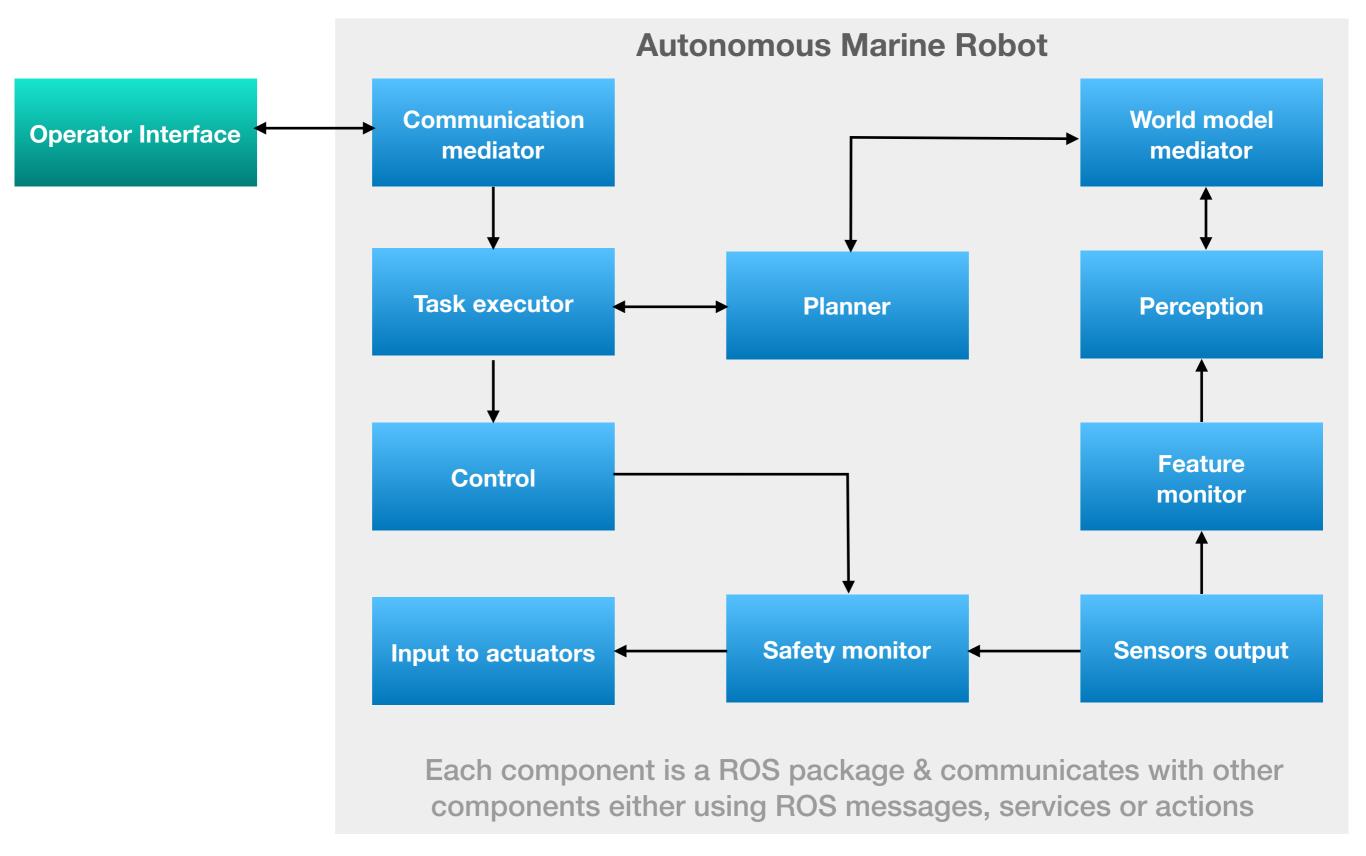
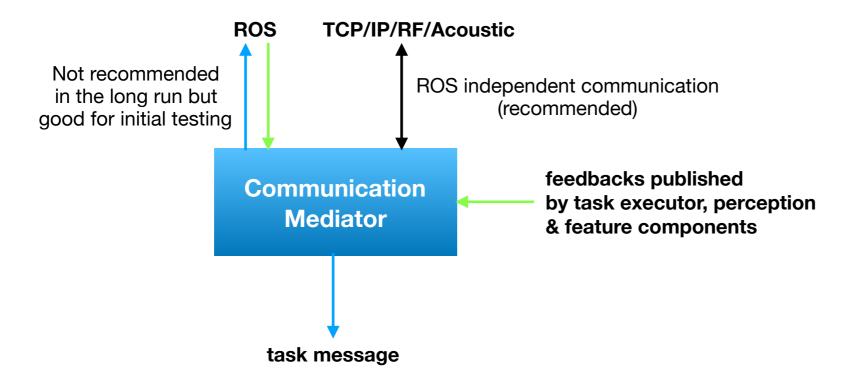
Software component modelling for autonomous marine robots

