LAB 3 – RIDE PROTOTYPE TEST PLAN

Team Orange

Old Dominion University: Computer Science Department

CS411W: Professional Workforce Development

Professor J. Brunelle

April 04, 2023

Version 1

Table of Contents

1. Snippets (Gavin St. Clair)

Test Category: Algorithms		**Description:** Ensure all of the snippets function		
Test Case:		**Case Name:** Snippets	**Version:** 1.0	**Written By:** Gavin St. Clair
Requirements Fulfilled: 3.1.2.2		**Purpose:** Ensure that R-IDE correctly generates snippets into new files and generates user selected snippets from the snippets catalog.		
Setup Conditions: 1. VSCode, Microsoft ROS, and R-IDE are installed and launched 2. The selected workspace folder contains a ROS package recognizable by R-IDE				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	From the R-IDE Creation Wizard select create new and select Ros Msg. Type in desired name, select location, then click next.			There should be a new message file created with the given name and auto populated with the default ROS message snippet.
2	From the R-IDE Creation Wizard select create new and select Ros Srv. Type in the desired name, select location, then click next.			There should be a new service file created with the given name and auto populated with the default ROS service file snippet.
3	From the R-IDE Creation Wizard select create new and select Ros Node. Type in			There should be a new cpp file created with the given name and

	the desired name, select location, then click next.			auto populated with the default cpp file snippet.
4	From the R-IDE Creation Wizard select create new and select Ros Node. Click the file type drop down and select python. Type in the desired name, select location, then click next.			There should be a new python file created with the given name and auto populated with the default python file snippet.
5	From the snippets catalog find the desired snippet for current file type. Type in the snippet name and select it from the drop down			The snippet should populate, and cursor location should be set in a logical place to begin development

3.1.1.2. Wizard View (Joshua Peterson)

The system shall provide the user with the ability to create ROS nodes, ROS messages, and ROS services containing code that is automatically generated via information gathered from the user. The system shall utilize predefined templates that consist of snippets to populate the respective file.

The following functional requirements shall be met:

1. The system shall provide a “Create new” button with a drop-down menu that has the following options:
    a. ROS Node
    b. ROS Msg
    c. ROS Srv
2. Upon making a selection from the list, the user shall be navigated to the Creation Wizard.
3. The Creation Wizard GUI shall require the following input from the user:
    a. File Type - Dropdown selections:
        i. Python

        ii. C++
        iii. Msg
        iv. Srv
    b. File Name (File Name) - String
        i. The system shall populate the file extension automatically
        ii. The system shall verify the extension was not already included in the user-specified name
            1. The system shall display an error if an extension is detected
    c. Node Location (Package Selection)
        i. The system shall present a menu of all available packages for the user to select
            1. The system shall display an error if no package is selected
    d. Publisher (Checkbox) - Boolean
    e. Subscriber (Checkbox) - Boolean
4. The Creation Wizard GUI shall provide the following buttons:
    a. Cancel Button: The cancel button shall close the wizard
    b. Submission Button: The submission button shall generate a file in the selected location with the selected name and file type
5. The contents of the file shall be generated automatically from snippets (see 3.1.2.2) depending on:
    a. File type
    b. Publisher Selection
    c. Subscriber Selection

2. Auto-Update CmakeLists.txt (Dan Koontz)

Test Category: Automation / Algorithms		**Description:** Auto-Update CMakeLists.txt		
Test Case:		**Case Name:** Update CMakeLists.txt	**Version:** 1.0	**Written By:** Daniel Koontz
Requirements Fulfilled: 3.1.2.1		**Purpose:** Ensure that R-IDE correctly updates the CMakeLists.txt file while the user edits and updates the other files within the package		
Setup Conditions: 1. VSCode, Microsoft ROS, and R-IDE are installed and launched 2. The selected workspace folder contains a ROS package recognizable by R-IDE 3. The selected ROS package contains a correctly formatted CMakeLists.txt file				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	From the R-IDE Creation Wizard, name and create a ROS message file in the desired package.			The message path should exist under \`add_message_files\`. \`find_package\` should contain \`message_generation\`. \`generate_messages\` should be uncommented.
2	From the R-IDE Creation Wizard, name and create a ROS service file in the desired package.			The service path should exist under \`add_service_files\`. \`find_package\` should contain \`message_generation\`. \`generate_messages\` should be uncommented.
3	From the R-IDE Creation Wizard, select new ros node and select a python node. Give the node a name and select any			The \`catkin_install_python\` file should be uncommented and

