Conceptual Design for a Research Oriented Web-based Traffic Simulation Platform

Xuan Shi (Corresponding Author)
Graduate Student
Department of Civil and Environmental Engineering
University of Wisconsin – Madison
(971) 344-0613
shi24@wisc.edu

Jing Jin, Ph.D.
Postdoctoral Fellow
Center for Transportation Research (CTR)
Department of Civil, Architectural, and Environmental Engineering
University of Texas at Austin,
Austin, Texas 78701
(512) 232-3124
jjin@austin.utexas.edu

Yang Cheng, Ph.D.
Research Associate
Department of Civil and Environmental Engineering
University of Wisconsin – Madison
Cheng8@ wisc.edu

Steven T. Parker, Ph.D.
IT Program Manager
Traffic Operations and Safety (TOPS) Laboratory
Department of Civil and Environmental Engineering
University of Wisconsin – Madison
sparker@engr.wisc.edu

Jian Zhang, Ph.D.
School of Transportation
Southeast University
No.2 Si Pai Lou, Nanjing 210096, China
101011599@seu.edu.cn

Bin Ran, Ph.D.
Professor
School of Transportation
Southeast University
No.2 Si Pai Lou, Nanjing 210096, China
and
Civil and Environmental Engineering
University of Wisconsin – Madison
Madison, WI 53706, USA
bran@wisc.edu

Word Count	
Abstract:	183
Main text:	5247
Figures (7*250)	1750
Tables (1*250)	250
Total:	7430

ABSTRACT

Researchers rely on microscopic traffic simulation for highly detailed analysis. However, existing software packages either provide very less flexibility in modeling or require very extra proficiency in programming so as to evaluate a new technology or benchmark a novel behavior model. Therefore, a significant part of efforts in utilizing simulation tools is consumed on non-research related work. In this paper, we propose a new simulation software platform with more freedom in injecting customized models, friendlier graphic user interface, and automation in trivial but time-consuming work. To achieve those goals, several state-of-the-art software engineering structural design patterns are adopted. The critical part of the proposed platform is to separate the simulation engine and user interface on the two ends of the web. The core simulation is centralized at the server, and different research tasks, which are undertaken simultaneously, are distributed on the clients with Internet browsers. This paper presents the conceptual design of the software platform with illustration of the software engineering concepts underneath. A demonstration, employing the browser techniques to animate the traffic on an online map, is shown to verify the advantages.

INTRODUCTION

Microscopic traffic simulation explicitly gives the representation of all the desirable elements in traffic dynamics *(1)*. Its beauty is that it mimics the behavior of entities in the system and their interactions with confident analytical forms. With the addition of proper animation techniques and data processing tools, microscopic simulation delivers an ideal presentation of traffic system graphically and statistically *(1)*, which is favorable by policy makers *(2)*. Yet the existing models are far away from reproducing the real circumstances. Tons of efforts are being made in the field of car-following models, lane-change models, route decision choices, etc. by researchers all over world, so as to improve the accuracy as well as the efficiency of microscopic simulation.

The option for researchers to conduct such kind of research usually lies in three categories: 1) commercial software, 2) open-source package, and 3) coding from the ground up. The strength of commercial software packages is usually found at the presentation of simulation results. Their attractive 3D animation of traffic and various charts of statistics lock the choice of private companies and government agencies. On top of that, those packages usually have been under continuous development for over a decade; the practicalness and stability of their frameworks and models have grown mature enough for a wide range of applications *(3, 4, 5, 6, 7, 8, 9)*. As a way of customization, those software packages often provide API (application programming interface) for user to explore the driver behaviors or roadway elements somehow. However, despite the fact that those packages require a significant amount of time in training respectively, the major drawback, from the perspective of researchers, lies in the opaque of the internal models and the interactions between them. Those black-box-style simulations make the evaluation of driver behavior models questionable, and decrease the creditability of the interpretation of the simulation outcomes.

Open-source packages, sometimes even called toolboxes, came into existence to solve the black-box problem. Users actually build their simulation models by different functional modules in those tools. The high flexibility and extendibility let the users fully customize their models easily. In addition, due to the absolute transparency of the simulation entities and workflow, researchers can explain their findings with more certainty. Strong programming background is usually demanded in the proficient use of open-source packages though.

The de facto way of building a simulation model is to code from the ground for many researchers. In fact, there are standard procedures to follow *(1, 2, 10)*. However, its un-reusable codes developed by different researchers make the cooperation harder. Besides, the redundant work in modeling similar roadway networks, or coding the well-known baseline models in various ways could affect the comparison between the works of individual researchers, or reduce the effectiveness of successive researchers. And that's also why properly maintained open-source packages emerged in research institutes *(11, 12, 13, 14, 15, 16)*.

Nonetheless, there is a common characteristic among those options. That is that they are all standalone software. Standalone software comes with a thick client. That client usually needs an installation or configuration, steps of which could be overwhelming. More critically, the computing performance of such simulation depends on the machine it lives on, and the sharing of individuals' models and codes takes extra efforts.

Web-based simulation, on the other hand, can completely change the experience of research via simulation. There are a lot of merits. 1) It is easy to use. The client side is usually very light, and is familiar with beginners to navigate or to attempt any simulation tasks. 2) Collaboration is made simple, because the core simulation engine is centralized at the server, where all the communications and interactions take place. 3) Code reuse, wide availability, cross-platform compatibility, and any advantages come with web are inherited *(17)*. Web-based simulation didn't become popular until the outspread of the use of Internet and rich web applications. Since the technology to enable web-based simulation is maturing, there emerges huge research and commercial opportunities *(18, 19)*.

Hence, we argue that there is a need of a simulation platform that is based on web, and also serves the transportation engineering researchers particularly. This paper presents a conceptual

architecture of such simulation platform, which takes the advantage of most recent technologies and software engineering concepts, to the research community. The rest of the paper will summarize the desired features of traffic simulation from researchers first, and then propose the web-based software platform to address such needs.

ARCHITECTURE DESIGN CONSIDERATIONS

Data Preparation

It is estimated that 30 to 50% of the efforts are spent on the network building and data processing in typical traffic simulation tasks *(16)*. Plus the works on calibration, and validation, researchers have to spend considerable time on non-research related work.

One of the major sources resides in the complexity of network building. In practice, a typical way of building the roadway network is to sketch from an aerial view photo, which takes considerable efforts and time. Besides, the sketched roadway network cannot precisely model the true world. As a way to save such undue efforts, leads in automatic network import and GIS products integration have been taken by the major market players. To illustrate this point, Table 1 shows the comparison between the major microscopic and mesoscopic simulation software packages in such capabilities.

