Summer 2015

Bar-Ilan University

Multi-Robot Systems with ROS
Lesson 6

Teaching Assistant: Roi Yehoshua
roiyeho@gmail.com

Agenda

- Decision making in ROS
- CogniTAO installation
- TAO plans definition

(C)2015 Roi Yehoshua

Decision Making

- http://wiki.ros.org/decision_making
- The goal of this package is to implement light-weight, generic and extendable tools for writing, executing, debugging and monitoring decision making models through ROS standard tools
- Decision making package is being actively developed by Cogniteam
  - For single robot – it is a public package
  - For multi robot – needs a commercial license

Installation

- We will start with the single robot case
- Create a new catkin workspace called dmw
- Check out and compile decision making packages

```
$ cd ~/dmw/src
$ git clone https://github.com/cogniteam/decision_making.git
$ cd ..
$ catkin_make
```

(C)2015 Roi Yehoshua

Installation

- If you have changed the ROS_PACKAGE_PATH variable in .bashrc, make sure its definition is at the end of the file:

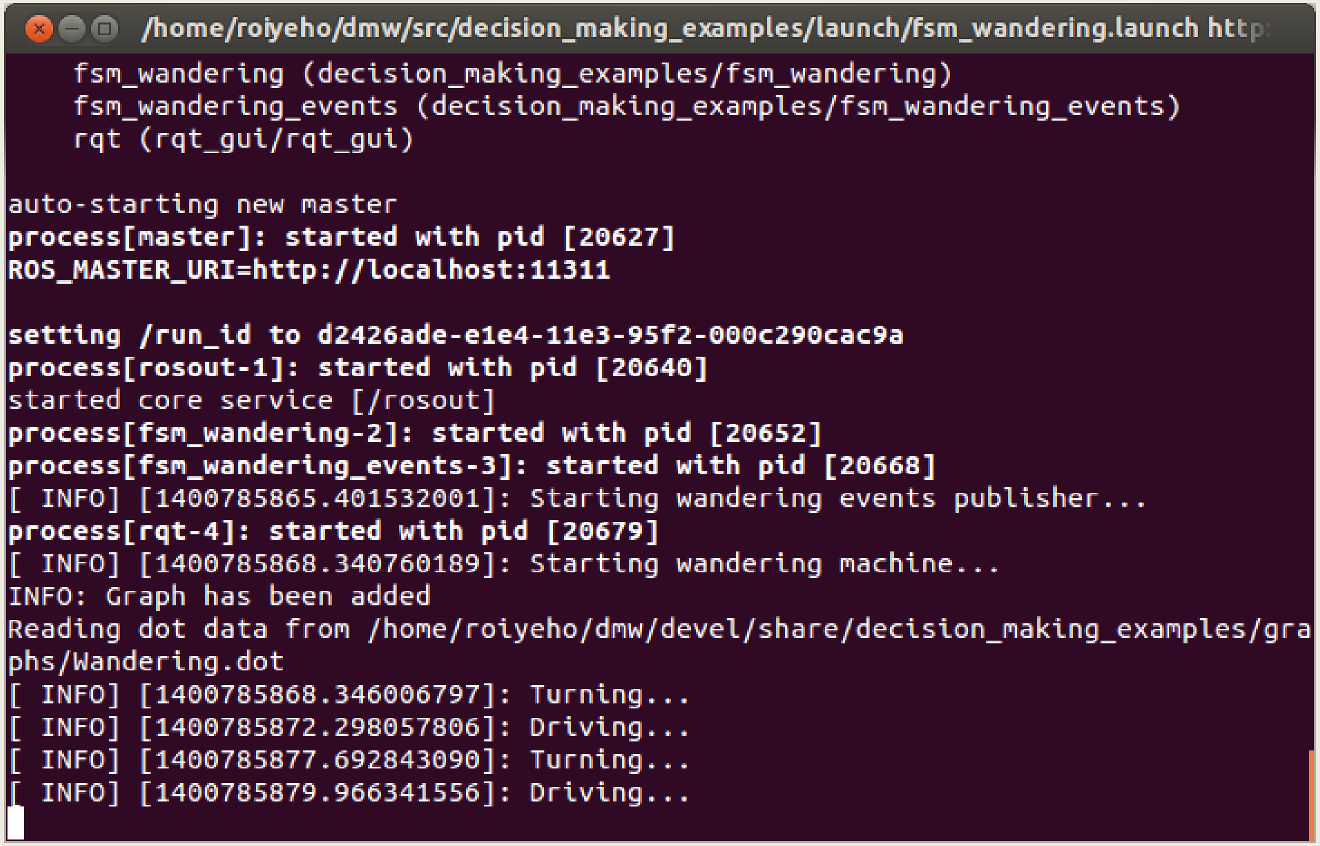
```
# ROS setup
source /opt/ros/hydro/setup.bash
source ~/catkin_ws/devel/setup.bash

source /home/roiyeho/dmw/install/setup.bash
source /home/roiyeho/dmw/devel/setup.bash

export ROS_PACKAGE_PATH=~/ros/stacks:${ROS_PACKAGE_PATH}
export EDITOR='gedit'
```

