POLITECNICO DI MILANO





Cognitive Robotics – Introduction

Matteo Matteucci – matteo.matteucci@polimi.it

About me and my lectures …

Lectures given by Matteo Matteucci

- +39 02 2399 3470
- matteo.matteucci@polimi.it
- http://www.deib.polimi.it/people/matteucci

Research Topics (several Thesis available)

- Robotics and Autonomous Systems
- Computer Vision and Perception
- Pattern Recognition & Machine Learning
- Benchmarking in Robotics

Aims of the lectures: learning how to design and implement the software which makes autonomous an autonomous mobile robot (e.g., symbolic planning, trajectory planning, localization, perception, mapping, etc.)

Lectures Outline

Middleware in robotics
- Motivations and state of the art
- A case study: ROS

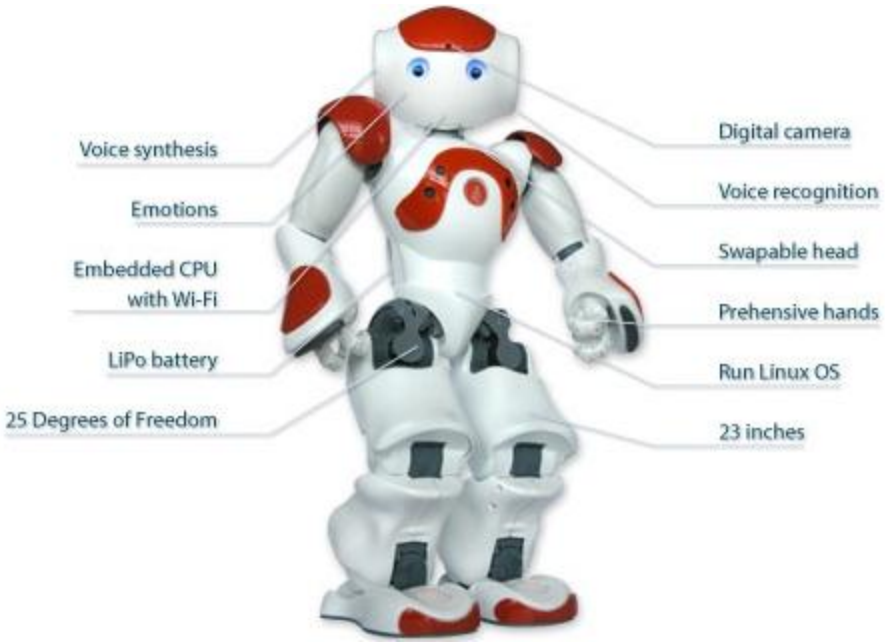
Symbolic Planning
- The PDDL language
- PDDL Extensions

Path planning
- Path planning in ROS
- SPBL vs OMPL

Localization and Mapping
- Localization vs Mapping
- Simultaneous Localization and Mapping
- 2D Navigation and localization in ROS

Object recognition
- 3D object recognition with RGBD cameras (kinect)

The NAO Case Study



Matteo Matteucci – matteo.matteucci@polimi.it

POLITECNICO DI MILANO





Cognitive Robotics – Robotics Middlewares

Matteo Matteucci – matteo.matteucci@polimi.it

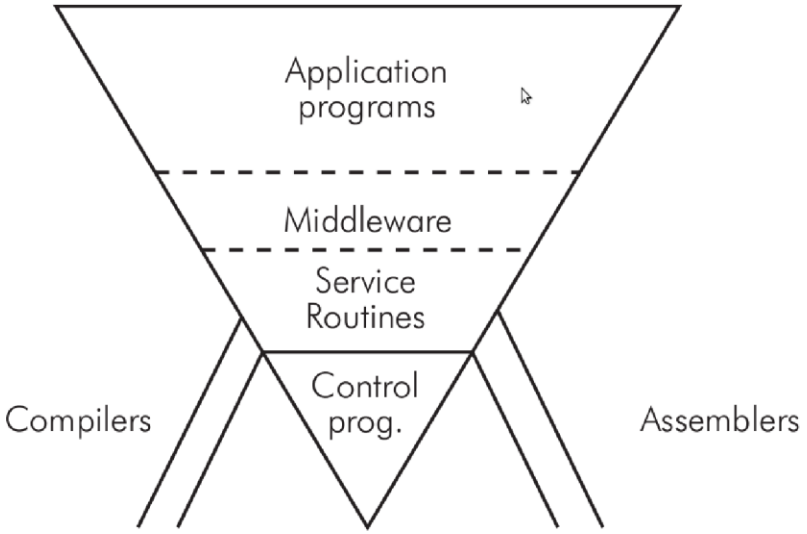
Why a middlewares for robotics?