TABLE 1 Comparison of Existing Microscopic Simulation Software Packages

Package	Host	License	Open Source	Customization[1]	API Language	Network Import	GIS Support
VISSIM	PTV (Germany)	Proprietary	No	Yes	VC++, VB	ESRI[2, 3], SYNCHRO, XML	ArcView, MapInfo
CORSIM	FHWA (US)	Proprietary	No	No	N/A	No	No
Paramics[4]	Quadstone (UK)	Proprietary	No	Yes	C, C++, Java	ESRI	ESRI, MapInfo, Google Earth, Bing
AIMSUN[5]	TSS (Spain)	Proprietary	No	Yes	Python, C++	ESRI, OpenStreet, TransCAD, CONTRAM, SYNCHRO, CUBE, Paramics, VISSIM, VISUM, RoadXML	ArcView, MapInfo
DRACULA	ITS Uni. of Leeds (UK)	Proprietary	No	No	N/A	SATURN	No
DYNASMART	FHWA (US)	Proprietary	No	No	N/A	TransCAD, TP+, VISUM, ESRI	ESRI Products
Dynameq	INRO (Canada)	Proprietary	No	No	N/A	EMME, STAN, SYNCHRO, etc.	ESRI
TRANSIMS	FHWA (US)	NASA	Yes	Yes	C++	ESRI	ArcNet
MITSIM	MIT (US)	Artistic	Yes	No	N/A	No	No
DynaMIT[4]	FHWA, ORNL (US)	N/A	No	No	N/A	No	No
MATSIM	ETH Zurich (Switzerland), T.U. Berlin (Germany)	GPL	Yes	Yes	Java	OpenStreet, VISUM	Google Earth, Coordinates
SUMO	Inst. transport research, ZAIK (Germany)	GPL	Yes	Yes	C++	OpenStreet, ESRI, VISUM, VISSIM, OpenDrive, MATSIM, etc.	Coordinates

OpenTraffic[6]	Queensland Uni. Of Tech. (Australia), Uni. of Delft (Netherlands)	LGPL	Yes	Yes	Unknown	Unknown	Unknown

1. Customization indicates the ability to create user-defined traffic elements, or inject user-developed car-following model and lane-change model.
2. ESRI indicates ESRI shape files.
3. VISSIM's support of ESRI is done via VISUM.
4. Supports distributed computing.
5. Integrates microscopic, mesoscopic, and macroscopic simulation.
6. OpenTraffic is still under development.

Due to the popularity of ESRI products, the ESRI shapefile *(20)* is most supported. ESRI shapefile is a geographic vector, which spatially describes geometries with additional attributes to describe itself. Shapefile data format has an open specification, which can be manipulated via open-source libraries; while the data itself, although used by lots of transportation agencies, is not freely available. OSM (Open Street Map), on the other hand, is distributed as open content and open database *(21)*. That's why OSM is popular among open-source packages.

The data processing and input for simulation is another source of painful work for users. Due to the shortage of common standards of the network, there is no simple pattern to automate the data input, which results in the manual entry for the traffic volumes of each link and turn for most simulation software. As the roadway network gets larger, the amount of work increases significantly. So does the process of validation and calibration of the simulation model afterwards. Standardized traffic data input has shown promises in efficient researches, and also has been raised to get attention *(22)*.

The proposed platform aims to support as many network format as possible, but with a focus on the ESRI shapefile and OSM. Thanks to the nature of web-based simulation, automated data processing in the proposed platform mostly relies on SQL queries on databases.

Modeling

The commercial companies focus on a much different aspect of the simulation software than researchers do. Commercial software packages are typically thin in modeling but rich in presentation while researchers usually desire the opposite. Here we will discuss the need of modeling in simulation package from the point of view of a researcher.

Firstly, researchers care more about the effects from the manipulation of the simulation models rather than the outcomes alone. Especially, for the researchers in the field of traffic flow modeling, it would be best of their interest to benchmark different models proposed all over the world on a common stage. Commercial simulation packages either employ widely accepted behavior models or stick with their original models *(23, 24)*. There are APIs available in those software packages for users to extend the functionality whereas the actual extendibility isn't that satisfying. For some commercial software, the API on the substitution of the internal models require extra compilation on the source files for every single change of the customized model, which takes extra cost for researchers to propose a new model. On the other hand, the powerful open-source simulation tools do have a lot of flexibility in injecting models while they are not easily comprehensible by most researchers because of its high bar for programming knowledge. The proposed simulation platform should balance the flexibility and complexity. That is, a researcher can implement his own model in a language of his choice, and the platform takes care of the compiling and linking. Moreover, because of web, the source code is editable online, which means the researcher can adjust the parameters of the model easily, and see the results on the fly. Respectively, advanced user also has the choice to extend the simulation model from low level and construct his complex scenario.

Secondly, new technology comes into use every day. The update of the objects in simulation software to model that technology merely cannot keep pace with the development of technologies, or

even just cover all the existing elements in the traffic system. Before launching the test of those new devices in the field, researchers certainly want to explore its advantage and disadvantage via simulation; whereas, for example, there currently isn't a simple way to model the Bluetooth detection device, or vehicle communication devices, etc. The proposed simulation platform should provide a simple way for users to appropriately represent those new technologies and to simulate them.

Last but not least, a repository of remarkable traffic models and classical roadway networks highly favors the researchers in seeking the new emerging trends. The proposed software platform should make the sharing of individual work on the Internet, and gathers all the contribution together. Those resources will benefit both education and research in a great manner. Newcomers of traffic simulation don't have to search learning materials across different sites while researchers can easily find a baseline to test against.

Other Design Considerations

There are several other less important but attracting features. 1) Automation of the whole simulation model building process. This feature can free researchers from excessive time on trivial jobs. 2) Interface for impact models and evaluation models. As the economic impact models significantly differ from environmental impact models, an interface for users to build their own impact models provides means for all the modelers to integrate them into the simulation. 3) Distributed computing. Several commercial packages have already provided such features. Opportunely, web-based simulation naturally has good extendibility in supporting distributed computing. 4) As the server gets more powerful, it is possible to simulate the traffic in a large area. As a result, the integration of the analysis at mesoscopic and macroscopic level would be an add-on. 5) Lastly, there is an increasing attention on LRS (Linear Reference System) *(25)* in transportation agencies in United States. It would be best to give them support in advance.

CONCEPTUAL DESIGN

Most open-source traffic simulation packages have followed modular design principles *(16, 26, 27)*, which is great for concurrent software teamwork and future development. However, this useful programming technique cannot construct the modules into a structure. UrbanSim *(28)*, urban simulation software, employs a more sophisticated technique, implicit invocation *(29)*, to properly fit those modules. ITERTIS (30), a Europe traffic management platform with ongoing development, is using layered system *(31)* to combine traffic simulation, which will be based on SUMO, and wireless simulation. Although complicated suitable web-based architectures *(17)* such as HLA (High Level Architecture) *(32)*, CORBA (Common Object Request Broker Architecture) *(33)*, etc. are proven successful in web-based simulation, this paper doesn't design with that detail but applies several software design concepts. The rest of this section describes how those concepts fit into traffic simulation along with some illustrative figures.

