Lecture 05.01 ROS methodology

ROS has several key aspects to its methodology that are worth considering at this point.

05.01.1 Distributed computing

nodes **messages** **P2P**

ROS *nodes* are software modules running on potentially different computers. Nodes communicate by sending *messages* over a network *peer-to-peer* (P2P) – that is, directly to each other. This lack of centralization is very flexible and scalable. Nodes, messages, and related concepts are described further in Lecture 06.01.

05.01.2 Use with other programs

ROS systems can easily interact with software tools for visualization, navigation, data logging, etc. This strength allows ROS to remain focused on its core tasks.

05.01.3 Multilinguality

ROS programs can be written in several languages, including Python, C++, and Matlab. The most popular are Python and C++, and we will use the former.

client libraries

The development of ROS programs with a specific language is enabled by a language-specific *client library*. All but the Python (`rospy`), C++ (`roscpp`), and LISP (`roslisp`) client libraries are considered experimental.[1]

05.01.4 Modularity

ROS developers (you!) are encouraged to write their programs in a modular manner such that each module performs some limited task, then *composing* several modules to perform more-complex tasks. This makes debugging, maintenance, and collaboration much easier.

packages

Previously developed ROS programs are available in the default ROS installation and in the form of additional *packages*. We will discuss packages more in Chapter 06.

[1]For more client libraries, see `wiki.ros.org/Client Libraries`.

05.01.5 Open sourceness

ROS is open-source! The licensing is such that commercial, proprietary software can include it, making it a good choice for research and industry.