A Model Driven Approach to Web-based Traffic Simulation

Deniz Çetinkaya
Software Engineering Department
Atılım University, Ankara, Turkey
deniz.cetinkaya@atilim.edu.tr

ABSTRACT
As the world population increases the number of vehicles in the traffic increases as well, and so the traffic becomes more complex. Problems in the urban traffic such as traffic congestion, car accidents, parking difficulties, etc. have a large impact on people's lives as well as the environment. Therefore, researchers, policy makers, decision takers and planners use expert tools to find the best solutions for traffic and transportation problems.

Traffic modeling and simulation has been used for analyzing, designing, planning and managing urban traffic for many years. Various techniques have been proposed and many tools have been developed by researchers to assist the modeling and simulation activities in the traffic domain for more than half a century. However, improving the existing methods and developing new tools for traffic simulation are gaining importance due to the emerging technologies. Web-based modeling and simulation has been popular in the last decade, and has a great promise in terms of collaborative and distributed simulations. Model driven approaches are employed in the simulation field for a long time and have provided rapid development solutions. In this paper, a model driven web-based traffic simulation framework is proposed and a prototype implementation is presented.

Author Keywords
Model driven development; traffic simulation; Web-based simulation.

ACM Classification Keywords
I.6.5 [Simulation and Modeling] Model Development I.6.7 [Simulation and Modeling] Simulation Support Systems: D.2.13 [Software Engineering] Reusable Software

1. INTRODUCTION

As the traffic systems become more sophisticated and complex, professionals, researchers, policy makers, decision takers and planners use expert tools and efficient methods to propose better designs and find the best solutions for various problems. Some common problems in the urban traffic are traffic congestion, car accidents, parking difficulties, high levels of emissions, etc. Traffic modeling and simulation has been widely used for analyzing, designing, planning and managing urban traffic for many years. Recent studies focus on applying new technologies to improve the existing methods and tools [39, 17, 18].

Web-based modeling and simulation has gradually been popular in the last decade with the new technologies, and has a great promise in terms of collaborative and distributed simulations. Web-based simulation tools or environments provide interfaces in the form of Web-applications for controlling and executing simulations. These tools can distribute simulation tasks over a network for load balancing [39]. Web-based simulations can benefit the many features of the Web 2.0 including common standards, interoperability, ease of use, collaborative activities in a project, etc. [30, 5]. Besides, Web-based simulation is highly related with cloud computing and cloud-based simulation [8]. In Web-based simulations, formal descriptions and formal approaches become more essential.

In software engineering, there is a trend towards modernization for moving from desktop applications to new generation Web-based applications. Simulation field will require this change in the next a few years. Hence, the focus of this research is on the successful implementation of collaborative Web-based simulation environments by utilizing new technologies. The term Web-based application (or Web application) may refer to different types of software such as browser-based Web applications, rich client applications that uses standard Web protocols without a browser or non-native mobile applications.

Modeling and simulation is a discipline that requires software engineering activities for designing and developing simulation models, and for implementing their software solutions. In software engineering, model-driven development approaches provide specific advantages for reusability, productivity and maintainability by using well-defined models at various abstraction levels. Model driven approaches are employed in the simulation field for a long time. Many researchers have already realized the potential impact of applying model-driven approaches to increase software quality and productivity via automatic code generation [10, 19, 9, 34, 11]. In a model driven approach, from a software engineering perspective, models are the primary artifacts of the software development process and a software system is developed through successive model transformations. The final software system is (semi)automatically generated and is expected to be well-structured.

In this paper, a model driven Web-based traffic simulation framework is proposed and a prototype implementation is presented. This research contributes to the studies in the field of traffic simulation, Web-based simulation and modeling tool development by proposing a framework that combines the model driven development approach with reusable and executable components. Component based software development relies on having pre-developed and validated software components which can be used to form a hierarchical software system .To develop Web-based applications in an effective and convenient way model driven and component based approaches can provide us a sound basis.

