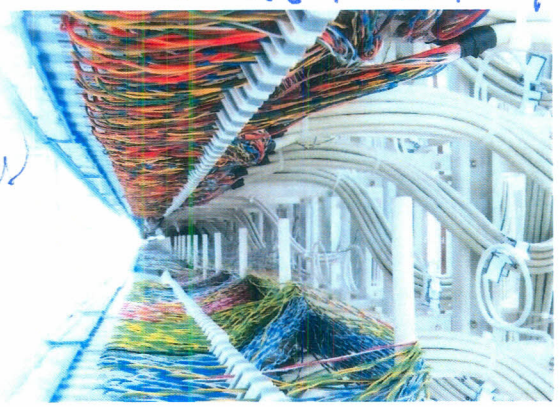
Robot OS: A New Day for Robot Design

Lee Garber



The Robot Operating System promises to make designing robotic software easier and less expensive.

Robotics has undergone many advances in recent years, with robots gaining new skills and the ability to undertake new tasks. For example, they are helping with surgery, and are even flying and executing complex instructions as drones.

However, robot use has not become as widespread as many people have predicted. Even the development of industrial robots—formerly a bright spot—has stagnated.

One reason is that robots have not had a standard platform on which developers could build. Instead, they typically have had to start from scratch with each project. This has frequently made the development process time consuming and expensive.

In addition, the lack of a standard platform has frequently kept different types of robotic software from being able to communicate with one another, which has limited developers' ability to use multiple applications within a single robot. This has also kept engineers from being able to reuse robotics software and otherwise collaborate.

This may all be about to change, though, as an increasing number of companies and researchers are working with the open source Robot Operating System (ROS) platform—first released in 2010—to provide software for their robots. And as this has occurred, the pace of robotics innovation has increased.

Despite its name, ROS isn't just an operating system, although it includes some OS functionality. It is also a framework that provides software modules for performing typical robot activities such as object recognition. The technology also helps control and coordinate the modules.

"ROS is a software framework that simplifies the task of writing complex robot software. With ROS, it is possible to quickly grab state-of-the-art software for many aspects of the system, which frees up R&D time for the novel aspects of a particular project," said Morgan Quigley, who developed the first version of ROS and is now chief architect of the Open Source Robotics Foundation (OSRF). Proponents hope ROS will enable developers to flexibly and inexpensively create more capable robots.

However, the technology faces challenges before it can become the dominant robotics platform.

A BRIEF HISTORY

When Quigley arrived at Stanford University to study machine learning for his doctoral program, he joined associate professor Andrew Ng's Stanford Artificial Intelligence Laboratory (SAIL). Students there were working on the Stanford Artificial Intelligence Robot.

STAIR reflected Ng's desire to design a general-purpose robot that could perform different types of tasks, rather than the one narrow job that office and industrial robots typically do.

To accomplish this, Quigley realized that SAIL required a software framework that could integrate programs from various students in a robot that would keep working even if one of the applications failed. In 2006, he designed a distributed, peer-to-peer system called Switchyard to meet this need.

The next year, in a collaboration that led to ROS's creation, Quigley began working with Willow Garage, a research lab that developed personal robotics hardware and open source software.

Willow Garage, which has since closed, released ROS version 1.0, called Box Turtle, in 2010. There

Published by the IEEE Computer Society

have been five releases since, the latest being ROS Groovy, introduced in December 2012.

The OSRF now acts as the hub for the ROS community, said Brian Gerkey, the organization's CEO.

"ROS addressed a very fundamental need in the research robotics community," said Carnegie Mellon University (CMU) associate professor Siddhartha Srinivasa, who founded the school's Personal Robotics Lab. It lets developers obtain and combine state-of-the-art software components from many sources, something critical for today's complex machines, he explained.

UNDER THE HOOD

ROS is currently used primarily with Linux operating systems, although some parts work with Mac OS or Windows, said Gerkey. It is distributed via a BSD license and is free for anyone to use, change, and base commercial projects on.

How it works

"ROS is a collection of software that makes it easy to program robots to do things," explained Steve Cousins, CEO of robotics vendor Savioke.

"ROS is a robotic middleware layer," said CMU's Srinivasa. "It provides standardized systems for the critical tasks that most pieces of robot software need to do, such as pass information around, read and set configuration parameters, and compile and run source code."

