ROBOTIC MIDDLEWARES

ROBOTICS

POLITECNICO
MILANO 1863

MIDDLEWARE ORIGINS

The origins

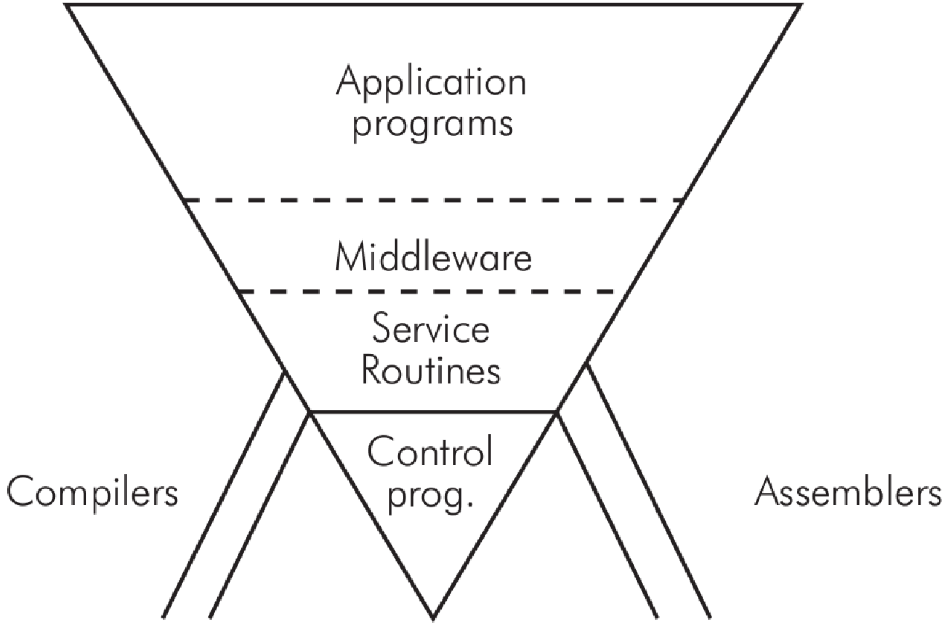
- 1968 introduced by d'Agapeyeff
- 80's wrapper between legacy systems and new applications
- Nowadays: widespread in different domain fields (including Robotics)

The Middleware idea

- Well-known in software engineering
- It provides a computational layer
- A bridge between the application and the low-level details
- It is not a set of API and library

Issues in developing real robots

- Cooperation between hardware and software
- Architectural differences in robotics systems
- Software reusability and modularity



WHAT IS A MIDDLEWARE?

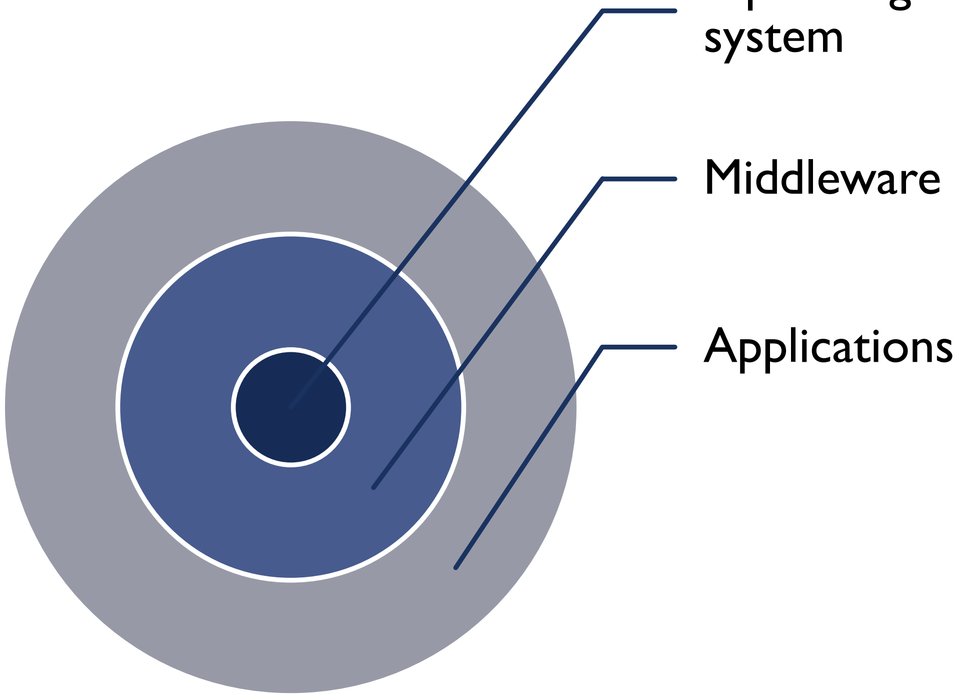
Software that ***connects*** different software components or applications:

- Set of services that permits to several processes to interact
- Framework used to reduce the developing time in complex systems.

Middleware vs. Operating System

- The middleware stays between software and different operating systems.
- The distinction between operating system and middleware is sometimes arbitrary.
- Some features of a middleware are now integrated in operating systems (e.g., TCP/IP stack).

<u>Some (non robotics) examples</u>: Android, SOAP, Web Services, …



MIDDLEWARES MAIN FEATURES

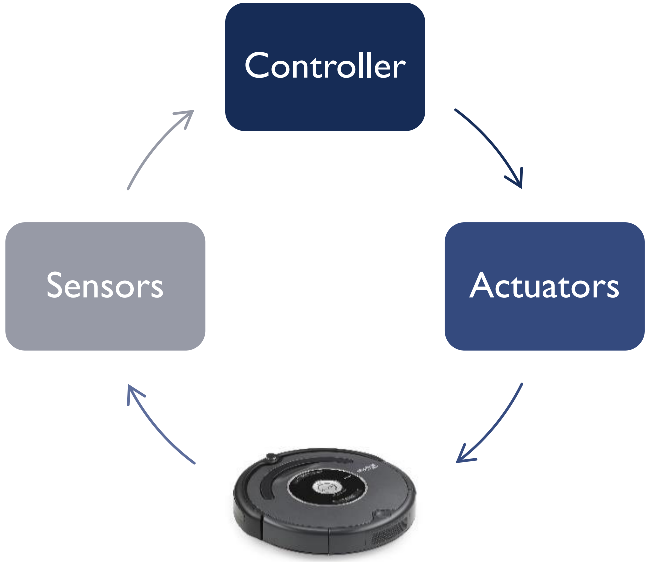
Portability: provides a common programming model regardless the programming language and the system architecture.

