Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

THE DESIGN, IMPLEMENTATION, APPLICATION AND COMPARISON OF TWO HIGHLY AUTOMATED TRAFFIC SIMULATORS

Peter Lorenz
Thomas Schulze

Department of Computer Simulation and Graphics
University of Magdeburg
D-39016 Magdeburg, GERMANY

Thomas J. Schriber

Computer and Information Systems
The University of Michigan
Ann Arbor, Michigan 48109-1234 U.S.A.

ABSTRACT

The combination of new developments in computing and new problems in practice stimulates new thinking about methods for producing models with which traffic systems can be simulated. This paper describes and compares two approaches aimed at largely automating the process of building rapidly executing micro-level simulation models for complex traffic networks. Two traffic simulators have been designed and implemented, one based on GPSS/H, the other based on a model-specific simulation engine written in Pascal. The alternative-language approaches are discussed in general terms and their use is illustrated in a specific application. Data file interfaces promote the integrated use of these simulation tools with other engineering tools commonly used for traffic system planning.

1 INTRODUCTION

Traffic problems are public problems with a high degree of public interest. Traffic engineers use both mathematical and simulation models in attempting to develop solutions for traffic problems. Mathematical models, based largely on queuing theory, are sometimes used to evaluate the performance of intersections that do not use traffic light controls or that use simple traffic-light control systems. However, mathematical models cannot be successfully used in all cases of practical interest, and this fact heightens the level of interest in simulation-based approaches for modeling transportation systems.

Simulation of transportation systems has been one of the important application areas of computer simulation for more than 20 years (Reitman 1971). Many different simulation tools have been developed throughout the world to address such problems (Lorenz 1993). Simulation is an engineering tool that is useful both for research and for practical applications. The further development and use of simulation-based approaches dealing with the design and control of traffic systems is stimulated by factors such as these:

✔ Increasing congestion in traffic systems;
✔ Use of new traffic-light control systems in which signal intervals are dynamically determined by taking traffic conditions into account;
✔ Availability of image processing software, which facilitates the use of maps as sources of topological and geometrical input data for traffic system models;
✔ The possibility of using animation software with flexible choices of scaling in time and space;
✔ Improved software for 3-dimensional presentation of results;
✔ Further advances in computer technology.

Micro-level traffic models and modeling systems have been developed to describe the performance of vehicles traveling over networks of streets. NETSIM (Lieberman and Andrews 1990; Lieberman 1991) appears to be the most popular and widely used traffic simulation software in the United States today. The internal logic of this micro-level traffic simulator describes the movement of individual vehicles responding to external stimuli including traffic control devices, the presence of other vehicles, pedestrian activity, and so on.

Micro-level models are complex and include abundant detail. Most micro-level models are based on a fixed time increment approach to clock management. For example, NETSIM uses a fixed time increment of one second. Use of a fixed time increment is the main reason that many micro-level traffic simulations consume large amounts of execution time.

The execution time of simulation models plays an important role in their acceptance and may indirectly influence the quality of results if long execution times discourage modelers from carrying out the full set of experiments required to reach statistically sound conclusions. Consider the fact that stochastic distributions are used to describe the characteristics of

traffic systems. The interarrival time of vehicles, for example, is often assumed to follow the high-variance exponential distribution. Stochastic output parameters such as average delay time per vehicle are then often correspondingly characterized by a high variance. In our experience, on the order of 50 replications may be needed to estimate the expected values of response variables with sufficiently precise confidence intervals in many traffic simulations. This means that traffic simulation modelers must be able to run experiments quickly. The use of a powerful CPU can't solve the problem alone. Efficient tools and algorithms are also needed.

The Department of Computer Simulation and Graphics at the University of Magdeburg is working on the development of prototypes for a new generation of traffic simulation tools. Among the characteristics of the prototypes are the following:

✔ Ability to incorporate a high level of modeling detail with regard to geometric data;

✔ Reduction in the time and money needed for model creation, thanks to use of topological and geometric data obtained from existing maps and/or layouts;

✔ Automatic incorporation of traffic-light control specifications from engineering information;

✔ Extremely short simulation runtimes;

✔ Integration of simulation and animation models with other tools commonly used by traffic engineers;

✔ Support for traffic engineers in the optimization of network layout and signal controls.

We have designed and implemented two different sets of software to support building and running traffic-system simulations. One uses a problem-specific simulation engine written in Pascal and the other uses GPSS/H. Both have the common objective of producing micro-level traffic-system simulation models that execute at high speed. Both work with a common input data interface and use a common Layout File to produce Animation Trace Files for subsequent post-simulation animation based on Proof Animation (Henriksen et al., 1992). Finally, both also produce the same set of output parameters, including statistics.