DM Examples

- Under the ~/dmw/src you will find two folders with code examples
  - decision_making_examples – for single robot
  - dm_teamwork_examples – for multi robots
- You can launch any of the examples using the standard roslaunch command
- For example, to launch the Wandering Robot FSM example, type:

```
$ roslaunch decision_making_examples fsm_wandering.launch
```

FSM Wandering Example

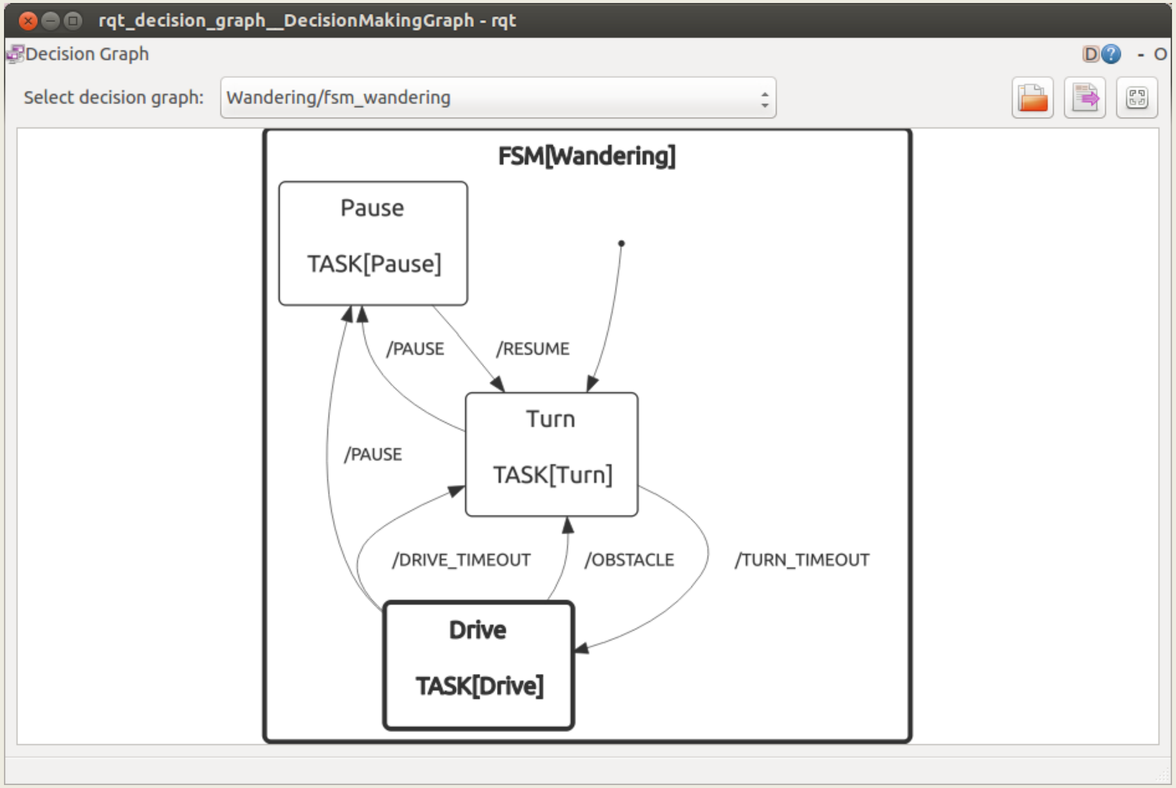


/home/roiyeho/dmw/src/decision_making_examples/launch/fsm_wandering.launch http:

```
    fsm_wandering (decision_making_examples/fsm_wandering)
    fsm_wandering_events (decision_making_examples/fsm_wandering_events)
    rqt (rqt_gui/rqt_gui)

auto-starting new master
process[master]: started with pid [20627]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to d2426ade-e1e4-11e3-95f2-000c290cac9a
process[rosout-1]: started with pid [20640]
started core service [/rosout]
process[fsm_wandering-2]: started with pid [20652]
