Lab 2 – R-IDE Prototype Specification

Dominik Soós

Old Dominion University, Computer Science Department

CS411W - Professional Workforce Dev. II

Professor J. Brunelle

March 20th, 2023

Version 1

Table of Contents

List of Figures

List of Tables

1. Introduction

Robot Operating System (ROS) is a widely used open-source framework for building robotics applications. ROS development presents high barriers of entry for new developers and environments. The ROS system comprises many nodes running simultaneously, each within its own terminal window, making it challenging to manage changes between nodes. Furthermore, debugging and organizing the ROS interface can be inefficient and time consuming, while outdated and confusing documentation hampers the learning process. In this report, we present R-IDE, a Visual Studio Code extension that simplifies the development of process and learning curve.

1.1 Purpose

The main purpose of the R-IDE extension is to address these challenges faced by new developers when working with ROS. These challenges include managing multiple nodes simultaneously, inefficient debugging and file organization, and navigating outdated documentation. R-IDE offers a streamlined developer environment by consolidating common tasks and commands, providing access to visualization tools and organizing the developer's ROS interface.

R-IDE aims to simplify the development lifecycle and learning process for ROS developers, including Dr. Belfore and his ECE students. The extension offers a setup process, allowing developers to quickly create ROS projects. R-IDE's template system simplifies the development process even further by providing templates for creating nodes in both C++ and Python, building launch files, handling other common tasks in ROS development.

1.2 Scope

R-IDE focuses on three main areas of improvement: package management, graphical user interface (GUI) for command-line functions, and code templates for rapid ROS package development.

Automating package management helps the user efficiently create, manage and understand ROS packages by simplifying and streamlining the process. This approach enables the developer to focus on creating the project without being overwhelmed by the complexities of package management and file updating.

The GUI for command-line functions offers a user-friendly approach to accessing command-line tools. R-IDE enables the user to perform the necessary tasks with ease and efficiency, minimizing the learning curve for new ROS developers.

The R-IDE prototype functions as a real-world product, equipped with automation features and interfaces that simplify the user interaction with ROS. To ensure the extension remains up-to-date and reliable, the R-IDE development team adheres to the guidelines and updates of the dependent software, while also actively seeking feedback from the stakeholders to constantly improve the product.

1.3 Definitions, Acronyms, and Abbreviations

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

Command Palette: refers to the input box in vscode where users can enter commands

Filtering: Create new ROS bag files from older ROS bag files, filtering the new bag based on requested ROS topics from the old one.

Git: Git is a distributed version control system: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

Graph View: a graphical representation of the communications between different nodes in a ROS system. It provides a visualization of the ROS topics being published and subscribed to by each node. Graphical View is a useful tool for understanding and debugging complex ROS systems.

GUI (Graphical User Interface): type of user interface that has elements such as buttons, menus to allow the user to interact with the application.

IDE (Integrated Development Environment): application software that provides tools and features for software development. They include features like debugger, source code editor, code completion and even version control in some cases.

IntelliSense: IntelliSense is a general term for various code editing features including: code completion, parameter info, quick info, and member lists.

Monarch 1: two-passenger autonomous vehicle design based on Polaris GEM e2, which is an electric vehicle that is intended to participate in the IGVC's "Self-Drive Challenge".

MongoDB: free non-relational database that stores data in JSON format. The data is stored in collections of documents that can have nested structures for flexible data storage.

Mongoose: library for Node.js that provides a higher level of abstraction on top of MongoDB that simplifies the process of interacting with MongoDB.

RQT: A QT-based framework for GUI development for ROS. It contains tools that support ROS topics, bags, node graphs, and many other tools for visualization and manipulation of ROS nodes.

ROS Bag: a file format in ROS for storing ROS message data. An important role in ROS, and a variety of tools have been written to allow the user 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 Messages: A message is a simple data structure, comprising typed fields. Nodes communicate with each other by publishing messages to topics. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays. As part of a ROS service call, a message can be exchanged between nodes in the form of request and response.

ROS Node: executable programs to perform tasks within the robotic system. 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 Package: software in ROS is organized by packages, a collection of ROS nodes, a ROS library, a dataset, or anything else that requires a useful module for basic functionality.

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

ROS Services: provide a way for nodes to communicate with each other in a synchronous, request-response fashion . The 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 Subscriber: Node that receives data from a topic published by another node. Nodes use ROS topics to communicate, and a subscriber subscribes to a topic to receive messages that are published on that topic.

ROS Topics: 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.

Robot Operating System (ROS): 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.

Rviz: (Ros Visualization) A 3D visualizer for displaying sensor data and state information from ROS.

Snippets: A coding tool to automatically generate repeating code when developing.

Template: An editable text or code snippet that can be filled with given values from another tool like a wizard.

Topic Monitor: Displays debug information about ROS topics including publishers, subscribers, publishing rate, and ROS messages.

