CUSTOMER STORIES

Report On Needs of the ROS 2 Community

February 7, 2019

Prepared By:

PickNik Consulting
picknik.ai

Dave Coleman, PhD
Andy McEvoy, PhD
Stephen Brawner, PhD
Mike Lautman, MSE

TABLE OF CONTENTS

1. EXECUTIVE SUMMARY

This report summarizes 15 interviews with users, or potential users, of the Robotic Operating System (ROS) conducted by PickNik Consulting for Amazon Web Services and the ROS 2 Technical Steering Committee (TSC). It is motivated by Amazon's customer-focused approach to developing their product specifications, and is intended to inform future ROS 2 resource allocations after the December 2018 ROS Crystal, the third release of ROS 2.

The feedback collected and types of feature requests varied greatly. Even considering the diverse applications of ROS, it is remarkable how many use cases and needs exist in the robotics community. This plethora of sometimes conflicting requests inspires great respect for the challenges faced by Open Robotics and the ROS 2 development team in making design decisions and choosing the current feature set. While reviewing this report we ask the reader keep in mind that criticizing ROS is much easier than actually creating an open source, robot-agnostic, multi-purpose robotics middleware.

The primary takeaway from this report is that most of the key features needed by potential ROS 2 users have been addressed, but robustness and reliability through hardware testing is a huge unaddressed concern of users. Similarly, a branding and marketing problem remains due to overdue promises and scattered online assets. The strategy we believe would best address both of these concerns would be to choose a flagship hardware platform and set inspiring milestones (stretch goals) that would demonstrate ROS 2's applied capabilities (such as navigation and manipulation) while generating impressive demo videos. This effort would also spotlight missing feature areas still required for using ROS 2 on actual hardware.

2. SURVEY METHODOLOGY

Survey Questions

The first step in creating this report was to generate a standard questionnaire we would use to consistently ask each customer their thoughts on robotics software, ROS 1, and ROS 2. The following questions were asked:

- High-Level
  - What are your biggest problems with robotics software?
- Non-ROS Users
  - What were the driving factors to not use ROS?
  - Why do they do create their own software? What barriers are preventing them from using ROS?
  - What other tools are they using?
  - What have you heard about ROS2?
- ROS1
  - What are your biggest problems with ROS1?
- ROS2
  - Have you looked at ROS2, what do you know about it?
  - What are you most excited about in ROS2?
  - Have you tried developing with ROS2?
  - What have you heard about ROS2?
  - What features do you believe are missing in ROS2 that are highest priority?
  - What would you like to see different in ROS2?
  - What is keeping you from fully adopting ROS2?
- More directed questions
  - How do you feel about the documentation in ROS2?
  - What visualizations approaches do you use with robotics?
  - Have you used RQT? What did you think about it?

Survey Participants

PickNik Consulting reached out to their greater robotics network requesting participants to spend 15-30 minutes discussing their thoughts with us on robotics software. We grouped invitees into ROS users and non-ROS users.

We invited 25 participants and had 15 actual interviews. The following companies are represented in this survey data:

ROS Users	Non ROS-users
Southwest Research Institute / ROS-Industrial Fetch Iron Ox Plus One Honeybee Houston Mechatronics Aescape Rapyuta (x2) Carbon	Verb Surgical Wolf Robotics In-position Technology MSI TEC MDA (x2)

3. KEY PATTERNS

We used our collected data to rank, in order of importance, the pain points in ROS 1, or that are still perceived issues in ROS 2. We combined the biggest problems in ROS 1 and biggest desires in ROS 2 from our surveys into a set of needs that we are calling the 'key patterns':

- Robustness and Reliability
- Powerful & Easy to Use Debugging Tools
- ROS Master as Single Point of Failure
- Leverage More Best-Practices
- Distributed/Multi-Robot Systems Support
- Cloud-Enabled Capabilities
- Lifecycle Management
- Deployment to Production
- Security

In the following sections we will further explain each customer request in detail. Again, these sections are ordered from most commonly requested to least commonly requested. Occasionally statements from interviewees will be directly quoted, but the sources of these quotes have been left out for anonymity reasons.

