Intelligent Robot: Integrating JaCaMo and ROS

Robótica Móvel Inteligente

Débora C. Engelmann and Túlio L. Basegio

Source and Documentation

- https://github.com/disaster-robotics-proalertas/pucrs_campus_gazebo/tree/master/src/jason

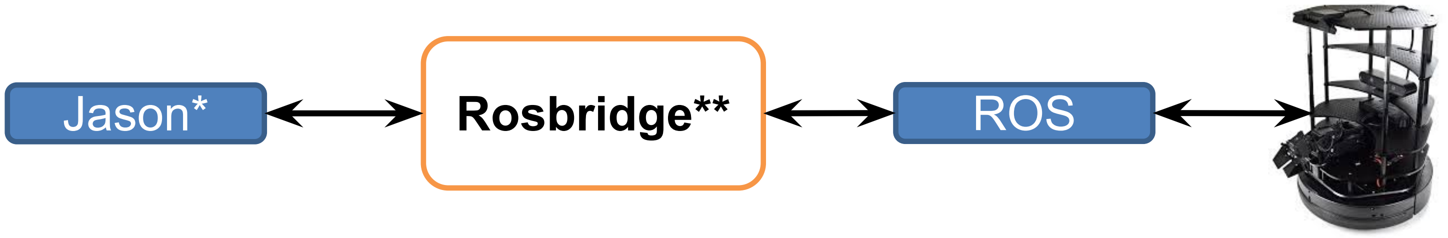
- http://pucrs-campus-on-gazebo.readthedocs.io/en/latest/source/jason/index.html

Outline

Introduction

- Scenario:
  - disaster at PUCRS
  - robots delivering supplies
- Robot intelligence: Jason
- Interface Jason x ROS

Project definition



*http://jason.sourceforge.net/wp/
**http://wiki.ros.org/rosbridge_suite

JaCaMo

Framework for Multi-Agent Programming that combines three separate technologies:

- Jason
- Cartago
- Moise.

To install JaCaMo Eclipse Plugin go to http://jacamo.sourceforge.net/eclipseplugin/tutorial/

Rosbridge suite

- Provides a JSON interface to ROS
  - allows publishing or subscribing to ROS topics by sending a JSON
  - covers also service calls, getting and setting params, and more.

Example:

```
{ "op": "subscribe",
  "topic": "/cmd_vel",
  "type": "geometry_msgs/Twist"
}
```

Rosbridge suite

- **rosbridge_library**
  - responsible for converting the JSON to ROS commands and vice versa.

- **rosapi**
  - provides service calls for getting ROS meta-information
    - list of topics, services, params, etc.

- **rosbridge_server**
  - provides a WebSocket connection/interface to rosbridge.