The reason for developing two simulators was based in part on an interest in trying to determine answers to the following questions:

✔ What range of approaches can be used to reduce the manual effort required to create micro-level traffic models that have a high degree of correspondence with the real system?

✔ Is it possible to design and implement a common approach that can be used to support two alternative simulators whose purpose is to reduce the manual modeling effort otherwise required to produce the models of interest?

✔ Is it possible to combine the GPSS/H simulator with other engineering tools often used by traffic system engineers?

Section 2 of the paper describes our view of the logical structure of a general micro-level traffic model and Section 3 discusses some basic ideas for reducing the manual effort through use of a special data file interface. Sections 4 and 5 comment on the GPSS/H and Pascal simulator implementations, respectively. Section 6 presents an application of the simulators. Our experiences to date in this ongoing project are summarized in Section 7. Conclusions are then drawn in Sec-tion 8, after which Acknowledgements and References are given.

2 LOGICAL MODEL STRUCTURE

Our view of a traffic system is like a view from a stationary helicopter hovering overhead. Entities or instances of various object classes either occupy fixed positions or move from point to point in the system. The movement experienced by objects depends on such things as road signs, traffic regulations, traffic lights, origin, destination, speed and object class characteristics.

A traffic system can be divided into subsystems, with each subsystem usually involving traffic nodes (intersections). A traffic system consisting of only one subsystem (e.g., an intersection) is of interest as a special case.

A subsystem is comprised of stationary and statistical elements which are local to the subsystem. Moving entities are global, moving through different subsystems as they progress through the overall system.

Moving entities can be divided into the following classes:

✔ Cars, trucks, buses and taxicabs (motorized vehicles that do not move on fixed paths);

✔ Street cars (motorized vehicles that move on fixed paths);

✔ Bicycles;

✔ Pedestrians.

These entities have common types of attributes such as origin, destination, speed, size and separation ("clearance"). Various techniques are used to model such things as vehicles passing other vehicles, overall compliance with traffic regulations, traffic control, and the behavior of other moving entities.

Stationary modeling elements include lanes or paths, traffic signs, traffic lights, and entry and exit points. There are lanes or path subtypes for the various types of moving entities.

Traffic lights can be controlled by a variety of approaches. Fixed-time control approaches are not as flexible as load-dependent control approaches, in which detectors collect information about traffic location and intensity and the control system regulates the traffic

lights accordingly.

Various statistics are collected, for example:

✔ Number of entities moving through the system;
✔ Average utilization of specific lanes and paths;
✔ Average travel time through the overall system and subsystems for each entity class and traffic stream (where a traffic stream is a set of moving entities with the same origin and destination);
✔ Average delay time within the overall system and subsystems for each entity class and traffic stream.

3 DATA-FILE INTERFACES

Per the Section 1 objective of reducing the amount of engineering effort required to create micro-level traffic models with details faithful to the real system, it is desirable to take advantage of information that is available in machine-readable form when creating the model for the system. Information needed by the model encompasses:

✔ Topological data for the system, including relations between different lanes (e.g., the crossing over of lanes, the splitting of single lanes into two or more lanes, and the merging of two or more lanes into single lanes); and between lanes and sources of traffic, sinks for traffic, traffic signs, traffic lights and traffic streams;
✔ Geometric data for the system, including the length of lanes, the positions of lane end points and the positions of detectors (traffic sensors);
✔ Traffic-load data as collected by traffic counting devices and stored in databases in traffic engineering CAD systems;
✔ Traffic control data and algorithms represented in flowcharts developed by traffic engineers (Walper 1994).

These data should be available to a traffic simulation model in highly automated fashion. Here we comment on one such possibility: the extraction of geometric and topological data from existing traffic-system maps, the capturing of this information supplemented by creative traffic-engineering decisions in a Proof Animation Layout File, and then the use of network analysis to produce a Final Path File, as shown in Figure 1.