2. RELATED WORK

Various techniques have been proposed and many tools have been developed by researchers to assist the modeling and simulation activities in the traffic domain for more than half a century. Trafc simulation is typically classied according to the represented level of detail as macroscopic, mesoscopic and microscopic traffic simulation [4]. While, the detailed movement of each individual vehicle is modeled at the microscopic level, a high level traffic flow modeling is generally described at macroscopic models. Mesoscopic models are at an intermediate level of detail [38]. Besides these three approaches, researchers sometimes use the term nanoscopic level to refer a more detailed level than microscopic level [28, 23]. For example, the vehicle and the driver are modeled as separate components at nanoscopic level.

Over the years, various methods are used in traffic simulation [36] including agent based systems [13, 1], queue models [20], cellular automata [14, 35, 33], DEVS [26, 35, 21] and Petri nets [32]. Besides, different commercial and freeware traffic simulation tools are available such as SUMO (Simulation of Urban Mobility) [25, 2], AIMSUN [7], TRANSIMS (TRansportation ANalysis SIMulation System) [31], PARAMICS [6], VISSIM [15], and MATSim (Multi-Agent Transport Simulation) [27]. More information can be found at [24].

Regarding the implementation of the simulation tools, improving the existing methods and developing new tools for traffic simulation are gaining importance due to the emerging technologies. Web-based modeling and simulation has gradually been popular in the last decade, and has a great promise in terms of collaborative and distributed simulations. Although, Web-based simulation is initiated via Java applet or Flash based animations [22, 37, 5], the research community has recently started to move toward utilizing the full functionality of the Web technologies.

Zehe *et al.* [39] propose an architecture for a cloud-based urban systems simulation platform which specifically aims at making large-scale simulations available on the Web. Bocciarelli *et al.* [3] propose a model-driven and cloud-based framework to support both the implementation of a distributed simulation system from a SysML (Systems Modeling Language) specification of the system under study and its execution over a public cloud infrastructure. Fortmann-Roe [17] presents a Web-based, general-purpose simulation and modeling tool. The tool, namely Insight Maker, integrates three general modeling approaches: system dynamics, agent-based modeling, and imperative programming in a unified modeling framework.

Among several Web-based simulation studies for urban traffic, the work done by Fortmann-Roe [17] is the most similar to our research. However, our research employs a model driven approach and focuses on reusability and component based simulation model development. Although, there have been some studies to apply model driven approach to traffic simulation [16], to the best of our knowledge, a complete model-driven solution for Web-based traffic simulations has not been proposed yet.

3. MODEL DRIVEN DEVELOPMENT OF TRAFFIC SIMULATION MODELS

Model driven development (MDD) is a software development approach that provides a set of means to develop a software system through successive model transformations [9]. The models are transformed into other models at different stages of software development lifecycle in order to (semi)automatically generate the final software system. The most important MDD methods are modeling, metamodeling and model transformations.

Modeling is the process of representing a source system for a specic purpose in a form that is ultimately useful for an interpreter [9]. The concrete form that represents the system is called the model. A model is specied in a modeling language. The MDD approach requires that the models and modeling languages are well dened. Metamodeling is the most commonly used method to describe a modeling language formally in the form of a metamodel, which in turn can be used to specify models in that language. Model transformation is the process of converting a model into another form according to a set of transformation rules. Model transformations are performed to utilize the knowledge in an existing model.

Applying MDD into traffic simulation requires domain specific modeling elements and metamodels. To perform an effective model-driven study, the following elements are needed [9]:

- a model-driven process definition,
- full featured metamodeling tool(s),
- platform independent metamodel(s),
- platform specific metamodel(s),
- pre-defined (or pre-developed) components,
- transformation rules.

While moving from desktop applications to web-based applications, model-driven development practice changes slightly. Existing design patterns such as Model-View-Controller (MVC) architectural pattern, Ubiquitous Web Application (UWA) design framework, or JavaServer Faces technology can be employed for the user-centered design of web-based applications [12].