MVC Concept

Our simulation software architecture is based on implicit invocation *(29, 31)*, a software engineering technique that allows individual components communicate with each other and explicitly invoke each other's routines. Instead of executing a straight procedure, implicit invocation breaks the whole procedure into a collection of procedures within functional components, which can be developed independently. The proposed simulation platform basically implements a system that connects those functional components interactively. And most functional components can be used, replaced, or adjusted at user's discretion.

MVC (Model-View-Controller) *(34)*, an event-driven design pattern as well as an application of implicit invocation, is employed in the realization of the software platform. As the name implies, MVC categorize the software components into three sets: 1) model, 2) view, and 3) controller, which represent information, display, and control, respectively. Its application in traffic simulation is discussed as follows.

Model is the central structure of the whole platform. It holds all the information from visible objects such as roadway networks, traffic control devices, vehicles, etc. to intangible mathematical models such as behavior models. Beyond those easily acknowledged models, any additional data within

the system are contained in models as well. The form of user input such as network editor, vehicle editor, behavior model editor, configuration, etc. are also treated as models, which provides user a way to define their input data structure so that those data processing codes are made re-usable.

View deals with all the graphical components. It requests data from models, transforms them into charts, graphs, plots, animations, web pages etc., and dumps them at display output. In addition to those classic 2D / 3D traffic animation, vehicle trajectory plot, etc., any form of representation of data could be developed as an individual component and shared by all.

Controller connects the associated models, views, and view-controller pairs among each other. It notifies state updates to all related components. For example, after every simulation time step, evaluation models and impact models are notified to do an update while animations are notified to generate an updated frame to display. Except those user defined models and view-controller pairs, the core interaction between the most basic models and views are one of the main objectives of the proposed simulation platform.

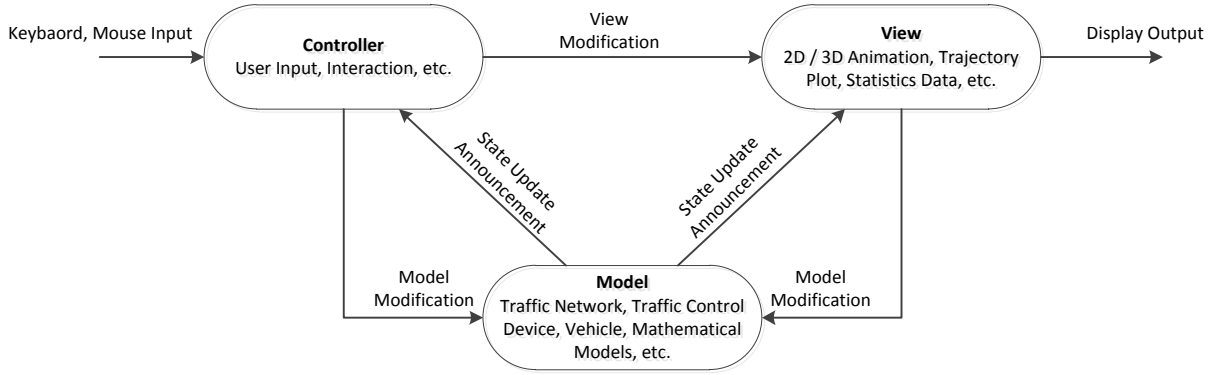

Figure 1 Illustration of Model-View-Controller concept.

In Figure 1, modification messages, generated by events, are sent to models to invoke series of component procedures. Upon successful completion, state update announcement messages are broadcasted to all associated controllers, views, and controller-view pairs, which will invoke another series of procedures to synchronize all the components with new information.

In other words, this kind of event-based software architecture is regarded as observer pattern *(34)*. Views and controllers observe the change of models. To be more specific, various analysis components are observing the change of the traffic simulation in the upper level, and the traffic simulation is observing the change of traffic condition and behaviors in the lower level. Observers are making self-adjustment subsequently if necessary.

Repository Concept

Having broken the whole simulation platform into relatively dependent components, this section will address the issues in user participation and contribution. Traditional modular programming gathers interchangeable components into modules *(35)*. And modules communicate with the main software through interfaces. The logical boundaries between components improve the maintainability due to separation of concerns *(36)*. MVC actually already incorporates the modular programming principles *(34)*. To take more advantage of Internet, repository *(31)* can make the modules between users more interactive.

Users participate in the standalone software development through contributing to the modules, but configuring others' customized modules into own system can be time consuming. To make those efforts truly contributable, a centralized repository, which contains and executes all the modules, can save the labor on searching and installing such modules.

