YASMIN

Yet Another State MachINe for ROS 2

Miguel Ángel González Santamarta

January 2026

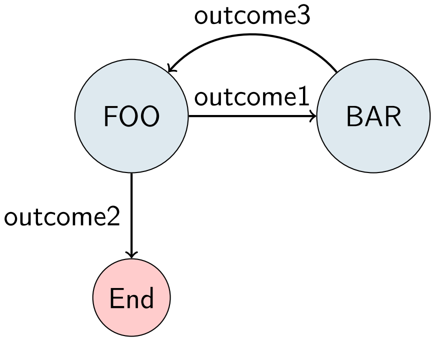
Outline

What is YASMIN?

YASMIN = **Y**et **A**nother **S**tate **M**ach**IN**e

A project focused on implementing **robot behaviors** using **Finite State Machines (FSM)**.

- Available for **ROS 2**
- Supports **Python** and **C++**
- Open source (GPL-3.0)
- Active development



Why YASMIN?

Motivation

Robot behavior programming requires structured approaches for:

- Managing complex state transitions
- Simple implementation of behaviors
- Integrating with ROS 2 ecosystem
- Rapid prototyping and iteration

Benefits

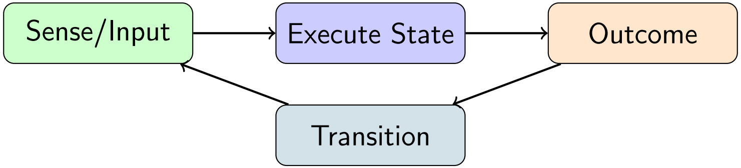
- ✓ Easy to use API for both Python and C++
- ✓ Native ROS 2 integration
- ✓ Built-in visualization tools
- ✓ Support for hierarchical state machines

Behavior Generation with State Machines

Reactive State-Based Control

YASMIN implements a **behaviors generation** using finite state machines:

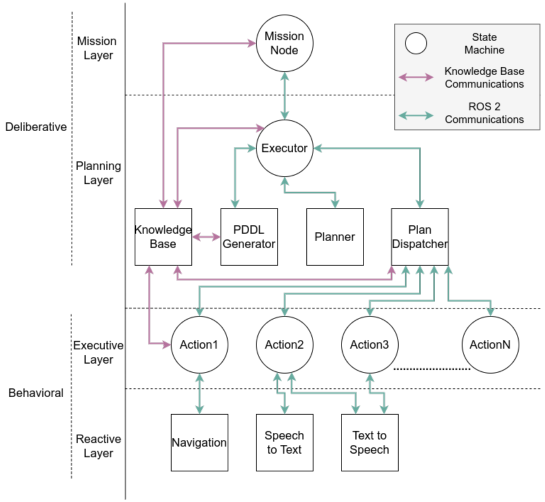
- **Deterministic**: Predictable state transitions
- **Execution**: State processes inputs and performs actions
- **Transitions**: Outcomes drive automatic state transitions
- **Modular**: Composable behaviors through state encapsulation
- **HFSM**: Hierarchical composition enables complex behavior generation



Behavior Generation with State Machines

- State machines can be integrated into more complex systems
- Can be used to control parts of cognitive architectures[a]
- Implements short-term behavior (actions of robots)

[a] https://github.com/MERLIN2-ARCH/merlin2



Key Features Overview

Multi-Language Support

- C++ and Python implementation (Pybindings)
- Consistent API across languages

ROS 2 Integration

- Action and Service Clients
- Topic Publishers/Subscribers
- Parameter Handling

Visual Editor

- Drag-and-drop interface
- No coding required
- XML export capability

Blackboard System

- Data sharing between states
- Key-value storage
- Remapping support

Visualization

- Real-time web viewer
- State machine monitoring
- Execution tracking

ROS 2 Distro Support

ROS 2 Distro	Build Status	Docker Image
Foxy	✓	[Docker]
Galactic	✓	[Docker]
Humble	✓	[Docker]
Iron	✓	[Docker]
Jazzy	✓	[Docker]
Kilted	✓	[Docker]
Rolling	✓	[Docker]

Installation

```
sudo apt install ros-$ROS_DISTRO-yasmin ros-$ROS_DISTRO-yasmin-*
```

Core Concepts

State

Basic execution unit with:

- Outcomes
- Execute method
- Cancel handling

StateMachine

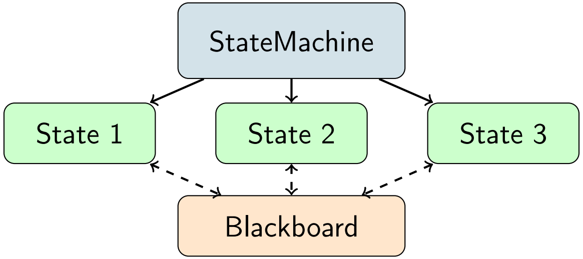
Container for states:

- Add states
- Define transitions
- Nested FSMs

Blackboard

Shared data store:

- Key-value pairs
- State communication
- Data remapping



YASMIN Viewer

Web-based Visualization Tool

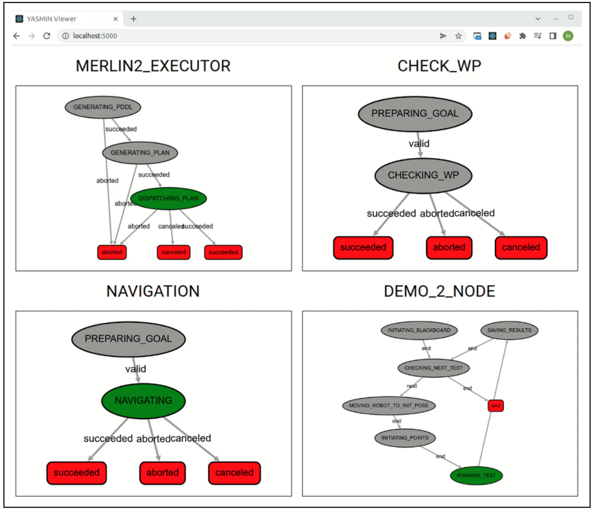
Features:

- Real-time state monitoring
- Visual representation of FSM
- Current state highlighting
- Transition visualization
- Multiple FSM support

Usage

```
ros2 run yasmin_viewer
yasmin_viewer_node
```

Open: `http://localhost:5000`



YASMIN Viewer Interface

YASMIN Editor

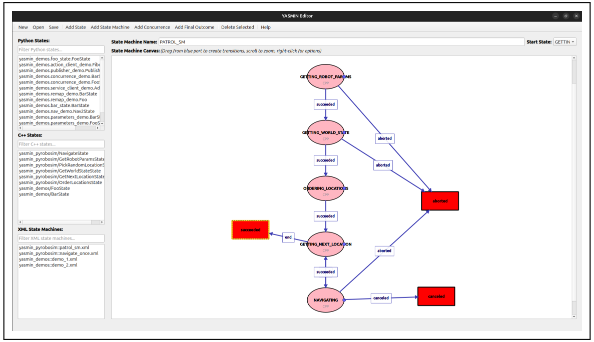
Visual State Machine Editor

Features:

- Drag-and-drop interface
- Visual state machine design
- State library browser
- Connection editor
- Property inspector
- Export to XML format

Usage

```
ros2 run yasmin_editor
yasmin_editor
```



Python States:

Filter Python states...

yasmin_demos.foo_state.FooState
yasmin_demos.action_client_demo.Fibo
yasmin_demos.publisher_demo.Publish
yasmin_demos.concurrence_demo.BarS
yasmin_demos.concurrence_demo.FooS
yasmin_demos.service_client_demo.Ad
yasmin_demos.remap_demo.BarState
yasmin_demos.remap_demo.Foo
yasmin_demos.bar_state.BarState
yasmin_demos.nav_demo.Nav2State
yasmin_demos.parameters_demo.BarSt
yasmin_demos.parameters_demo.FooS

C++ States:

Filter C++ states...

yasmin_pyrobosim/NavigateState
yasmin_pyrobosim/GetRobotParamsState
yasmin_pyrobosim/PickRandomLocationS
yasmin_pyrobosim/GetWorldStateState
yasmin_pyrobosim/GetNextLocationState
yasmin_pyrobosim/OrderLocationsState
yasmin_demos/FooState
yasmin_demos/BarState

XML State Machines:

Filter XML state machines...

yasmin_pyrobosim::patrol_sm.xml
yasmin_pyrobosim::navigate_once.xml
yasmin_demos::demo_1.xml
yasmin_demos::demo_2.xml

State Machine Name: PATROL_SM **Start State:** GETTIN

State Machine Canvas: *(Drag from blue port to create transitions, scroll to zoom, right-click for options)*

