AN INTRODUCTION TO ROBOT OPERATING SYSTEM (ROS)

Mukul Kulkarni[1], Minakshi More[2] , Swapnaja Patwardhan[3] Darshana Yadav[4]
[1234]Department of MCA, MES Institute of Management and Career Courses, Pune, India

Article Info
***Page Number:** 45 - 51*
Publication Issue:
***Jan-June** 2015*

Abstract:
This paper gives an introduction to Robot Operating System. In the first section we have focused on historical development and contributions made through the timeline. In addition, we have also discussed about features and functionality of ROS with open-source advancement.

Article History
***Article Received:** 02 Jan 2015*
***Revised:** 12 Feb 2015*
***Accepted:** 24 March 2015*
***Publication:** 28 April 2015*

Keywords: ROS, Robotics, Open Source, Robot, Operating System

1. INTRODUCTION

Research in the field of robotics is increasing. For working with robots, a new open-source operating system called Robot Operating System (ROS) is getting recognized. This ROS has application in both in research and commercial as well. The major working associated with it involves creating applications for robot which includes hardware level access, writing device drivers and extending to communication using message passing. In ROS libraries are being build everyday with community efforts and largely used for research programs. For the process of getting results different researchers are creating some frameworks for rapid prototyping and experimentation modules which are used in academia and industry [2]

2. HISTORY

From many years' contribution from different developers ROS was developed. Due to the need for an open-source collaborative framework for the field for robotics and research associated with it an open operating system for robots was developed. ROS was created around twentieth century with major efforts from scholars at the Stanford University. With their efforts a dynamic operating system was built with functioning for flexible prototyping and integration.

Initially projects like STanford AI Robot (STAIR) and the Personal Robots (PR) were developed within institution as a prototype for robots operating software's. Willow Garage, in the year 2007 further develop this and created software program which can be used with any robot. Many research scholars contributed to his work and current ROS was evolved. Gradually it was completely moved from prototype to completely open source. [1]

ROS was truly collaborative environment from beginning. It was developed by contribution from multi-institutional scholars. Even though it was collaborative, still they manage they allowed user to start new repository for code. This flexibility was initially available due to less restrictions and there was no need for permission making it truly open-source work. [1]

3. TIMELINE DEVELOPMENT IN ROS [1]

Date	Development	Details
May 1, 2007	**Switchyard at Stanford**	**Before the ideas fully coalesced to become ROS, several robotics software frameworks were prototyped in research projects at Stanford, including the STanford Artificial Intelligence Robot (STAIR) and the Personal Robotics (PR) program**
November 1, 2007	**ROS at Willow Garage**	**Willow Garage begins their Personal Robotics project, and ROS becomes a formal entity.**
January 1, 2008	**Churn**	**The ROS team goes through many iterations at Willow Garage and Stanford, aggressively building (and discarding) various design and implementation concepts**
January 1, 2009	**ROS 0.4 Release**	**ROS gradually starts taking shape into the framework we are now familiar with**
June 1, 2009	**Willow Garage Milestone 2 Reached**	**Relying heavily on ROS, the PR2 alpha robot navigated through eight doors and plugged its power cord into nine outlets (aka "Milestone 2").**
January 22, 2010	**ROS 1.0**	**Following an extensive phase of documentation and user testing (aka "Milestone 3"), ROS 1.0 is released, establishing many of the components and APIs that are still used today.**
March 1, 2010	**ROS Box Turtle Release**	**The first ROS distribution, code-named ROS Box Turtle, is released. The distribution is now a key concept for ROS, with most users relying on a specific distribution.**
June 29, 2010	**11 PR2s Ship to Beta Program Recipients**	**With the goal of furthering ROS and robotics development in general, Willow Garage sends eleven PR2s to industry and university labs around the world**
August 3, 2010	**ROS C Turtle Release**	**ROS C Turtle, the second distribution, is released. The first batch of PR2s ship with a pre-release version of C Turtle.**
February 3, 2011	**ROS 3D Contest**	**Following the release and subsequent hacking of the Microsoft Kinect, Willow Garage sponsors a contest to spur development of cool demonstrations of what can be done with ROS and a low-cost 3D sensor**
February 15, 2011	**ROS Answers Debuts**	**To better support the burgeoning developer and user community, a dedicated Q&A forum is launched. ROS Answers is now an indispensable community resource, with over 10,000 questions.**
February 16, 2011	**ROS Day @ ISR Coimbra**	**Roboticists at the ISR University of Coimbra organize a "ROS Day" to introduce their colleagues to ROS.**
March 2, 2011	**ROS Diamondback Release**	**ROS Diamondback, the third distribution, is released.**
April 18, 2011	**TurtleBot Launch**	**To reach a broader audience than the PR2 ever could, the low-cost TurtleBot is introduced. The hardware specification is released under an open source license, allowing for individuals and companies to build, customize, and share (or sell) their own versions of TurtleBot.**
August 30, 2011	**ROS Electric Emys Release**	**ROS Electric Emys, the fourth distribution, is released.**