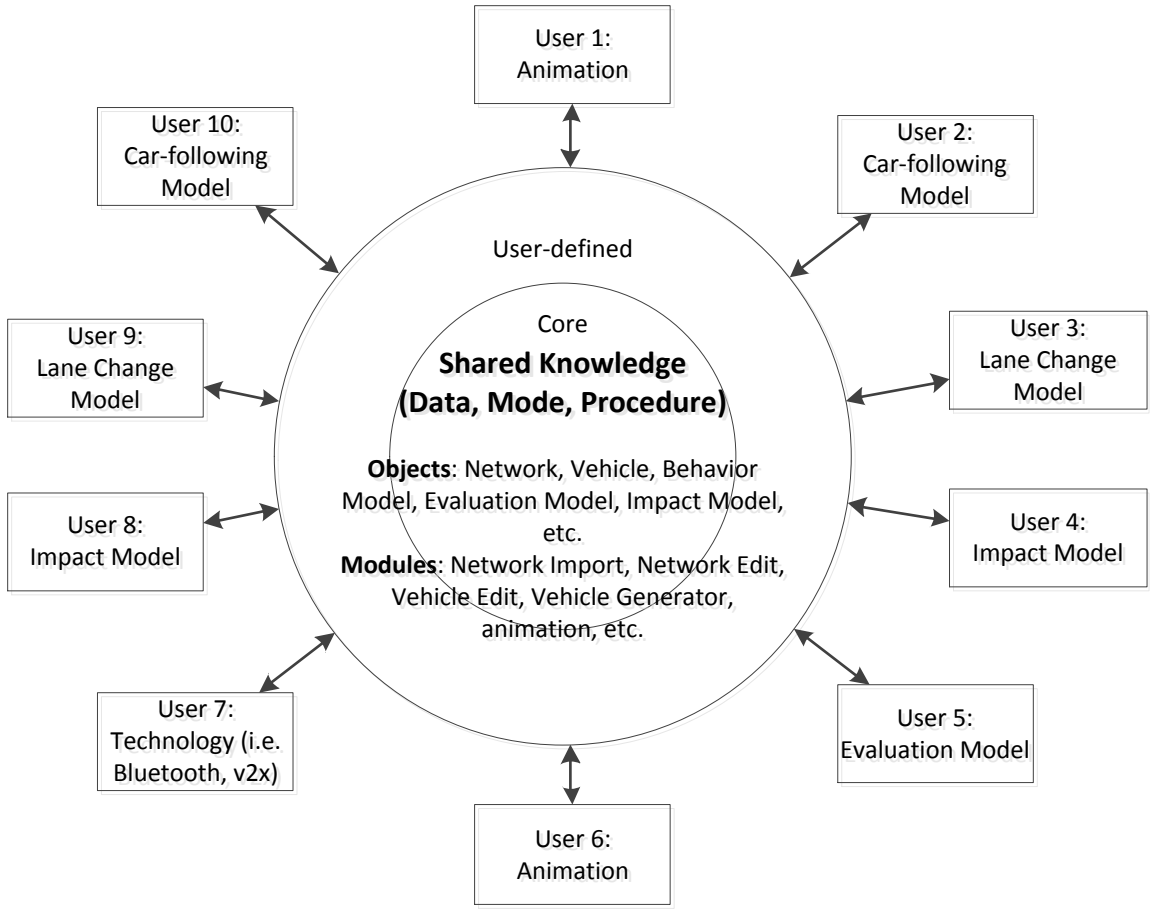

Figure 2 Illustration of repository concept.

As Figure 2 illustrates, there is a central repository holding all the shared knowledge, which includes data as well as components. In traffic simulation, the shared knowledge includes everything defined as models or views in MVC. Knowledge such as well-coded specific regions, types of vehicles, classic simulation models, utility tools, etc. are ready to be accessed and used by all users. Individual users can work on a certain piece of knowledge independently without affecting other users. For example, User2 and User10 can both develop a new car-following model tested by the traffic data in the shared knowledge. User7 can test a new type of technology for the simulation platform. As those works get mature, users can make them available to others by uploading them into the user-defined part of central knowledge repository. Notably, the repository can fulfill a role as a testbed to benchmark user-defined models as well.

Representation of controls between users and knowledge are the controller part in MVC concept. It provides an interface between knowledge and users. So another main task for the proposed simulation platform is to identify essential modules and clearly define their interfaces. Those works have already been somewhat studied by several open-source packages *(14, 15, 16)*.

Server-Client Concept

To realize such repository of shared knowledge, a server-client pattern is introduced. In Figure 2, the shared knowledge can also be regarded as a server, and the users can be regarded as clients. The server stores and maintains all the knowledge. Different clients communicate with the server to accomplish different tasks. Figure 3 gives the typical functions server and client handle respectively as well as its communication process.

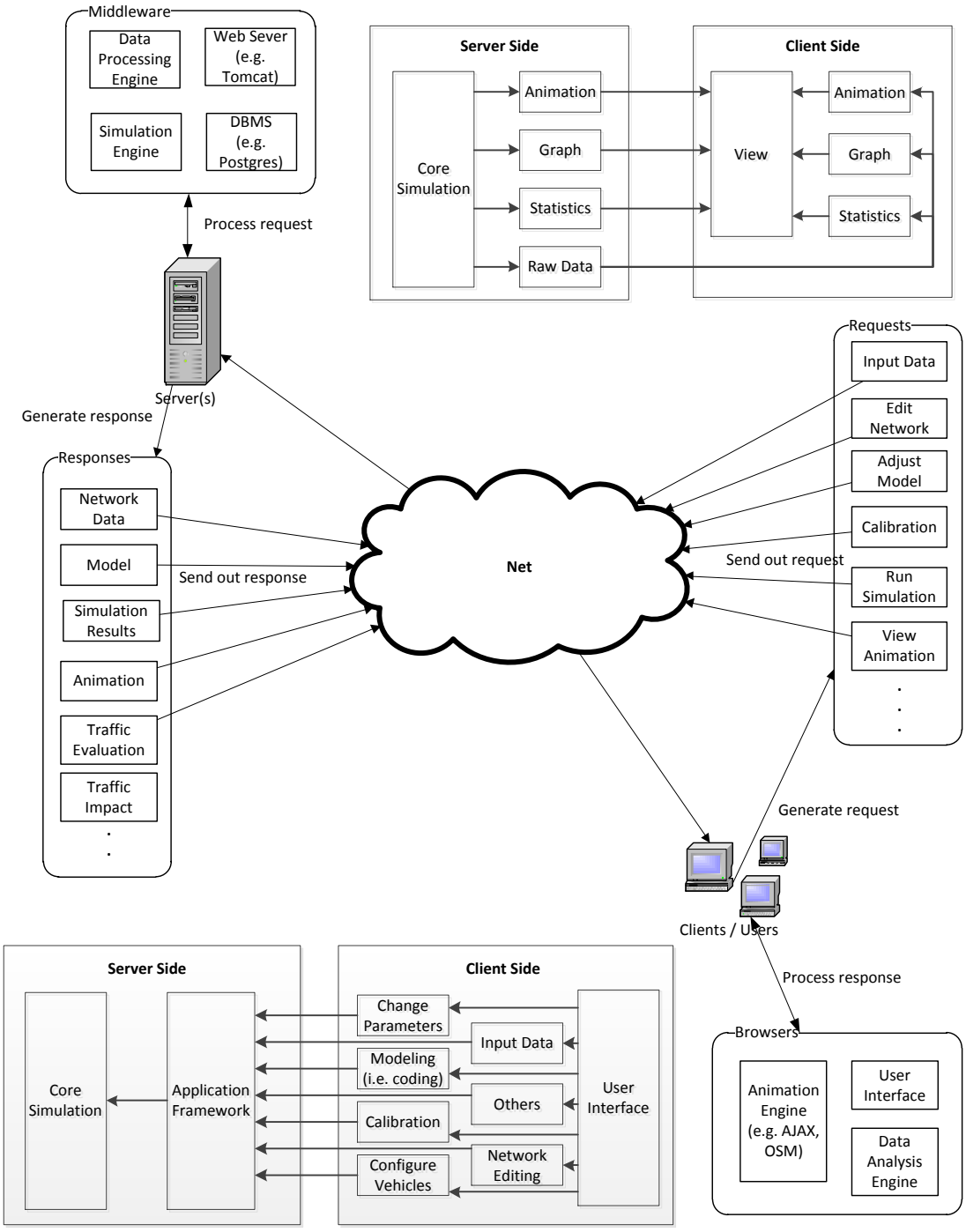

Figure 3 Illustration of communication between server and client.