Issues in developing real robots

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

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



The origins

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

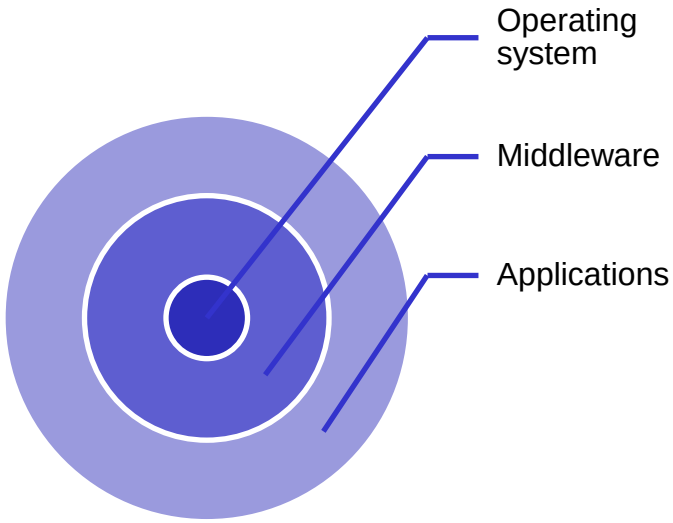
What is a Middleware?

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).



Some (non robotics) examples: Android, SOAP, Web Services, …

Why should we use a middleware?

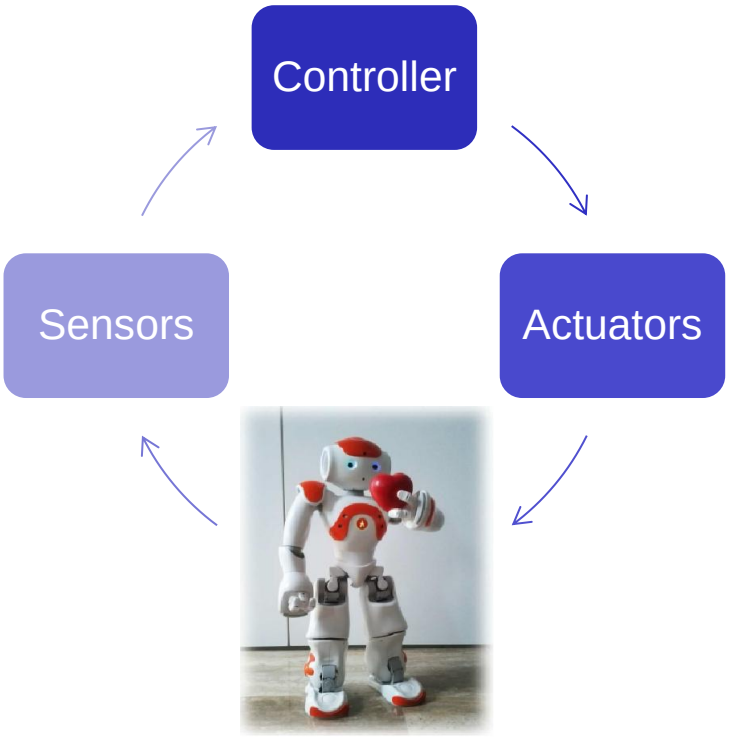
Portability: provides a common programming model regardless the programming language and the system architecture.

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.

A Classical Approach to 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.



Siciliano et al., 2011

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**!

How Many Middleware Out There?

Several middleware have been developed in recent years:

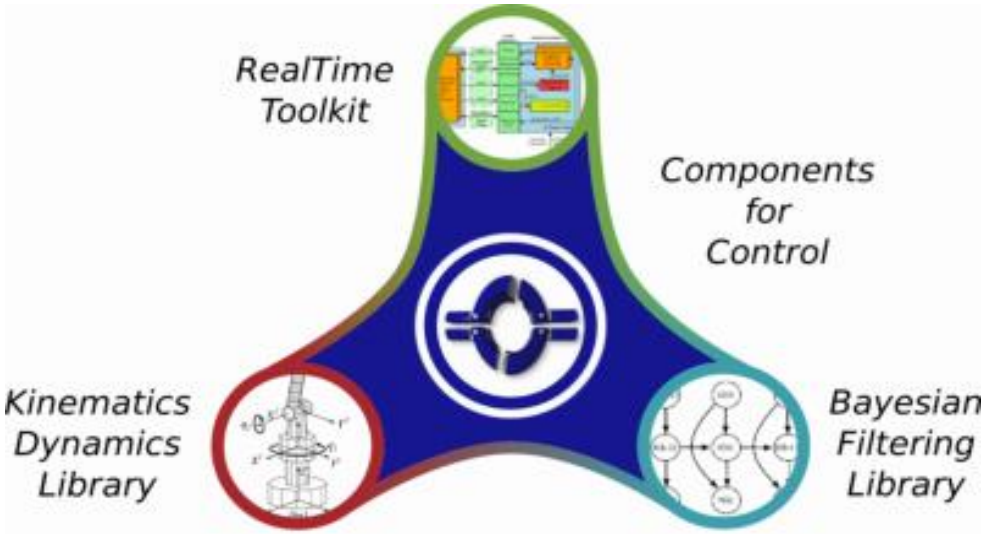
- OROCOS [Europe]
- ORCA [Europe]
- YARP [Europe / Italy]
- BRICS [Europe]
- OpenRTM [Corea]
- OpenRave [US]
- ROS [US]
- …

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 and 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 (C++ Libraries)

OROCOS Real-Time Toolkit (RTT)