Robustness and Reliability

This was by far the most reported issue with ROS 1, and the most cited reason for not yet migrating to ROS 2.

Many roboticists who have worked with ROS 1 for actual applications have fought long, hard battles addressing the "few and far between errors" that occur every "~30 days" when running a ROS-enabled system in production. These kinds of issues were completely overlooked when

academic researchers focused on ROS 1 for research purposes, but are critical when any downtime potentially means huge manufacturing line losses or bad brand reputation.

The more traditional integrators we talked to, who were non-ROS users, compared ROS users to "just kids playing" and observed that there has been a "high failure rate of companies that choose ROS". They also cited that using ROS would require their clients to employ highly skilled and costly robotics software engineers to maintain their automation programs.

The ROS 1 users emphasized difficulty addressing the many edge cases that affect reliability. They fear that, while many patches have been submitted to the core middleware of ROS 1, a whole new host of edge-case issues are looming in ROS 2 that have barely begun to be addressed.

A common concern around ROS 2 is whether it is being tested and verified enough on actual hardware. The original company that developed ROS, Willow Garage, did a great job of creating incredible demo videos of real robots running ROS in unstructured environments for longer period of time - on the order of hours and days. In particular, Willow Garage established several milestones for their flagship PR2 robot. The first milestone had the PR2 autonomously drive 3.14 kms, twice (two days in a row). The second milestone demonstrated the PR2 navigating through eight doors and plugging its power cord into nine outlets using its arm. These videos, published on YouTube in 2008 and 2009, demonstrated to the world that "we have a robust platform to share with the Open Source robotics community" [Willow Garage blog post, Dec 2008]. ROS 2 does not yet have, to our knowledge, hardware milestones to inspire adoption and drive development use cases.

Our interviews suggest potential ROS 2 users are holding out until ROS 2 has been demonstrated on real robotic systems running "thousands of hours". They expressed concern that Open Robotics' approach of being a software-only company results in sometimes misdirected development efforts to things like build systems. To their knowledge, ROS2 projects like the *navigation2* package have only been demonstrated on the *TurtleBot* platform, and then, only in simulation.

Representatives from ROS Industrial and from manufacturing were naturally the most vocal in concerns about robustness and reliability. One representative said "I have not seen one thing that I could even say ROS 2 is ready for primetime", emphasizing the need for tangible demonstrations to show their industrial partners.

ROS 2 is likely far more tested on hardware than is currently known to common users. Large funders of ROS 2 like Toyota are probably running robust systems, but clear marketing and branding issues exist, perhaps due to the confidentiality of the projects driving ROS 2 development at Open Robotics. However, the ROSCon 2018 presentation from Toyota titled "ROS2: Supercharging the Jaguar4x4" was not nearly as inspiring as the PR2 milestone videos in 2008, leaving some attendees wondering why ROS hasn’t progressed in the past decade.

Related to reliability were general concerns around the maintenance and support of existing ROS 1 packages that have left a bad taste for ROS users considering migrating to ROS 2. Will ROS 2 be any different, or will these ever resource-constrained open source projects continue to be "pseudo-maintained" and suffer from endless forks? A problem with ROS 1 is that rather than having patches and improvements accepted into the original projects, many one-off forks were created that stagnated. Will there be any quality assurance in what gets released in ROS 2? Of course, many of these concerns are common pitfalls of open source projects that are somewhat unavoidable. Perhaps the needed support has already been found through the ROS 2 technical steering committee resources.

Powerful and Easy To Use Debugging Tools

The next most common request in ROS 2 was for more robotics "tooling" to debug the complexity of robotics. A live robotics system has many actuators, sensors, and processes interconnected in non-deterministic ways. Unexpected behaviors and malfunctions can have many subtle failure-points, and robotics developers want a middleware framework that aids in "reducing the resolution time of bugs". They expressed the desire for more and better tools along the lines of *rostopic, rosservice, rosnode,* including ways to "easily expose information about the network". Network inspection "for the sophisticated middleware" was requested, such as tools for telling you "which machines things are running on".

