**System Architecture 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 Introduction 3](#_Toc319500625)

[2 References 3](#_Toc319500626)

[3 Architecture 3](#_Toc319500627)

[3.1 System Analysis 3](#_Toc319500628)

[3.2 Component Design 5](#_Toc319500629)

[3.3 Component Interface Specification 7](#_Toc319500630)

[3.4 System Design Rationale 7](#_Toc319500631)

[4 Mid-Level Design 7](#_Toc319500632)

[5 Component Interaction 8](#_Toc319500633)

[6 USE/OCL Model 9](#_Toc319500634)

# Introduction

This document provides system design information for the MultiAgent Control of Traffic Signals (MACTS) system. This system is used to simulate agent based control of traffic light signals. This document covers the system components and component interfaces. However, it does not cover all of the interfaces methods in detail. A system analysis diagram as well as a high-level overview of the whole system is included in this document. Mid-Level design is also included for all of the components. A sequence diagram is included which shows how the system components interact during run time.

# References

1. “Vision Document” available at <http://people.cis.ksu.edu/~bnehl/>.

# Architecture

This section documents the system component design, the interfaces of those components and provides high-level design with rationale for design within the system context.

## System Analysis

Clearly describe the high level relationship between model elements

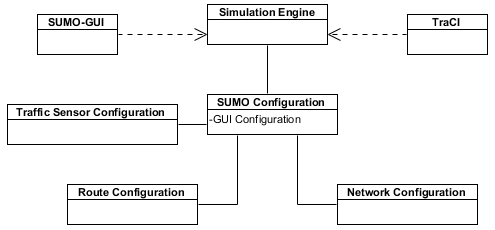


Figure 1System Analysis Diagram

Referring to Figure 1, the Simulation Engine takes care of the work of simulating the movement of the vehicles in the system. The SUMO-GUI is the front-end graphical user interface that displays the state of the simulation. TraCI is the TCP/IP interface to the simulation engine. TraCI is how external entities can interact with the simulation. The Simulation Engine relies on a SUMO Configuration file. The SUMO Configuration file includes specific settings information for the GUI as well as references or pointers to three other configuration files. The Traffic Sensor Configuration file contains information about sensors like the e1 inductor that are on the road network. The Route Configuration includes information about the routes that cars take. Details regarding the types of vehicles, vehicle distribution are specified. For the routes, the flow rates and probabilities are specified. The Network Configuration file is the result of running three files: Nodes, Edges and Connectors through the NETCONVERT utility. NETCONVERT is a SUMO utility. The Nodes, Edges and Connectors files detail where connections happen (nodes), streets are described by the edges and connectors handle the mapping from one edge to another at a junction node.

## System Context Diagram

This system context diagram shows how the components of the MACTS system interact with each other and with the external systems.

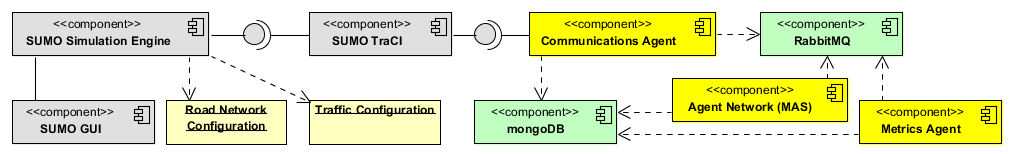


Figure 2System Context Diagram

In Figure 2, the System Context Diagram, grey components are SUMO components, the ivory components are SUMO configuration files created by me. The green components represent third party infrastructure servers. The yellow components (Communications Agent, Agent Network (MAS) and Metrics Agent) are the aspects that I will be creating. What we don’t see at this level is the possibility for multiple Agent Networks. That would come in to play when we have a MAS working at every intersection.

In Figure 3, the basic processing for a single simulation step is shown. Data is received from the simulator and sent to RabbitMQ. From there, there the Metrics Processing pulls data from its queue and does its own parallel operations. In the Analyze Data step, the MAS node planning agent uses sensor information that it received from specific queues. The planning agent then creates a suggested plan and sends it to the safety officer for checking that the command is safe. At that point, if the plan is safe, operation continues on the happy path sending the commands to the communications agent. Otherwise, the planning agent is informed that the plan isn’t safe.