Reliability: middleware are tested independently. They permit to develop robot controllers without considering the low level details and using robust libraries.

Manage the complexity: low-level aspects are handled by libraries and drivers inside the middleware. It (should) reduce(s) the programming error and decrease the development time.

ROBOT DEVELOPMENT

1. Modelling
   - Kinematic model
   - Differential kinematics
   - Dynamic model
2. Planning
   - Motion laws
   - Trajectory generation
3. Control
   - Translate the movement into motor commands
   - Several type of control: motion, force, etc.



Before the introduction of middleware

- Monolithic approach
- Little if any reuse of models or components
- Hard to maintain code and hard to integrate components

Some people believe the real issue with Robotics is **integration**!

ROBOT MIDDLEWARES: A LIST

Several middleware have been developed in recent years:

OROCOS	[Europe]
ORCA	[Europe]
YARP	[Europe / Italy]
BRICS	[Europe]
OpenRTM	[Japan]
OpenRave	[US]
ROS	[US]

…

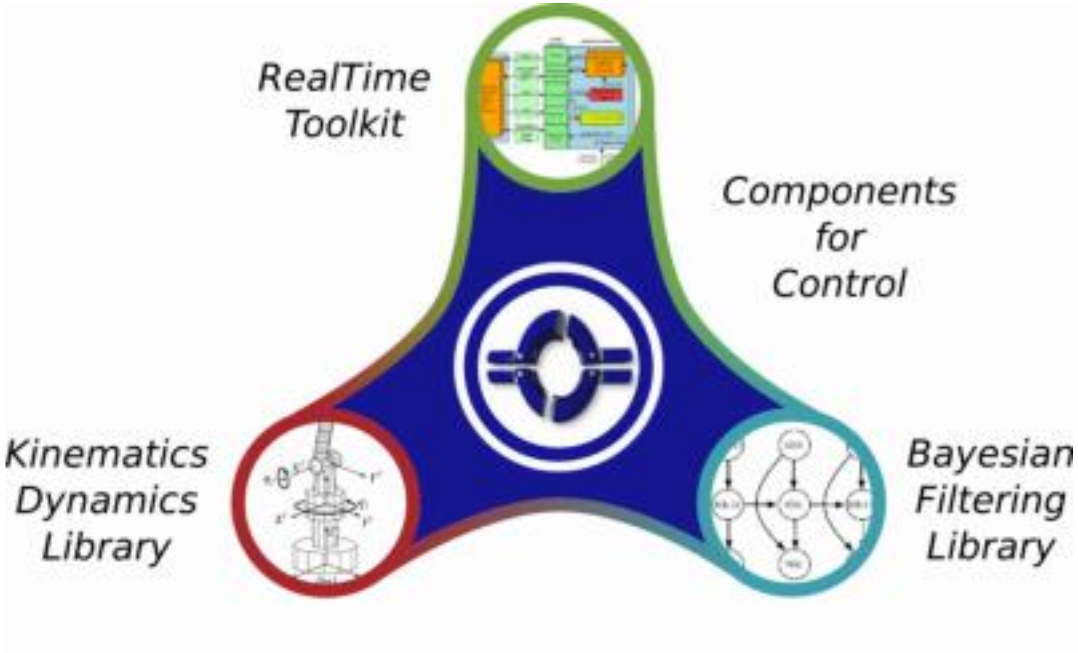
Let’s see their common features and main differences

OROCOS: OPEN ROBOT CONTROL SOFTWARE

The project started in December 2000 from an initiative of the mailing list EURON then it become an European project with 3 partners: K.U. Leuven (Belgium), LAAS Toulouse (France), KTH Stockholm (Sweden)

OROCOS requirements:

- Open source license
- Modularity and flexibility
- Not related to robot industries
- Working with any kind of device
- Software components for kinematics, dynamics, planning, sensors, controller
- Not related to a unique programming language



OROCOS STRUCTURE

OROCOS Real-Time Toolkit (RTT)

- infrastructure and functionalities for real-time robot systems
- component-based applications

OROCOS Component Library (OCL)

