

Software componentization for robotics

*Mixing middleware, architectures, and several robot
types*



ISTITUTO ITALIANO
DI TECNOLOGIA



Giorgio Metta

Scientific Director

Italian Institute of Technology

L. Natale, D. Pucci,

U. Pattacini

Italian Institute of Technology



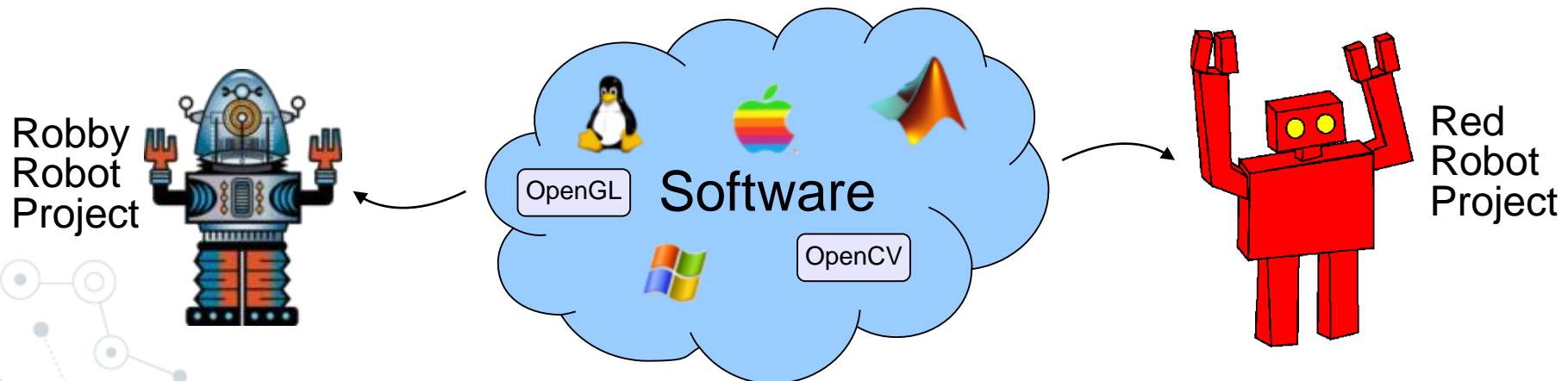
ISTITUTO ITALIANO
DI TECNOLOGIA



Once
upon
a
time...

The sad fate of most robot software

- Writing software is difficult and time consuming
- Our software tends to die with our projects/students
- Sad! Software collaboration **speeds things up**
- Code sharing could **promote successful components**



Barriers to software collaboration

- Groups developing on **different robots face obstacles**
 - Differences in sensors, actuators, bodies...
 - Differences in processors, operating systems, libraries, frameworks, languages, compilers...
- Lack of **reward** for producing reusable code
- Research groups that all use a **specific robot** (Khepera, Pioneer, AIBO, ...) often form a **natural software community**
 - But each alone is a **small subset** of robotics

The popular
robots in year
2001

Yet Another Robot Platform



- YARP is an open-source (BSD) middleware for humanoid robotics
- History
 - An MIT / Univ. of Genoa collaboration
 - Born on Kismet, grew on COG, under QNX
 - With a major overhaul, now used by RobotCub consortium
- Exists as an independent open source project (GitHub)
- C++ source code (mostly)



2000-2001



2001-2002



2003



2004-Today

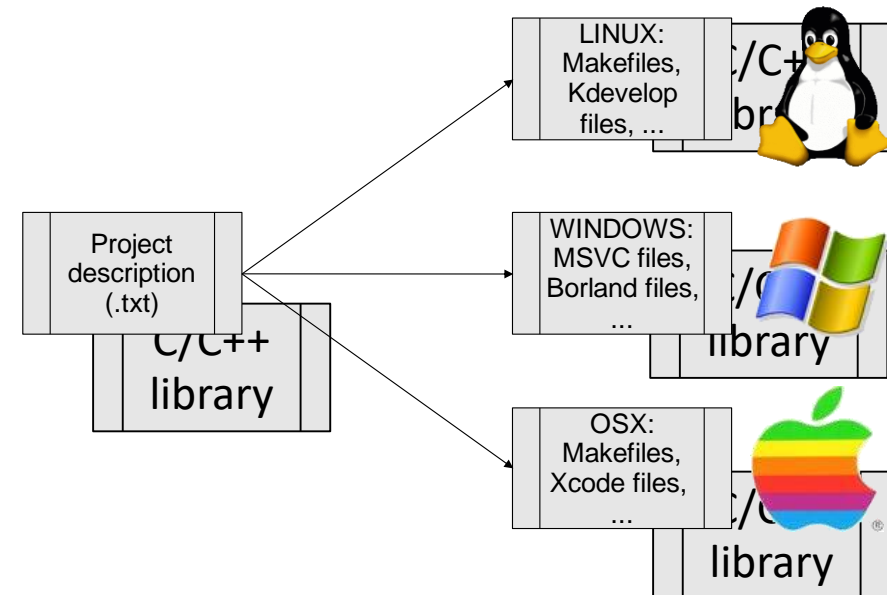


philosophy

- One processor is never enough
- Modularity
- Minimal interference
- Stopping (the robot) hurts
- Humble approach (thin middleware)
- Exploit diversity

Exploit diversity: portability

- Operating system portability:
 - Adaptive Communication Environment , C++ OS wrapper: e.g. threads, semaphores, sockets
- Development environment portability:
 - CMake
- Language portability:
 - Via Swig: Java (Matlab), Perl, Python, C#



Achieving modularity

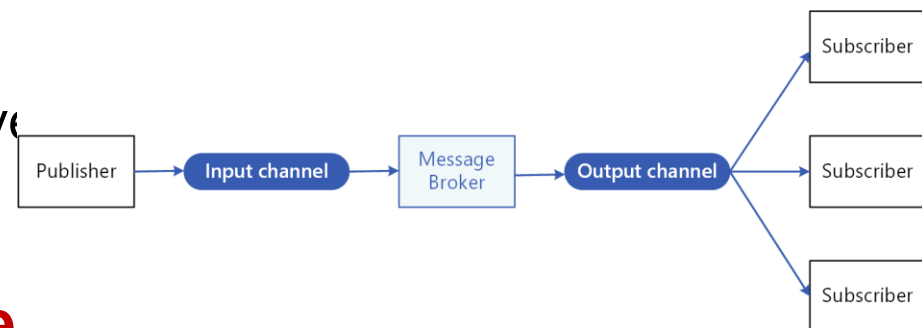
- Factor out **details of data flow between programs** from program source code

- Data flow is very specific to robot platform, experimental setup, network layout, communication protocol, etc.
- Useful to keep “algorithm” and “plumbing” separate

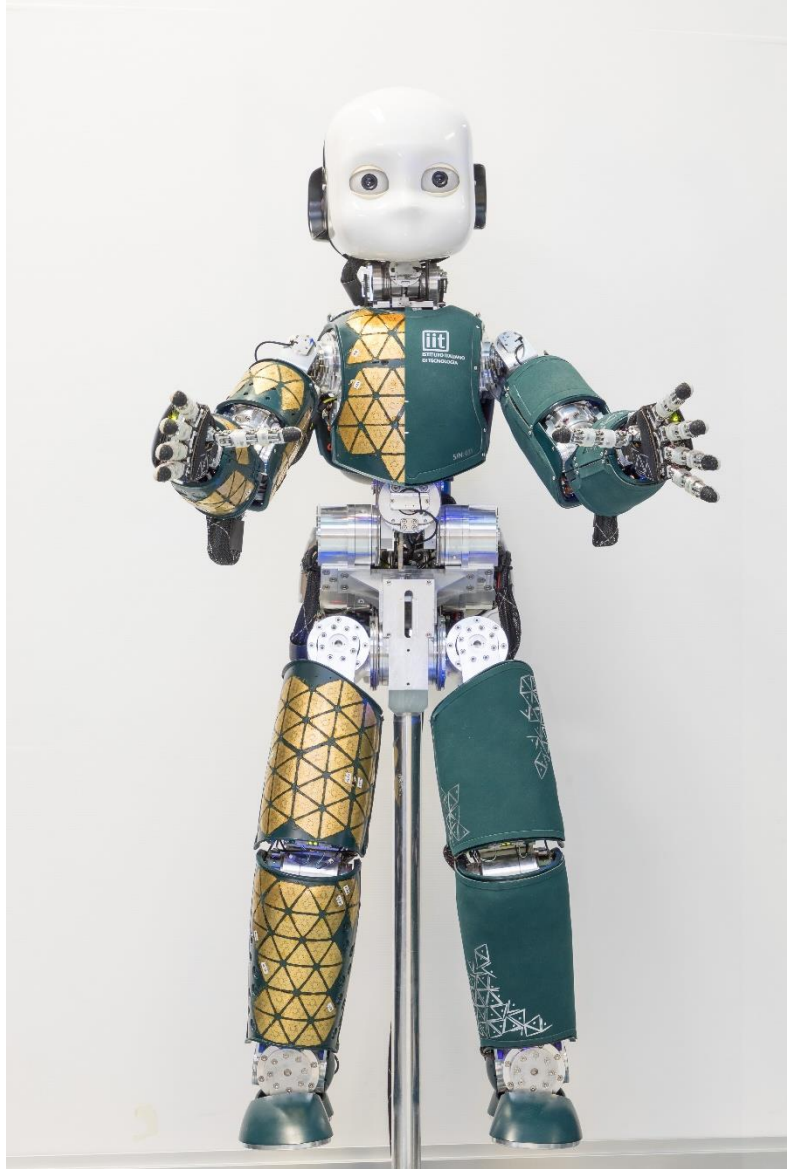
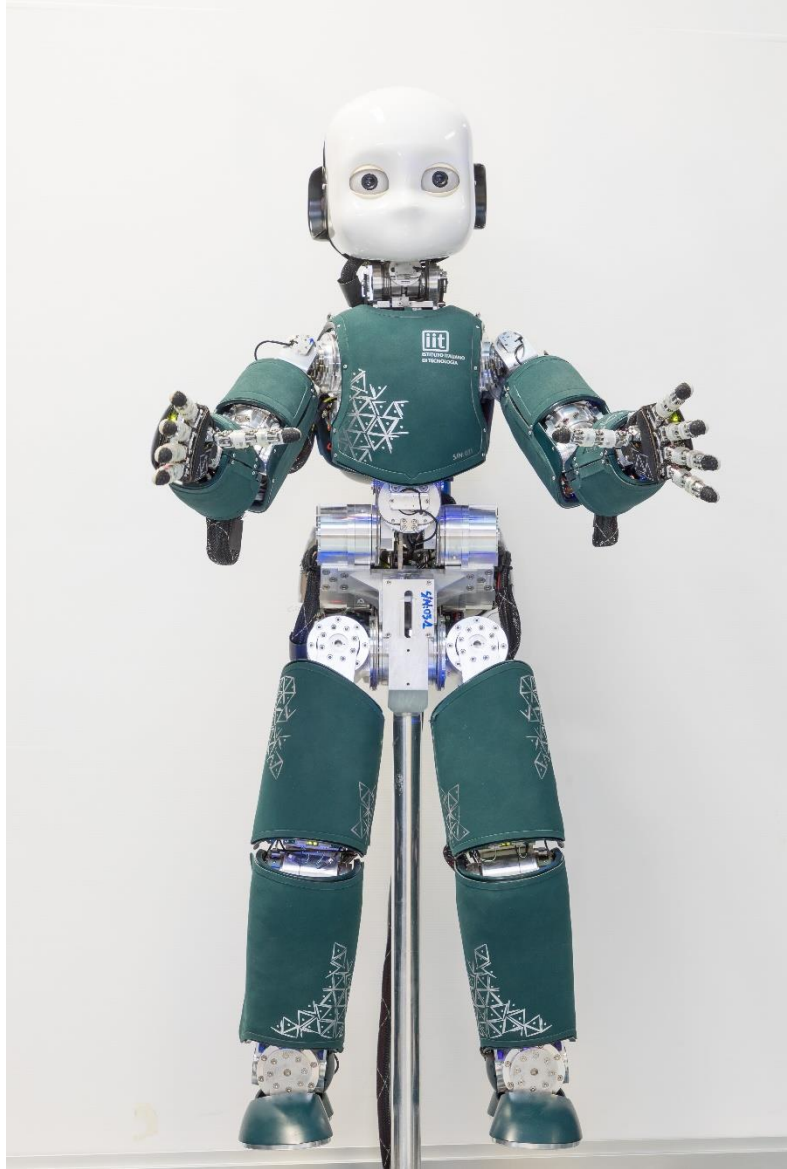
- Factor out **details of devices used by programs** from program source code

