Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

DEVELOPMENT AND APPLICATION OF AN INTERMODAL MASS TRANSIT SIMULATION WITH DETAILED TRAFFIC MODELING

Joseph C. Brill

Systemflow Simulations, Inc.
578 Parliament Street
Marietta, GA 30066, U.S.A.

Dudley E. Whitney

Parsons Brinckerhoff Quade & Douglas, Inc.
301 North Charles Street, Suite 200
Baltimore, MD 21201, U.S.A.

ABSTRACT

This paper summarizes the design and application of a discrete event simulation model and animation of an intermodal transportation facility. We discuss the study objectives, the software selection criteria, the model architecture, the model results and conclusions drawn from the simulation study. Distinguishing features of the model architecture and development using SLX and Proof Animation are presented in the context of the application. The model was used to investigate a number of layout, operational, and scheduling issues.

1 INTRODUCTION

The Frankford Transportation Center (FTC) simulation study combines bus, trackless trolley, automobile, train, and pedestrian movements in an integrated model of the proposed changes and additions to an intermodal facility in eastern Philadelphia.

Section 2 of this paper provides a description of the study with an overview of the system, the project goals and objectives, and the simulation requirements. Section 3 describes the software selection decisions framed in the context of the FTC application. Section 4 includes a discussion of the model inputs, architecture, and outputs. Section 5 provides general observations and specific recommendations concerning the operating plan of the FTC facility. Finally, Section 6 presents the conclusions concerning the facility design based on the simulation study.

2 STUDY DESCRIPTION

2.1 System Overview

The Bridge-Pratt Street Station, the eastern terminus of the Market-Frankford Subway Elevated (MFSE) line, currently serves 17,600 boardings per day. Transfers from bus and trackless trolley routes serving the adjacent Frankford Terminal facility represent 56 percent of the daily boardings for the MFSE. The Frankford Terminal serves 23,000 boardings per day, with 50 percent transferring from the MFSE. More than 500 buses and 180 trains operate out of the Frankford Terminal facility daily.

The proposed changes and additions to the joint facility are designed to integrate the bus and rail functions to improve the transfer between modes and to increase vehicle and passenger capacity. The changes to the facility include:

- moving the station and associated tracks for the MFSE closer to the Frankford Terminal,
- construction of a new, larger station and parking structure,
- construction of bus islands around the new station to facilitate transfers, and
- reconfiguration of the street network to improve circulation of automobiles and buses.

2.2 Project Objectives

The model was developed to test the facility design and assess the impact of current and future passenger demand on the system. The simulation model of the FTC facility gives the project team the capability to:

- analyze the quality of service for bus and train passengers as a function of schedule adherence,
- analyze the traffic patterns of buses inside the facility based on drop zone and berth location,
- assess the contention for drop zones and berths as a function of the bus schedule, physical layout, and dedicated assignment,
- evaluate the effect of surface street congestion on the bus schedule and bus activity inside the facility,
- determine the impact of bus traffic on surface street traffic, and
- analyze the effect of intersection signal phasing on surface street congestion.

2.3 Project Requirements

A partial list of the required features is included to illustrate the breadth of scope and level of detail of the model:

- routing for twenty-one bus services with shared passenger drop zones and dedicated berth locations (including bringing buses into service and taking buses out of service from the storage facility),
- terminal station operations of an elevated train system (including bringing trains into service and taking trains out of service at the yard leads),
- pedestrian movements including "choke points" (doors, fare gates, escalators, etc…),
- signalized intersection control (e.g., red, yellow, green, left turn arrow, right turn arrow) and synchronized signals,
- detailed vehicle modeling including acceleration, deceleration, and following distance,
- detailed driver behavior including left turn on green across oncoming traffic (no arrow), right turn on red, and stop signs.