One cited issue with ROS 1 was that the "only way to debug a ROS 1 system is to run it and inspect it"; this is an approach that scales poorly. New model-driven development approaches are needed to aid in verification; they cited Matlab's Simulink as a primitive example of hierarchical design of systems. "Composition can get complex quickly" was one comment. They asked how do we still have segfaults and bugs in mature computer programs "like MS Powerpoint" and expect our robotic software operating in the even more difficult real world to not have bugs? New approaches to software development are needed as systems get more complex.

ROS Master As Single Point of Failure

The users we talked to were most excited about the lack of *roscore* in ROS 2. They cited that this was a "huge weak point" in their ROS 1 implementations, and allowed for a new level of distributed systems. The choice of DDS and its lack of a master has been a big lure for almost everyone we talked to.

Leverage More Best-Practices

A lot of criticism of ROS 1 and ROS 2 during our interviews was directed at a lack of standards leveraged in ROS from much larger technology communities. For ROS 1, this lack of standards is more understandable given the state of related technology in 2008. But various interviewed ROS users were disappointed that more common standards today weren't utilized when bringing together ROS 2, saying that the "tech stack is out of sync with developers".

The biggest examples of existing standards that ROS is not leveraging are the very mature internet tech stacks and exploding IoT standards, which have solved many similar problems to robotics. Another cited example is the gaming industry: robotics problems typically attempt to recreate its perceived world in a video game-like environment and then make decisions based on this digital representation. Why not build Rviz on modern gaming engines like Unity? Other cited examples include the myriad of open source libraries popular today such as protobufs, bazel, or just plain CMake files.

A similar concern was that even within ROS there are not enough best-practices. Aside from commonly accepted interfaces for sensors and actuators, the ROS community leverages very few well-defined interfaces. ROS's "loosely federated model" creates widely varying packages that result in less-consistent user experiences for ROS users. More published standards for how to develop with ROS could improve the quality of the ROS ecosystem. A suggestion from the author is a single source of automated code formatters, code scanners, code linters, and continuous integration.

Distributed/Multi-Robot Systems Support

When asked what features in ROS 2 are of highest priority to them, many users were keenly interested in improved multi-robot support. ROS 1 was highly robot-centric, assuming there was one robot per ROS master. Various implementations of distributed ROS 1 systems were developed over the years, but they were second-class solutions that suffered from many gotchas.

Potential ROS 2 users are looking for a robotics framework that supports maintaining a distributed fleet of robots. They were interested in "mesh networks and redundancy". In particular, better namespace resolution capabilities of multiple robotics was requested.

Cloud-Enabled Capabilities

Similar to the above discussion of best-practices, a common request was for better integration into the web. ROS 2 should have out-of-the-box support for creating REST APIs. A desire for "less bridges required to connect ROS to the internet" was expressed. The ROS 1 approach using the *ROS Bridge* package is hopelessly out of date and the scorn of many companies we talked to.

Better Lifecycle Management

From our interviews, lifecycle management was a much anticipated feature of ROS 2, sorely lacking in ROS 1. Potential ROS 2 users are excited about having an "orderly way to shut things down" and "manage [their] runtime environment". This promises to improve the ability to create integration tests that previously were a "nightmare" without lifecycle management.

Several interviewees expressed concern that this feature was still in very early prototype stages, though after reviewing the documentation, the author believes this is not actually the case. However, this again points to a problem in ROS 2's maturity awareness.

Deployment to Production

As we near the end of our customer use case list, it should be noted that these are the items of least concern reported from our interviewed ROS users.

Several interviewees expressed that improved deployment was an important feature desired in ROS 2. After standing up a ROS 1 system, they expressed frustration in an unclear path to deploying to production. There was "no gradual development pathway from very rapid prototype ROS systems". In particular, one interviewee needed solutions for managing processes on remote systems.

Arguably, this is the domain of commercial companies and not an open source project, but it seems there is room for creating best practices and tools for deployment.

Security

Surprisingly, only three of our ROS users expressed desires for more security in ROS 2. Perhaps this was an assumed attribute of ROS 2 due to the advertised feature set of DDS. A specific desire was for a "locked-down production-ready" ROS system.