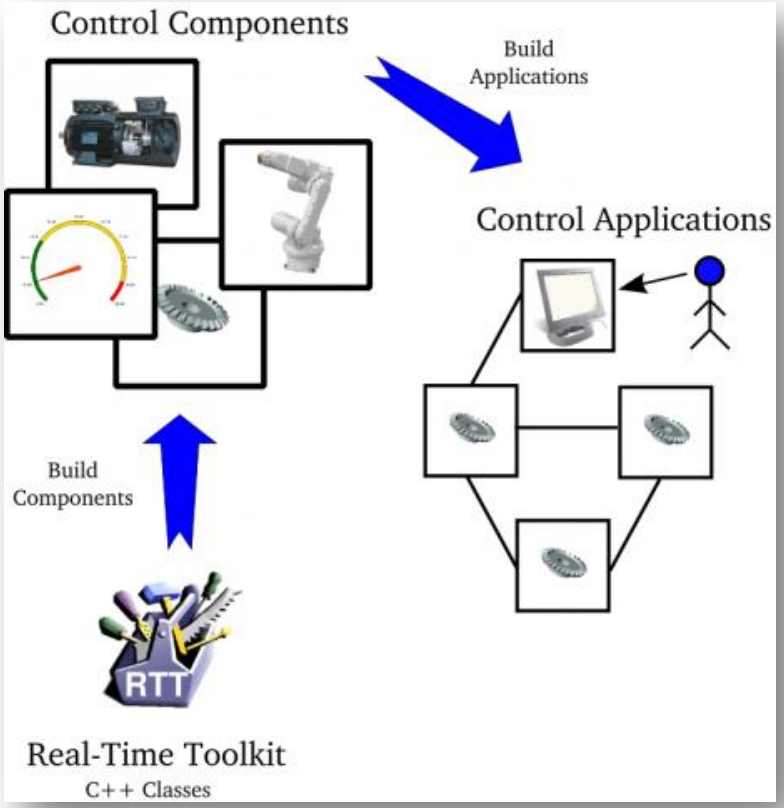
- infrastructure and functionalities for real-time robot systems
- component-based applications

OROCOS Component Library (OCL)

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

OROCOS Bayesian Filtering Library (BFL)

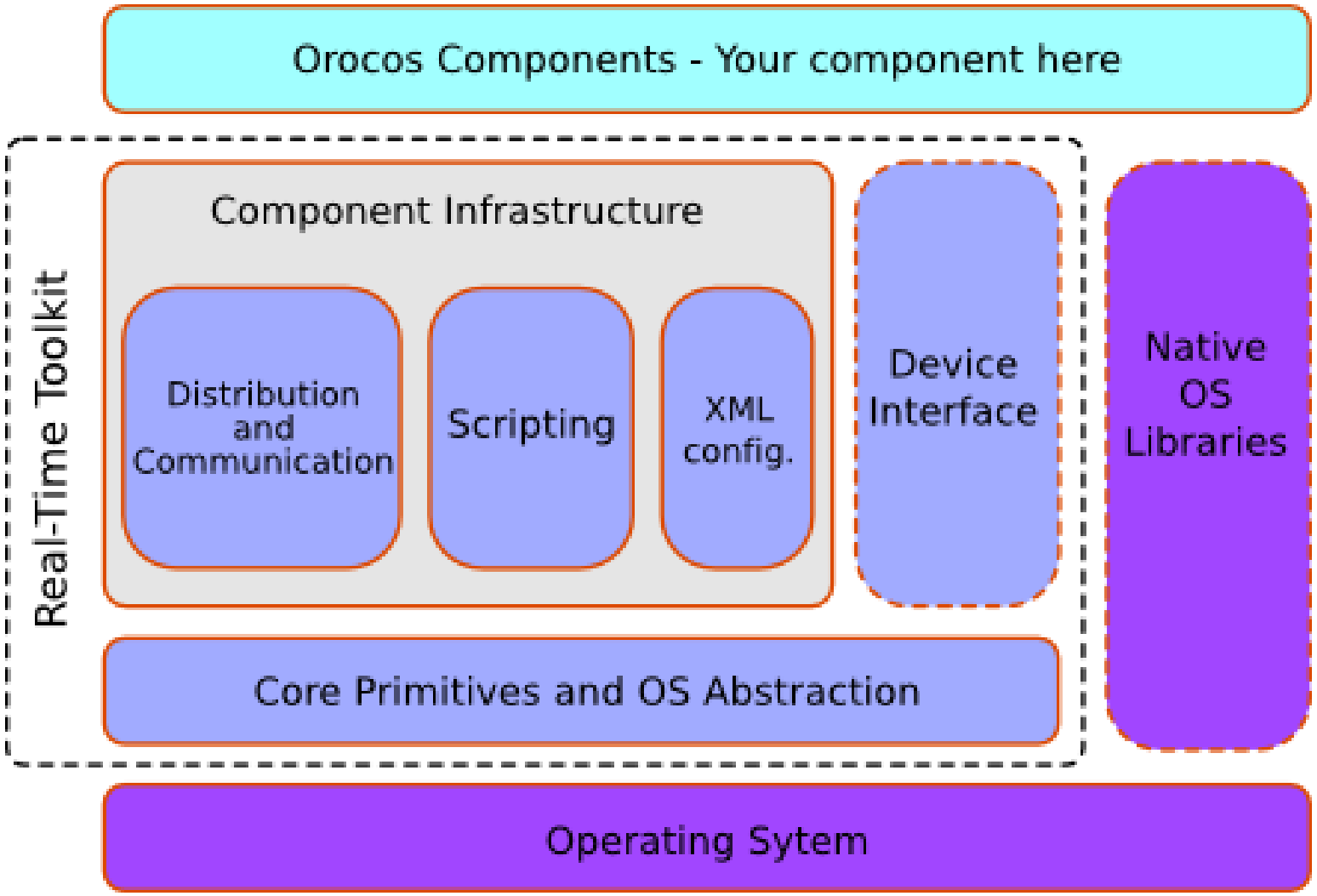
- application independent framework, e.g., (Extended) Kalman Filter, Particle Filter



