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

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 are included. 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.

2 References

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

3 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.

3.1 System Analysis

Clearly describe the high level relationship between model elements

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

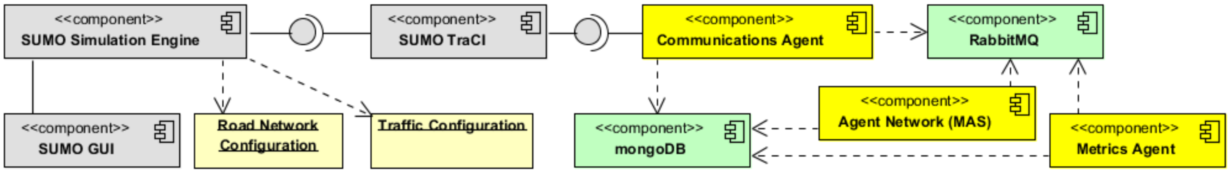

Figure 1System Context Diagram

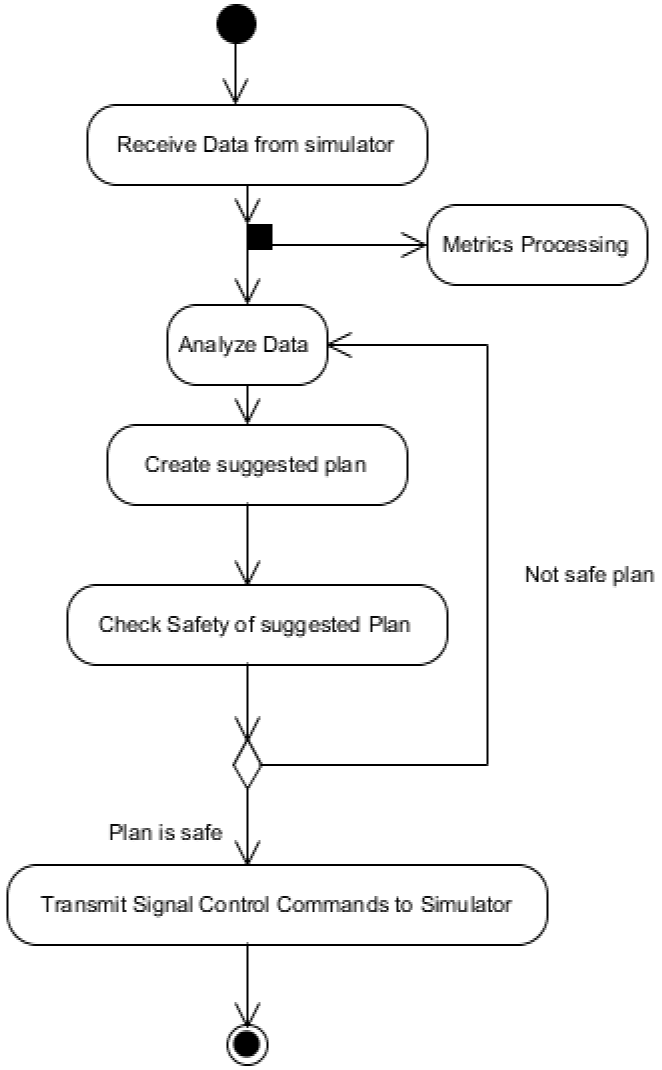
Receive Data from simulator
Metrics Processing
Analyze Data
Create suggested plan
Not safe plan
Check Safety of suggested Plan
Plan is safe
Transmit Signal Control Commands to Simulator

