Lab 2 – R-IDE Product Specification

Daniel Koontz

Old Dominion University: Computer Science Department

CS411W: Professional Workforce Development

Professor J. Brunelle

April 11, 2023

Version 2

Table of Contents

1. Introduction

Automation and robotics are becoming the new norm in daily life. For example, vacuum cleaners can now map out a house and move on their own without intervention. Semi-autonomous vehicles are beginning to enter the mainstream market in wealthy areas. The advancement of robotics and automation is inevitable. However, the tools used to build these machines are cumbersome and require significant training to create competent and efficient developers.

Robot Operating System (ROS) is an open-source suite of software tools for building software applications used in and for robots. The primary building blocks of ROS are ROS topics, messages, and nodes. ROS nodes represent individual parts of a robot, such as a camera, brake, or computation node. These nodes programmatically subscribe and publish to named ROS topics. ROS messages are the data passed between a node and a topic. ROS messages represent simple data structures, typically composed of primitive data types such as numbers and strings as well as arrays. From this data, the user can write software to interpret and interact with the hardware components of their robot. ROS topics and messages can be recorded and placed into a file called a ROSbag and replayed later.

ROS development contains high barriers to entry for new developers and environments. This is due in large part to the unintuitive user interfaces and poor documentation. In large projects, many ROS nodes may require a large number of terminal windows cluttering a user's screen since each node requires a separate window. This cluttered environment makes it difficult to distinguish when changes in one node affect another. Thus debugging is time-consuming and inefficient. Additionally, the documentation for ROS and many of the existing tools are outdated.

Some pages contain broken and dead links to other documentation. All of these problems combined make it difficult to build a ROS environment.

1.1 Purpose

R-IDE (ROS - Integrated Development Environment) is a tool that simplifies and improves the development process for ROS applications for all developers. R-IDE exists as an extension available on Visual Studio Code (VSCode) and is available to all ROS developers. These developers may be industry professionals, university students, or robotics hobbyists. The purpose of R-IDE is to create a single workspace with a single window that contains all of the tools commonly needed by ROS developers. The first priority of R-IDE is to simplify common tasks and command-line scripts. This includes providing templates for common tasks such as creating nodes and launch files. R-IDE provides the functionality for developers to efficiently build a new environment for any ROS application. By existing in a single workspace, R-IDE can organize the workspace for developers and encourage best practices. R-IDE also features several visualization tools that can exist inside VSCode.

1.2 Scope

R-IDE provides many benefits to ROS developers. One of the primary features available to developers involves R-IDE's simplification and partial automation of package management. R-IDE has the capability to update package files automatically or from the VSCode command palette. The command line interface (CLI) for ROS is made available as a graphical user interface (GUI) to developers.

The R-IDE prototype functions as a real-world product (RWP). This is so that it can act as a case study paired with the engineering department at Old Dominion University. R-IDE

provides configurable automation features for ROS development. R-IDE provides interfaces to simplify and ease the interaction between users and ROS.

R-IDE mitigates risks by following the best practices maintained by the software industry. R-IDE follows the guidelines and updates of the libraries and software that R-IDE depends on, primarily the ROS APIs and VSCode API. Additionally, the developers of R-IDE receive constant feedback through surveys and direct discussions from the members of the engineering department at ODU. This feedback will be used to optimize the templates provided and to inform the creation of new templates.

[THIS SPACE INTENTIONALLY LEFT BLANK]

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.

[THIS SPACE INTENTIONALLY LEFT BLANK]

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

Koontz, Daniel. (2023). *Lab 1 - R-IDE Product Description* [Unpublished]. Old Dominion University. https://jpete020.github.io/team-orange/assets/files/dkLab1F.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/

[THIS SPACE INTENTIONALLY LEFT BLANK]

1.5 Overview

This product specification provides the software configuration, external interfaces, and features of R-IDE. The remaining sections provide a detailed description of each feature and its requirements.

2. General Description

R-IDE is a full extension of VSCode available on the Extension Marketplace. As a proof-of-concept, R-IDE can create fully functional ROS node, message, and service files via a wizard GUI. R-IDE provides snippets that developers can use to quickly write common pieces of code. In addition, R-IDE can play previously recorded ROSbags provided by ODU's engineering department. All of these features will be combined inside a single viewport inside VSCode.

2.1 Prototype Architecture Description

Figure 1 shows the major functional components of R-IDE. R-IDE exists as an extension of VSCode and works along with the VSCode API. There are multiple starting points for a R-IDE user.

The creation wizards allow users to fill out a form and generate a code template for common ROS components. These send updates to the database for the R-IDE developers hosted on MongoDB via the Mongoose API to ensure that these wizards function correctly. As files are created, R-IDE's package management system updates the CMakeLists.txt and package.xml files so that the node is compiled correctly in the future.