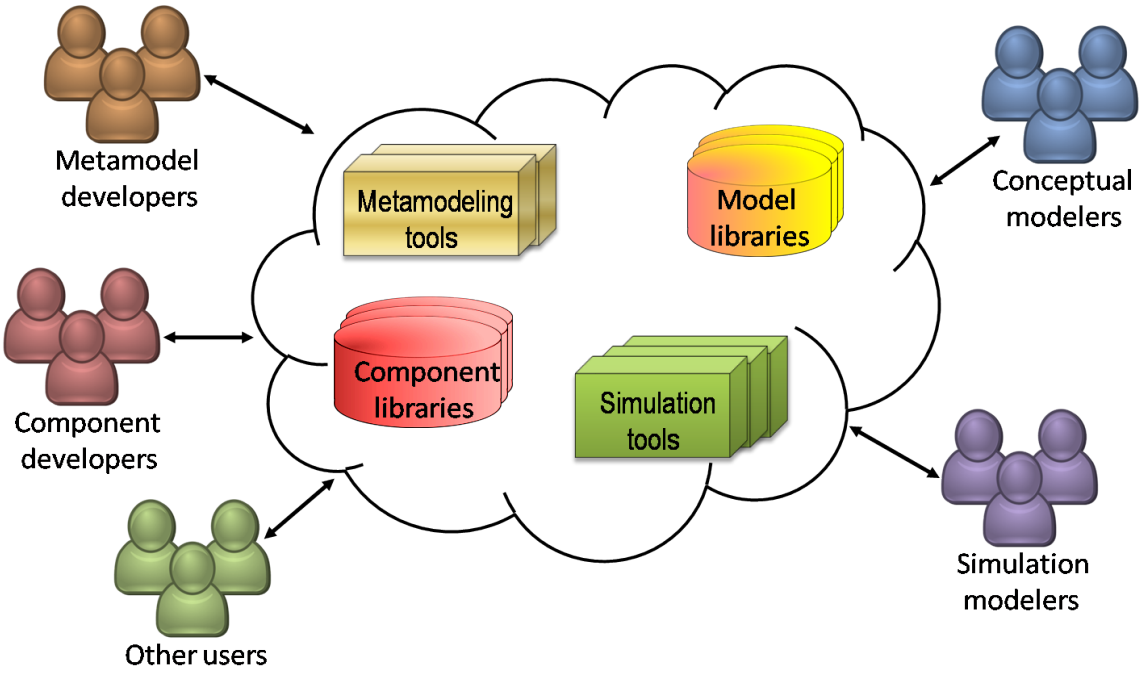

Figure 1. A general framework for model driven development of Web-based simulations.

Figure 1 presents a general framework for model driven development of Web-based simulations. Various repositories for model libraries and component libraries should exist on the Internet which can be provided by cloud services. Component developers, conceptual modelers, and metamodel developers provide items for these libraries. Metamodel developers use web-based metamodeling tools to define metamodels and to generate web-based simulation tools. Then, simulation modelers can use the simulation tools.

Simulation tools and libraries can be implemented as a Software as a Service (SaaS) implementation, while metamodeling tools can be implemented as a Platform as a Service (PaaS) implementation. Existing cloud computing solutions such as Microsoft Azure App Service, Google App Engine or Amazon Web Services (AWS) can be used. However, developers can have new challenges such as data condentiality and scalability when large-scale simulations are migrated to the cloud [39]. Developing a web-based application is often simplified by open source software and web application frameworks such as Django, Drupal, Bootstrap, Ruby on Rails or Symfony [29].

3.1 Web-based traffic simulation

Traffic simulation is an effective method for modeling the operations of dynamic traffic systems. A traffic simulation process generally consists of four main tasks:

- modeling the road network and the static environment,
- modeling the vehicles, drivers and other dynamic entities,
- modeling the controllers, signals and other rules,
- executing the overall model.

To accomplish these tasks, a metamodel for component based traffic simulation is proposed. Figure 2 shows the high level view of the metamodel. In the metamodel, `SimulationArea` is the main platform for the simulation animation. It requires several maps that shows the positions of the static components and needs initial parameters for simulation data.

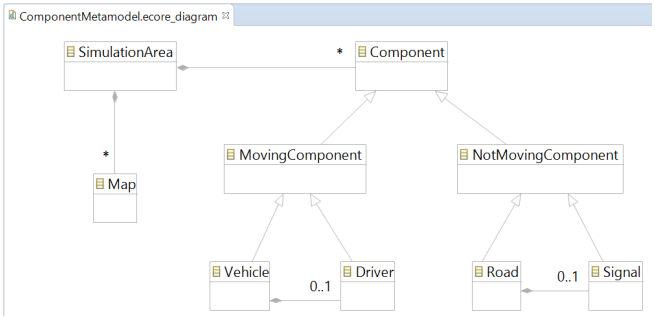

Figure 2. High level view of a metamodel for component based traffic simulation.