3 SOFTWARE SELECTION

3.1 Software Considerations

The inclusion of vehicle acceleration, deceleration, and following distance significantly increases the complexity of vehicle modeling. The control of vehicles is based on both the network layout (vehicle-to-network) and the state of other vehicles (vehicle-to-vehicle).

In our experience, all of the simulation packages with built-in vehicle movement systems (e.g., AGVs) can be characterized as vehicle-to-network systems. Typically in these systems, vehicles are controlled by fixed points attached to the network layout. Vehicles proceed across the network by claiming and later releasing forward control points. The conceptually simple task of accumulating vehicles of different length is usually difficult to model.

In order to implement vehicle following, there must be a mechanism for communication between vehicles and the capability to directly control individual vehicles (e.g., accelerate, decelerate, cruise). The ability to modify a vehicle's behavior is required at an infinite number of arbitrary points on the network layout. For example, the stop position of an individual vehicle at a red light is predicated on the number and length of all preceding vehicles.

Based on our collective modeling experience with the built-in capabilities of vehicle movement systems in existing simulation packages, we were left with the following choice: attempt to build our application on top of a movement system in an existing simulation package that does not support our modeling needs directly versus building a model from scratch in a general purpose simulation language that fully supports our modeling needs. The team chose to develop the application from scratch in a general purpose simulation language rather than trying to force our application into the constructs of an existing tool.

3.2 SLX and Proof Animation

The simulation model of the FTC application was developed in SLX (Henriksen, 1996) and animated with Proof Animation (Henriksen, 1996), both products of Wolverine Software Corporation. In addition, a number of support scripts for input data processing were developed using Perl.

SLX is a layered software system for discrete-event simulation. The motivation for using SLX stems from the fact that the set of SLX kernel level simulation primitives is both small and powerful. Distinguishing features of SLX include:

- user defined objects and types,
- sets,
- a generalized wait until mechanism, and
- a statement definition capability.

These features will be discussed in the context of the FTC application, where appropriate, in the following sections.

Proof Animation was chosen as the animation package, in large part, based on its successful use in earlier large-scale rapid transit studies (see Atala, Brill, and Carson, 1992). Proof is a general purpose, post-processing animator designed for use with a wide variety of simulation tools and application areas. For the FTC application, we used Proof as both the animator on the "back-end" and as a graphical data entry tool on the "front-end". The use of Proof on the front-end is described in the Model Inputs section of this paper.

4 THE MODEL

4.1 Model Inputs

The model is driven by a number of external data files provided by the user. The input data files include:

- a bus schedule,
- a train schedule,
- automobile arrival rates and routing matrix,
- pedestrian arrival rates and routing matrix, and
- intersection signal phasing.

The input data files were created and maintained using spreadsheets in file formats developed by the team

members responsible for collecting the data. Perl scripts were written to translate the user input file formats to the simulation model file formats.

One of the preliminary design goals for the model was the ability to rapidly modify the automobile, train, and pedestrian networks. To support this goal, a pre-processing program was written in SLX that reads a Proof layout file and generates the necessary data file(s) for the FTC simulation model. This allowed the various networks to be defined graphically using Proof. Each system (automobile, train, and pedestrian) was developed and maintained in a separate Proof layout file. An additional Proof layout file contained the background graphics and all other necessary animation constructs. The "master" Proof animation layout file is generated on demand by merging the relevant portions of the four layout files together.

4.2 Model Architecture

For purposes of exposition, the project activity can be divided into three distinct components: design and implementation of the simulation engine, development of the FTC application, and engineering analysis based on model results. Verification and validation spanned all three phases of model development. The simulation engine and FTC application are addressed below; the operations analysis is discussed in the Results and Conclusions sections.

4.2.1 The Simulation Engine

In general, the internal architecture of the model is hierarchical. As we move up the hierarchy, we operate at increasing levels of abstraction. For example, when we route a bus through the facility, we do not want to be concerned with the mechanics of vehicle movement, but instead achieving the end goal of successfully arriving at our desired destination. We allow the implementation of lower levels to handle the necessary details of vehicle movement and interaction.

