Analysis, Design and Implementation of an Agent Based System for Simulating Connected Vehicles

Elahe Paikari, Behrouz H. Far

Department of Electrical and Computer Engineering, University of Calgary, AB, Canada, {Paikari, Far}@ucalgary.ca

Abstract- PARAMICS traffic microsimulator is a popular simulator among universities and government agencies since it is capable of representing many parts of the world's street maps and designed to handle scenarios ranging from a single intersection to a congested freeway, or the modeling of a complete traffic system. However, it lacks the ability of simulating Connected Vehicle (CV) system and its applications of the Intelligent Transportation System (ITS) through designated traffic simulation network. In this study, we utilized the Multi Agent System Engineering (MaSE) methodology, step by step, to model CV as a Multiagent System (MAS). We implemented the MaSE artifacts as extensions for the PARAMICS using two APIs (Application Programming Interface) to add the ability to simulate CV systems. In this paper we provide detailed explanation of the MAS design and at the end introduce two experiments, made based on this research, as the case studies to evaluate the proposed CV system for estimating and improving traffic safety and mobility parameters in the network.

Keywords – Traffic simulator; PARAMICS; Connected Vehicles; Intelligent Transportation System; Application Programming Interface; MaSE methodology

I. INTRODUCTION

Chain collisions can be potentially avoided, or their severity lessened, by reducing the delay between the time of an emergency event and the time at which the vehicles behind are informed about it [1]. One way to provide more time to drivers to react in emergency situations is to develop ITS applications to create connected vehicle (CV) systems using emerging wireless communication technology. The primary benefit of such communication will be to allow the emergency information to be propagated among vehicles much quicker than a traditional chain of drivers reacting to the brake lights of vehicles immediately ahead.

CV research has the potential to improve safety, reduce congestion, benefit the environment and enhance traveler services by enabling vehicles to wirelessly communicate with roadside infrastructure, nearby vehicles, cell phones, and other mobile devices [2, 3]. The basic connected vehicle concept is the establishment of a networked environment between vehicles and roadway infrastructure (V2I) and among vehicles (V2V) through wireless communications. With connected vehicles, V2V, V2I and other services are integrated to work together [4].

V2V is a main component of vehicular communication systems and allows detailed information to be exchanged among the individual vehicles. V2V systems may lead to preventing road collisions and alert motorists and decreasing travel time and crash risk in traffic networks in case of a reasonable penetration rate, i.e. percentage of equipped vehicles [5]. In Vehicle Infrastructure systems (V2I), vehicles exchange information with roadside beacons, which are fixed. These beacons act as an interface between the vehicle network and external networks [5]. V2I takes an effective role in improving road safety and mobility through multiple in-vehicle and roadside units' technologies. The information collected by these roadside components will be shared with the transportation infrastructure operators who will in turn adjust the operation of various control devices (e.g. VMS) to maximize the efficiency of the transportation system and improve safety in response to traffic demand and road condition. VMS (Variable Message Sign) is an electronic message board located close to a roadway. It represents a cost-effective mechanism for disseminating information to drivers unequipped to receive guidance about advisory speed or incidents ahead. Hence, unequipped drivers can be directly influenced through VMS messages.

In this research, extensions of PARAMICS for CV have been designed using MaSE methodology and implemented by two APIs. API #1 creates predefined [5] or random incidents [6] in the network by taking user inputs on the probabilities of collisions and weather-related incidents. API #2 simulates the broadcasting of V2V and V2I communication in the network.

II. BACKGROUND

Microscopic traffic simulation models are becoming increasingly important tools in modelling complex transport networks and evaluating various traffic management alternatives in order to determine the optimum solution for traffic problems that cannot be studied by other analytical methods. In the transportation simulation field there is a general agreement that micro-simulation, i.e., a computational resolution down to the level of individual travelers, may be the only answer to a wide variety of problems. Most existing commercial programs, such as PARAMICS [7] and VISSIM [8], provide ways of simulating traffic models which usually require quite computer related knowledge and coding efforts. Since they are closed-source, they provide API functions through which underlying simulation logic can be changed. Thus, researchers can simulate traffic models other than built-in models through these API functions.

