Rapid time to value with the open-source Robot Operating System (ROS)

Katherine Scott
Developer Advocate
Open Robotics

Camilo Buscaron
Open Source Technologist
Amazon Web Services
Who is Open Robotics?

We create open software and hardware platforms for robotics. We use those platforms to solve important problems and we help others to do the same.

*I like to think that ROS is to Robots what Linux is to operating systems.*
We are the stewards of two large open source projects

- Robot application SDK
  - www.ros.org

- Robot simulator
  - www.gazebosim.org
Agenda

1. Development advantages of using ROS
2. Simulation, visualization and validation tools
3. ROS2 and production robot deployments
Robot Operating System (ROS)

Powering the world’s robots

Most widely used software framework for robot application prototyping, development and deployment.

- Over 800 maintainers for ROS core distros and over 2800 contributors to released packages since 2009
- Over 20 Million debian packages downloaded in July 2019 (alone)

ROS2 Reaching Maturity and Feature Parity Fast

1st ROS2 LTS Q3 2019

LTS May 2016
Ubuntu 16.04
2589 Released Packages

LTS May 2018
Ubuntu 18.04
1624 Released Packages

Noetic Ninjemys
May 2020

ROS2 Eloquent
Q4 2019

Most widely used software framework for robot application prototyping, development and deployment.

ROS2 Reaching Maturity and Feature Parity Fast

1st ROS2 LTS Q3 2019

LTS May 2016
Ubuntu 16.04
2589 Released Packages

LTS May 2018
Ubuntu 18.04
1624 Released Packages

Noetic Ninjemys
May 2020

ROS2 Eloquent
Q4 2019
Why ROS?

• Developers can focus on delivering value, not infrastructure
• Analogy: ROS is to Robots what Node/Rails is to web development
• Tried and True robot design patterns
• CLI tools for deployment, monitoring and debugging
• Simulation tools allows for more flexible design
• Library of hardware interfaces and patterns
• A community of experts to help
Robot Design Patterns

• A well characterized publication/subscription bus for moving messages
  • ROS 2 Built on DDS layer to handle QoS and other aspects specific to robotics
  • Systems can be inspected and modified at run-time
• A blackboard parameter system for changing configurations
• Tools for state machines and data processing
• Libraries to help define the robot’s state and transformations
• Tools to start, stop, pause robot software
More Robot Design Patterns

- Logging and playback of data and events
- Tools to visualize robot data online and offline
- A tool chain for installing, building, and running components
- Modular package library allowing for loose code coupling / reusability
- Modularity allows users to interleave C++, Python, and more
- Many user contributed tools to help with things like building a web interface
How much time does ROS save?

- Let’s consider ROSBag features alone
- ROSBag is ROS’s custom message serialization and playback library
- Real world performance on autonomous vehicles that store > 1 GB/min of data
- Capability to save data, replay it, query data, build modules and tests against it
- Just this feature alone would take a small team years to re-implement
- While there are analogs like protobuf none of them are built for robotics applications
Hardware, less hard with ROS

The ROS pub/sub bus uses common messages to move data.

Built in messages for common sensors and actuators:

- Cameras
- Depth Sensors
- LIDAR / RADAR
- IMU
- Force Feedback Servos
- Power systems
- GPS

Plus easily extensible
Hardware, less hard with ROS

• Many hardware types already have driver packages
  • This makes installing and testing new hardware a snap
• Cameras for examples all use `sensor_msgs/image`
• These base messages can be extended to address user requirements
• Common message types mean shared debugging tools
  • All cameras use the same intrinsic calibration package
  • All cameras visible in RViz for debugging
  • RQT can open up Image message windows for debugging
Example: Asynchronous behavior is hard

- Writing asynchronous behaviors can be difficult, ROS makes it easy
- ROS Nodes subscribe to ROS messages. Callbacks triggered on new messages
- User only focuses on processing incoming data
- Nodes can then publish new messages or trigger events
- ROS nodes are composable: allow for complex events to be built up from simple modules
Testing software is crucial to building reliable systems

“Hardware in the loop” is difficult

ROS enables developers to create simulated components

ROS Bags allow for recording and playback of real sensor data

*If you can capture a failure you can write a test for it*

These tools allow developers to focus on testing their software, not running the robot
ROS Packages - clean and modular code

- Software in ROS is organized into logical packages
- A package might contain ROS nodes, ROS-independent libraries, datasets, configuration files, or third-party software
- The goal of packages is modular and useful functionality
- These packages range in fidelity from proof-of-concepts to industrial-quality
- **rosdep** works like a package manager enabling command line installation
ROS Wiki

- wiki.ros.org
- Documentation, how-to, and tutorials
- 75,000 daily wiki page views
- 100,000 wiki pages with nearly 25 daily edits
ROS Discourse

- discourse.ros.org is the community’s discussion forum
- For community collaboration, announcements and news
- Over 4,500 users on discourse with nearly 30,000 posts across over 3,000 topics
ROS Answers

- answers.ros.org
- Technical questions and answers website for the community
- Can be filtered by tags and ensures the relevant community involvement
- It has over 34,000 users with over 32,000 answered questions
ROSCon

- ROSCon annual developer conference
- ROSCon 2019 in Macau, China had:
  - 600 developers in attendance
  - From 40 countries
  - 80% of participants coming from enterprises and startups.
- Regional conferences with over 200 attendees in France and Japan
Federated Community