It is, in essence, a software framework for robot software development. According to the OSRF's Quigley, the framework provides software modules that users can plug in as is or modify as needed to perform typical robot activities such as motion planning, the manipulation of physical objects, and visualization.

The technology enables functions such as hardware abstraction, message passing, and software-package management.

ROS also provides middleware services, as it can let an ROS implementation's many small programs talk to one another, Quigley noted.

OSRF CEO Gerkey said, "At the core of ROS is the communication system, the software that is managing the communications between different parts of the system."

ROS can run on various types of hardware—including workstations, and single PCs—simultaneously. Thus, developers could control a robot via multiple computers, each handling a different task and each running a different type of control software.

Multiple software modules could also operate at the same time. And developers could connect and disconnect them for purposes such as testing and debugging without destabilizing or having to shut down the entire system. This allows an organization to debug or otherwise work on just one part of a robot—such as its computer vision component—while still running the machine and using its other capabilities.

ROS is based on several key principles.

Peer-to-peer. Complex robot systems consist of multiple onboard and offboard computers. ROS uses a peer-to-peer (P2P) architecture to let computers talk directly with one another. Thus, Quigley said, they don't have to spend time and effort communicating through a central hub.

Language neutral. ROS modules could be written in various languages. There are core ROS libraries written in languages such as Python, Lisp, and C++, and there are experimental libraries in other languages.

Decentralization. ROS uses multiple microkernels—rather than a large centralized runtime environment—for various robot components. This makes systems more robust, Quigley noted, because the failure of one executable doesn't affect others.

In its decentralized architecture, he explained, ROS works with *nodes*, each of which represents a robot executable such as a motor, sensor, or algorithm. Each node announces itself to the *system master*, which is a node-declaration and -registration service. This lets nodes locate and communicate with one another.

The lack of a standard platform has made robotic design more difficult.

Small and reusable. In ROS, Quigley said, drivers and other system elements are contained in standalone executables, which enhances reusability and minimizes application size.

Benefits

By providing prebuilt software packages that can be adapted as necessary, ROS eliminates the need to build programs from scratch for every project. This reduces robot-application development time and costs.

The constant use and modification of applications by developers leads to ongoing improvements. And because ROS is open source, it is easier for users to adopt. In addition, developers' additions or improvements are available to the larger community.

Robot-software developers working on a project, who may have expertise only in certain fields such as machine vision or embedded systems, can use

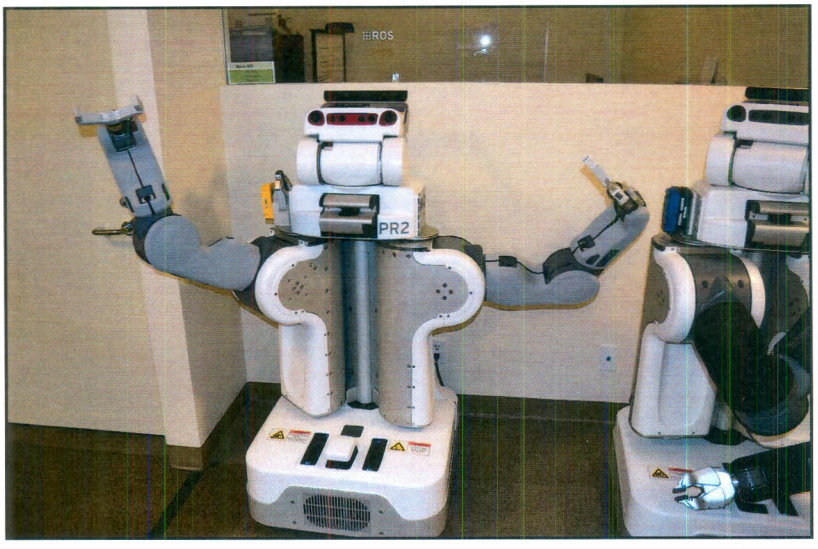


Figure 1. Willow Garage's ROS (Robot Operating System)-based PR2 (Personal Robot 2) robot was a development platform on which users could design their own robots.

ROS modules built by people with expertise in other fields.

And because developers could use so many different types of free, already-built modules, ROS makes robot creation more flexible, less expensive, and faster, Quigley said. In addition, the ability to creatively combine modules lets developers build robots with new capabilities.

"ROS's biggest blessing," said CMU's Srinivasa, "is that it is an open source project. This means that it has an incredible following and a tremendously large repository of compatible packages."