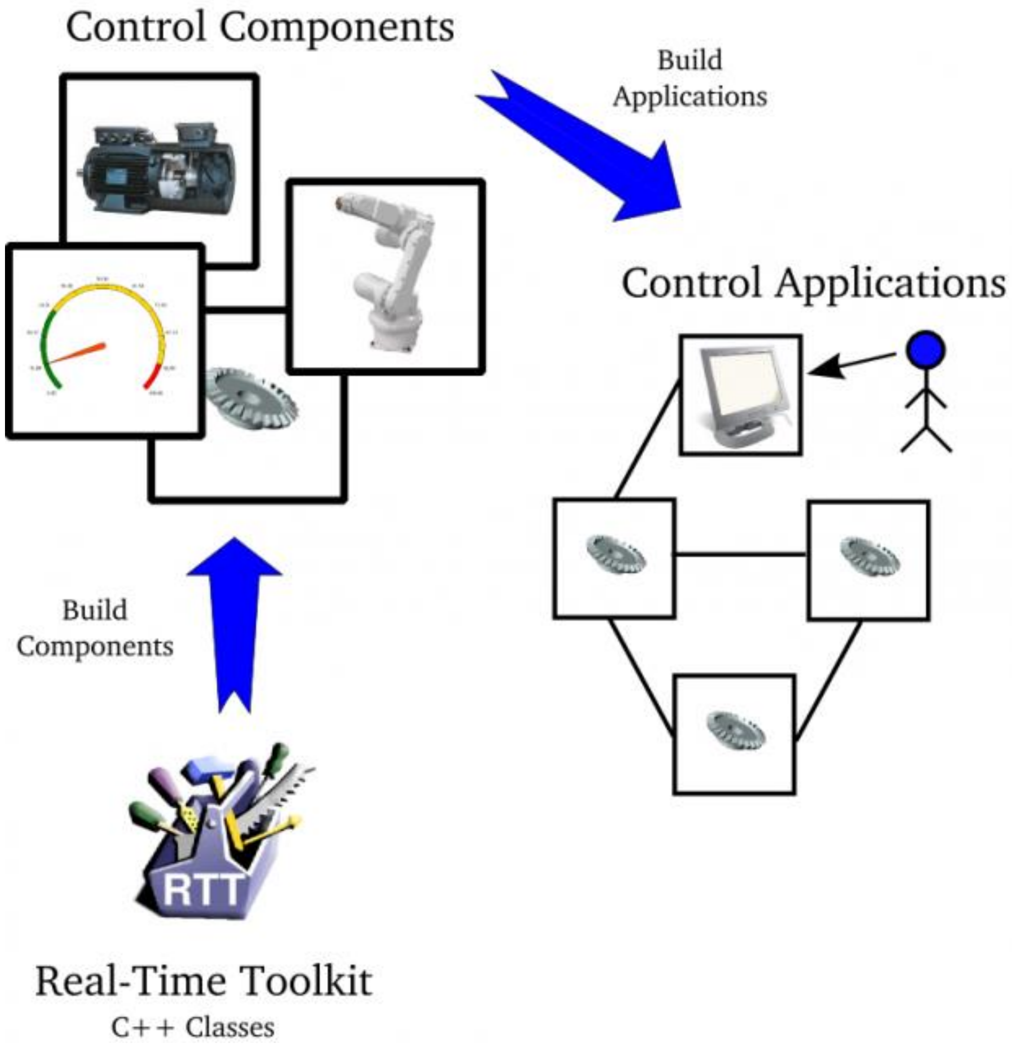
- provides ready-to-use components, e.g., device drivers, debugging tools, path planners, task planners

OROCOS Bayesian Filtering Library (BFL)

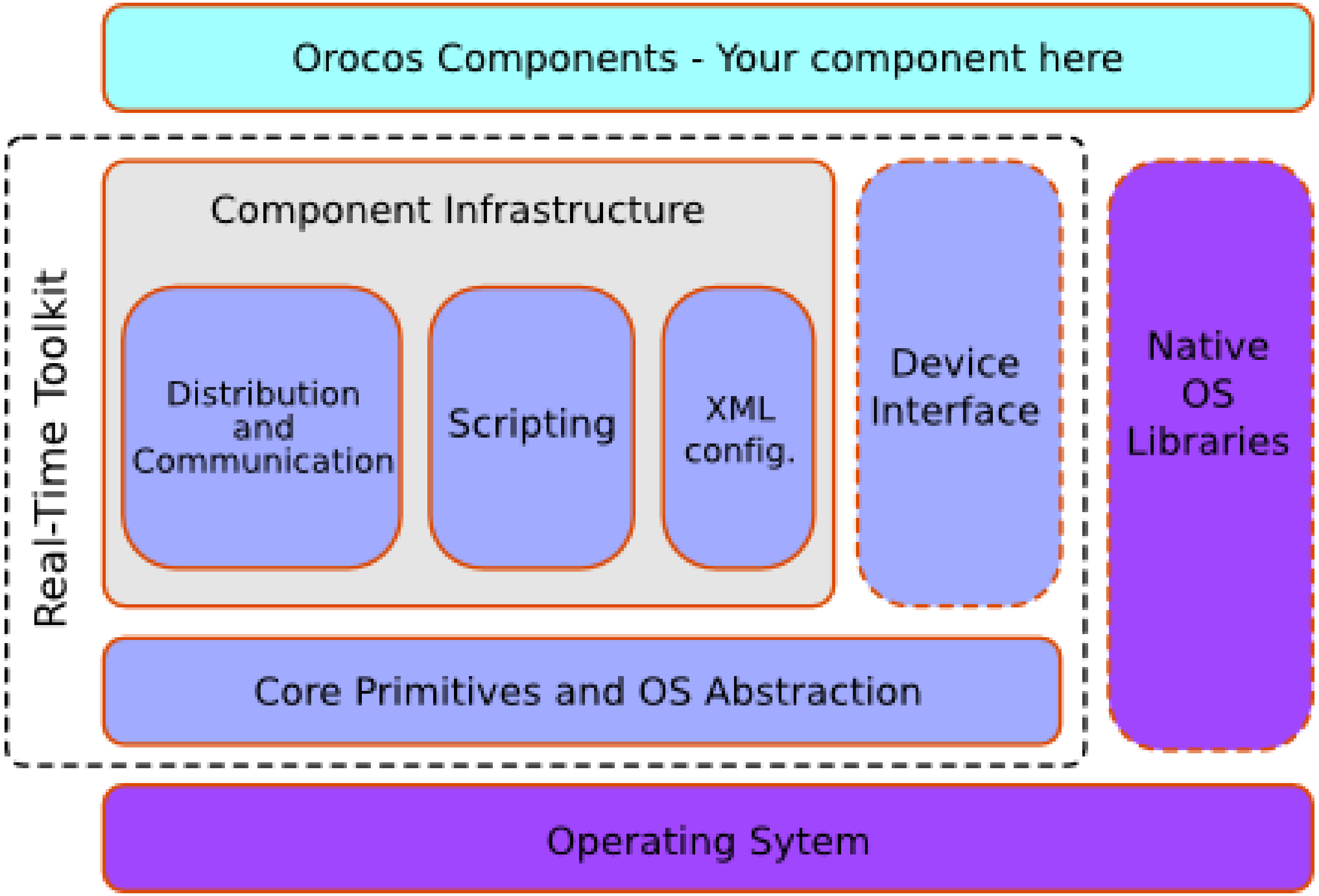
- application independent framework, e.g., (Extended) Kalman Filter, Particle Filter

OROCOS Kinematics & Dynamics Library (KDL)