Other Requests

The remainder of the collected requests varied greatly - anything you could possibly request of a robotic framework was requested. We've omitted that feedback here for brevity.

4. MIGRATION BLOCKAGES

We asked our interview participants "What is holding you back from switching?" Here we seek to understand what it will take to migrate the whole ROS 1 community smoothly to ROS 2, such that our shared efforts are maximized.

Many of the reasons we just outlined above in the "Key Patterns" section of this report. In particular, robustness and reliability was the biggest cited reason.

Another popular response was that they were waiting for all packages they depend on to be ported before they will take the leap and start migrating the packages they maintain. Speaking to the Github issue tracker for ROS 2 features, one cited "All boxes must be checked for core functionality". Another requested the "100 most popular ROS 1" packages be ported. Many cited MoveIt! as a key package that would indicate enough of ROS had been ported, though the interviewees may have biased their responses knowing the interviewers are the maintainers of MoveIt!.

This waiting until someone else puts the effort first into migration is the common prisoner's dilemma, as concerted cooperation would result in a better ROS 2 for everyone. Mitigating this blockage seems to be one of the dominant strategies by the TSC currently in accelerating the adoption of ROS 2.

A few established companies expressed no desire to switch, saying "there's absolutely nothing" driving them to upgrade. These larger corporations that already invested heavily in ROS 1 can't switch due to legacy support or long development cycles. Another issue is the now maturing batch of robotic startups that built their tech on ROS 1. These companies are now entering "their series B, C, etc" and "won't see it worthwhile to switch" as the investors begin demanding returns.

Others cited reasons for not switching that, to the best of the author's knowledge, are no longer true. They cited they were waiting for Rviz to be ported, actionlib, or other recently released components in ROS Crystal.

Some of our interviewed ROS users said they actually were ready to switch, with the Crystal release indicating that "times seem right to pull the trigger". For example, a small startup wants "to migrate as soon as possible so new developers can learn ROS 2" instead of ROS 1. One reported "I'm happy with the roadmap".

5. VISUALIZATION AND RQT

We asked users "What visualizations approaches do you use with robotics?" and the results were not surprising. Essentially everyone uses and loves Rviz. One veteran ROS user noted that "Rviz is a large lock-in thing driving people to continue to use ROS. No one sees value in re-writing Rviz."

Most have used RQT to varying degrees. A few mentioned using Python's matplotlib for data analysis, others Matlab. Interactive markers (a feature in Rviz) are also popular. Lastly, many use custom Web-based user interfaces for customer-facing parts of their product.

Due to PickNik's recent collaboration with Amazon, we specifically asked users at the end of the interview "What issues do you have with RQT?". The most popular feature was RQT's plotting capabilities (rqt_plot) with everyone essentially loving and hating it. While it's a great tool for visualizing live data streams from ROS topics, it is clunky and hard to use. Many had tried a recent rewrite/fork of the plugin, a stand-alone application called *plotjuggler*. They cited it also was missing features and was problematic.

Most people seem to use only a small subset of the RQT plugins, though they still cite that RQT is what makes ROS friendly to beginners. They like the plugin interface aspect of the framework. One cited the *robot_monitor* plugin is "very popular with firmware guys". Complaints included RQT being too resource hungry and another company was discouraged by Qt's licensing model that requires per-seat licensing for robotic installs.

6. DOCUMENTATION

In our interview we asked "How do you feel about the documentation?" We had a middle-of-the road response to this question, with several saying "it's ok". Others had not really tried using ROS 2 yet so could not comment.

Still others were less enthused: "when I looked at it, it was not very good" and another "it's not even as documented as C-turtle was for ROS 1" which indicates it is really bad. Several commented that it's all over the place and very fragmented, due to a move from Open Robotics to have an even more federated model in ROS 2. Some commented that its harder to contribute to than the ROS 1 wiki, because of the more tightly-controlled pull request approach.

One responder said they really appreciated the effort put into design documentation, but that there was a "disconnect between design docs and what was actually done". The documentation was described as "hard to find" and required "going to the source code".