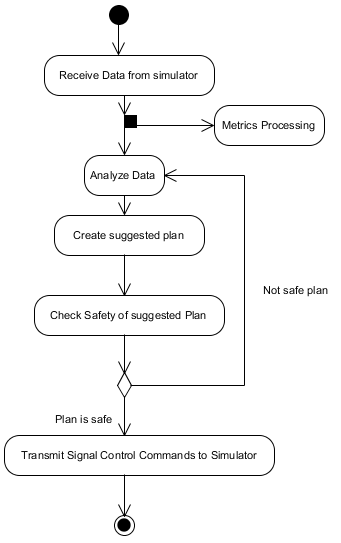


Figure 3Basic Processing for single simulation step

## Component Design

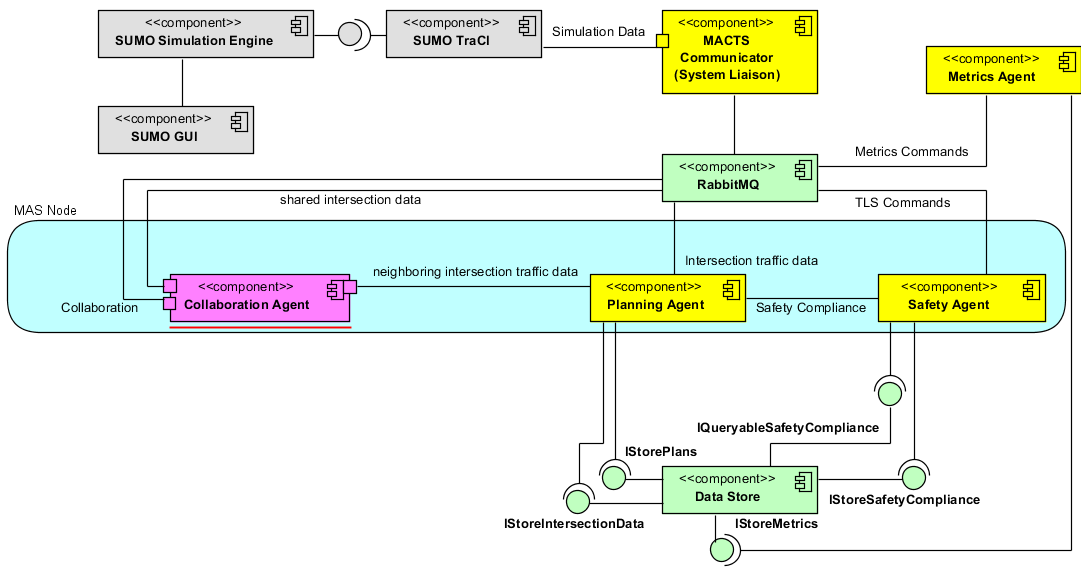


Figure 4MACTS with single MAS Node

*Description of system component responsibilities here.*

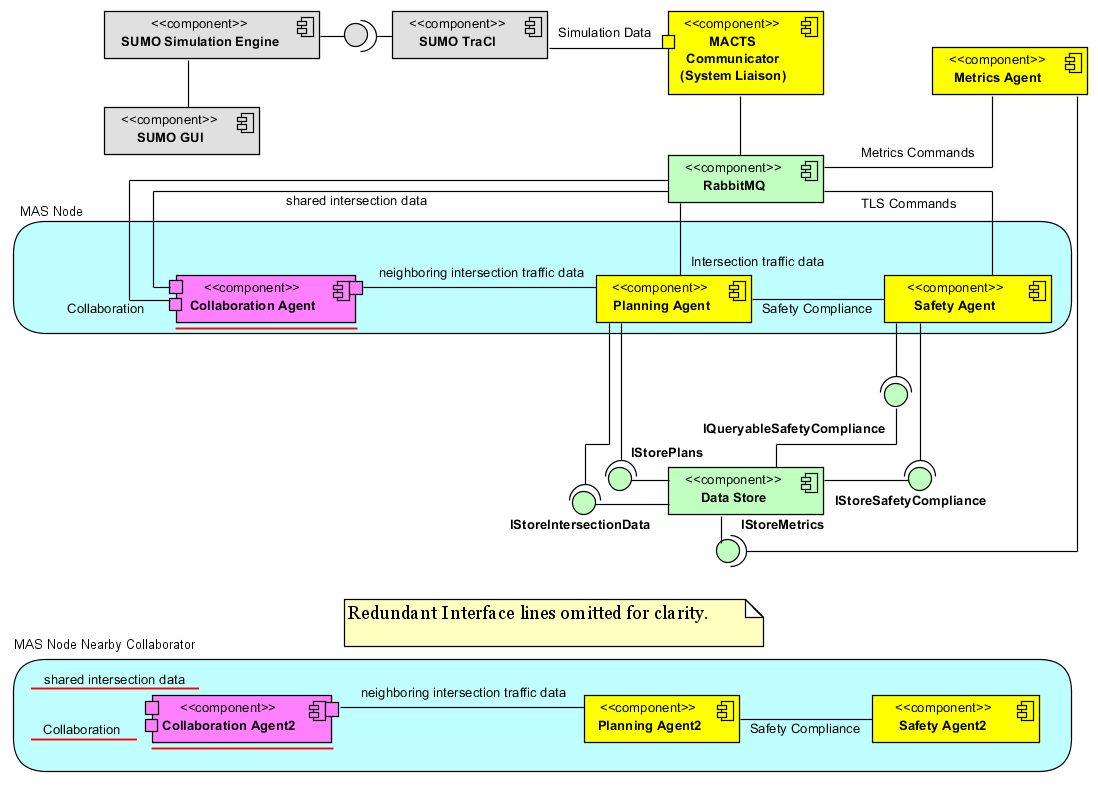


Figure 5MACTS with collaborating MAS Node

*Description of how additional MAS Node interacts here.*

## Component Interface Specification

*Documentation of key interface members for system components here.*

|  |  |
| --- | --- |
| **Signature** |  |
| **Purpose** |  |
| **Pre-Conditions** |  |
| **Post-Conditions** |  |