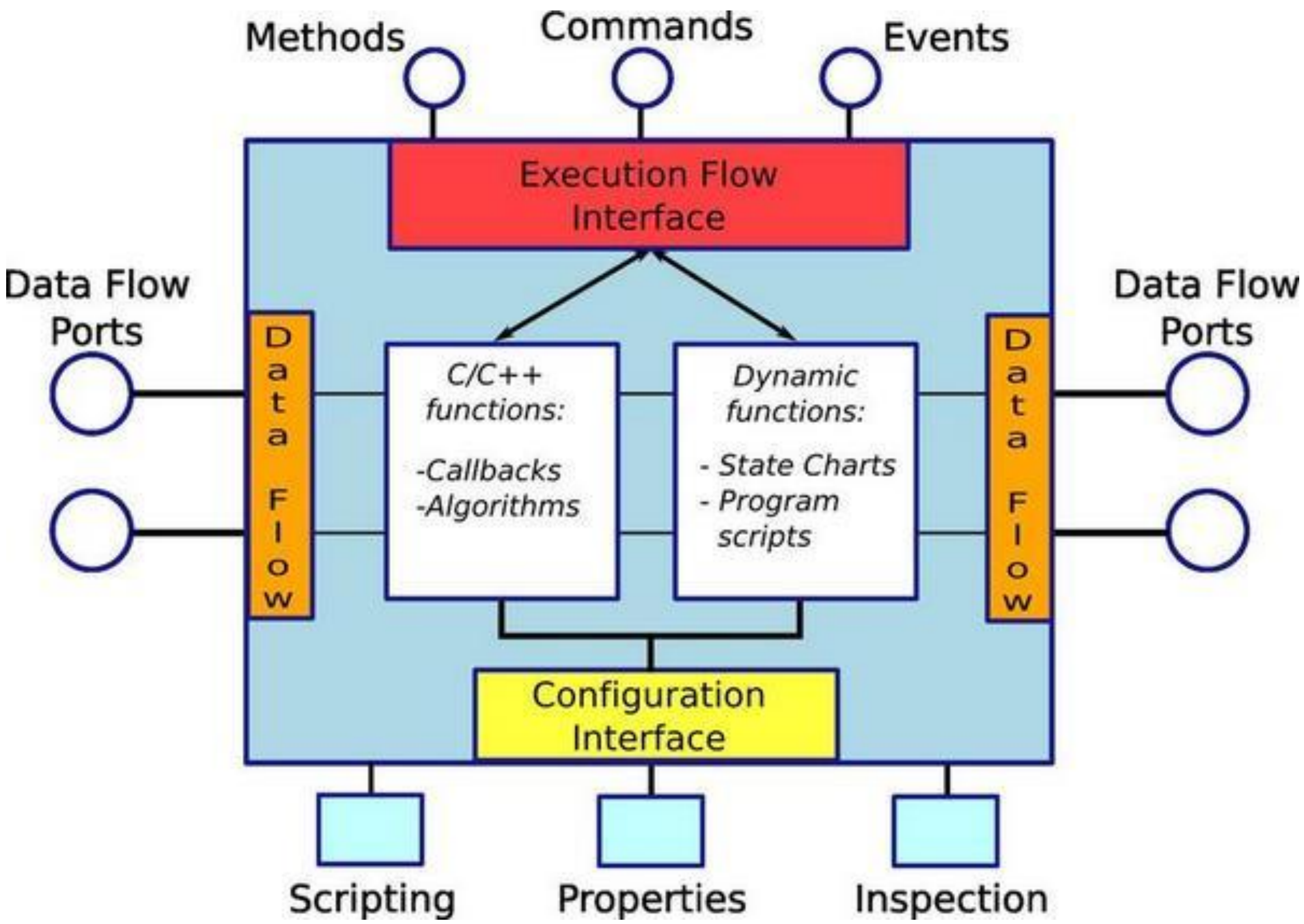
- real-time kinematics & dynamics computations



OROCOS RTT FRAMEWORK



OROCOS COMPONENT



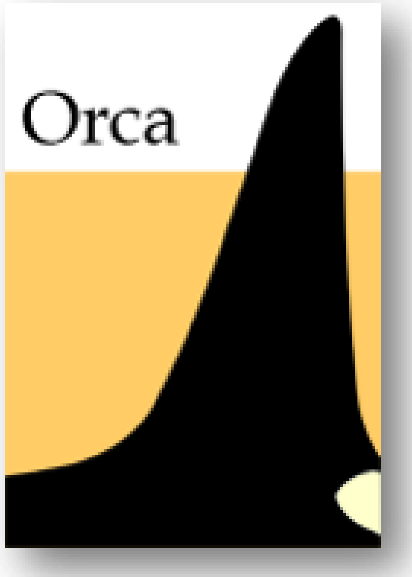
ORCA: COMPONENTS FOR ROBOTICS

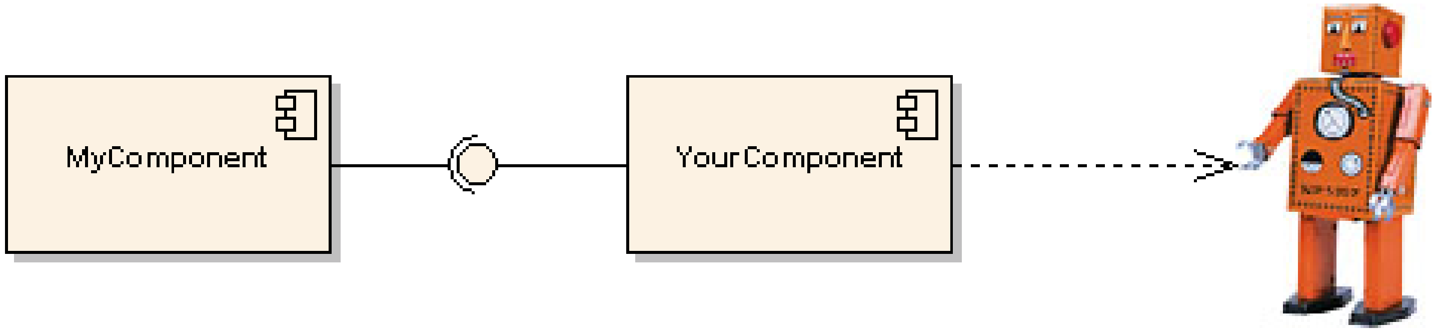
The aim of the project is to focus on software reuse for scientific and industrial applications

