UNIVERSITY OF
SOUTH CAROLINA

CSCE 574 ROBOTICS

Research Robotic Software Design, Development, and Testing

Alberto Quattrini Li
albertoq@cse.sc.edu

Research Process

- Identify problem of interest
  - Why is it important to solve it
  - What has been done in the literature
- Study, design, and develop the algorithm for robotic applications
  - Approximability, approximation
  - Space and computational complexity
  - Heuristics
- Deploy algorithm
  - Simulation
  - Fielded robots
- Evaluate performance of algorithm on experiments

Software for a robot

- Robots are complex systems that involve a large number of individual capabilities
- Robot architecture is the set of principles, building blocks, and tools for designing robots
  - Architectural structure: system into subsystems with interaction
  - Architectural style: how communication happens
- Currently in a robot there might be multiple robot architectures
- However, a well-conceived architecture can have significant advantages for specification, execution, and validation

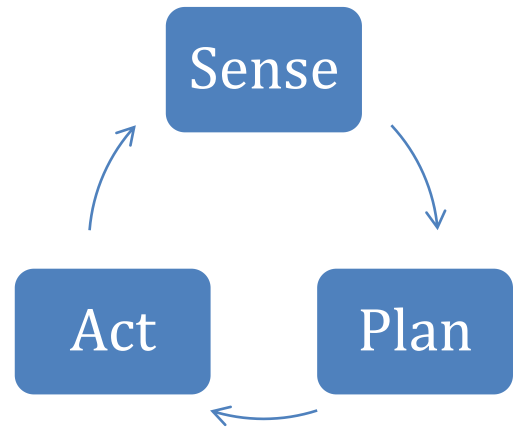
Robot Architectures Decomposition

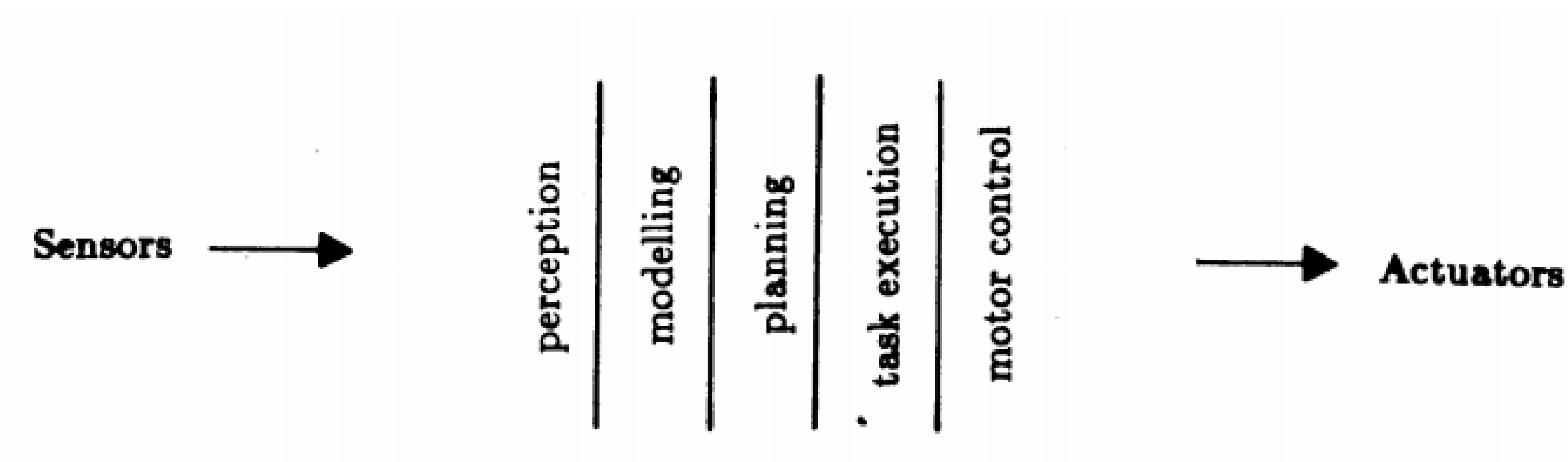
- *Modular* decomposition reduces complexity bu decomposing systems into simpler independent pieces