- The devices can then be replaced over time and can be used in other systems

- The pattern: **publisher-subscribe.**



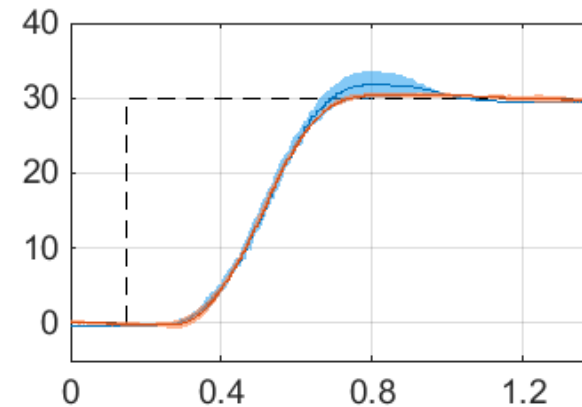
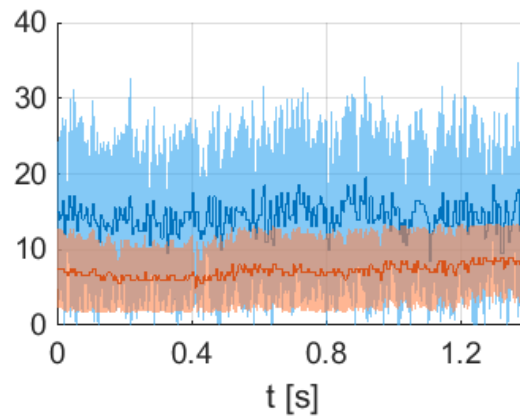
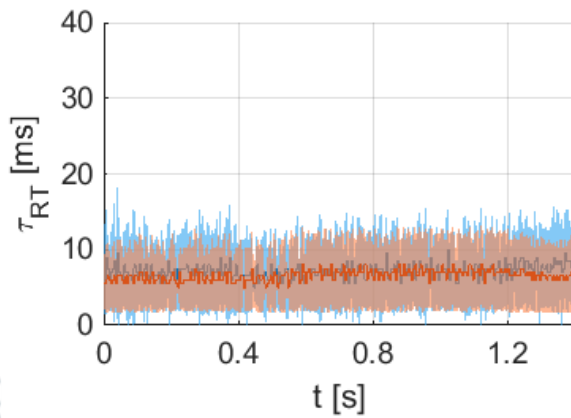
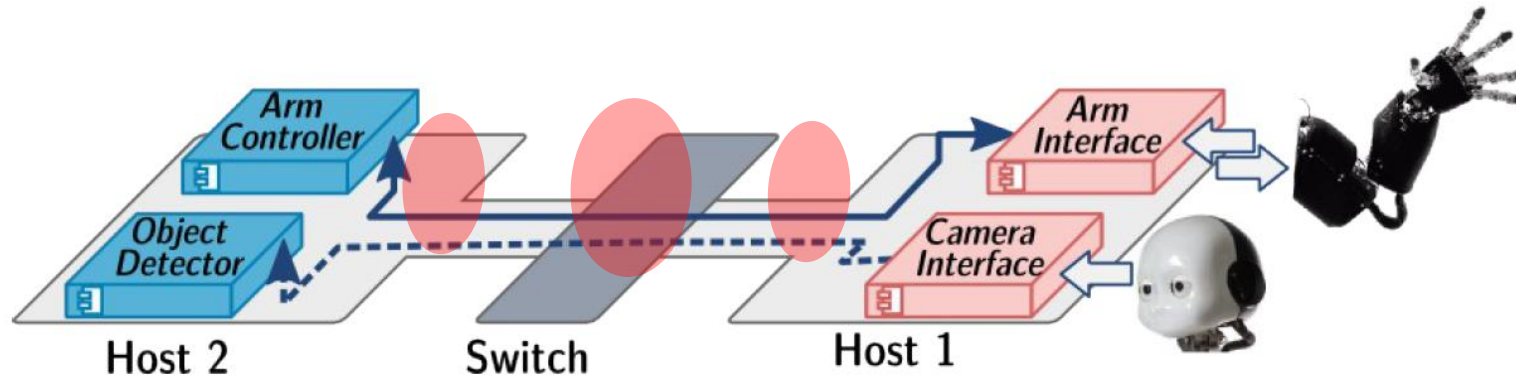
source code



**SCUDERIA
FERRARI**



channel prioritization



Paikan, A., et al., A Best-Effort Approach for Run-Time Channel Prioritization in Real-Time Robotic Application, IROS 2015

Paikan, A., et al. Data Flow Port's Monitoring and Arbitration, Journal of Software Engineering for Robotics, 2014



