ROS 2:
Etat des lieux

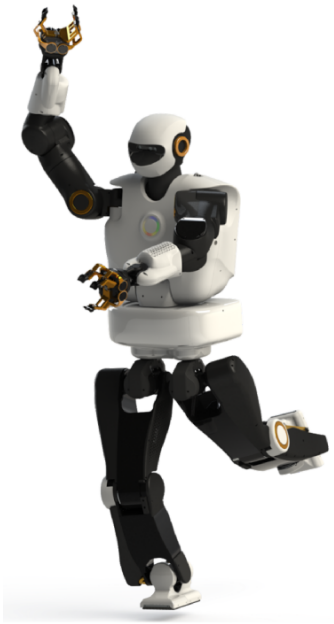
thanks to Deanna Hood, William Woodall - ROSCON 2016

29 Septembre 2017, Lille, 2RM, France

O. Stasse, Gepetto,
LAAS-CNRS

Table of Contents

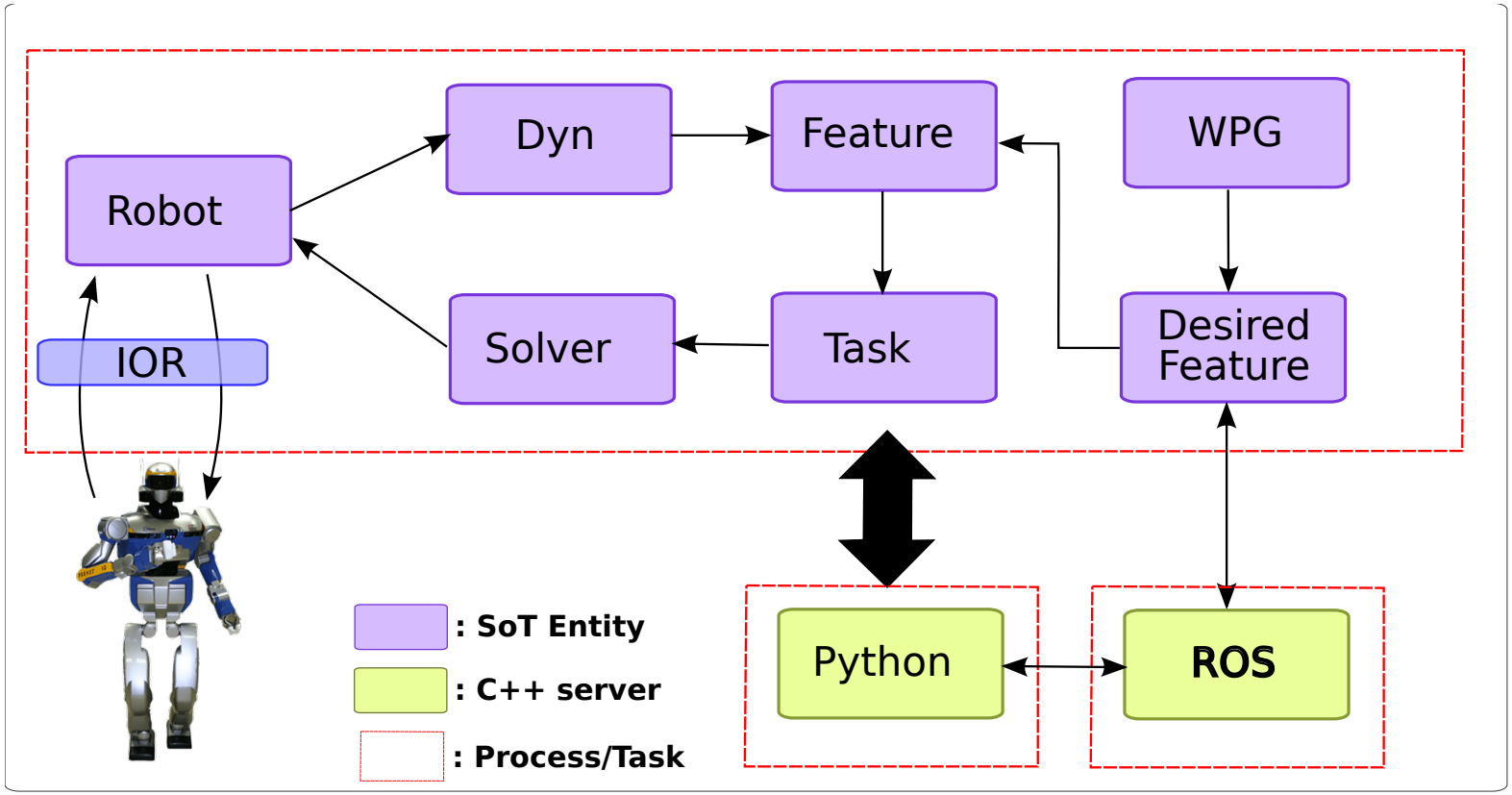
ROS applications



- Research applications
- High volume sensors
- Complex kinematics
