Lab 2 - R-IDE Product Specification

Justin Tymkin

Old Dominion University, Computer Science Department

CS411W – Professional Workforce Development II

Professor J. Brunelle

April 21, 2023

Final Version

Table of Contents

Figures

Tables

1. Introduction

Robot Operating System (ROS) is a flexible and powerful open-source framework that is widely used in robotics and development projects such as: autonomous vehicles, industrial automation and drone development. ROS has gained significant attention in recent years due to its modular and scalable architecture that enables researchers and developers to build complex robotics systems easily. According to the Global Robot Operating System Market Research Report (2022), the robotics industry has grown significantly over the past decade, with ROS being the software of choice. The report estimates that the industry is currently worth 249.2 million and is expected to reach a value of nearly double that amount in the next decade.

Although ROS works in a multitude of developer environments, often these environments are overwhelming and frustrating for new users. The ROS system uses a node-based architecture for communication between different components. Each node is a process that communicates with other nodes using messages. Learning how to create, configure, and manage ROS nodes is challenging for new developers. Programming packages for robotics using ROS on a Linux system require multiple terminal windows, leading to a messy User Interface/User Experience (UI/UX). As a result of the complex architecture of ROS and the messy UI/UX, debugging nodes in ROS is also a challenge for new developers.

ROS-Integrated Development Environment (R-IDE) is an extension that facilitates the development of robotics and automation projects using ROS. It is designed as an IDE for ROS users, with a user-friendly interface that is accessible through the VSCode marketplace. By integrating with VSCode, R-IDE offers an intuitive and streamlined user experience for new ROS users developing robotics programs. Features included in R-IDE: wizards to allow users to quickly create components within a ROS package, the ability to record and play back data in a

ROS bag, visual tools such as Rviz and node graphs for users to visualize and understand the behavior of nodes created, and a topic monitor to track and analyze data as it flows between different nodes in a project.

1.1 Purpose

R-IDE empowers new ROS developers and helps to overcome the steep learning curve associated with ROS. R-IDE is a software extension for VSCode that facilitates the development of robotics and automation projects and is being developed as a joint effort between the Computer Science (CS) and Electrical and Computer Engineering (ECE) teams, led by Dr. Belfore. R-IDE provides an integrated development environment that caters to ROS developers, accessible through the VSCode marketplace. Its user-friendly interface streamlines the development process for new ROS developers by simplifying common tasks and commands. R-IDE also offers visualization tools, organizes the interface developed, creates a process to quickly build an environment for ROS projects, and provides templates for multiple tasks such as creating nodes or building launch files.

1.2 Scope

The R-IDE extension functions as a real-world product as it is intended for use by the ECE team and Dr. Belfore. The extension includes wizards to allow users to quickly create components within a ROS package, the ability to record and play back data in a ROS bag, visual tools such as Rviz and node graphs for users to visualize and understand the behavior of nodes created, and a topic monitor to track and analyze data as it flows between different nodes in a project. Along with these features, R-IDE automates ROS package management for things such as, CMake and .xml package files, and provide ROS code templates for fast development of ROS

nodes, services, or messages within a ROS package. By integrating R-IDE as an extension within VSCode, the CS team develops in a risk-mitigated environment. To ensure the project expectations are met, the CS team meets weekly with Dr. Belfore, and has open dialogue with the ECE team.

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/

Tymkin, J. (2023, April 21) *Lab1 – R-IDE Product Description* (Final Version)

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 product specification addresses the architecture of R-IDE and the functional description of the product. The architecture addressed includes the software configuration of R-IDE, the external interfaces of the extension in VSCode, the purposes and scope of the extension, and the features provided by R-IDE. The specification requirements for the functional and nonfunctional components of R-IDE are in a separate document.

2. General Description

R-IDE is being developed to be a full functional extension within VSCode. All features of R-IDE are to be developed within the extension, as proof of concept the extension creates functional nodes, services, and messages. Additionally, it can play back previously recorded ROS bags from Dr. Belfore as another proof of concept. Visual Studio Code handles risk mitigation for extensions by providing a curated marketplace, which is reviewed by Microsoft to ensure that extensions meet quality standards.