	other attributes of the node. Click next to complete creating the wizard.			contain the path to the python file.
4	From the R-IDE Creation Wizard, select new ros node and select a cpp node.Give the node a name and select any other attributes of the node. Click next to complete creating the wizard. Enter the command palette (Ctrl + shift + P) and select `R-IDE: Add an executable`. Select the package and the newly created source file and give the node an appropriate name			There should exist in the CMakeLists.txt an `add_executable` call with the given name and source files. Above it should be the commented name of the node and below should be a `target_link_libraries` call with the name.
5	From the R-IDE Creation Wizard, select new ros node and select a cpp node.Give the node a name and select any other attributes of the node. Click next to complete creating the wizard. Enter the command palette (Ctrl + shift + P) and select `R-IDE: Add a library`. Select the package and the newly created source file and give the node an appropriate name			There should exist in the CMakeLists.txt an `add_library` call with the given name and source files. Above it should be the commented name of the node and below should be a `target_link_libraries` call with the name.
6	Enter the command palette (Ctrl + shift + P) and select `R-IDE: Add a new package to the find package call`. Select the desired package and select the updated contents of the `find_package` call			The contents under the `find_package` call should be updated to reflect the user selection and should all be valid packages available on the system.
7	Delete a .msg file that exists in the `add_message_files` call in CMakeLists.txt			The file should no longer be listed
8	Delete a .srv file that exists in the `add_service_files` call in CMakeLists.txt			The file should no longer be listed
9	Delete a .py file that exists in the `catkin_install_python`` call in CMakeLists.txt			The file should no longer be listed

3.1.2.1 Update CMakeLists.txt (Daniel Koontz)

The system shall update the CMakeLists.txt file when changes to the current workspace package are made and require changes to the CMakeLists.txt file.

The following functional requirements shall be met:

1. The package shall monitor and notify the system for the following events:
    A. A file is created
    B. A file is updated
    C. A file is deleted
2. The package shall monitor files with the following file extensions:
    A. .msg
    B. .srv
    C. .py
    D. .cpp
3. The system shall identify if the contents of CMakeLists.txt need to be modified
4. The system shall change one or more of the following sections of the CMakeLists.txt file (As described by http://wiki.ros.org/catkin/CMakeLists.txt#overall_structure_and_ordering) :
    A. Required CMake Version
    B. Package Name
    C. Finding other packages needed to build
    D. Enabling Python module support
    E. Message/Service/Action generators

   F. Invoking message/service/action generation
   G. Specifying package build info to export
   H. Libraries/Executables to build
   I. Tests to build
   J. Install rules
5. R-IDE shall not inhibit the user from adding additional content to CMakeLists.txt.
6. The system shall be made configurable so as to change auto-updates to CMakeLists.txt from being handled on save, to being handled on command.

3. User Interface (Joshua Peterson)

Test Category: User Interface	**Description:** Main RIDE GUI renders and displays upon clicking icon button		
Test Case: 3	**Case Name:** RIDE's Main GUI	**Version:** 1.0	**Written By:** Joshua Peterson
Requirements Fulfilled: 3.1.1 3.1.1.1	**Purpose:** Ensures the activity bar within VSCode contains an icon button that renders the main RIDE GUI.		
Setup Conditions: 1. VSCode Installed 2. Microsoft ROS extension installed 3. RIDE extension installed 4. VSCode Launched			
Test Case Activity	**Pass/Fail**	**Comments**	**Expected Result**