YASMIN Editor Interface

Demos

YASMIN Demos using Pyrobosim

Demos:

1. `navigate_random` – Random navigation
2. `patrol` – Area patrol mission
3. `clean_waste` – Full waste collection

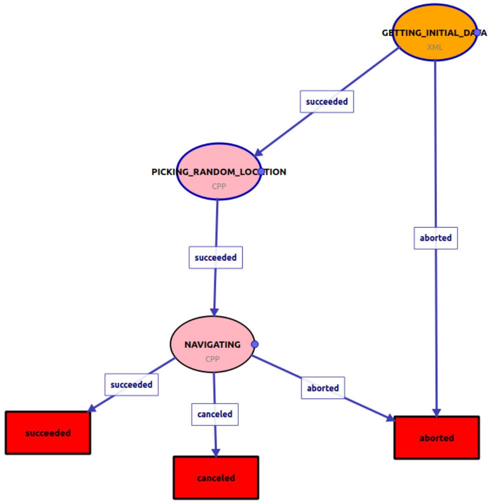
Key Features

- Hierarchical state machines
- Nested XML FSM includes
- C++ state plugins via pluginlib
- Blackboard data remapping
- Navigation action integration

Repository

`https://github.com/mgonzs13/yasmin_pyrobosim`

Demo: Navigate Random

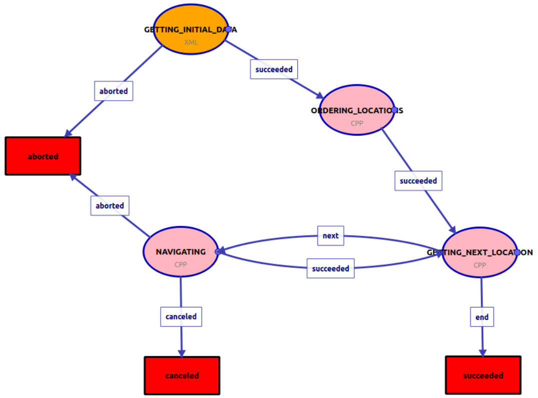


Purpose: Navigate to a random location

Flow:

1. Initialize robot/world data
2. Pick a random location
3. Navigate to location

Demo: Patrol

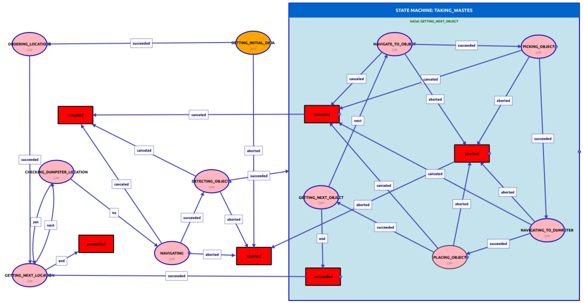


Purpose: Sequential area patrol

Flow:

1. Get initial data (sub-FSM)
2. Order locations optimally
3. Loop: Get next → Navigate

Demo: Clean Waste (Hierarchical FSM)



Purpose: Complete waste collection

Hierarchical Structure:

- Main FSM: Navigation logic
- Sub-FSM: `GETTING_INITIAL_DATA`
- Sub-FSM: `TAKING_WASTES`

TAKING_WASTES Flow:

1. Get next object
2. Pick $\rightarrow$ Navigate $\rightarrow$ Place
3. Repeat until done

Resources

[GitHub] YASMIN

https://github.com/uleroboticsgroup/yasmin

[GitHub] YASMIN PyRoboSim

https://github.com/mgonzs13/yasmin_pyrobosim

[Docs] Documentation

https://uleroboticsgroup.github.io/yasmin

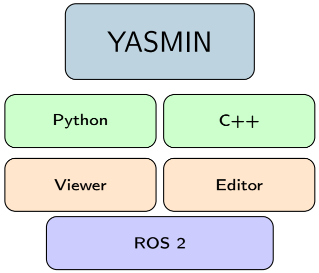
[Docker] Docker Hub

https://hub.docker.com/r/mgons/yasmin

Summary

YASMIN provides:

- ✓ Easy-to-use FSM framework
- ✓ Native ROS 2 integration
- ✓ Python and C++ support
- ✓ Built-in visualization tools
- ✓ XML-based declarative definition
- ✓ Active development and support



Thank you!

Questions?