Numbers of researchers investigated on providing comparative and functional evaluations on various traffic simulators [9-13]. From all of the studied simulators, the models AIMSUN [14], PARAMICS, and VISSIM are found to be suitable for congested arterials and freeways, and integrated networks of freeways and surface streets. Also, these models are potentially useful for ITS applications. While these packages have many similarities, each has its own specific characteristics that make it more or less suitable for certain modelling purposes.
In this research, PARAMICS was selected for its outstanding illustration capabilities for a potential demonstration purpose. Due to its scalability, capability, its use in previous works examining variable speed limits and real-time crash risk [15], proven background on freeways, urban roads and VMS concept, which is used in this study, PARAMICS is able to simulate ITS applications required for implementing and evaluating CV systems properly and allows users to extend and test their own traffic control strategies. Given that, with the use of API, PARAMICS satisfies the transmission/generation requirements of warning messages.

Although, PARAMICS is beneficial over other types of microsimulation packages, it lacks the ability of simulating CV systems and their applications in the designated traffic simulation network. Moreover, simulated vehicles in PARAMICS strictly operate under car-following and lane changing models, which lead to an ideal error-free driving world, and therefore zero incidents happen. To allow for testing CV applications, it was necessary to manipulate the model in a way that a predefined incident or various stochastic incidents such as running red light, rear end collision, and weather caused incidents can be reproduced in the simulation world.

The MaSE methodology is a detailed and top-down approach which will cover information needed in models. Also it is a full-lifecycle methodology for analyzing, designing, and developing heterogeneous MAS. MaSE views MAS as a further abstraction of the object-oriented paradigm where agents are specialized objects and it builds upon well-founded object-oriented techniques and applies them to the specification and design of MAS.

The MaSE Analysis phase consists of three steps: Capturing Goals, Applying Use Cases, and Refining Roles. The Design phase has four steps: Creating Agent Classes, Constructing Conversations, Assembling Agent Classes, and System Design. A major strength of MaSE is the ability to track changes throughout the process [16]. Every object created during the analysis and design phases can be traced forward or backward through the different steps to other related objects. For instance, a goal derived in the Capturing Goals step can be traced to a specific role, task, and agent class. Likewise, an agent class can be traced back through tasks and roles to the system level goal it was designed to satisfy.

In the MAS, agents coordinate their actions via conversations to accomplish individual and community goals instead of objects whose methods are invoked directly by other objects. Since our proposed system consists of several modules, such as CV and infrastructures, which send messages to each other in order to connect and transmit warning messages, we selected MAS as the approach to the simulation network. MAS composed of multiple interacting intelligent agents used to solve issues which are hard for an individual agent or module to solve. Our proposed CV model multi-system includes 4 independent agents, which can offer different functions and accomplish CV simulation. In this modeling process, there will be V2V, Non V2V, Database and PARAMICS agents.

III. ANALYSIS

The overall approach in the Analysis phase is defining the system goals from a set of functional requirements and then defining the roles necessary to meet those goals. While a direct mapping from goals to roles is possible, MaSE suggests the use of Use Cases to help validate the system goals and derive an initial set of roles.

A. Capturing goals

The proposed system consists of four components: CV, non-CV, RSU (Road Side Units), and VMS. CV is equipped with a wireless system to create V2V and V2I connection. RSU, the infrastructure placed within the network, is capable of collecting information about traffic conditions and communicate with the CV located within a Dedicated Short Range Communication (DSRC) range, and sending this information to the VMS. VMS located in network to show warning messages and traffic information collected by RSU to communicate with non-CV. Non-CV do not have the equipment to connect with other vehicles or RSUs.

In the first step, we created a set of functional requirements:

1- Two types of vehicles are created in the network (CV/Non-CV).
2- CVs are connected to each other and RSUs.
3- Non-CVs have a connection with RSUs.
4- RSUs and VMS are linked.
5- If a vehicle is in accident, its flag is set to 1.
6- All the CVs and RSUs are informed about the accident.
7- Non-CVs are notified about the accident via VMS.
8- Consequently, some vehicle might change their route or speed.

From the requirements these goals are extracted:

1- Generate vehicles (CV/Non-CV) and infrastructures (RSU/VMS).
2- Create connections.
3- Transmit messages.
4- Change route/speed.

Next, the goals are analyzed and put into a hierarchical form. A Goal Hierarchy Diagram is a directed graph where the nodes represent goals and the arcs define a sub-goal relationship. The overall system goal is placed at the top of it, which is simulating CV as an independent agent added to PARAMICS module. Once a basic goal hierarchy is in place, goals may be decomposed into new sub-goals and each sub-goal must support its parent goal. System goals are depicted in a goal hierarchy diagram, as shown in Fig. 1.