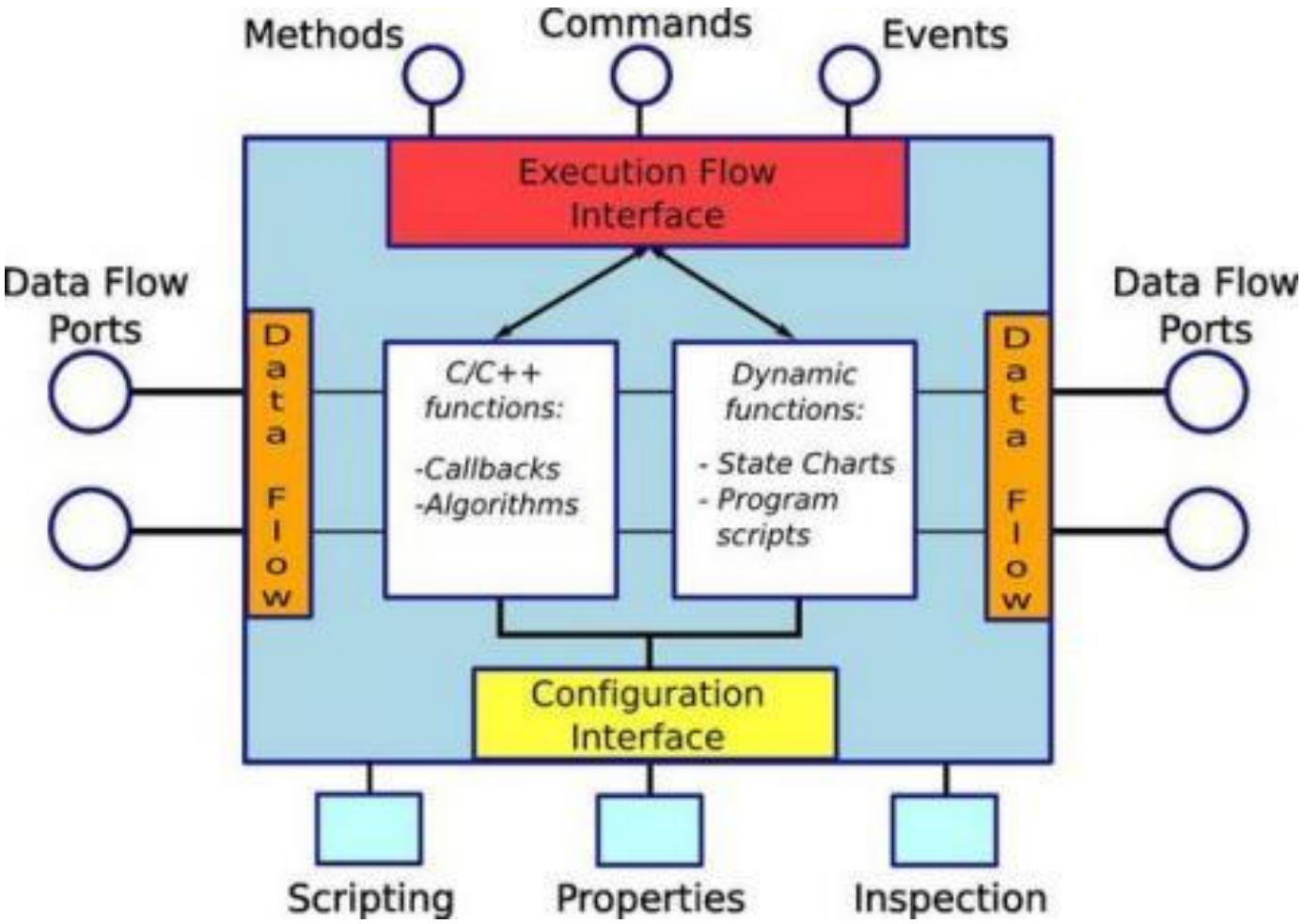
OROCOS Kinematics & Dynamics Library (KDL)