SLX provides both data and procedural abstraction mechanisms. Data abstraction is provided by the ability to define new data types and to build classes which are aggregations of both native SLX and user-defined data types. Procedural abstraction is provided by a statement definition capability. The statement definition capability allows the introduction of new statements into the language. Statement definitions are compiled by SLX, effectively extending the native SLX compiler. This capability can be further exploited to generate SLX data types and procedural code definitions at compile-time. In essence, a model can build its own data and code support during compilation.

The implementation of the simulation engine takes full advantage of the abstraction mechanisms of SLX. Higher level functionality "encapsulates" lower level implementation details. At the lowest level, the simulation engine contains a general network data structure and the relevant code support for construction and maintenance of an arbitrary number of network instances. In fact, all three movement systems (automobile, train, and pedestrian) are built on top of the same underlying network data structure. Similarly, the automobile and train systems are built upon the same vehicle movement system.

The vehicle movement system is comprised of all vehicle-to-network and vehicle-to-vehicle control mechanisms. Characteristics of an individual vehicle include: length, acceleration rate, deceleration rate, following time, and maximum allowed speed. Note that vehicle following is defined as a constant time. This yields a non-linear following distance that is a function of the vehicle characteristics of both the leader and follower. As a vehicle moves across the network, the vehicle following time is converted to a target following distance and the appropriate vehicle action is taken (accelerate, decelerate, maintain speed).

All vehicles are autonomous "drones" subject to the actions of surrounding vehicles and any network control (e.g., stop lights). A state change of a single vehicle may (or may not) trigger a state change in one or more surrounding vehicles. For example, as a stream of cars approaches an intersection at which the signal is red, the lead car begins to slow down. In turn, following cars will begin to slow down. Eventually, all of the cars will come to rest forming a line end-to-end.

The communication mechanism between vehicles was built primarily with pointers to SLX objects and sets. Set construction primitives include insertion and deletion. Iteration primitives for sets include the first, last, predecessor, and successor objects. Additionally, sets can be ranked on any number of object attributes. Sets provide a built-in mechanism for list construction and maintenance.

Each edge (or link) in the vehicle network has an associated SLX set. Each vehicle has an object associated with its head (leading edge), middle, and tail (trailing edge). As the vehicle traverses the network, the objects for the vehicle are continuously inserted into and removed from the relevant network edge sets. To find a following vehicle, the successor object to the tail of the vehicle in the associated network set is evaluated. When the successor is null, there is not a following vehicle on the current network link. If desired, the network can be "searched" backward looking for one or more vehicles based on the network connectivity. When a vehicle changes state, following vehicles are notified so that they

may act accordingly. Similarly, the network can be searched forward for preceding vehicles. A forward search is usually used to determine the distance between two or more vehicles.

The vehicle-to-network control is implemented through control points and fixed points. Control points and fixed points are physically attached to the network layout. Control points have an associated speed. When we want vehicles to stop at a control point, we assign a speed of zero. When we want vehicles to slow down when approaching a control point (e.g., for a turn), the associated speed is specified as needed. Posted speed limits are implemented in this manner. Control points provide the capability to anticipate upstream speed restrictions. User code can be called when an approaching vehicle "hits" the speed profile associated with a control point and when an approaching vehicle passes over, or stops on, a control point. Fixed points do not have an associated speed. Therefore, user code is executed only when a vehicle passes over a fixed point. The user routines associated with control points and fixed points are referred to as handlers. Handlers are discussed in the next section.

4.2.2 The FTC Application

The simulation engine provides upper layers with all the functionality required to simulate complex vehicle behavior. The FTC application was built on top of the simulation engine primarily through the use of event handlers. SLX does not directly support a "pointer to a procedure" mechanism. However, the statement definition capability allowed us to develop a mechanism that is a proxy for a pointer to a function. The details of the handler implementation are encapsulated into a handful of user level statements. The user receives the functional benefit without being concerned with the implementation details.