Simulation area can be grid-based (or tile-based) where the simulation space is divided into individual cells or nodes. Each cell is generally identical in size and shape. While a square-shaped cell is common in browser-based animations, it is also possible to use hexagonal or triangular cells as shown in Figure 3. Simulation area can have different implementations. `SimulationArea` includes different components, which is represented with `Component` in the metamodel. A component is a single unit which is self-contained and reusable. Components communicate with each other and with its environment via well-defined interfaces.

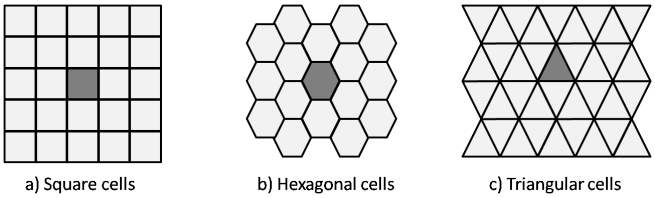

Figure 3. Different tile-based design alternatives.

A component can be either a `MovingComponent` or a `NotMovingComponent`. Moving components have a dynamic location according to their speed and moving direction while not-moving components have a static location. However, all components are changeable according to the time and not-moving components can also have event and time based changes. All of the components can use a shared clock, however each component should have an individual local time which is mostly represented by a discrete system.

For each component in a two-dimensional (2D) simulation area, the following data should be defined:

- position (x,y),
- size (width,height),
- component type,
- animation image.

Additionally, for moving components, speed and angle data are introduced. The movement can be managed by using the radian transformation. An angle's measurement in radians is numerically equal to the length of a corresponding arc of a unit circle. The relationship between degrees and radians are as follows:

$$1\ radian = \frac{\pi}{180^o}$$

Figure 4 represents how the angle and position data are related in a 2D area. If the angle of the moving component is θ, then the new position of the component is calculated according to the following formula after a unit time:

$$newx = x + speed * sin(\theta * \frac{\pi}{180^o})$$

$$newy = y - speed * cos(\theta * \frac{\pi}{180^o})$$

Each moving component can have a controller. If they don't have a controller, they are assumed to be autonomous. Controller components can update the speed and angle of the moving components. This relation is handled with composition relation. For example, in traffic simulation a vehicle can have a driver. If they don't have a driver they are assumed to be automated vehicle. A driver can give decisions according to his/her/its driving style and the driver component can update the speed and angle of a vehicle. For example, driver can

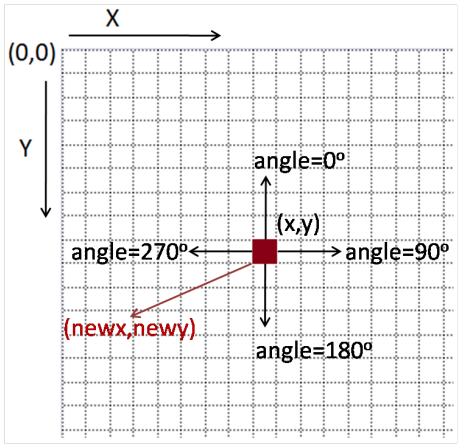

Figure 4. Calculating the new position.

decide to turn, stop, or change lane. Each moving component should define the following functions which are called by the controller.

- start,
- stop,
- changeAngle,
- accelerate,
- decelerate,
- setController,
- calcNextPosition,
- updateAnimationView.

Besides, each moving component should be able to reach the necessary data through its environment. For example, in traffic simulation a driver can reach the road information, traffic flow, traffic lights data, etc. for its environment. But, he/she/it cannot see the whole traffic directly. The definition of the boundaries for the environment depends on the grid type, and the neighborhood principles. For example, Figure 5 shows an example for a square-based grid. First, it is shown the von Neumann neighborhood consisting of the 4 neighbors. Second, the Moore neighborhood is shown that consists of the 8 neighbors. In both cases the range equals $r = 1$. However, depending on the context, the extended neighborhood where $r > 1$ can be used.

