Open Source Robotics Foundation

Next-generation ROS: Building on DDS

Sep 12th, 2014

Esteve Fernandez, Tully Foote, William Woodall, Dirk Thomas

ROSCon Chicago

What is ROS?

Where are we now?

- Maturity
- Robustness
- Community
- Openness
- Interoperability
- Modularity
- Federated development model
- Richness

Celebrating 5 year of ROS video




Jenkins

MoveIt!

pcl

Hundreds of contributors to Hydro

GAZEBO

MOOS

Light, Fast, Cross Platform Middleware for Robots

OpenCV

THE OROCOS PROJECT

Open Robot Control Software

EURON Interest Group

Modular Architecture

YARP

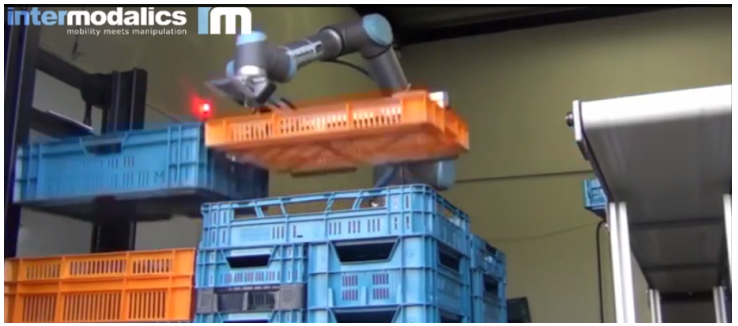
open source

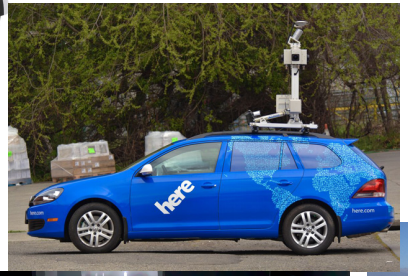
github SOCIAL CODING

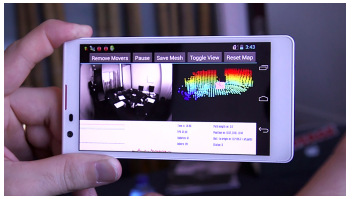
Open Source Robotics Foundation

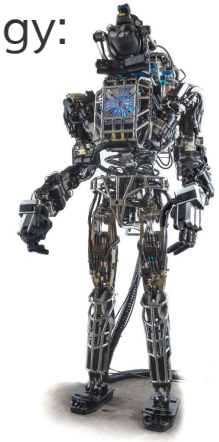
Where is ROS used?

- Rethink Robotics: Baxter
- Unbounded Robotics: UBR-1
- ROS-Industrial: (de)palletizing
- RightHand Robotics: ReFlex Hand
- Boston Dynamics: ATLAS
- PAL Robotics: REEM-C
- HERE: 3D mapping cars
- Google ATAP: Project Tango
- Avidbots: Sweeper and Scrubber
- Blue River Technology: Precision Farming
- ...









Open Source Robotics Foundation

How did we get there?

Ease of use

Flexibility

Scalability

Enabling reuse

Where do we want to go?

Use the OSRF resources for ROS to:

Maintain what we have
with only little improvements over time
in order to not break backward compatibility?

vs.

Drive new development
to address current and upcoming needs!

Address underserved use cases

support multi-robot systems
involving unreliable networks etc.

reduce the gap between
prototyping and final products

“bare-metal” micro controllers

(better integration with)
real-time control

Exploration and prototyping

Looking back:

- 7 years ago there were not many suitable libraries available

But times have changed...

While each addresses very specific parts
building ROS on-top of these still requires a huge effort.

