Vision Document

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

The intent of this project is to create an efficient, scalable, modular Multiagent System (MAS) that controls traffic signals given sensor data. A white paper from the THALES group discusses at a high level using genetic algorithms with a MAS to improve traffic flow. That paper also provides a sample road network with traffic flow data that can be used for comparison.

The purpose of this project is to show a Multiagent System can be used to create a more positive experience for the driver and benefit the environment at the same time.

I intend to show improvements over a baseline typical timing based network with the following metrics: travel time, loss time (# of stops), fuel consumption, hydrocarbon production. To show these improvements I will model timing based behavior with a standard SUMO model. Then I will run simulations with simple reactive MAS, more intelligent reactive MAS, MAS that uses genetic algorithms and finally a goal-oriented MAS that uses mesh network based collaboration.

1.1 Motivation

Most traffic light systems today are strictly timing based. Traffic flow studies are required to create timing plans and atypical conditions cause problems. It is also very frustrating to have to stop for a red light when there is no opposing traffic.

1.2 Terms and Definitions

A software agent is a piece of software that interacts with its environment by sensing, reasoning, making decisions and then effecting a change on the environment.

Multiagent System (MAS) is made up of multiple software agents. Some MAS use communication and coordination between the agents. These are typically cooperative environments.

The Simulation for Urban MObility or SUMO is traffic simulation software that is geared towards micro-simulation.

TraCI is a TCP based interface that is used to interact with a SUMO simulation.

RabbitMQ is Advanced Message Queuing Protocol compliant message-oriented middleware.

MongoDB is a document store oriented database system.

1.3 References

SUMO, “Simulation for Urban MObility,” Sep. 2011;
http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Main_Page.

T. Masterton and D. Topiwala, “Multi-Agent Traffic Light Optimisation and Coordination,” white paper, Thales Group, Reference VCS081002, Issue 2, 2008.

2 Project Overview

2.1 Project Goal

The goal of this project is to create a multiagent system that is capable of traffic light signal control which results in an improved travel experience.

2.2 System Context

<<system context diagram>>

MAS ←→ TraCI ←→SUMO←→SUMO Gui

→ Knowledge base, RabbitMQ server

Description of diagram

3 Project Requirements

Driving requirements of project
Use case diagrams and data flow diagrams
The requirements will describe all key functionality required of the resulting system. At a minimum, the requirements will include the valid range of inputs and the expected outputs associated with those inputs. Each requirement will also be given a unique identifier. This document will continue to evolve at least until the architecture presentation and will be continually updated.

3.1 Critical Use Cases

3.1.1 Displays Simulation

Description: The Simulation for Urban Mobility will be used to display the active simulation.
Pre-Conditions: asdf
Details: asdf
Post Conditions: asdf
Specific Requirements:

3.1.2 Provides current simulation state data

Description: asdf
Pre-Conditions: asdf
Details: asdf
Post Conditions: asdf
Specific Requirements:

3.1.3 Produces individual intersection state data

Description: asdf
Pre-Conditions: asdf
Details: asdf
Post Conditions: asdf
Specific Requirements:

3.1.4 Executes traffic light signal plan

Description: asdf
Pre-Conditions: asdf

Details: asdf
Post Conditions: asdf
Specific Requirements:

3.1.5 Information about current intersection status is shared
Description: asdf
Pre-Conditions: asdf
Details: asdf
Post Conditions: asdf
Specific Requirements:

3.1.6 Produce data to share with neighboring intersections
Description: asdf
Pre-Conditions: asdf
Details: asdf
Post Conditions: asdf
Specific Requirements:

3.1.7 Create traffic light signal plan
Description: asdf
Pre-Conditions: asdf
Details: asdf
Post Conditions: asdf
Specific Requirements:

3.1.7.1 Incorporate data that was shared into planning
Description: asdf
Pre-Conditions: asdf
Details: asdf
Post Conditions: asdf
Specific Requirements:

3.1.8 Verify plan is safe
Description: asdf
Pre-Conditions: asdf
Details: asdf
Post Conditions: asdf
Specific Requirements:

3.1.9 Submit plan to be executed
Description: asdf
Pre-Conditions: asdf
Details: asdf
Post Conditions: asdf
Specific Requirements:

