Development of the Capability-Enhanced PARAMICS Simulation Environment

UCI-ITS-TS-WP-02-11

Lianyu Chu ¹
Henry X. Liu ¹
Will Recker ²

¹ California PATH, ATMS Center, Institute of Transportation Studies
University of California, Irvine
lchu@translab.its.ucl.edu, hlliu@translab.its.ucl.edu

² Department of Civil and Environmental Engineering and
Institute of Transportation Studies, University of California, Irvine
wrecker@uci.edu

August 2002
Institute of Transportation Studies
University of California, Irvine
Irvine, CA 92697-3600, U.S.A.
http://www.its.uci.edu
Development of the Capability-Enhanced PARAMICS Simulation Environment

Lianyu Chu
California PATH, ATMS Center
Institute of Transportation Studies
University of California, Irvine
Irvine, CA 92697
Tel: 949-824-1876 Fax: 949-824-8385
Email: lchu@translab.its.uci.edu

Henry X. Liu
California PATH, ATMS Center
Institute of Transportation Studies
University of California, Irvine
Irvine, CA 92697
Tel: 949-824-2949 Fax: 949-824-8385
Email: hliu@translab.its.uci.edu

Will Recker
Institute of Transportation Studies
University of California, Irvine
Irvine, CA 92697
Tel: 949-824-5642 Fax: 949-824-8385
Email: wrecker@uci.edu

Submitted to 2003 TRB Annual Meeting
ABSTRACT

PARAMICS is one of the widely used microscopic traffic simulation program. One important feature of PARAMICS is that PARAMICS allows users to customize many features of the underlying simulation model through Application Programming Interfaces (API). This paper presents our practices on the development of the capability-enhanced PARAMICS simulation environment, PARAMICS-E, through integrating a number of plug-in modules implemented in PARAMICS API. These APIs complement the functionalities of the commercial PARAMICS model and enable the mechanism of the real-time traffic data collection, processing, and communication, as well as control of operations through dynamic linking. This enhanced PARAMICS simulation can thus have capabilities to model not only the target traffic conditions and operations but also various potential ATMIS strategies.

1 INTRODUCTION

Simulation modeling is an increasing popular and effective tool for analyzing a wide variety of dynamical problems, which are not amenable to study by other means. Traffic simulation models can be classified as being either microscopic, mesoscopic, or macroscopic according to their representation of traffic flow (or vehicle movement). Microscopic models, such as PARAMICS, CORSIM, VISSIM, AIMSUN2, TRANSIM, MIITSIM, continuously or discretely calculate and predict the state of individual vehicles, and measure the speed and location of each individual vehicle in the simulation. Macroscopic models, such as FREFLO, AUTOS, METANET, VISUM, aggregate the description of traffic flow to speed, flow, and density of each link in the network. Mesoscopic models, such as DYNASMART, DYNAIMIT, INTEGRATION, METROPOLIS, have aspects of both macroscopic and microscopic models. These models have been applied successfully to particular studies, but their applications are relatively limited. Most are designed for particular applications and useful only for specific purposes, missing some components of Advanced Transportation Management and Information Systems (1).

Microscopic simulators, which do not depend on theoretical traffic flow models but on vehicle-vehicle interactions, are deemed more appropriate for evaluating various ATMIS applications. PARAMICS (PARAllel MICROscopic Simulation) is a suite of microscopic simulation tools used to model the movement and behavior of individual vehicles on urban and highway road networks (2). One important feature of PARAMICS is that PARAMICS allows the user to customize and extend many features of underlying simulation model through Application Programming Interfaces (API). Such an API should have a dual role, first to allow researchers to override the simulators’ default models, such as car following, lane changing, route choices for instance, and second, to allow them to interface complementary modules to the simulator.

Based on the commercial PARAMICS model, a capability-enhanced PARAMICS simulation environment, PARAMICS-E, has been developed through integrating a number of plug-in modules implemented with Application Programming Interfaces (API).
These APIs complement the functionalities of the commercial PARAMICS model and enables the mechanism of the real-time traffic data collection, processing, and communication, as well as control of operations through dynamic linking. This enhanced PARAMICS simulation can thus have capabilities to model not only the target traffic conditions and operations but also various potential ATMS strategies. This paper will focus on the descriptions of our practices on using APIs to enhance the functionalities of commercial micro-simulation models.

This paper is organized as follows. First, we will give a brief overview of the framework of the capability-enhanced PARAMICS simulation environment. Then, we explain the features of this PARAMICS-E simulation model according to its exclusive features. Finally, concluding remarks are presented.

2. ENHANCING THE CAPABILITIES OF MICRO-SIMULATION

Currently, micro-simulation models are mostly used for evaluation purposes. The simulation world is regarded as the real world where the new algorithms, strategies can be easily tested and evaluated. The new strategies include ATMIS applications, which are regarded as effective operational alternatives of transportation planning methods.