- real-time kinematics & dynamics computations

OROCOS RTT Framework

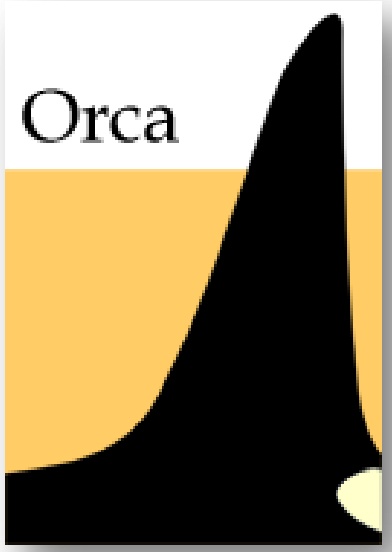


OROCOS Components



ORCA: Components for Robotics

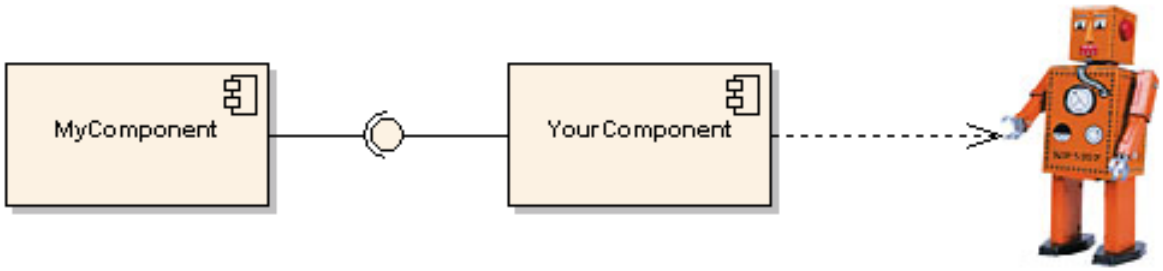
The aim of the project is to focus on software reuse for scientific and industrial applications



Key properties:

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

ORCA defines itself as “unconstrained component-based system”