- *Hierarchical* decomposition reduces system complexity through abstraction

Main Robot Architectures

- Deliberative
  - Top-down approach

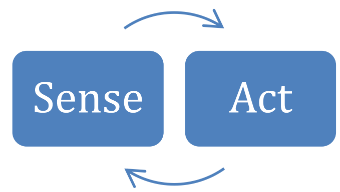


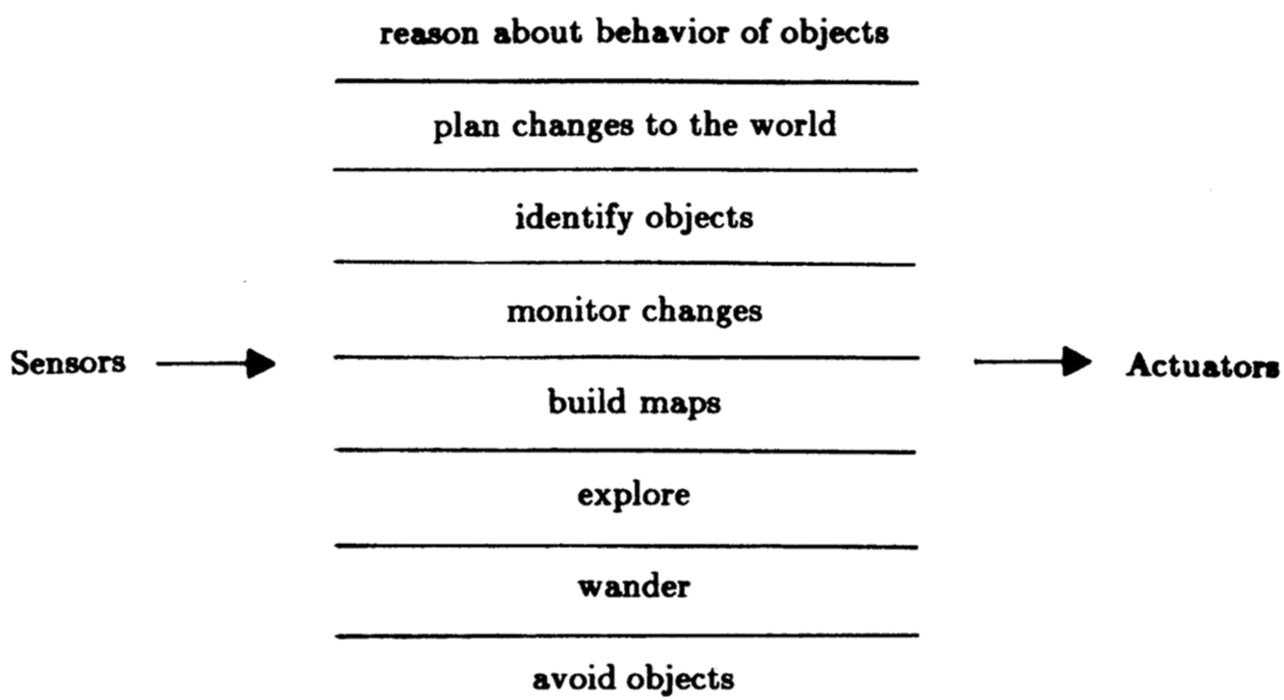


Source: [Brooks, 1985, MIT]

Main Robot Architectures

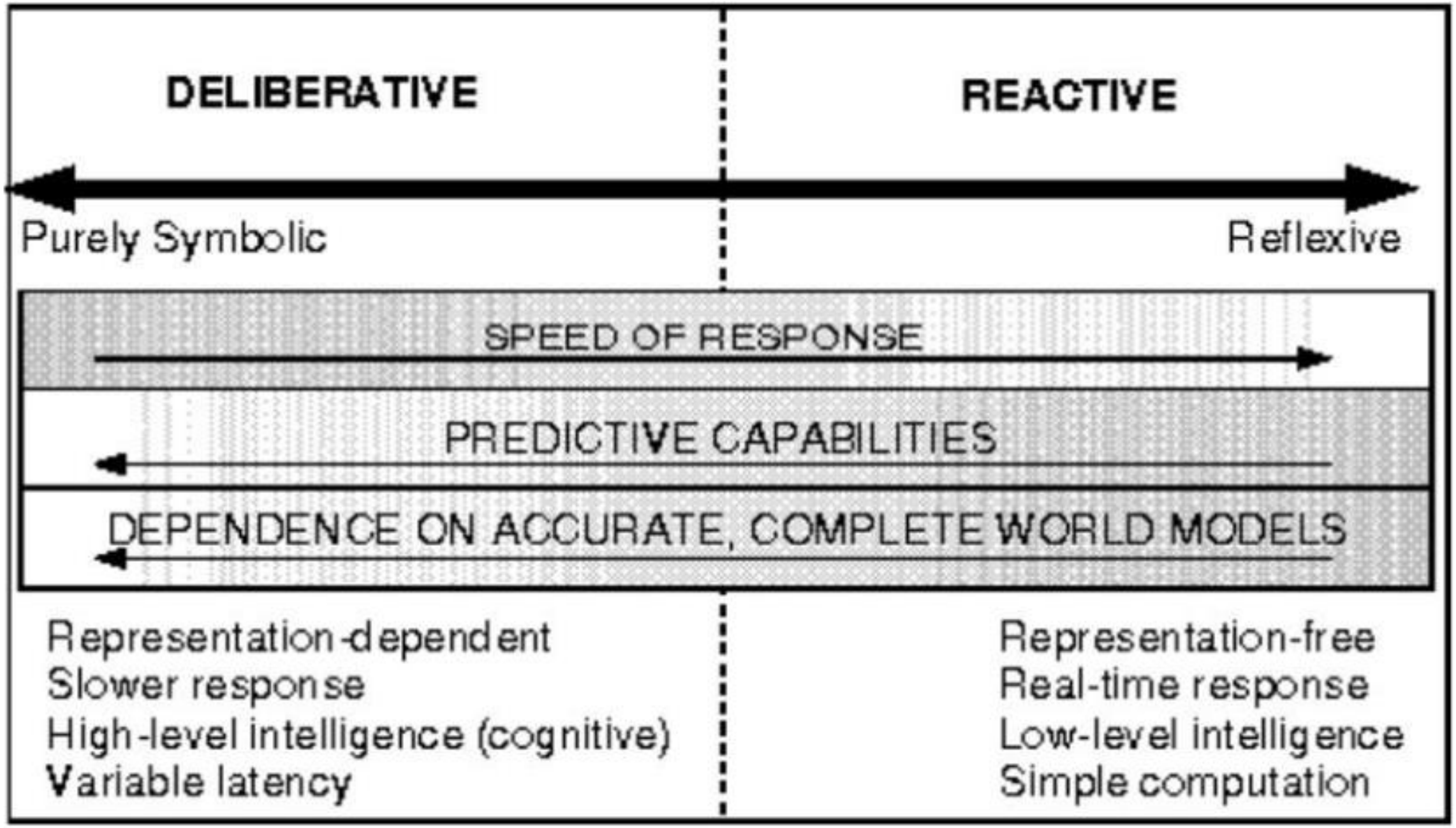
- Reactive/Behavior-based/Subsumption
  - Responsive to dynamic changes