April 23, 2012	**ROS Fuerte Release**	**ROS Fuerte, the fifth distribution, is released.**
May 19, 2012	**ROSCon 2012**	**The inaugural edition of ROSCon, the ROS developers' conference, is held in St. Paul, Minnesota. Over 200 people come together to spend the weekend getting to know each other and discussing ROS.**
August 20, 2012	**ROS RoboCup Rescue Summer School**	**A summer school is held in Graz, Austria to foster the development of common software components and architectures for the RoboCup Rescue League based on ROS.**
September 17, 2012	**Rethink Robotics Releases Baxter**	**Rethink Robotics releases Baxter, a ROS-based robot for industry**
November 1, 2012	**Cotesys-ROS Fall School**	**Willow Garage teams up with the Technical University of Munich (with help from others) to run a week-long hands-on class to introduce students from around Europe to ROS.**
November 18, 2012	**"ROS By Example" Published**	**The first book on ROS appears, authored by Patrick Goebel.**
December 31, 2012	**ROS Groovy Galapagos Release**	**ROS Groovy Galapagos, the sixth distribution, is released.**
February 11, 2013	**ROS moves to the Open-Source Robotics Foundation**	**Responsibility for core development and maintenance of ROS transfers from Willow Garage to the still-young Open-Source Robotics Foundation.**
March 20, 2013	**ROS-Industrial Consortium kicks off**	**Spearheaded by the Southwest Research Institute, the newly-founded ROS-Industrial Consortium aims to bring ROS-based capabilities to industrial robots used in production environments.**
May 11, 2013	**ROSCon 2013**	**The second edition of the ROS developers' conference is held in Stuttgart, Germany. Approximately 300 people come together to meet and discuss ROS**
July 22, 2013	**ROS Summer School at FH Aachen**	**Localization, Mapping, Navigation, RGBD cameras, Laser Range Finders, Arduino hardware, Image processing, SLAM, IMUs and a lot more. The FH Aachen is offering a ROS Summer School for all interested students in the field of Robotics, Mechatronics and Mechanical Engineering.**
September 1, 2013	**"Learning ROS for Robotics Programming" Published**	**The second book on ROS appears, authored by Aaron Martinez and Enrique Fernández.**
September 9, 2013	**ROS Hydro Medusa Release**	**ROS Hydro Medusa, the seventh distribution, is released.**
September 16, 2013	**TEDUSAR ROS Summer School**	**The second edition of the TEDUSAR summer school is held at the University of Maribor in Slovenia.**
November 23, 2013	**ROS Workshop at Shanghai Jiatong University**	**The Future of ROS and its applications. This workshop aims to discuss recent developments within ROS, and how they can be best utilized by Robotics researchers in China.**
April 12,	**First ROS Japan User**	**Scheduled to be held every month.**