Key properties:

- commonly-use interfaces
- high-level libraries
- updated software repositories

ORCA defines itself as “unconstrained component-based system”





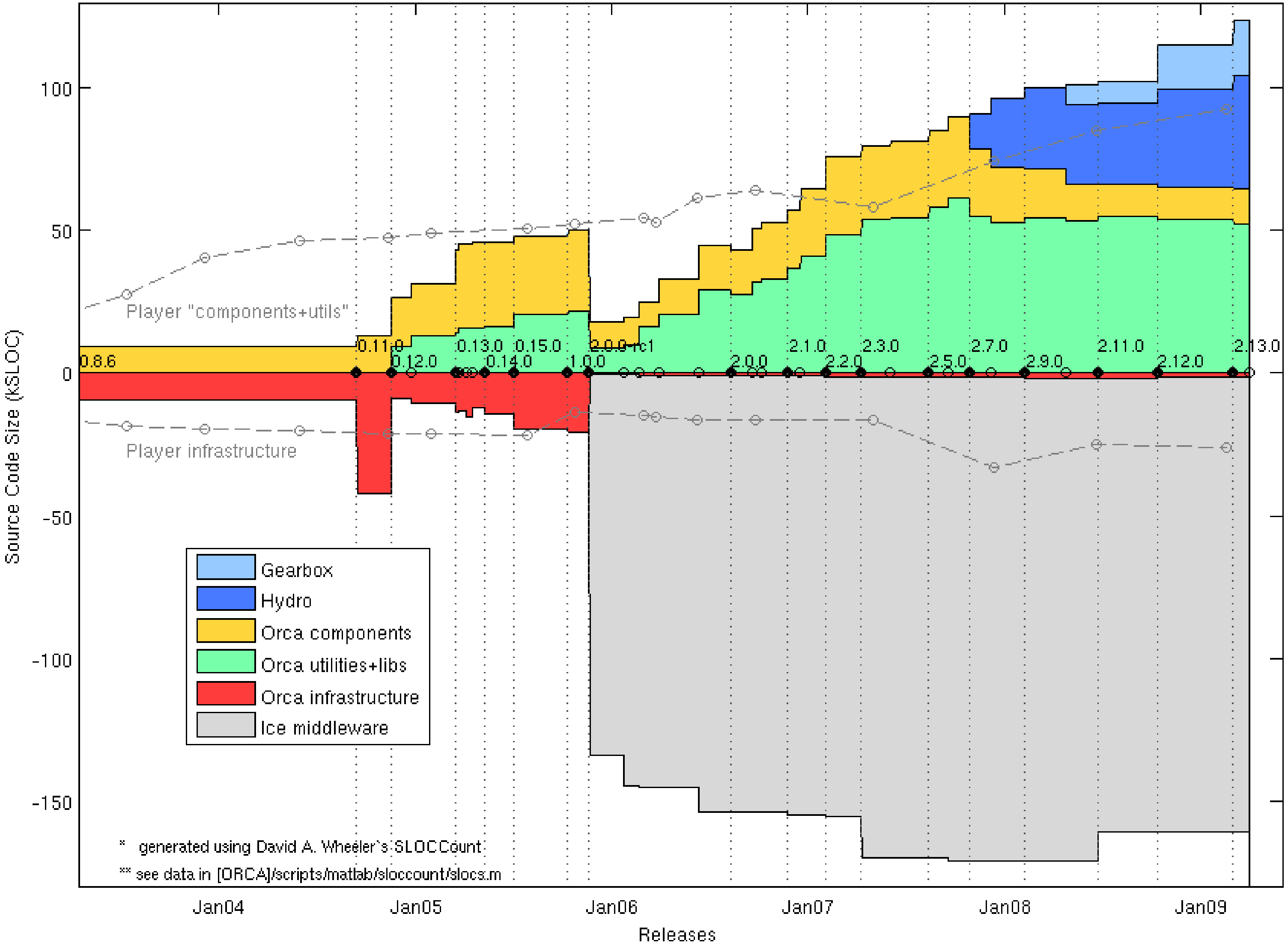
ORCA AND ICE

The main difference between OROCOS and ORCA is the communication toolkit; OROCOS uses CORBA while ORCA uses ICE

- ICE is a modern framework developed by ZeroC
- ICE is an open-source commercial communication system
- ICE provides two core services
  - IceGrid registry (Naming service): which provides the logic mapping between different components
  - IceStorm service (Event service): which constitute the publisher and subscriber architecture

“A component can find the other components through the IceGrid registry and can communicate with them through the IceStorm service.”

ORCA: LIBRARY EVOLUTION



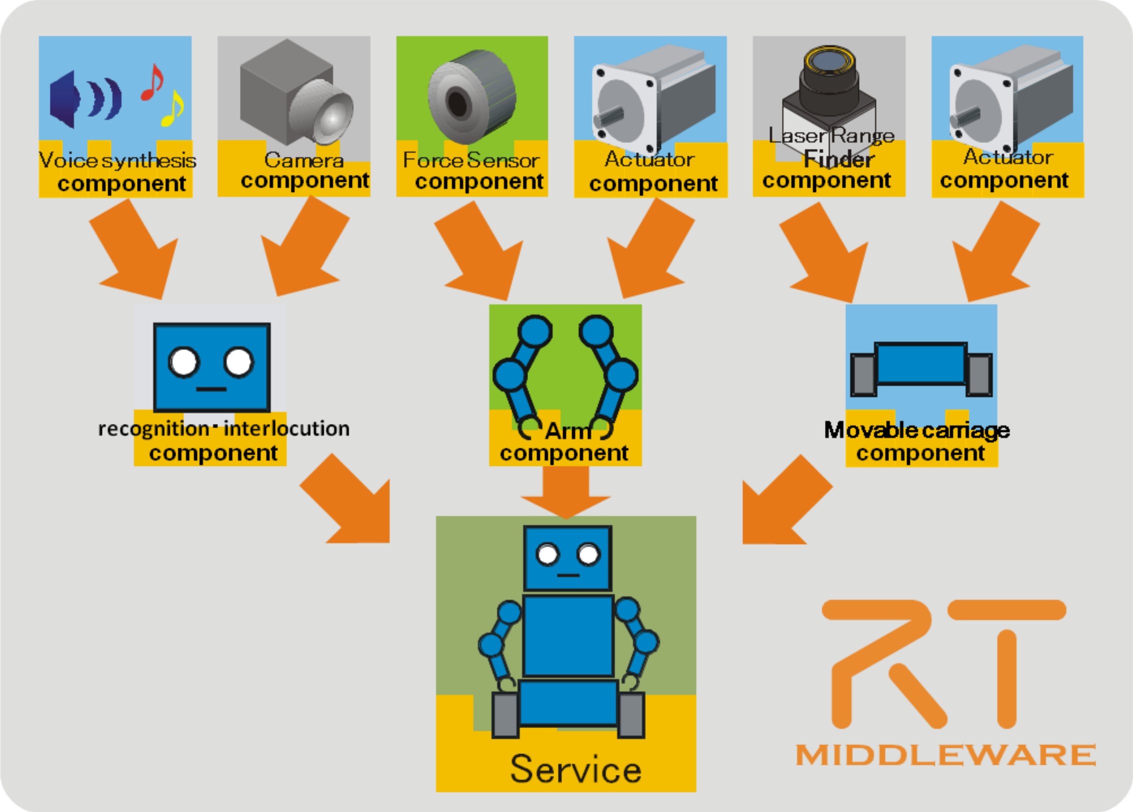
* generated using David A. Wheeler`s SLOCCount