USING ROS

There are numerous ways to access ROS-based code for use in robots.

For example, there are repositories of ROS software packages. ROS.org (http://wiki.ros.org) is one source of the software. Through October 2013, 175 organizations and individuals publicly released ROS software in the site's indexed repositories (http://code.ros.org), up from 50 in 2009.

By the end of 2012, there were 135 other publicly released and indexed repositories, up from 80 in 2011 and 20 in 2010. Some are listed at www.ros.org/browse/list.php and http://wiki.ros.org/RecommendedRepositoryUsage/CommonGitHubOrganizations.

In December 2012, the OSRF reported that it had 3,699 public ROS packages, compared to 1,600 three years ago.

The OSRF's Quigley said his organization encourages people to create their own repositories so that they can maintain control of and receive credit for their contributions.

ROS-Industrial

Clay Flannigan, manager of robotics and automation engineering in the Southwest Research Institute's (SwRI's) Automation and Data Systems Division, noted that his organization founded ROS-Industrial as "an open source project that builds upon the ROS core and addresses the unique needs of industrial robot users."

The ROS-Industrial Consortium (http://rosindustrial.org/ric) oversees the program, whose goals are to develop and provide robust ROS applications and APIs for industrial robots, encourage cutting-edge research into further development, and create a community for researchers and industry professionals.

ROS-Industrial also focuses on the software quality, support, and documentation that industrial users expect but that are not always found in open source software.

Robots

The OSRF says that many hundreds of robots—including 90 different types, up from 50 in 2009—are running ROS. A partial list is at http://wiki.ros.org/Robots.

"ROS is widely adopted in the robotics research community and, in many fields, is the lingua franca for software," said CMU's Srinivasa.

Numerous academic researchers have used ROS in their robotics projects.

For example, at CMU's Robotics Institute, Srinivasa and Intel researchers utilized ROS to develop HERB (Home Exploring Robotic Butler). HERB has performed household tasks such as opening refrigerator and cabinet doors, finding coffee cups, and discarding trash.

Many other labs worldwide are adopting ROS and are constantly adding capabilities to the software, creating advanced robotics packages in the process.

DARPA is now requiring that developers use the technology in several of its robotics projects.

And in this year's DARPA Robotics Challenge—designed to generate R&D that will enable robots to perform hazardous activities in disaster-response operations—every contestant will have to use ROS.

In recent years, a few commercial robotics projects have used ROS. Proponents say this demonstrates the technology's marketplace viability.

In 2010, Willow Garage used ROS to build PR2 (Personal Robot 2), shown in Figure 1, as a development platform that customers could use to create their own robots.

PR2—considered a landmark in ROS use—was able to perform tasks such as walking dogs, folding laundry, and plugging itself into an outlet when its battery ran low.

According to the OSRF's Quigley, the successor to PR2 is Unbounded Robotics' UBR-1, a ROS-based mobile manipulation platform. But whereas PR2 cost $400,000, Unbounded Robotics says UBR-1 will cost $35,000 when released in mid-2014. The robot has one arm capable of seven different motions and is designed to automate tasks such as stocking shelves and inspecting products.

One of today's best-known commercial robots is Rethink Robotics' Baxter—shown in Figure 2—which runs on ROS. Baxter, introduced in September 2012, is a two-armed collaborative manufacturing robot used for simple industrial jobs such as loading, unloading, sorting, and handling materials.

Clearpath Robotics, which builds unmanned vehicles for R&D projects, used ROS in its Husky A200 robot.

Robotics Engineering Excellence, which develops intelligent modular object-manipulation systems for use with robotic systems, recently adopted ROS as its development platform.

Tecnalia, a Spanish multidisciplinary technology development company, won the 2012 European Manufacturing Awards' Factory of the Future award for its automated airplane-part assembly process, performed by Kawada Industries' HIRO robot, which is programmed in ROS.

CHALLENGES AND LIMITATIONS

ROS typically must work with a full operating system, meaning it isn't well suited for small, simple systems like those used in low-cost commercial robots, noted the OSRF's Quigley.