Data Distribution Service (DDS)

- an industry-standard communication system

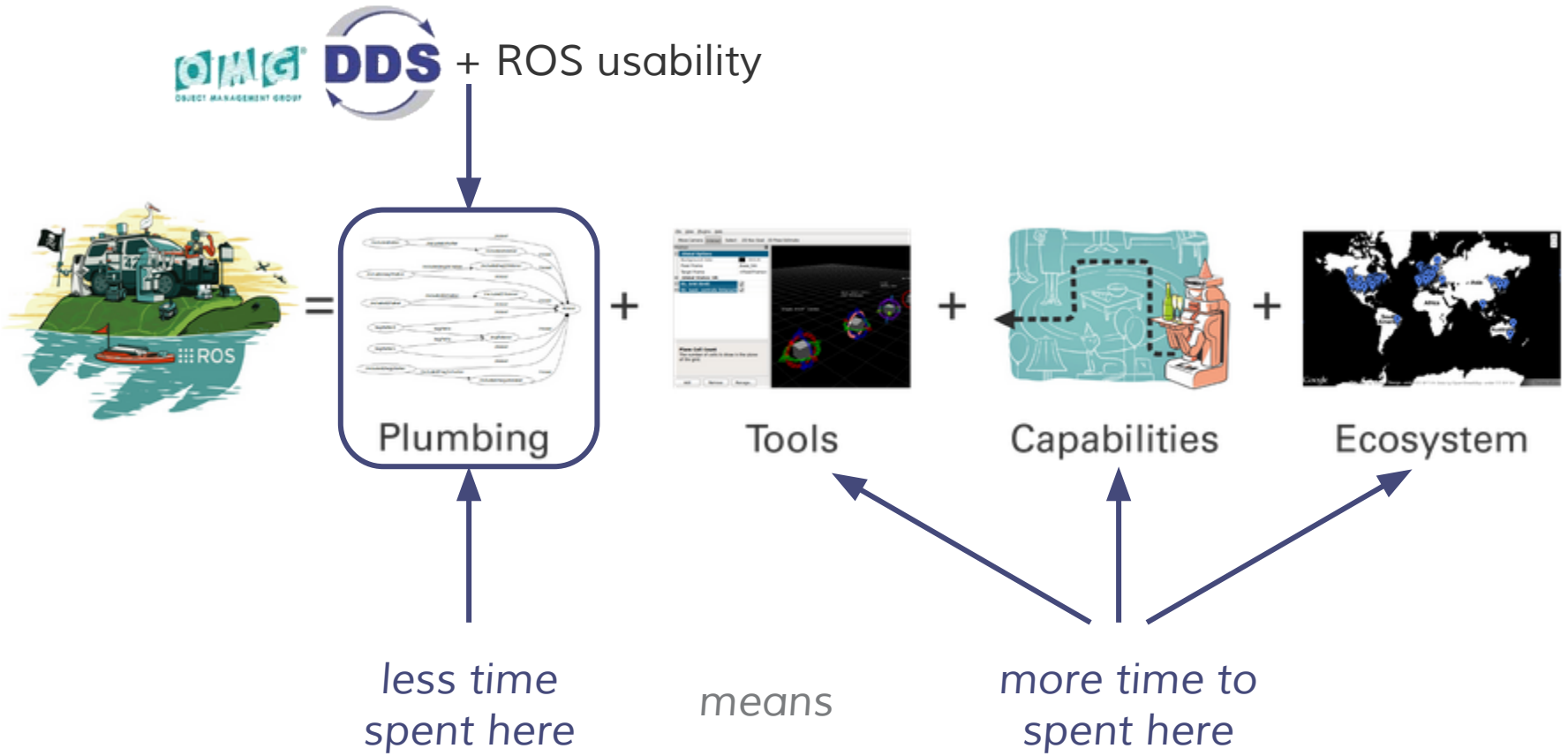
  

  = discovery + serialization + transport

- an active/evolving OMG standard
  - peer-to-peer middleware
  - configurable quality of service
    to handle many networking situations
  - real-time capable