Source: [Brooks, 1985, MIT]

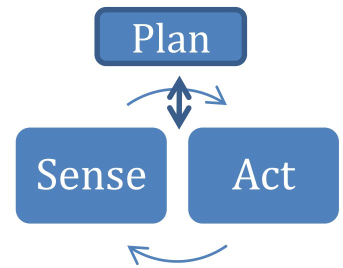
Spectrum of control

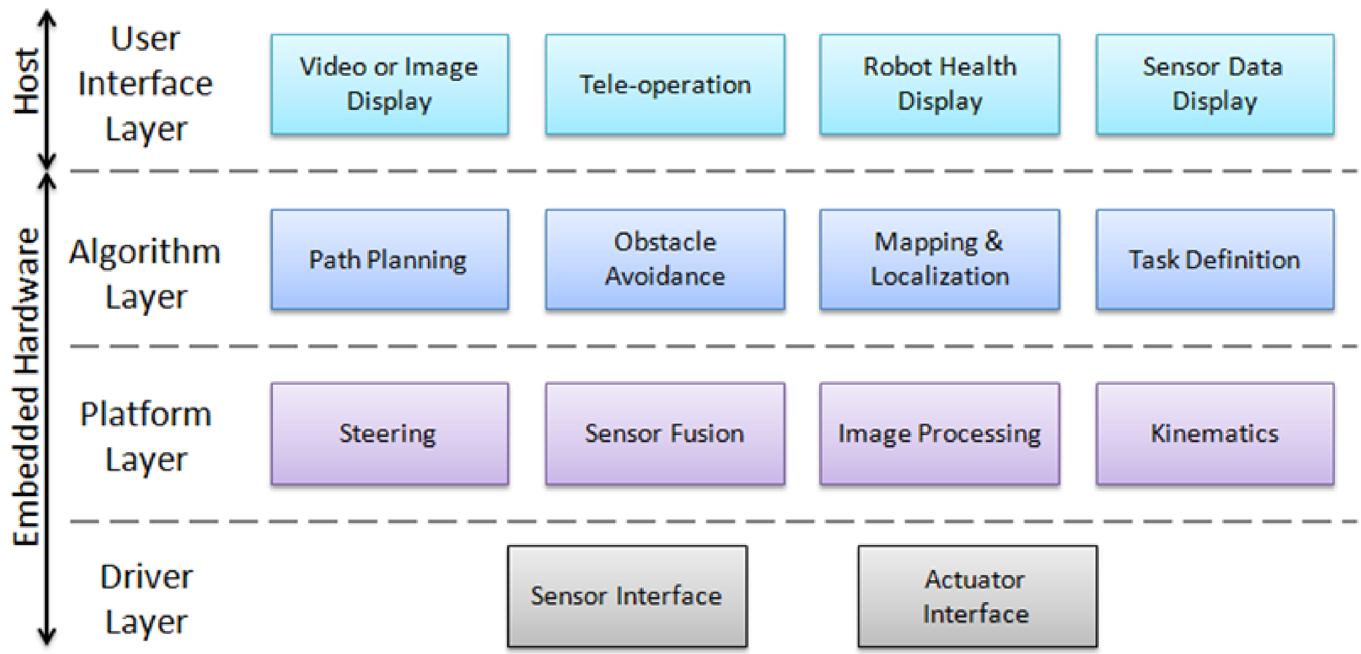


Source: [Arkin, 1998, MIT Press]