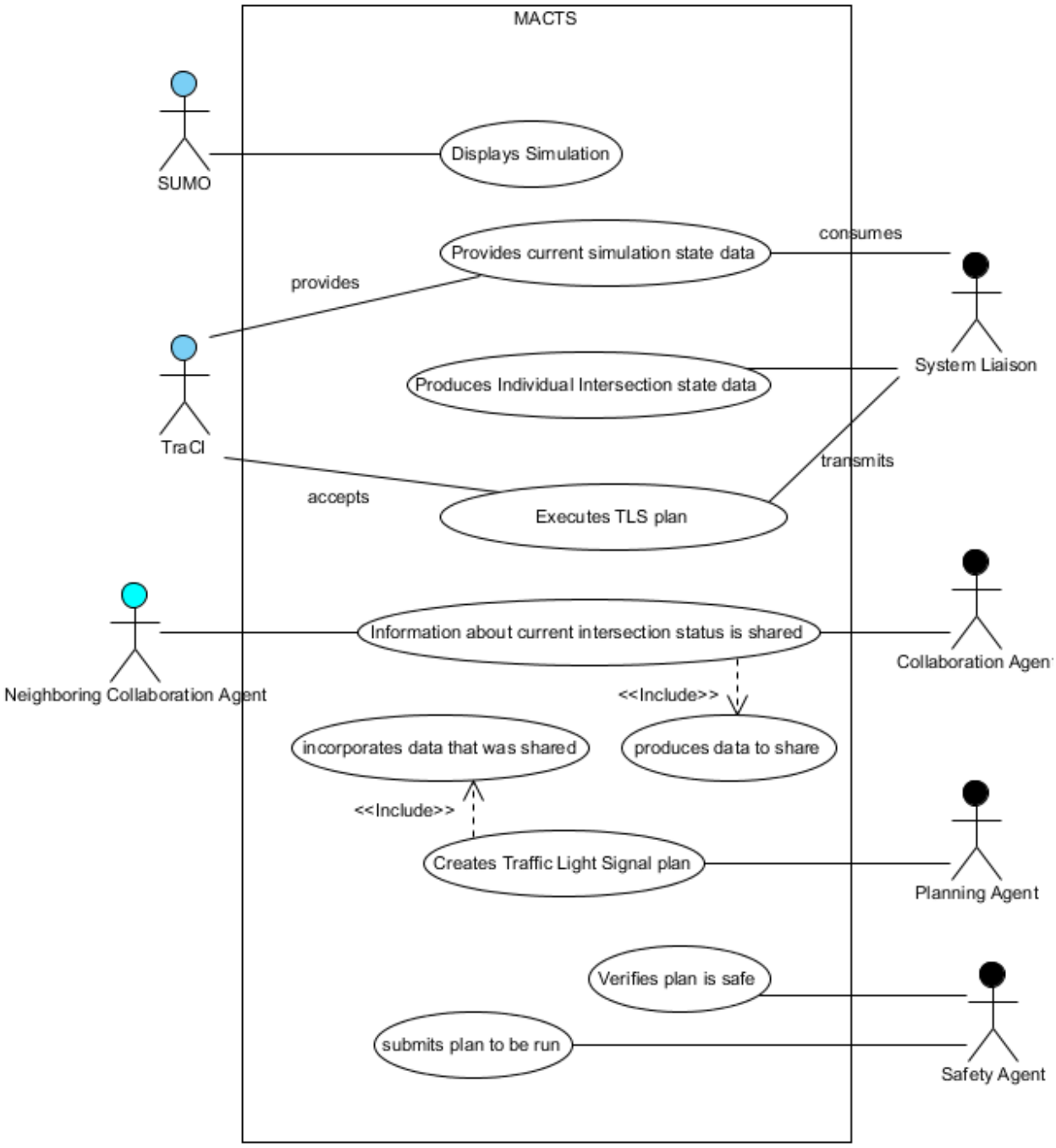

Figure 1Critical Use Cases

3.2 System Activity

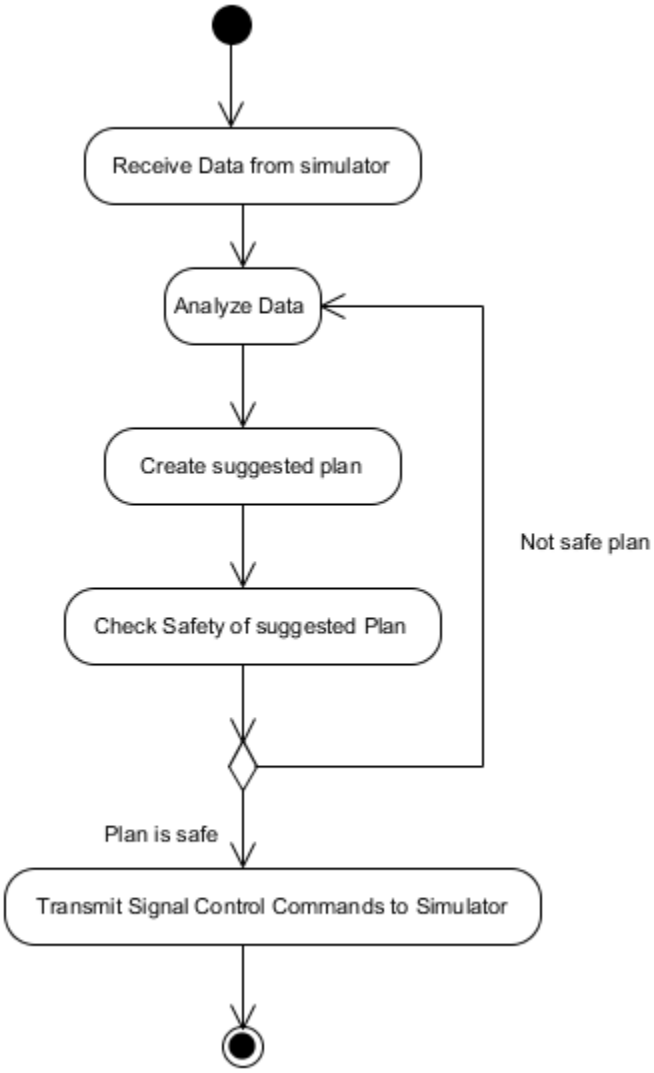

Figure 2UML Activity Diagram for a single simulation iteration

4 Assumptions
Like has specific version of SUMO and Python, etc.

5 Constraints
This system will not attempt to model the behavior of individual cars nor will it visualize system behavior. Those aspects are left to the simulation engine SUMO.

6 Environment

6.1 Purpose
The purpose of this section is to describe the development platform, tools, environment practices and processes that will be employed for the Multi-Agent Control of Traffic Signals project.

6.2 Hardware
I will use a Sony laptop as the prime development station. The laptop has an Intel core I7 processor and sufficient RAM to run the simulations. The computer can support the development environment as well as running the simulation. Additional machines may be used to show the distributed nature of the MACTS system. An External Hard Drive will be used for local but off computer repository. A flash drive will be used for OneNote file sync.

6.3 Tools
This section describes tools (software) that will be used for the project and supporting activities.

6.3.1 Process Support
Google docs will be used for maintaining the engineering notebook, for capturing information about references and for storage of guides, research & reference documents. Swift-Kanban is an electronic Kanban board which will be used for project management. Google Calendar will be used for scheduling work. OneNote is used for collecting project notes and for putting together working documents. Visual Paradigm for UML will be used for creating UML diagrams and when possible for generation or reverse engineering of application source code. MS Word 2010 will be used to create the documentation. Draft documents will be exported in PDF format.

6.3.2 Development
The Simulation for Urban Mobility or SUMO will be the simulation engine that I interface with.
