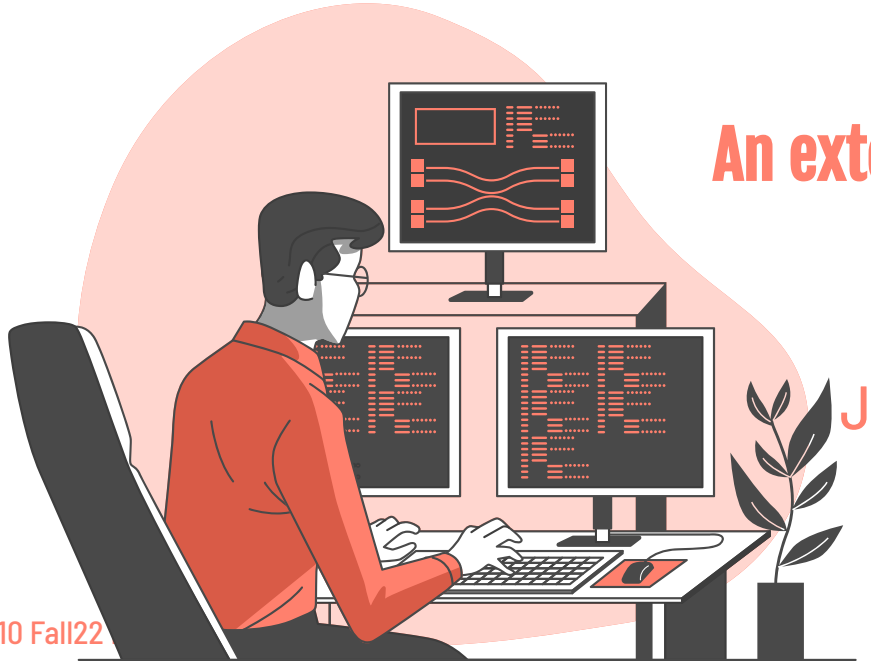
R-IDE

An extension to simplify robotics development with ROS



CS 410 - Design
Josh Peterson, Dominik Soós,
Dan Koontz, Gavin St.Clair,
James Hart, Justin Tymkin

Table of Contents

Feasibility

Design

The Team

Josh Peterson
Team Lead, Webmaster,
Frontend UI/UX Specialist

Dominik Soós
Database Specialist

Dan Koontz
Backend/Algorithm Specialist

Dr. Lee Belfore
Mentor

Laura Sakos
Mentor

Gavin St. Clair
Database Specialist

Justin Tymkin
Backend/Algorithm Specialist,
Documentation Specialist

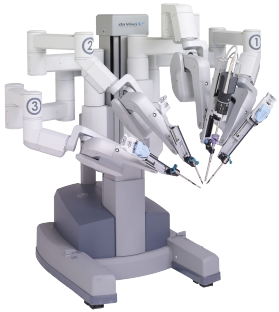
James Hart
Frontend UI/UX Specialist

Gavin Alberghini
Mentor

Robots & Autonomous Machines

- "Recent reports … expect that the mobile robots market is going to grow at a [rate of] 24% per year from the US$19 billion in 2018 to US$23 billion by 2021 and further more to US$54 billion in 2023”[9]

- Data published by the International Federation of Robotics suggests that the number of annual installations of industrial robots will increase from 315,000 in 2022 to 370,000 by 2024 in Asia/Australia, and from 50,000 to 60,000 in America during the same period.



Robot Operating System (ROS)

- The Robot Operating System (ROS) is an open source set of software libraries and tools that help developers build robot applications. [1]

- ROS estimated current market value
  - 2022 ≈ $270 million [2]

- ROS projected market value
  - 2028 ≈ $380 million [2]
  - 2032 ≈ $460 million [3]

Industries that utilize ROS:

- Autonomous vehicles
- Aerospace
- Health Care
- Agriculture
- Manufacturing
- Military

Some companies that use ROS-based robotics:

- NASA [4 & 5]
  - Curiosity Rover
  - VIPER program
  - Robonaut 2 [8]
- Microsoft [6]
- SONY [7]

IGVC: A collaborative effort

- ODU's autonomous vehicles utilize ROS to convert raw sensor data, perform calculations, and make intelligent decisions on the vehicles upcoming route.

- With the combined efforts of the following departments and their respective mentors, ODU hopes to place in the top 6 for the Self-drive challenge at the Intelligent Ground Vehicle Competition (IGVC).
  - Electrical and Computer Engineering
  - Computer Science
  - Mechanical and Aerospace Engineering

ROS Nodes

- A node is a process that performs computation, an executable file within a ROS package.
- ROS nodes are combined together in a graph and communicate with each other using topics, services, and Parameter server.
- To create a robot application with ROS requires many nodes. The LiDAR alone requires 3-5 nodes and services.

ROS Node Sample Code

publisher.cpp

C: > Users > Josh > Desktop > ODU fall 2022 > publisher.cpp > ...

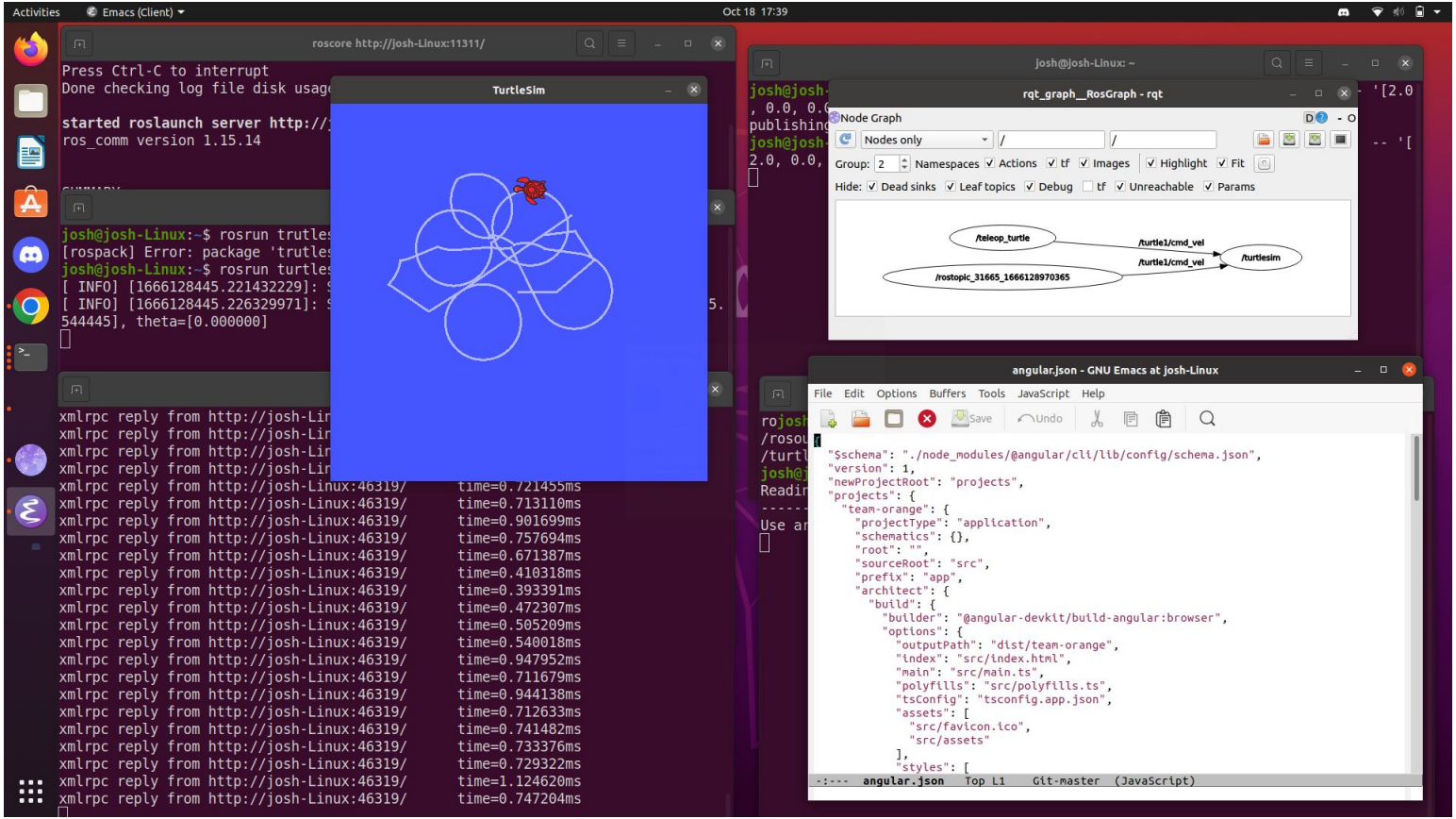
```cpp
#include <sstream>

int main(int argc, char **argv)
{

  ros::init(argc, argv, "talker");
  ros::NodeHandle n;

  ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);

  ros::Rate loop_rate(10);

  int count = 0;

  while (ros::ok())
  {

    std_msgs::String msg;

    std::stringstream ss;

    ss << "hello world " << count;

    msg.data = ss.str();

    ROS_INFO("%s", msg.data.c_str());

    chatter_pub.publish(msg);

    ros::spinOnce();

    loop_rate.sleep();

    ++count;
  }
  return 0;
}