B. Applying Use Cases

The objective of the Applying Use Cases step is to capture a set of use cases from the initial system context and create a set of Sequence Diagrams to help the system analyst identify an initial set of roles and communications paths within the system. Use cases of our system are drawn from the system requirements provided before and describe sequences of events that define desired system behavior; they are examples of how the system should behave. By creating use cases we intended to identify paths of communication. Fig. 2 shows the sequence diagram and how each goal can be accomplished.
C. Refining Roles

The purpose of the Refining Roles step is to transform the Goal Hierarchy Diagram and Sequence Diagrams into roles and their associated tasks, which are more suitable for designing MAS. Role definitions are captured in a MaSE Role Model, which includes information on interactions between role tasks and is more complex than traditional role models [17]. Fig. 3 illustrates system role model diagram that shows CV systems’ roles and behaviours in PARAMICS.

The concurrent task model provides a built in timer activity. We build an initial Concurrent Task Model from the scenarios of creating CVs and creating accidents by taking the sequence of messages sent or received by the roles of CV/non-CV and use them to create a sequence of corresponding states and messages. Concurrent tasks in our system are defined in Concurrent Task Models Fig. 4 and are specified as finite state automata, which consist of states and transitions.

IV. DESIGN

There are four steps to the designing a system with MaSE. In the first step we assigned roles to the 4 agent types in the system to create Agent Classes. In the second step, Constructing Conversations, the actual conversations between agent classes are defined while in the third step, assembling Agents Classes, the internal architecture and reasoning processes of the agent classes are designed. Finally the actual number and location of agents in the deployed system are defined. Each of these steps is discussed below.

A. Creating Agent Classes

In the Creating Agent Classes step of the Design phase, agent classes are created from the roles defined in the Analysis phase. The result of this phase is an Agent Class Diagram, which illustrates the overall agent system organization consisting of agent classes and the conversations between them. At this point, the roles and tasks, which an agent class must play, are determined. The conversations, that an agent class must participate in, are derived from the external communications of the agent roles. The Agent Class Diagram is the first design object in MaSE that shows the entire MAS in a way it can be implemented. The CV model MAS mainly defines 4 agent classes as show in Fig. 5.

B. Constructing Conversations

A MaSE conversation defines a coordination protocol between two agents. The Communication Class Diagram, as shown in Fig. 6, is similar to a Concurrent Task Model and defines the conversation states of V2V and non-V2V as the two participant agent classes. This conversation is about constructing connection between CVs and RSUs. CV, the initiator, begins the conversation by sending “create connection” as the first message. When the agent receives the message, it checks if CV is in range of corresponding RSUs. Only the vehicles which their distance from RSUs fall into DSRC range can be linked with them. Otherwise, the request will be declined and RSU cannot get the message from the CV and show it on VMS.

C. Assembling Agents

During this step, the internals of agent classes are created. This is accomplished via two sub-steps: defining the agent architecture and defining the components that make up the architecture. Components consist of a set of attributes, methods, and, if complex, may have sub-architecture. Internal component behaviour may be represented by formal operation definitions as well as state-diagrams that represent events passed between components. Basically, each task from each role, played by an agent, defines a component in the agent class. Fig. 7 shows agents’ architecture of CV model multi agent system.

D. System Design

The final step of the MaSE methodology takes the agent classes defined previously and instantiates actual agents using Deployment Diagram to show the numbers, types, and locations of agents within a system. The concept of instantiating agents from agent classes is similar to...
instantiating objects from object classes in object-oriented programming. Deployment Diagrams describe a system based on agent classes defined in the previous steps of MaSE. Strength of MaSE is that a designer can make these modifications after designing the system organization, thus generating a variety of system configurations. System Deployment Diagram includes 5 agents, there into, V2V Agent and DB1 Agent run in the same physical node, Non V2V Agent and DB2 Agent in one node and PARAMICS Agent run in the other physical node, as shown in Fig. 8. PARAMICS agent acts as the PARAMICS software itself where the other agent classes should be implemented and communicate with it. The Connected vehicles and No Connected Vehicles platforms should be added to the simulation platform to apply CV system as the extension to the software. This is done by programming APIs.