** see data in [ORCA]/scripts/matlab/sloccount/slocs.m

RT MIDDLEWARE

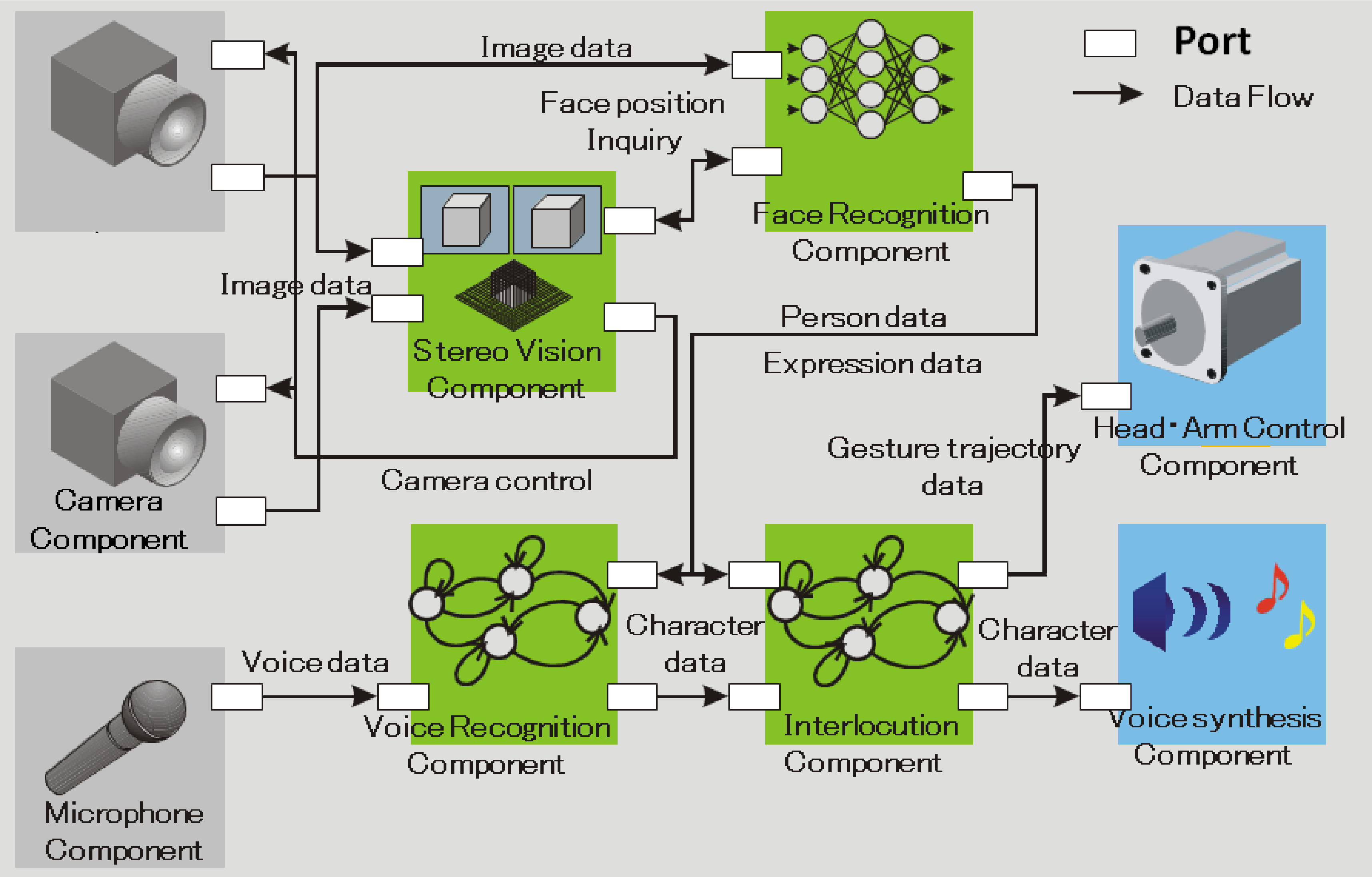
RT-Middleware (RTM) is a common platform standard to construct the robot system by combining the software modules of the robot functional elements (RTC):

- Camera component
- Stereovision component
- Face recognition component
- Microphone component
- Speech recognition component
- Conversational component
- Head and arm component
- Speech synthesis component


Voice synthesis component
Camera component
Force Sensor component
Actuator component
Laser Range Finder component
Actuator component
recognition・interlocution component
Arm component
Movable carriage component
Service
RT
MIDDLEWARE

