Lab 1 - R-IDE Product Description

Dominik Soós

Old Dominion University, Computer Science Dept.

CS 410 – Professional Workforce Dev. I

Professor J. Brunelle

Dec 14, 2022

Version 1 Draft

Table of Contents

1. Introduction *3*

2. R-IDE Product Description *5*

2.1. Key Product Features **6**

2.2. Major Components **8**

3. Identification of Case Study *10*

4. Product Prototype Description *10*

4.1. Prototype Architecture (Hardware/Software) **10**

4.2. Prototype Development Challenges **10**

4.3. Prototype Development Challenges **10**

5. Glossary *11*

6. References *13*

List of Figures

Figure 1 Current Developer Interface Diagram **5**

Figure 2 Solution Process Flow **7**

Figure 3 Major Functional Component Diagram of R-IDE **9**

1. Introduction

ROS (Robot Operating System) is a popular open-source framework that allows developers to easily create and integrate applications for robotics. It is designed to simplify the process of creating complex and sophisticated robot behavior by allowing developers to focus on the core functionalities of their applications (Kerr J. & Nickels K., 2012). It also has a large and growing research community of developers and users, which provides a wealth of resources and support for working with ROS.

The market for robotics is rapidly growing and is expected to continue to expand in the future. As a result, the demand for ROS and other robotics frameworks is also expected to increase. This is due to the increasing adoption of robotics in various industries, such as manufacturing, logistics, healthcare and others. ROS is well-positioned to capitalize on this growing market and is likely to continue to be a key player in the robotics industry (Garber L., 2013). According to market research, the current market value of ROS was valued around $274.2 million in 2021 (Global Robot Operating System Market Research Report, 2022). The market for ROS could potentially grow even higher, the predicted market value of ROS is expected to reach $460 million by 2030 (Global Robot Operating System Market Research Report, 2022). ROS is being used by some major companies including NASA in the Mars Curiosity Rover and at the International Space Station (Gerkey B., 2014).

NASA uses ROS for a variety of applications in its robotics research and space exploration missions. One example of how NASA uses ROS is in the development of autonomous robots that can operate in challenging environments, such as the surface of Mars (Neel V. Patel, 2021). ROS offers a set of powerful tools and libraries that allow NASA engineers to easily program and visually control these robots using Gazebo, enabling them to perform complex tasks and gather valuable data (Cardoso et al., 2020). Additionally, ROS is also

used in the development of robotic spacecraft and rovers, which are used to explore the surface of other planets and gather scientific data (Wessling B., 2022). Overall, ROS plays a critical role in NASA's efforts to advance robotics and space exploration.

Despite its many advantages, ROS also has some significant challenges that can make it difficult to use and maintain. One of the main challenges of using ROS is the need to set up and configure a ROS environment, which can be time-consuming and require specialized knowledge. This process involves installing the ROS software, creating a workspace, and setting up the environment variables. It can be especially difficult for beginners who are not familiar with the ROS architecture and conventions. Since it requires specialized knowledge, there is a steep learning curve. For new users, getting started with ROS can be overwhelming due to the large number of libraries, tools and conventions that need to be understood in order to effectively use the framework. This can lead to long development times and frustration for those who are trying to build their first ROS applications.

Another challenge of using ROS is the time-consuming debugging process. Since ROS is a complex system with many interconnected components, tracking down and fixing errors can be a daunting task. This is compounded by the fact that ROS applications need to run on multiple terminal windows shown in Figure 1, requiring developers to switch between them frequently in order to manage and debug their code. Suppose a developer is working on a ROS application that uses three nodes: a sensor node, a processing node, and an actuator node. In order to manage and debug these nodes, the developer would need to switch between three separate terminals, each running a different node. This can be a tedious and confusing process, especially if the nodes are interconnected and rely on each other to function properly.

