Development Environment

Multi-Agent Control of

Traffic Signals

Bryan Nehl

MSE Candidate

Kansas State University

# Purpose

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

# 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.

# Tools

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

## 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.

## 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 if a data store is needed. The Python module pymongo is used to work with MongoDB.

## 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. PyChecker will be used for checking for typical errors, similar to C’s lint. 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.

## 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.

# Process

## 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.

## Releases

There will be a weekly commit to external hard drive and to the KSU CIS server of all current work. Any Microsoft Word documents that were updated will also have their last modified date time stamp updated and a PDF version generated.

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