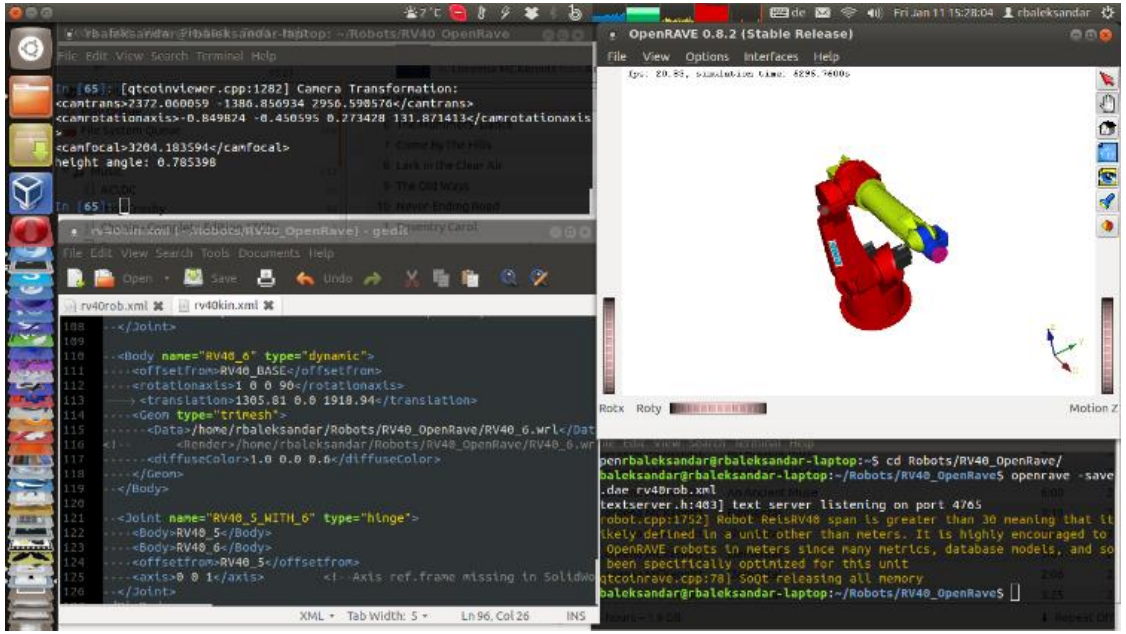
OpenRTM-aist (Advanced Industrial Science & Technology) is based on the CORBA technology to implement RTC extended specification

OPENRTM-AIST



OPENRAVE: OPEN ROBOTICS AUTOMATION VIRTUAL EVNIROMENT

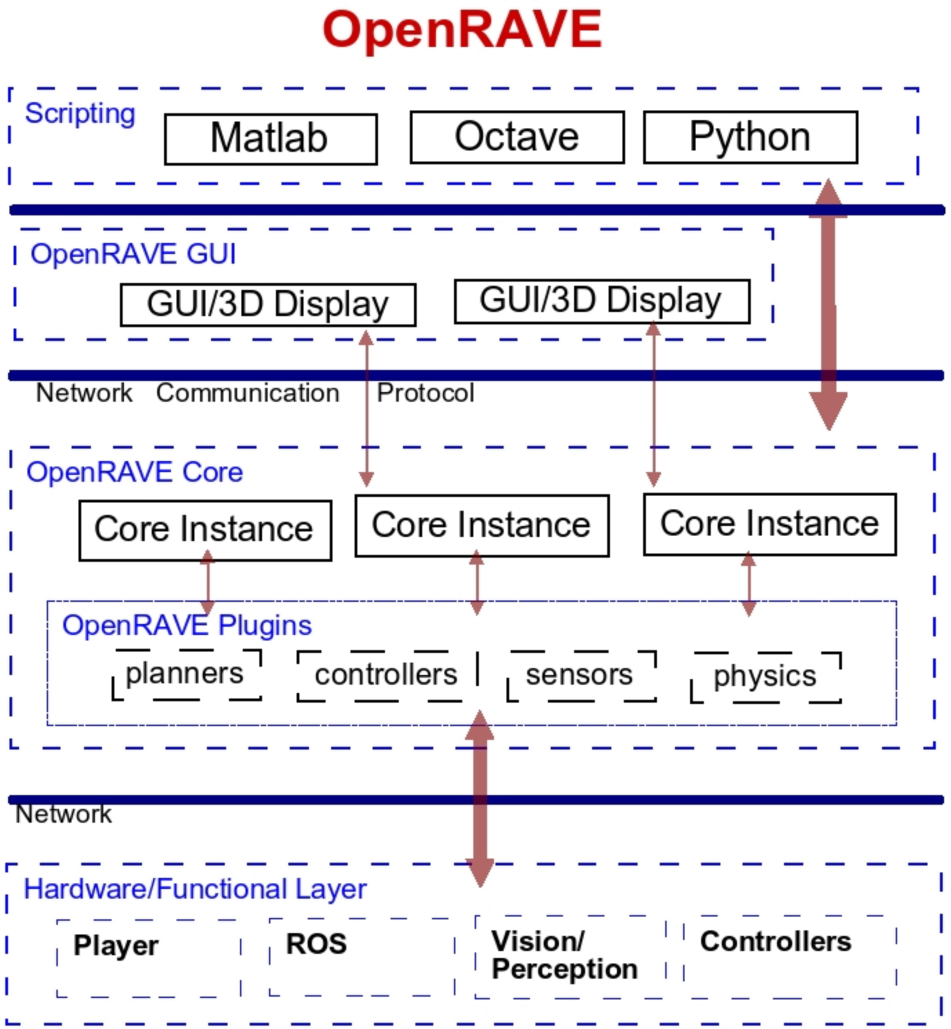
Proposed by Rosen Diankov provides an environment for testing, developing, and eploying motion planning algorithms in real-world robotics applications.