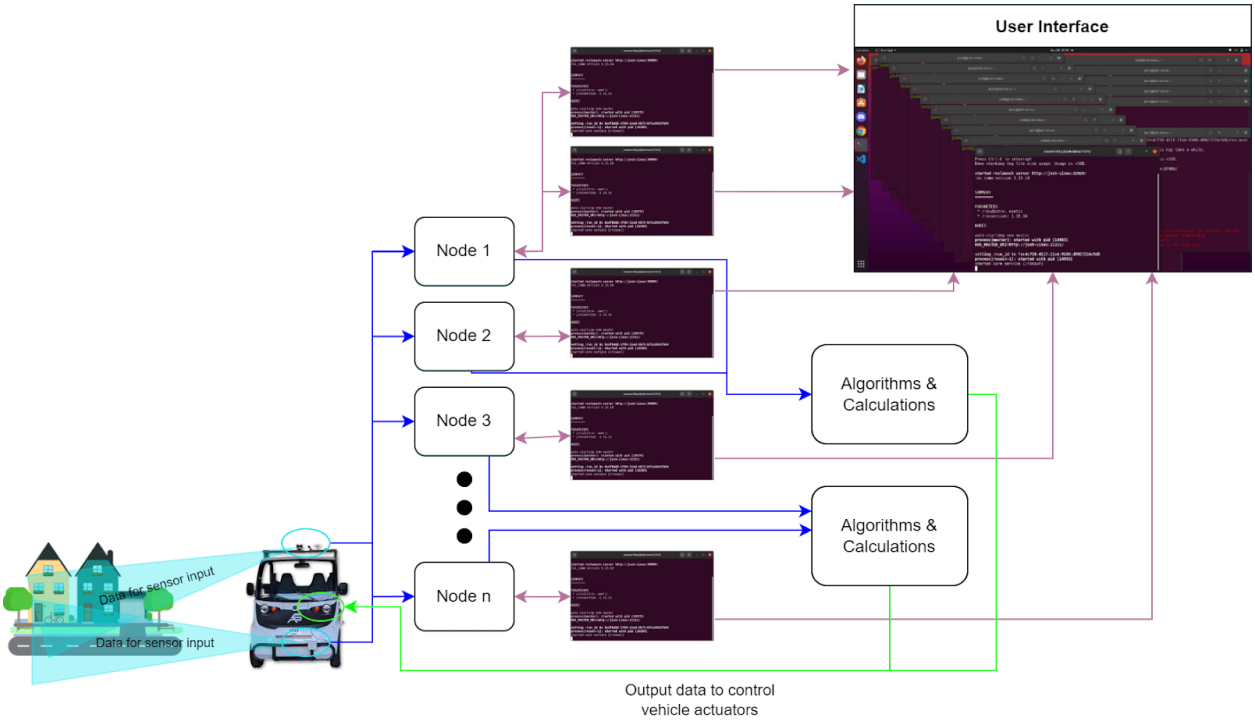

Figure 1 Current Developer Interface Diagram

Fortunately, there are tools available that can help alleviate these challenges and make ROS more accessible and user-friendly. One such tool is R-IDE, a Visual Studio Code extension that offers a simplified user interface, quick environment setup, and wizards for common tasks. In addition, R-IDE provides debugging capabilities for ROS nodes and ROS bag recording, making it easier to troubleshoot and optimize ROS applications. This report explores how R-IDE can help overcome these challenges. It looks at the features of R-IDE and how they can improve the development experience for ROS users, as well as the benefits of using this tool in the context of robotics development.

2. R-IDE Product Description

R-IDE is a powerful Visual Studio Code extension that is designed for developers working with ROS. It offers tools and features to help users quickly set up a ROS environment and develop ROS applications. The main features can be seen in Figure 2 that includes wizards

for common tasks, code autocompletion, and tools for working with ROS bag recording and visualization tools. With R-IDE, users can use the Creation Wizard to easily perform common tasks like creating new ROS nodes, creating messages and services, and updating your CMake and package.xml files.

2.1. Key Product Features

R-IDE is equipped with a comprehensive environment to speed up the development cycle for ROS applications. One of the key features is its ability to quickly set up a ROS environment within VSCode. With just a few clicks, users can get started with their ROS projects, saving valuable time and effort.

R-IDE includes Creation Wizard for common tasks like creating new ROS nodes, creating messages and services, and updating CMake and package.xml files. This makes it more time-efficient for developers to focus on the core functionalities of their projects, rather than getting bogged down in boilerplate code.

A distinct aspect of R-IDE is its support for advanced features like code auto completion and usage analytics. With these tools, users can write code more efficiently and even track their wizard usage over time. R-IDE also includes tools for starting, stopping and playing back ROS bag recordings, allowing users to easily test and debug their applications.

Other functionalities of R-IDE offer debugging capabilities for ROS nodes. This allows the user to identify and fix errors quickly and efficiently, saving them time and frustration. R-IDE's debugging tools are user-friendly and intuitive, making them easy to use even for developers who are new to ROS.

R-IDE also provides convenient visualizations for the ROS environment using RQT, Rviz, and Node Graph. This allows users to see the stature of their ROS nodes and topics, and to

quickly identify any potential issues. With R-IDE, users can also easily monitor and publish messages on ROS topics, giving them complete control over their ROS environment.