Figure 2Basic Procesing for single simulation step

3.2 Component Design

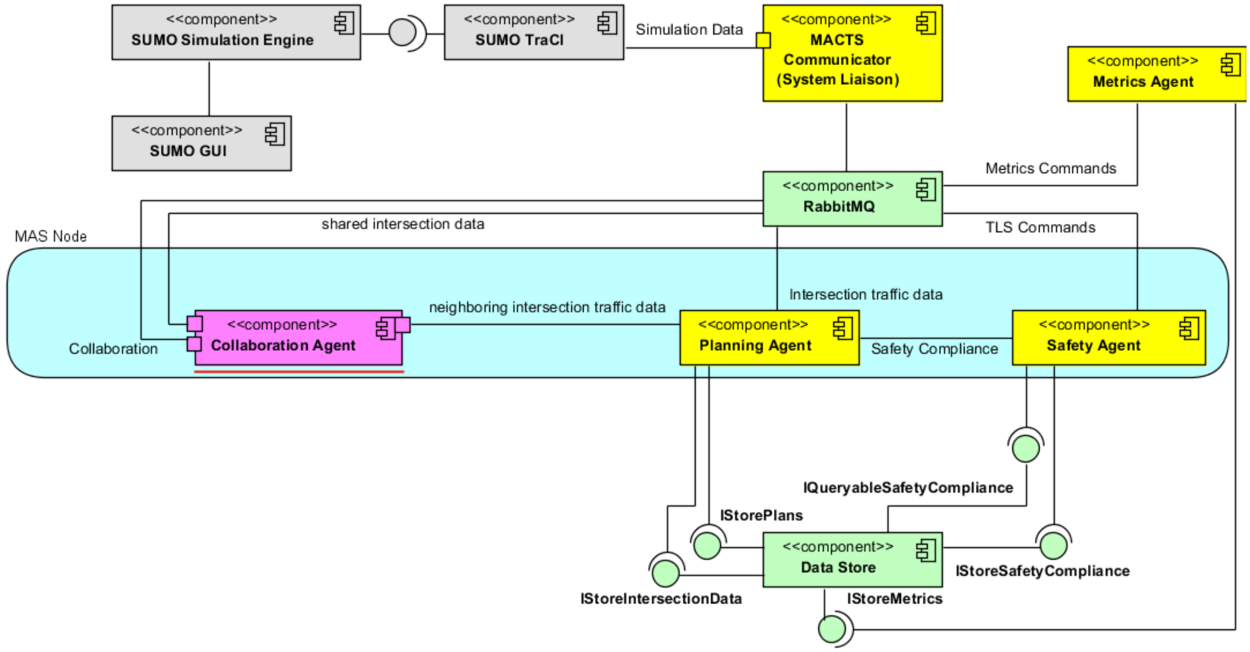

Figure 3MACTS with single MAS Node

Description of system component responsibilitieshere.