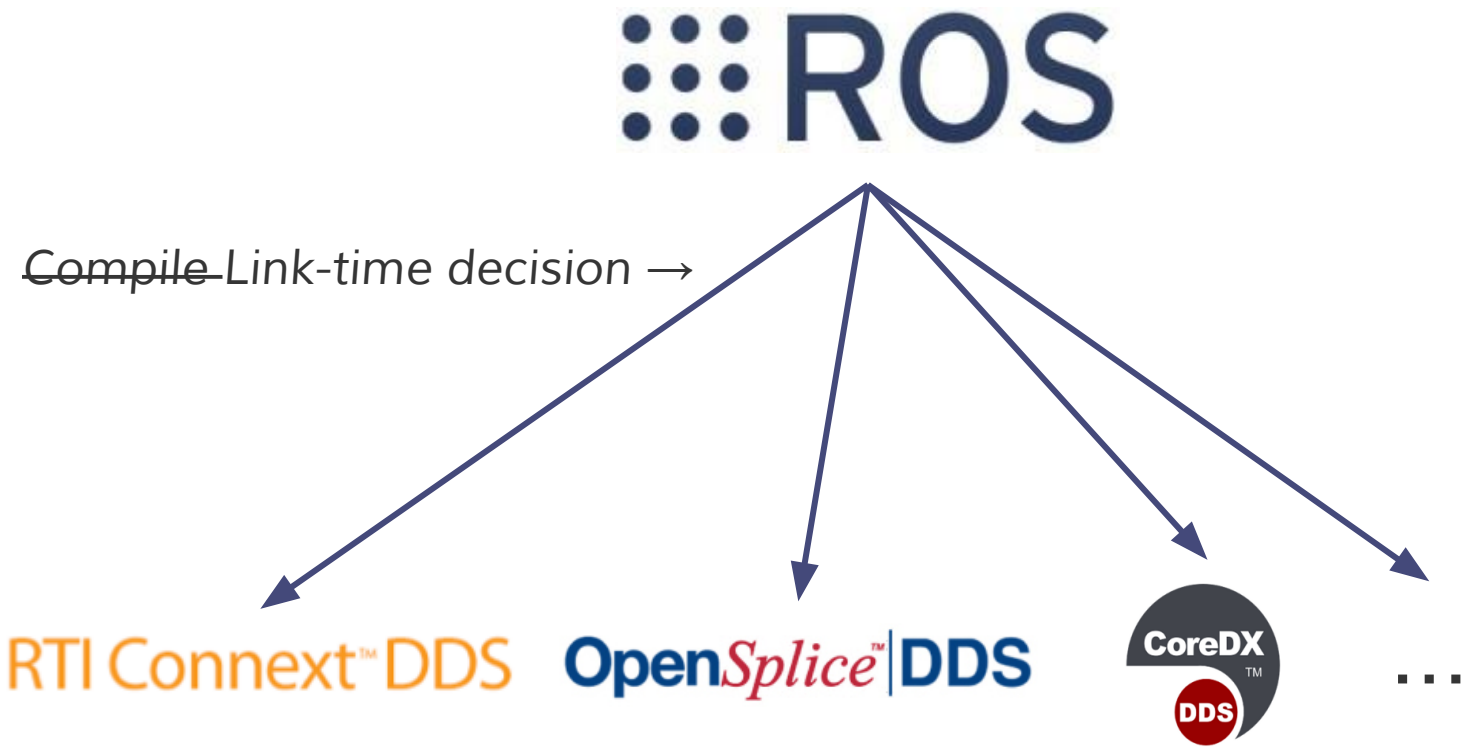
- Multiple implementations (~12)
  - *commercial* as well as *open source* implementations
  - proven in mission-critical environments (trains, dams, ship, etc.)
  - some are NASA- / DOD-verified
  - some small / embedded solutions