Main Robot Architectures

- Layered
  - Integration between reactivity and deliberative



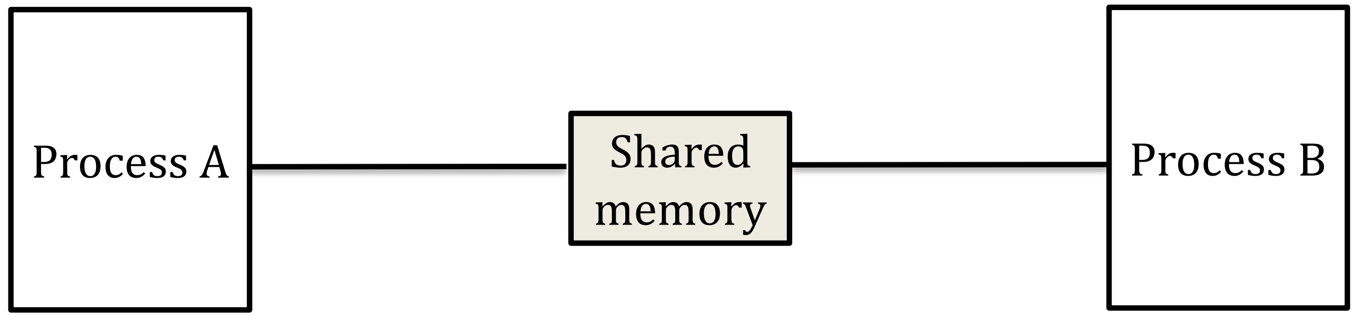


Source: ni.com

Middleware

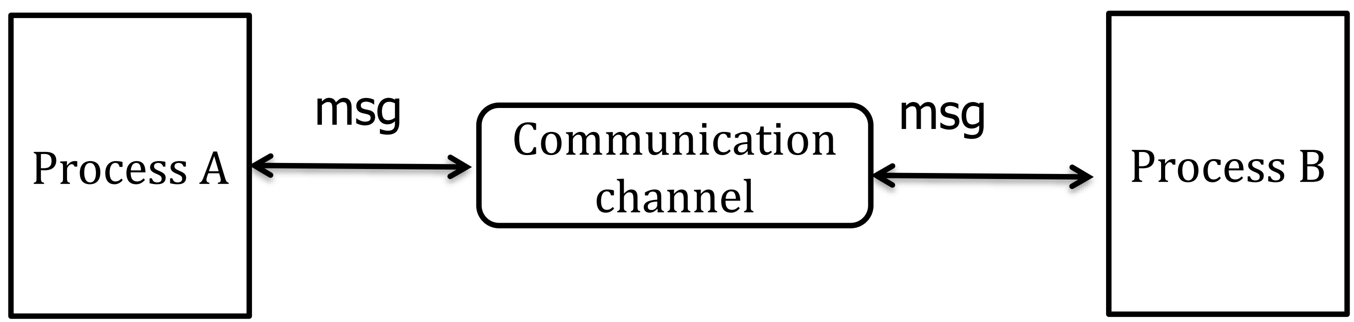
- Components need to share information

Direct



- To make the system modular, a *middleware* can be designed, i.e., the way that components in a robot architecture communicate

Message passing



Basic approaches for message passing

- Client/server: send information as generated by producer (push) or as requested by consumer (pull)
- Publish/Subscribe: consumers request a subscription to a producer and producer sends generated subscribed data to consumers
  - Peer-to-peer: direct connection with producer that sends timestamped data
  - Blackboard-based: middle entity that stores the last instance of data

How to design a robot architecture

- Drawing from software engineering, first all of the requirements and desiderata should be explicitly defined
  - What are the tasks?
  - What actions are necessary to accomplish them?
  - What data is necessary to do the tasks?
  - What capabilities the robot will have?
  - Who are the robot's users?
  - Will the robot architecture used for other tasks/robots?

Robot Architecture Features for Research

- Hardware abstraction
- OS independent
- Open access
- Robustness
- …

Robot Architecture Features for Research

- Modularity
  - Support for multiple components
  - Communication between components
  - Easy way to write own components
  - Possibility to replace individual components
- Support for decentralized components
- …

Robot Architecture Features for Research

- Support for setting at runtime parameters, handled centrally
  - Fixed, through files
  - Dynamically
- Support to log data (timestamped)
- Way to visualize the system and the data
- …

Robot-Dependent Frameworks

- Ndirect, seriald (Nomadics)
- RHeXLib (University of Michigan - Ann Arbor, McGill)
- …

Robot Independent Frameworks

- ROS (Willow Garage)
- MOOS (Paul Newman, Oxford)
- IPC (Reid Simmons — CMU)
- LCM (Albert Huang, Edwin Olson, David Moore — MIT)
- Player (Bryan Gerkey, Richard Vaughan, Andrew Howard — USC)
- OROCOS (Herman Bruyninckx, Peter Soetens, KU Leuven)
- OpenRTM (Japan's National Institute of Advanced Industrial Science and Technology)
- YARP (Italian Institute of Technology)
- Microsoft Robotics Studio
- …

Best practices

- All modules should use
  - The same units (SI units)
  - The same coordinate frames (or provide relations between a common reference frame)

Best practices

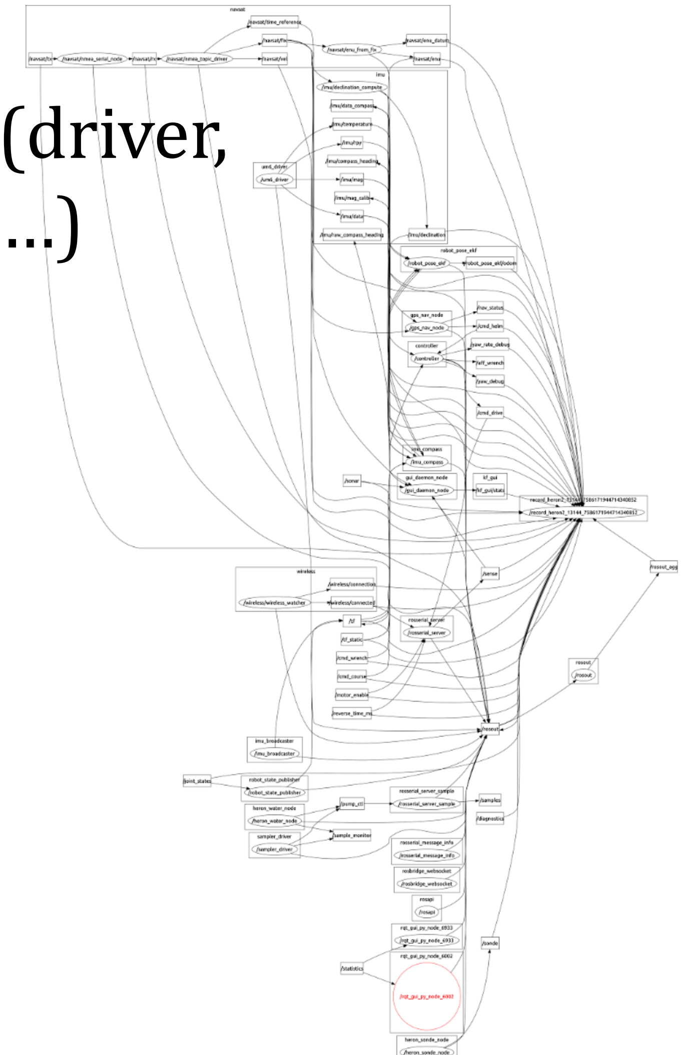
- When writing software given the modularity it is advisable to follow some style guide
- Also properly documenting the code is important
- Unit testing should be performed to ensure no problem with other components

