ROSfunc-EKLAVYA 2023

November, 2023

~Richa Sawant

Mentors : *Viraj Shah* and *Aryan Karawale*

From *Society of Automation and Research* , VJTI , Mumbai

OVERVIEW

ROSfunc showcases how a PubSub system can be used to facilitate communication between various components within a larger system. The project provides an *Elixir*-based implementation of a PubSub model, allowing different modules to communicate efficiently.

Why Functional programming ?

We chose functional programming to write the Publisher-Subscriber (PubSub) library because functional programming encourages a clean and predictable approach to handling data and state, which is crucial in a distributed system like our PubSub model.

Here are some reasons why we opted for functional programming:

1. Immutability: Functional programming promotes immutability, meaning data doesn't change once it's set. This helps prevent unexpected side effects and makes the system more reliable.

2. Concurrent and Parallel Execution: Functional languages like Elixir excel in handling concurrent and parallel tasks, a key

requirement for efficient communication in a distributed system.

3. State Management: Functional programming simplifies state management. In a PubSub system, it's vital to keep track of subscribers and messages, and functional programming provides a structured way to handle this.

4. Clarity and Maintainability: Functional code tends to be more readable and maintainable. This is essential when building a complex communication system like PubSub, as it's easier to understand and modify.

In summary, we used functional programming for our PubSub model because it offers a robust and organized way to manage data, state, and communication in a distributed environment, making our system more reliable and maintainable.

PubSub Library

The heart of the project is the PubSub library, which enables modules to publish messages to specific topics and subscribe to those topics to receive messages. The PubSub system is built using the **GenServer** behavior, providing a robust, concurrent, and fault-tolerant communication mechanism.

Key features of the PubSub system:

- **Publishing Messages:** Modules can publish messages to specific topics.
- **Subscribing to Topics:** Modules can subscribe to topics and provide callback functions to handle incoming messages.
- **Concurrent Handling:** Messages are handled concurrently, allowing for efficient communication between modules.

Implementation description

Several modules are implemented to demonstrate the PubSub model:

- **Demo.Main:** Orchestrates the initialization of various components.
- **Demo.Mcu:** Represents the microcontroller component that subscribes to LSA and MPU topics, processes data, and publishes output.
- **Demo.Motors:** Represents the motors component that subscribes to the output topic from the MCU.
- **Demo.Lsa:** Simulates the LSA (Laser Sensor Array) component by publishing readings to the LSA topic.
- **Demo.Mpu:** Simulates the MPU (Motion Processing Unit) component by publishing readings to the MPU topic.

- **Demo.Pubsub:** The core PubSub system responsible for managing topics and message distribution.

Conclusion

ROSfunc successfully introduces functional programming concepts while providing a practical implementation of a PubSub system in Elixir. The project demonstrates the power of functional programming in creating a modular and robust system for communication between components. The Wall-E use case illustrates how such a system can be used in real-world scenarios to manage complex interactions.