin@ubuntu:ch13$ roslaunch turtlebot_stage turtlebot_in_stage.launch
11:53