process[fsm_wandering_events-3]: started with pid [20668]
[ INFO] [1400785865.401532001]: Starting wandering events publisher...
process[rqt-4]: started with pid [20679]
[ INFO] [1400785868.340760189]: Starting wandering machine...
INFO: Graph has been added
Reading dot data from /home/roiyeho/dmw/devel/share/decision_making_examples/gra
phs/Wandering.dot
[ INFO] [1400785868.346006797]: Turning...
[ INFO] [1400785872.298057806]: Driving...
[ INFO] [1400785877.692843090]: Turning...
[ INFO] [1400785879.966341556]: Driving...
```

rqt Decision Graph

- Once the model is running, its visualization is displayed using the Decision Making rqt plugin



Select decision graph: Wandering/fsm_wandering

Decision Making Models

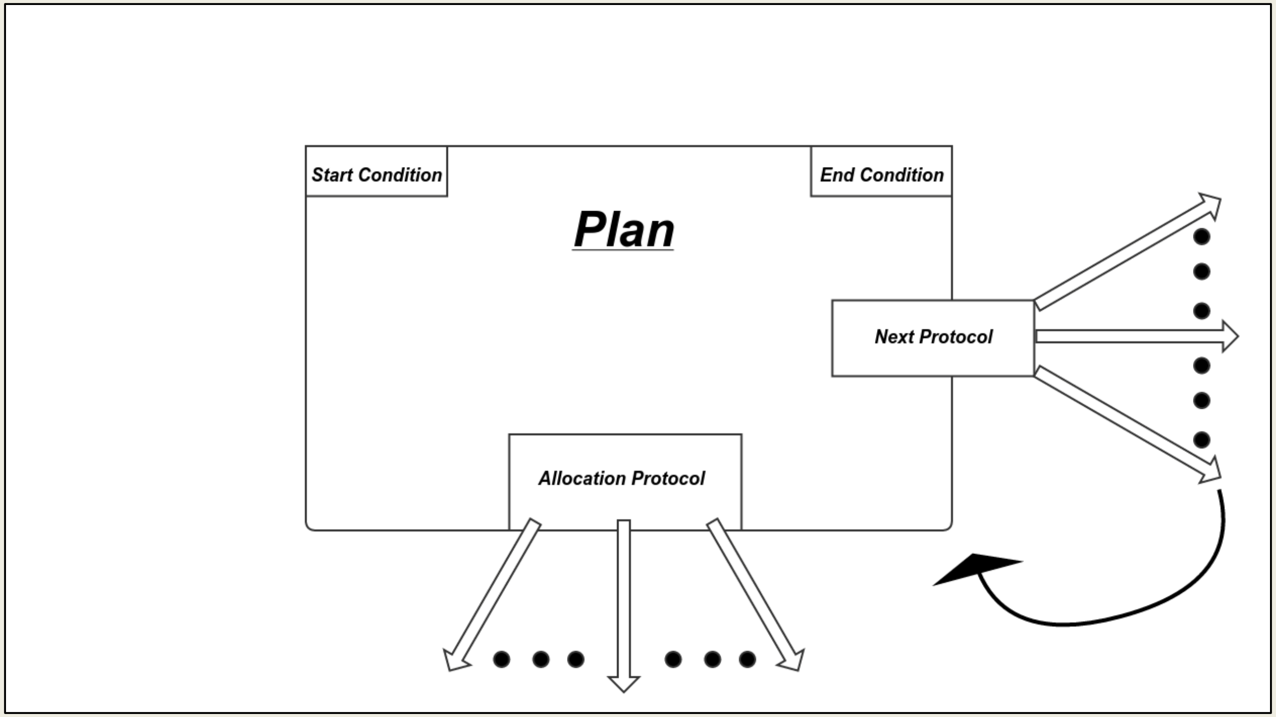
- The decision making system supports different types of models:
  - FSM – Finite State Machines
  - HSM – Hierarchical FSM
  - Behavior Trees
  - CogniTAO – implementation of BDI architecture
- In this course we will focus on CogniTAO