2014	**Group Meeting**	
June 6, 2014	**ROS Kong**	**The first international ROS user group meeting**
June 14, 2014	**Summer School at Middlesex University**	**Middlesex University London is running an Introduction to ROS summer school in Lundon, June 14th-18th. It will be a practical hands on class with 10 turtlebot 2 robots and a Baxter Research Robot.**
June 26, 2014	**ROS Industrial Europoe Kickoff**	**The ROS Industrial Consortium Europe heald it's kickoff event in Stuttgart Germany**
July 22, 2014	**ROS Indigo Igloo Released**	**ROS Indigo Igloo, the eigth release, is released. This is the first Long Term Support release**
September 1, 2014	**The Robonaut 2 aboard the ISS runs ROS**	**With the latest upgrades to the Robonaut 2 aboard the ISS it is now running ROS.**
September 12, 2014	**ROSCon 2014**	**The third ROS developers' conference was held in Chicago, Illinois**

4. FEATURES OF ROS [1]

ROS is the middleware operating environment. It provides below mentioned robot specific features. It also provides libraries and tools. Details are mentioned in the following table.

Features	Particulars
Message Passing	A communication system is often one of the first needs to arise when implementing a new robot application. ROS's built-in and well-tested messaging system saves you time by managing the details of communication between distributed nodes via the anonymous publish/subscribe mechanism. Another benefit of using a message passing system is that it forces you to implement clear interfaces between the nodes in your system, thereby improving encapsulation and promoting code reuse. The structure of these message interfaces is defined in the message IDL (Interface Description Language).
Recording and Playback of Messages	Because the publish/subscribe system is anonymous and asynchronous, the data can be easily captured and replayed without any changes to code. Say you have Task A that reads data from a sensor, and you are developing Task B that processes the data produced by Task A. ROS makes it easy to capture the data published by Task A to a file, and then republish that data from the file at a later time. The message-passing abstraction allows Task B to be agnostic with respect to the source of the data, which could be Task A or the log file. This is a powerful design pattern that can significantly reduce your development effort and promote flexibility and modularity in your system.
Remote Procedure Calls	The asynchronous nature of publish/subscribe messaging works for many communication needs in robotics, but sometimes you want synchronous request/response interactions between processes. The ROS middleware provides this capability using services. Like topics, the data being sent between processes in a service call are defined with the same simple message IDL.
Distributed Parameter System	The ROS middleware also provides a way for tasks to share configuration information through a global key-value store. This system allows you to easily modify your task settings, and even allows tasks to change the configuration of other tasks
Standard Robot	Years of community discussion and development have led to a set of standard message

Messages	formats that cover most of the common use cases in robotics. There are message definitions for geometric concepts like poses, transforms, and vectors; for sensors like cameras, IMUs and lasers; and for navigation data like odometry, paths, and maps; among many others. By using these standard messages in your application, your code will interoperate seamlessly with the rest of the ROS ecosystem, from development tools to libraries of capabilities.
Robot Geometry Library TF visualization	A common challenge in many robotics projects is keeping track of where different parts of the robot are with respect to each other. For example, if you want to combine data from a camera with data from a laser, you need to know where each sensor is, in some common frame of reference. This issue is especially important for humanoid robots with many moving parts. We address this problem in ROS with the tf (transform) library, which will keep track of where everything is in your robot system. Designed with efficiency in mind, the tf library has been used to manage coordinate transform data for robots with more than one hundred degrees of freedom and update rates of hundreds of Hertz. The tf library allows you to define both static transforms, such as a camera that is fixed to a mobile base, and dynamic transforms, such as a joint in a robot arm. You can transform sensor data between any pair of coordinate frames in the system. The tf library handles the fact that the producers and consumers of this information may be distributed across the network, and the fact that the information is updated at varying rates.
Robot Description Language	Another common robotics problem that ROS solves for you is how to describe your robot in a machine-readable way. ROS provides a set of tools for describing and modeling your robot so that it can be understood by the rest of your ROS system, including tf, robot_state_publisher, and rviz. The format for describing your robot in ROS is URDF (Unified Robot Description Format), which consists of an XML document in which you describe the physical properties of your robot, from the lengths of limbs and sizes of wheels to the locations of sensors and the visual appearance of each part of the robot. Once defined in this way, your robot can be easily used with the tf library, rendered in three dimensions for nice visualizations, and used with simulators and motion planners
Preemptable Remote Procedure Calls	While topics (anonymous publish/subscribe) and services (remote procedure calls) cover most of the communication use cases in robotics, sometimes you need to initiate a goal-seeking behavior, monitor its progress, be able to preempt it along the way, and receive notification when it is complete. ROS provides actions for this purpose. Actions are like services except they can report progress before returning the final response, and they can be preempted by the caller. So, for example, you can instruct your robot to navigate to some location, monitor its progress as it attempts to get there, stop or redirect it along the way, and be told when it has succeeded (or failed). An action is a powerful concept that is used throughout the ROS ecosystem.
Diagnostics	ROS provides a standard way to produce, collect, and aggregate diagnostics about your robot so that, at a glance, you can quickly see the state of your robot and determine how to address issues as they arise.
Pose Estimation, Localization, and Navigation	ROS also provides some "batteries included" capabilities that help you get started on your robotics project. There are ROS packages that solve basic robotics problems like pose estimation, localization in a map, building a map, and even mobile navigation.