I. IMPLEMENTATION

By using MaSE methodology, the essential agents, relationships and conversations between them, and the particular parameters of proposed CV system were identified. Moreover, MaSE specified how every agent should act in possible scenarios (Figure 2) to eventually reach the goals and sub-goals defined in Analysis phase (Figure 1). This specified design is implemented by adding two APIs to the PARAMICS programmer module. In API #1, the connection between PARAMICS agent to V2V and Non-V2V agents was created through the use of Database. However, the main duty of API#1 is to generate predefined or randomly [6] incidents for running test case scenarios.

Most of the proposed design by MaSE methodology is used in API #2, where both V2V and No-V2V agent classes and their relevant agents (i.e. CV, non-CV, RSU, VMS) were created (Figure 5). Additionally, the roles and tasks for each agent class and the conversations between these agent classes were programmed, such as the example conversation showed in Figure 6. It specifies the procedure of creating connection and message transmission between CVs and RSUs when they are in predefined DSRC range, and then showing the message context on the VMS. API #2 consists of 2 packages: V2V package in which V2V agent class was implemented and V2I package where the Non-V2V agent class were created. API #2 attempts to mitigate or to minimize the effect of the generated incidents by creating V2V and V2I communications.

Once incidents were created (randomly or predefined) applying API #1, the second API, API #2, accordingly simulates the broadcasting of V2V and V2I communication in the network and models its effects on drivers’ behavior. This API was developed to allow vehicles communicate with each other and infrastructures to apply the connection and conversations between V2V and Non-V2V agent classes. When a CV is involved in an incident, API #2 turns the vehicle’s color to red to indicate this condition. This allows users to visually pick out which vehicle on the studied network was involved in the incident. API #2 then calculates the distance of surrounding CVs, notifies them of the incident ahead and provides an advisory speed. At this point, all notified CVs will have an improved awareness and a decreased aggressiveness (vehicle parameters that programmed intentionally). These two modules will be set randomly for different CV. In other words, their awareness ranges between 6 and 9, whilst their aggression rate will be between 1 and 4 (the threshold of awareness module is 1-9).

API#2 allows CVs to communicate with the RSUs. When CVs encounter an incident, they send a message to the nearest RSU along the road and transmit the location and the type of the incident to the unit. The corresponding RSU will forward aforementioned information to the control center. The control center defined as a decision maker chooses which VMS should be enabled to reflect the message to the upstream and downstream vehicles and what will be the context of message. Usually the message contains warning information for drivers to let them know that there is a collision ahead alongside an advisory speed. The advisory speed is lowered in increments as the distance from collision decreases and it is increased for downstream of traffic to let the congestion around the accident be cleared easily.

API #1 also retrieves and stores the simulated vehicles’ information from PARAMICS. Vehicle information storage is completed through a vehicle map. Each vehicle’s information is stored in the Vinfo struct. Vinfo struct includes information related to vehicles: Float x, Float y, Float z, Int ID, Float

![Figure 4. Concurrent Task Diagram](image4.png)

![Figure 5. Agent Class Diagram](image5.png)

![Figure 6. Create Connection conversation initiator and responder](image6.png)
Figure 7. System architecture for agent class bearing. Float gradient, Float length, Float width, Int age, Bool incident, Int usertag, Bool stopped, Bool collide_stop, Float stop_time, Float distance, VEHICLE *vpoint, LINK *link, Int lane.

API #2 retrieves CV information from the Vinfo struct. The retrieved information includes coordinate information for calculation of the DSRC range. The class of functions involved in creating incidents, information storage, information retrieval and sending V2V and V2I messages will be described in the following sections. The developed APIs along CV agents is shown in Fig. 9.

A. API #1

API #1 creates incidents and requires a set of functions to extract vehicle information from the network, to check for incidents that have occurred, to identify and register any incidents, and finally to create a readable database of vehicles. A majority of its classes of functions are described below.

Qpx_NET_postOpen is a PARAMICS extension (prefix qpx) built-in function. When a model is first opened in PARAMICS, the postOpen function is called. The total number of links and lanes in the network is recorded by its functions and vehicles and infrastructures are generated in the traffic network.

Qpx_NET_second, its functions are executed every second when the simulation is running. It is essentially the core of the program. The subroutine includes all of the functions in API#1.

Qpx_VHC_arrive, its functions remove the vehicle from the vehicle map when a vehicle reaches its destination zone.