rbaleksandar@rbaleksandar-laptop: ~/Robots/RV40_OpenRave
File Edit View Search Terminal Help
OpenRAVE 0.8.2 (Stable Release)
File View Options Interfaces Help
rv40kin.xml (~/Robots/RV40_OpenRave) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
rv40rob.xml
rv40kin.xml
XML
Tab Width: 5
Ln 96, Col 26
INS
Rotx
Roty
Motion Z

```
In [65]: [qtcoinviewer.cpp:1282] Camera Transformation:
<camtrans>2372.060059 -1386.856934 2956.598576</camtrans>
<camrotationaxis>-0.849824 -0.450595 0.273428 131.871413</camrotationaxis
>
<camfocal>3204.183594</camfocal>
height angle: 0.785398

In [65]:
```

```
  </Joint>

  <Body name="RV40_6" type="dynamic">
    <offsetfrom>RV40_BASE</offsetfrom>
    <rotationaxis>1 0 0 90</rotationaxis>
      <translation>1305.81 0.0 1918.94</translation>
    <Geom type="trimesh">
      <Data>/home/rbaleksandar/Robots/RV40_OpenRave/RV40_6.wrl</Dat
<!--      <Render>/home/rbaleksandar/Robots/RV40_OpenRave/RV40_6.wr
      <diffuseColor>1.0 0.0 0.6</diffuseColor>
    </Geom>
  </Body>

  <Joint name="RV40_5_WITH_6" type="hinge">
    <Body>RV40_5</Body>
    <Body>RV40_6</Body>
    <offsetfrom>RV40_5</offsetfrom>
    <axis>0 0 1</axis>        <!--Axis ref.frame missing in Solidwo
  </Joint>
```

```
penrbaleksandar@rbaleksandar-laptop:~$ cd Robots/RV40_OpenRave/
baleksandar@rbaleksandar-laptop:~/Robots/RV40_OpenRave$ openrave -save
.dae rv40rob.xml
textserver.h:403] text server listening on port 4765
robot.cpp:1752] Robot ReisRV40 span is greater than 30 meaning that it
ikely defined in a unit other than meters. It is highly encouraged to
 OpenRAVE robots in meters since many metrics, database models, and so
 been specifically optimized for this unit
qtcoinrave.cpp:78] SoQt releasing all memory
baleksandar@rbaleksandar-laptop:~/Robots/RV40_OpenRave$
```

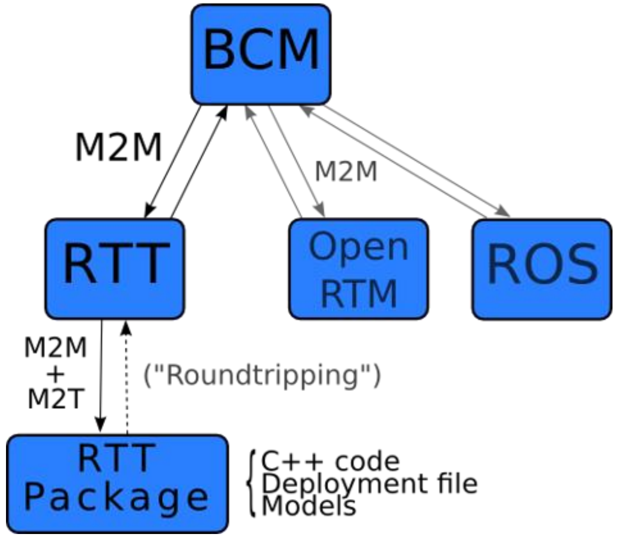


BRICS: BEST PRACTICES IN ROBOTICS

European project (2009) aimed at find out the "best practices" in the developing of the robotic systems:

- Investigate the weakness of robotic projects
- Investigates the integration between hardware & software
- Promote model driven engineering in robot development
- Design an Integrated Development Environment for robotic projects (BRIDE)
- Define showcases for the evaluation of project robustness with respect to BRICS principles.

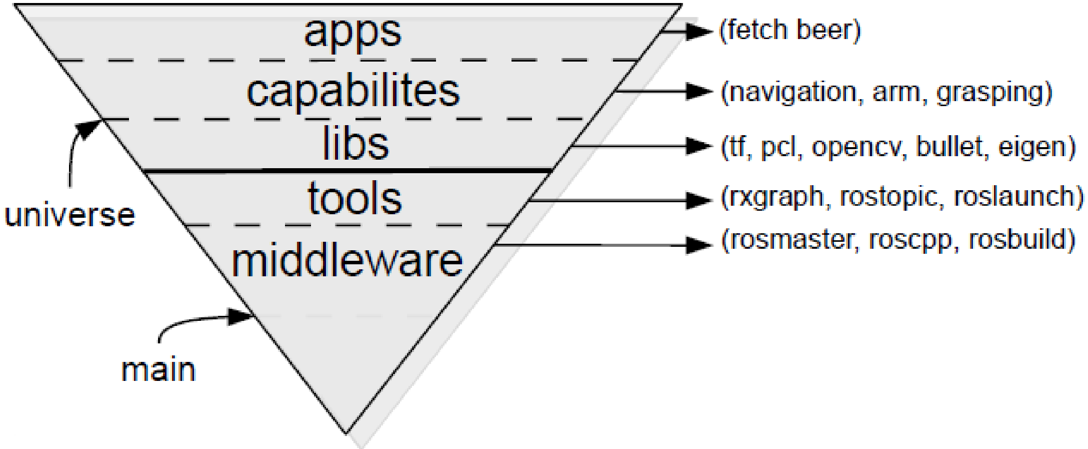
"The prime objective of BRICS is to structure and formalize the robot development process itself and to provide tools, models, and functional libraries, which help accelerating this process significantly."



ROS: ROBOT OPERATING SYSTEM

Presented in 2009 by Willow Garage is a meta-operating system for robotics with a rich ecosystem of tools and programs

ROS
=
Plumbing
+
Tools
+
Capabilities
+
Ecosystem



apps → (fetch beer)

capabilites → (navigation, arm, grasping)

libs → (tf, pcl, opencv, bullet, eigen)

tools → (rxgraph, rostopic, roslaunch)

middleware → (rosmaster, roscpp, rosbuild)

universe

main

CONCLUSIONS

Middleware in Robotics :

- Are widely used
- Component-based
- Based on asynchronous communication
- Implement some form of messages exchange architecture
- Support different robot architectures by default (PR2, NAO, AIBO, ROOMBA, iCUB, etc..)
- Way too many…

In the course we will use ROS as reference middleware because:

- Easy to learn, install, and deploy
- Lots of components already available
- Middleware used (currently) on AIRLab robots