Therefore, whether the simulator can model both the real-world traffic conditions are essential to the successful applications of micro-simulations. The current situation is that each micro-simulator has its own features to simulate the traffic in the real world. These features can be the use of a set of specific models, which are calibrated by specific datasets. Thus it is possible that the simulator cannot correctly model the real-world traffic condition, such as the queuing or shockwave, or a new model is found to have a better representation of the traffic. API programming provides users with the capability to override the simulator’s default internal model and test new alternatives.

In addition, the simulator should have the capabilities to replicate the real-world traffic operations. For example, PARAMICS can basically model the fixed-time signal control. Besides, PARAMICS also provide a plan/phase language to simulate actuated signals, widely used in the real world. However, our experiences found this script language is difficult to be used to model the complex control logic of full-actuated signal control and to replicate the logic to multiple signals. This becomes a reason for us to compliment the functionality of PARAMICS through API programming.

Furthermore, we hope the developed actuated signal API can be re-useable for the future development of some advanced signal control algorithms, which optimize the signal timing based on real-time traffic information. Therefore, appropriate interface functions are required in the basic module of actuated signal API.

3. FRAMEWORK OF PARAMICS-E

3.1 PARAMICS
As a suite of ITS-capable, user-programmable, high-performance microscopic traffic simulation package, PARAMICS offers very plausible detailed modeling for many components of an 'ideal' simulator. Accurate geometry of network and smooth functioning of links in PARAMICS are important for simulation results because driver's behavior relies on characteristics of drivers and vehicles, the interactions between vehicles, and network geometry as well. The ability of PARAMICS to simulate the real-world traffic has been shown by former efforts on the model calibration and validation of PARAMICS (4,5). In the ATMS Testbed at UC Irvine, PARAMICS is integrated to the testbed development environment with the objective to serve as both an off-line evaluation/design tool, and an on-line control/guidance tool for dynamic transportation management.

PARAMICS is fit to ATMIS research due to its high performance, scalability and the ability of modeling the emerging ITS infrastructures, such as loop detectors and VMS (6). In addition, PARAMICS provides users with API through which users can customize and extend many features of the underlying simulation model without having to deal with the underlying proprietary code.

3.2 Framework of PARAMICS-E

![Diagram of PARAMICS-E Framework](Image)

Figure 1 Framework of PARAMICS-E

Based on the commercial PARAMICS model, we establish a capability-enhanced PARAMICS simulation environment, PARAMICS-E, through integrating a number of plug-in modules implemented by API. Figure 1 shows the framework of PARAMICS-E. The core of PARAMICS-E is the commercial PARAMICS model. A set of API modules, including full-actuated signal controller, time-based ramp metering controller, VMS controller, path-based routing, loop data aggregator and performance measure API, are
used for enhancing the capabilities of the commercial PARAMICS model. An API module exchanges dynamic data with the core PARAMICS model and other API modules through the Dynamic Linking Library (DLL) mechanism. Compared to the commercial PARAMICS/PARAMICS model, PARAMICS-E has functionality enhancements in the following aspects:

1. Basic control modules of signal, ramp metering and VMS
2. Traffic data collection, processing, and communication
3. Path-based routing
4. Database connection
5. Performance measures

3.3 Basic control modules

3.3.1 Full-actuated Signal Control

This API implements the eight-phase, dual-ring, concurrent controller logic. The data input to this API is the signal timing plan, the geometry and detector information of each intersection. Interface functions have been provided by this API for external modules to acquire and change the default timing plan. This API provided a couple of interface functions for external API modules to acquire the current signal timing plan and set a new timing plan to a specific signal. An advanced signal control algorithm API can be further developed based on them. The prototypes of these interface functions are shown below (7).

```c
Signal* signal_get_parameters(char *nodeName);
```

Function: Querying the current signal timing plan of a specific actuated signal
Return Value: The current timing plan of an actuated signal.
Parameters: `nodeName` is the name of the signal node.
`Signal` is the structure of actuated signal data, whose definition is:

```c
typedef Signal
{
    // intersection name and location
    char *nodeName;
    char *controllerLocation;

    // signal parameters
    int movements[8];
    float maximumGreen[8];
    float minimumGreen[8];
    float extension[8];
    float storedRed[8];
    int recallPhase[2];

    // current phase information
    int currentPhase;
    int expiredTime;
} Signal;
```
Void signal_set_parameters(Signal *sig);
Function: Setting a new timing plan to a specific signal.
Return Value: None
Parameters: sig stores the new timing plan.