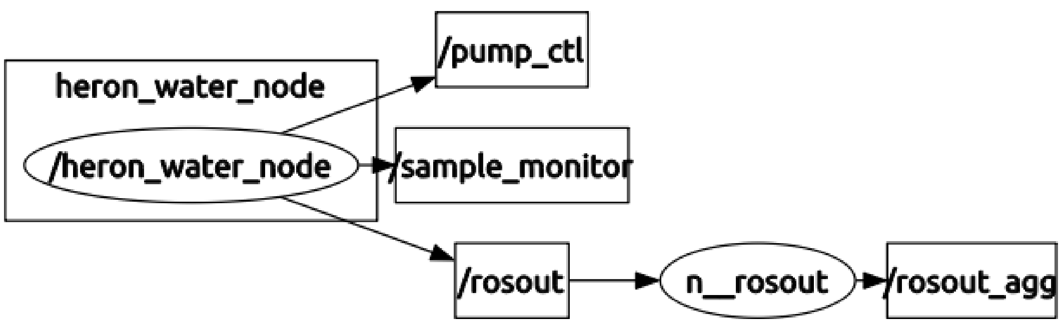
ROS example

- ROS suggests to follow some style guide
  - http://wiki.ros.org/StyleGuide
- For documenting, the code, Doxygen is used
- For unit testing, basically unittest and gtest are used
- Debug for C++ node can be performed through gdb by using `launch-prefix="xterm -e gdb --args“` when running the node

ROS example

- The component used in ROS is a *node*
- Typically, a node represents one task (driver, localization, mapping, path planning, …)
- Nodes run in parallel
- To debug problems, use rqt_graph





ROS example

- The main mean of communication in ROS are topics and messages

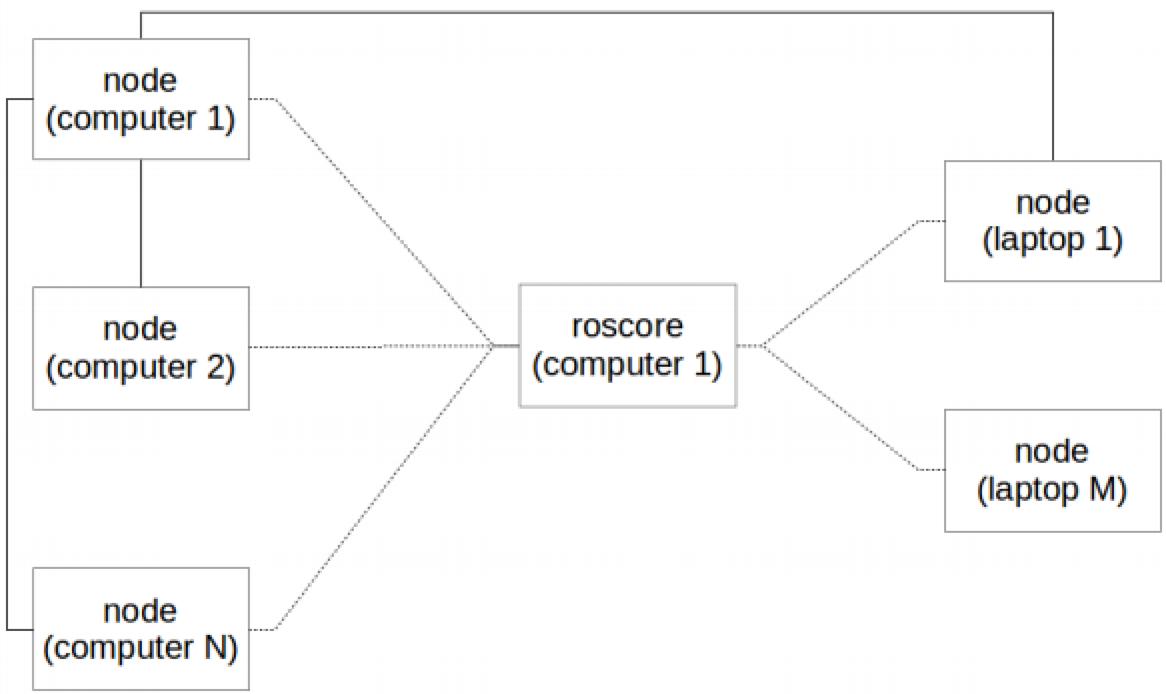
- However, there are other ways for nodes to communicate with each other
  - Services: similar to Remote Procedure Calls
  - Actionlib: preemptable tasks

ROS example

- How to decide what to use:
  - Topics: especially for stream of data
  - Services: execution of fast tasks
  - Actions: execution of tasks that need to be tracked and should be preempted in some cases