carrier plug-ins



yarp connect /camera /receiver



yarp connect /65.52.88.202:5159 /receiver mjpeg

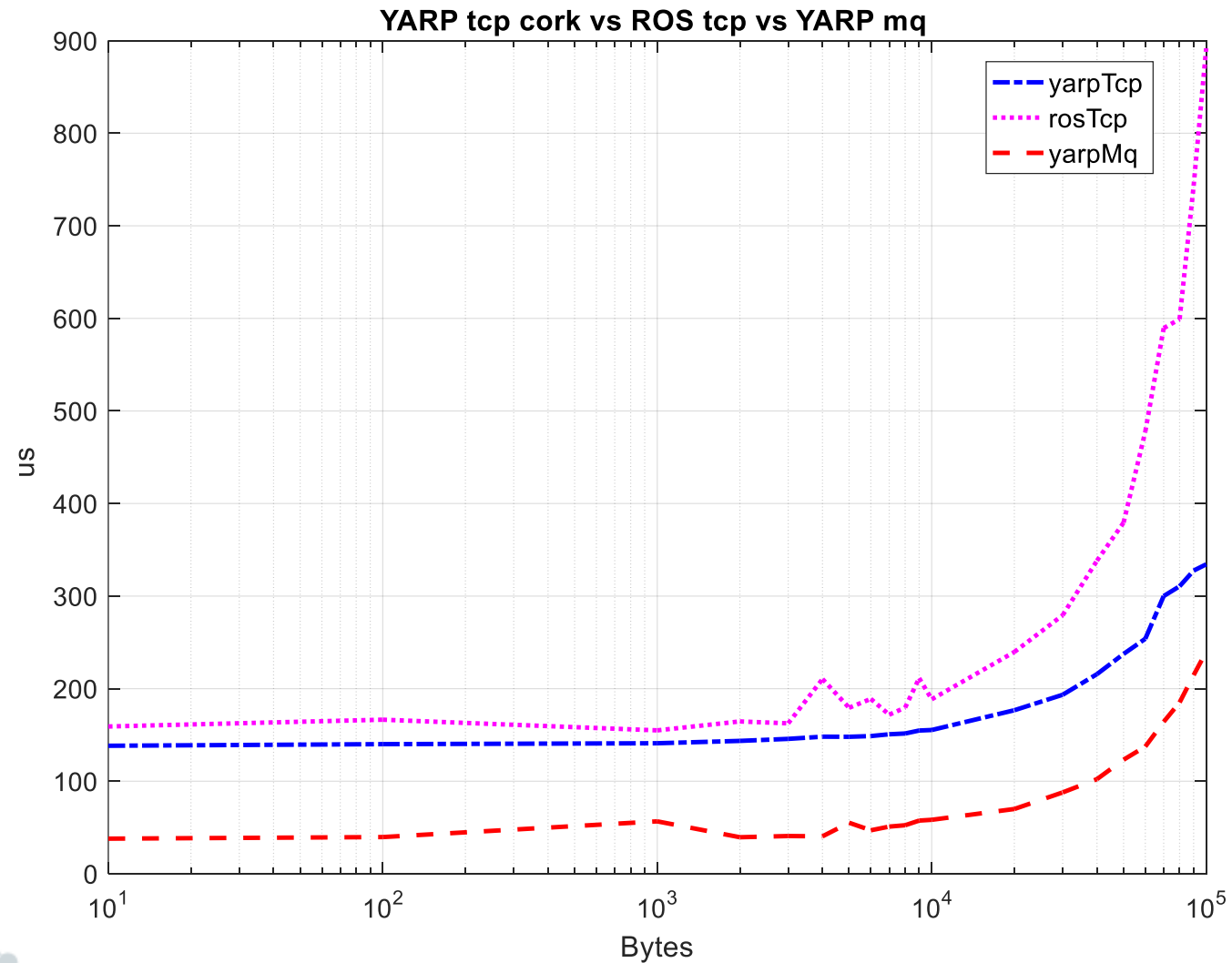


yarp connect /image@/camera /receiver

Camera.msg

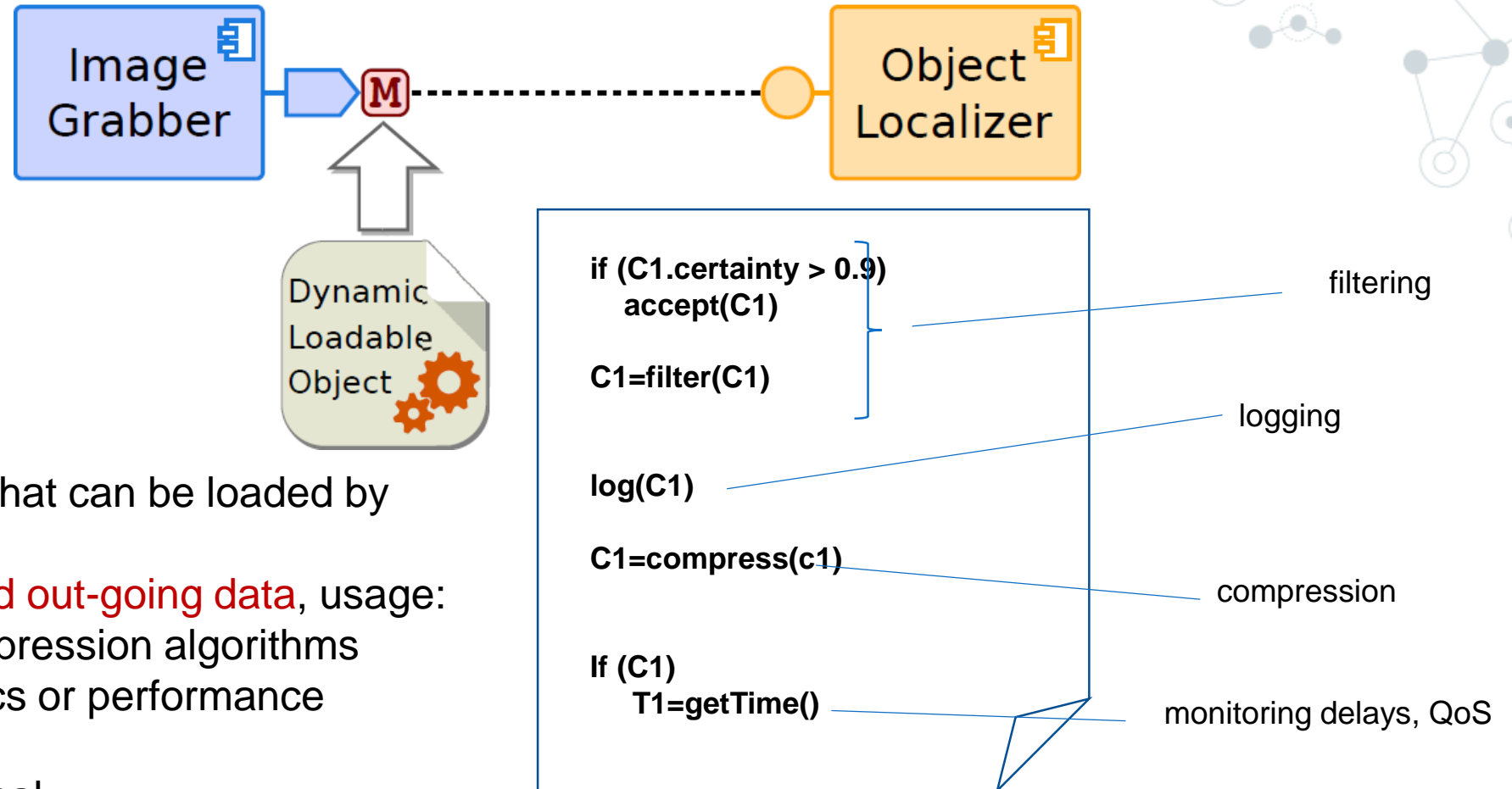


custom, efficient, protocols





Port Monitor



The **Port Monitor** is a plug-in that can be loaded by any connection point

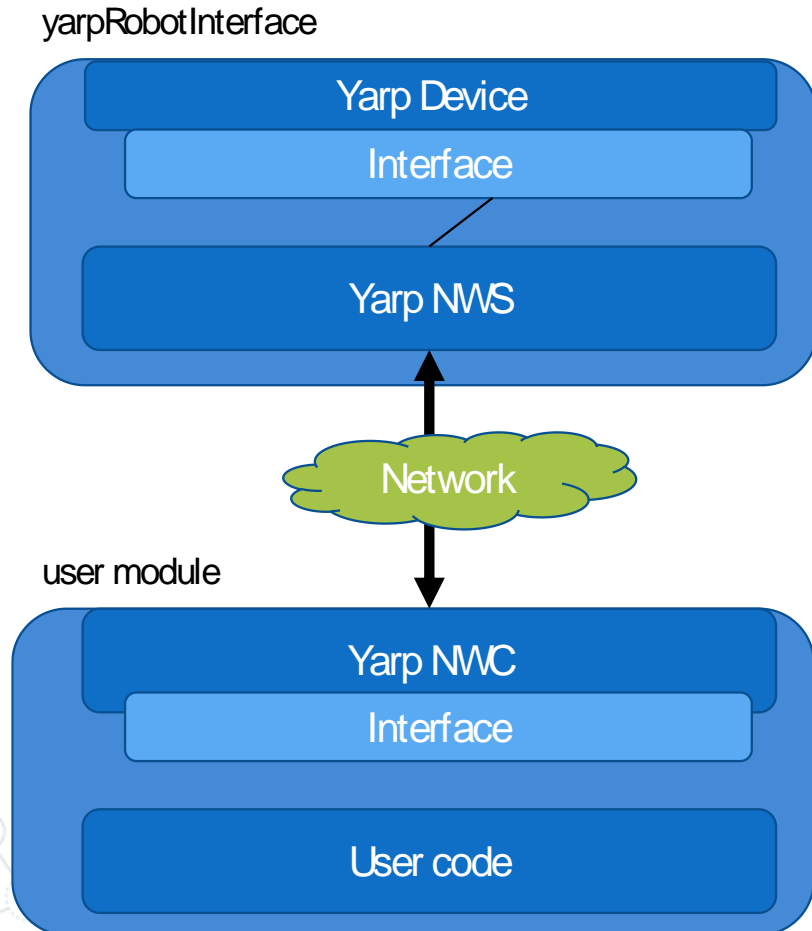
It has access to **in-coming and out-going data**, usage:

- Add compression/de-compression algorithms
- Log (e.g. compute statistics or performance indicators)
- Sniff data, also bi-directional

Avoid explicit **man-in-the-middle** components



device modularity



Yarp Device:

- A plugin which exposes the functionalities of a hardware device through a standardized Yarp C++ Interface.

Yarp NWS:

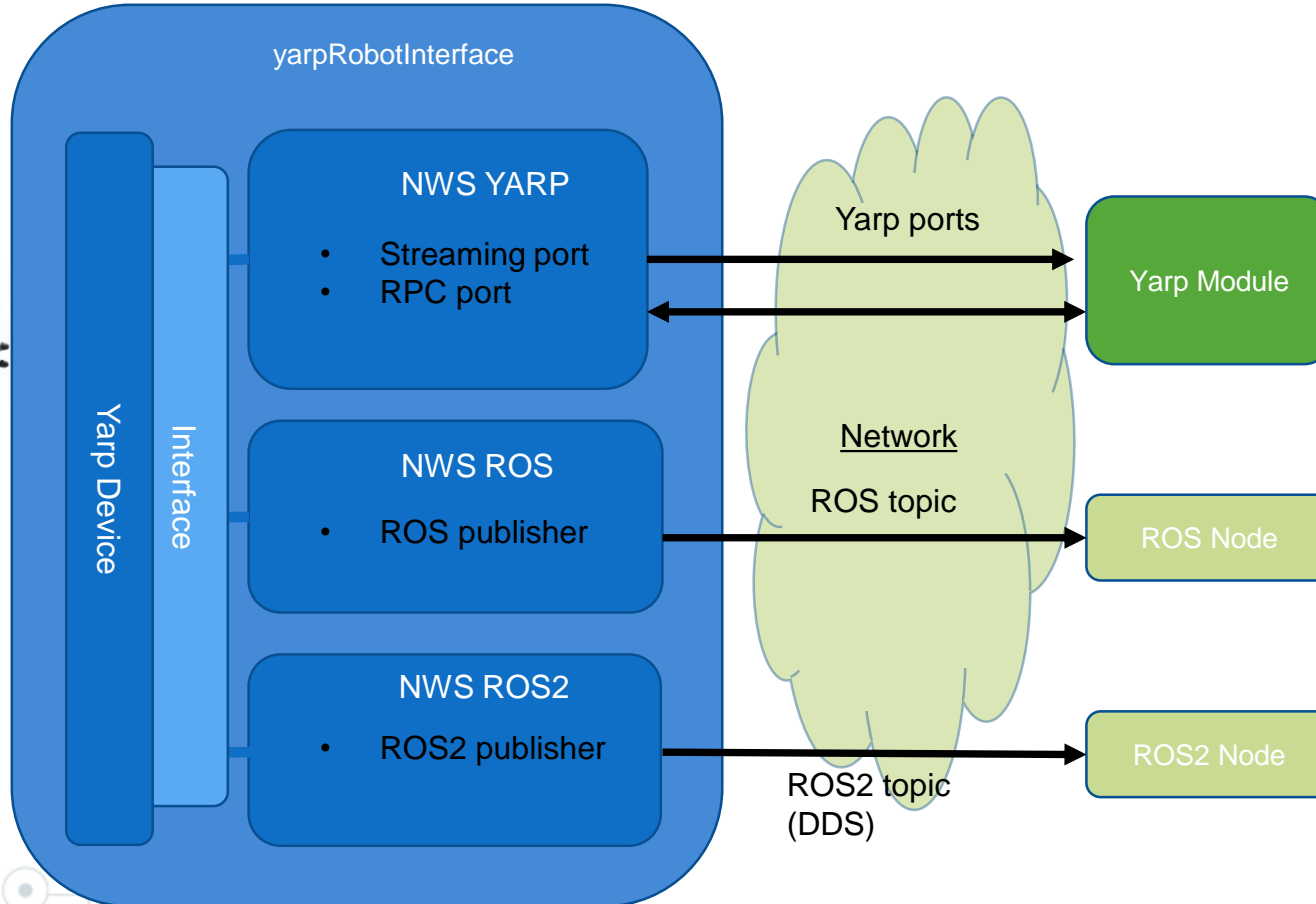
- A Network Wrapper Server (NWS) is a software component (plugin) attached to a physical device. It does not contain any logic. It just exposes the interface to the network.

