ROS workspaces where the application resides

* Catkin Workspace – build system that ros uses

Builds the environment where the nodes will reside

ROS software is organized into packages

Packages (inside the workspace and made up of ros nodes) – consist of code, data and documentation

Existing packages can be used to include certain functionalities

catkin Package – used to set up a workspace

1. package.xml file that contains info about the package dependencies that the code will require

tags:

<buildtool\_depend> which build tool should be used which is catkin in our case

<build\_depend> libraries that the package depends on

<build\_export\_depend>

1. CMakeList.txt file contains info about the content of the package and how catkin should interact with this package when building the workspace.

Creating a catkin workspace

mkdir -p catkin\_ws/src

cd catkin\_ws

catkin\_make – prepares a workspace to put packages in it

Folders created

1. Build space - libraries
2. Devel space – setup files

.bashrc file runs first initially on opening a terminal

nano – text editor

environment variables are global to the entire system: ros checks the environment variables before executing

creating a package – catkin\_create\_pkg <package\_name> <dependencies>

dependencies examples

rospy – python library for ros

std\_msgs – ros defined classes (Eg String.msg to exchange strings data between nodes)

nv\_msgs – contain the position information of the robot

geometry\_msgs – to give velocity data and more

Building a Package

catkin\_make – makes a workspace and makes sure all the packages have their libraries and there are no errors

roscd – to enter into a particular workspace

python nodes have a scripts file

shebangs (#!) – used to point to the interpreter that the code is supposed to use to execute

Starting an object in a class example:

publisher\_name = class.Publisher(‘Topic Name’, message type, queue size)

queue\_size – if a subscriber is unable to receive messages from the publisher queue size defines the limit as to the number of messages that can wait in the queue after the limit it’ll throw an exception

Example: pub = rospy.Publisher(‘Chat’, String, queue\_size = 10)