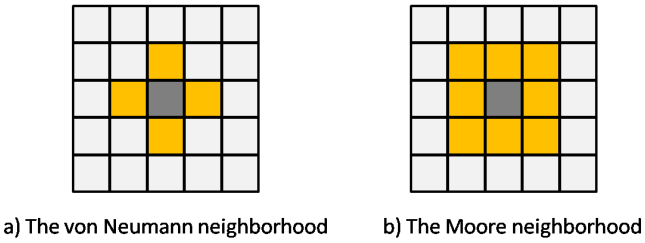

Figure 5. Neighbors of a cell can be defined in different ways.

Roads and signals are inherited from not moving component. Roads can have signals. New components can be added into the metamodel, each component will have a model definition as well as a software implementation. Once the metamodel is defined, the selected metamodeling tool will generate a web-based simulation environment for the defined metamodel. The auto-generated simulation environment should have an animation window, properties window, component window with drag-drop facility, and menu for available functionality. The modeling and simulation process can be summarized as in Figure 6.

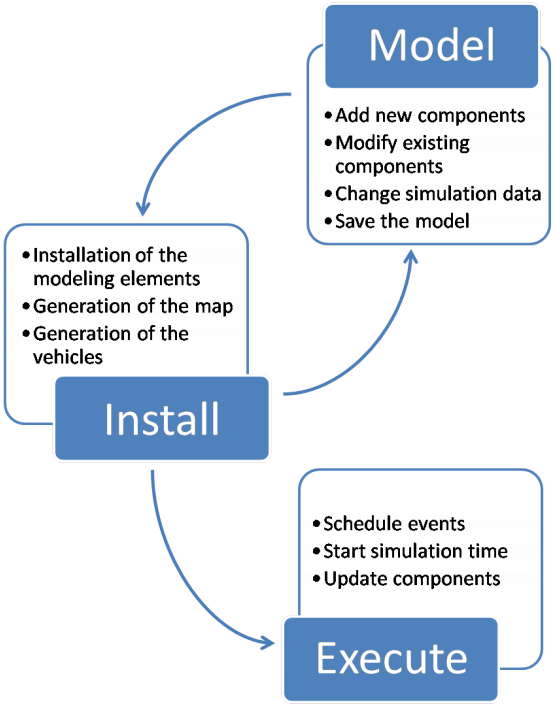

Figure 6. Steps for modeling and simulation.

4. A PROOF OF CONCEPT IMPLEMENTATION

In this work, a browser-based Web application for traffic simulation is developed. Some popular languages and technologies such as HTML5, Cascading Style Sheets (CSS), JavaScript and JavaScript Object Notation (JSON) are used for development. The simulation tool has a top menu for File, Edit, View, Run and Help functions. Left menu shows the draggable modeling elements. Properties window shows the basic properties for any component. The simulation tool has a simulation area which is an HTML5 canvas and mapped to a grid with a predefined cell-size. Every cell has a component or a sub-component.

Grid can be filled in with different images for each component. Figure 7 shows the tool layout with different images for each cell. It is also possible to put a map to the simulation area while keeping the component information for the grid. Figure 8 shows an example with a map.

To execute the models, simulation data, road network and signals are loaded first. Then, tool layout is generated and static (not moving) components are positioned. After that,

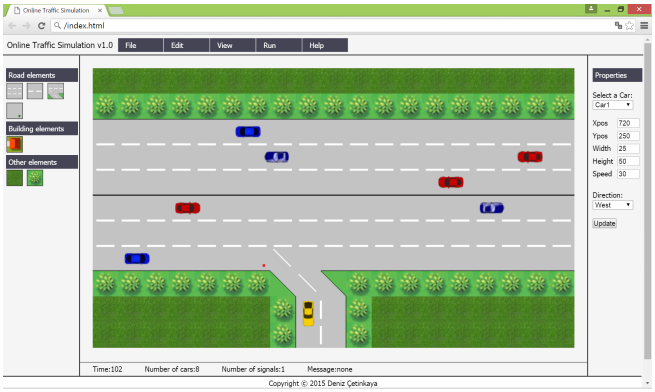

Figure 7. Tool layout with different images for each component.