The properly configured server needs to be running and listening over the net. And the clients initiate requests through the user interface defined by the proposed software platform. The user interface is basically interactive web pages created by the server. The only requirement for the client is, initially, a compatible Internet browser for those pages. The request, wrapped up with user input, is sent to the server through protocols. The web server translates the request and invokes proper procedures as required by the request. It can use the data processing engine to filter the data, talk to the DBMS (database management system) to update objects and models, or run the simulation with different settings. Web server then

generates a response that contains the results as requested. After receiving that, the client interprets the received response and displays the information. It can be animation of simulation, or analysis of results.

The request and response *(37)* pair is the messaging system between the server and the clients, which is really between the main simulation and the user interface. The available requests cover all the information user can input and edit. For example, user can send out request to import a new network, to edit an existing network, to add a new vehicle type, to create a new behavior model, etc. Those tasks could be done either through graphical interface with parameter editing textboxes or direct coding in available programming languages. And the response gives instant feedback and displays at the output. Besides, since the actual computing of simulation takes place at the server, the web interface literally provides a development environment that manages the software building.

Simulation Platform Conceptual Architecture Design

Based on the concepts discussed previously, Figure 4 gives an example of a typical simulation workflow of the platform. 1) Clients define the system by either feeding the information into the database or selecting existing data from the database. Those data includes supply data such as networks, demand data such routes, model data such as behavior models, and other data such as event information. 2) As simulation starts, those kinds of information are loaded into normalized formats for core simulation to read. 3) Core simulation runs iteratively over all the vehicles per time step. At each iteration, a frame is generated to record all the updated vehicle information, after applying traffic controls, restrictions, communications, and behavior models. Meanwhile, those updated information are also delivered to evaluation models and impact models to compute certain objectives. 4) Those frames are continuously fed back to the client as animation or pure statistics.

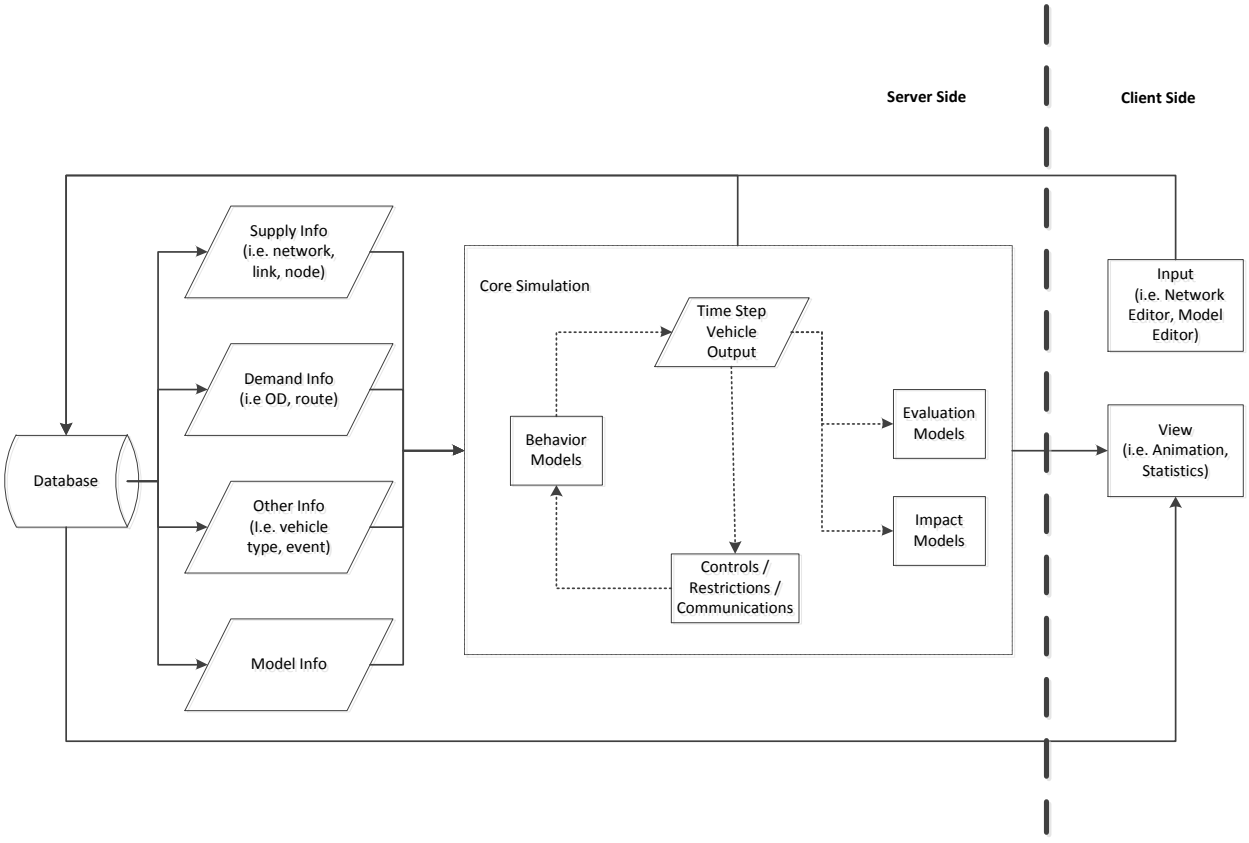

Figure 4 Typical simulation data workflow.

Figure 4 only shows the workflow for one run of simulation. As for those tasks that require multiple runs, the core simulation in the figure surrounded by the dotted line can be invoked as many times as wanted through external controller module. Those tasks, typically, after the definition and configuration of the system, include verification, validation, calibration, and any other automated procedures. The interaction between major components explains how the proposed platform makes the procedures more interactively communicate with the simulation, as shown in Figure 5.