Another key feature of R-IDE is its versatility. It can be used in real-time, allowing the user to test their applications as they develop them. Alternatively, users can also use R-IDE with previously recorded data, allowing developers to test their applications using previously recorded data sets. This flexibility makes R-IDE a valuable tool for developers working on a wide range of ROS applications.

R-IDE is a must-have extension for any developer looking to streamline their workflow and improve their productivity. Its user-friendly environment and powerful debugging capabilities make it an invaluable asset for speeding up the development process and ensuring the quality of ROS applications. Whether you are new to ROS or an experienced developer, R-IDE is a valuable tool that can help build and test ROS applications more efficiently and effectively.

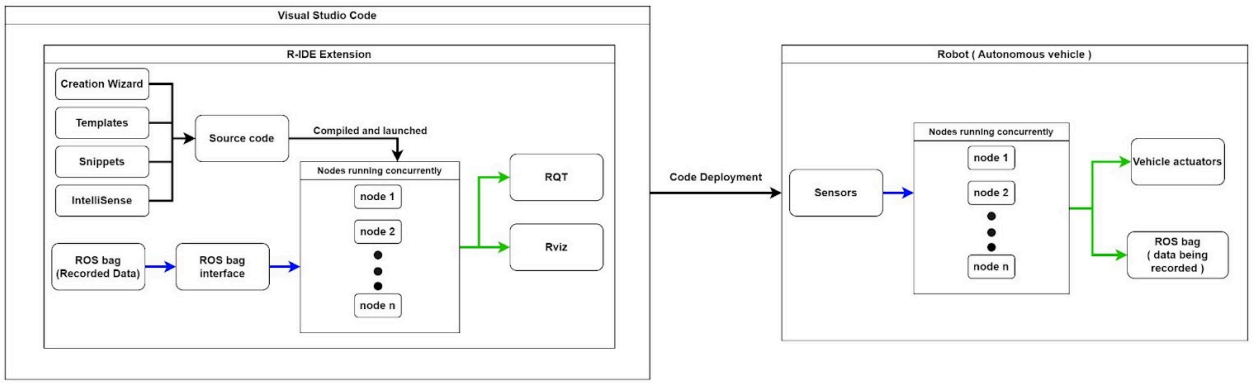

Figure 2 Solution Process Flow

2.2. Major Components

The major components of the VSCode extension R-IDE for ROS applications are the creation wizard, the VSCode analyzer, the ROS bag recorder, and the RQT and Rviz visualizers.

The Creation Wizard is a user-friendly tool that allows developers to easily create new ROS nodes, messages and services. It does this by asking the user to fill out a request form, where the user can specify the file type, node name, path to the file and other features. The filled out for is then used to automatically update the associated package files. This makes it easy for developers to quickly set up a new ROS project without having to manually create and configure all of the necessary files.

The VSCode analyzer is a helpful tool that allows developers to easily write code for ROS applications. It uses advanced algorithms to analyze the text of the code and give helpful suggestions in the form of code snippets and Intellisense. This makes it easier for developers to write correct and efficient code, and it also helps identify any errors or missing pieces in the code.

Another key component of R-IDE is the ROS bag interface. This allows users to connect their VSCode environment to a ROS bag, which is a file format used to store data from a variety of sensors and other sources. By connecting to a ROS bag, users can access and visualize the data contained within it using tools like RQT and Rviz.

The RQT and Rviz visualizers are practical tools that allow developers to see what is happening in their ROS application. The RQT visualizer supplies a node graph that shows the connections between nodes, as well as a topic monitor that allows the user to see the data being published on different topics in real time. The Rviz visualizer, on the other hand, provides a live camera feed and sensory imagery, allowing developers to see the environment in which their

ROS application is running. This is particularly useful for debugging and understanding the behavior of the nodes. Together, these tools allow users to get a better understanding of the data being collected by their sensors and other devices.

Overall, the functional components of R-IDE provide an effective and intuitive tool for working with ROS and developing source code for robotics applications. The Creation Wizard, ROS bag interface, and code analysis capabilities make it a valuable tool for any ROS developer. Figure 3 illustrates several major functional components, including the Creation Wizard and ROS bag interface. The Creation Wizard allows users to create template source code, which can then be customized to fit their specific needs. The ROS bag interface connects to ROS bags, allowing users to access and visualize the data contained within them. Additionally, the VSCode extension has the ability to analyze text and include code pattern suggestions, as well as identify errors and missing pieces in the source code.

