Sharing Software with ROS

Steve Cousins, Brian Gerkey, Ken Conley, and Willow Garage

Robot operating system (ROS) is designed to promote code sharing and enable the development of open-source robotics commons. Sharing code will help the robotics community to progress faster by letting the researchers in the community replicate and extend the results of other research groups. ROS makes it easy to find the software and integrate it into robot systems. In this issue, we'll discuss how sharing is facilitated by the ros.org Web site, look at some examples of code being developed and refined at different sites, and finally consider the personal robot (PR2, version 2) program, which is making the common hardware available to accelerate robotics innovation.

Willow Garage is setting an example of code sharing by creating and releasing a set of mature and stable software stacks, and other institutions around the world have been following suit. ROS contains the software stacks for everything, from building blocks such as controllers and filters to applications like autonomous navigation.

Breaking Down the User/Developer Divide

A successful open-source project can attract a staggering number of users. Most of them will strictly use the software, a minority of them will report problems or ask questions, a smaller minority will yet provide bug fixes or minor enhancements, and only a tiny percentage will become developers and contribute heavily to the project. With ROS, we are aiming for a substantially different mix. While a small development team works well for many projects, the endeavor to build a robotics commons is too broad and varied in scope to be accomplished by any single group. We need help from experts in all of the areas that make up the interdisciplinary field of robotics, from low-level control to high-level reasoning, and everything in between. With the goal of enlisting those experts as contributors, we made three key design decisions in ROS.

First, we enforce symmetry in the development and runtime environments. ROS development is governed by a package system, with no one package more privileged than another. Code is added to ROS by simply creating a new package. Similarly, a running ROS system is a graph comprising nodes, with no one node more privileged than another. The way to add functionality to a system is to launch a new node. The users naturally become developers, because their work can be reused by others. If you create new functionality in your work, we can easily take advantage by adding your package(s) to our ROS installation and launching your nodes on my system.

We build on this symmetry in our second design decision; instead of going through an official gatekeeper, ROS code is maintained in a decentralized federation of repositories. When someone asks, "How can I contribute to ROS?" our answer is, "Create a publicly accessible ROS repository." By hosting their code in their own repository, the user developers retain control over their software (deciding on licenses, development policies, etc.) and get credit for their work (by convention, repositories are named after the contributing institution). The overhead of maintaining a repository is minimized by the use of community-hosting sites such as SourceForge.net, Google Code, and GitHub. To see the benefit of the federation model, we have to only look at the repositories hosted by the ROS community. At the time of writing, we know of 18 public repositories that are maintained by institutions other than Willow Garage, and they collectively host 300 packages.

Third, we have established a convention that the granularity of ROS software is very fine, i.e., the packages are very small. Besides being common-sense engineering practice, fine-grained modularity pushes overarching architecture decisions out of packages and to higher-level system configuration. This lack of architecture in ROS libraries makes them much easier to integrate into other platforms and also makes it easy to integrate code from other platforms (e.g., OpenRAVE, Player, OROCOS) into ROS. The lack of enforced architecture also supports the federated repositories. While each repository representing a particular robot software platform does have an architecture, the individual components are architectureless and easily used elsewhere. This model is embodied by a repeated mantra among ROS developers, "We don't wrap your main."

Of course, decentralized development of many small packages provides an opportunity for confusion: how do you know what ROS software is available, where to find it, and how to use it? To solve this problem, we created the community site http://ros.org as a clearing house for the ROS software. The aim of the site is to enable search across the federation of code repositories. For software to be searchable from ros.org site, an institution just has to register a code repository once, and from then on, ros.org crawlers will check for updates, create an index, and integrate search results from many sites into a single page with a common look and feel.

In addition to indexing code repositories, ros.org site hosts documentation and tutorials for ROS packages, stacks, and applications. The ros.org site is modeled on python.org, which has evolved a system for sharing a massive amount of code written in the Python language.

Robots Powered by ROS

All around the world, many groups are using ROS to power their robots. Let's look at some examples of software created for the Care-O-bot 3, iRobot Create, and Aldebaran Nao.

Digital Object Identifier 10.1109/MRA.2010.936956

These are all hardware platforms that can be acquired for research, so any ROS code running on them is directly sharable.

The Care-O-bot 3 is a mobile manipulator from Fraunhofer IPA in Germany (Figure 1). The robot is designed to have a human facing the side with a tray and a Schunk lightweight arm that can pick items up and set them on the tray. The robot's software is open source and available via ros.org and includes some simple applications (a dashboard and teleoperation application), packages for controlling the motors and arms, and a simulator. Many of these packages make use of the underlying ROS capabilities, including the messaging system and transform library.