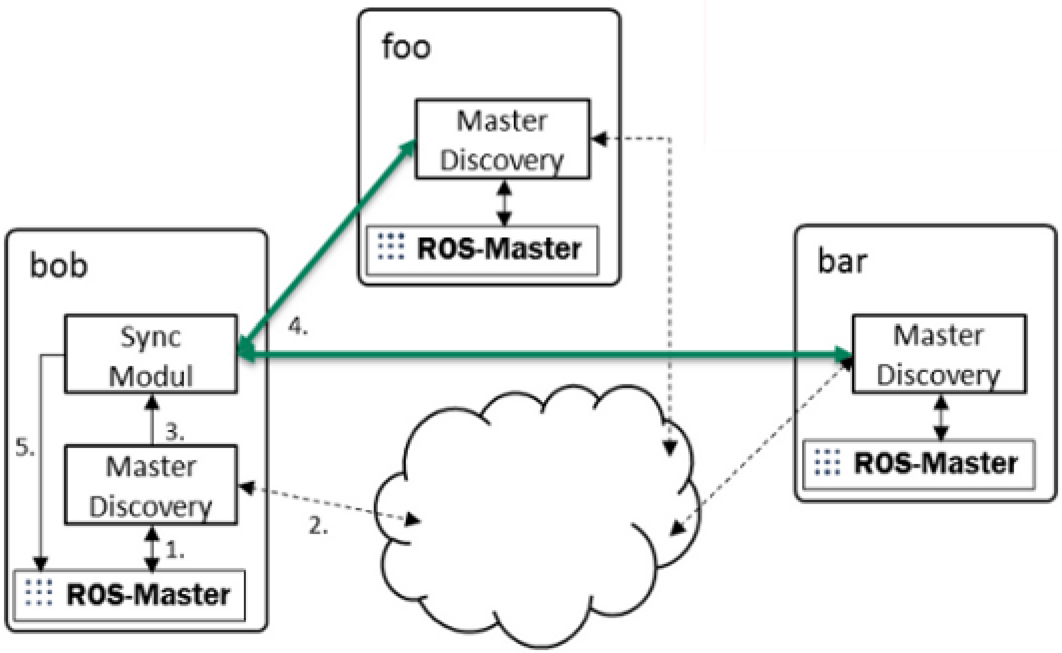
ROS example

- In a multirobot settings, a possibility is to share the ROS master over all of the computers



ROS example

- However, to have the system more robust, *multi-master-fkie* can be used to allow robots to see other ROS masters

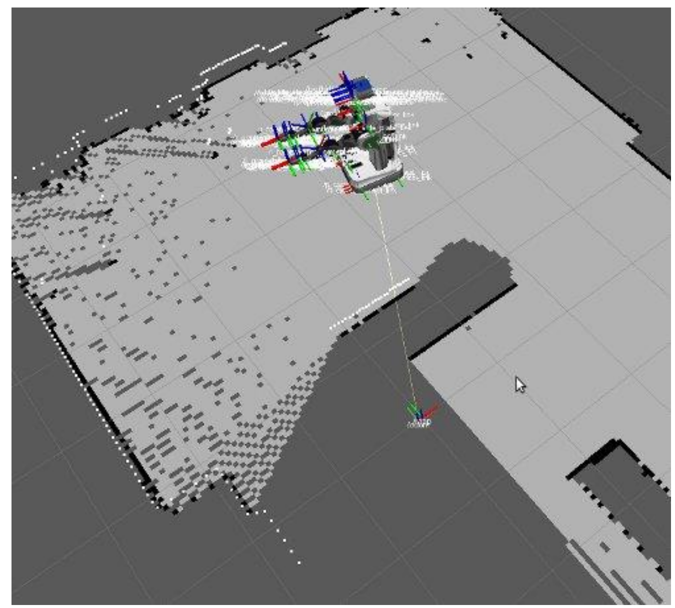


ROS example

- Parameters can be easily set
  - Statically: rosparam
  - Dynamically: rqt_reconfigure
- Be careful in which namespace the parameters are defined: global or private

ROS example

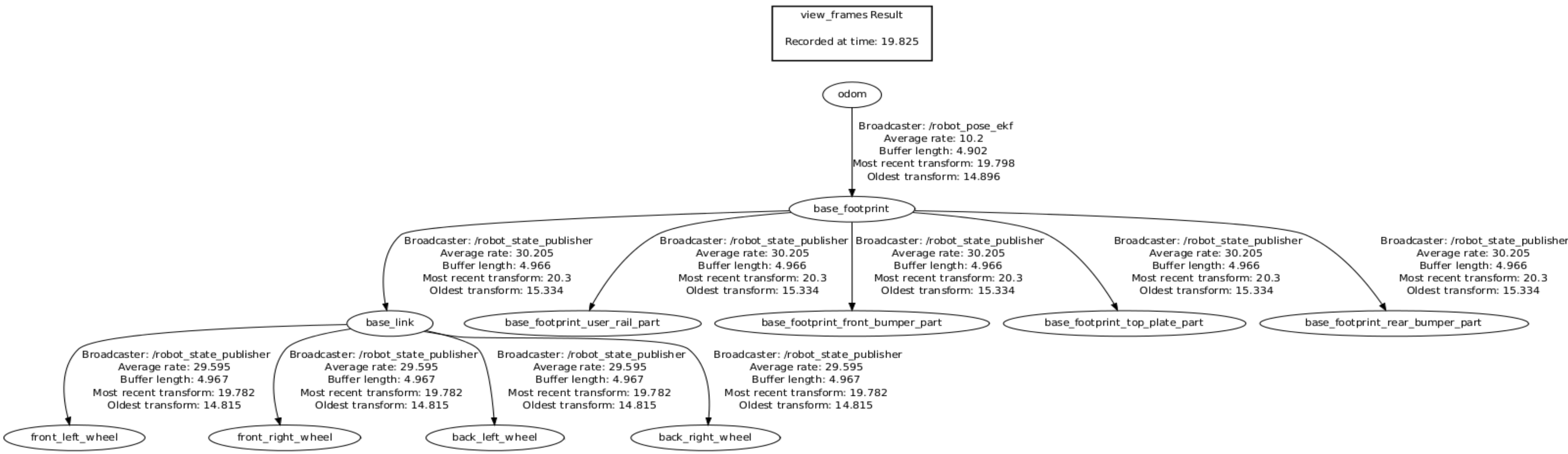
- There are some standards defined in ROS for unit measures and reference frames (REP 103, 105)
- Reference frames are usually stored with *tf*