InformationExtraction sets up the vehicle map. The vehicle map is where vehicle information is stored. API #2 retrieves vehicle information from the vehicle map.

Accident generation determines when an incident should occur. It can be a predefined incident or it can be several numbers of incidents based on the probability that user provided when the model is first loaded.

B. API #2

API #2 reads vehicles’ information in the vehicle map and stores all V2V enabled vehicles in a V2V map. It then searches for occurred incidents in the network and sends warning messages to surrounding vehicles and the nearest RSUs fall in DSRC protocol. These messages increase driver awareness and enable them to avoid a consequent collision. When the incident has been totally cleared, the situation resumes to the initial state.

Qpx_NET_timeStepPostLink checks every link and adds CVs to the V2V map and infrastructures to the network. The V2V map consists only of CVs. It updates vehicle positions in the V2V map every time step and checks the incident flag for each vehicle. If there has been an accident, it calls the Send Message class of functions. If accident has been cleared, it gets back to the initial state.

SendMessage, when a vehicle is involved in an accident, its Vinfo struct is sent to the sendMessage. It iterates through the V2V map and calls calculateDistance. If any CV is within the DSRC range (1000m) of the vehicle involved in an accident, that vehicle is signaled of the danger ahead. With the increased driver awareness, the vehicle does not partake in the same incident. Based on the DSRC, CV in incident sends message to RSU, RSU sends it to control center and control center decides about the advisory speed and which beacons should show that. It should be noted that DSRC can be changed accordingly for experimenting its effects on CV system.

CalculateDistance receives vehicles’ Vinfo struct classes, calculates and returns the distance between the two V2V enabled vehicles. It can also calculate and returns the distance between vehicle and RSU.

II. CASE STUDY

To test the ability of the CV model during road hazard, two studies have been assessed [5, 18]. Evaluation scenarios (Fig.2) were developed for the morning AM peak traffic from 7:00 AM to 8:10 AM. The proposed model in PARAMICS was used to study the impact of deploying CV on an 8-km southbound section of Deerfoot Trail, Calgary, Alberta.

In these case studies, V2V and V2I communications were evaluated under various load conditions. Moreover, advisory speed recommendation and re-routing guidance were implemented to recommend the optimum treatments, reduce rear-end and lane change crash risks and decrease travel time.

In the first case study, several incidents happen in the network based on the probability provided by user in the start
of the simulation to evaluate CV system in hazard condition. The advisory speed recommendation was only for upstream traffic of the location at accident where speed differences between upstream and downstream vehicles were high. Also, safety and mobility indices were measured where the percentage of CVs changes but the other elements including demand factor remain constant.

Although the first case study found the improvement in travel time and crash likelihood, we did not develop the DSRC range as a factor for distributing messages in V2I module and we only focused on upstream traffic. In the second case study, an incident that blocks one lane was generated at predefined location to precisely study the impact of CV applications. Also, we demonstrate effect of considering DSRC, re-routing guidance and advisory speed for upstream and downstream traffic, where we evaluated different demand loading. DSRC range and penetration rate of CVs. The results show that CV technology can enhance traffic safety and mobility in freeways, if the percentage of CVs is significant (e.g. 30-40%) and the CV technology is accompanied by advisory speed reflected on VMS on both upstream and downstream of the incident location using DSRC range.

III. CONCLUSION

In this study we developed, implemented and demonstrated a DSRC based V2V-assisted V2I traffic information system for estimating and disseminating traffic safety and mobility parameters. One of the main research priorities of the ITS is to facilitate wireless communication between vehicles and infrastructure so that traffic safety information data can be exchanged. We used PARAMICS microsimulator as the traffic network simulator to implement CV systems. Since it does not have the ability to simulate V2V and V2I connections, we designed our proposed CV scenarios using MaSE methodology to view the CV modules as the MAS. By using API, we implemented our system as the extension to the PARAMICS. Two studies have been made to evaluate this system and the overall results showed that applying CV in Calgary’s freeways will improve the safety and mobility applications.

The future work would be enhancing the connected vehicles study by providing a simulation environment that combines the capabilities of a traffic network simulator together with wireless communication functionalities in order to optimize the communication (frequency of information update), provide a cost effective alternative through traffic microscopic and wireless communication simulators to detect failure or latency of communication thorough testing of V2V and V2I communications systems based on different communication and DSRC ranges.

REFERENCES