Yarp NWC:

- A Network Wrapper Client (NWC) is a software component which implements the same interface of a real device, but instead of being connected to a physical hardware, it communicates with a Yarp NWS.



NWS/NWC



NWS/NWC ARCHITECTURE

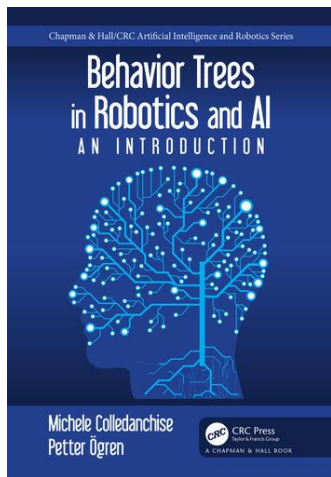
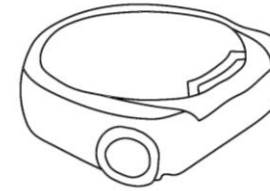
- The code is well separated, and the functionality of each component is clear.
- Easy to maintain.
- Easy to extend.
- **NWS** can be used to make Yarp to communicate with different middlewares (which use different network/serialization protocols)
 - Yarp (yarp ports protocol)
 - ROS noetic (ros topics)
 - ROS2 humble (ros2 topics with DDS)
 - IsaacSDK Nvidia
- **Multiple NWSs** can be used simultaneously to expose the same plugin to multiple middlewares.

Orchestration of behaviors: the problem



Due to a **human programming error**, the robot fell when transitioning from the **driving task** to the **egress task** (the foot **throttle controller wasn't turned off**). This caused the robot to fall and faceplant out of the car onto the asphalt.

Source: <http://drc.mit.edu/>



Rethink's Robots Get Massive Software Upgrade, Rodney Brooks "So Excited" (IEEE Spectrum)

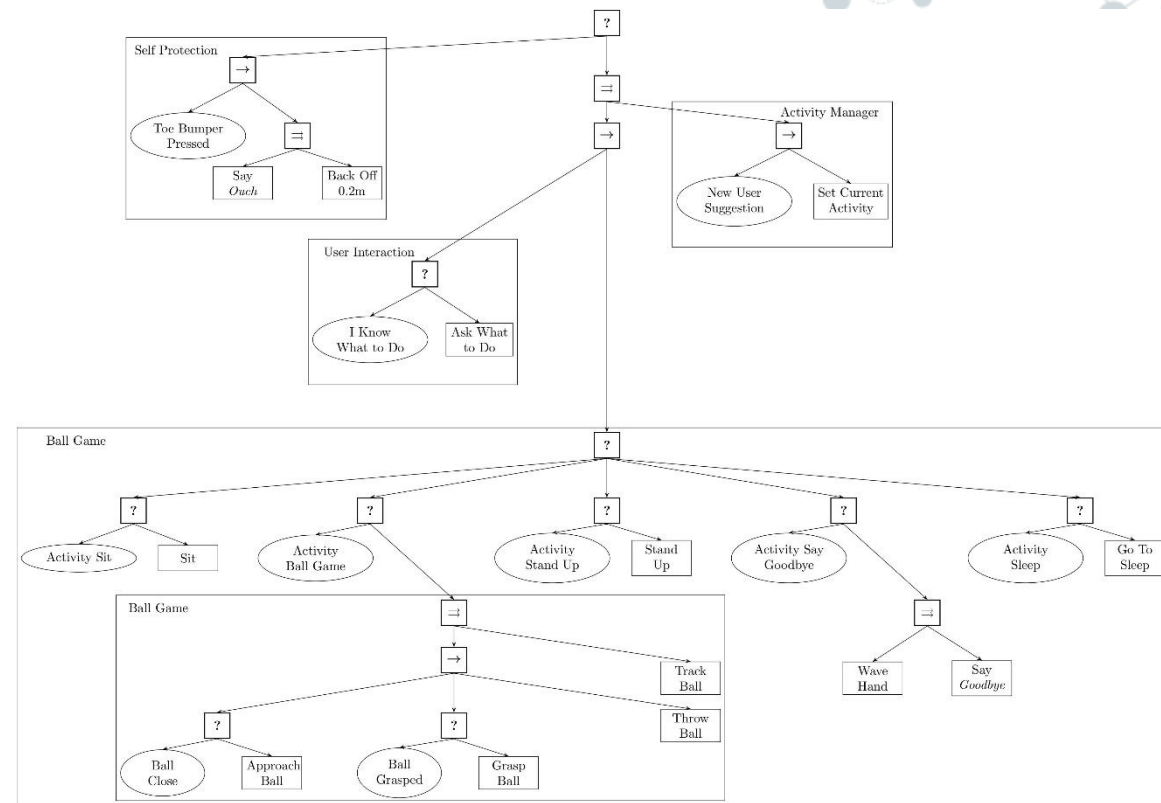
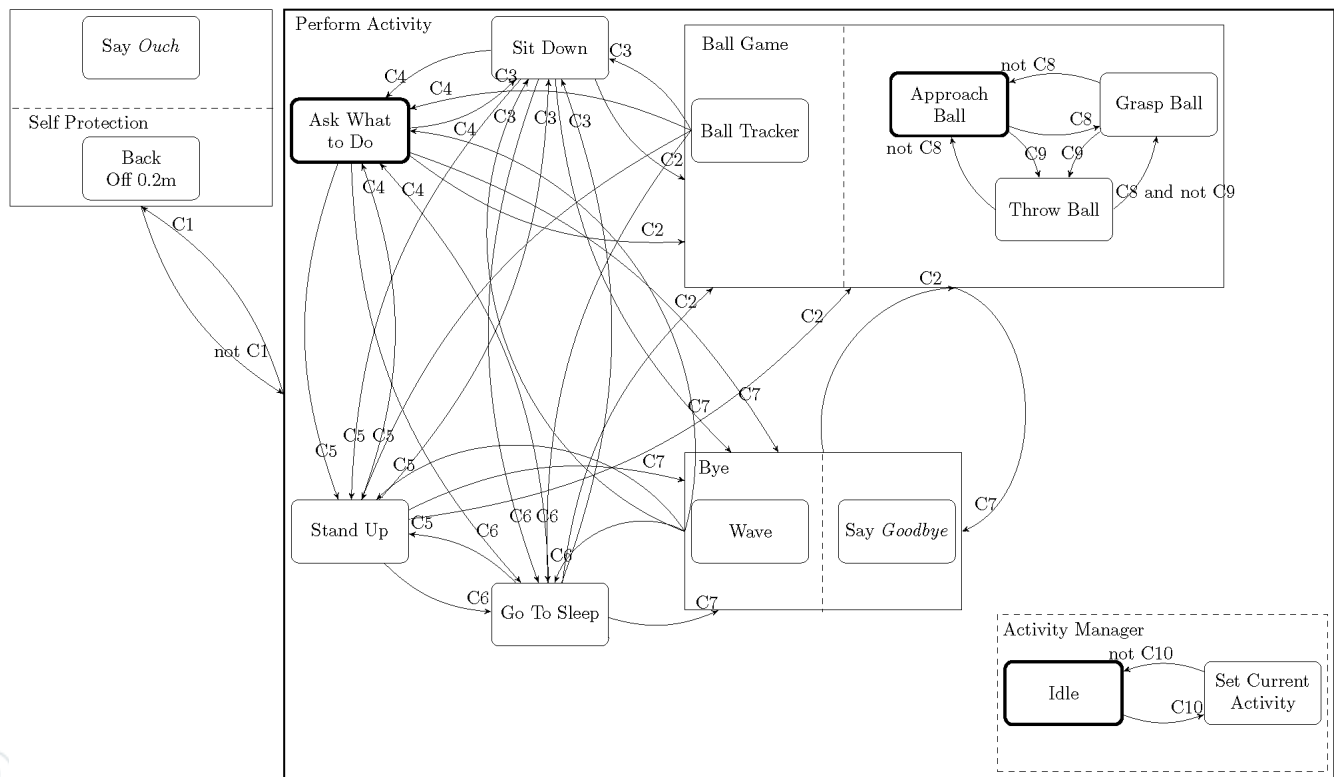


bostondynamics.com



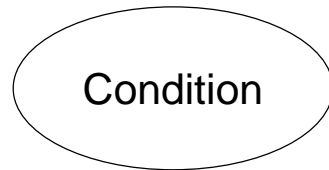
toyota.com

State charts vs. behavior trees (BT)