- Lots of computation power
- Ideal network connectivity

- Multi robot system
- Small processors
- Battery power
- Unreliable network connectivity

Our software architecture



Goals of ROS 2

Supporting multi-robot systems involving unreliable networks

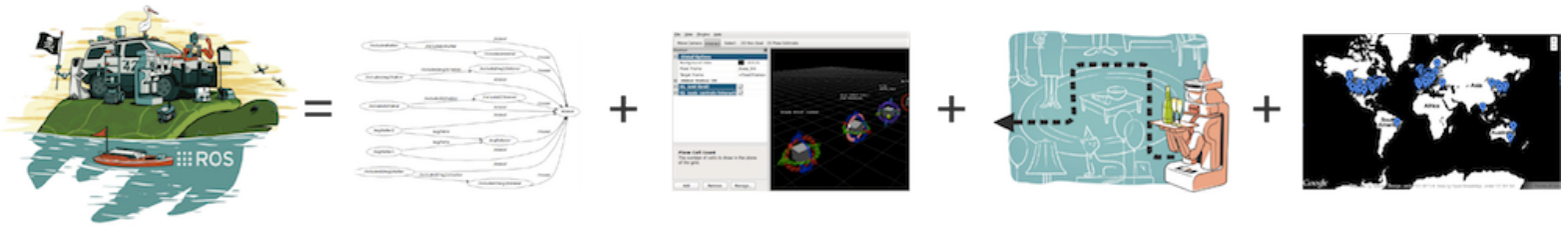
Remove the gap between prototyping and final products