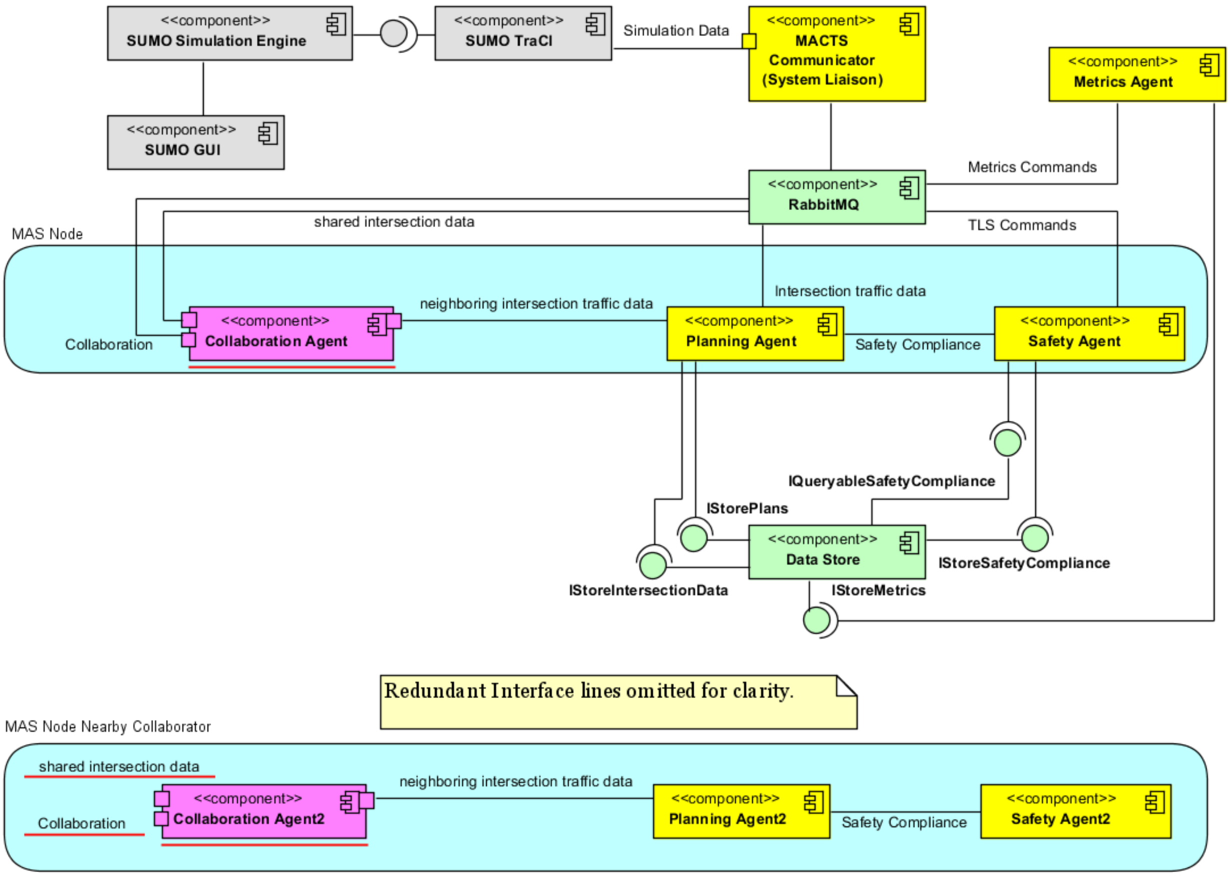

Figure 4MACTS with collaborating MAS Node

Description of how additional MAS Node interacts here.

3.3 Component Interface Specification

Documentation of key interface members for system components here.

3.4 System Design Rationale

Description of rationale behind design here.