CogniTAO

- CogniTAO (**T**hink **A**s **O**ne) is an implementation of the BDI architecture for both single robot missions and for multiple robots working in teams
- Main features:
  - Simulate entities that can execute complex missions in dynamic environments, where it is impossible to foresee all possible decisions
  - Coupling between the decision-making and the world modeling components
  - Mixing goal-oriented and reactive control, according to the principals of BDI

Plan

- Defines the current Task to be performed, contains Start and Stop conditions, and is coupled with corresponding Plans through *Allocation* and *Next* protocols



TAO

- A TAO is a level of a number of Plans and their corresponding *"Sons"*, *"Allocations"*, and *"Next"* protocols
- A deeper level of sub-plans creates another TAO
- Each TAO defines its starting plan via the TAO_START_PLAN tag

Start and Stop Conditions

- Boolean conditions defined inside each *"Task"*
- Start conditions are validated before the plan is selected for running
- Stop conditions are validated throughout the entire running time of the plan
- The start and stop conditions are usually based on the world model (more on this later)

(C)2015 Roi Yehoshua

Next Protocol

- The *Next* Protocol takes place when a current plan ends, and essentially chooses one next plan to be performed
- It has the possibility to loop its own plan (i.e. 'return' back and re-run the plan)
- Built-in *Next* protocols:
  - NextFirstReady
- You can also create your own *Next* protocols

(C)2015 Roi Yehoshua

Allocation Protocol

- The *Allocation* Protocol takes place in a running plan, and essentially chooses (or divides) a sub-plan that will be performed
- It does not have the possibility to loop the given plan node
- If all children of an Allocation Protocol 'die' (i.e. there is no additional child that continues to its own *Next* protocol) then the father 'dies'
- Built-in *Allocation* protocols:
  - AllocFirstReady

(C)2015 Roi Yehoshua

Next and Allocation Protocols in Teams

- In teams, all members of the team correspond to the same *Next* protocol, while *Allocation* divides the team into sub-groups that correspond to the same *Next* protocols respectively

TAO Machine Definition

- TAOs are defined in a .cpp file using the following syntax:

```
TAO(TAO_NAME)
{
    TAO_PLANS { PLAN_1, PLAN_2, ... }
    TAO_START_PLAN(PLAN_NAME);
    TAO_BGN {
        PLANS
    }
    TAO_END
}
```