3.3.2 Ramp Metering Control

This plugin is designed to model pre-timed ramp metering control on either one-car-per-green basis or n-cars-per-green basis (with \( n > 1 \)). It also supports multiple timing plans, HOV bypass, and the use of ramp detectors for metering control. The data input of this API is a time-of-day ramp control plan and the detector information of each meter. In addition, this API provides a couple of interface functions for external API modules to acquire the current metering rate and set a new metering rate to a specific ramp meter. An advanced ramp-metering algorithm API can be further developed based on these interface functions. The prototypes of them are shown below.

RAMP *ramp_set_parameters(char *rampnode)
Function: Querying the current metering plan of a specific ramp meter.
Return Value: The current metering control plan of an on-ramp signal.
Parameters: rampnode is the name of an on-ramp signal node.

RAMP is the structure of ramp control data, whose definition is:

```c
typedef struct {
    // on-ramp signal node name and its location
double *rampNode;
    char *controllerLocation;
    // ramp control types and parameters
    int controlType;
    int meteringCycle;
};
```

Where controlType is the status (or type) of the ramp metering control, which can be 0 (if RAMP_CLOSURE), 1 (if RAMP_ON with single-entry metering), 2 (if RAMP_ON with platoon metering) and 9 (if RAMP_OFF).

void ramp_set_parameters(RAMP *ramp, Bool status)
Function: Setting a new metering rate to a specific ramp meter.
Return Value: None
Parameters: ramp stores the new metering control data of a specific on-ramp; status is a Boolean value. status = TRUE means to set a new metering rate based on an external algorithm; status = FALSE means to restore the default time-of-day timing plans.

3.3.3 Variable Message Sign Control
This API module interacts ATMIS applications with VMS signs. Through interface functions provided by this API, various ATMIS applications can dynamically update VMS information and then affect driver’s behavior timely.

3.4 Data collection and broadcasting

3.4.1 Loop detector data

In the real world, loop detectors are placed on freeways and arterials for collecting aggregated data at a certain time interval (typically, 30 seconds) for the purposes of traffic analysis and traffic control. These aggregated loop data are stored either in the database or shared memory. Other traffic operation components can get access to these data through data communication networks and use them for generating real-time control strategies.

The loop data aggregator API works as the traffic data collection and provision server in the PARAMICS-E environment. It emulates the real-world data collection from inductive loop detectors and broadcasts the latest aggregated loop data to the dynamic memory during simulation. Other API modules can obtain these data in real time through the interface function provided by this API. In addition, this API can report the aggregated loop data to text files or the MYSQL database as performance measures for data analysis and performance comparison.

The interface function of this API can be used for querying the aggregated loop data at a detector station at a certain time interval. The aggregated loop data includes grouped volume, average occupancy and average speed, as well as lane-based volume, average occupancy and average speed.

LOOPAGG  loop_agg (char *detectorName)
Return Value: The aggregated detector data of a loop detector
Parameters: detectorName: loop detector name

LOOPAGG is a structure that has the following definition:

```c
#define LOOPAGG
{
    int detectorIndex;
    float AggregationTime;
    int lane;
    int g_vol;
    float g_occ;
    float g_spd;
    int *vol;
    float *occ;
    float *spd;
};
```

where
detectorIndex is the network-wide index for the detector;
Aggregationtime is the time of the latest aggregation, determined by the loop data
collection interval;
g_vol is the total traffic counts passing all lanes of a detector station;
g_occ is the average occupancy of all lanes at a detector station;
g_spd is the average speed of all vehicles passing a detector station;
lane is the total number of lanes at the detector station;
*vol, *occ, *spd are pointers for recording values of volume, occupancy and average
speed at each lane of a detector station.

3.4.2 Point-to-point travel time data collection

Similarly, we develop a probe vehicle API to simulate point-to-point travel time data
collection through GPS equipped vehicles. The sample rate, which is equivalent to the
percentage of GPS equipped vehicles, can be specified through control interface. Point-to-point travel time data can be output to the dynamic memory and text files or the
MySQL database.

3.3 Path-based routing

PARAMICS is a link-based simulator and vehicles do not carry their whole routes but
deck their route based on the routing table stored at each node along its route. The path-based routing API establishes the mechanism for vehicles to follow a given path, which is
an essential requirement for the simulation of driver responses to the information supply
and the resulting route choice. Only those vehicles that need to follow specific paths will
be guided by this API and other vehicles will select route based on the internal routing
method of PARAMICS. The interface function used to set a path to a vehicle to follow is
as follows:

void uci_vehicle_route_set (void *Vp, VROUTE route)
Parameters:   Vp: the pointer of a vehicle, equivalent to vehicle ID
route is the initial address of the whole path
VROUTE is a link list storing whole path the vehicle should follow, which
is defined as:

```c
struct VROUTE {
   char *linkName;  
   VROUTE *next; 
};
```

3.4 Database connection

MySQL database is widely regarded as a highly efficient database. It can be used for
storing intermediate or final results during the simulation process if large amounts of data
are generated.
3.5 Performance measures

PARAMICS has strong abilities on the collection of statistics data. Except some general performance data, PARAMICS can output link-based, trip-based, intersection-based, and detector-based data. With the increase of the size of the network, the number of links increases drastically in PARAMICS. The current difficulties are

- Large amount of data are required to be processed after simulation runs in order to obtain the expected performance measures.
- Some performance measures cannot be extracted from output measurement data
- PARAMICS has a restriction on the number of output files to be opened during simulation under WINDOWS version.