Notably, Willow Garage's previously mentioned milestones for the PR2 program and ROS 1 development included a third milestone. The third milestone was essentially having the entire company spend a full month writing tutorials and documentation. This resulted in incredibly high quality online assets that guided the beginner roboticist into learning ROS 1.

Creating a consistent user experience in learning ROS 2, starting from the basics and gradually into more advanced topics, is a much needed area of investment. One of our interviewees pointed out that (as of this writing) the first 10 "Basic" tutorials for ROS 2 cover the mundane topic of build systems (ament and colcon) and cross compiling the code[1]. For comparison, the first tutorial of TensorFlow teaches you how to "train your first neural network" to identify different images of clothing. "If the entry barriers for configuring and using the software on robots is too high, even the most powerful of frameworks are useless"[2].

[1] https://index.ros.org/doc/ros2/Tutorials/
[2] Coleman et. al, "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study" https://arxiv.org/pdf/1404.3785.pdf

7. MARKETING

One of the key takeaways the author has from these interviews is that ROS 2 is suffering a branding problem. One ROS user said "it's been oversold for too long - 5 years - and everyone is skeptical [of ROS 2]." Another pessimistic community member in our interviews said that "Only non-ROS 1 users get excited about ROS 2." When asked why they haven't switched one said "OSRF is pushing out an unfinished product... ROS 2 is still missing a ton".

Many of the concerns we heard were due to outdated information about ROS 2 and features that are no longer actually missing. Despite the recent speed up of ROS 2 development, thanks in part to the TSC, ROS 1 users have stopped paying as much attention to ROS 2 announcements due to pessimism that it will never actually be ready.

One veteran observed that the original "ROS 1 developer community is burning out because they've been pushing on robotics for so long". Yet a younger ROS developer remarked that "during job searching ROS 2 seems to be in high demand," so it's all a matter of perspective.

We believe that a bigger marketing effort needs to be made to signal to the community ROS 2 is finally ready for adoption. Understandably, Open Robotics has historically not chosen to focus their limited resources on brand image, perhaps due to their academic roots. However, a little more polish on the ROS 2 project could go away. The documentation and brand of ROS 2 is scattered and inconsistent, with multiple sources of information of varying quality. A clear promotional channel of new features and hardware video demonstrations is lacking, with the heated next-gen Discourse forum the primary outlet.

8. CONCLUSION

From our interviews, validating ROS 2 on hardware platforms and promoting these efforts through demo videos would be the single biggest driver of adoption. There is a need for a clear flagship robot to develop ROS 2 against, as discussed in the *Robustness and Reliability* section. This PR2-esq platform would keep the ROS 2 development requirements aligned with applied roboticists solving complex real-world problems.

We recommend that to test the durability of ROS 2 and inspire the community to migrate, hard robustness milestones should be established and worked towards, just as Willow Garage did for ROS 1. These milestones should be bigger than the ones ROS 1 achieved a decade ago, otherwise why should someone migrate from ROS 1? For example, perhaps Toyota's HSR *Human Support Robot* could have an open-source version running pure ROS 2, that can do everything the PR2 could do in its milestones but with 10x longer autonomous run time, with multi-robot collaboration... and maybe with full Amazon cloud support.

Backing this milestone would be a huge library of documentation and tutorials to recreate the demo, just as we had with the PR2. The first tutorial would be a quick-start to ROS 2 that would guide beginners into easily running a fully simulated robotic system doing useful industrial or consumer tasks. This would rebrand ROS 2 as a finished product that can wow users, just as ROS 1 did for an entire decade of roboticists and developers.

Overall, ROS 2 has made huge progress in the lastest release towards becoming a mature platform ready to supplant ROS 1. It's time to prove out the middleware in the real world, add marketing polish, and change the community opinions about ROS 2's readiness. We look forward to contributing to the success of the ROS 2 project and thank all the contributors who have dedicated themselves to this momentous open source effort.

About PickNik Consulting

Our team is rooted with strong academic backgrounds of robotics theory combined with applied software experience. PickNik's mission is to support the worldwide open source robotics movement through community building, consulting expertise, and the development of multi-purpose highly-capable motion planning software.