The iRobot Create is a version of the Roomba robot sold as a platform for experimentation (Figure 2). Brown University has made drivers available for the Create platform, one example being a simple Webcam driver. Of course, for simple robots like the Create, Player (http://playerstage.sourceforge.net) is a fine software solution (and in fact, Brown's code is an

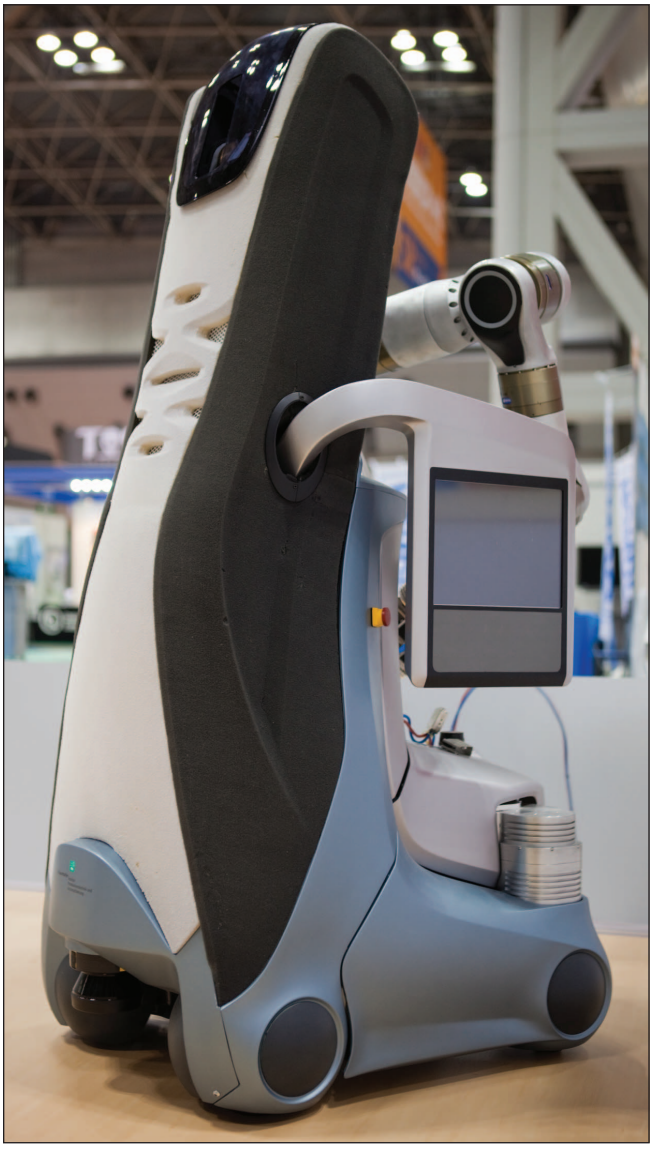

Figure 1. *The Care-O-bot, first prototyped in 1998 at Fraunhofer IPA, is designed to be a mobile personal assistant.*

Figure 2. *Brown University's small universal robotics vehicle, based on the iRobot Create, provides a versatile platform for ROS development.*

Figure 3. *Aldebaran Robotics' interactive Nao robot hopes to become a staple in robotics classrooms.*

Sharing code will help the robotics community to progress faster by letting the researchers in the community replicate and extend the results of other research groups.

ROS wrapper around the existing Player driver for the Create), but the ability to run ROS to control this platform gives developers access to all of the other capabilities in ROS, from flexible-distributed computing to powerful data visualization.

The Brown Robotics group also released drivers for the Aldebaran Nao platform (Figure 3), including basic movement, head control, speech, and camera access. Researchers at the Albert-Ludwigs-Universität in Freiburg, Germany, added joystick teleoperation, joint state inspection, and a basic robot model. This extension is a nice example of groups around the world, building on each others' work to achieve more than they would have alone.

Figure 4. *Willow Garage's PR2 uses ROS to open doors, serve drinks, and fold laundry.*

Finally, the PR2 Beta program is making approximately ten PR2 robots available to selected institutions (Figure 4). The PR2 runs software for, among other basic functions, calibration, navigation, and manipulation, as well as for higher-level applications including mapping and plugging into standard wall outlets. All of the code developed for the PR2 at Willow Garage will run on the distributed robots, and in exchange for use of the PR2 robot, the recipients will make their code available open source.

These are just a handful of examples of how ROS is making it possible for researchers around the world to work together to advance robotics. In "ROS Topics" in the next issue, we'll review the proposed work of the ten organizations selected in the PR2 Beta Program (http://www.willowgarage.com/pages/pr2-beta-program/cfp). This work should really accelerate code sharing in robotics!