Behavior trees: a primer

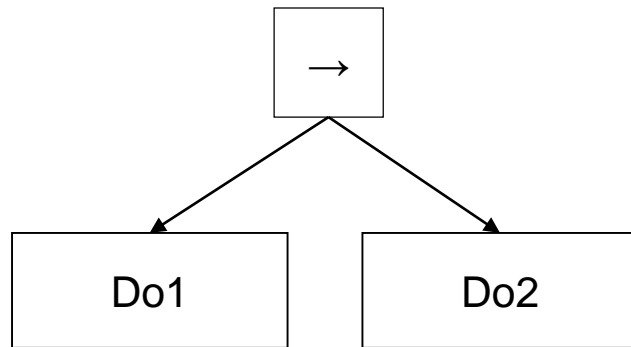
Condition



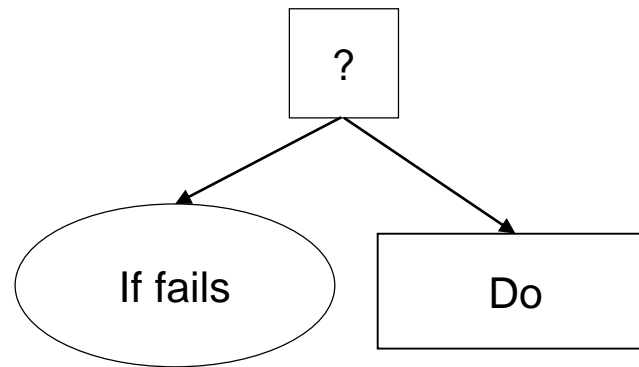
Action



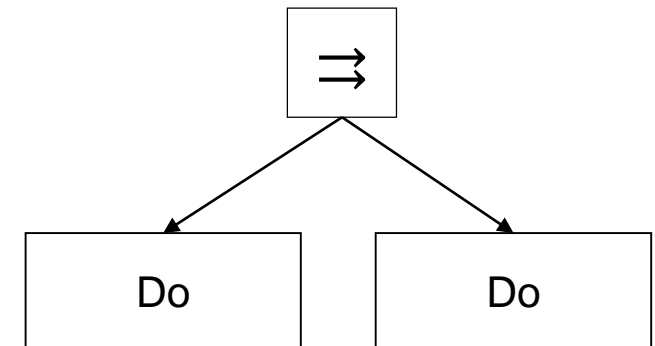
Sequence



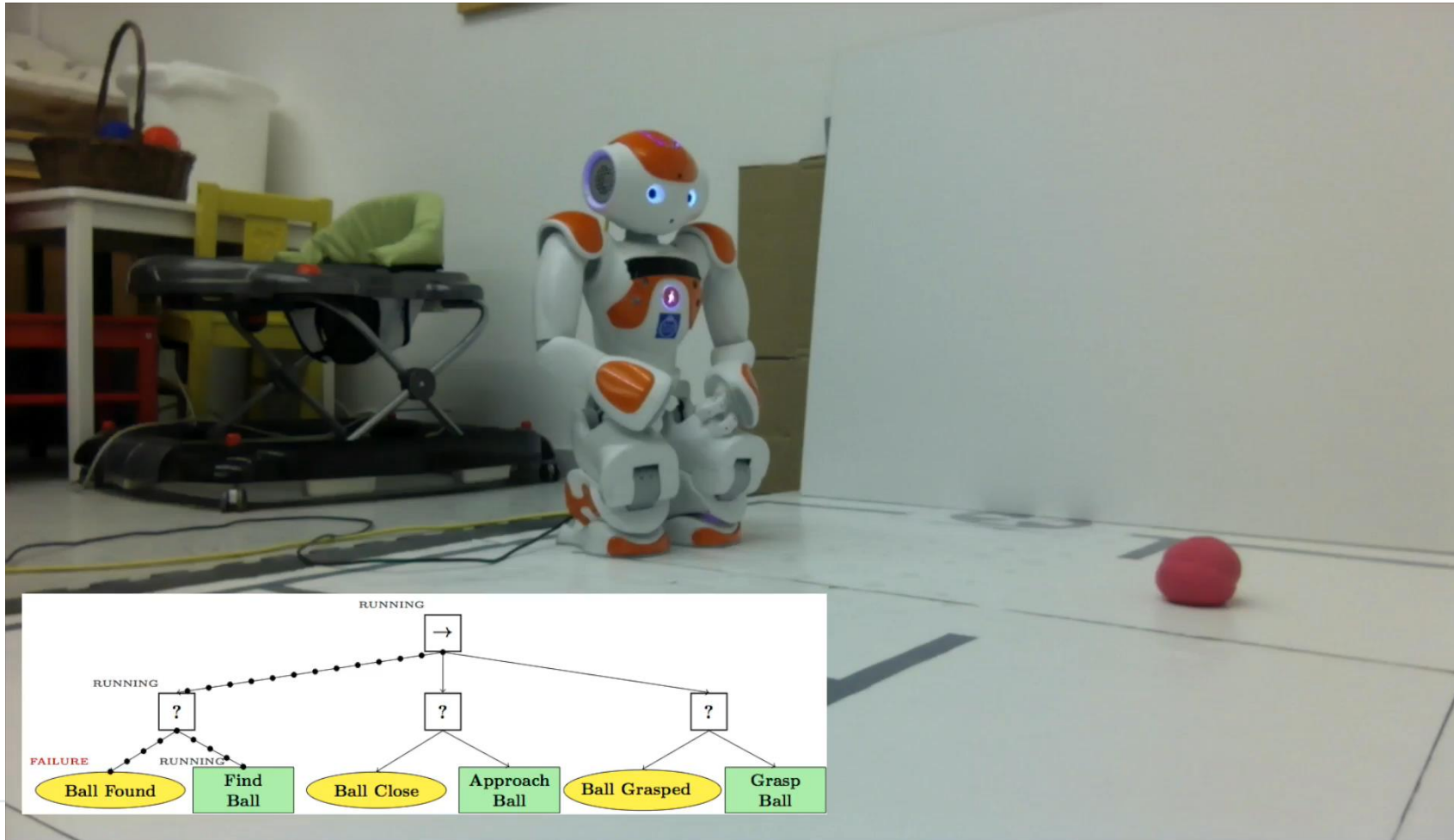
Fallback

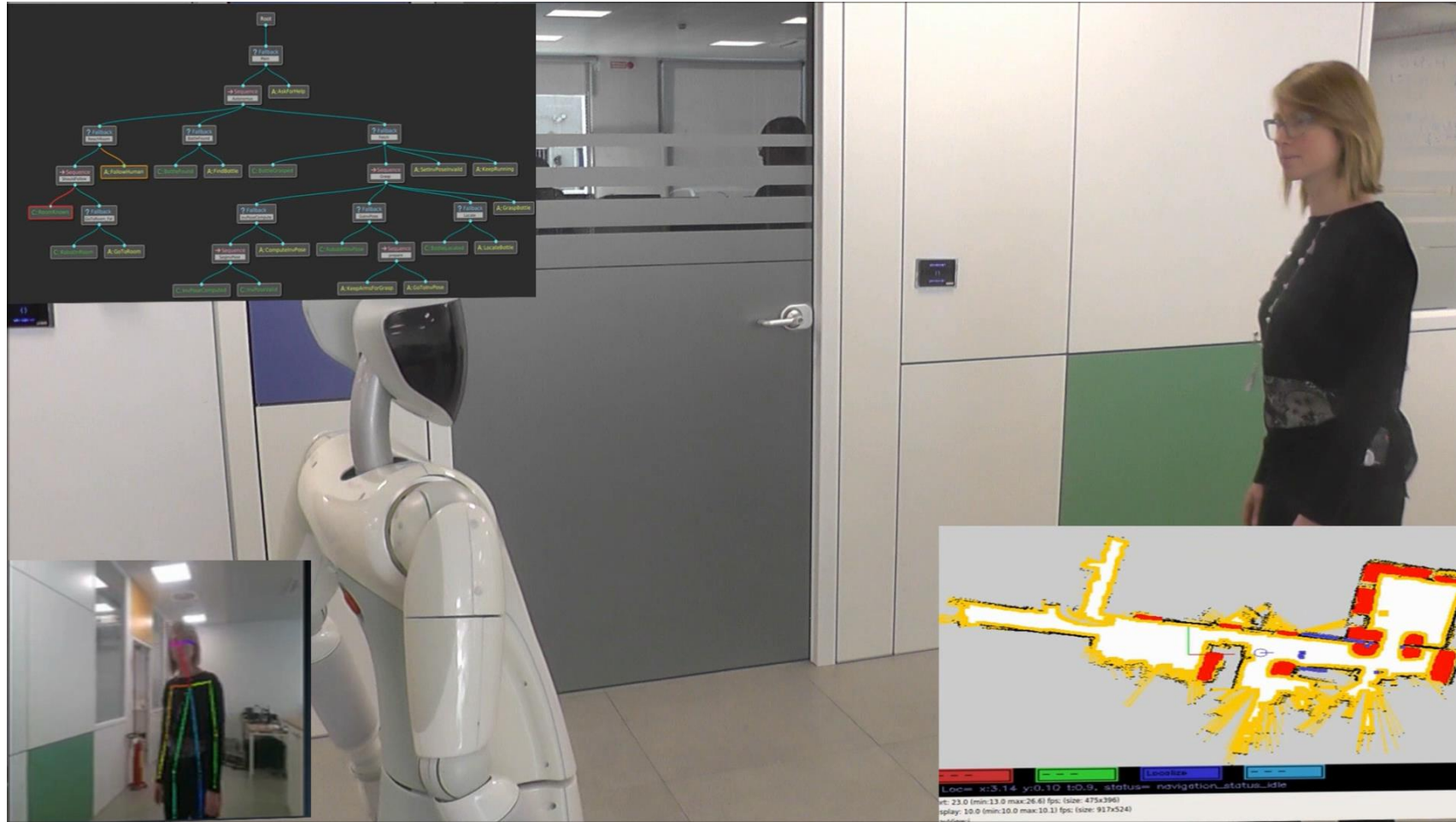


Parallel



Reactive behaviors: a simple example

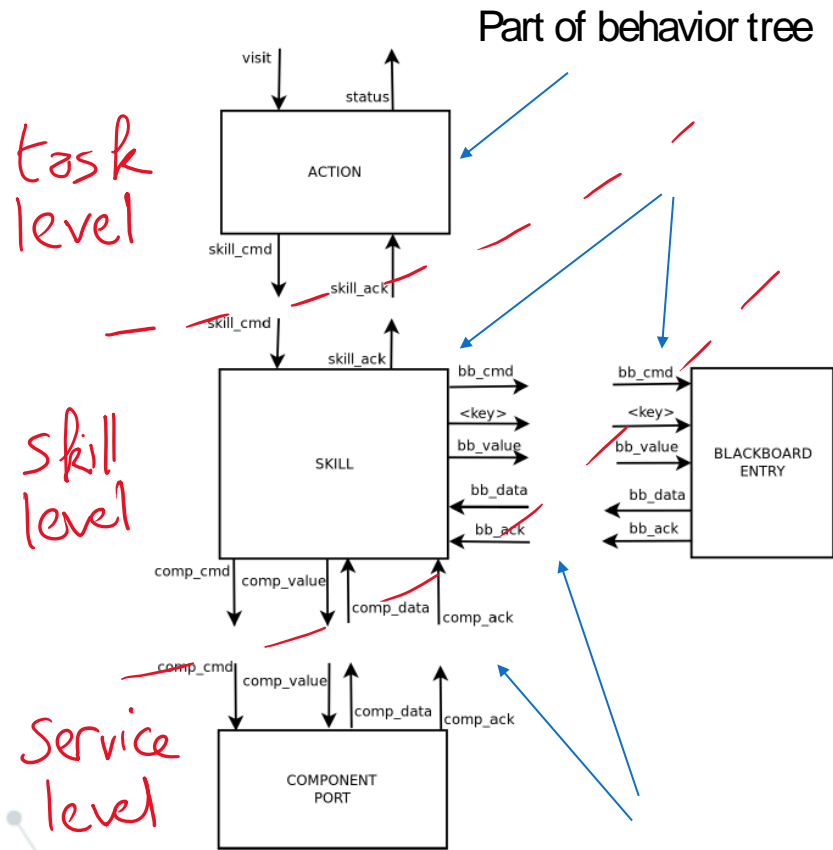




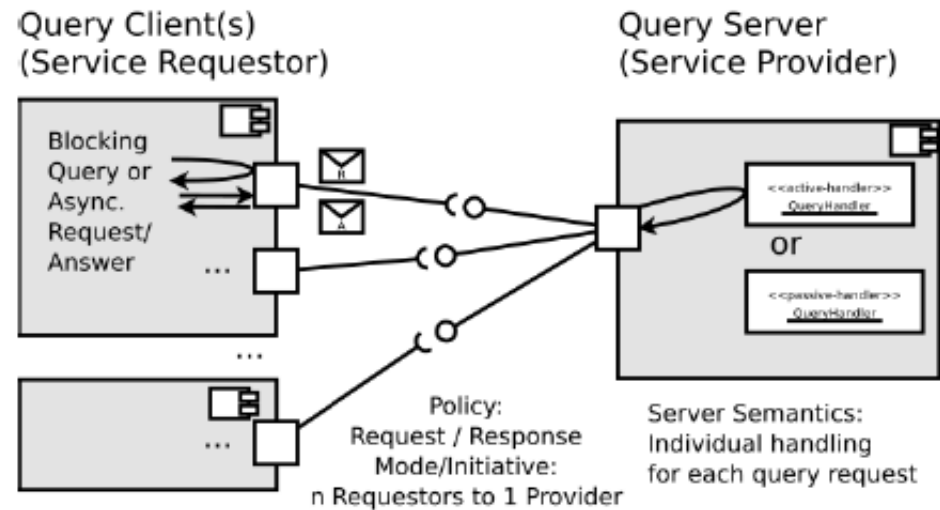
Colledanchise, et al., *Formalizing the Execution Context of Behavior Trees for Runtime Verification of Deliberative Policies*, IROS 2021

Colledanchise and Natale, *On the Implementation of Behavior Trees in Robotics*, IEEE Robotics and Automation Letters, 2021

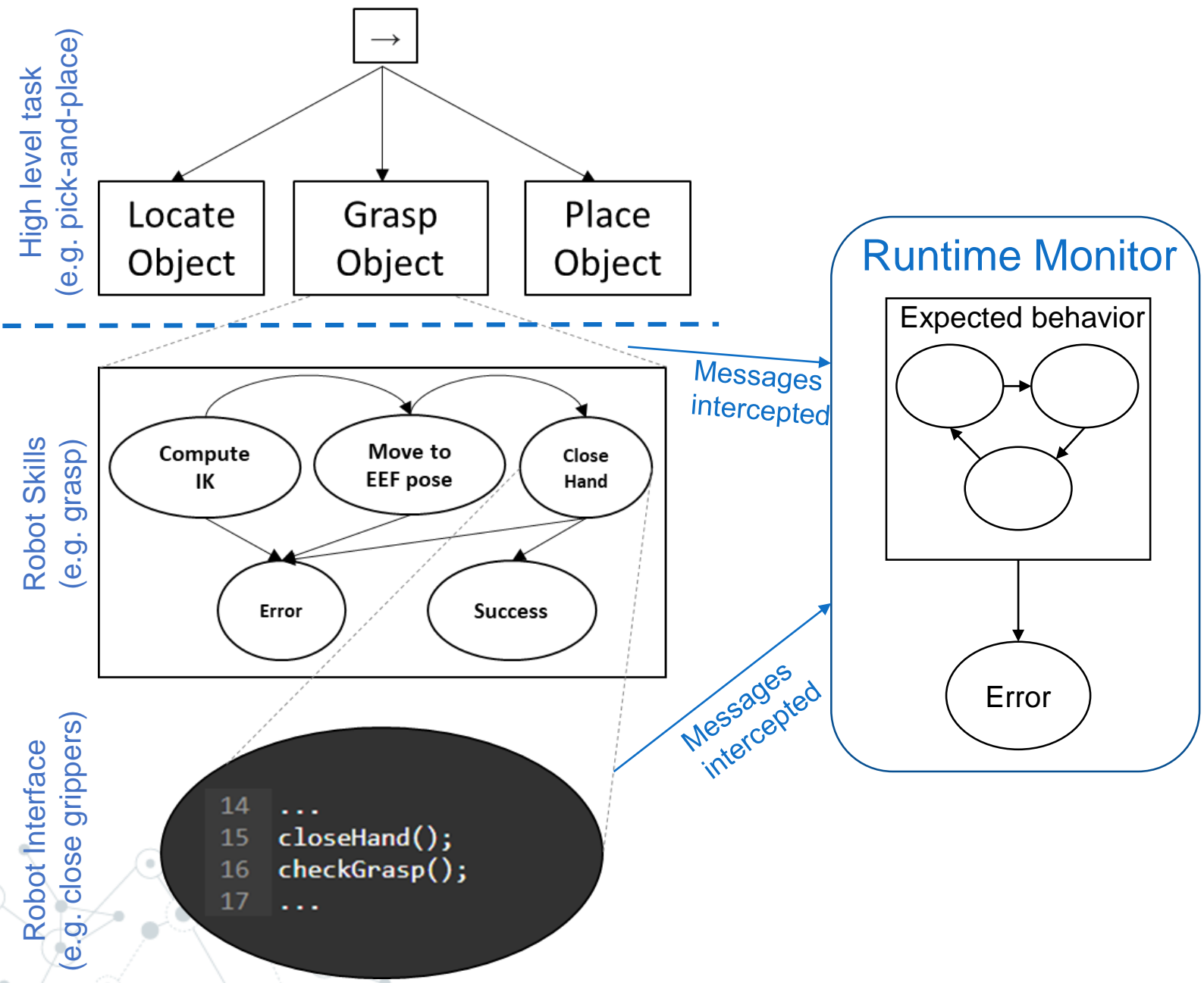