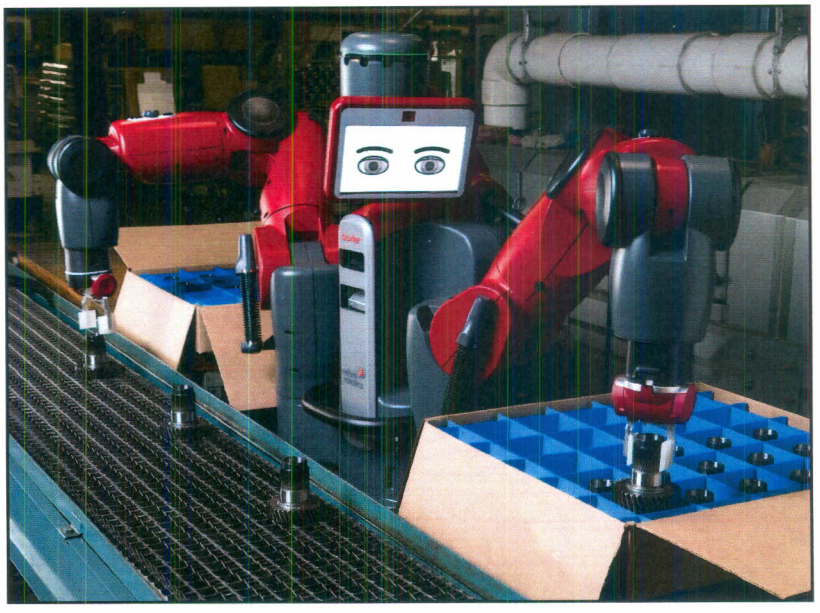


Figure 2. Baxter, a well-known commercial robot, runs on ROS. It performs basic industrial tasks.

In addition, he stated, organizations can't easily use ROS to control and coordinate multiple robots because it was designed to work on a single device via a centralized system master. Currently, users who want to control several robots must design a separate software layer to coordinate their individual masters.

"ROS has a ton of packages, which means what you need will quite likely be there but also that it can be hard to find," noted Cousins, "and ROS is widely used by the academic community, which means that it comes with lots of state-of-the-art packages that may have been written by graduate students, not professional programmers."

"There are a lot of misconceptions in industry about open source software, so adoption there is slower," he added.

Because ROS is open source, said CMU's Srinivasa, "the pieces of it that get maintained and developed are at the mercy of the contributors and their interests and funding. This means that some things never get fixed. There are many small inconsistencies and bugs that plague the ROS communication and transport layers, meaning that nodes can fail to start, freeze, or crash."

Also, he pointed out, "There is a lot of missing or out-of-date documentation because much of the system has been developed by volunteers who don't have time or have moved on to other projects."

And ROS works primarily with Linux, while many manufacturers use Windows systems, said SwRI's Flannigan.

The OSRF's Gerkey said, "The greatest challenge ROS faces to really widespread adoption is the widespread adoption of robots. There just aren't that many now. We need to come up with new applications for robots. That's when things will really break wide open."

As the number and types of robots using ROS have grown, developers have written more kinds of ROS software. This, in turn, has brought more people into the ROS development community.

"ROS has been widely adopted, and today, it has hundreds of active

developers and thousands of software packages," noted Flannigan. "It may not be the best technical solution for all problems in robotic software development, but because of strong community support, it is a first choice for many."

"The effects of ROS are definitely here to stay, in one form or another," said CMU's Srinivasa. "Regardless of how it continues, it has set the bar in robotics for what it means to be interoperable, easy-to-use, and open. And the core foundation behind ROS, the necessity of software that can connect [the parts of] a complex robotic system, will exist as long as robots do."

Now, though, the OSRF's Quigley stated, "We need to slim ROS down. We want to develop an ROS for use in smaller systems, such as embedded systems, that [use low-cost processors and] are too small to run Linux."

He said this would enable companies to use ROS to design simple, low-cost, mass-market robots, a key to the framework's commercial success.

And, he added, the OSRF is working on ways to make ROS useful for controlling multiple robots working together.

According to Savioke's Cousins, "The robotics industry is poised for explosive growth outside of the manufacturing sector, and ROS will be a catalyst for that growth."

"As the robotics industry continues to grow," Quigley said, "many of the new markets will be in challenging domains that require advanced perception and planning systems. The software required for many of these tasks and environments is enormously complex, and it is my opinion that software frameworks that simplify collaborative development will be valuable for many years to come."

"ROS is already widespread in the robotics research community," he concluded, "and it is exciting to see ROS start to emerge in the marketplace as well."

Lee Garber is the IEEE Computer Society's senior news editor. Contact him at lgarber@computer.org.

cn Selected CS articles and columns are available for free at http://ComputingNow.computer.org.