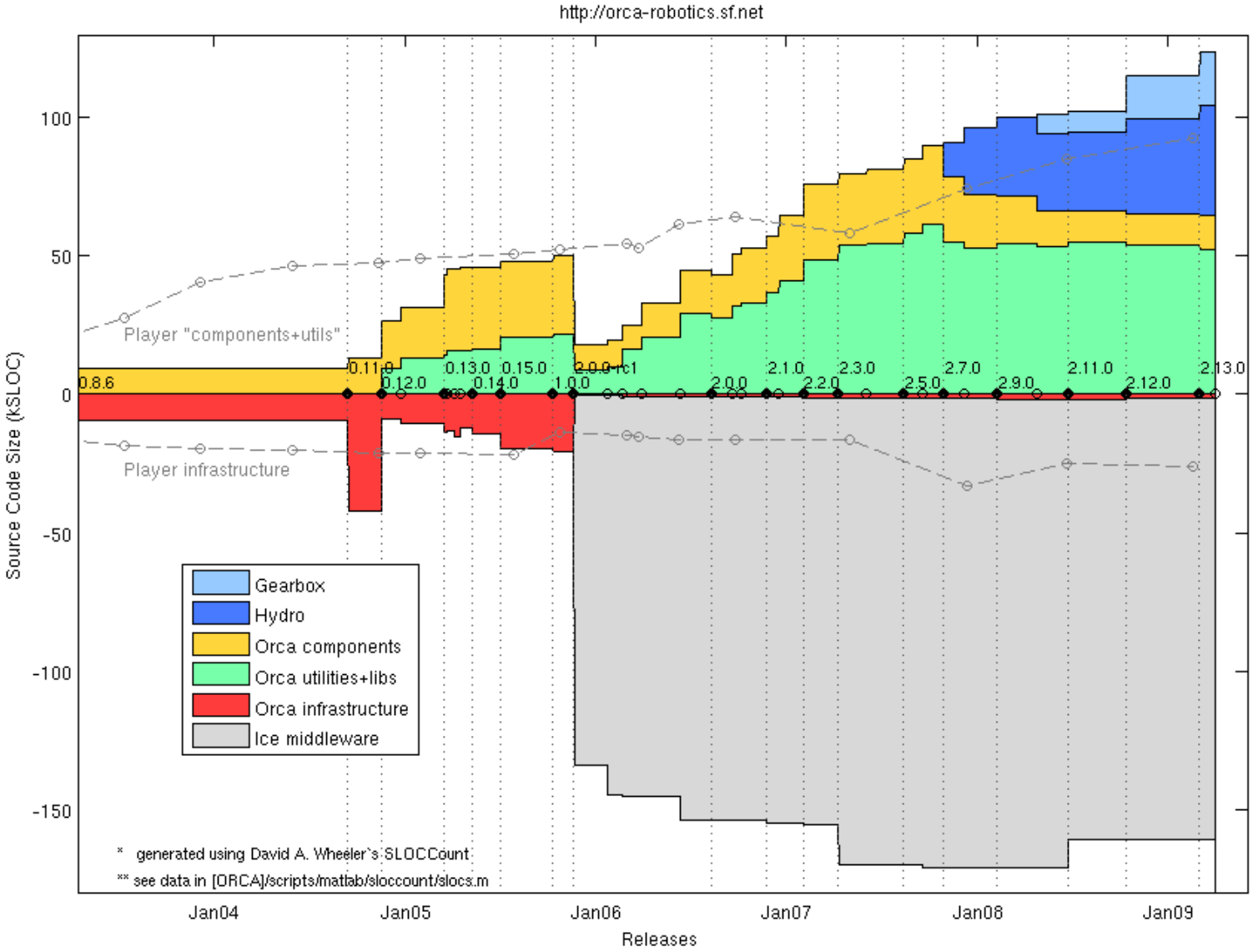
ORCA and ICE

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 Libraries Evolution

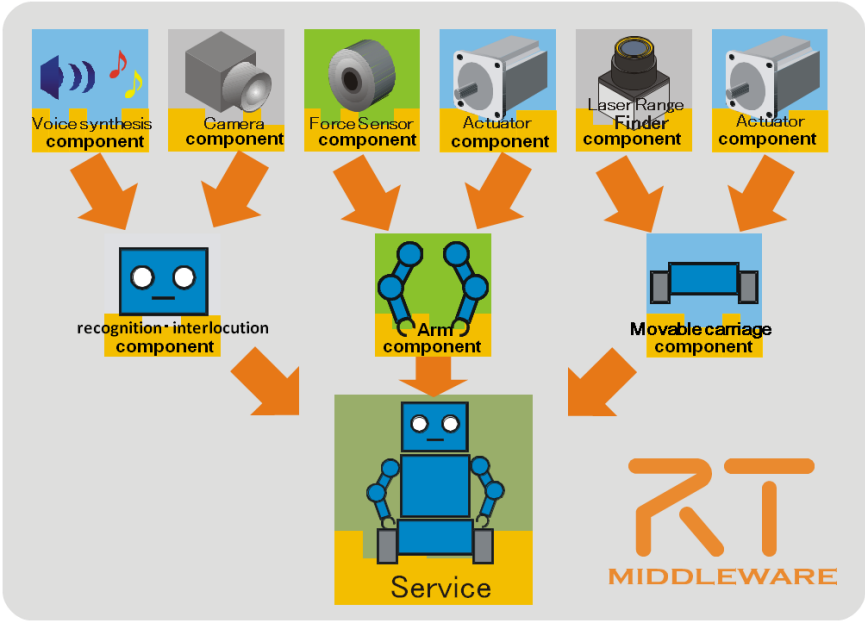


Matteo Matteucci – matteo.matteucci@polimi.it