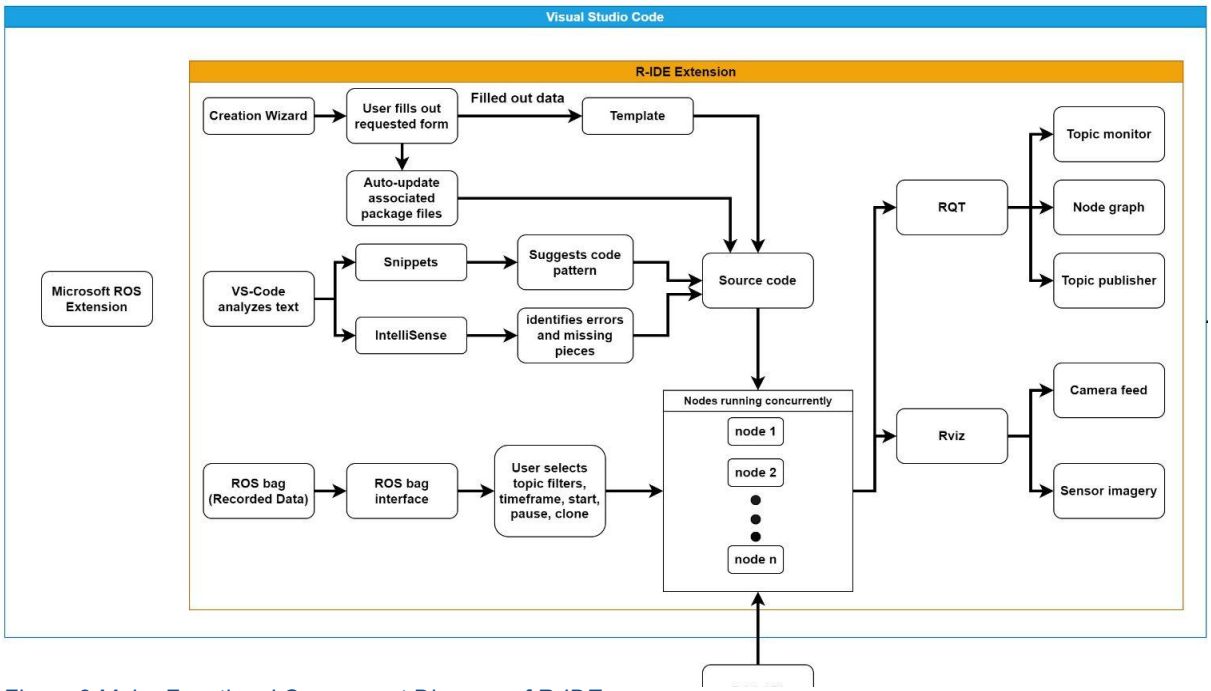

Figure 3 Major Functional Component Diagram of R-IDE

3. Identification of Case Study

The collaboration between computer science and the electrical engineering department has been instrumental in the development of R-IDE. The computer science students have provided expertise in software development and user experience, while the engineering students gave their insight into the needs and challenges of ROS developers. Together, the team has designed a useful tool that is tailored to the needs of ROS developers. Working closely with Dr. Belfore and his engineering students, who are our intended users, sought to create a solution that would help improve their productivity, simplify common tasks as well as attacking the steep learning curve of ROS. The intended use of R-IDE is to offer an intuitive yet powerful development environment for building ROS-based applications, and to supply tools that can help reduce the initial overhead of learning ROS and enable new developers to quickly become proficient in using it. We believe that R-IDE is a valuable asset for ROS developers, and we look forward to seeing it used in a variety of applications.

4. Product Prototype Description

4.1. Prototype Architecture (Hardware/Software)

4.2. Prototype Development Challenges

4.3. Prototype Development Challenges

5. Glossary

Robot Operating System (ROS): ROS is a set of software libraries that helps to build robot applications. Ranging from drivers, to algorithms, and powerful developer tools, ROS is the preferred tool for robotics projects.

ROS Node: A node is a process that performs computation. Nodes are combined together and communicate with one another using streaming topics, RPC services, and the Parameter Server. These nodes are meant to operate at a fine-grained scale; a robot control system will usually comprise many nodes. For example, one node controls a laser range-finder, one Node controls the robot's wheel motors, one node performs localization, one node performs path planning, one node provides a graphical view of the system, and so on.

ROS Bag: A bag is a file format in ROS for storing ROS message data. These bags have an important role in ROS, and a variety of tools have been written to allow you to store, process, analyze, and visualize them. Bags are the primary mechanism in ROS for data logging, which means that they have a variety of offline uses.

ROS Master: The ROS Master provides naming and registration services to the rest of the nodes in the ROS system. It tracks publishers and subscribers to topics as well as services. The role of the Master is to enable individual ROS nodes to locate one another. Once these nodes have located each other they communicate with each other peer-to-peer. The Master also provides the Parameter Server and is most commonly run using the roscore command, which loads the ROS Master along with other essential components.