```

subscriber.cpp

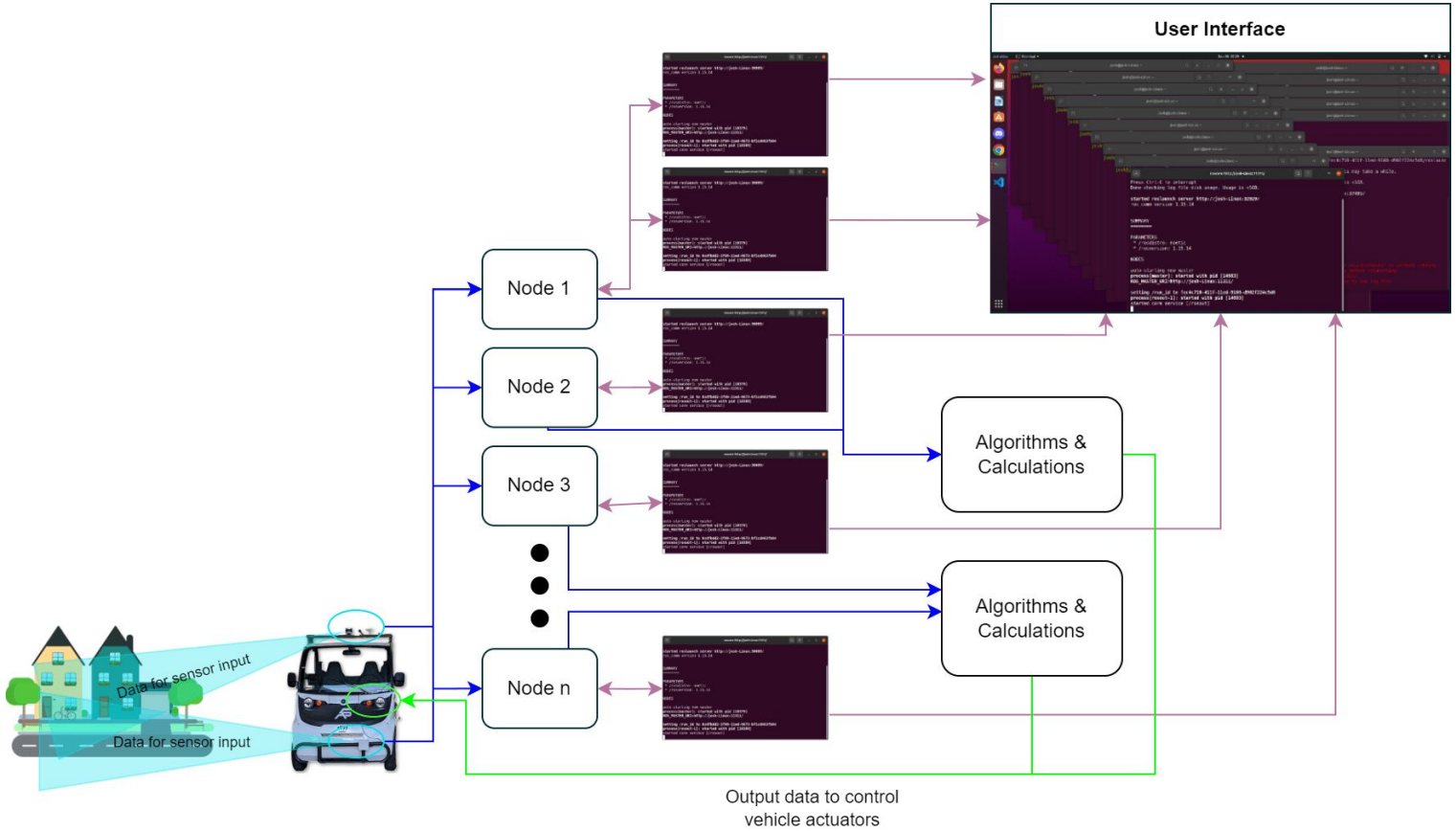
C: > Users > Josh > Desktop > ODU fall 2022 > subscriber.cpp > ...

```cpp

void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
  ROS_INFO("I heard: [%s]", msg->data.c_str());
}

int main(int argc, char **argv)
{
  /**
   * The ros::init() function needs to see argc and argv so that it can perform
   * any ROS arguments and name remapping that were provided at the command line.
   * For programmatic remappings you can use a different version of init() which takes
   * remappings directly, but for most command-line programs, passing argc and argv is
   * the easiest way to do it.  The third argument to init() is the name of the node.
   *
   * You must call one of the versions of ros::init() before using any other
   * part of the ROS system.
   */
  ros::init(argc, argv, "listener");

  ros::NodeHandle n;

  ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);


  ros::spin();


  return 0;
}

```

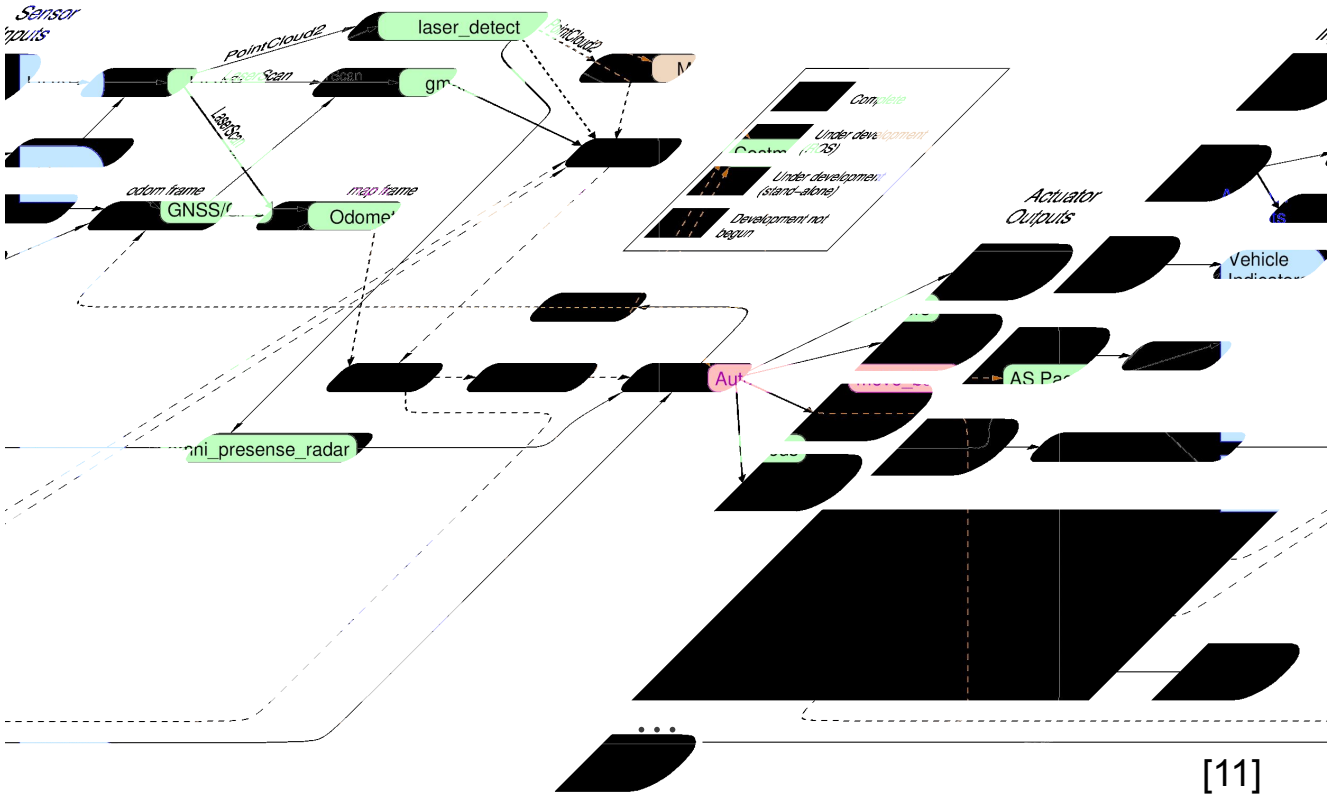
Current Developer Interface



Current Developer Interface Diagram



Architecture for Monarch 1



[11]

Problem Characteristics

ROS workflows leverage several non-attributional windows that elevate the difficulty of debugging, monitoring & understanding

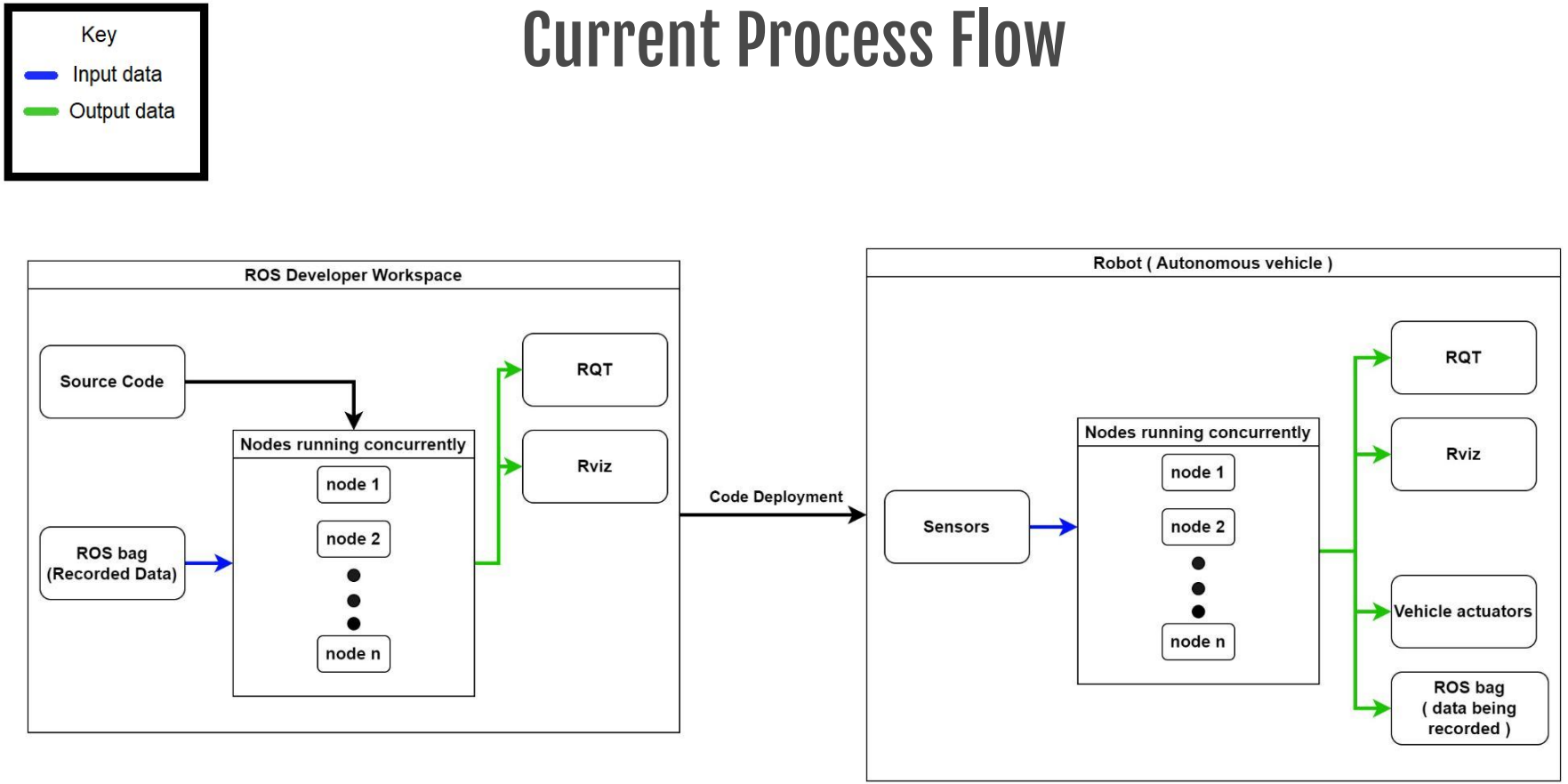
- Each ROS node has at least one terminal window associated with it
- There is no distinction when changes in one node affects a related node
- Simple debugging methodologies become time consuming due to how ROS displays errors

Many parts of the documentation are not up to date and require significant technical knowledge

Environment setup can be difficult in any project, but especially in one that can use multiple versions and distributed dependencies

ROS development contains high barriers of entry for new developers and environments.