PyCharm is an integrated development environment that will be used for Python 2.7 coding. Python 2.7 was chosen since Python 3 is still new enough that I was having difficulty gathering the necessary interface modules.
RabbitMQ is a message queuing server which I will use for inter-agent communication. The Python module pika will be used to work with RabbitMQ.
MongoDB is a document store and will be used for persisting agent configuration information and for any knowledge base needs. The Python module pymongo is used to work with MongoDB.

6.3.3 Testing
For unit testing, PyUnit the unittest module will be used. If mock objects are needed, mockito-python will be used. However, if mockito-python proves to not be suitable, I'll switch to Michael Foord's Mock. PyCharm's internal code inspection feature will be used to check for common coding errors. To see code coverage of unit tests, Ned Batchelder's Coverage.py

module will be used. PyMetrics will be used for code analysis, particularly cyclomatic complexity.

6.3.4 Version Control

The distributed version control system git will be used for all project documents. That is, code and supporting portfolio documents will all be included in to the repository. I will update the git repository on the KSU CIS server, a local on machine repository as well as an external on hard drive repository during development.

6.4 Process

6.4.1 Development

I will create UML diagrams, unit tests and code to meet requirement acceptance criteria. Before committing code to the repository I will run coverage tests and code metrics. I will seek to have 100% coverage of non-trivial code. I will also strive for a low cyclomatic complexity. As features/requirements are completed, they will be committed.

6.4.2 Releases

There will be a weekly commit to external hard drive and to the KSU CIS server of all current work. Version numbers of Microsoft Word created documents will have their version numbers updated to correspond with the presentations. PDF versions of the Word documents will also be created at the same time. They will be named to relay what presentation they correspond with.

A weekly progress report will be made to my advisor in the form of an email.