## High-Level Design

## System Design Rationale

*Description of rationale behind design here.*

# Mid-Level Design

*Class Diagrams here*

# Component Interaction

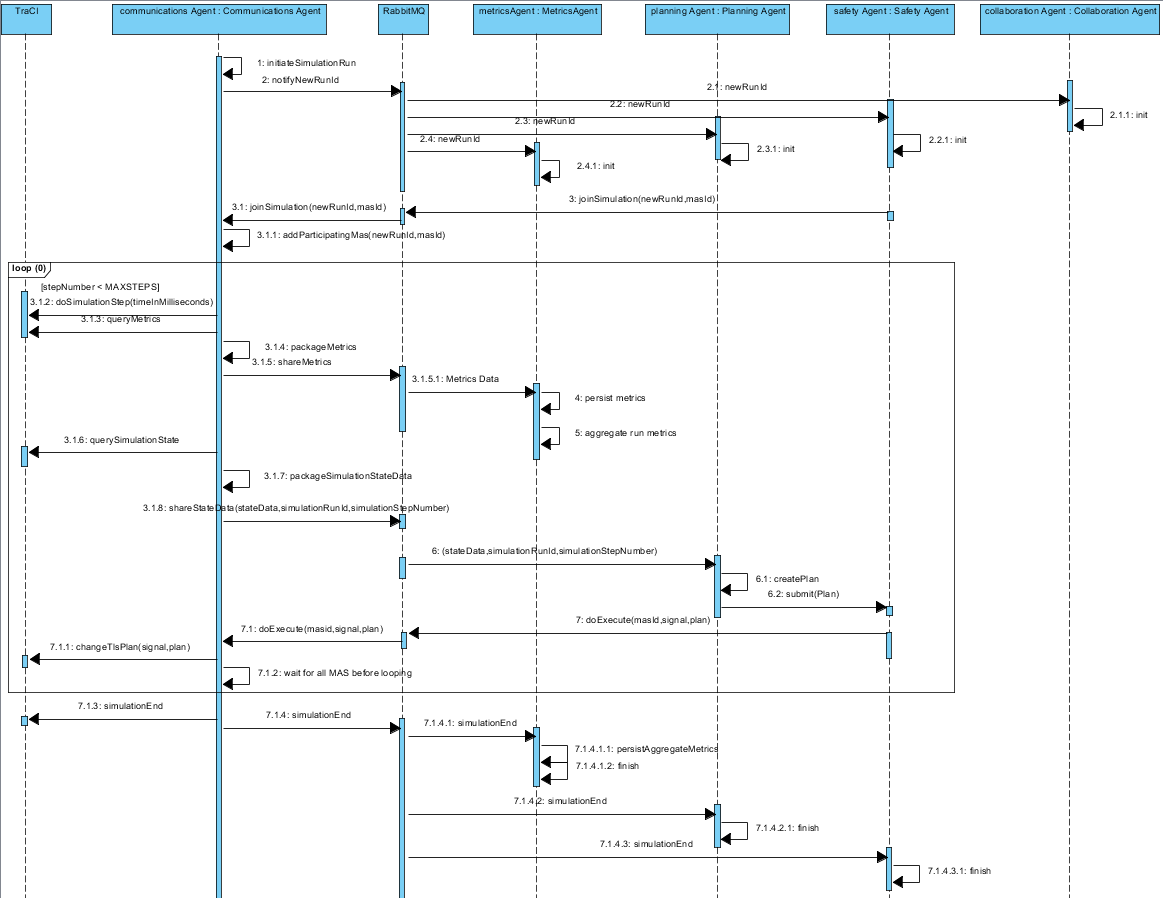
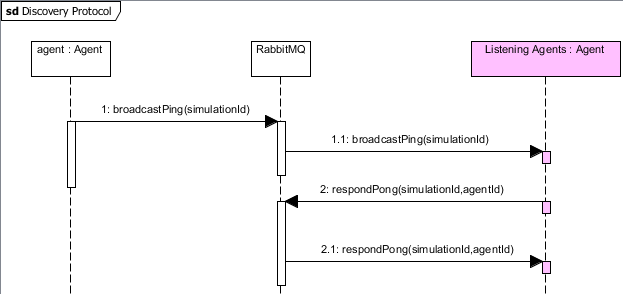


Figure 6Sequence diagram for process interactions

## Discovery Protocol



# USE/OCL Model

-- CIS 895 MSE Project Formal Specification MACTS Architecture

--

-- macts.use

--

-- The MACTS model rendered in USE OCL,

--

-- Author: Bryan Nehl

--

model Macts

-- classes -------------------

-- external, only one interface to this

class TraCI

end

-- abstract, no instances of

class Agent

end

-- one

class CommunicationsAgent < Agent

end

-- one

class MetricsAgent < Agent

end

-- multiple uses of

class DataStore

end

-- MAS Node

-- abstract

-- may only have one "PlanningAgent" type

class MasNode

attributes

planningAgent : PlanningAgent

safetyAgent : SafetyAgent

collaborator : CollaborationAgent

end

class PlanningAgent < Agent

end

class StandardTimingBasedAgent < PlanningAgent

end

class ReactiveAgent < PlanningAgent

end

class CollaborativeAgent < PlanningAgent

end

class GeneticAgent < PlanningAgent

end

class SafetyAgent < Agent

end

class CollaborationAgent < Agent

end

-- associations -----------------

association persists between

PlanningAgent[1] role planproducer;

DataStore[\*] role datastore;

end

association interacts between

TraCI[1] role simulator;

CommunicationsAgent[1] role liaison;

end

-- constraints --------------------

constraints

context mn:MasNode

inv planningAgentIsAPlanningAgent:

mn.planningAgent.oclIsKindOf(PlanningAgent)