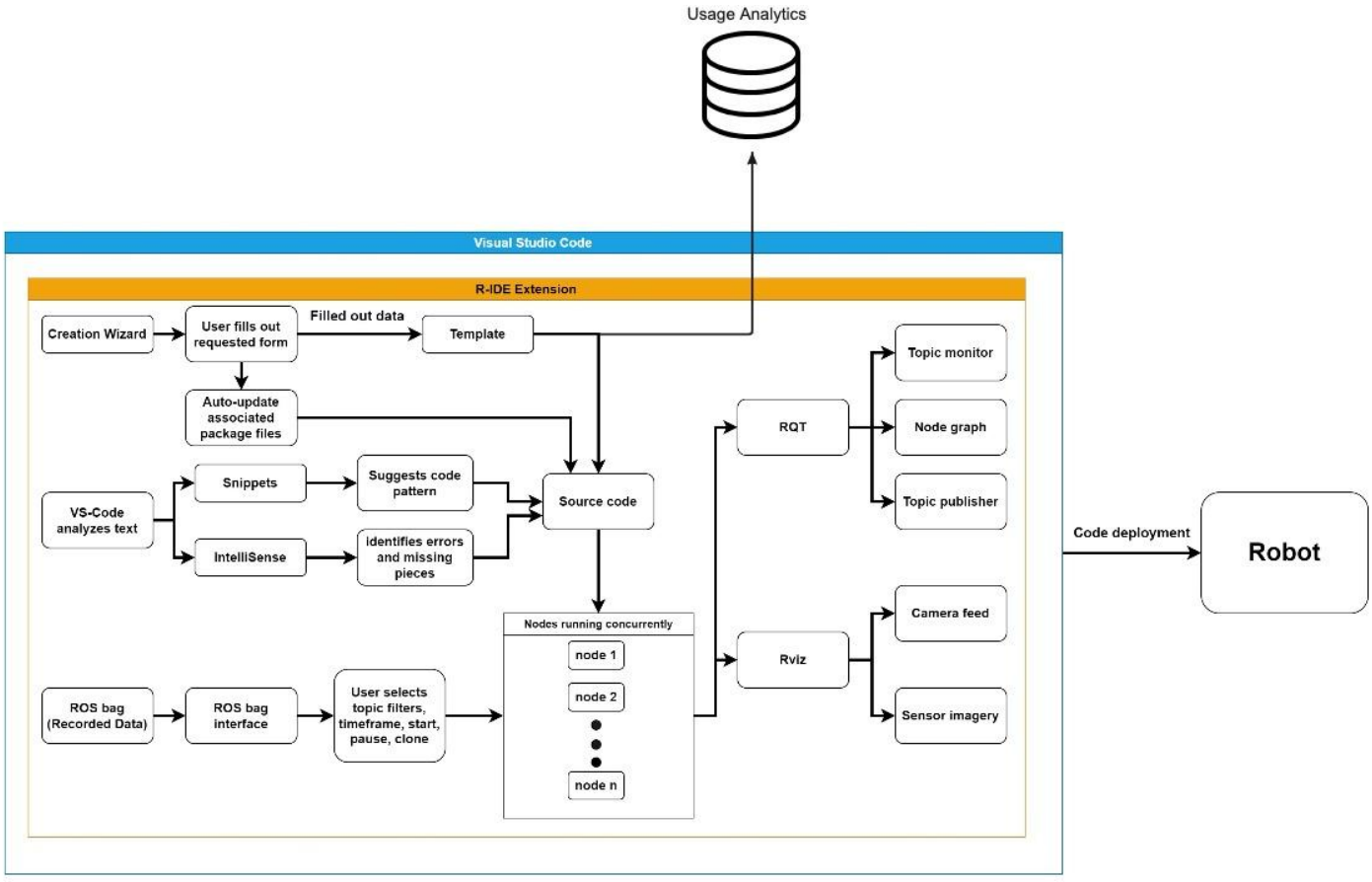
The text editor contains two tools to allow users to efficiently read and write code. Snippets are provided for node files, allowing users to quickly build ROS nodes that can publish

and subscribe to topics. The VSCode Intellisense feature provides basic syntax highlighting and error detection to the user in the VSCode text editor.

Recordings of the message passing, otherwise known as ROS bags, can be managed in the ROSbag interface. The contents of the bag can be monitored via the visualization tools, such as the ROS Topic Monitor. Important features and tools that exist in RQT and RViz are cloned in the visualization view.

At the user's discretion, the source code can be compiled and deployed to the robot. R-IDE does not assist in the code deployment process, although this functionality may exist in other extensions available on the Extension Marketplace.

Figure 1. Major Functional Component Diagram



2.2 Prototype Functional Description

The key features of R-IDE are included in the prototype. The main features of R-IDE include the automation of package management and the ability to interact with the ROS components inside VSCode. Table 1 lists the features to be included in the R-IDE prototype.

Table 1. R-IDE Feature Description and Prototype Implementation

	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	Partial: Depending on performance of embedded features
	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

Since the R-IDE prototype functions as a RWP, all features are fully implemented. Some visualization tools may face limitations depending on both the complexity of the user's projects, as well as the machine the user may be developing on.

2.3 External Interfaces

R-IDE uses specific software and user interfaces for users to interact with all components of the application. The ROS and VSCode APIs are required in order to provide functionality. All communication done through the application will be done using secure protocols and ports.

2.3.1 Software Interfaces

R-IDE uses many ROS APIs for JavaScript, the VSCode API, and the Svelte frontend framework. This includes roslibjs, cruise-automation/rosbag, rosbridge, rospack, and mongoose for communication with MongoDB.

2.3.2 User Interfaces

R-IDE is accessed as a VSCode Extension. The size of the window can be adjusted. There are multiple view panels on the lefthand side where the wizards, visualization tools, ROSbag interface, and topic manager tools can be accessed.

2.3.4 Communications Protocols and Interfaces

The application will use the protocols and encryptions provided by ROS and MongoDB for all communications. This includes inserting event data and error handling.

3. Specific Requirements

** Section 3 is contained in another document **

Appendices

** The appendices are contained in another document **