Source: openrobots.org

ROS example

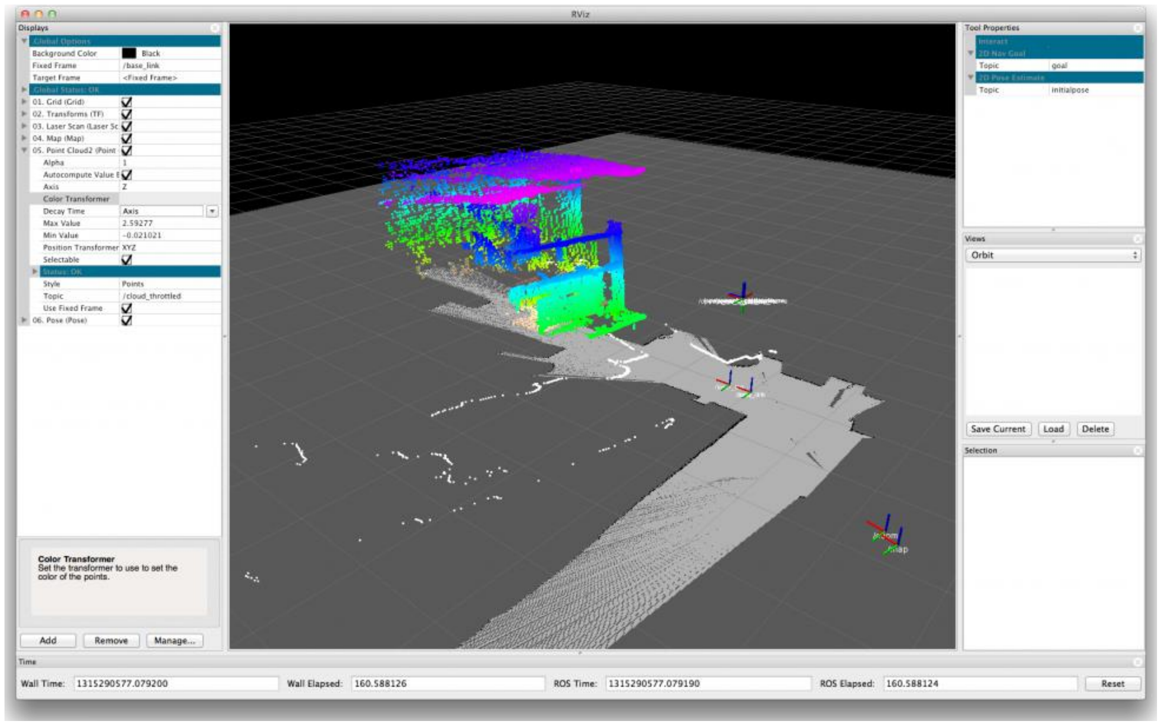
- There are some useful tools to debug tf problems from the tf package (see http://wiki.ros.org/tf/Debugging%20tools)
- e.g., view_frames