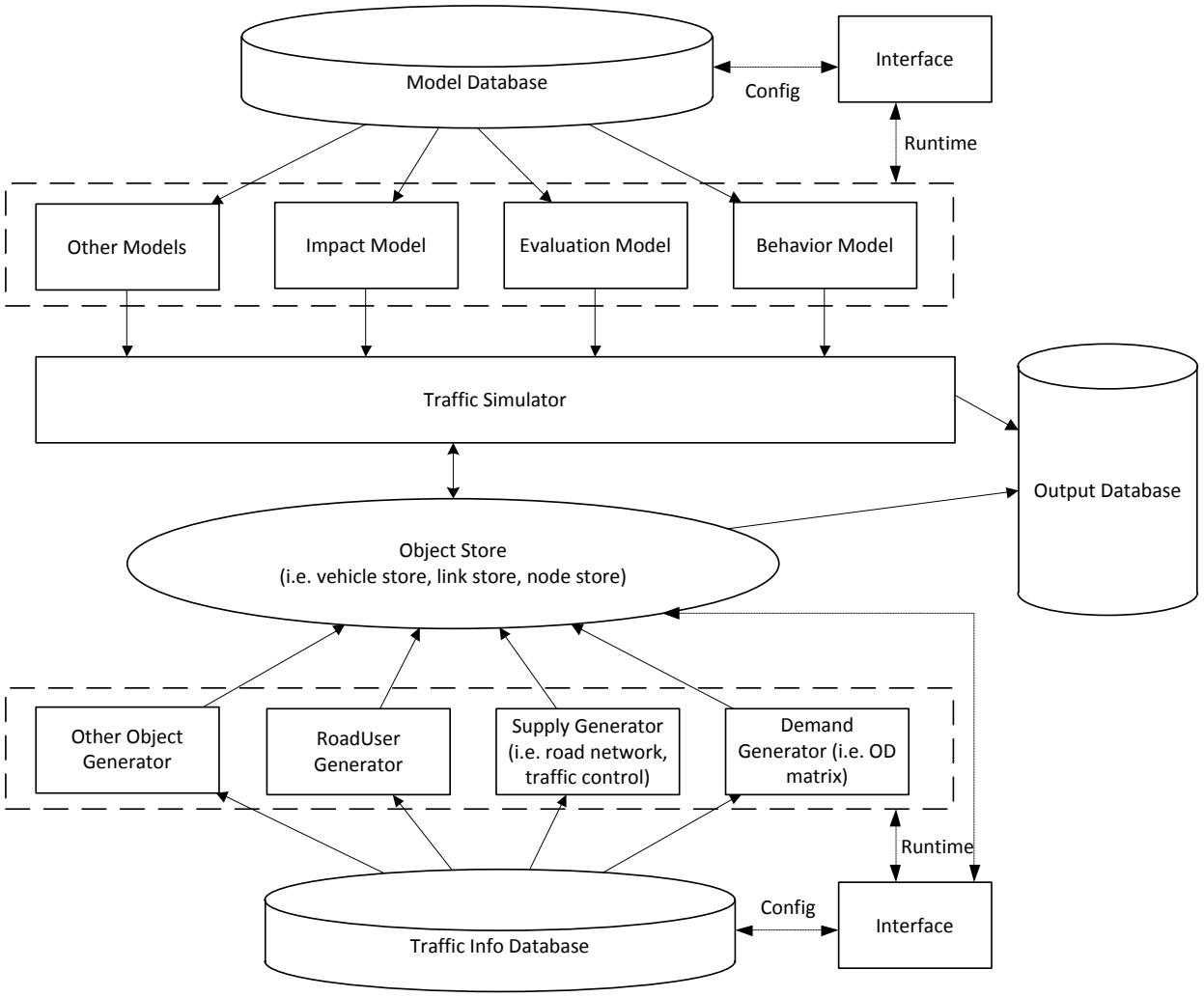

Figure 5 Server side simulation components.

The traffic data and mathematical model information are configured in different schemas of database through user interfaces at start. Before simulation, Object Generators load needed information to generate all the real world objects and put them at Object Store whereas model database feed parameters to all kinds of model components. Traffic Simulator essentially acts as a translation-aggregation layer to connect the objects and the models iteratively and dumps results at output. There are also runtime-programming interfaces on models and generators. Those interfaces provide access to the runtime status of models and objects. And that's the part enables automation. Furthermore, those rectangle components are easily substitutable. As mentioned in the repository concept, there are stores of those components available for user to utilize. As a result, it gives user a lot more flexibility on extension and customization.

The proposed platform employs the hybrid simulation and visualization technique *(38)*. As described in previous sections, the server side specializes in simulation, and the client side specializes in

graphical user input interfaces. That's the nature for using remote simulation with local visualization. However, the actual computing power distribution doesn't have to be like that. The engine to create animation, graphs, and statistics could be on both the server side and the client side, as Figure 3 shows. The client side receives pre-compiled animation, graph, and statistics as well as the raw data, which can be engaged with local visualization engine. One of the advantages is that the computing resources of both sides can be exhausted. From the point view of developers, the client side components are not only easier to develop, but simpler to deploy as well.

Last but not least, as a basic rule of object oriented programming, all the objects are extended from base object further into different abstract structures. Inheritance concept simplifies the maintenance of the software. The streamlined class diagram, in Figure 6, demonstrates the idea of categorizing object. Users can easily define a new type of road user by creating a subtype of RoadUser, for example, Pedestrian. Since this kind of extension normally requires the recompilation and redeployment, the proposed platform uses plug-ins *(39)* as support for the user customized components. Plug-in is a set of software components that add extra ability to the main application. Here, plug-in is used at the instantiation of the objects. For example, the vehicle class only holds its attributes but the calculation of those attributes is realized in the plug-ins. Those behavior models work in the same way, which makes the injection of user-defined objects and models seamless.

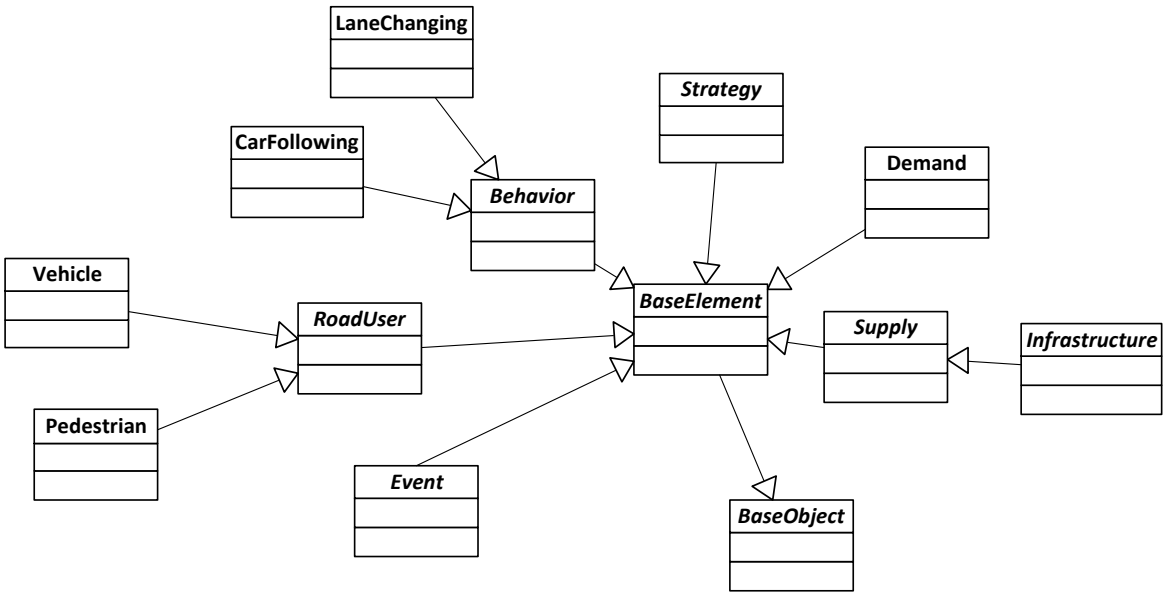

Figure 6 Streamlined class diagram.