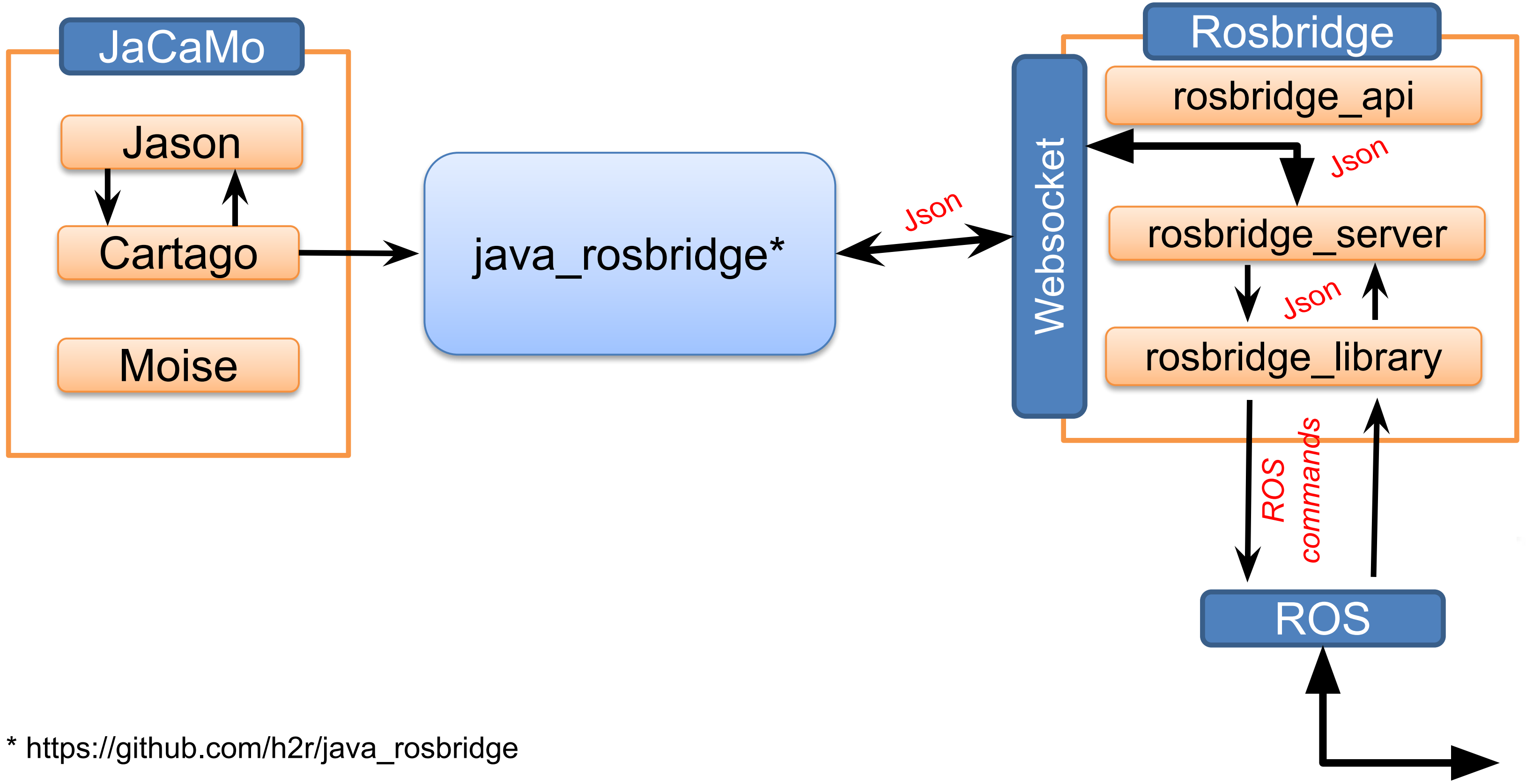
Rosbridge suite

- **Source Code**
  - https://github.com/RobotWebTools/rosbridge_suite
- **Installation**
  - `sudo apt-get install ros-<rosdistro>-rosbridge-server`
- **Running Rosbridge**
  - `source /opt/ros/<rosdistro>/setup.bash`
  - `roslaunch rosbridge_server rosbridge_websocket.launch`
  - websocket server will be created on port 9090 by default

Rosbridge suite

- **Clients**
  - program that communicates with rosbridge using its JSON API.
  - **java_rosbridge** - library using Jetty 9 to connect Java code to a ROS Bridge server.

Project design



* https://github.com/h2r/java_rosbridge

Rosbridge Json vs Jason Beliefs

```
{"topic": "/cmd_vel", "msg": {"linear": {"y": 0.0, "x": 0.5, "z": 0.0}, "angular":
{"y": 0.0, "x": 0.0, "z": 0.0}}, "op": "publish"}
```

```
cmd_vel(linear,0.0, 0.5, 0.0)
cmd_vel(angular,0.0, 0.0, 0.0)

vel_linear(0.0, 0.5, 0.0)
vel_angular(0.0, 0.0, 0.0)

cmd_vel(linear,0.0, 0.5, 0.0,angular,0.0, 0.0, 0.0)
```

Implementation

RosBridge.java

A socket for connecting to ros bridge that accepts subscribe and publish commands.

Connect to rosbridge:

```
connect(String rosBridgeURI, boolean waitForConnection)
rosBridgeURI = ws://localhost:9090
```

Subscribing

```
subscribe(String topic, String type, RosListenDelegate delegate, int
throttleRate, int queueLength)
```

Publishing is also supported with the

```
public void publish(String topic, String type, Object msg)
```

Set the interface between Jason / Ros

```
setRosInterface(RosInterface rosInterface)
```

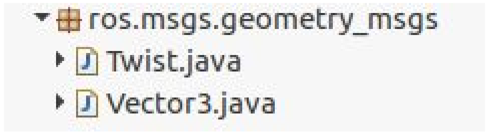
Implementation

RosInterface.java

```
AbstractQueue<Collection<Percept>>  perceptsQueue  = new
ConcurrentLinkedQueue<Collection<Percept>>();

private void setup(){
     rosbridge = new RosBridge();
     rosbridge.setRosInterface(this);
}

public void run() {
while (!rosbridge.hasConnected()) {
     rosbridge.connect("ws://localhost:9090",true);
}
this.subscribre();
}
```

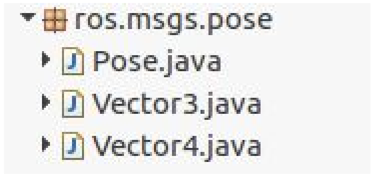
Implementation

```
private void subscribre() {

rosbridge.subscribe(SubscriptionRequestMsg. generate("/cmd_vel_mux/input/teleop" )
      .setType("geometry_msgs/Twist" )
      .setThrottleRate(1)
      .setQueueLength(1),
      new RosListenDelegate() {
            public void receive(JsonNode data, String stringRep) {
                  MessageUnpacker<Twist> unpacker = new MessageUnpacker<Twist>(Twist. class);
                  Twist msg = unpacker.unpackRosMessage( data);
                  twistPercepts.clear();
                  twistPercepts.add(new Percept("twist_linear", new Numeral(msg.linear.x),
                                                                          new  Numeral(msg.linear.y),new
Numeral(msg.linear.z)));
                  twistPercepts.add(new Percept("twist_angular", new Numeral(msg.angular.x),
                                                                          new Numeral(msg.angular.y),new
Numeral(msg.angular.z)));
                  perceptsQueue.add(Collections. synchronizedSet(new
HashSet<Percept>( twistPercepts)));
                  }
            }
      );
```