2.1 Prototype Architecture Description

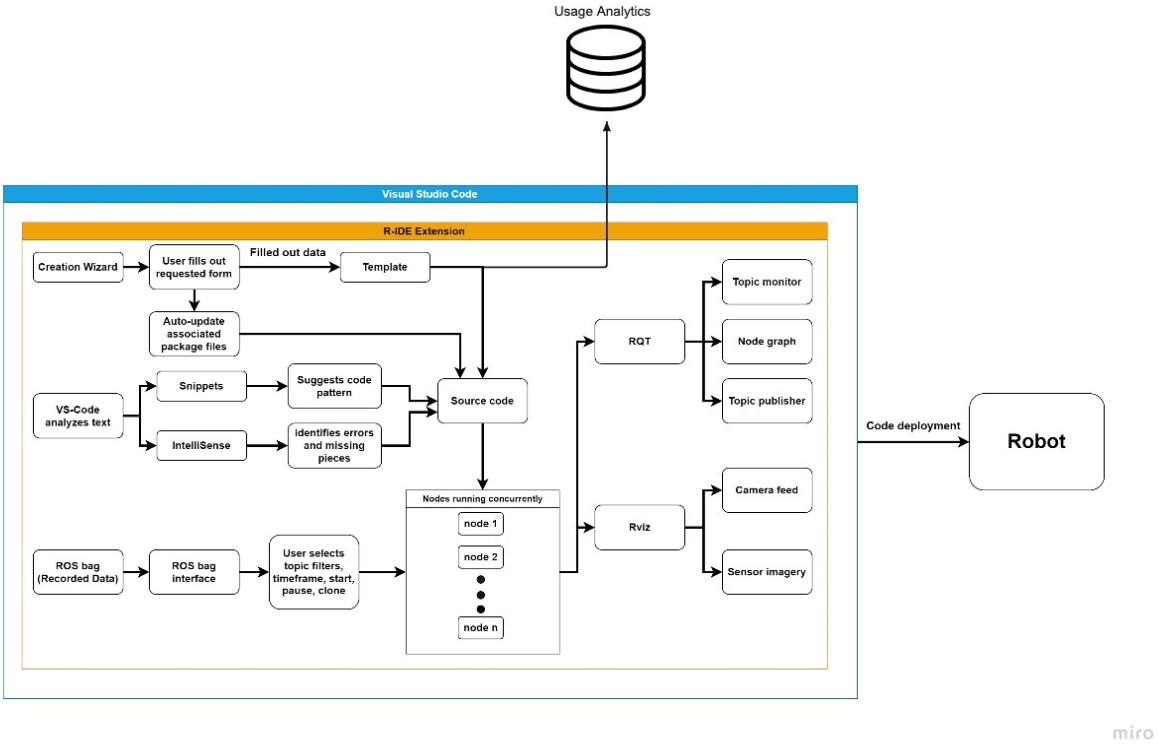
R-IDE requires VSCode and internet access to install the extension. R-IDE functions alongside the ROS extension provided by Microsoft in VSCode. The extension is being developed within VSCode using a Svelte framework with a combination of JavaScript, Typescript, HTML, and Python. The code repository is stored and managed within GitHub, a Trello board is used for project management, and Discord is used for communication between team members. Usage analytics of features used in R-IDE are recorded and stored by MongoDB via Mongoose.

Figure 1 illustrates the major components of R-IDE. The major components of R-IDE are the creation wizard for nodes, services, and messages, the ROS bag tool for playing previously recorded bags or recording a new bag, visual tools such as Rviz and node graphs for users to visualize and understand the behavior of nodes created, and a topic monitor to track and analyze data as it flows between different nodes in a project. The creation wizard walks the user through the process of creating a node, service, or message then automatically updates CMake and .xml package files. Using VSCode Intellisense and ROS snippets/templates created by R-IDE, the wizard automatically creates generic code for the selected node, service, or message. It should be noted, if the creation process is successful deploying the code to the robot is outside of the scope of R-IDE and handled separately.