Source: clearpathrobotics.com

ROS example

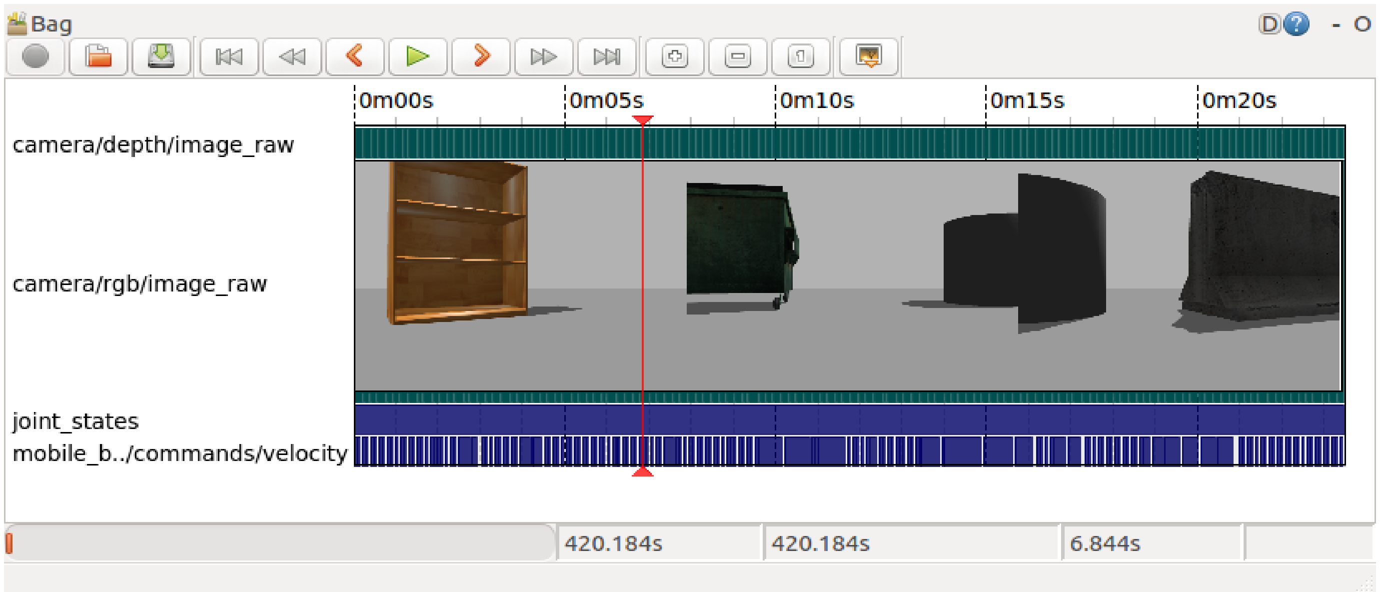
- rviz can be used to visualize data



Source: iheartrobotics.com

ROS example

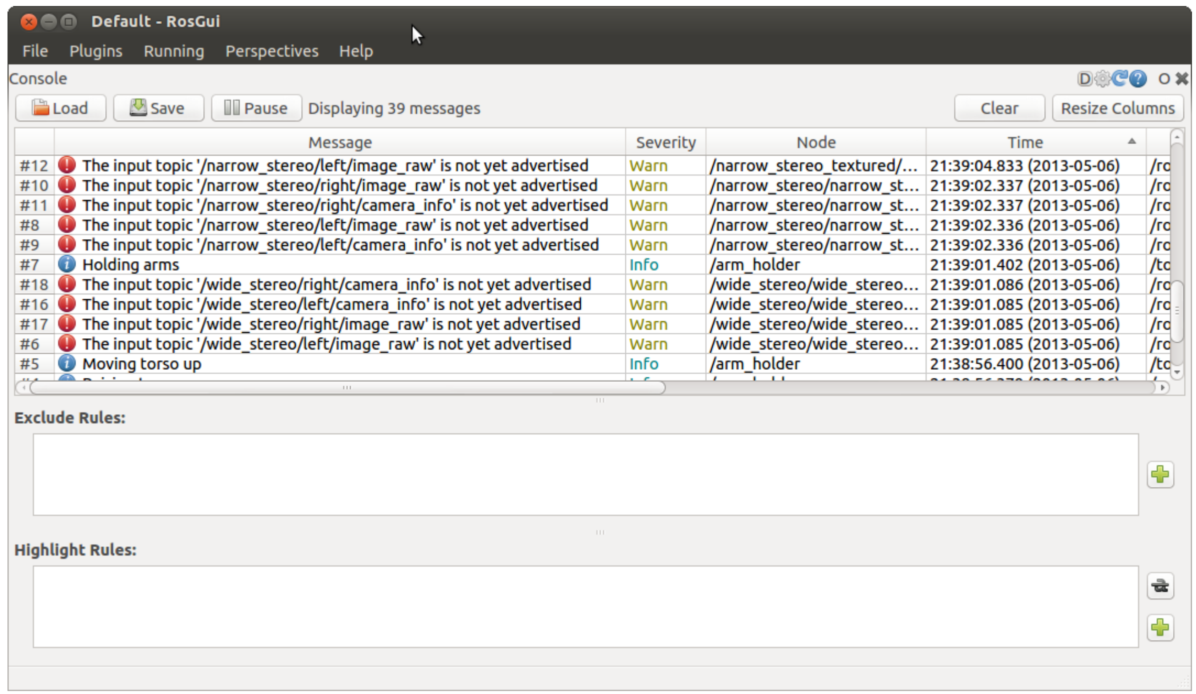
- Logging data streams can be achieved by using rosbag
- Remember the ROS parameter sim_time especially to run algorithms on bag files



Source: ros.org

ROS example

- Logging messages are published in rosout topic
  – Different log levels should be used according to the severity of the message
- rqt_console can be used to visualize them



Source: ros.org

Best practices

- Sound experimental methodologies should be in place, following scientific principles
  - Comparison
  - Reproducibility
  - Repeatability
  - Justification/explanation
- For properly assessing the goodness of an algorithm, the following aspects should be considered:
  - Realism of environments and robot setup
  - Evaluation criteria
  - Sensitivity analysis
  - Statistical analysis (e.g., ANOVA)
  - …

Best practices

- Logging data is important so that
  - No continuous human supervision
  - Collecting training data
  - Some post-processing can be applied
  - Performance evaluation
  - Debugging and reproducing failures
  - …

- The type of data to log depends on the specific task, algorithm being evaluated, …

- Visualization is important especially in robotics given the grounding to the real physical world

Simulations

- Simulators partially model the world and as such will never replace real world experiments
- "Simulations are doomed to succeed"
  - Simulations must be verified

- However, if critically used, simulations are useful because
  - Easy to compare results with ground truth
  - Control the amount of noise
  - Control the time
  - Possibility to execute thousands of runs
  - No hardware problems
  - Ease the debugging process
  - ...

Robotic Simulator

- Gazebo (OSRF)
- Stage (Vaughan – Simon Fraser University)
- UWSim (Prats, Perez, Fernandez, Sanz – Universitat Universitat Universitat Jaume I)
- USARSim (Carpin – UC Merced, Lewis , Wang – U Pittsburgh, Balakirsky, Scrapper – NIST)
- v-rep (Coppelia Robotics)
- RHeX SimSect
- Webots (Cyberbotics)
- MORSE (LAAS-CNRS)
- Nclient, server (Nomadics)
- RD11 (McGill)
- …

Best practices

- Before performing any field experiments, carry out any calibration process needed for the system to work properly
  - e.g., collecting footage for calibrating cameras

Best practices

- For field experiments, it is important to plan missions
  - Where to perform experiments
  - What are the goals for the experiment
  - Estimate time and energy
  - Mission logistics
  - Is there any regulation that must be complied?
  - Plan the data to be logged and collected and the parameters to be set
- Note that before actually going for a field experiment
  - Ensure everything is tested and software is updated and running
  - Batteries are fully charged

Discussion

- Currently no single architecture has proven to be suited for all applications
- Robot architectures should provide
  - Transparent flexible message-based communication network
  - Easy to use and transparent logging and playback capabilities
  - Centralized parameter handling
  - Abstraction of the actual hardware to focus on higher level components