ROADMAP AND DEMONSTRATION

Having the design considerations and the conceptual architecture in hand, this section describes the roadmap for the initial development of the proposed platform, and two demonstrations on client side.

The first remaining step would be to clearly define all the objects such as road networks, vehicles, behavior models, and etc., guided by the object oriented programming concept. Basically, this step is to complete the diagram in Figure 6 with all the classes plus their attributes and methods. It will require considerable analysis on the real traffic system.

The following step would be to identify all the external and internal modules of the simulation, and to clearly draw the boundaries between them and define their interfaces. This is the logic as well as the most critical part of the platform. Our key guideline is to keep the structure as open as possible, and every non-system module is a container open for user to customize.

The third step would be the actual implementation. It involves choosing proper tools for framework, database, front end user interface and animation, etc., which will be carefully examined and chosen according to the in-depth discussion in *(17)*.

Of course, after completion of those steps, it needs extensive usage to improve the stability, and popularity to fill the repositories with numerous data sets, models, and algorithms.

As a way to let the readers feel the experience that how such web-based simulation platform works on a web browser. A preliminary demonstration on simple web animation and user interface is given below. Figure 7 gives the snapshot of the animation. The raw data is obtained from NGSIM *(40)* as vehicle trajectories in terms of continual frames of coordinates. And the animation engine is an open-source GIS library, OpenLayers *(41)*, written in JavaScript. The engine, after transforming the vehicle coordinates into desired coordination system, draws the vehicle as dots, the color of which depends on its vehicle type, over the online map frame by frame.

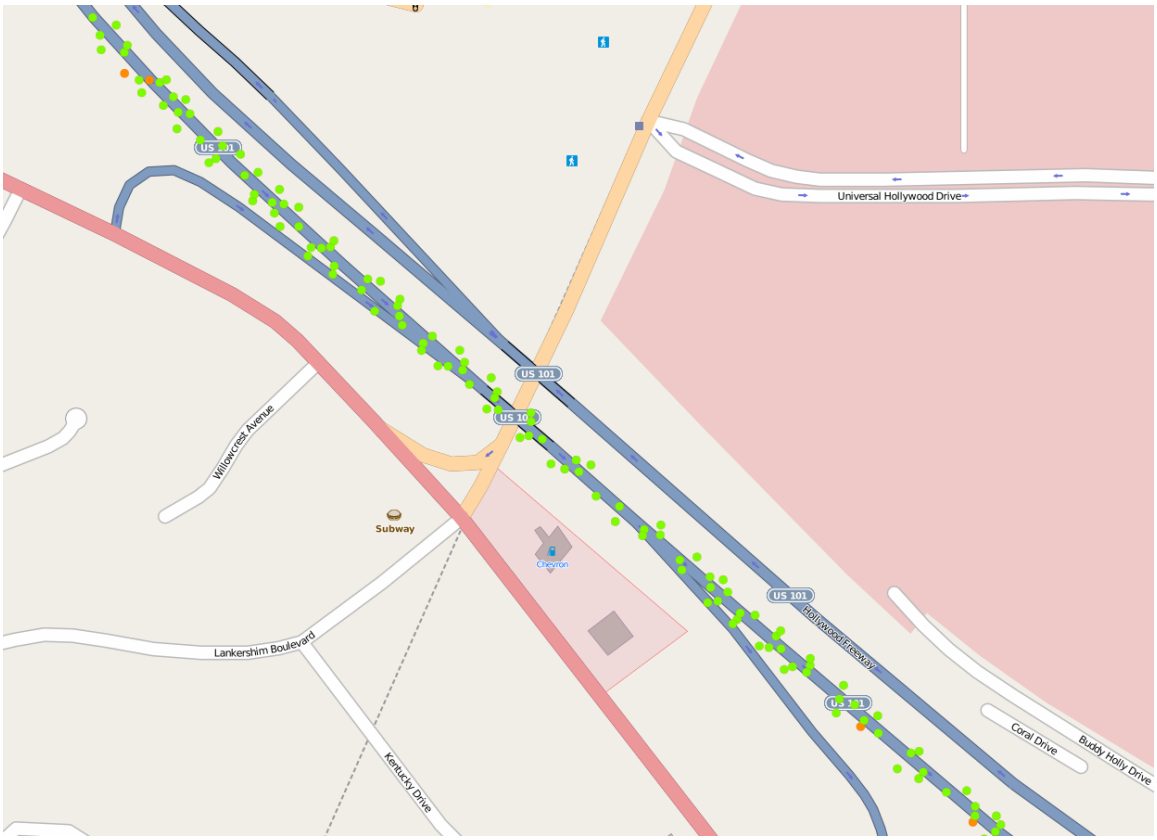

Figure 7 Snapshot of animation frame.

A web page of user interface where car-following models can be either configured or coded online are also created for demonstration purpose. As researcher finishes the programming, the model can be submitted and tested in the simulation on the fly.

CONCLUSION

This paper summarizes the needs as researchers in traffic simulation, and gives a conceptual architecture, which serves as the guideline for future development. The idea of such a web-based simulation platform comes from several failures in using commercial simulation packages to conduct certain researches in behavior models, and v2x communications, due to their limited extendibility and convenience. In order to make our valuable efforts on research effective, we argue that the existence of such a research-oriented platform is necessary. To take advantages of booming Internet in centralizing knowledge and distributing

works, we propose the maturing web-based simulation technology. There are certainly disadvantages of that. For example, it requires constant Internet connectivity, periodic maintenance on the heavy server, and so on *(17)*. However, we do believe that its advantage in ease of use, simple collaboration, and model and code reuse would outstand in simulation for research purposes. The platform is scheduled to be under realization soon. We hope this paper can draw the attention of researchers in the field of simulation, and some feedback on how you like this platform to be according to your desires and wishes so that we can adapt and proceed in the right direction.

ACKNOWLEDGEMENT

The paper was supported by the National High Technology Research and Development (863) Program of China (Grant No. 2011AA110405).