Support Multiple OS

Support for real time control

Bare metal micro controller

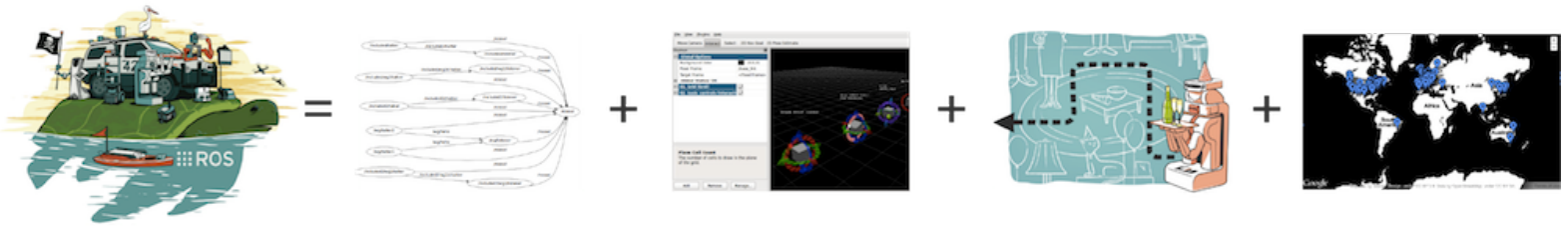
ROS equation



Plomberie Outils Capacités Ecosystème

ROS equation

DDS



Plomberie + Outils + Capacités + Ecosystème

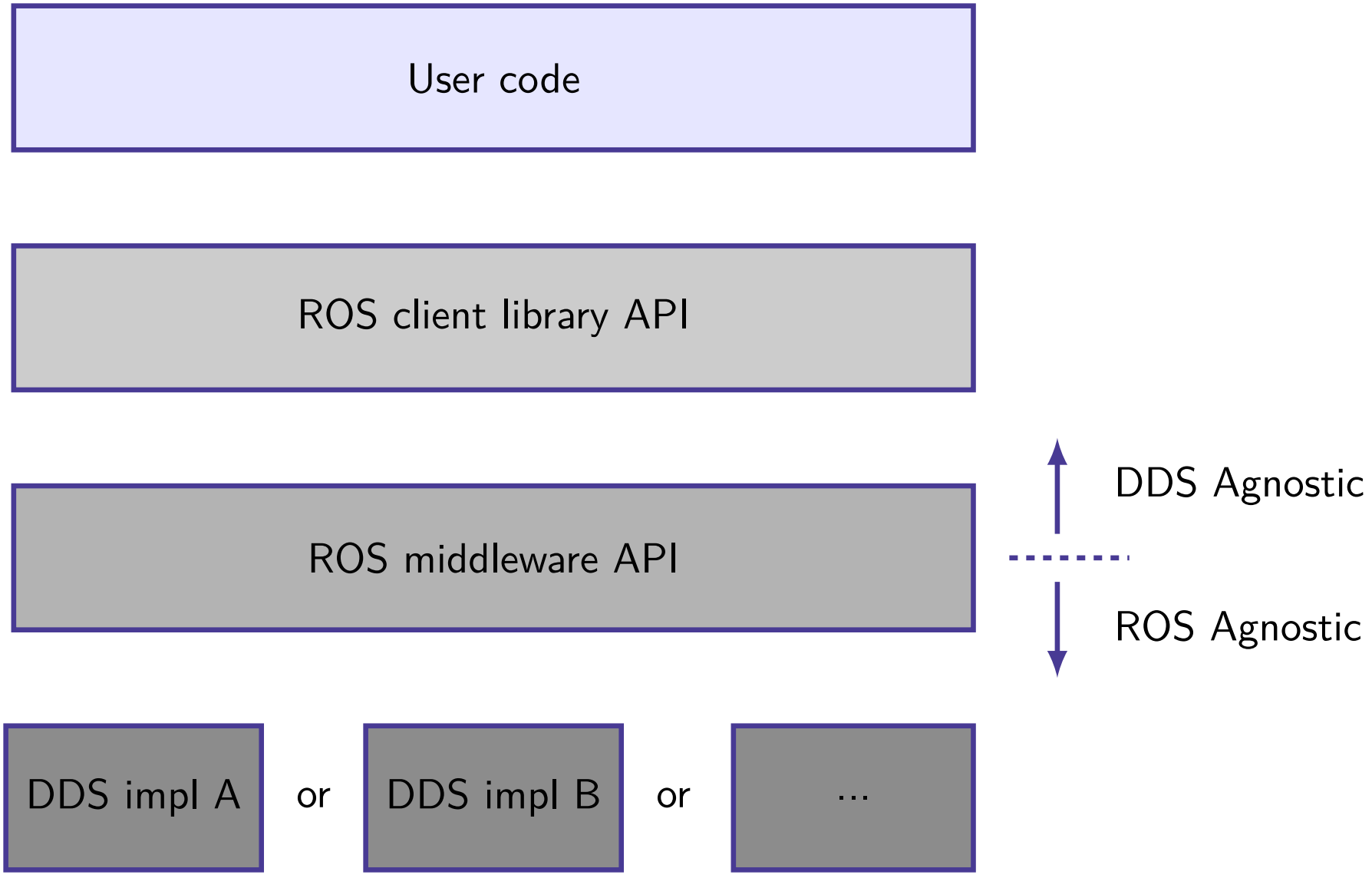
Main Features

- Multiple DDS support
- Multiple Operating Systems support: Linux, OS, Windows
- Multiple programming languages
- Co-existence with ROS-1 systems
- Release Beta version every two months

WARNING !