Current Process Flow



Our Customers

Dr. Belfore

Hobbyists

Industry

Universities

Solution Characteristics

Simplify UI

Create a GUI that simplifies common tasks and commands and accesses visualization tools

Quick Environment Setup

Create a process to quickly build an environment for any ROS project existing or new

Custom Solutions

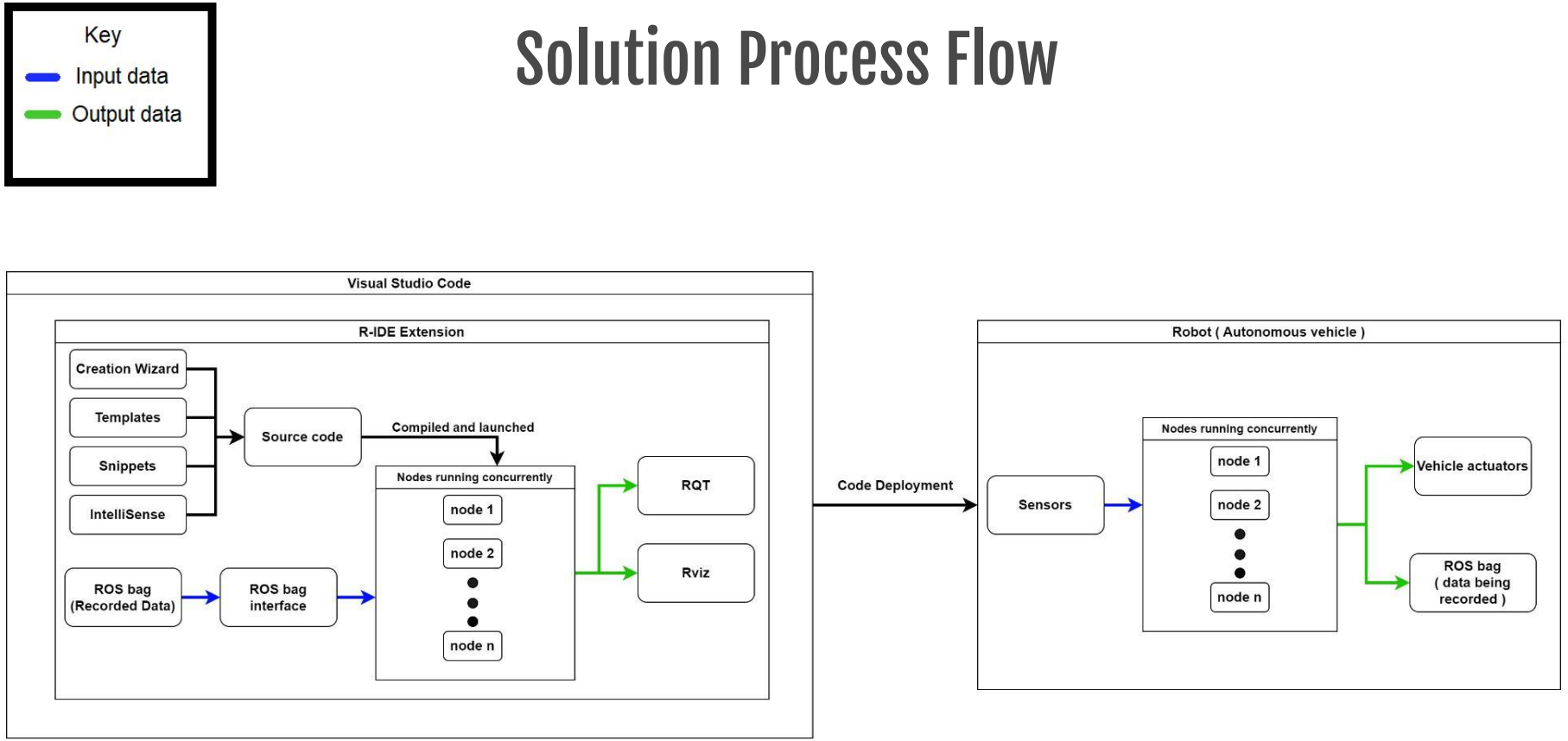
Create custom solutions that can analyze and autofill based on preexisting code

Perspective Shifts

Create a system that allows for perspectives to quickly change and mutate as required

An extension pack to simplify and speed up the development lifecycle and learning process.

Solution Process Flow



Features Table & Constraints

	x = have access	RIDE Developers	RIDE Extension Developers
Wizard	Create node	x	x
	Create msg	x	x
	Create srv	x	x
Auto update	cmake file	x	x
	package.xml	x	x
Snippets	Autocomplete Code Patterns	x	x
ROS bag	Start Rosbag recording	x	x
	Stop Rosbag recording	x	x
	Play back Rosbag	x	x
Visuals	rviz	x	x
	rqt	x	x
	Node Graph	x	x
ROS Topic	ROS topic monitor	x	x
	ROS topic publisher	x	x
Data Management	Usage Analysis		x

Our product will not:

- Completely automate the development of software solutions for robots
- Handle physics simulation from tools like Gazebo

GUI Competition Matrix

	R-IDE	RQT
Dockable Windows	Yes	Yes
Modular Plugins	Yes	Yes
ROS2 Support	Yes	Yes
Debugger Support	Yes	No
RViz Support	Yes	Yes
OS Support	Yes	Yes
ROSTopic support	Yes	Yes
ROS bag support	Yes	Yes
Web page View	Yes	No
Launch nodes from GUI	Yes	From Launch File
Perspective Shifting	Yes	Yes
Most Recent Update (< 6 months)	In Active Development	2019?

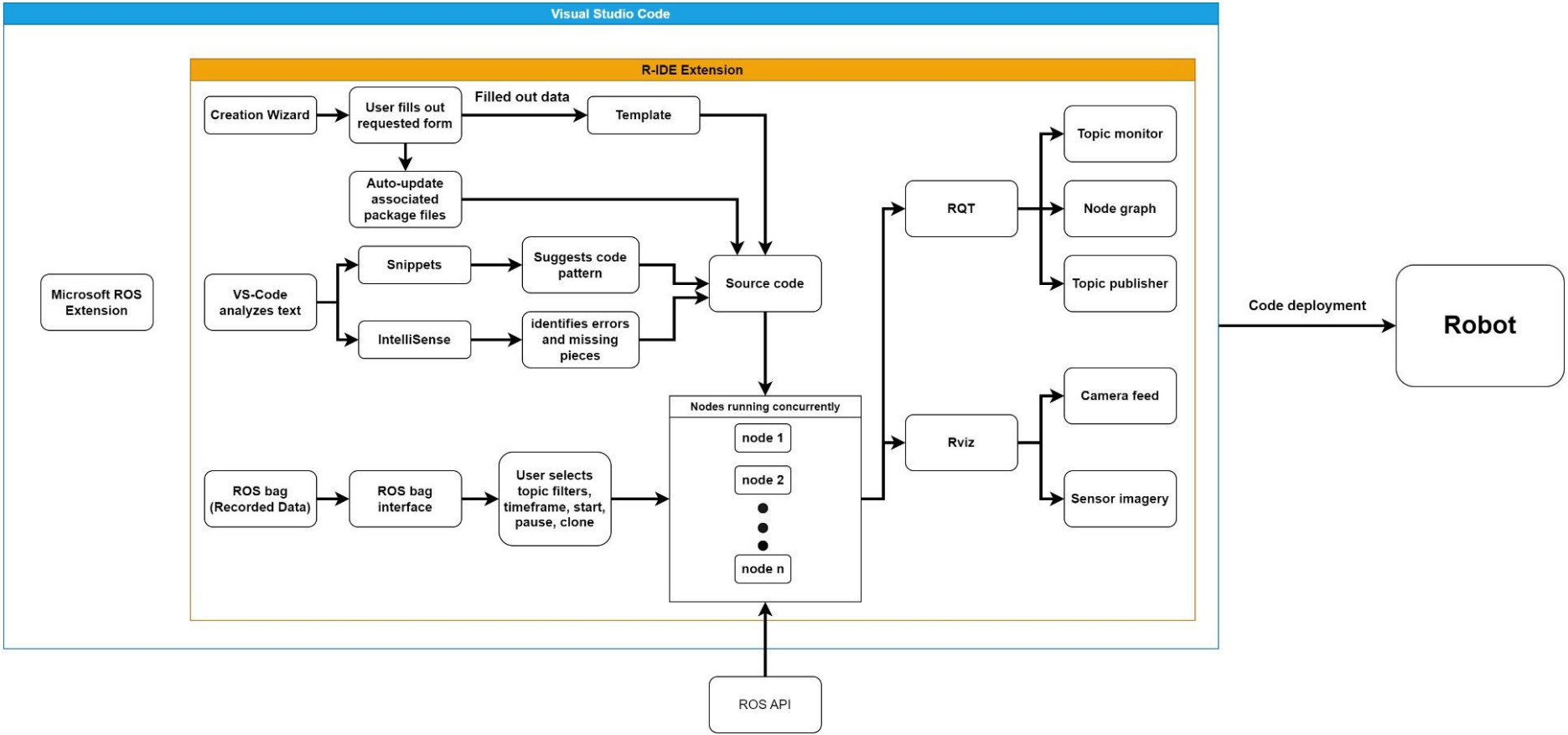
Plugin Competition Matrix

	R-IDE	ROS Hatchery	ROS 0.8.4	helm-ROS	vim-ROS
IDE	VS-Code	JetBrains	VS Code	Emacs	Vim
Automatic ROS enviroment config	Yes	No	Yes	No	No
Start/Stop/Status of ROS Core	Yes	No	Yes	Yes	No
Automatically create catkin_make or catkin build build tasks	Yes	No	Yes	No	No
Automatically create launch.json file	Yes	No	Yes	No	No
Launch Debugging, debug ROS nodes	Yes	No	Yes	Yes	No
Automatically add the ROS C++ include and Python import paths	Yes	Yes	Yes	No, C++ only	No, Python only

Plugin Competition Matrix cont..