Implementation

```
rosbridge.subscribe(SubscriptionRequestMsg. generate("/odom")
    .setType("nav msgs/Odometry")
    .setThrottleRate(1)
    .setQueueLength(1),
    new RosListenDelegate() {
        public void receive(JsonNode data, String stringRep) {
            JsonNode msg = data.path("msg");
                        JsonNode pose1 = msg.path("pose");
            JsonNode pose2 = pose1.path("pose");
            MessageUnpacker<Pose> unpacker = new
MessageUnpacker<Pose>(Pose. class);
            Pose msgPose = unpacker.unpackRosMessage(pose2);
            currentPose =msgPose;
            }
        })
```


ros.msgs.pose
Pose.java
Vector3.java
Vector4.java

Implementation

```
public static void publishing(
            Publisher pubp,double lx,double ly,double lz,double ax,double ay,double az,int
      sleepTime) {

            Twist t1 = new Twist();
            t1.linear.x=lx;
            t1.linear.y=ly;
            t1.linear.z=lz;
            t1.angular.x=ax;
            t1.angular.y=ay;
            t1.angular.z=az;
            pubp.publish(t1);

            try {
                  Thread.sleep(sleepTime);
            } catch (InterruptedException e) {
                  e.printStackTrace();
            }
      }
Publisher pub = new Publisher("/cmd_vel_mux/input/teleop", "geometry_msgs/Twist",
rosbridge);
```

Implementation

```
public void move2goal(double goalx,double goaly,double goalz) {

     while (Math.sqrt(Math.pow((goalX - currentPose.position.x), 2) + Math.pow((goalY -
currentPose.position.y), 2)) >= distance_tolerance) {

     linearX = 1.5 * Math.sqrt(Math.pow((goalX - currentPose.position.x), 2) +
     Math.pow((goalY - currentPose.position.y), 2));
     linearY = 0;
     linearZ = 0;

     angularX = 0;
     angularY = 0;
     angularZ = 4 * (Math.atan2(goalY - currentPose.position.y, goalX -
     currentPose.position.x) - currentPose.orientation.z);

     Publisher pub = new Publisher("/cmd_vel_mux/input/teleop", "geometry_msgs/Twist",
rosbridge);
     publishing(pub,linearX,linearY,linearZ,angularX,angularY,angularZ,500);
     }
}
```

Implementation

```
protected void init(String config) throws IOException, InterruptedException 
{

    ri = new RosInterface(config);
    try {
        ri.start();
    } catch (Exception e) {
        e.printStackTrace();
    }

    ri.attachAgentListener( this);

    receiving = true;
    execInternalOp("receiving", "ag1");
}
```

Implementation

```
@INTERNAL OPERATION
  void receiving(String agent) throws JasonException {
     lastStep = -1;
     Collection<Percept> previousPercepts = new ArrayList<Percept>();
     while(!ri.isEntityConnected(agent))
          await time(100);
     while (receiving) {
          await time(500);
          if (ri != null) {
               try {
                    Collection<Percept> percepts = ri.getNextPerception(agent,agent);
                    if (!percepts.isEmpty()) {
                         updatePerception(previousPercepts, percepts);
                         previousPercepts = percepts;
                    }
               } catch (Exception e) {
                     e.printStackTrace();
               }
          }
     }
}
```

Implementation

```java
private void updatePerception(Collection<Percept> previousPercepts,
Collection<Percept> percepts) throws JasonException {
    for (Percept old: previousPercepts) {
        if (previousList.contains(old.getName())) {
                Literal literal = Translator.perceptToLiteral(old);
                previousList.remove(old.getName());
                removeObsPropertyByTemplate(old.getName(), (Object[])
literal.getTermsArray());
            }
        }

for (Percept percept: percepts) {
        Literal literal = Translator.perceptToLiteral(percept);
        previousList.add(percept.getName());
        defineObsProperty(percept.getName(), (Object[])
literal.getTermsArray());
    }
}
```

Simulation

Limitations

- move to goal
- tests with Jason plans (Vinicius)
- using more than one robot

Intelligent Robot: Integrating JaCaMo and ROS

Robótica Móvel Inteligente

Débora C. Engelmann and Túlio L. Basegio

FACIN PUCRS

PUCRS
Pontifícia Universidade Católica do Rio Grande do Sul