Beta-release
Code Unstable

Architectural overview



DDS: Data Distribution Service

- OMG Standard
- No master
- First version 1.0 in December 2004 (freely available)
- Last version 1.4 in April 2015 (freely available)
- Focus on real-time data-flow applications
- Data-Centric Publish-Subscribe (DCPS) model
- 9 vendors (among which OCI and eProsima providing a free implementation)

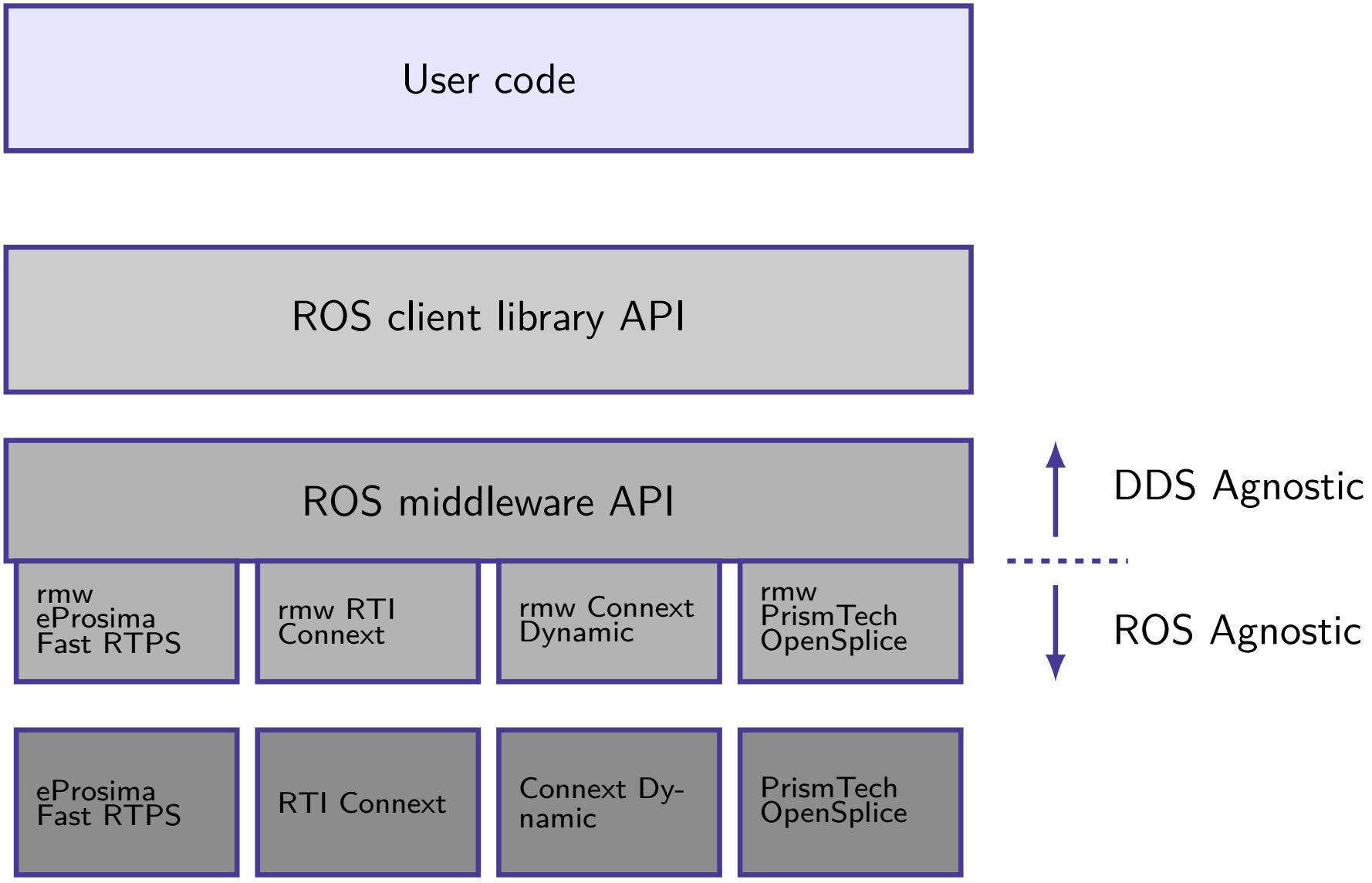
DDS has been used in

- battleships
- large utility installations like dams
- financial systems
- space systems
- flight systems
- train switchboard systems