	R-IDE	ROS Hatchery	ROS 0.8.4	helm-ROS	vim-ROS
IDE	VS-Code	JetBrains	VS Code	Emacs	Vim
Syntax highlight for .msg and .urdf files	Yes	No	Yes	No	No
ROS master monitoring	Yes	No	Yes	No	No
ROS 2	Yes	No	Yes	No	No
Debugging single ROS node	Yes	No	Yes	No	No
Wizards for common tasks	Yes	No	No	No	No
Templating	Yes	No	No	No	No
Visualization	Yes	No	No	No	No
ROS Bag Recording/Play back	Yes	No	No	No	No
Last update	In Active Development	Oct 18, 2019	Oct 31, 2022	Aug 12,2016	Nov 17, 2022

Major Functional Component Diagram



Extension Software/Hardware Development Resources

Software Requirements

ROS
Rviz
OpenGL

Database

MySQL

Languages

Javascript
Html/CSS
Python
C++
XML

Developer OS

Linux
Windows*
Mac*

IDE

VS-Code

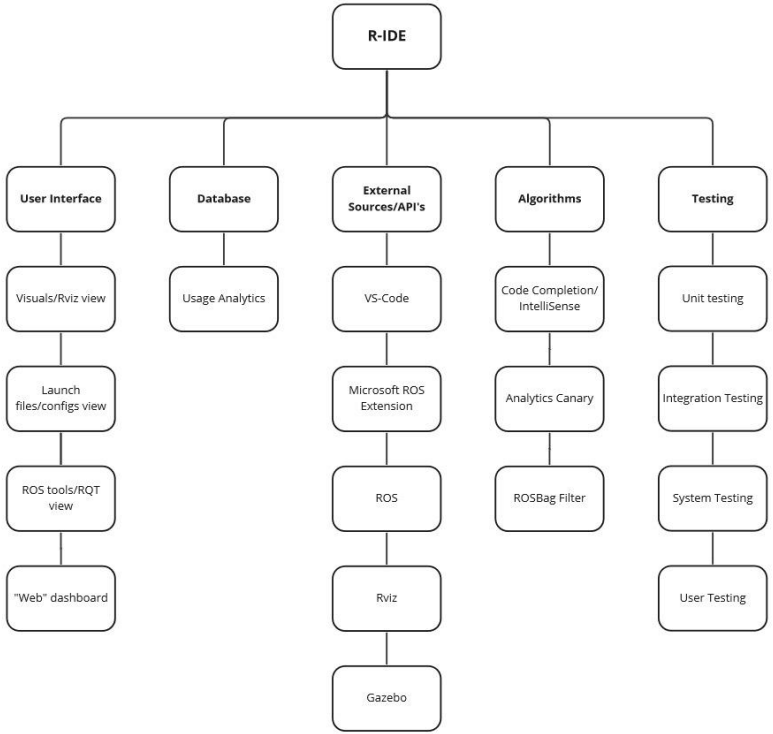
Hardware

Computer that meets inherited hardware requirements for VS code, ROS, and Rviz

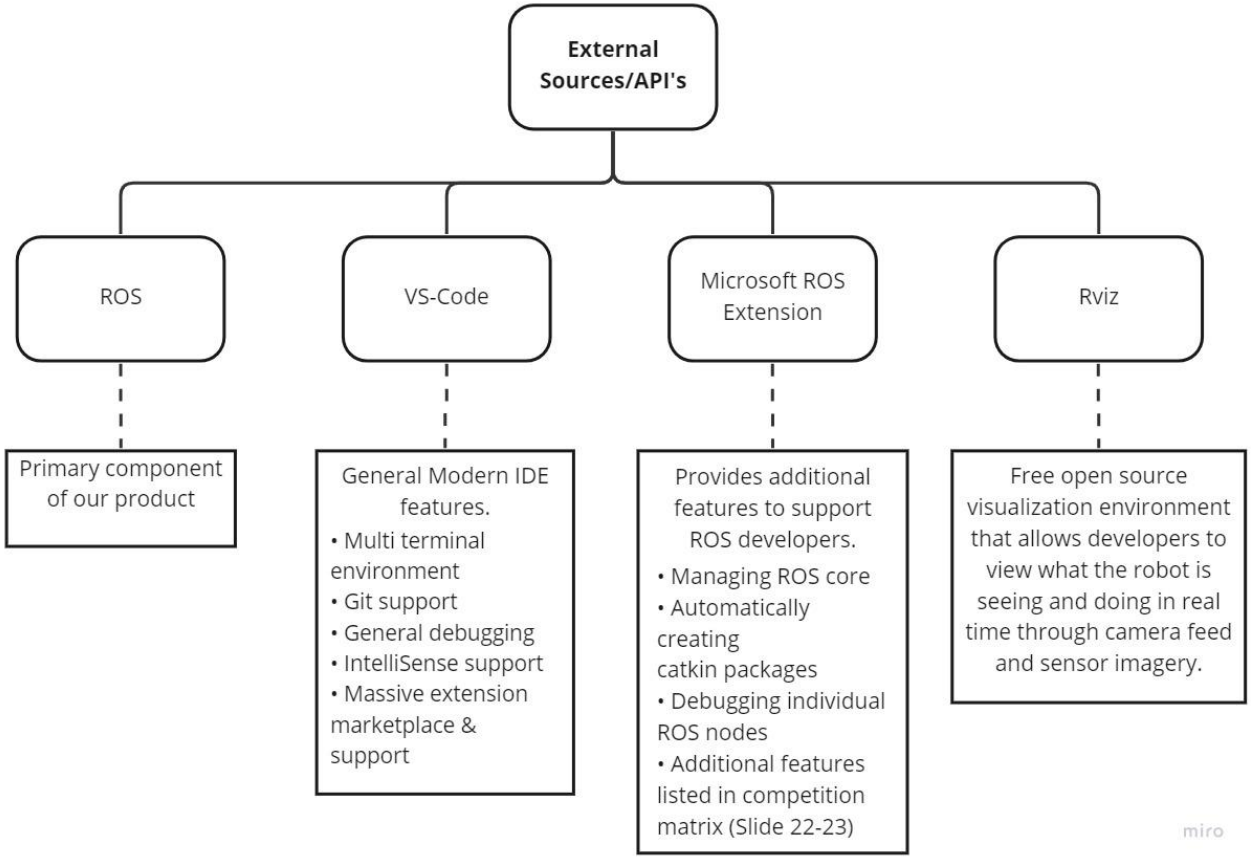
https://code.visualstudio.com/docs/supporting/requirements

*ROS1 requires some kind of linux environment such as WSL or a virtual machine with linux