Once the user has successfully created a ROS package, the ROS bag and visual tools help to determine if the package is working. The ROS bag records data collected from the package running, then the user can play it back. Rviz creates a 3D visualization of what the robot currently sees if the package is running. If the nodes are communicating, the node graph shows the user what subscriber nodes are communicating to publisher nodes, and what topics are being used to communicate between them.

Figure 1

R-IDE Major Functional Component Diagram



2.2 Prototype Functional Description

R-IDE is a full functioning extension within VSCode, all features in Table 1 exist in the R-IDE extension except for Rviz. The Creation Wizard creates either a node, service or message. Once a node, service, or message is selected to be created the wizard requires the user to select to create the selected node, message, or service in C++ or python. The Creation Wizard also asks the user for the location of the ROS package for the file created and the name of the file.

After the file for the ROS package is created, R-IDE automatically updates CMake and .xml package files. The node, service, or message created is then populated with a generic code template provided by R-IDE. While the user is developing their node, service, or message

VSCode Intellisense and R-IDE ROS snippets support the user in developing by suggesting auto complete patterns.

The ROS bag interface facilitates functionalities such as starting, stopping, and recording ROS bag files. The visual tools, including node graph, topic monitor and message publisher, provide additional insights into data collection and node communication. The node graph shows the user the active nodes running and what topics are being used to communicate between them, the topic monitor and message publisher show the topics being used and the messages they are using to communicate. Rviz was eliminated as a result of time constraints, it is possible a future team could implement this feature. R-IDE collects usage analysis data to determine popularity of implemented features. The mock data and automated testing are only available to the developers of R-IDE to test new features and ensure functionality.

Table 1

R-IDE Real World Product vs. Prototype

	Feature	RWP	Prototype
Wizard	Create node w/ templated publisher or subscriber	Full	Full
	Create msg	Full	Full
	Create srv	Full	Full
Auto update	cmake file	Full	Full
	package.xml	Full	Full
Snippets	Autocomplete Code Patterns for Srv	Full	Full
	Autocomplete Code Patterns for Msg	Full	Full
	Autocomplete Code Patterns for C++ Packages	Full	Full
	Autocomplete Code Patterns for Python Packages	Full	Full
ROS bag	Start Rosbag recording	Full	Full
	Stop Rosbag recording	Full	Full
	Play back Rosbag	Full	Full
	Filter topics from Rosbag	Full	Full
Visuals	rviz	Full	Eliminated
	Node Graph	Full	Full
ROS Topic	ROS topic monitor	Full	Full
	ROS topic publisher	Full	Full
	ROS topic media viewer	Full	Full
Data Management	Usage Analysis	Full	Full
Test Management	Create Mock Data	Eliminated	Full
	Automated Tests	Eliminated	Full

2.3 External Interfaces

R-IDE utilizes the friendly-user interfaces provided by VSCode, as well as support from separate software extensions in VSCode such as the Microsoft version of ROS. R-IDE interacts with the API of ROS 1 and VSCode, for full functionality.

2.3.1 Software Interfaces

R-IDE utilizes the API of VSCode, MongoDB, ROS Bridge, ROSLib, and ROS. The VSCode API provides editor and language services, user interface components, debugging, testing and a workspace for the CS team developing R-IDE. The MongoDB API provides a simple and intuitive data model for the CS team developing R-IDE. The ROS bridge and ROSLib API provide a WebSocket-based communication which is useful for communicating between ROS and non-ROS systems. The ROS API provides tools and libraries for building robotics systems, such as message formats and communication protocols that enable ROS nodes to communicate with each other.

2.3.2 User Interfaces

R-IDE uses a Svelte framework to build the user interface in VSCode. Svelte has a simple and intuitive way of building complex UI components, helping R-IDE developers reduce time and effort required to create a user-friendly interface.

2.3.3 Communications Protocols and Interfaces

To record event data and errors, R-IDE uses MongoDB due to its flexible and efficient capabilities for handling this type of data.

3. Specific Requirements

Separate Document