REFERENCES

1. LIEBERMAN, E., and A.K. RATHI. *Traffic Simulation*, in *Revised Monograph on Traffic Flow Theory*. FHWA, U.S. Department of Transportation, 2012.
2. Dowling, R., A. Skabardonis, and V. Alexiadis. *Traffic Analysis Toolbox Volume III: Guidelines for Applying Traffic Microsimulation Modeling Software*. Publication No. FHWA-HRT-04-040, FHWA, U.S. Department of Transportation, 2004.
3. VISSIM. http://www.ptvamerica.com/software/ptv-vision/vissim/. Accessed June 2012.
4. CORSIM. http://mctrans.ce.ufl.edu/featured/tsis/version5/corsim.htm. Accessed June 2012.
5. Paramics. http://www.paramics-online.com/. Accessed June 2012.
6. Aimsun. http://www.aimsun.com/. Accessed June 2012.
7. Dracula. http://www.its.leeds.ac.uk/software/dracula/. Accessed June 2012.
8. DynaSmart. http://mctrans.ce.ufl.edu/featured/dynasmart/. Accessed June 2012.
9. DynaMeq. http://www.inrosoftware.com/en/products/dynameq/index.php. Accessed June 2012.
10. Barceló, J. *Models, Traffic Models, Simulation, and Traffic Simulation*, in *Fundamentals of Traffic Simulation*, F.S. Hillier, Editor, Springer New York, Sepetember 2010, pp. 1-62.
11. TranSims. http://code.google.com/p/transims/. Accessed June 2012.
12. MitSim. http://mit.edu/its/mitsimlab.html. Accessed June 2012.
13. DynaMit. http://mit.edu/its/dynamit.html. Accessed June 2012.
14. MATSIM, Multi-Agent Transport Simulation Toolkit. http://www.matsim.org/. Accessed June 2012.
15. SUMO, Simulation of Urban Mobility. http://sumo.sourceforge.net/. Accessed June 2012.
16. Tamminga, G., M. Miska, E. Santos, H. van Lint, A. Nakasone, H. Prendinger, and S. Hoogendoorn. Design of an Open Source Traffic and Travel Simulation Framework. in 91st Annual Meeting of the Transportation Research Board, Washington, D.C., 2011.
17. Byrne, J., C. Heavey, P.J. Byrne. A review of Web-based simulation and supporting tools. *Simulation Modelling Practice and Theory*, Elsevier, Vol. 18, No. 3, March 2010, pp. 253–276.
18. Millera, J.A., P.A. Fishwickb, S.J.E. Taylorc, P. Benjamind, and B. Szymanskie. Research and commercial opportunities in Web-Based Simulation. *Simulation Practice and Theory*, Elsevier, Vol. 9, No. 1-2, October 2001, pp. 55-72.
19. Kuljisa J., and R.J. Paula. An appraisal of web-based simulation: whither we wander? *Simulation Practice and Theory*, Elsevier, Vol.9, No. 1-2, October 2001, pp. 37-54.
20. ESRI. ESRI Shapefile Technical Description. ESRI white paper, July 1998.
21. OpenStreetMap. http://wiki.openstreetmap.org/wiki/Main_Page. Accessed June 2012.
22. Barceló, J., and M. Kuwahara. *Traffic Data Collection and its Standardization*, Springer New York, June 2010.
23. Wiedemann R. Microscopic Traffic Simulation The Simulation System Mission. PhD Dissertation, University of Reiter, 1991.
24. Fritzsche, H.T. A Model for Traffic Simulation. Daimler-Benz AG, May 1994.

25. Noronha, V., and R.L. Church. Linear Referencing and Alternate Expressions of Location for Transportation. Vehicle Intelligence and Transportation Analysis Laboratory, National Center for Geographic Information and Analysis, Santa Barbara, California, 2002.
26. Balmer, M., M. Rieser, K. Meister, D. Charypar, N. Lefebvre, K. Nagel, and K.W. Axhausen. *MATSim-T: Architecture and Simulation Times*. in *Multi-Agent Systems for Traffic and Transportation Engineering*, A.L.C. Bazzan and F. Klügl, Editors, pp. 57–78, Information Science Reference, Hershey, 2009
27. Behrisch M., L. Bieker, J. Erdmann, and D. Krajzewicz. SUMO - Simulation of Urban MObility: An Overview. in *SIMUL 2011, The Third International Conference on Advances in System Simulation*, 2011, pp. 63-68.
28. Noth, M., A. Borning, and P. Waddell. An Extensible, Modular Architecture for Simulating Urban Development, Transportation, and Environmental Impacts. *Computers, Environment and Urban Systems*, Vol. 27 No. 2, March 2003, pp. 181-203.
29. Edwards B. An Introduction to Implicit Invocation Architectures. http://www.mach-ii.com/resources/intro_to_implicit_invocation.pdf. Accessed June 2012.
30. Rondinone, M. iTETRIS: The Integrated Wireless and Traffic Simulation Platform for Real-Time Road Traffic Management Solutions. *COMeSafety Newsletter*, No 8, January 2010.
31. Garlan, D., and M. Shaw. An Introduction to Software Architecture. CMU Software Engineering Institute Technical Report CMU/SEI-94-TR-21, ESC-TR-94-21, 1994.
32. Page, E., S. P. Griffen. Providing Conceptual Framework Support for Distributed Web-Based Simulation within the Higher Level Architecture. in Proceedings of the SPIE Conference on Enabling Technologies for Simulation Science II, Orlando, Florida, USA, 1998.
33. Henning, M. The Rise and Fall of CORBA. *Communications of the ACM - Designing games with a purpose*, Vol. 51, No. 8, August 2008, pp. 52-57.
34. Krasner, G.E., and S.T. Pope. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. *Journal of Object Oriented Programming*, Vol. 1, No. 3, 1988, pp. 26-49.
35. Gamma, E., R. Helm, R.E. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.
36. Dijkstra, E.W. *On the role of scientific thought*. In *Dijkstra, Edsger W. Selected writings on Computing: A Personal Perspective*. New York, NY, USA: Springer-Verlag New York, Inc. pp. 60–66.
37. Hohpe, G. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solution*. ISBN 0-321-20068-3, pp. 184.
38. Myers, D.S. An Extensible Component-Based Architecture for Web-Based Simulation Using Standards-Based Web Browsers. M.S. Thesis, Department of Computer Science, Virginia Polytechnic Institute and State University, 2004.
39. Birsan, D. On Plug-ins and Extensible Architectures. *Queue - Patching and Deployment*, Vol. 3 No. 2, March 2005, pp. 40-46.
40. NGSIM. http://ngsim-community.org/. Accessed June 2012.
41. OpenLayers. http://openlayers.org/. Accessed June 2012.