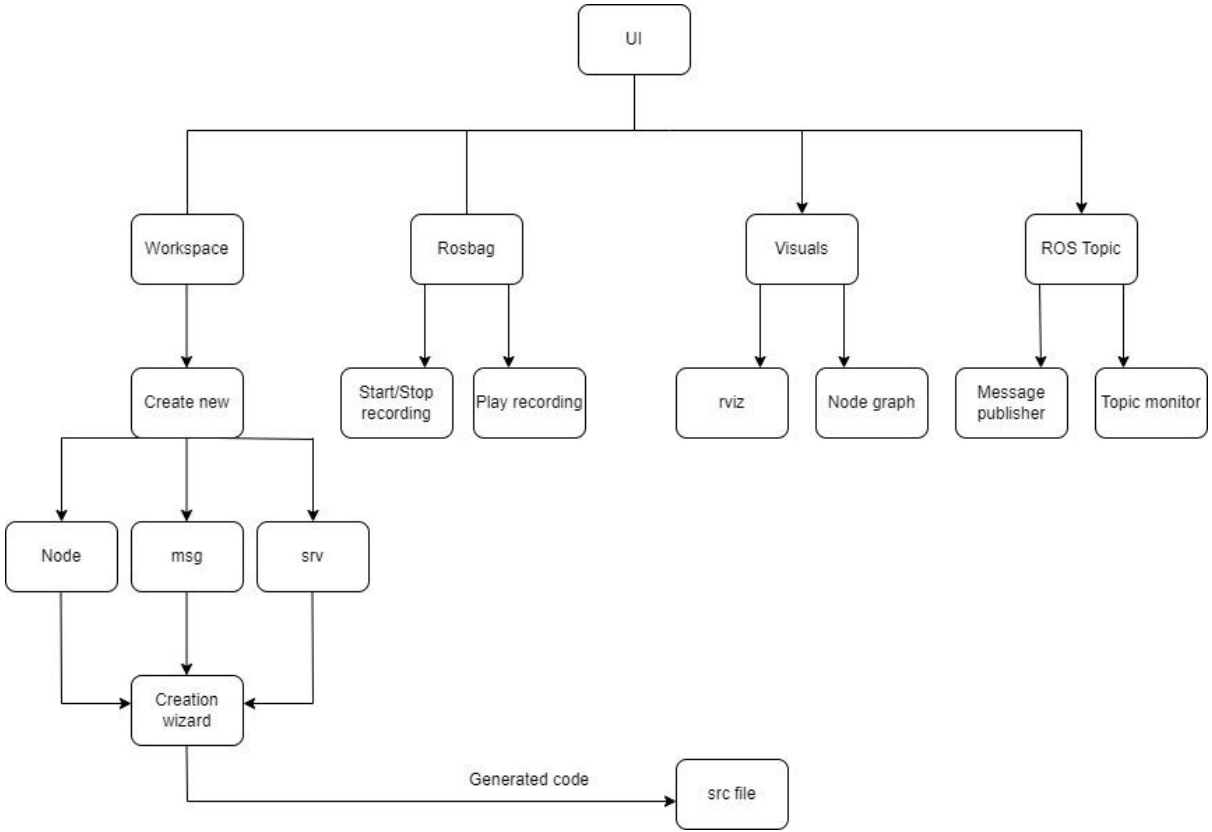
Work Breakdown Structure



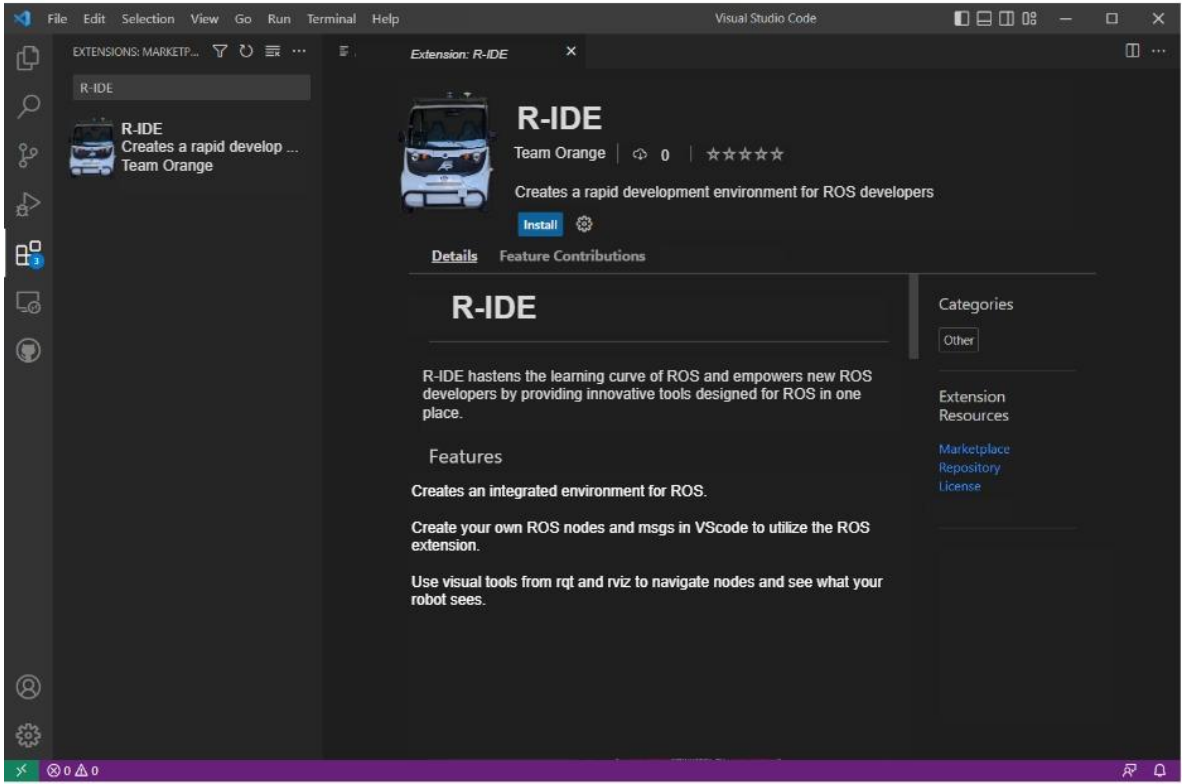
External Sources / API's



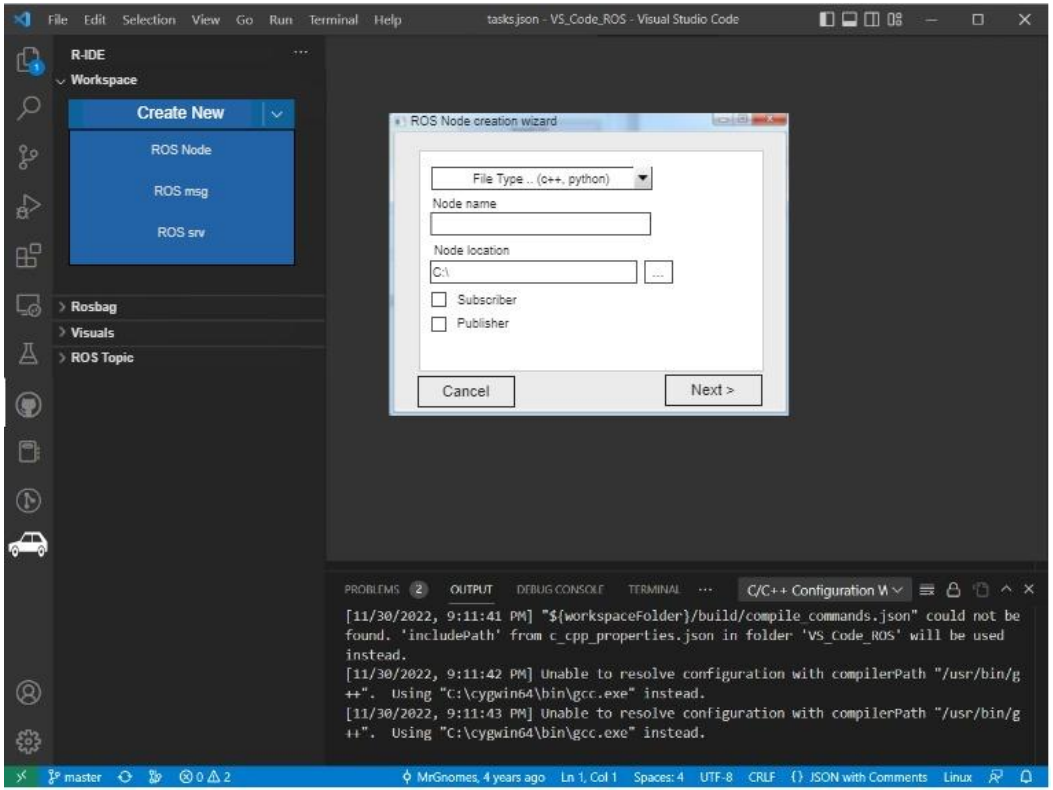
Sitemap


UI
Workspace
Rosbag
Visuals
ROS Topic
Create new
Start/Stop recording
Play recording
rviz
Node graph
Message publisher
Topic monitor
Node
msg
srv
Creation wizard
Generated code
src file

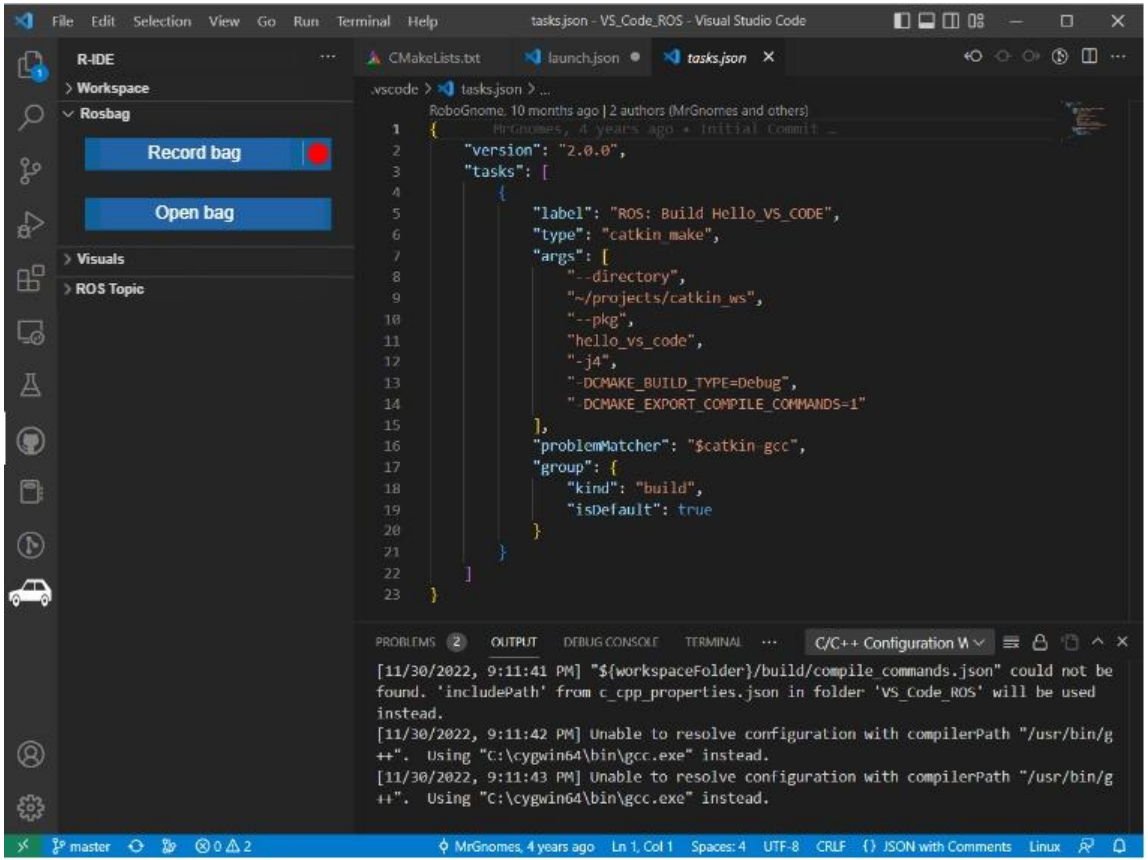
Mockup – Where to find us?