Software component diagram



Communication mediator

Communication link between operator interface and robot



Code structure

- Adapter module for converting messages from channel specific to ROS format
- Multiple nodes for listening on each channel & publishing a task message

subscriber

publisher

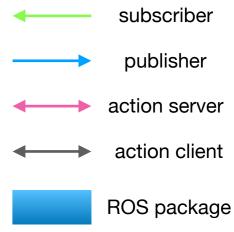
action server

action client

ROS package

Task executor

Responsible for executing different tasks



task (published by task progress communication message mediator) Gets plan for high level control action or Task executor validates plan sent by user by requesting action server provided by planning component. Based on type of planning standard ROS messages should be used eg. nav msgs/Path Mission executor Requests high level control component to perform desired task. Communication should happen using standard ROS messages such as geometry msgs/Pose etc. **Teleoperation**

Publishes commands directly to the different controllers such as velocity, depth, heading etc.
Command should be published over ROS topics using std ROS msgs such

Code structure

as Twist, Wrench etc.

- Bridge module for decoding a task message into missions, high level and low level control actions
- Bridge module for decoding a mission into high level control actions
- Module for high level control action execution (planning, validation & execution)
- Node for mission execution and teleoperation

World model mediator & Planner

World model mediator acts as communication link between different world models (knowledge about the environment)

Planner uses information from world models to create and validate plans

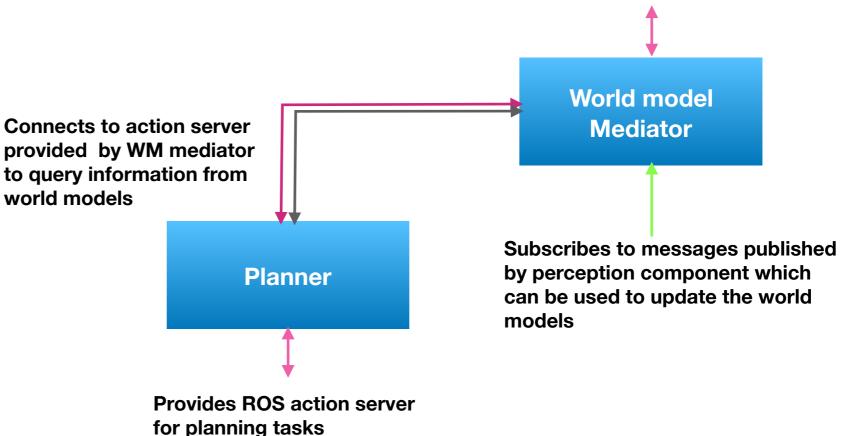
subscriber publisher action server action client ROS package

ROS action server for

querying different world models

Code structure

- Module for each world model (create, update, query)
- Node for querying and updating world models
- Modules for planning
- Node for planning



Sensors, feature monitor & perception

Reads information from sensors and use it for updating the world model

Subscribes to monitored features and use them to draw information about how world model has changed and publish it over ROS topic

Uses action client to request information from world models & associating it with monitored features

subscriber

publisher

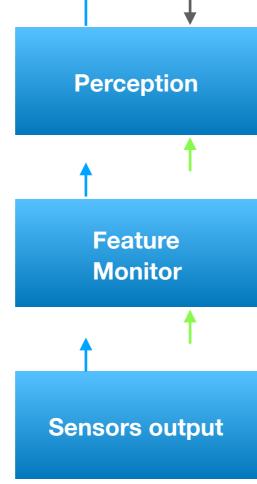
action server

action client

ROS package

Subscribes to sensor messages & publish extracted features

Reads and publishes sensor data using standard sensor messages provided by ROS



Code structure

- Adapter modules for reading and converting each sensor data to ROS
- Nodes for reading & publishing data from each sensor
- Modules for feature monitoring eg. filters
- Node for different feature monitoring
- Modules for perception eg. localization
- Node for perception

Control & actuation

High level control -> Low level control -> Safety -> Actuation

executor

publisher
action server

subscriber

action client

ROS package

High level control such as way-pt following, line following, diving etc.

High level control

Action server to execute

commands sent by task

Low level control

Commands to

control

Listens to commands sent

by task executor or high level

Provides low level controllers such as velocity, heading, depth etc.

Code structure

- Module for high level controls
- Node for executing high level controls
- Node for executing low level controls
- Node for safety monitor
- Nodes for sending commands to each actuators

Safety monitor

actuator eg. thrusters

Checks safety limits before sending commands to actuators. Also subscribes to sensors such as battery & monitors frequency of critical control loops