Each code handler is defined with a unique name. Once defined, a handler can be registered with any number of control points, fixed points, or even called directly from other handlers. The handler mechanism allows us to write short paragraphs of control logic and attach this logic directly to the network layout. As vehicles move across the network, the handlers are automatically called by the simulation engine. When a handler is called, a pointer to the vehicle and a pointer to the control/fixed point are automatically passed to the handler code by the simulation engine. This allows references to the vehicle (and surrounding vehicles) and to the network itself. Arbitrarily complex control logic can be developed by developing handlers and graphically placing control/fixed points.

A discussion of the control logic for routing vehicles through an intersection highlights the fundamental concepts of handlers, control points, fixed points, and SLX "wait until". Figure 1 shows the Frankford Avenue and Pratt Street intersection. The arrows indicate the direction of travel. The paths into and out of the intersection are labeled with the origin or destination direction (NB = North Bound, SB = South Bound, EB = East Bound, WB = West Bound) and lane number when there are multiple lanes.

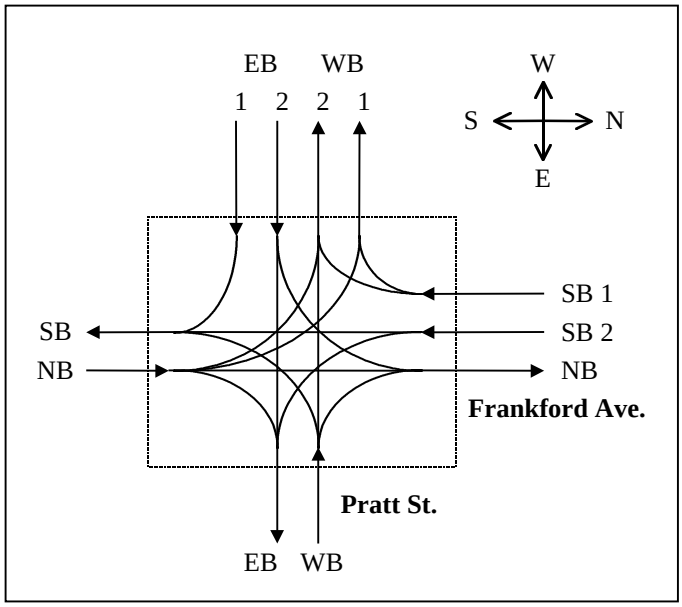

Figure 1: Frankford Ave. and Pratt St. Intersection

For example, there is one lane entering the intersection traveling north (left to right) on Frankford Avenue. For vehicles turning left from Frankford Avenue onto Pratt Street, there are two westbound lanes (1 and 2) exiting the intersection. For vehicles continuing north on Frankford Avenue, there is one northbound lane exiting the intersection. For vehicles turning right from Frankford Avenue onto Pratt Street, there is one eastbound lane exiting the intersection.

For discussion, the notation "NB-WB2" indicates the direction and lane (if necessary) for entering the intersection and the direction and lane (if necessary) for exiting the intersection. Thus, "NB-WB2" is the path entering the intersection northbound and exiting the intersection westbound in lane 2. This is a left turn from Frankford Avenue onto Pratt Street.

The dashed rectangle represents the extents of the intersection for control purposes. The intersection stop lines are located at the intersection of the dashed rectangle and the entrance paths of the intersection. The stop lines are modeled with zero speed control points. The intersection of the dashed rectangle and the exit

paths of the intersection are modeled with fixed points. These fixed points represent the physical point at which a vehicle leaves the intersection. Each path through the intersection can be described by a unique control point and fixed point pair. A vehicle enters the intersection passing over a control point and leaves the intersection passing over a fixed point.

