Test Plan

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 Test Plan Identifier
MACTS-TestPlan-1.0

2 Introduction
This document describes the testing that will be done for the MACTS. Requirements of the system and detailed design information can be found in the Vision Document 2.0 and Software Quality Assurance Plan 1.0.

3 Test Items
The following system components will be tested. Detailed specifications of these components can be found in the System Architecture Design 1.0 document.

- Communications Agent
- Metrics Agent
- Safety Agent
- Standard Timing Based Agent : Planning Agent
- Reactive Agent : Planning Agent
- Collaborating Agent : Planning Agent
- Genetic Agent : Planning Agent

4 Features to be inspected
The following features from the Vision 2.0 document require an initial inspection.

4.1 SR1 [Critical Requirement]
A road network configuration will be created for the simulation. This network has been built and configured by the developer in a static configuration file.

4.2 SR2 [Critical Requirement]
Traffic route configurations will be created for the simulation. The routes have been built and configured by the developer in a static configuration file.

4.3 SR3
The SUMO graphical user interface (GUI) will display simulation activity. Simulation runs should display activity through the SUMO GUI.

5 Features to Be Tested
The following features from the Vision 2.0 document will be tested.

5.1 SR4 [Critical Requirement]
The Liaison (see Use Case diagram in Figure 2.) requests information from the simulation.

5.2 SR5 [Critical Requirement]
The Liaison checks if there are any command requests or plan submissions from the MAS.

5.3 SR6 [Critical Requirement]
The Liaison submits the commands or plans if there were any provided.

5.4 SR7 [Critical Requirement]

The Liaison instructs the simulation to continue with the next step once all information has been obtained and commands sent.

5.5 SR8 [Critical Requirement]

Translate incoming simulation data. Parse out information for individual intersections.

5.6 SR9 [Critical Requirement]

Publish the individual intersection data to RabbitMQ topic queues.

5.7 SR10

See SR5 above. (The Liaison checks if there are any command requests or plan submissions from the MAS.) The Liaison will check a RabbitMQ TLS command queue. The plan will consist of a script of instructions that the Liaison can send to TraCI.

5.8 SR11

See SR6 above. (The Liaison submits the commands or plans if there were any provided.) The Liaison plays the script of commands, sending them to TraCI.

5.9 SR12

Collaboration agents include a discovery interface which is used to self-identify when an agent sends a broadcast querying for agents that are associated with intersections that send traffic into their intersection or receive traffic from their intersection. Agents respond with a list of queues where their outbound traffic information can be found.

5.10 SR13

Collaboration agents share information regarding traffic leaving their intersection by putting it in distribution queues.

5.11 SR14

The Collaboration agent creates information regarding traffic leaving their intersection.

5.12 SR15 [Critical Requirement]

The planning agent examines incoming data and creates a new TLS plan.

5.13 SR16 [Critical Requirement]

The planning agent submits the plan to the Safety Agent for review.

5.14 SR 17

The Planning agent incorporates data that was shared from other collaboration agents regarding traffic that is flowing into this current intersection.

5.15 SR 18 [Critical]

The safety agent examines the TLS plan to verify that there are no simultaneously active paths that will cross each other in such a way as to create an unsafe condition.

5.16 SR 19

If the plan is not safe, it lets the planning agent know and provides the reason why.

5.17 SR 20 [Critical]

Submit verified safe plan to TLS command queue.

5.18 SR 21 [Critical]
Request direct simulation metrics from TraCI via the System Liaison.

5.19 SR 22 [Critical]
Do any internal processing necessary for computing metrics.

5.20 SR 23 [Critical]
Upon simulation run completion, persist the run metrics to MongoDB. Corresponding network configuration information will be stored with the metrics.

6 Approach
I will establish baseline metrics for comparison by running 5 simulation runs of the default timing based configuration for a simulated hour of time. I will use the average of the results as a basis for comparison.

Black box testing will be done by watching the simulation and checking run metrics.

White box testing will be done by monitoring interactions between agents. This could be done by reviewing, verifying and comparing the output log files, watching message traffic in the RabbitMQ queues and/or reviewing information captured in the MongoDB document store.

I will build test scenarios for light (10%), medium (50%) and full traffic load (100%).

It is my desire to see better metrics for the agent based approaches than the straight timing based approach. Better would be defined as reduced wait times, increased over all travel speed, reduced emissions.

7 Item Pass/Fail Criteria
Items will be considered passing if they meet the expectations as specified in the Vision 2.0 document. Not meeting the expectations it will be considered failing.

8 Suspension Criteria and Resumption Requirements

8.1 Suspension Criteria
If a failing test happens, further testing of the requirement and dependent requirements will be suspended. Test cases that do not rely on the failing feature may be tested. Any time spent modifying the code base as a result of failed tests should be documented as rework effort per the SQAP.

8.2 Resumption Requirements
When the root cause of the failure has been identified, resolved and documented in the testing log testing can resume.

9 Test Deliverables
A test log will be created as part of the testing process. The log will identify the test case, the success or failure of the test. If there was a failure the root cause will be identified and documented along with the corrective action taken.

10 Testing

10.1 Requirements Based Testing

Requirements based testing is testing that is done with the goal of verifying system behavior for specific requirements.

Scenario	System Requirement Tested	Passing Criteria
Do basic simulation run	SR4, SR8, SR9	The communication agent puts simulation state and metric data into RabbitMQ queues.
Do basic simulation run	SR5, SR6, SR7, SR10, SR11	Commands put into RabbitMQ Command Queue are retrieved and submitted to TraCI. The do next simulation step instruction is sent.
Collaborating agent simulation run.	SR7, SR10, SR11	Communication agent waits for all MAS Nodes to respond before sending do next simulation step.
Collaborating agent simulation	SR12, SR13, SR14	Review message queues and log files to see that collaboration is occurring.
Do basic simulation run with Reactive Agent	SR15, SR16	Review output logs
Do basic simulation run with Collaborating Agent	SR15, SR16	
Do basic simulation run with Genetic Agent	SR15, SR16	
Do collaborating agents run	SR15, SR16, SR17	
Unit test simulating calls with unsafe conditions	SR18, SR19	Doesn't permit unsafe TLS configurations. Reason returned to requester via RabbitMQ.
Do basic simulation run	SR18, SR20	Safe commands are submitted to command queue.
Do basic simulation run	SR21, SR22, SR23	Review output log for simulation step and aggregate metrics. Check MongoDB for aggregated metrics.

10.2 Scenario Based Testing

The scenario based testing consists of running the scenarios for light, medium and heavy traffic loads for the basic timing agent, the reactive agent, the collaborating agents and the genetic agent MAS five times and capturing the resulting metrics. No system failures should be observed. Metrics for the runs will be graphed for comparison and included in the Test Log.

11 Environment Needs

The MACTS will be tested on a Windows 7 machine with:

- Python 2.7 environment
- Supporting Python libraries
- MongoDB Server
- RabbitMQ Server
- SUMO 14.0