- index.ros.org is the canonical list of packages and repositories
- Federated model for repository tracking and visibility
- Developers can host code themselves or use free services
- Creating a repository here allows it to indexed, tracked, and searched.
- Sorted by ROS release and sortable / searchable
ROS Development: tools for today’s distributed team

- ROS’s modular code base, strong logging tools, and simulation tools empower modern distributed teams

- In the past a robotics lab and local team was needed to build a robot

- ROS’s core infrastructure empowers remote developers and academic collaboration
Simulation, visualization and validation tools
Development, debugging and validation tools

- **Gazebo / Ignition**: Simulation
- **RViz**: Visualization
- **rqt suite**: Analysis
Ignition Blueprint (May 31, 2019)

- Physically based rendering (PBR) materials
- GUI tools for model placement
- Payload-dependent battery model
- New Command line tools
- Incremental level loading
- Distributed simulation (work-in-progress)
Ignition physically based rendering (PBR)
Modularity

- Monolithic Gazebo decomposed into Ignition libraries
- Libraries can be reused in other applications
- Ignition Gazebo is just one particular composition
Gazebo / Ignition Physics

- No single engine is best for all situations
- Common API atop multiple physics engines
- Choose engine at runtime
- Maximal and reduced coordinate approaches
- Allows simple engines; e.g., kinematics-only
Gazebo / Ignition Sensing

- Parameterizable models of common sensor types
- Parameterizable models of common noise types
- Common API atop multiple rendering engines
- Export sensor data via middleware (e.g., ROS)
Gazebo / Ignition Extension

- C++ plugin API allows any kind of extension
- Get and/or set the world between physics steps
- Add or extend sensors
- Interface with hardware input devices
- Fake interactions that impractical to simulate
- Delegate interactions to other systems
High fidelity testing – The new R&D approach

• Simulation is quickly becoming the de facto standard for robotics research and development

• DARPA has used ROS’s simulation in both the Subterranean Challenge (SubT) and Robotics Grand Challenge

• Software developers can test and validate robotics concepts in a high-fidelity simulation without ever touching hardware

• Virtual Robot X showed that this capability works in maritime environments

• NIST continues to use this approach for industrial applications
High fidelity testing with Ignition Gazebo

- Ignition/Gazebo can be used for unit and integration testing decoupled from hardware

- Allows users to create CI/CD pipelines just like normal software development

- Users can simulate with physics and other phenomenon like power consumption

- This capability also allows developers to rapidly test hardware hypothesis without ever touching a robot
Simulation – The power of cloud parallelization

• Not only does simulation allow for easier robot testing, it also allows users to use the cloud to run multiple simulations in parallel

• This means users can test hundreds or thousands of variations of environments at the same time

• This allows developers to concentrate on fixing failure modes, not re-validating results

• This also allows developers to quickly tune parameters, trying all possible values in parallel
ROS 2 and production robot deployments
ROS 2 Design Goals

1. Quality of Design & Implementation
2. System Reliability
3. Real-Time Control & Deterministic Execution
4. Validation, Verification, and Certification
5. Flexibility in Communication
6. Support for Small Embedded Systems
Architectural changes

The core architectural change between ROS 1 and ROS 2 is the addition of the ROS Middleware (RMW) and Data Distribution Service (DDS), which support security, reliability, and determinism.
Data Distribution Service (DDS) & ROS middleware interface (RMW)

• The core architectural change between ROS 1 and ROS 2 is the addition of the ROS RMW and DDS

• DDS is a protocol specification by a standards body, like the “post office” — they set a standard format for addresses, zip codes, and letter sizes

• Each RMW is like a letter carrier such as FedEx, UPS, or the USPS – they use the DDS standard to move letters (data)

• If you need guaranteed next-day delivery, someone can provide it – this makes ROS on embedded microcontrollers now possible
ROS 2: Quality Improvements

• The decoupling of ROS messages from their underlying delivery system has greatly improved ROS performance

• Each RMW implementation layer can now be validated and verified to confirm its security, reliability, and determinism

• The addition of the DDS layer also makes it possible to use ROS in resource-constrained environments like microcontrollers

• Other improvements in ROS include ROS Node lifecycle management

• ROS nodes now have well-defined behavior at every stage of life
ROS 2: Quality Improvements

• ROS 2 also has a new build system, ament/colcon, which improves package management and greatly speeds up build times

• Ament also creates a “devel” workspace, allowing on-the-fly editing of python modules and greatly improving developer productivity

• The ROS client library unifies the ROS C++ and Python interfaces, creating a smaller codebase that’s easier to review and improve

• New ROS 2 features are being added with every release, such as recently contributed zero copy message passing for large binary message formats by Bosch
DDS-Security Specification

- Adding security enhancements by defining a Service Plugin Interface (SPI) architecture, a set of built-in implementations of the SPIs, as well as the security model enforced by the SPIs.

  Authentication  Access Control  Cryptographic
ROS 2 Technical Steering Committee (TSC)

- Manage roadmap
- Contribute development efforts for core tools and libraries
- Set developer policies
- Establishes working groups to focus on important topics
Is ROS 2 Ready?

Designed for Production Grade Applications

Multi-Platform (Linux, Windows, Android, FreeRTOS, etc.)

No Vendor Lock-In

Built on Open Standards

Permissive Open Source License (BSD and Apache)

Global Community
Thank you!