1	Locate and click the RIDE icon button. The RIDE icon button is located on the left side of the screen on VSCode's activity bar. The icon displayed should look like the head of a robot.			VSCode's Primary side panel opens containing RIDE's main GUI
2	Within the main RIDE GUI, click the "WIZARDS" button to expand the Wizards view.			Nested view expands containing Creation Wizard UI
3	Click the "WIZARDS" button again to retract the Wizards view			Nested view retracts and hides Creation Wizard UI
4	Click the "ROS BAGS" button to expand the ROS Bags view			Nested view expands containing the ROS bags UI
5	Click the "ROS BAGS" button again to retract the ROS bags view.			Nested view retracts and hides the ROS bags UI
6	Click the "VISUALIZATION TOOLS" button to expand the Visualization Tool view.			Nested view expands containing the Visualization tools UI
7	Click the "VISUALIZATION TOOLS" button again to retract the Visualization Tool view.			Nested view retracts and hides the Visualization Tool view
8	Click the "ROS TOPIC TOOLS" button to expand the ROS topic tools view.			Nested view expands containing the ROS topic tools UI
9	Click the "ROS TOPIC TOOLS" button to again to retract the ROS topic tools view.			Nested view retracts and hides the ROS topic tools UI

3.1.1 User Interface (Joshua Peterson)

The user interface for RIDE shall be integrated into Visual Studio Code (VSCode) as an extension. The system shall utilize webviews to provide Graphical User Interfaces in the following supported VSCode containers (subset):

1. Activity Bar
2. Primary Sidebar
3. Editor
4. Functionality of VSCode containers shall not be reduced

3.1.1.1. Main RIDE GUI (Joshua Peterson)

The system shall provide an icon button on VSCode's activity bar that opens the system's main GUI in VSCode's primary sidebar.

The following functional requirements shall be met:

1. RIDE's main GUI shall contain nested webviews
2. RIDE's nested webviews shall be expandable/contractible
3. RIDE's main GUI shall be within VSCode's primary sidebar containing the following clickable options as part of the expandable options:
    A. Wizards
    B. ROS Bags
    C. ROS Topic Tools
    D. Visualization Tools

4. Node Graph (Justin Tymkin)

Test Category: Visualization		**Description:** This test verifies that the Node Graph in the Visualizations tab functions properly		
Test Case: 1		**Case Name:** Node Graph	**Version:** 1.0	**Written By:** Justin Tymkin
Requirements Fulfilled: 3.1.1.3.1		**Purpose:** To ensure the Node Graph opens with the active nodes in the user environment		
Setup Conditions: 1. Launch VSCode 2. Click the R-IDE icon in VSCode to open the extension 3. Run a node 4. Click the Visualization tab to open the Node Graph				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Click on the Node Graph button in the Visualization tab			A new tab is opened in VSCode with an empty webview
2	Ensure Rosbridge is running			Window pops up asking user if they want to start Rosbridge, if not already running select start Rosbridge
3	Check if Rosbridge is running			Select the refresh button at the top of the webview, three nodes should appear as circles, inside the circles the names rosbride_websocket, rosapi, and rosout should appear with lines between the nodes showing the topic being used to communicate between the nodes, the line should be labeled as

				rosout, this shows Rosbridge running in the user environment
4	Execute a publisher node			Select the refresh button, a circle is created in the webview to represent the publisher node with the name of the node in the circle
5	Check for the correct name of publisher node			The name of the node will appear as the name entered in the initialize node function defined by the use
6	Check for the correct name of topics being used for communication of publisher node			A line will be created to the rosout node labeled rosout to show the publisher node is communicating with Ros API
7	Execute the corresponding subscriber node for the publisher node			Select the refresh button, another circle is created in the webview to represent the subscriber node with the name of the node in the circle
8	Check for correct name of subscriber node			The name of the node will appear as the name entered in the initialize node function defined by the user
9	Check for the correct name of topics being used for communication of subscriber node			A line connecting the nodes is also created with the name of the topic used to communicate between the two nodes on the line, the line will be labeled as the name defined by the user in

				the publisher and subscriber functions, a line will be created to the rosout node labeled rosout to show the subscriber node is also communicating with Ros API
10	Execute numerous publisher or subscriber nodes			Repeat test case numbers 4 - 9 with different nodes

3.1.1.3 Visualization (Rviz & Node Graph) (Justin Tymkin)

The system shall provide the user tools that visualize information from their current ROS workspace. The user shall be able to utilize a button on the sidebar to select:

1. Node Graph

3.1.1.3.1 Node Graph

The Node Graph shall present a visualization of the active publisher node, the subscriber nodes communicating with the publisher node, and the topics being used to communicate between the publisher and subscriber nodes. The nodes will appear as circles in the webview. The topics will appear as the lines connecting the nodes, to display the connection being used between nodes.