ROS Parameter Server: A parameter server is a shared, multivariate dictionary that is accessible via network APIs. Nodes use this server to store and retrieve parameters at

runtime. As it is not designed for high-performance, it is best used for static, non-binary data such as configuration parameters. It is meant to be globally viewable so that tools can easily inspect the configuration state of the system and modify it if necessary.

ROS Messages: Nodes communicate with each other by publishing messages to topics. A message is a simple data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays. Nodes can also exchange a request and response message as part of a ROS service call.

ROS Services: Request / reply is done via a Service, which is defined by a pair of messages: one for the request and one for the reply. A providing ROS node offers a service under a string name, and a client calls the service by sending the request message and awaiting the reply. Client libraries usually present this interaction to the programmer as if it were a remote procedure call.

ROS Topics: Topics are named buses over which nodes exchange messages. Topics have anonymous publish/subscribe semantics, which decouples the production of information from its consumption. In general, nodes are not aware of who they are communicating with. Instead, nodes that are interested in data subscribe to the relevant topic; nodes that generate data publish to the relevant topic. There can be multiple publishers and subscribers to a topic. Topics are intended for unidirectional, streaming communication. Nodes that need to perform remote procedure calls, i.e. receive a response to a request, should use services instead. There is also the Parameter Server for maintaining small amounts of state.

Autonomous Machine: A machine capable of sensing its environment, carrying out computations to make decisions, and performing actions in the real world.

6. References

Ackerman, E. (2021, June 24). Microsoft announces experimental release of ROS for Windows 10. Retrieved November 3, 2022, from https://spectrum.ieee.org/microsoft-announces-experimental-release-of-ros-for-windows-10

Belfore, L. (jul 8, 2022). *Software Design Report for the ODU Monarch I* (Vol. 1.3, p. 3, Tech.).

Cardona, M., & Manzanares, J. (2020). *COVID-19 Pandemic Impact on Mobile Robotics Market* (pp. 1-4) (A. Palma, Ed.). IEEE. doi:10.1109/ANDESCON50619.2020.9272052

Cardoso, R.C., Farrell, M., Luckcuck, M., Ferrando, A., Fisher, M. (2020). Heterogeneous Verification of an Autonomous Curiosity Rover. In: Lee, R., Jha, S., Mavridou, A., Giannakopoulou, D. (eds) NASA Formal Methods. NFM 2020. Lecture Notes in Computer Science(), vol 12229. Springer, Cham. https://doi.org/10.1007/978-3-030-55754-6_20

Fujita, T. (2018). Tomoya Fujita R&D center Sony Corporation - roscon.ros.org. Retrieved November 3, 2022, from https://roscon.ros.org/2018/presentations/ROSCon2018_Aibo.pdf

Garber, L. (2013). Robot OS: A new day for robot design. *Computer*, *46*(12), 16-20.

Gerkey, B. (2014, September 1). Ros running on ISS. Retrieved November 3, 2022, from https://www.ros.org/news/2014/09/ros-running-on-iss.html

Global Robot Operating System Market Research Report 2022(status and outlook). (2022, June 29). Retrieved November 3, 2022, from https://www.marketreportsworld.com/global-robot-operating-system-market-21185690

Guerry, M., Müller, C., Kraus, W., & Bieller, S. (2021, October 28). IFR International Federation of Robotics. Retrieved November 3, 2022, from https://ifr.org/downloads/press2018/2021_10_28_WR_PK_Presentation_long_version.pdf

Kerr J and Nickels K., "Robot operating systems: Bridging the gap between human and robot," Proceedings of the 2012 44th Southeastern Symposium on System Theory (SSST), 2012, pp. 99-104, doi: 10.1109/SSST.2012.6195127.

Lunar Rover. (2021). Retrieved November 3, 2022, from https://www.openrobotics.org/customer-stories/lunar-rover

Neel V. Patel (2021, April 12). NASA's next lunar rover will run open-source software. Retrieved December 14, 2022, from https://www.technologyreview.com/2021/04/12/1022420/nasa-lunar-rover-viper-open-source-software/

Robot Operating System Market. (2022, August). Retrieved November 3, 2022, from https://www.futuremarketinsights.com/reports/robot-operating-system-market

Robot operating system. (n.d.). Retrieved November 3, 2022, from https://www.ros.org/

Wessling, B. (2022, February 11). Open robotics developing space ROS with Blue Origin, NASA. Retrieved November 3, 2022, from https://www.therobotreport.com/open-robotics-developing-space-ros/