ROS 2 - Built on DDS

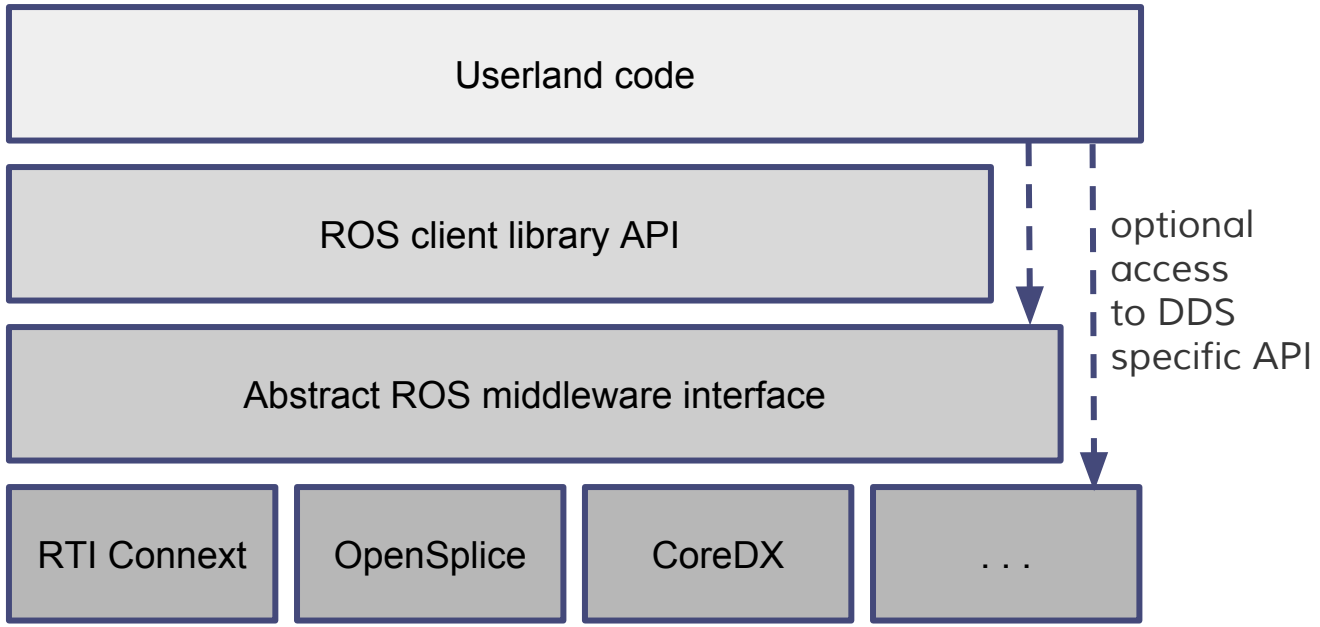


No vendor lock-in

Use your favorite DDS implementation

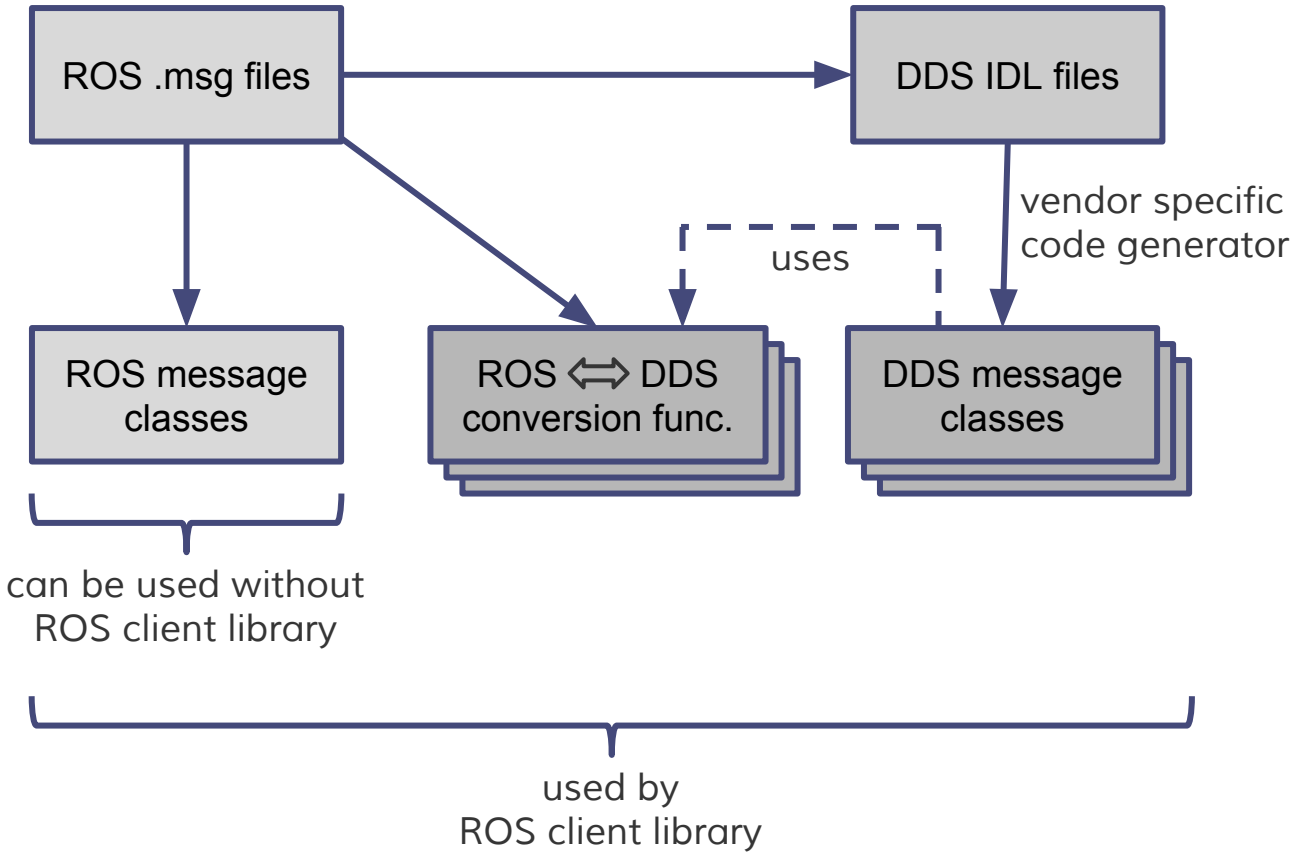


Abstract ROS middleware interface

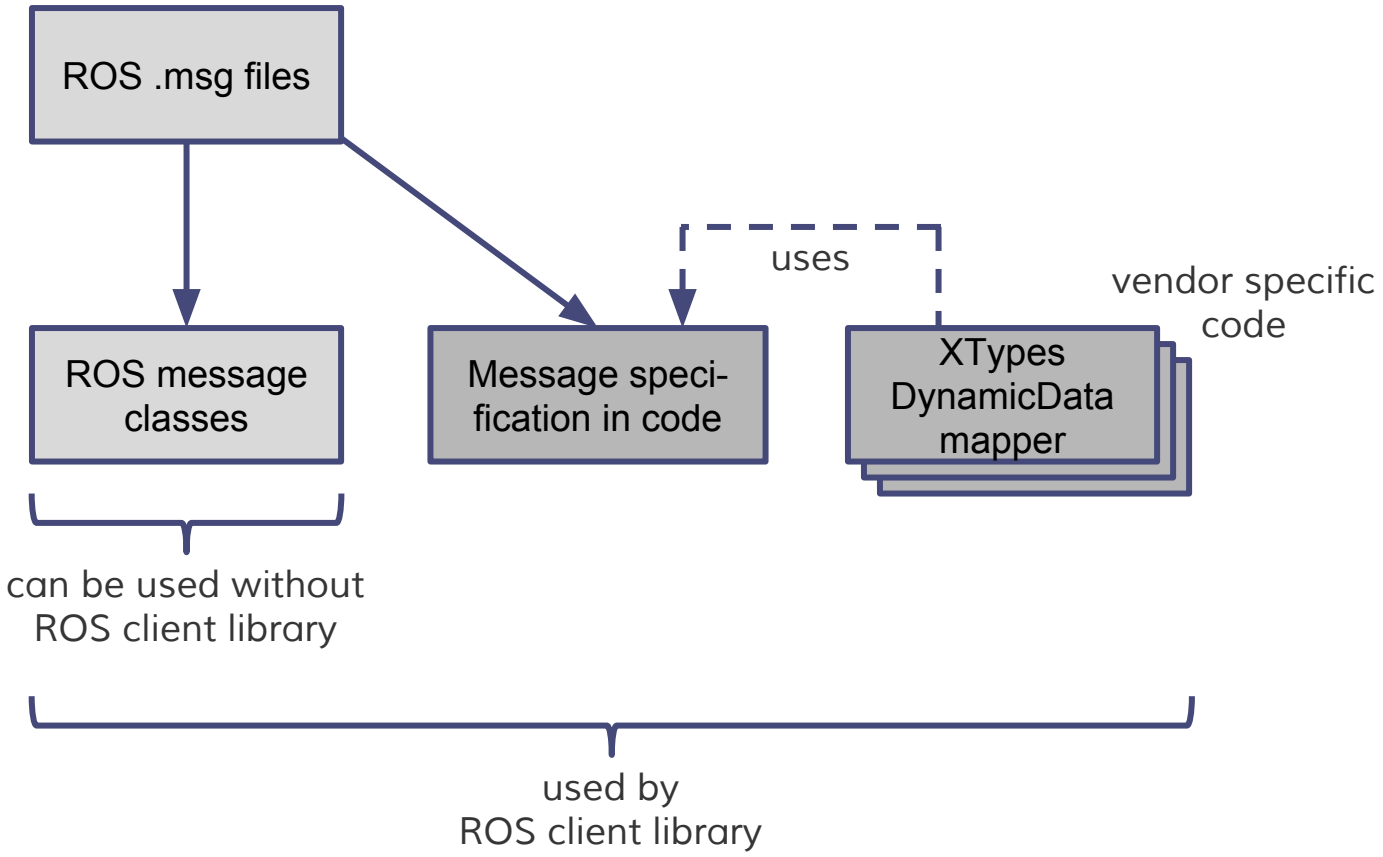


for more details see:
http://design.ros2.org/articles/ros_middleware_interface.html

ROS Messages
Static code generation



ROS Messages
XTypes DynamicData



Package resource index

Looking up packages or plugins in ROS 1 requires crawling the filesystem

- conceptionally expensive
- caching only provide some level of improvement
  - trade-off between being outdated vs. recrawling to often

Goal: “answer common questions in constant time”

- which packages are available?
- where is the share-folder of package X?
- query a list of rviz plugins?

Package do require a build step anyway

- Shift the work into the build phase
  - The necessary information is added to an index
- Queries boil down to looking at a known location in the file system without requiring any crawling

for more details see:
https://github.com/ament/ament_cmake/blob/master/ament_cmake_index/README.md

Current prototype

- Talker / Listener demo
  - Arbitrary messages: all primitive types, fixed-size arrays, bounded/unbounded arrays, built-in types for Time / Duration, default values for primitives (yes, really)
- Working with the following DDS implementations:
  - RTI Connext (statically generated code & XTypes DynamicData)
  - PrismTech OpenSplice (statically generated code)
    - of course a talker of impl. A is interoperable with a listener of impl. B ;-)

- Experience so far
  - 1:N communication faster (*multicast!*)
  - for some vendors: localhost comm done through shared memory
  - optionally reliable message delivery
  - better reconnection behavior when dropping out of wireless

ROS 2 - What else to expect?