Tutorial: A method of transferring knowledge that teach via example and supplies the information to complete a given task.

Trimming the ROS bag time range: Reducing the duration in time of a requested ROS bag.

VSCode Extension: A tool designed to add additional features and capabilities to VSCode. It can create new dialogues, add functions, or change the appearance of VSCode.

Webview: A type of web browser that can be embedded within applications allowing it to display web content.

Wizard: A user interface that presents dialog to lead a user through a sequence of steps. Often used to configure a service for the first time or to simplify a complex or unfamiliar process.

1.4 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

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

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

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

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/

1.5 Overview

This lab report provides an overview of the R-IDE software configuration, external interfaces, purpose, and scope. It also outlines the key features of R-IDE, including its ability to simplify common tasks and commands, provides visualizations tools, and organizes the developer's interface. With R-IDE, ROS developers can expect a more streamlined and efficient development process.

2. General Description

R-IDE is a full extension for VSCode designed for developing and testing ROS applications. It is not a prototype, but a fully functional extension that provides various tools and features to help ROS developers build and debug their applications. The extension allows

developers to create new functional nodes, servers, and messages while also providing access to previously recorded ROS Bags from Dr. Belfore play as a proof of concept.

2.1 Prototype Architecture Description

R-IDE is a VS-Code extension that coexists with Microsoft's ROS extension. R-IDE features a creation wizard that requires users to fill a form to generate a code template for common ROS components such as publisher and subscriber nodes for both C++ and Python. R-IDE has the ability to generate a node that can be both publisher and subscriber. The code editor then suggests and fixes errors using customized snippets and IntelliSense, enabling users to write code quickly and efficiently.

The ROS Bag interface allows users to select, clone and manipulate and manage ROS Bags easily. The output data is then displayed through the Topic Monitor embedded within the extension. The user has the ability to manipulate a selected ROS Bag. They have the option to clone the entire or trimmed version of the ROS Bag or play the bag. While playing, they have the ability to pause or stop the ROS Bag. The ROS Topics are filtered through the ROS Bag, which can be displayed in the ROS Topic Monitor. R-IDE does not handle code deployment to robots.

The event data is being collected and stored in a MongoDB database using Mongoose to facilitate usage analytics. This process enables the developers of the R-IDE extension to track

user interaction and handle potential errors.

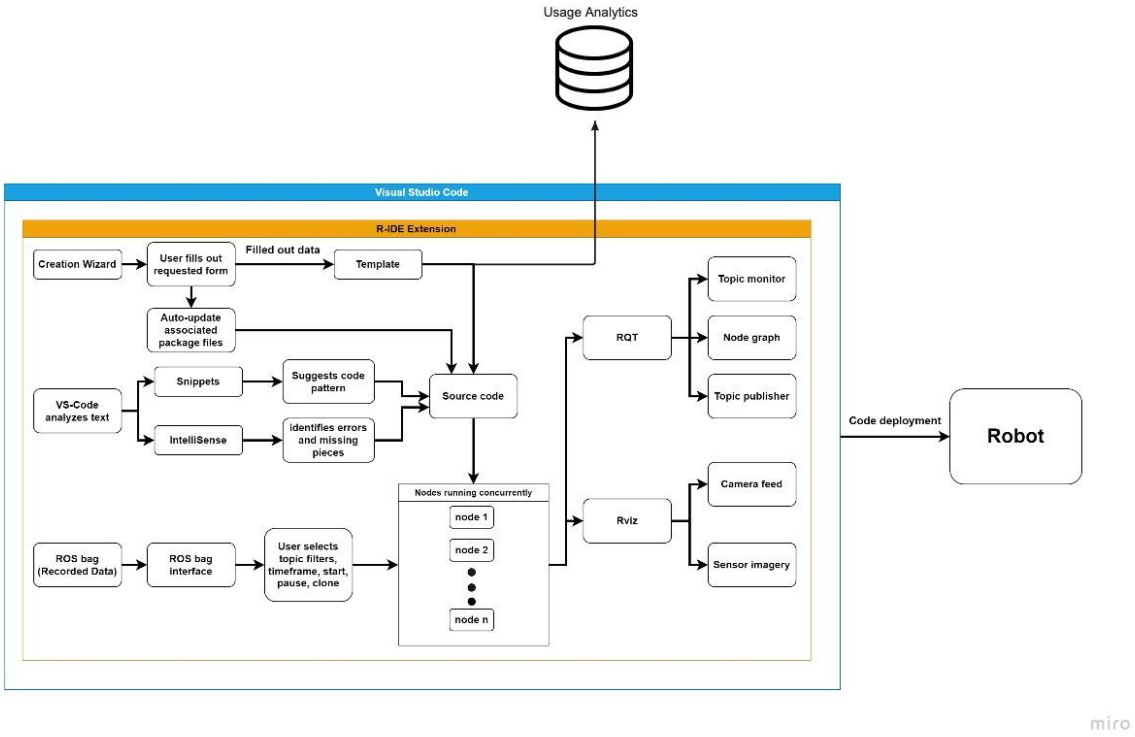

Figure 1 Major Functional Component Diagram

2.2 Prototype Functional Description

R-IDE has several features that make it a useful tool for ROS developers. R-IDE is equipped with several features that enhance the user experience when developing ROS applications. These features include the Creation Wizard for common files, autoupdate package files, autocomplete common code patterns, ROS Bag features, visualization tools, ROS Topic features and data management.