The following functional requirements shall be met:

1. The system shall automatically recognize current active nodes/topics.
2. The system shall create a webview inside of VS Code.
3. The system shall open the Node Graph in the webview.
4. The system shall display any nodes being executed as circles in the webview.
5. The system shall display any topics being used by nodes as lines connecting the circles.

5. ROS Topic Monitor (Dominik Soos)

Test Category: ROS Topic Monitor		**Description:** This test will verify that the system allow subscribers to choose the ROS Topics they would like to monitor		
Test Case: 5		**Case Name:** Choose ROS Topics to subscribe	**Version:** 1.0	**Written By:** Dominik Soós
Requirements Fulfilled: 3.1.1.4 3.1.1.4.1		**Purpose:** The purpose of this test is to ensure that a user can choose the ROS Topic they would like to monitor. By presenting users with a dynamic list of currently executing ROS Topics, the system ensures that the user has the most up-to-date information. This feature simplifies the process of subscribing to ROS Topics and enhances the overall user experience		
Setup Conditions: 1. Install VSCode, Microsoft ROS and R-IDE in a Linux Environment (WSL, Ubuntu 20.04) 2. Launch VSCode and R-IDE within 3. Open ROS Workspace within VSCode 4. Click on ROS Topic Monitor tab to make sure ROS is connected using ROS-Bridge				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Click on the boxes on the left panel of the ROS Topic Monitor to listen to ROS Topics currently running. Verify if the selected boxes turn to a blue checkmark and the corresponding ROS Topic is displayed in the middle section.			The selected box should turn to a blue checkmark. Display the correct ROS Topic that was selected under the middle section of the ROS Topic Monitor.
2	Click on the arrow to open the message data from the selected ROS Topic. Verify if the message data is displayed correctly for the selected ROS Topic.			Display the correct file format for each subscribed topic and display the values coming through
3	Select multiple boxes on the left panel to subscribe to multiple ROS Topics simultaneously. Verify if the boxes turn into blue checkmarks and the corresponding ROS topics are displayed			The box should turn to a blue checkmark and the selected ROS Topic shall appear under

	in the middle section. Click on another box to check for the functionality to allow users to subscribe to multiple ROS.			the middle section of the ROS Topic Monitor.
4	Click on each subscribed ROS Topic in the middle section of the extension to expand and view their messages. Verify if each ROS Topic maintains a correct tree structure upon expansion. Verify if the correct messages are displayed for each selected ROS Topic.			The messages that are specific to each ROS Topic, should be displayed with proper formatting and real-time updates. The displayed messages should match the expected message types and content for the corresponding ROS Topic that was selected.
5	Unsubscribe from the selected ROS Topics by clicking on the boxes with blue checkmarks in the left panel. Verify if the system unsubscribes correctly and removes the corresponding ROS Topics from the middle section.			The selected ROS Topic should not be displayed after unchecking the box on the left.
6	Verify if the ROS Topic Monitor maintains the subscribed state of each ROS Topic even after a page refresh or application restart.			After application restart, the ROS Topic Monitor should not maintain the subscribed state from previous sessions.

3.1.1.4 ROS Topic Monitor (Dominik Soós)

The system shall offer a method for subscribing to a designated ROS Topic. While running, the ROS Topic Monitor shall actively listen to ROS Topics. *The following functional requirements shall be met:*

1. The system shall allow subscribers to choose the ROS Topic they would like to monitor.
    a. Currently executing ROS Topics shall be options to subscribe to.
    b. Subscribing to multiple ROS Topics simultaneously.
2. The system shall provide functionality to display messages through ROS Topics that are currently exchanging messages.
    a. ROS Topics from ROS Bags

3. The system shall provide the functionality to handle raw image and stereo data:
    a. Left camera
    b. Right Camera
    c. Stereo pair
    d. Depth and point cloud
    e. Tracking
    f. Mapping
    g. Sensor data
    h. Object Detection

3.1.1.5 ROS Topic Publisher (Dominik Soós)

The system shall offer a method for publishing messages to chosen ROS Topics. It shall dynamically return the ROS Topic type format.