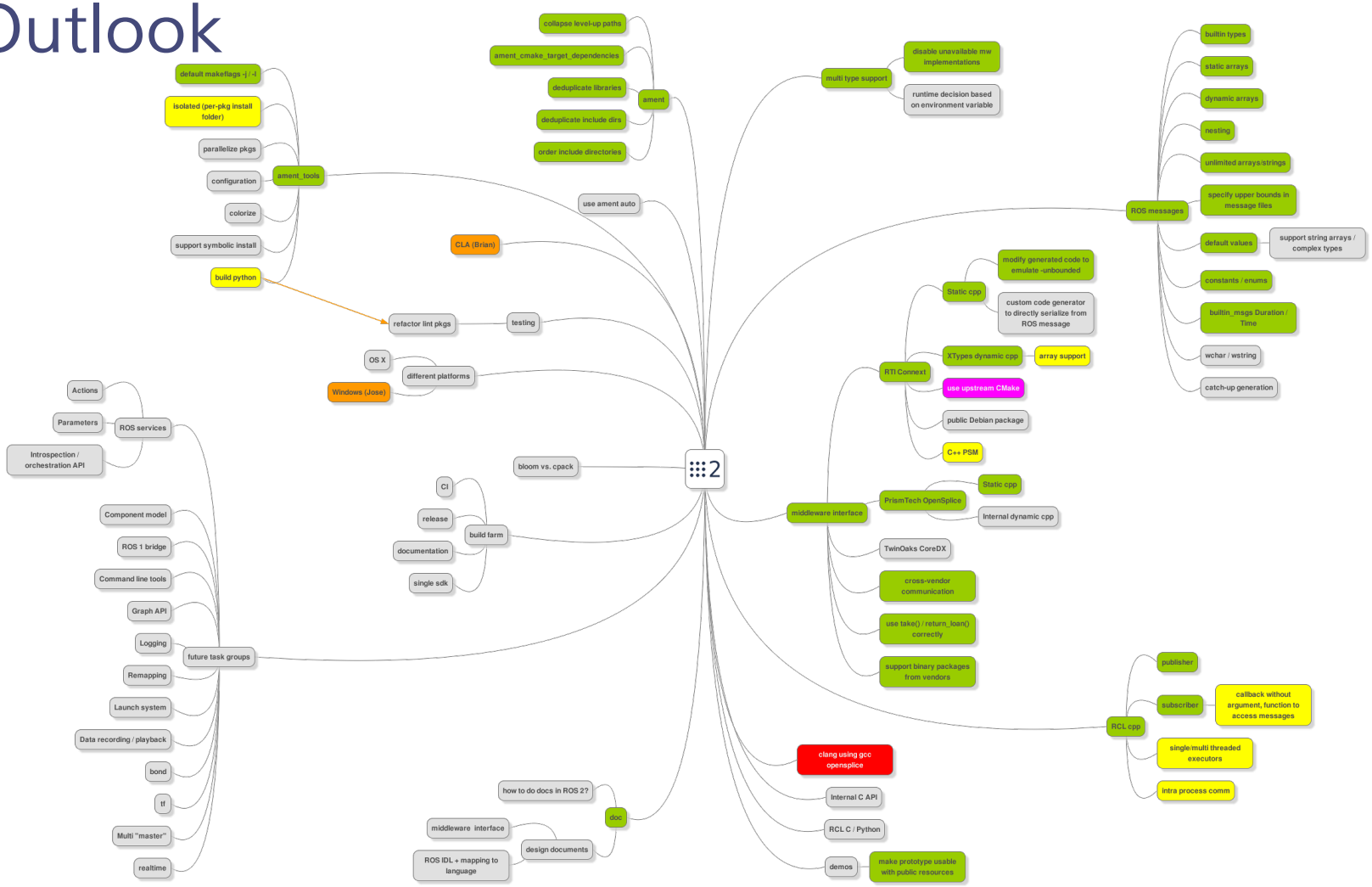
- Better support for various network configurations
- Reliable / best effort communication, QoS parameters
- Better error detection, heartbeat for each node

- Better introspection and dynamic configuration of a ROS system
- Deterministic startup of complex systems
- Notification for added / removed nodes and topics etc.

- Same API for *nodes* and *nodelets*, decide at runtime how to use
- *Actions* realized as preemptable services with a feedback publisher
- *Dynamic reconfigure* / node specific parameters as a default, global parameters belong to the node named *"parameter server"*

- Support different platforms and architectures from day one
- Communicate with ROS 1 nodes to enable mixed systems

Outlook



First released version planned for May 2015

(beside ROS 1.x Jade turtle)

Thank you!