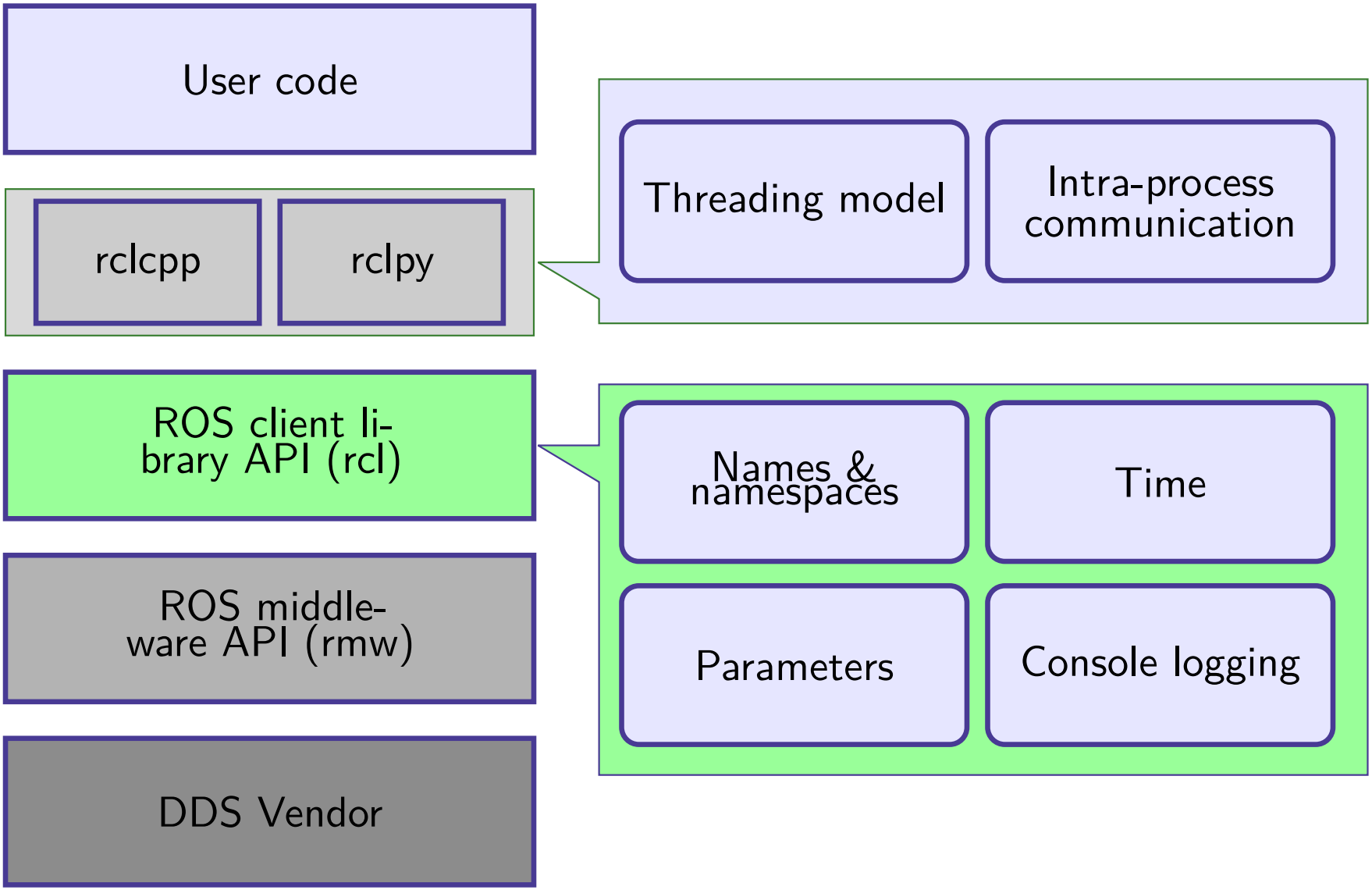
DDS Features

- A Quality of Services definition of data transmission through topics.
- The possibility to define objects such as DataWriter and DataReader is allowing a fine policy for publishing and subscribing.
- Real-Time Publish Subscribe (RTPS)
- Several of the DDS vendors have special implementations of DDS for embedded systems which boast specs related to library size and memory footprint on the scale of tens or hundreds of kilobytes.
- No Remote Procedure Call services.

Architectural overview



ROS client libraries: rcl



ROS 2 Features

Discovery, transport and serialization over DDS	Article	
Support for multiple DDS implementations, chosen at runtime	Tutorial	Currently eProsima Fast RTPS and RTI Connext are fully supported
Common core client library that is wrapped by language-specific libraries	Tutorial	
Publish/subscribe over topics	Sample code	Namespacing limitation for Connext

ROS 2 Features

Clients and services	Sample code	Python support experimental
Set/retrieve parameters	Sample code	Parameters not yet available in rcl/Python
ROS 1 - ROS 2 communication bridge	Tutorial	Only for topics and services, not yet available for actions
Quality of service settings for handling non-ideal networks	Demo	