Semantics of BT + Skills + Components



- BT, skills and components modelled as communicating transition systems - **asynchronous** execution (threads)
- Properties specified with SCOPE language (OTHELLO subset)



- Communication follows the **Query Pattern**
- Interfaces are specified using an **interface definition language**



- “Sniff” the messages passed across layers.
- Intercept message by a *runtime monitor*
- A runtime monitor detects differences between the expected behaviors and the actual one

A robotic museum guide



What's needed:

- Dialog management
- Human-detection
- Self-localization
- Navigation

Cloud connectivity:

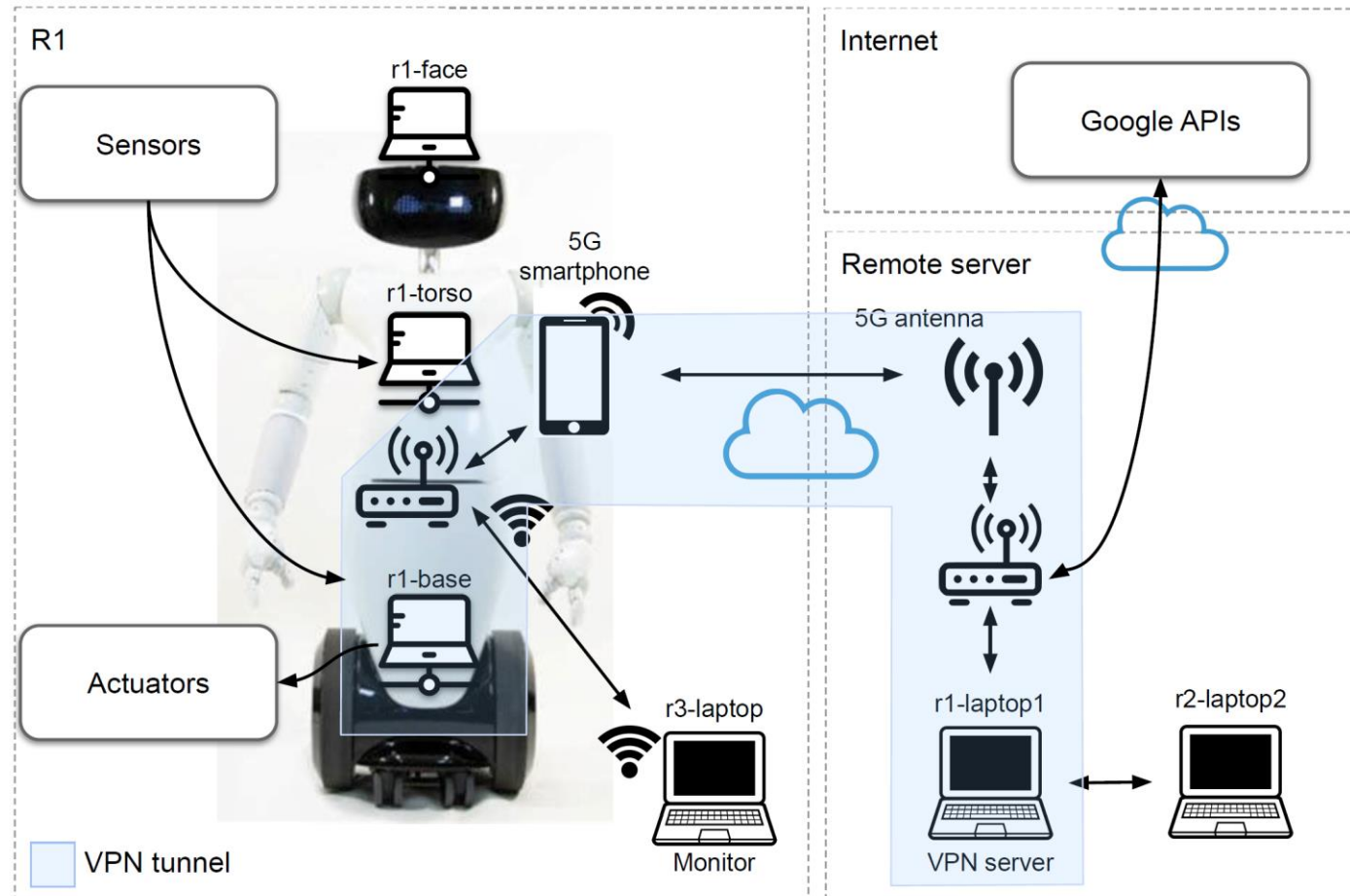
- Through 5G

How long:

- 200 meters, 20+ minutes (70 with questions)
- 110+ tours in two weeks



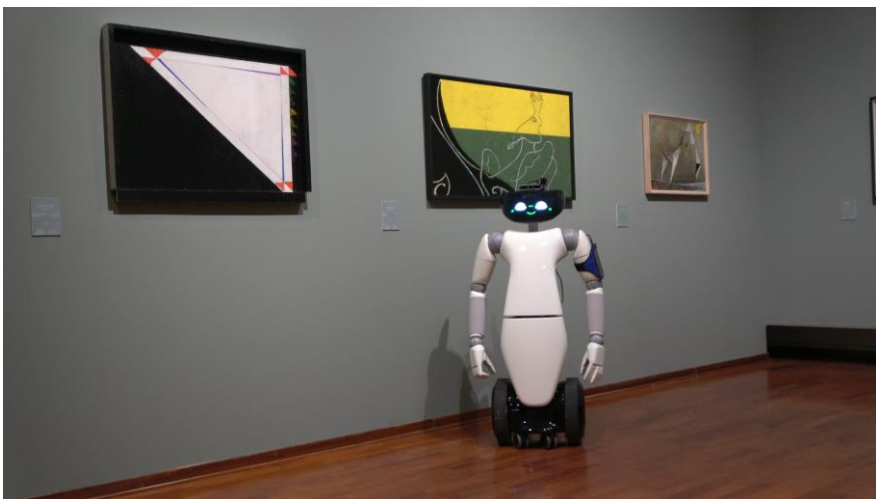
Hardware architecture





Software “tricks”

Galleria Arte Moderna, Turin



Palazzo Madama, Turin



- **Port monitors** to implement data compression: images and LIDAR over 5G
- **Behavior trees** to implement the behavior coordination as shown earlier
- **Multiple middleware** systems: ROS for navigation, YARP to control the robot, Google APIs for speech, etc.
- **Flexible plug-ins** and remotization to handle distributed processing with controlled latency

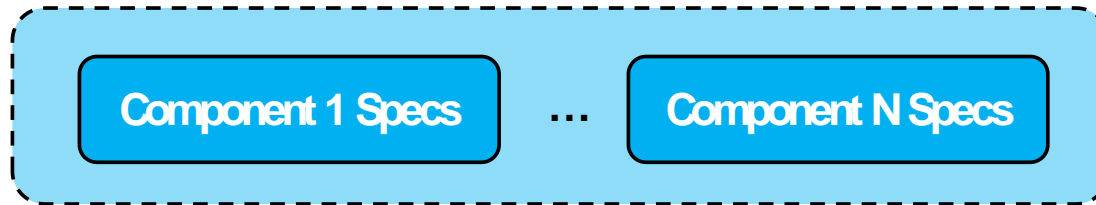


Below **YARR**

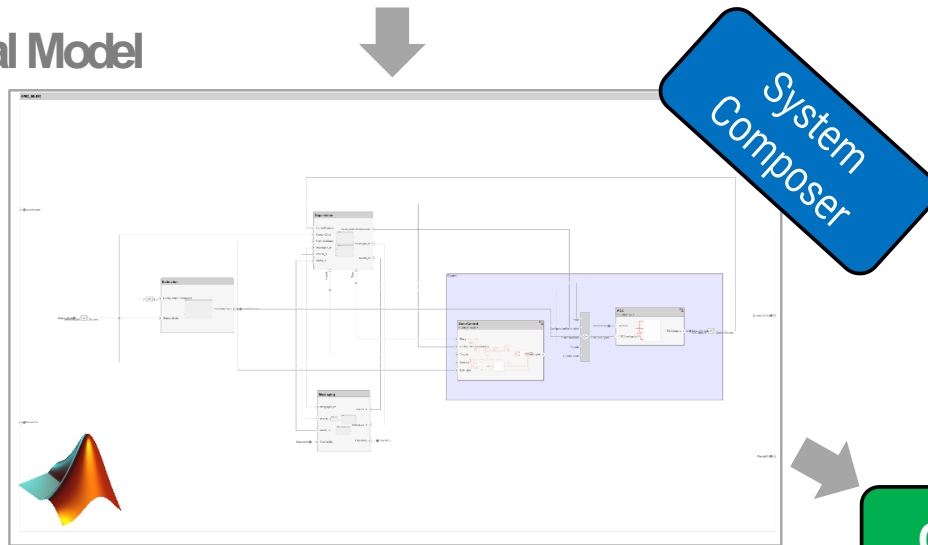
Model-Based System Engineering (MBSE)

MBSE = MBD + System Engineering

System Specs



Architectural Model



System Integration

Code

Code

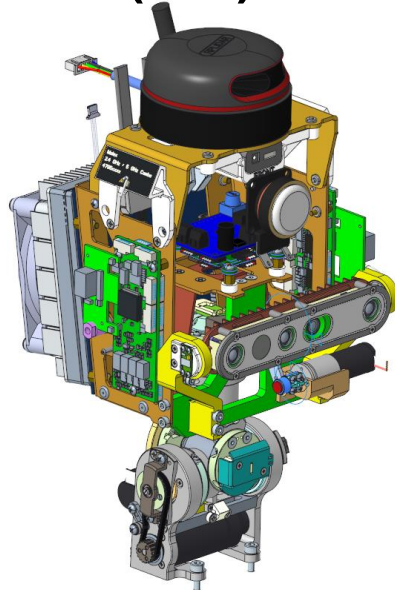
RTOS

- Complex Systems
 - Hierarchical components
 - Functional, logical, physical decompositions
 - Catch errors early, minimize rework
- Standardization
 - Data dictionaries for I/F's
 - Ports and connections
- Design Optimization
 - Static analysis
- Effective Communication
 - Implementable descriptions
 - Requirements

From CAD design to realistic simulations

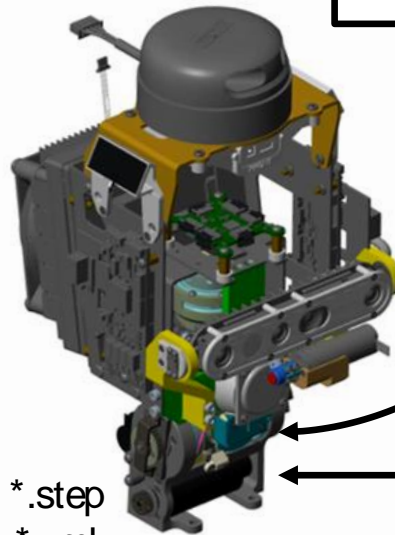
Simulink/Simscape

CAD (Creo)



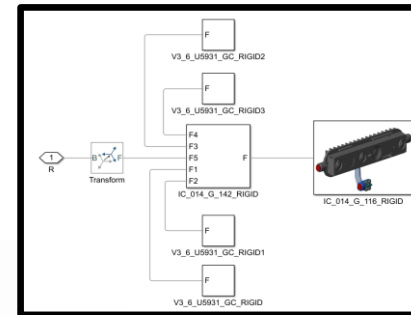
Simscape
Multibody
Link

Simulink

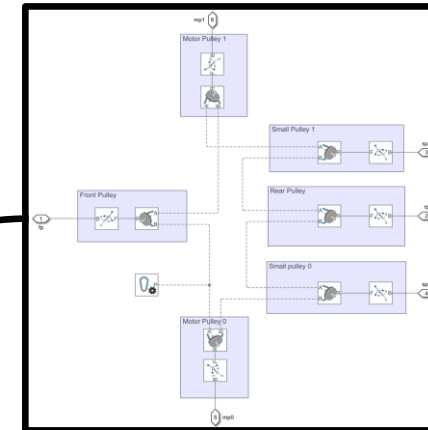


*.step
*.xml

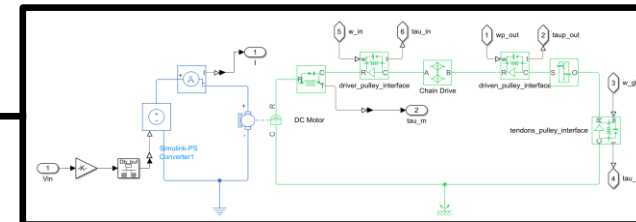
Assemblies



Tendons circuit



Transmission



JAXsim

A scalable physics engine for robot learning
implemented in pure Python with JAX.