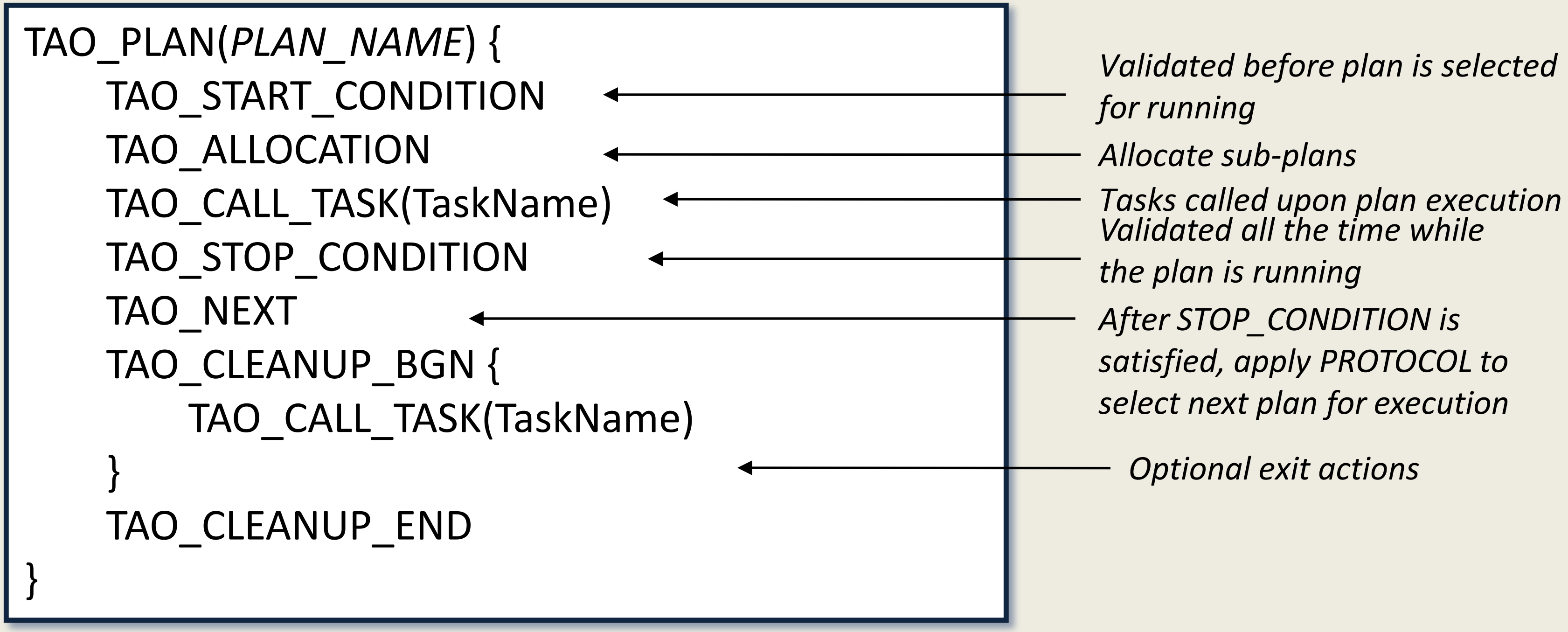
(C)2015 Roi Yehoshua

TAO Machine Definition

- To reference another TAO before defining it, use the following declaration:

```
TAO_HEADER(TAO_NAME);
```

TAO Plan Definition



```
TAO_PLAN(PLAN_NAME) {
    TAO_START_CONDITION
    TAO_ALLOCATION
    TAO_CALL_TASK(TaskName)
    TAO_STOP_CONDITION
    TAO_NEXT
    TAO_CLEANUP_BGN {
        TAO_CALL_TASK(TaskName)
    }
    TAO_CLEANUP_END
}
```

Validated before plan is selected for running

Allocate sub-plans

Tasks called upon plan execution

Validated all the time while the plan is running

After STOP_CONDITION is satisfied, apply PROTOCOL to select next plan for execution

Optional exit actions

- Must be located inside a TAO_BGN-TAO_END block
- All task calls after TAO_ALLOCATION section run in parallel as the plan begins

(C)2015 Roi Yehoshua

TAO Protocols

- TAO_ALLOCATE syntax:

```
TAO_ALLOCATE(PROTOCOL) {
	TAO_SUBPLAN(TAO_1),
	TAO_SUBPLAN(TAO_2),
	…
}
```

- Each sub-plan is a start plan of a selected *TAO*
- If there are no sub-plans to allocate, then use the tag TAO_ALLOCATE_EMPTY

(C)2015 Roi Yehoshua

TAO Protocols

- TAO_NEXT syntax:

```
TAO_NEXT(PROTOCOL) {
        TAO_NEXT_PLAN(PLAN_1),
        TAO_NEXT_PLAN(PLAN_2),
        …
}
```

- If there are no next plans to execute, then use the tag TAO_NEXT_EMPTY