Figure 8. Tool layout with a map in the simulation area.

the dynamic (moving) components are added into the simulation area. For example, the simulation area is initialized as follows:

```
// size in the world in sprite tiles
var simAreaWidth = 19;
var simAreaHeight = 11;
// width/height of a tile in pixels
var tileSize = 50;
loadMap(tileSize,
        tileSize*(simAreaWidth+1),
        tileSize*(simAreaHeight+1));
loadSignals(tileSize,
        tileSize*(simAreaWidth+1),
        tileSize*(simAreaHeight+1));
```

To implement the animation image requirement for a component, we use sprite sheets. Sprite sheets are commonly used in game development. A sprite is a 2D image or animation that is integrated into a larger image called a sprite sheet. Sprite number shows the current image. Sprite sheet is handled for each component as follows:

```
// an image containing all sprites
this.spritesheet = new Image();
//link to the image
this.spritesheet.src = imglink;
//spritesheet sprite number
this.spriteNum = sprite;
```

New position for moving objects is calculated as follows:

```
x+=speed*Math.sin(angle*Math.PI/180);
y-=speed*Math.cos(angle*Math.PI/180);
```

The following functions are defined as well:

```
this.turnLeft = function(m) {
  this.moveAngle -= m;
}
this.turnRight = function(m) {
  this.moveAngle += m;
}
this.accelerate = function(s) {
  this.speed += s;
}
this.decelerate = function(s) {
  this.speed -= s;
}
```

Figure 9 shows an example of a 20 degrees right turn. The car images are extracted from the screen-shots of the simulation and the turn function is visually validated.

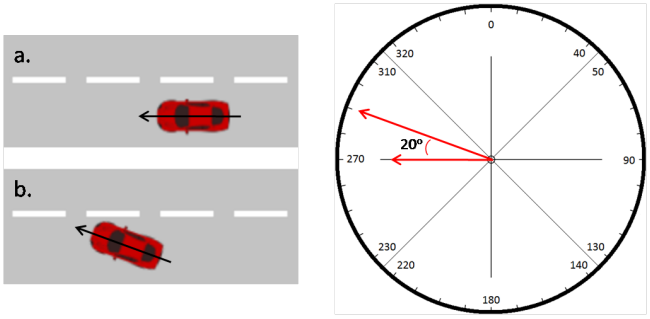

Figure 9. Validation of the turn function.

Signal component has red-time and green-time data to schedule the lightning. The following function used to manage signals:

```
this.changeLight = function() {
 if (signal==red AND timer <= 0) {
  signal=green;
  timer = this.greentime;
 }
 else if (signal==green AND timer <= 0){
  signal=red;
  timer = this.redtime;
 }
 else if (timer > 0){
  timer -= tickTime;
 }
}
```

Other elements are simply defined as a basic component such as tree or grass in the prototype implementation. Defining a new component template is easy with extending the component via inheritance. New data definitions and function declarations can be added in the metamodeling environment and required JavaScript code is automatically generated with the model transformer.

5. CONCLUSION

In this paper, methods and techniques of model driven development approach are applied to web-based traffic simulation. The future of simulation model development is not simply about using desktop applications, but it will include web-based simulation environments, cloud-computing, and Internet of things. Hence, this research will hopefully contribute to the future of modeling and simulation research.

Tool development and validation of the proposed framework are currently in progress. Collision detection for the moving objects is primitively handled for now. The following research problems and implementation issues are being studied:

- improving the collision detection algorithm for the moving components,
- improving the route following algorithm for the moving components,
- adding heuristics and stochastic behavior to the driver,
- improving the drag and drop functions,
- testing 3D support,
- adding the editing functions such as cut, copy, paste, undo, redo, zoom-in, zoom-out.