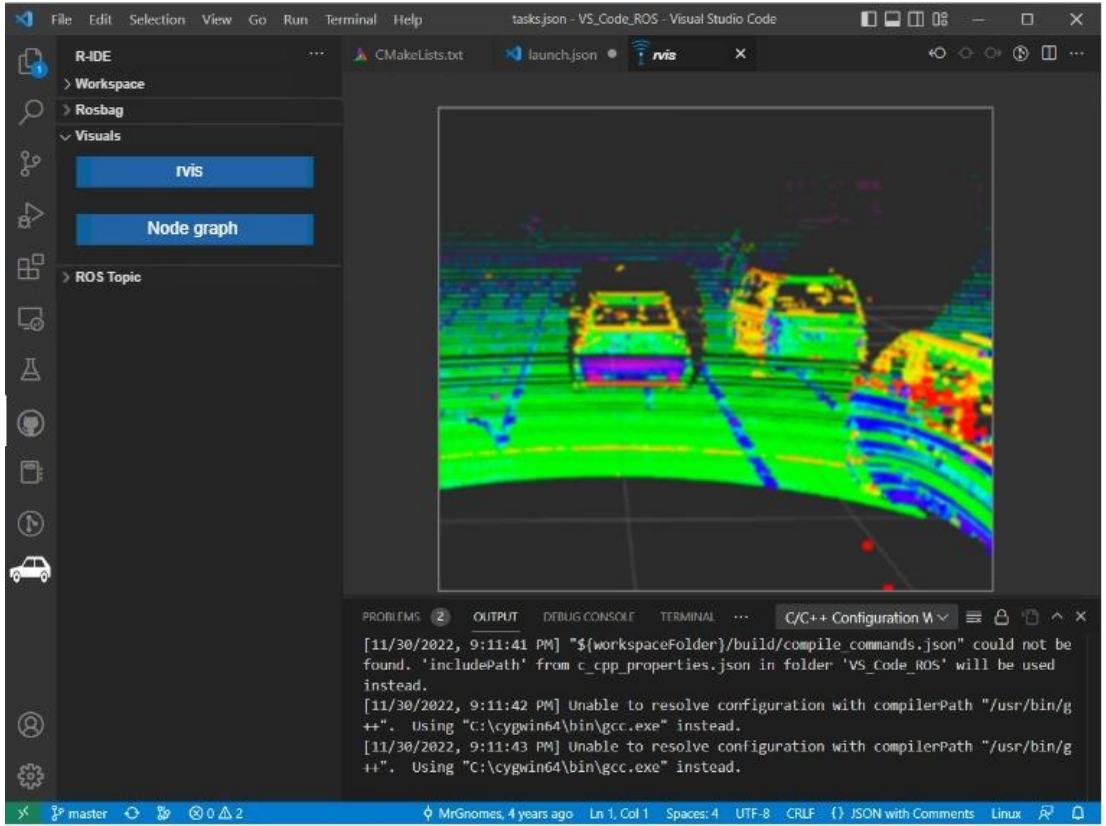
Mockup – Workspace



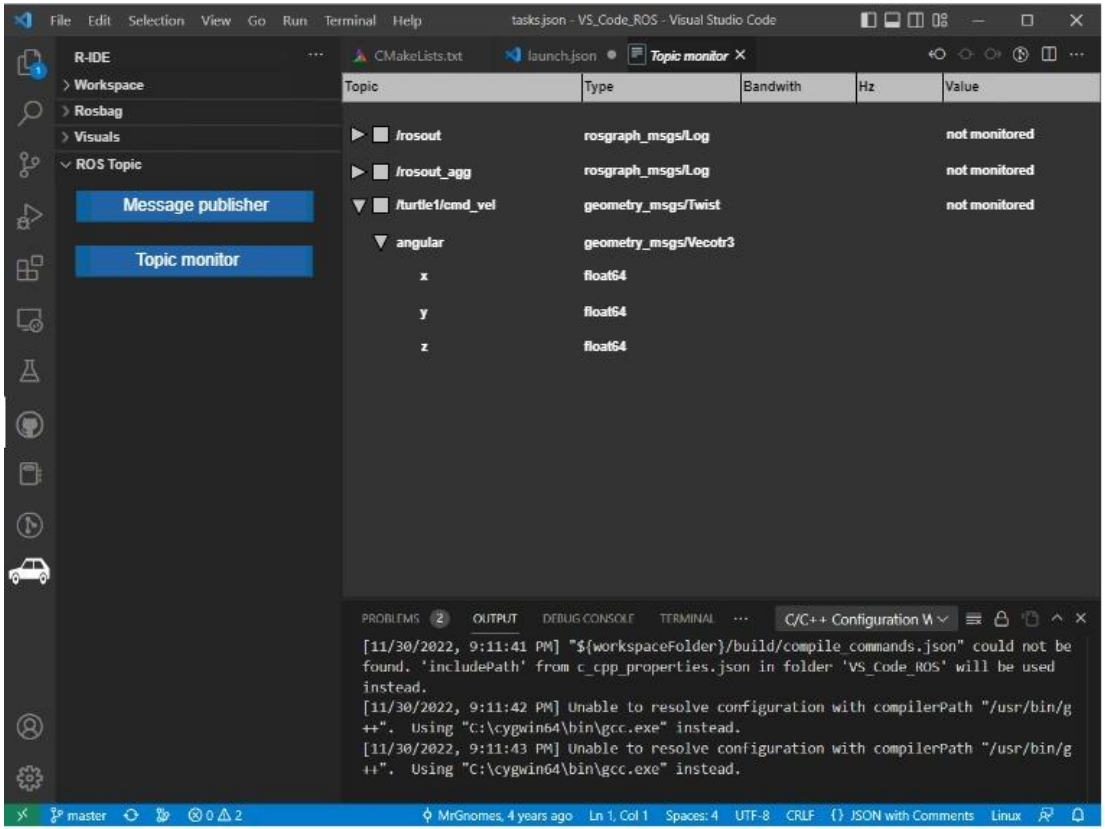
Mockup – Rosbag



Mockup – Visuals



Mockup – ROS Topic



Algorithms

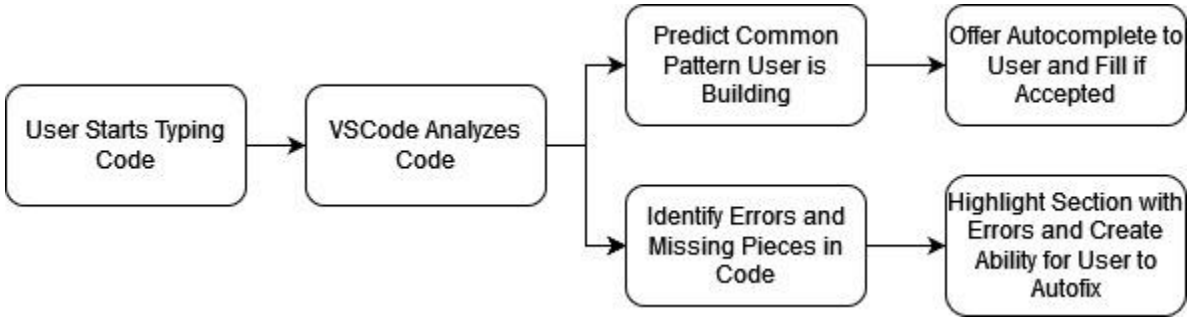
Our algorithms are extremely simple and very generic. They mostly take advantage of tools built into VSCode:

- Utilizing IntelliSense and Snippets to create code completion tools inside the text editor workspace
- Taking advantage of Wizarding and Dialogue tools to gather data from a user and implement inside a template
- Utilizing the Wizarding and caches to mark files affected by mutations and creations made by user

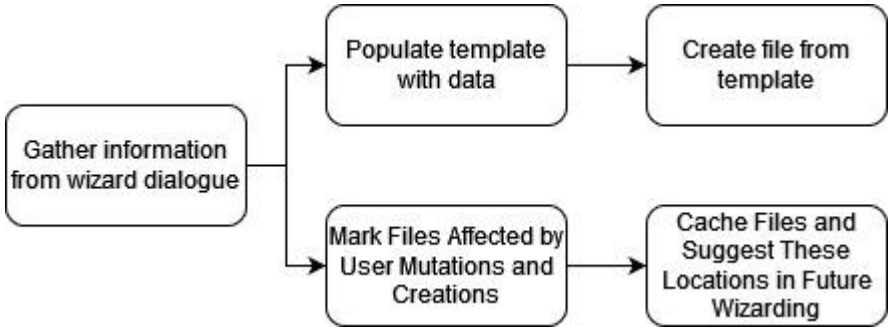
Connecting outside of VSCode

- Gathering and analyzing data from our users and alerting developers when needed
- Filtering ROS topics out of ROS bags

Intellisense And Snippets

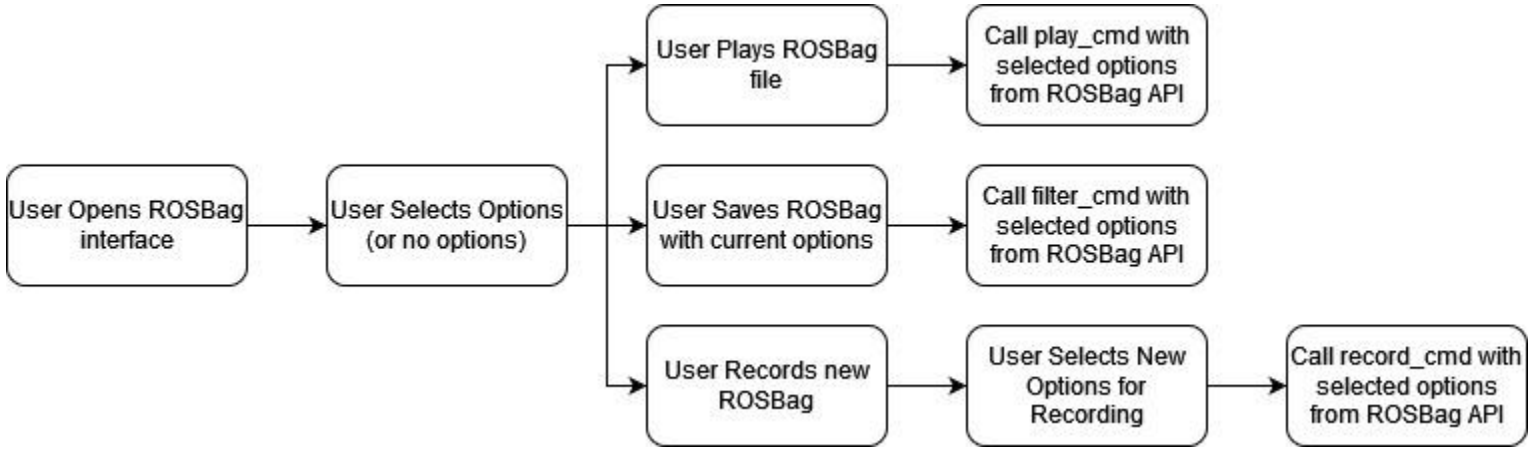


Wizarding


Gather information from wizard dialogue
Populate template with data
Create file from template
Mark Files Affected by User Mutations and Creations
Cache Files and Suggest These Locations in Future Wizarding

Analytics Canary

ROSBag

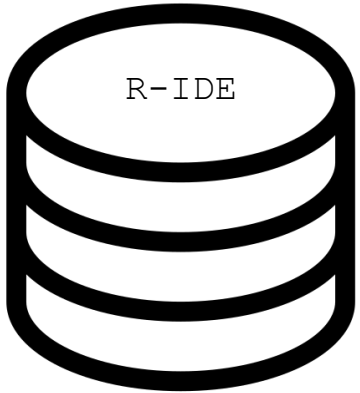

User Opens ROSBag interface
User Selects Options (or no options)
User Plays ROSBag file
Call play_cmd with selected options from ROSBag API
User Saves ROSBag with current options
Call filter_cmd with selected options from ROSBag API
User Records new ROSBag
User Selects New Options for Recording
Call record_cmd with selected options from ROSBag API