The debugging capabilities within R-IDE allows the developer to conveniently debug and terminate ROS nodes. This feature simplifies the debugging process, enabling developers to focus on their application logic and to identify issues more efficiently.

The Creation Wizard streamlines the process of creating publisher and subscriber nodes in both C++ and Python. The wizard automatically generates message and service files, saving time and reducing the likelihood of errors during the initial stages of development.

R-IDE incorporates auto update functionality for CMakeLists.txt and package.xml files, ensuring that these essential files are being up-to-date with the current ROS workspace. The autocomplete feature for common code patterns in C++ and Python leverages IntelliSense to provide suggestions and fix code based on ROS standards, enhancing the overall code quality.

Automated features include auto updating the CMakeLists.txt and the package.xml files. R-IDE also allows a feature to autocomplete common code patterns in C++ and Python for ROS. The code autocompletion feature suggests code snippets and fixes code based on ROS standards for C++ and Python using IntelliSense.

The ROS Bag interface features gives the user the ability to play or clone a selected ROS Bag from the ROS workspace. While playing a ROS bag, the user can stop or play back. When the selected ROS Bag is being cloned, the user can trim the ROS Bag by the durations of the bag.

R-IDE includes visualization tools such as RViz and Node Graph, which aid developers in understanding and analyzing the behavior within the ROS Topic Monitor. RViz lets the user view what the robot is seeing, which provides a valuable insight into where an error might occur.

The ROS Topic Monitor offers a mechanism to subscribe to a designated ROS Topic that is currently running and display the messages of the selected ROS Topic’s coming through. While running, R-IDE actively listens to the current ROS Topics and dynamically updates the ROS Topic Monitor. The ROS Topic Publisher makes it seamless to enter message data and publish it to the appropriate ROS Topics, ensuring the correct message format is being published,

streamlining the communication within the ROS ecosystem. Usage Analytics is collecting data about the Creation Wizard and uploading the data via Mongoose.

	Feature	RWP	Prototype
Wizard	Create node	Full	Full
	Create msg	Full	Full
	Create srv	Full	Full
Auto update	cmake file	Full	Full
	package.xml	Full	Full
Snippets	Autocomplete Code Patterns	Full	Full
ROS bag	Start Rosbag recording	Full	Full
	Stop Rosbag recording	Full	Full
	Play back Rosbag	Full	Full
Visuals	rviz	Full	Partial: Depending on performance of embedded features
	Node Graph	Full	Full
ROS Topic	ROS topic monitor	Full	Full
	ROS topic publisher	Full	Full
Data Management	Usage Analysis	Full	Full
Test Management	Create Mock Data	Eliminated	Full
Test Management	Automated Tests	Eliminated	Full

Table 1 Real World Product Features Table

2.3 External Interfaces

R-IDE will use specific software, and user interfaces for users to interact with all components of the application. These external interfaces include Virtual Studio Code, ROS 1, and ROS 2. Virtual Studio Code is a popular open-source code editor developed by Microsoft, while ROS 1 and ROS 2 are frameworks used to build ROS applications. These interfaces are essential for R-IDE to function correctly and efficiently.

2.3.1 Hardware Interfaces

There are no hardware interfaces for the R-IDE extension.

2.3.2 Software Interfaces

R-IDE requires specific software interfaces to function correctly. These interfaces include VScode API, MongoDB via Mongoose, ROS Bridge, ROSLib, RViz, and ROS. The VScode

API extension allows R-IDE to interact with the VS-code editor and its APIs. MongoDB via Mongoose is used by R-IDE to monitor usage analytics and collect information about the error messages occurring while creating ROS nodes. The ROS Bridge enables communication between ROS and non-ROS systems, while ROSLib provides an interface for ROS programs to communicate with. RViz is a 3D visualization tool that displays sensor data and other information from ROS topics. Finally, ROS is the core framework used to build ROS applications.

2.3.3 User Interfaces

R-IDE provides a Graphical User Interface (GUI) that allows users to interact with the application efficiently. The Svelte Framework is used to develop the GUI.

2.3.4 Communications Protocols and Interfaces

R-IDE communicates with the database to insert event data and handle errors. However, the communication protocols and interfaces used for these purposes are not specified.

3. Specific Requirements

Separate Document