4 Mid-Level Design

Class Diagrams here

5 Component Interaction

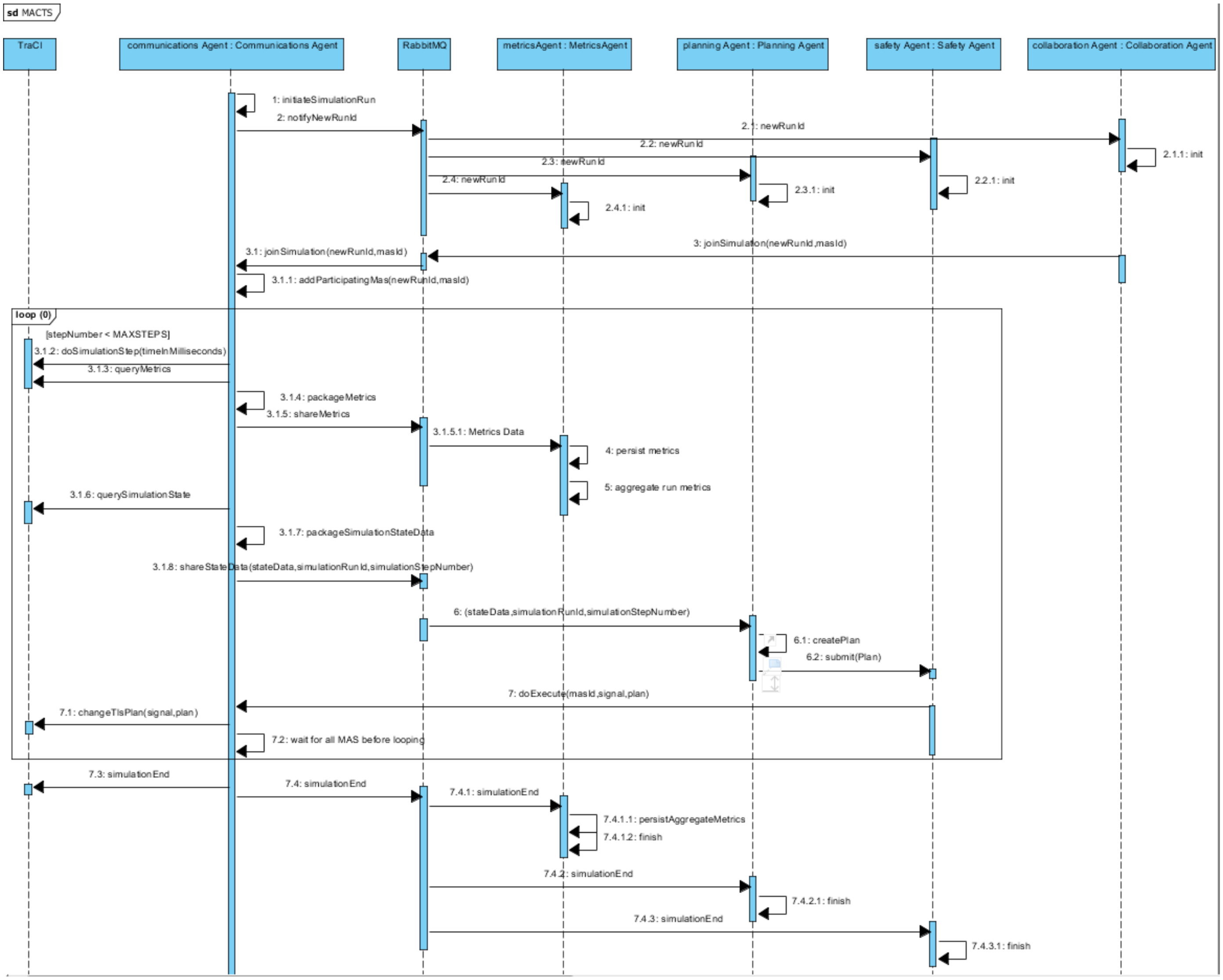
sd MACTS
TraCI
communications Agent : Communications Agent
RabbitMQ
metricsAgent : MetricsAgent
planning Agent : Planning Agent
safety Agent : Safety Agent
collaboration Agent : Collaboration Agent
1: initiateSimulationRun
2: notifyNewRunId
2.1: newRun Id
2.1.1: init
2.2: newRun Id
2.2.1: init
2.3: newRun Id
2.3.1: init
2.4: newRun Id
2.4.1: init
3: joinSimulation(newRunId,masId)
3.1: joinSimulation(newRunId,masId)
3.1.1: addParticipatingMas(newRunId,masId)
loop (0)
[stepNumber < MAXSTEPS]
3.1.2: doSimulationStep(timeInMilliseconds)
3.1.3: queryMetrics
3.1.4: packageMetrics
3.1.5: shareMetrics
3.1.5.1: Metrics Data
4: persist metrics
5: aggregate run metrics
3.1.6: querySimulationState
3.1.7: packageSimulationStateData
3.1.8: shareStateData(stateData,simulationRunId,simulationStepNumber)
6: (stateData,simulationRunId,simulationStepNumber)
6.1: createPlan
6.2: submit(Plan)
7: doExecute(masId,signal,plan)
7.1: changeTlsPlan(signal,plan)
7.2: wait for all MAS before looping
7.3: simulationEnd
7.4: simulationEnd
7.4.1: simulationEnd
7.4.1.1: persistAggregateMetrics
7.4.1.2: finish
7.4.2: simulationEnd
7.4.2.1: finish
7.4.3: simulationEnd
7.4.3.1: finish

Figure 5Sequence diagram for process interactions

6 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)
```