Data Management

Data Input

Data is external from

- ❏ Sensors
- ❏ Actuators
- ❏ Onboard DB



Data Store

ROS bag in local repository

MySQL server

- ❏ Log usage and errors

Data Collect

Usage Analytics

Based on event

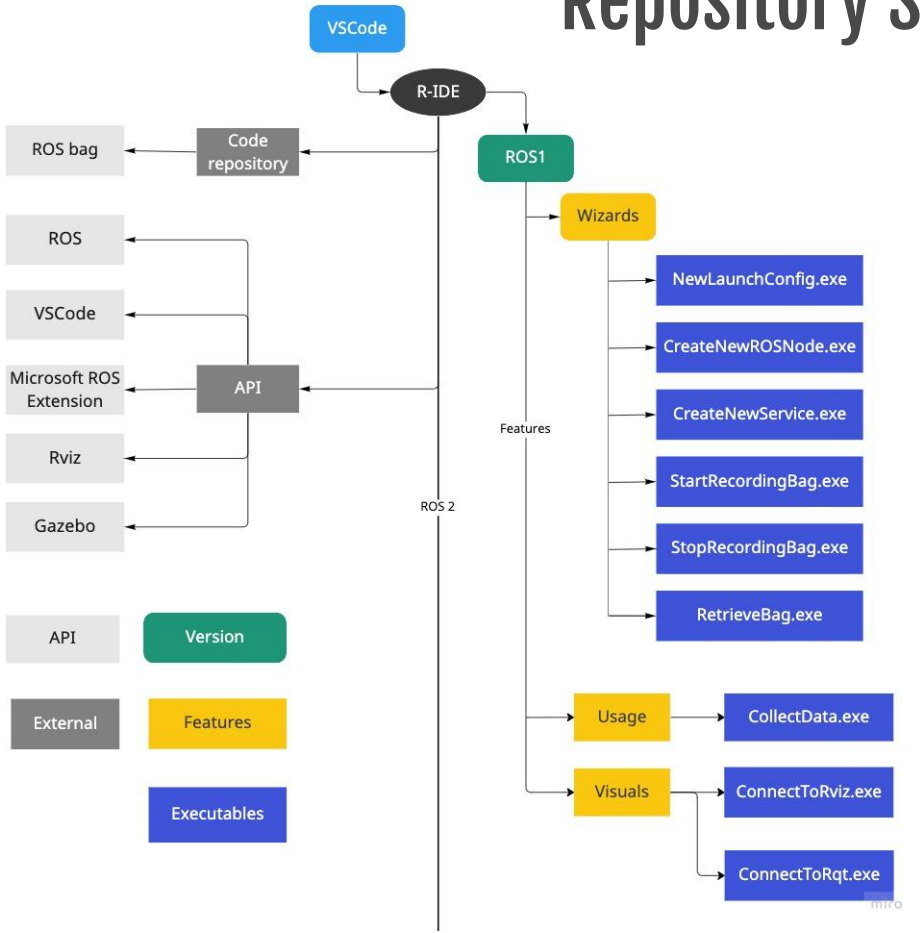
Avoid sensitive information

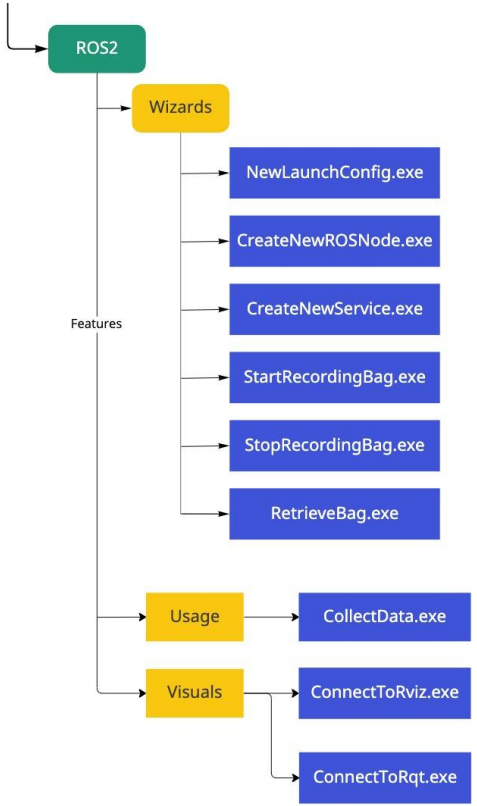
Data Security

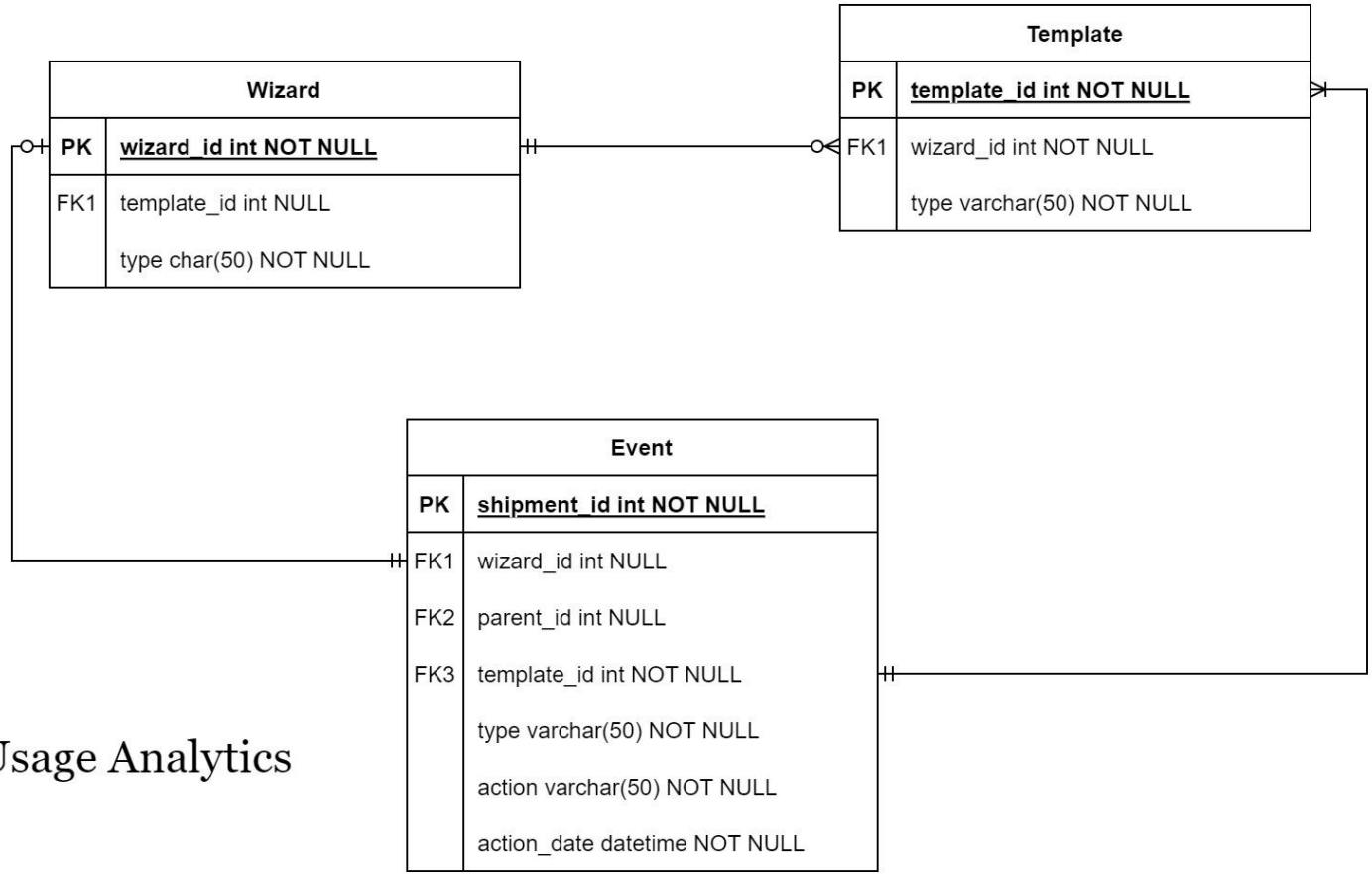
Uploading usage data into a database

Using encryption to protect data

Repository Structure

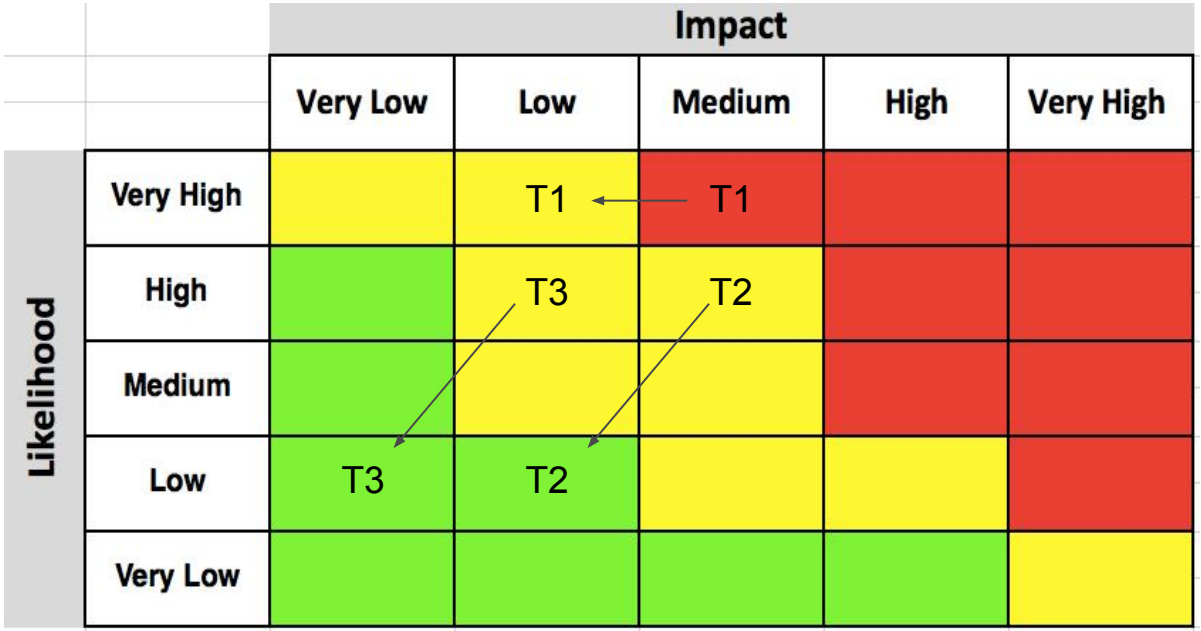






Usage Analytics

Technical Risk Matrix



T1: VSCode API dependencies

- **Risk**: New updates may cause dependency issues
- **Mitigation**: Use vscode-test npm package