REFERENCES

1. Araujo, F., Valente, J., and Wenkstern, R. Z. Modeling agent-based traffic simulation properties in Alloy. In *Proceedings of the 2012 Symposium on Agent Directed Simulation*, Society for Computer Simulation International (2012), 5:1–5:8.
2. Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. Sumo–simulation of urban mobility. In *Proceedings of the 3rd International Conference on Advances in System Simulation* (2011).
3. Bocciarelli, P., D'Ambrogio, A., Giglio, A., and Gianni, D. A SaaS-based automated framework to build and execute distributed simulations from SysML models. In *Proceedings of the Winter Simulation Conference (WSC)* (2013), 1371–1382.
4. Burghout, W., Koutsopoulos, H., and Andreasson, I. Hybrid mesoscopic-microscopic traffic simulation. *Transportation Research Record*, 1934 (2005), 218–255.
5. Byrne, J., Heavey, C., and Byrne, P. A review of web-based simulation and supporting tools. *Simulation Modelling Practice and Theory 18*, 3 (2010), 253–276.
6. Cameron, G. D., and Duncan, G. I. PARAMICS-parallel microscopic simulation of road traffic. *The Journal of Supercomputing 10*, 1 (1996), 25–53.
7. Casas, J., Ferrer, J., Garcia, D., Perarnau, J., and Torday, A. Traffic simulation with Aimsun. In *Fundamentals of Traffic Simulation*, J. Barcel, Ed., vol. 145 of *International Series in Operations Research & Management Science*. Springer, 2010, 173–232.

8. Cayirci, E. Modeling and simulation as a cloud service: A survey. In *Proceedings of the Winter Simulation Conference (WSC)* (2013), 389–400.

9. Çetinkaya, D., Verbraeck, A., and Seck, M. D. Model continuity in discrete event simulation: A framework for model-driven development of simulation models. *ACM Trans. Model. Comput. Simul. 25*, 3 (2015), 17:1–17:24.

10. D'Ambrogio, A., Gianni, D., Risco-Martín, J. L., and Pieroni, A. A MDA-based approach for the development of DEVS/SOA simulations. In *Proceedings of the 2010 Spring Simulation Multiconference*, Society for Computer Simulation International (2010).

11. Denil, J., Mosterman, P. J., and Vangheluwe, H. Rule-based model transformation for, and in Simulink. In *Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative*, Society for Computer Simulation International (2014).

12. Distante, D., Pedone, P., Rossi, G., and Canfora, G. Model-driven development of web applications with UWA, MVC and JavaServer Faces. In *Web Engineering*, L. Baresi, P. Fraternali, and G.-J. Houben, Eds., vol. 4607 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, 457–472.

13. Dresner, K., and Stone, P. Multiagent traffic management: An improved intersection control mechanism. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, ACM (2005), 471–477.

14. Esser, J., and Schreckenberg, M. Microscopic simulation of urban traffic based on cellular automata. *International Journal of Modern Physics C 8*, 5 (1997), 1025–1036.

15. Fellendorf, M., and Vortisch, P. Microscopic traffic flow simulator VISSIM. In *Fundamentals of Traffic Simulation*, vol. 145 of *International Series in Operations Research & Management Science*. Springer, 2010, 63–93.

16. Fernández-Isabel, A., and Fuentes-Fernández, R. A model-driven engineering process for agent-based traffic simulations. *Proceedings of the 5th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)* (2015), 21–23.

17. Fortmann-Roe, S. Insight maker: A general-purpose tool for web-based modeling & simulation. *Simulation Modelling Practice and Theory 47* (2014), 28–45.

18. Garaizar, P., Vadillo, M., and Lopez-de Ipina, D. Benefits and pitfalls of using HTML5 APIs for online experiments and simulations. In *Proceedings of the 9th International Conference on Remote Engineering and Virtual Instrumentation*, IEEE (2012), 1–7.

19. Gianni, D., Bocciarelli, P., and D'Ambrogio, A. Model-driven performance prediction of HLA-based distributed simulation systems. In *Proceedings of the Winter Simulation Conference* (2012).

20. Grether, D., Neumann, A., and Nagel, K. Simulation of urban traffic control: A queue model approach. *Procedia Computer Science 10* (2012), 808 – 814.

21. Guin, P., and Syriani, E. Model-based animation of micro-traffic simulation (WIP). In *Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative M&S Symposium*, Society for Computer Simulation International (2013), 19:1–19:6.

22. Jacobs, P. H. M., Lang, N. A., and Verbraeck, A. Web-based simulation 1: D-SOL; a distributed java based discrete event simulation architecture. In *Proceedings of the 34th Conference on Winter Simulation: Exploring New Frontiers* (2002), 793–800.

