**Component Design**

For Multiagent Control of Traffic Signals

Version 1.0

Submitted in partial fulfillment of the requirements of the degree of MSE

Bryan Nehl

CIS 895 – MSE Project

Kansas State University

Table of Contents

[1 Intro 3](#_Toc323102433)

[2 Components 3](#_Toc323102434)

[2.1 Agent 3](#_Toc323102435)

[2.2 BasePlanningAgent 4](#_Toc323102436)

[2.3 JRKL\_Reactive Agent and JSS\_ReactiveAgent 4](#_Toc323102437)

[2.4 Safety Agent 4](#_Toc323102438)

[2.5 Safety Agent Rose Kiln and Safety Agent St Saviours 4](#_Toc323102439)

[2.6 Metrics Agent 4](#_Toc323102440)

[2.7 CommunicationsAgent 4](#_Toc323102441)

[2.8 SignalState 5](#_Toc323102442)

[2.9 SignalStateTests 5](#_Toc323102443)

[2.10 SignalPhase 5](#_Toc323102444)

[2.11 SignalPhaseTests 5](#_Toc323102445)

[2.12 Metric 5](#_Toc323102446)

[2.13 SensorState 5](#_Toc323102447)

[2.14 MactsExchange 5](#_Toc323102448)

[2.15 MactsExchangeType 6](#_Toc323102449)

# Intro

In this document I will review the components that make up the MACTS system. Recall the System Context diagram from earlier in the project. The classes covered here would fit inside of the yellow boxes in the System Context diagram.

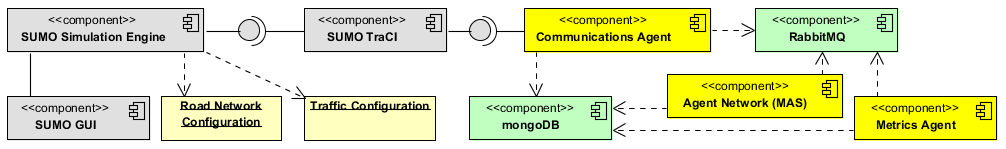


Figure System Context Diagram

# Components

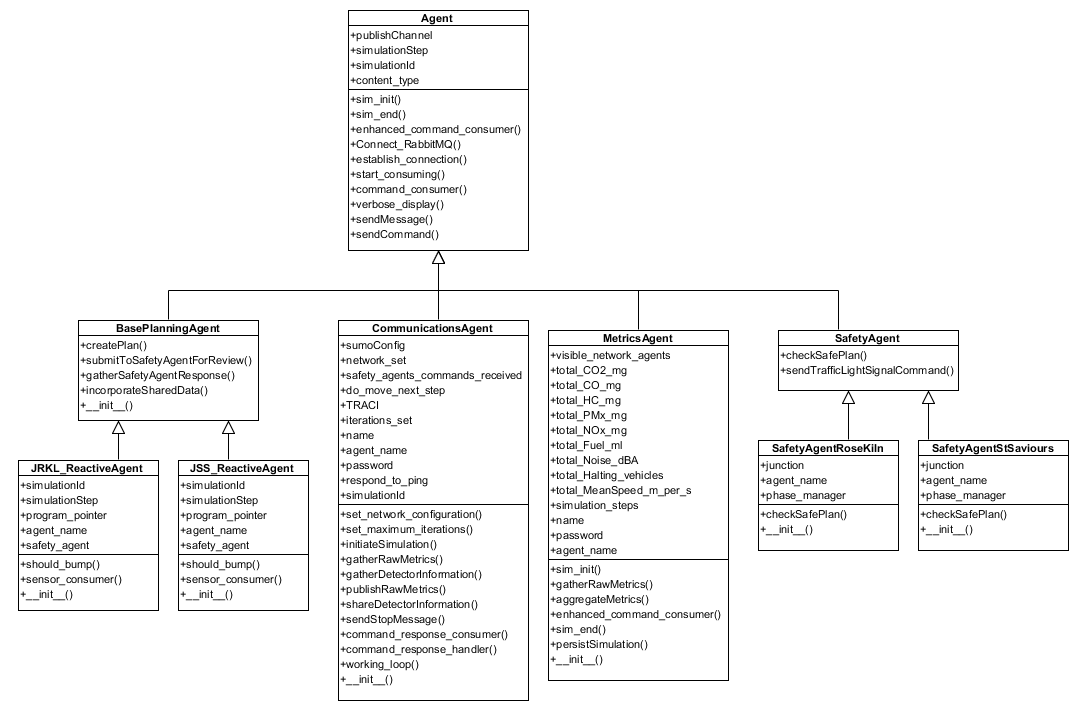


Figure Agent Classes

## Agent

The agent class serves a base class for the other agents in the system. It contains methods for connecting to RabbitMQ, sending messages and receiving and handling some common simulation commands. It has stubs for enhanced\_command\_consumer, sim\_init and sim\_end which may be used by the implementing agent to enhance and extend operation of the agent when it receives commands during operation. In addition the extending code will be able to hook in code that should be run at simulation init and end.

## BasePlanningAgent

This class serves more as a specification for how planning agents should be implemented.

## JRKL\_Reactive Agent and JSS\_ReactiveAgent

These classes are at the heart of the TLS control. They are responsible for deciding the next TLS phase based on current state and sensor input.

*Talk about algorithms.*

## Safety Agent

The safety agent is responsible for checking the submitted phase and verifying that it is a safe transition to make. If the transition is not acceptable it will return a status array which indicates the offending signal.

Exchanges used

Data formats

## Safety Agent Rose Kiln and Safety Agent St Saviours

These agents are concrete implementations of the SafetyAgent class.

## Metrics Agent

The metrics agent is responsible for consuming all simulation run metrics and saving the total values and network configuration to MongoDb.

Exchanges used

Data formats

## CommunicationsAgent

*Lorem ipsum*

Threads

Multiple connections to RabbitMQ

Exchanges used

Data formats

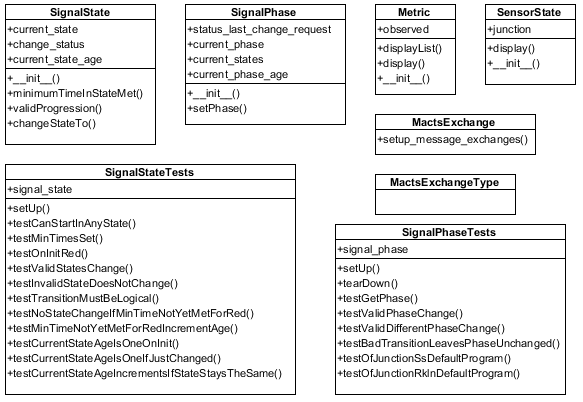


Figure Supporting Classes

## SignalState

What it does

Room for extension

## SignalStateTests

## SignalPhase

## SignalPhaseTests

## Metric

## SensorState

## MactsExchange

## MactsExchangeType

This class is a simple container for constant values used when declaring a RabbitMQ exchange.