OpenRTM-aist

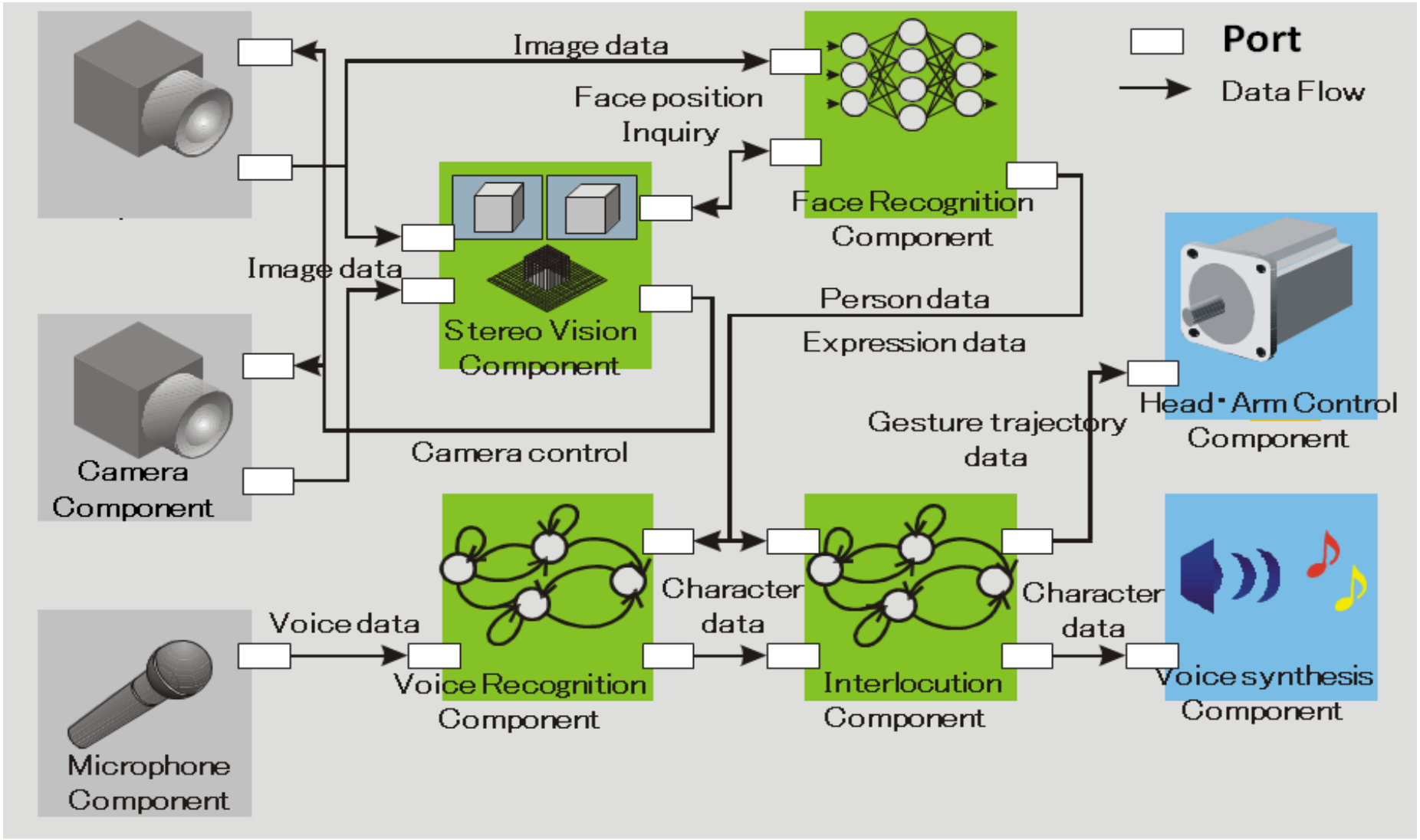
RT-Middleware (RTM) is a software platform to construct the robot system by combining the software modules of the robot functional elements (RTC):