Diego Ferigo, Silvio Traversaro, Daniele Pucci

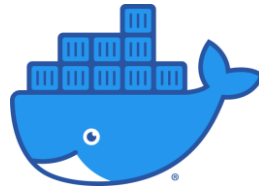
 [ami-iit/jaxsim](https://github.com/ami-iit/jaxsim)





Above **YARP**

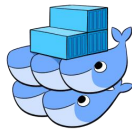
Collaborative software & the robot apps



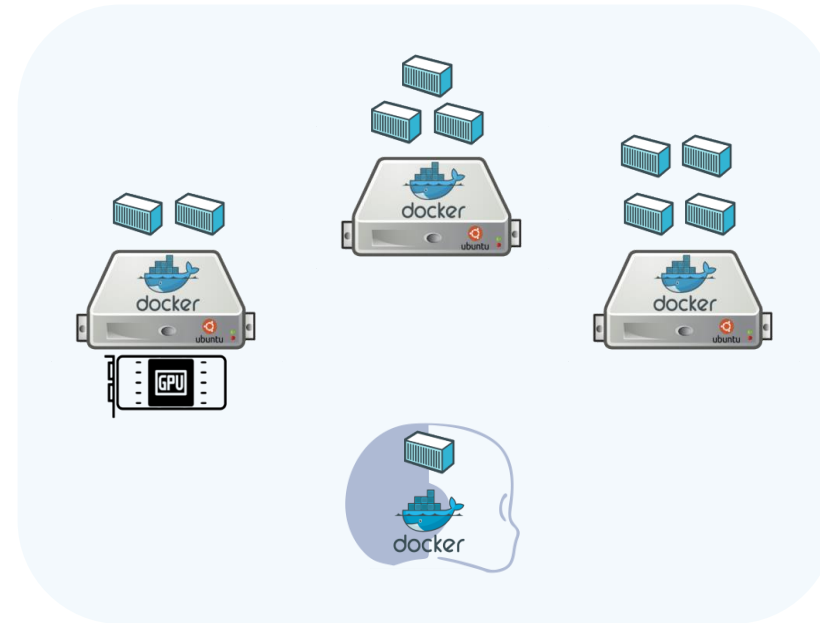
Docker



Docker Compose



Docker Swarm



Community hub

How can we help?
Welcome to the robotology's Knowledge-Base page

Software Documentation

- [Yet Another Robot Platform - YARP](#)
YARP is a library and toolkit for communication and device interfaces, used on everything from humanoid robots to embedded devices.
- [The iCub Main Software Repository](#)
This repository contains all the source code for the iCub platform.
- [Tutorials on iCub code](#)
This repository contains all the necessary tutorials for the iCub platform.
- [Meta Repository robotology-superbuild](#)
This is a meta repository to automatically download and compile software developed in the robotology GitHub organization.

Hardware Documentation

- [The iCub Mechanical Documentation](#)
The iCub mechanical documentation for collecting resources on the iCub robot.
- [The iCub Version Table](#)
This page contains all the information currently known iCub versions.
- [The iCub Cabling Documents](#)
This page contains all the information for iCub cabling procedures.
- [The iCub Upgrade Kits](#)
This page contains information on Upgrade Kits.

Welcome to the robotology Announcements
The iCub Tech announcements section aims to provide you with all the latest information/releases

- [YCM 0.11.4 Released](#)
September 2020
- [Distro 2020.05.feature-01 Released](#)
August 2020
- [YARP 3.0 Released](#)
August 2020

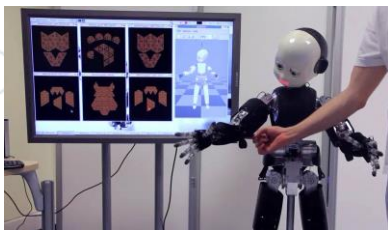
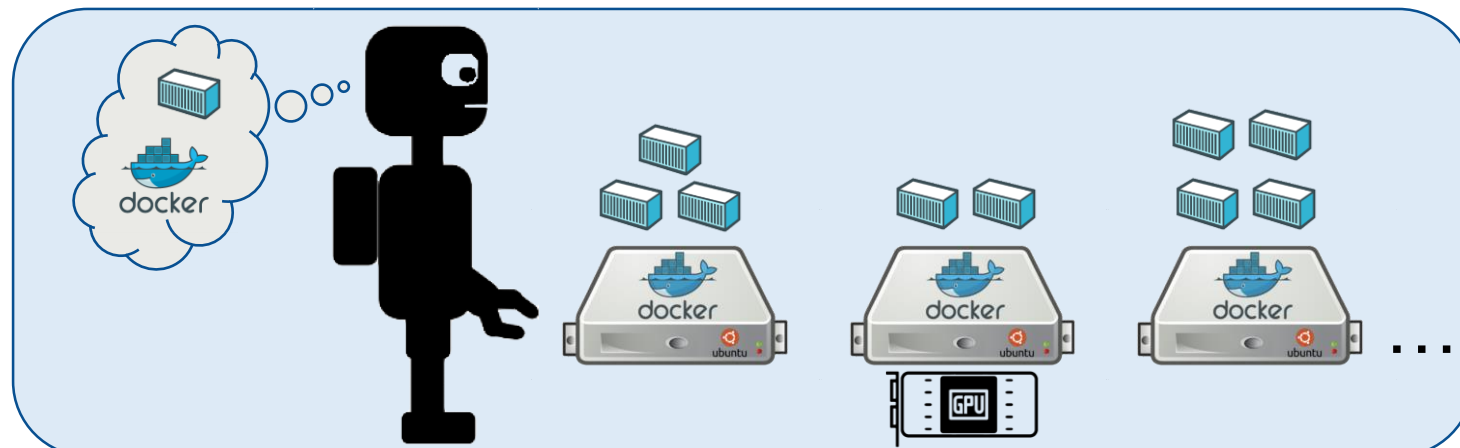
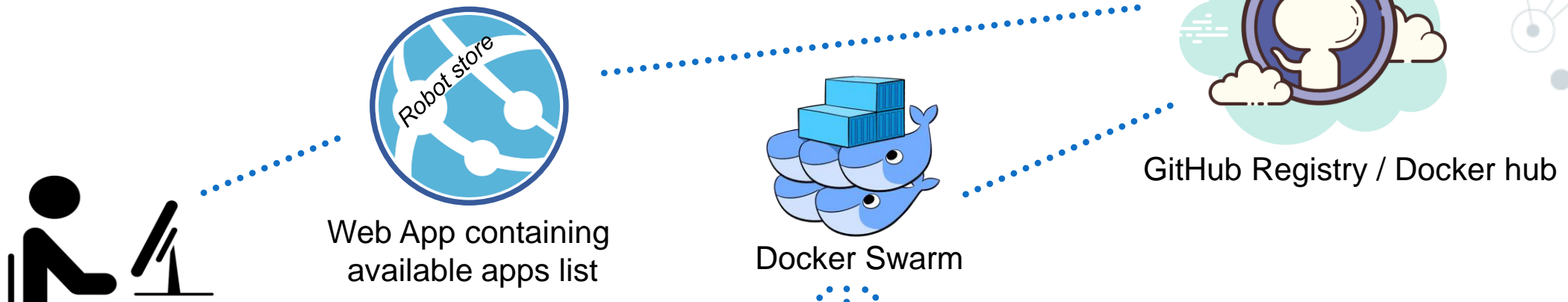
Applications

| Application | Updated | START |
|-------------------------------|-------------------|-------|
| Basic Docker Deployment | Nov 2019 | START |
| Gazebo Simulation Deployment | May 2020 | START |
| yarprobotinterface Deployment | Nov 2019 | START |
| Face and Gaze Tracking | Aug 2020 | START |
| Gazebo Grasping Sandbox | Aug 2020 | START |
| Grasp the Ball Application | Nov 2019 | START |
| Human Sensing | Jul 2020 | START |
| Interactive Object Learning | Under development | START |
| Navigation applica' | Under developme | STAF |

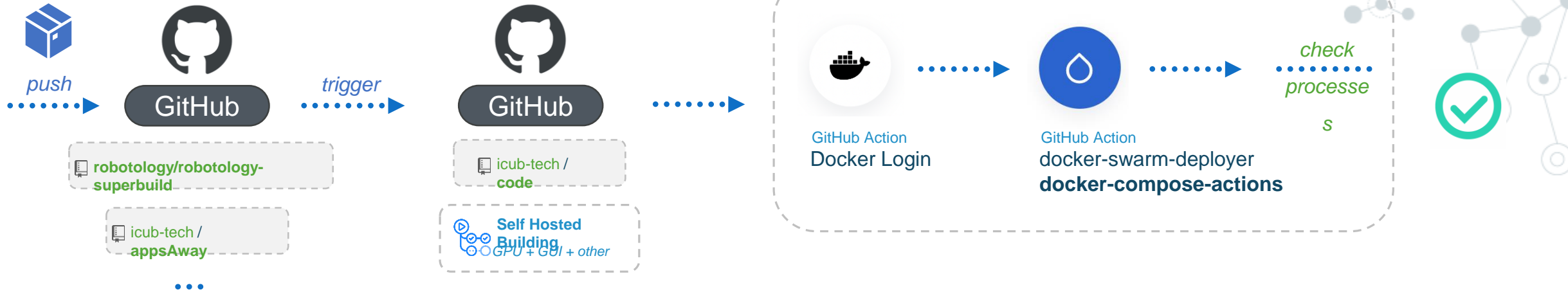
Where are all the robots?

| Robot Model | Count |
|----------------------|-------|
| iCub Humanoid Robot | 41 |
| R1 Humanoid Robot | 2 |
| iCub3 Humanoid Robot | 1 |

Development steps



Automatic building & testing



| | | | |
|--|---|------|-----|
| <input type="checkbox"/> icub-GPU2-runner1 | self-hosted Linux X64 code silo-sw | Idle | ... |
| <input type="checkbox"/> icub-GPU2-runner2 | self-hosted Linux X64 code silo-sw | Idle | ... |
| <input type="checkbox"/> icub-GPU2-runner3 | self-hosted Linux X64 code silo-sw | Idle | ... |
| <input type="checkbox"/> icub-GPU2-runner4 | self-hosted Linux X64 code silo-sw | Idle | ... |
| <input type="checkbox"/> icub-GPU2-runner5 | self-hosted Linux X64 code silo-sw test-rig | Idle | ... |
| <input type="checkbox"/> icub-GPU2-runner6 | self-hosted Linux X64 code-cleanup | Idle | ... |

app_testing_from_appsaway app_testing #162

Triggered via repository dispatch 8 minutes ago

Status: Success Total duration: 2m 13s Artifacts: -

Jobs:

- printing_app_list
- running_test_script (yarpBasicDeploy)

triggering_app_test.yml

on: repository_dispatch

Matrix: running_test_script

printing_app_list 0s → 1 job completed

Show all jobs

giorgio.metta@iit.it

iit

ISTITUTO