Control points and fixed points are associated with either the leading edge, center, or trailing edge of a vehicle. For intersections, the stop lines are modeled as leading edge control points (i.e., the speed profile is computed based on the leading edge of the vehicle coming to rest on the control point). The fixed points exiting the intersection are based on the trailing edge of the vehicles (i.e., the handler is called when the tail of a vehicle passes over the fixed point).

All vehicles that enter the intersection in Figure 1 northbound on Frankford Avenue will stop, if necessary, at the same control point regardless of their destination path (i.e., the paths "NB-WB1", "NB-WB2", "NB-NB", and "NB-EB" all share the same entrance location). However, the decision logic for proceeding through the intersection is very different for each possible path. Each path through the intersection has an associated handler for logical control of the intersection. When a vehicle hits the speed profile for a stop line control point, the route list of the vehicle is searched for an exit fixed point of the intersection. Once an exit fixed point is found, the handler associated with the control point/ fixed point pair is called.

The speed profile represents the critical point (based on vehicle speed and remaining distance to the control point) for deciding whether to stop at, or continue beyond, the control point. Once called, the vehicle will decelerate and/or remain at rest at the control point as long as the handler is active. As soon as the handler becomes inactive, the vehicle may proceed through the intersection. Since the mechanics of vehicle control are automatically handled by the simulation engine, our focus at this level is on the logical control of the intersection. The handlers for intersection control typically tend to be very small (most are 4 statements).

The primary statement in an intersection handler is an elaborate "wait until" expression. All of the necessary conditions for intersection control can be specified in a single statement. The wait until expressions can be decomposed into three logical groups, all of which must be satisfied simultaneously: (1) the signal aspect is favorable and all rules-of-the-road are satisfied, (2) there is available capacity on the destination link, and (3) there are no vehicles that have been granted conflicting routes. The first group is used to test the signal aspect and governs driver behavior (e.g., left turn arrow is green, right turn on red is okay, etc…). The second group guarantees that there is room for the vehicle upon exiting the intersection (i.e., vehicles are not allowed to stop in the middle of an intersection). The third group is "zone control" (i.e., there are no vehicles with conflicting routes about to enter, or still in, the intersection).

In the FTC application, each logical group is comprised of a list of sub-expressions. For example, the entire wait until statement for the path "NB-WB1" contains seventeen (17) terms. Even though the wait until statements tend to be lengthy, this mechanism is dramatically easier (and less error prone) than the alternative of managing the model state changes and re-evaluating the necessary conditions ourselves.

4.3 Model Outputs

The outputs from the simulation model include a summary report, detailed boarding and alighting logs, and the animation trace file.

The summary report provides the number of buses, automobiles, and cars to enter and leave the simulation. In addition, the number of automobiles by entry location, the number of buses by service, and the number of passengers picked-up and dropped-off by service is reported. The information contained in the summary report is a reflection of the input data.

The boarding log provides an entry for each pick-up event. Each entry is composed of the service, berth location, arrival time, departure time, scheduled departure time, total dwell time, number of passengers boarded, and the number of passengers left waiting (if any). The number of passengers left waiting is an indication that the headway between buses is too long (poor scheduling) and/or significant delays due to congestion, drop zone availability, or berth availability.

Similarly, the alighting log includes an entry for each drop-off event. The alighting log includes the service, drop zone (location name and position), arrival time, departure time, and the number of passengers to alight.

Features of the animation include the road boundaries and lane markings for the entire study, the building and street furniture outlines, all vehicle movements (bus, trolley, and MFSE rail), pedestrian counts (numerical values at places of interest), and traffic signal aspects at each intersection.

The animation proved to be a valuable tool for evaluating and resolving schedule conflicts. The interaction and competition between buses, services, drop zones, berths, and passengers becomes evident by viewing the animation. Modifications to the schedule could be tested and compared under identical system conditions.