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



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

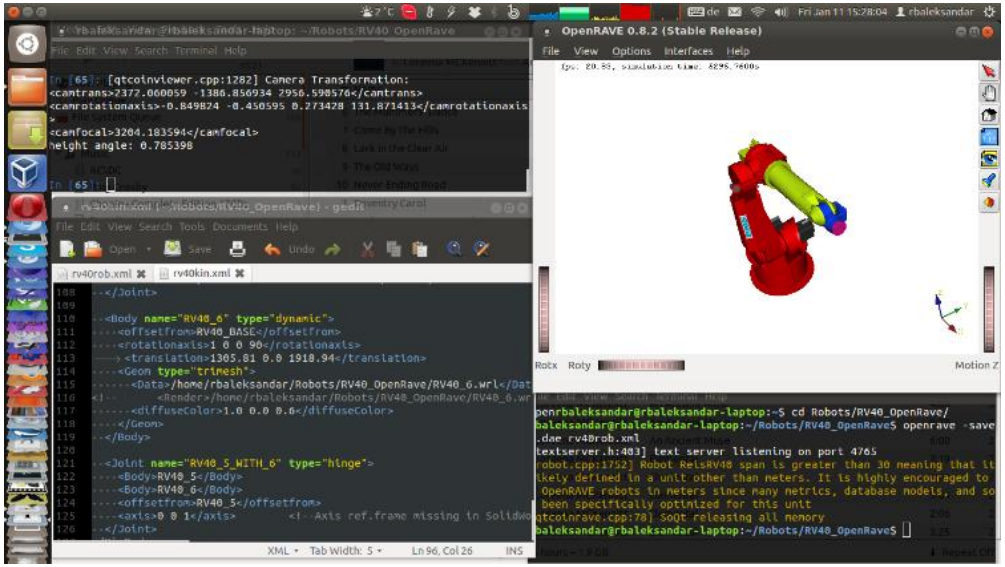
OpenRTM-aist Overview

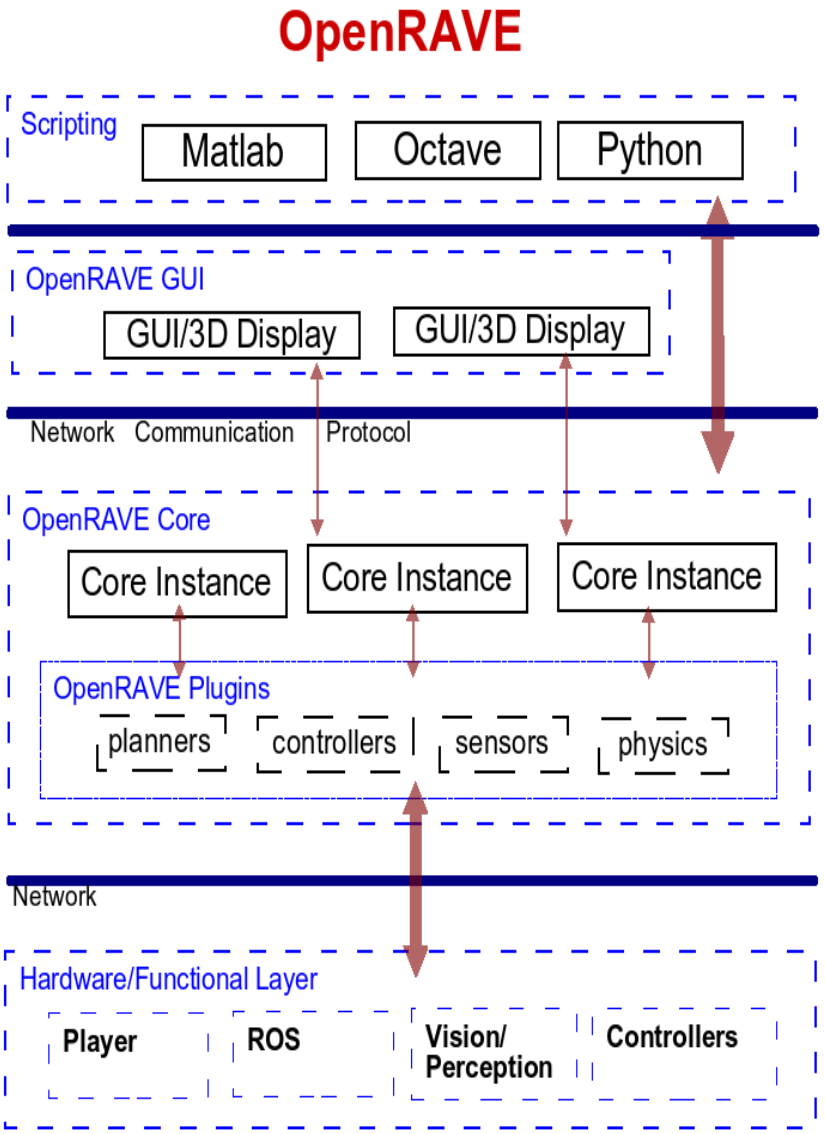


Matteo Matteucci – matteo.matteucci@polimi.it

OpenRave: Open Robotics Automation Virtual Environment

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

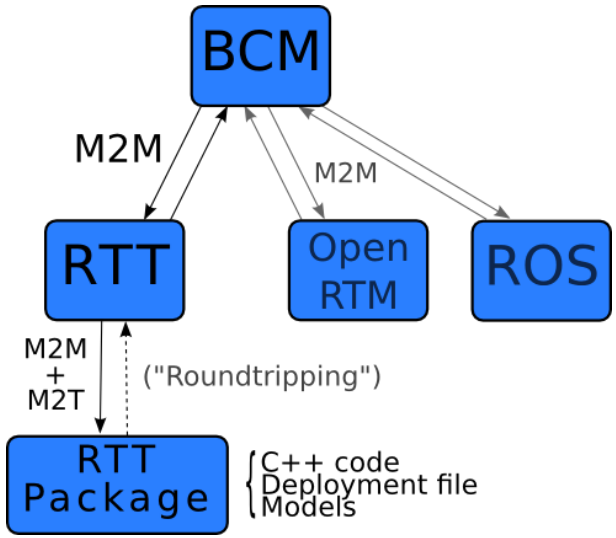




BRICS: Best Practices in Robotics

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

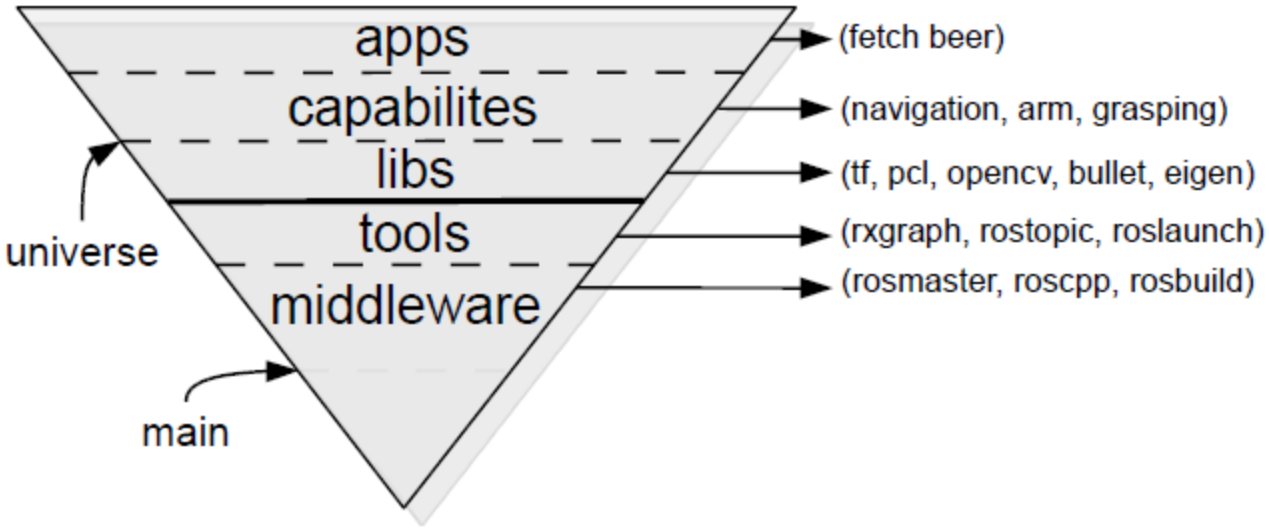
- Investigate the weakness of robotic projects
- Investigates the integration between hardware & software
- Design an Integrated Development Environment for robotic projects (BRIDE)
- Define showcases for the evaluation of project robustness with respect to BRICS principles.



"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



Final (?) Remarks

Middleware in Robotics :

- Are widely used
- Component-based
- Based on asynchronous communication
- Implement Publisher-Subscriber architectures
- Support different robot architectures by default (PR2, NAO, AIBO, ROOMBA, iCUB, etc..)
- …
- They are 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

POLITECNICO DI MILANO





Cognitive Robotics – Robotics Middlewares

Matteo Matteucci – matteo.matteucci@polimi.it