23. Kokkinogenis, Z., Passos, L. S., Rossetti, R., and Gabriel, J. Towards the next-generation traffic simulation tools: a first evaluation. In *Proceedings of the 6th Iberian Conference on Information Systems and Technologies* (2011), 15–18.

24. Kotusevski, G., and Hawick, K. A review of traffic simulation software. *Research Letters in the Information and Mathematical Sciences 13* (2009), 35–54.

25. Krajzewicz, D., Hertkorn, G., Rössel, C., and Wagner, P. Sumo (simulation of urban mobility). In *Proceedings of the 4th Middle East Symposium on Simulation and Modelling* (2002), 183–187.

26. Lee, J. K., Lee, M. W., and Chi, S. D. DEVS/HLA-based modeling and simulation for intelligent transportation systems. *Simulation-Transactions of the Society for Modeling and Simulation International 79*, 8 (2003).

27. MATSim. Multi-agent transport simulation. http://www.matsim.org/. [Online; accessed 5-Jan-2015].

28. Ni, D. 2DSIM: a prototype of nanoscopic traffic simulation. In *Proceedings of the Intelligent Vehicles Symposium*, IEEE (2003), 47–52.

29. Norrie, M., Di Geronimo, L., Murolo, A., and Nebeling, M. The forgotten many? a survey of modern web development practices. In *Web Engineering*, S. Casteleyn, G. Rossi, and M. Winckler, Eds., vol. 8541 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, 290–307.

30. Onggo, S., and Hoare, S. Online collaborative simulation conceptual model development. In *Proceedings of the 23rd European Modeling and Simulation Symposium (EMSS)* (2011), 333–340.

31. Smith, L., Beckman, R., and Baggerly, K. TRANSIMS: transportation analysis and simulation system. Tech. rep., Los Alamos National Lab., NM (United States), 1995.

32. Tolba, C., Lefebvre, D., Thomas, P., and Moudni, A. E. Continuous and timed Petri nets for the macroscopic and microscopic traffic flow modelling. *Simulation Modelling Practice and Theory 13*, 5 (2005), 407–436.

33. Tonguz, O., Viriyasitavat, W., and Bai, F. Modeling urban traffic: A cellular automata approach. *IEEE Communications 47*, 5 (2009), 142–50.

34. Topçu, O., Adak, M., and Oğuztüzün, H. A metamodel for federation architectures. *ACM Trans. Model. Comput. Simul. 18*, 3 (2008), 10:1–10:29.

35. Wainer, G. ATLAS: A language to specify traffic models using Cell-DEVS. *Simulation Modelling Practice and Theory 14*, 3 (2006), 313–337.

36. Wainer, G. A. *Discrete-Event Modeling and Simulation: A Practitioner's Approach*. CRC Press, Inc., 2009.

37. Wang, Y.-H., and Liao, Y.-C. Implementation of a collaborative web-based simulation modeling environment. In *Proceedings of the 7th IEEE International Symposium on Distributed Simulation and Real-Time Applications* (2003), 150–157.

38. Xu, Y., Aydt, H., and Lees, M. SEMSim: a distributed architecture for multi-scale traffic simulation. In *Proceedings of the 26th Workshop on Principles of Advanced and Distributed Simulation (PADS)* (2012), 178–180.

39. Zehe, D., Knoll, A., Cai, W., and Aydt, H. SEMSim cloud service: Large-scale urban systems simulation in the cloud. *Simulation Modelling Practice and Theory 58* (2015), 157–171.

BIOGRAPHY

Deniz Çetinkaya is an Assistant Professor in Software Engineering Department at Atılım University, Ankara, Turkey. She graduated from Department of Computer Engineering at Hacettepe University, Ankara, Turkey with honors in 2002. She received her M.Sc. degree in Computer Engineering from Middle East Technical University, Ankara, Turkey in 2005. She received her Ph.D. degree in Systems Engineering from Delft University of Technology (Technische Universiteit Delft) in the Netherlands in 2013. Her research focuses on model driven development, component based software engineering, modeling and simulation, discrete event simulation and conceptual modeling.