ROS 2 Features

Inter- and intra-process communication using the same API	Demo	Currently only in C++
Composition of node components at compile-, link- or dlopen-time	Demo	Currently only in C++
Support for nodes with managed lifecycles	Demo	Currently only in C++
DDS-Security support	Demo	

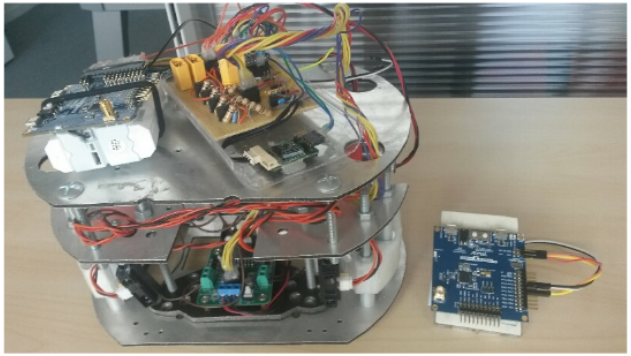
ROS 2 Features

Command-line introspection tools using an extensible framework	Tutorial	
Namespace support for nodes and topics	Article	
Demos of an all-ROS 2 mobile robot	Demo	
Preliminary support for real-time code	Demo, demo	Linux only. Not available for Fast RTPS.
Preliminary support for "bare-metal" microcontrollers	wiki	

Example of ROS 2 on ARM-M0

Loïc Dauphin, Team Interface, INRIA

Core:	ARM M0
Microprocessor:	ATSAMR21G18A 32Kb RAM 256Kb Flash
OS:	RIOT
2 others network protocols:	NDN - MQTT
rcl implementation:	C



[Dauphin,ACM, ICN, 2017]

Example on Turtlebot 2

- Joystick teleop
- Follower
- Cartographer
- ACML Localization

Conclusion - Take away messages

- New middleware system: DDS
- Multi platform
- Better network QoS handling
- New building system catkin $\rightarrow$ ament (cament)
- New client library
- Node composability
- Potential higher programming complexity