5 RESULTS

For the most part, the design of the proposed facility works well under the current bus and train schedules. However, certain changes to the operating plan will improve overall reliability at the FTC. The following are general observations and specific recommendations concerning the operating plan:

- Where there are 2 or more boarding berths, buses seldom if ever wait for a berth to clear. The few conflicts can be corrected easily by small shifts in the schedule.
- Where there are shared berths, care must be taken to schedule arrivals at appropriate intervals so that conflicts are minimized.
- Where there are single berths, care must be taken to schedule revenue moves and pull-outs to minimize conflicts.
- Layover/recovery time for all routes must be reduced at FTC by moving the layover/recovery time to the other end of each route. Minimizing the time at FTC will provide greater reliability and throughput.
- Spare boarding berths located close to the designated berths provides an easy fix to unanticipated conflicts. Placing the spare berths near to the designated berths will reduce confusion to passengers.
- Unloading berths can be reduced in length. The number of buses unloading concurrently was consistantly less than expected. The free space can be used for spare loading berths.

6 CONCLUSIONS

The simulation model and animation allowed the project engineers to evaluate bus and train schedules against the physical layout, assignment of drop zones, assignment of berth locations, intersection signal phasing, and the quality of service for passengers. The simulation results showed that the facility, as designed, provides for future expansion of bus and rail service. There were no significant delays to vehicles or passengers during the AM or PM peak periods.

ACKNOWLEDGEMENTS

The authors would like to thank Jim Henriksen of Wolverine Software Corporation for his unparalleled technical support, his unflagging patience, and for answering an inordinate number of questions concerning the innards of SLX.

REFERENCES

Atala, O.M., J.C. Brill, and J.S. Carson, 1992. A General Rapid Transit Simulation Model With Both Automatic and Manual Train Control. *Proceedings of the 1992 Winter Simulation Conference*, ed. J.J. Swain, D. Goldsman, R.C. Crain, J.R. Wilson, 1307-1311.

Banks, J. and R. Gibson, 1997. Simulation Modeling: Some Programming Required. *IIE Solutions*, February 1997, 26-31.

Banks, J. and R. Gibson, 1997. Selecting Simulation Software. *IIE Solutions*, May 1997, 30-32.

Henriksen, J.O., 1996. The Power and Performance of Proof Animation. *Proceedings of the 1996 Winter Simulation Conference*, ed. J.M. Charnes, D.J. Morrice, D.T. Brunner, J.J. Swain, 460-467.

Henriksen, J.O., 1996. An Introduction to SLX. *Proceedings of the 1996 Winter Simulation Conference*, ed. J.M. Charnes, D.J. Morrice, D.T. Brunner, J.J. Swain, 468-475.

Schriber, T.J. and D.T. Brunner, 1996. Inside Simulation Software: How It Works and Why It Matters. *Proceedings of the 1996 Winter Simulation Conference*, ed. J.M. Charnes, D.J. Morrice, D.T. Brunner, J.J. Swain, 23-30.

AUTHOR BIOGRAPHIES

JOSEPH C. BRILL is a Senior Simulation Engineer with Systemflow Simulations, Inc., a simulation consulting firm. He has developed a number of large-scale simulation models in the areas of manufacturing, material handling, distribution, warehousing, rapid transit, traffic systems, health care, mining, smelting, communications, and computer systems. He received a B.S. degree in Industrial Engineering and an M.S. degree in Manufacturing Systems from the Georgia Institute of Technology in 1987 and 1988, respectively. He is a member of IIE.

DUDLEY E. WHITNEY is a Senior Transportation Planner with Parsons Brinckerhoff Quade & Douglas, Inc., an engineering/planning consulting firm. He specializes in planning, analysis, and design of multimodal transportation systems. He is experienced in transit operations planning, simulation modeling, travel demand forecasting, conceptual level capital cost estimating, operating and maintenance cost analysis, and demographic forecasting. He received a B.S. in Environmental Design from North Carolina State University in 1982 and an M.S. in City and Regional Planning from the University of North Carolina in 1992.