The following functional requirements shall be met:

1. The system shall provide the interface to publish messages to ROS Topics.
    a. Currently executing ROS Topics shall be options to publish messages to.
2. The system shall provide the functionality to publish data in every ROS Topic type.
3. The system shall have a user interface that makes it seamless to enter message data in a format that is compatible with the selected ROS Topic.
4. The system shall provide the functionality to specify the frequency of messages being published to the subscribed ROS Topics.

Additional Test Cases:

Test Category: ROS Topic Monitor		**Description:** This test will verify that the system provides the functionality to listen to messages through ROS Topics and from ROS Bags.		
Test Case: 2		**Case Name:** Listen to messages from ROS Topics	**Version:** 1.0	**Written By:** Dominik Soós
Requirements Fulfilled: 3.1.1.4 3.1.1.4.2		**Purpose:** The purpose of this test is to ensure that a user can choose the ROS Topic they would like to monitor and display the message exchanges between the nodes. The test also ensures that the messages are displayed when a ROS Bag is selected in the ROS Bag Manager.		
Setup Conditions: 1. VSCode, ROS, R-IDE installed. 2. Open Linux environment (WSL, Ubuntu 20.04) 3. Open ROS Workspace and R-IDE 4. Click on ROS Topic Monitor tab and make sure ROS is connected				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	In ROS Bag Manager, click on the Manage Bags tab.			Option to select a ROS bag should become available if ROS is connected
2	Click on the Select Bag button			Since ROS is connected, the extension should navigate the user to select the ROS bag in the catkin workspace
3	Click on the refresh button			New ROS Topics should be available
4	Click on the boxes to listen to ROS Topics currently running.			Display the correct file format for each subscribed topic and display the values coming through
5	Click on Play button to observe the incoming messages			Update the messages as the time passes in the ROS Bag

Test Category: ROS Topic Monitor	**Description:** This test will verify that the system provides the functionality to listen to messages through ROS Topics that are being listened to from the ROS Bag.		
Test Case: 3	**Case Name:**	**Version:** 1.0	**Written By:** Dominik Soós

	Listen to messages from ROS Topics		
Requirements Fulfilled: 3.1.1.4 3.1.1.4.3	**Purpose:** The purpose of this test is to ensure that the extension can handle raw image and stereo data, which is then displayed to the user using visualization tools.		
Setup Conditions: 1. VSCode, ROS, R-IDE installed. 2. Open Linux environment (WSL, Ubuntu 20.04) 3. Open ROS Workspace and R-IDE 4. Click on ROS Topic Monitor tab and make sure ROS is connected			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	In ROS Bag Manager, click on the Manage Bags tab.			Option to select a ROS bag should become available if ROS is connected
2	Click on the refresh button			New ROS Topics should be available to subscribe and publish messages to.
3	Click on the boxes to listen to the currently executing ROS Topics			Display the correct file format for each subscribed topic and display the values coming through

Test Category: ROS Topic Publisher	**Description:** This test will verify that the system provides the interface and functionality to allow subscribers to choose the ROS Topics they would like to publish messages to.		
Test Case: 4	**Case Name:** Correctly publish to ROS Topics	**Version:** 1.0	**Written By:** Dominik Soós
Requirements Fulfilled: 3.1.1.5 3.1.1.5.1 3.1.1.5.2	**Purpose:** To ensure that a user can choose the ROS Topic they would like to subscribe to and publish messages in real-time that are displayed in the ROS Topic Monitor.		
Setup Conditions: 1. VSCode, ROS, R-IDE installed. 2. Open Linux environment (WSL, Ubuntu 20.04) 3. Open ROS Workspace and R-IDE 4. Click on ROS Topic Monitor tab			

5. Handle when ROS is not connected.
6. To ensure ROS is connected, click on the refresh button.

Test Case Activity		Pass/Fail	Comments	Expected Result
4	Click on the pen to publish to ROS Topics that are currently running.			Message Publisher should populate the field with correct format based on the selected ROS Topic.
5	Insert the messages in the Message Publisher and set the frequency			Message Publisher should publish the correct messages to the selected ROS Topics.
6	Select another ROS Topic to check if the Topic Publisher supports multiple message types			Display the correct format for each ROS Topic based on which one was selected, supporting multiple message types