Whether you are an engineer looking to do some rapid research and development, a robotics researcher wanting to get your research done in a timely fashion, or a hobbyist looking to learn more about robotics, these out-of-the-box capabilities will help you do more, with less effort.

Tools

One of the strongest features of ROS is the powerful development toolset. These tools support introspecting, debugging, plotting, and visualizing the state of the system being developed. The underlying publish/subscribe mechanism allows you to spontaneously introspect the data flowing through the system, making it easy to comprehend and debug issues as they occur. The ROS tools take advantage of this introspection capability through an extensive collection of graphical and command line utilities that simplify development and debugging.

Command-Line Tools

Do you spend all of your time remotely logged into a robot? ROS can be used 100% without a GUI. All core functionality and introspection tools are accessible via one of our more than 45 command line tools. There are commands for launching groups of nodes; introspecting topics, services, and actions; recording and playing back data; and a host of other situations. If you prefer to use graphical tools, rviz and rqt provide similar (and extended) functionality.

rviz

Perhaps the most well-known tool in ROS, rviz provides general purpose, three-dimensional visualization of many sensor data types and any URDF-described robot.

rviz can visualize many of the common message types provided in ROS, such as laser scans, three-dimensional point clouds, and camera images. It also uses information from the tf library to show all of the sensor data in a common coordinate frame of your choice, together with a three-dimensional rendering of your robot. Visualizing all of your data in the same application not only looks impressive, but also allows you to quickly see what your robot sees, and identify problems such as sensor misalignments or robot model inaccuracies.

Rqt

ROS provides rqt, a Qt-based framework for developing graphical interfaces for your robot. You can create custom interfaces by composing and configuring the extensive library of built-in rqt plugins into tabbed, split-screen, and other layouts. You can also introduce new interface components by writing your own rqt plugins

he rqt_graph plugin provides introspection and visualization of a live ROS system, showing nodes and the connections between them, and allowing you to easily debug and understand your running system and how it is structured.

With the rqt_plot plugin, you can monitor encoders, voltages, or anything that can be represented as a number that varies over time. The rqt_plot plugin allows you to choose the plotting backend (e.g., matplotlib, Qwt, pyqtgraph) that best fits your needs.

For monitoring and using topics, you have the rqt_topic and rqt_publisher plugins. The former lets you monitor and introspect any number of topics being published within the system. The latter allows you to publish your own messages to any topic, facilitating ad hoc experimentation with your system.

For data logging and playback, ROS uses the bag format. Bag files can be created and accessed graphically via the rqt_bag plugin. This plugin can record data to bags, play back selected topics from a bag, and visualize the contents of a bag, including display of images and plotting of numerical values over time.

5. SUMMARY

We have reviewed Robot Operating system (ROS) in this paper. We were able to list all the historical moments related to development of it and its globalization. The object of this paper is to spread the information about Asian Continent more specific to Indian region. We are expecting with this paper increase in the awareness about open-source contribution, research and collaboration.

REFERENCES

1. "history," [Online]. Available: http://www.ros.org/history/.
2. J. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: A survey," *Autonomous Robots*, vol. 22, pp. 101-132, 2007.