To be more efficient to simulation studies, we developed a MOE API for computing, gathering and reporting a number of user-preferred overall performance measures, each of which corresponds to a specific aspect of interest.

3.5.1 Overall performance

MOE #1 system efficiency measure: average system travel time (ASTT) of the whole simulation period. ASTT is calculated as the weighted mean of the average travel times of all OD pairs

\[
ASTT = \frac{\sum_{o,i} (T_{o,i} \cdot N_{o,i})}{\sum_{o,i} N_{o,i}}
\]

(1)

where \( N_{o,i} \) is the total number of vehicles that actually traveled from origin \( o \) to destination \( i \); \( T_{o,i} \) is the average OD travel time from origin \( o \) to destination \( i \).

MOE #2 system reliability measure: average standard deviation of OD travel times (Sdev_TT) of the entire simulation period. Sdev_TT is calculated as the weighted standard deviation of the average travel times of all OD pairs for the whole study period:

\[
Sdev_{-TT} = \frac{\sum_{o,i} (Sd(T_{o,i}) \cdot N_{o,i})}{\sum_{o,i} N_{o,i}}
\]

(7)

where \( Sd(T_{o,i}) \) is the standard deviation of the average OD travel time from origin \( o \) to destination \( i \).

3.5.2 Freeway Performance

MOE #3 freeway efficiency measure

1. average mainline travel speed of the entire simulation period (AMTS)
2. average mainline travel speed during the congestion period (peak_AMTS). The congestion period is defined as the congestion period of the baseline scenario.

MOE #4 on-ramp efficiency measure

1. total on-ramp delay (TOD)
2. time percentage of the on-ramp queue spillback to the local streets (POQS)
3.5.3 Arterial Performance

MOE #5 arterial efficiency measure:
(1) average travel time from the upstream end to the downstream end of an arterial (ATT)
(2) the standard deviation of ATT (std_ATT)

In summary, PARAMICS-E complements the functionalities of the commercial PARAMICS model in many aspects. In addition,

4. API DEVELOPMENT OF ADVANCED ATMS MODULES

PARAMICS-E provides users a hierarchical API development framework, which can facilitate the development of advanced ATMS strategies. The basic control modules of PARAMICS-E include actuated signals, time-based ramp metering, VMS control, and path-based routing. The data collection and broadcasting modules are loop data aggregator and probe vehicle API. All these API modules provide interface functions in order to make themselves re-usable in the dynamic linking PARAMCIS-E environment. An advanced ATMS module, such as a signal coordination strategy, adaptive ramp metering algorithm, or a corridor control strategy, generally involves the control of one or several control components, including signals, ramps, VMS, and routing vehicles, based on real-time traffic information.

![Diagram](image)

Figure 2 Working mechanism of advanced ramp metering API

As an example, Figure 2 describes the working mechanism of advanced ramp metering API. The advanced API is built on top of two basic plug-in modules, i.e., ramp metering controller and loop data aggregator. The entrance ramp signals in the simulation network are controlled by the ramp metering API, through which metering rates can be queried and set by other API modules. The loop data aggregator API emulates the real-world loop data collection, typically with a thirty-second interval, and broadcast the latest loop data to the dynamic memory. The advanced API can access the dynamic memory and obtains the required loop data through interface functions provided by the loop data aggregation API. Then the metering rate for the next control interval is calculated based on advanced
ramp metering algorithm. The new metering rate is sent back to the ramp controller API for implementation (8).

5. CONCLUDING REMARKS

This paper presents our practices on developing a capability-enhanced PARAMICS simulation environment through integrating some plug-in modules implemented in PARAMICS API. These developed APIs complement the functionalities of the commercial PARAMICS model and enable the mechanism of the real-time traffic data collection, processing, and communication, as well as control of operations through dynamic linking. This enhanced PARAMICS simulation can thus have capabilities to model not only the target traffic conditions and operations but also various potential ATMIS strategies.

Currently, PARAMICS PROGRAMMER users can access to most parts of the micro-simulation. However, some other components of the simulation are still black boxes for users, such as the routing table and the merge model. The ideal situation is that users can get a full access to the micro-simulation through the interface functions in the API library. The commercial models can thus become used as a shell and users can activate the simulation based on their own models.

REFERENCES