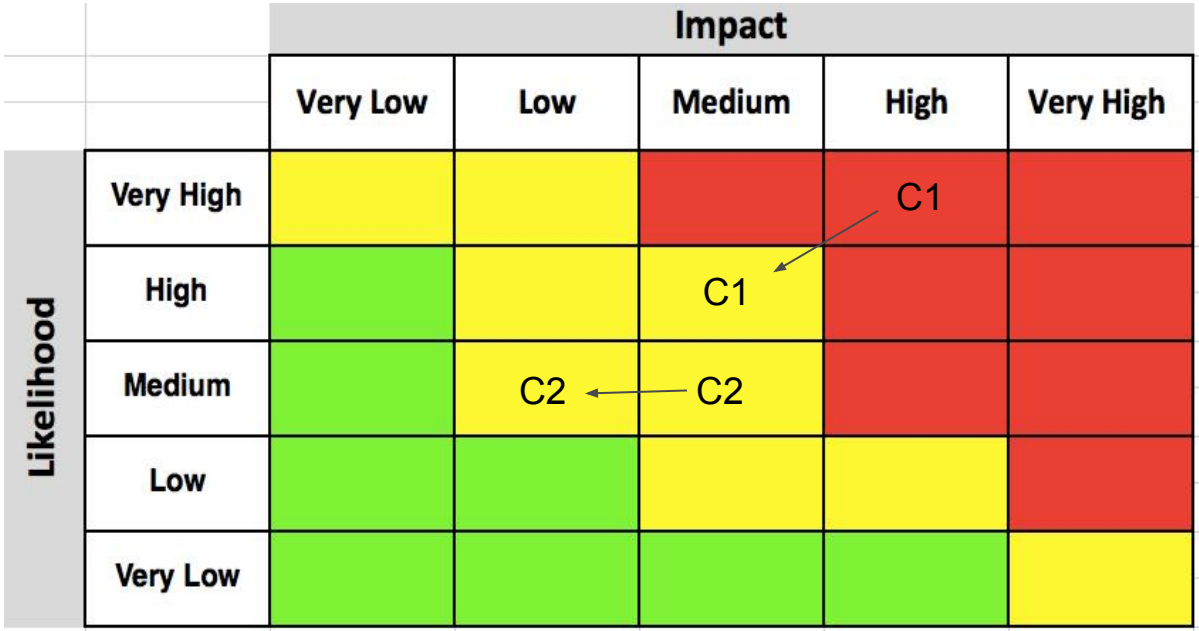
T2: Dependency issues between wizards

- **Risk**: New ROS versions make breaking changes
- **Mitigation**: Pinned version of ROS

T3: ROS1 → ROS2

- **Risk**: Communication between ROS1 and ROS2
- **Mitigation**: Using ros1_bridge, which allows bidirectional communication

Customer Risk Matrix



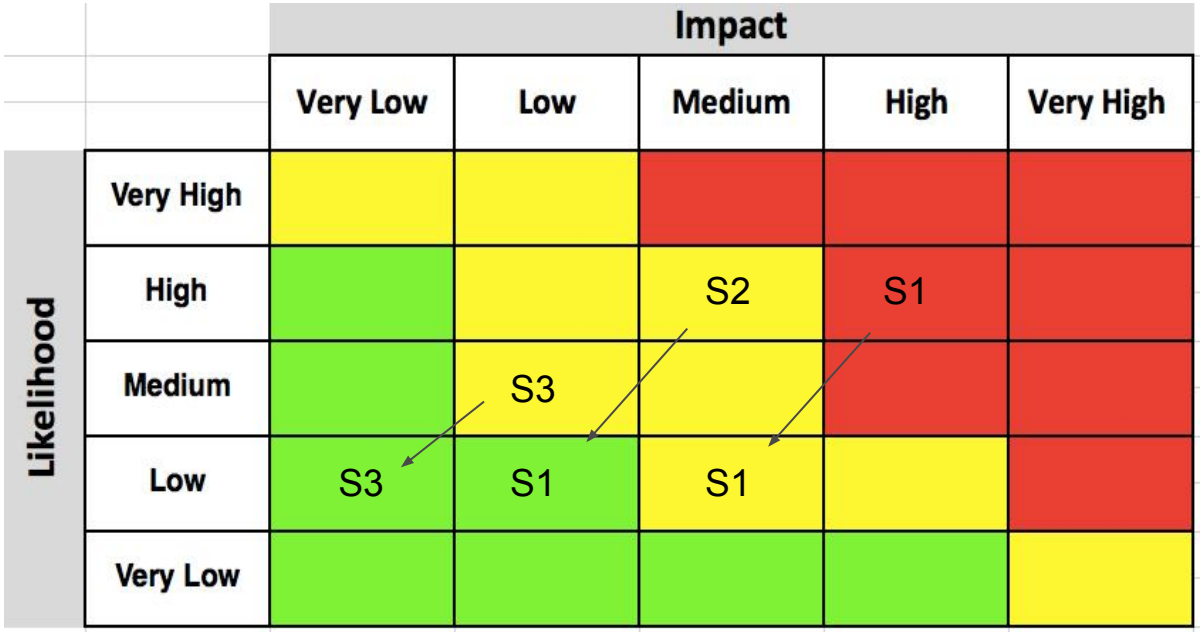
C1: Knowledge gap

- **Risk**: Tutorials are too complex or too easy for users
- **Mitigation**: Design in a modular manner

C2: Lack of Resources

- **Risk**: May not be able to cover all topics
- **Mitigation**: Documentation and well-written user requirements

Security Risk Matrix



S1: Unauthorized data access

- **Risk**: Gain sensitive data about user from ROS nodes
- **Mitigation**: Avoid collecting sensitive data and secure communication channel

S2: Uploading data into the database

- **Risk**: Unauthorized access to data while inserting
- **Mitigation**: Encryption while transmitting data

S3: MySQL Server

- **Risk**: Unauthorized access to data while inserting
- **Mitigation**: User authentication and encryption

Sprint Plan

Sprint 0: Setting up development environment
Creating an extension pack with Microsoft ROS and features from rqt included

Sprint 1: Frontend and Backend
Backend - database with mock data
Frontend - framework that can communicate with backend

Sprint 2: All views built and all algorithms finished
Building wizards and templates

Sprint 3: Analytics - show we solved a problem
The implementation of all the plugins that we create into the extension pack

Sprint 4: Make pretty- fine tune
This includes all of the bug fixes that might occur from adding all of the code together

Conclusion

- ROS developers need an accessible way to build ROS projects
- RIDE will simplify and speed up development with code automation tools
- RIDE will allow developers to visualize and manipulate data from ROS nodes
- Our system will allow us to monitor our product and make changes quickly

User Stories

User Type	Need/Want	Story	Story Tags
Developer	Need	support various file extensions like urdf, srv, msg.	
Developer	Need	A wizard to create a new ros node	Node Creation
Developer	need	a wizard to create ROS messages	
Developer	Need	a wizard to debug ROS nodes	
Developer	Need	a wizard to create a service	
Developer	need	a wizard to replay a ROS bag recording	
Developer	Need	a wizard to debug individual nodes as they are executed	
Developer	Need	Update cmake/package files on creation of new node	Node Creation
Developer	Need	ability to download the plugin from the VS code extension marketplace	
Developer	Want	ability to run simulation tests with gazebo	
Developer	Want	Integrate and use ROS/rqt plugins	ROS Plugins
Developer	Want	create visuals for Rviz and rqt	RViz
Developer	Want	Easily start a ROS bag recording	ROS bag
Developer	Want	Strip transforms and create a subset bag	ROS bag
Developer	Want	Easily Play a ROS bag recording	ROS bag
Developer	Want	Observe and manipulate ROS topics	ROS Topics
Developer	Want	a wizard to create launch configs and files	Launch
Developer	Want	a wizard to create buttons associated with launch files I created	Launch
Developer	Want	oversee error log	Error Log
Developer	Want	Securely subscribe to ROS topics on a different device	ROS Topics
Developer	Want	Utilize ROS 1 or ROS 2 seamlessly	Backwards Integratior
Developer	Want	Remotely connect to ROS applications	Remote Work
Developer	Want	Identify top-level/initial errors in runtime	Error Detection
Developer	Want	double click node on graph and have source for node appear	
RIDE Developers	Want	Telemetry/ Analytics support for RIDE Developers	
Spectator	Want	View a GUI dashboard of a running application over a network connection	Dashboard
Test Writer	Want	Write effective tests quickly	

References

Citation

1. “Robot Operating System.” ROS, www.ros.org/.
2. “Global Robot Operating System Market Research Report 2022(Status and Outlook).” Market Reports, www.marketreportsworld.com/global-robot-operating-system-market-21185690.
3. “Robot Operating System Market.” Future Market Insights, www.futuremarketinsights.com/reports/robot-operating-system-market.
4. “Lunar Rover.” Open Robotics, www.openrobotics.org/customer-stories/lunar-rover.
5. Wessling, Brianna. “Open Robotics Developing Space ROS with Blue Origin, NASA.” The Robot Report, 11 Feb. 2022, www.therobotreport.com/open-robotics-developing-space-ros/.
6. Ackerman, Evan. “Microsoft Announces Experimental Release of ROS for Windows 10.” IEEE Spectrum, IEEE Spectrum, 24 June 2021, spectrum.ieee.org/microsoft-announces-experimental-release-of-ros-for-windows-10.
7. Tomoya Fujita. “aibo with ROS”, https://roscon.ros.org/2018/presentations/ROSCon2018_Aibo.pdf
8. Brian Gerkey on September 1, 2014 8:07 AM. “ROS Running on ISS.” Ros.org, www.ros.org/news/2014/09/ros-running-on-iss.html.
9. M. Cardona, A. Palma and J. Manzanares, "COVID-19 Pandemic Impact on Mobile Robotics Market," 2020 IEEE ANDESCON, 2020, pp. 1-4, doi: 10.1109/ANDESCON50619.2020.9272052. https://ieeexplore.ieee.org/abstract/document/9272052
10. International Federation of Robotics. Annual installations of industrial robots 2019-2020 and 2021*-2024* [Graph]. Retrieved from https://ifr.org/downloads/press2018/2021_10_28_WR_PK_Presentation_long_version.pdf
11. “Wiki.” Ros.org, wiki.ros.org/Security.
12. Dieber, Bernhard, et al. "Security for the robot operating system." *Robotics and Autonomous Systems* 98 (2017): 192-203.

References Cont.

Competition Matrix References

1. CLion IDE: https://www.jetbrains.com/clion/
2. CLion Hatchery: https://plugins.jetbrains.com/plugin/10290-hatchery
3. CLion ROS Support: https://plugins.jetbrains.com/plugin/11235-ros-support
4. VS Code: https://code.visualstudio.com/
5. QT Creator with ROS: https://github.com/ros-industrial/ros_qtc_plugin/blob/devel/gh_pages/index.rst
6. Emacs with Rosemacs: http://wiki.ros.org/rosemacs
7. VIM ROS: https://github.com/taketwo/vim-ros
8. TMUX: https://github.com/tmux/tmux
9. GNU Screen: https://www.gnu.org/software/screen/
10. Terminator: https://github.com/gnome-terminator/terminator
11. "Software Design Report for the ODU Monarch I", Version 1.3, Jul 8, 2022, page 3. Author Lee Belfore"

Glossary

https://www.cs.odu.edu/~410orang/#/glossary

ROS - The Robot Operating System (ROS) is an open source set of software libraries and tools that help developers build robot applications.

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.

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

ROS Bag - A bag is a file format in ROS for storing ROS message data. Bags are the primary mechanism in ROS for data logging, which means that they have a variety of offline uses.

ROS Topic - Topics are named buses over which nodes exchange messages. Topics have anonymous publish/subscribe semantics, which decouples the production of information from its consumption.

Glossary cont..

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

Gazebo - Gazebo is a real-world physics simulator that creates a world and simulates the robot

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

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.

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

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

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