The Initial Layout File or "drawing," which contains the contours of the traffic system elements, is produced by computer from a map of the real system via scanning and vectorizing. The construction of paths, the choice of path names, and the selection of positions for various classes of objects (e.g., traffic lights, detectors, traffic signs) must then be accomplished interactively by the modeler or traffic engineer (using tools provided by Proof Animation), as depicted in Figure 1. The results of creative traffic engineering are captured and expressed through this interactive input, in which the traffic engineer or modeler transforms the initial Layout File-

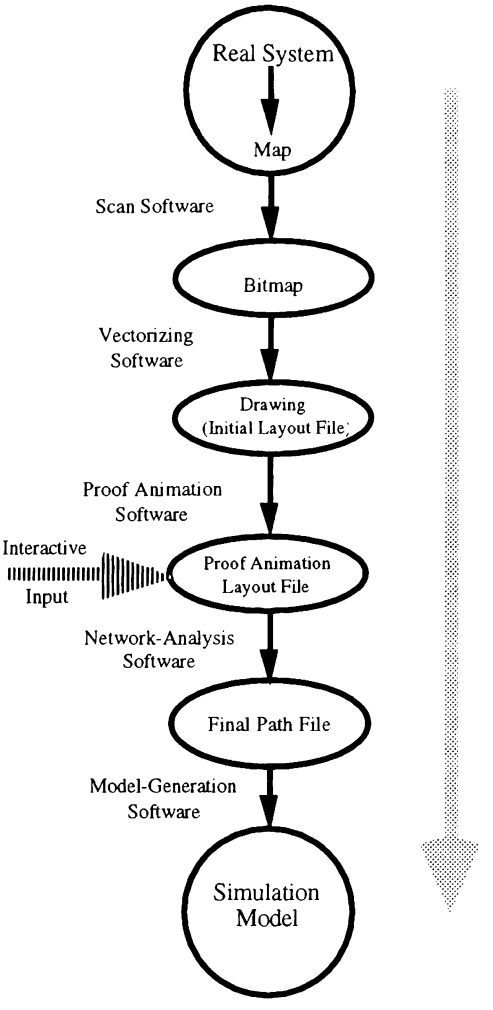

Figure 1: Data Transfer from the Real System to the Simulation Model

into a Proof Animation Layout File that contains-all needed information for the next step: network analysis.

As shown in Figure 1, network analysis uses as its input the Proof Animation Layout File, which contains the names and numbers of all paths and input/output points to which the paths belong. The network analysis looks for all the paths, input and output points and nodes defined in the Layout File and identifies the crossing, splitting and merging of these paths. The positions of signs, traffic lights and detectors on paths are also registered as attributes of the corresponding paths. Finally, it is determined which target directions are accessible for each path. The resulting Final Path File then serves as input to the simulation model, as suggested in Figure 1.

An example of the records making up a typical Final Path File is shown in Figure 2. The information in a Final Path File can be used by different simulation mod-

```
Final_Path= Record
  case Rec_Ident of
  'F': {Following Paths}
    (Number_of_FP, List_of_FP);
  'E':  {mErging Paths}
    (Number_of_Right_MP,Number_of_Left_MP,
                                  List_of_MP);
  'C':  {Crossing Paths}
    (Number_of_CP, List_of_CP);
  'S': {Signs on Paths}
    (Number of Signs, List_of_Signs);
  'L': {Traffic Lights on Paths}
    (Number of Lights, List_of_Lights);
  'D': {Detectors on Paths}
    (Number of Detectors, List_of_Detectors);
```

Figure 2: An Example of the Records Making Up a Typical Final Path File

els. Other model description data needed for model construction are traffic load data and traffic light control data. These data can be extracted from data files available in the context of traffic engineering projects.

In the following sections two different modeling approaches are discussed and compared: a GPSS/H-based approach and an approach based on an application-specific simulation engine written in Pascal.

4 IMPLEMENTATION OF THE GPSS/H-BASED SIMULATOR

In this section we describe our experiences to date in using GPSS/H to build a prototype simulator corresponding to the conceptual "Simulation Model" of Figure 1. The initial work has been aimed at building a simulator designed with the objective of modeling single but generalized and realistically complex traffic intersections. A model of a single traffic intersection has to represent the following aspects of the intersection:

✔ Topology and geometry of traffic lanes;
✔ Rules of behavior for drivers, pedestrians, bicyclists and street cars;
✔ Traffic load and traffic streams;
✔ Traffic light and sign positions and relations to traffic lanes;
✔ Traffic light control cycles, phases, decision rules and transition schemes;
✔ Data collection and report generation;
✔ Experimental conditions.

The first five of the above items usually involve an abundance of intersection-specific data. It is a challenging task in its own right to reduce to a minimum the amount of engineering time required to supply this information. Quite apart from this task, it is also a challenging task to specify a general solution not only for one intersection, but eventually for a set of linked traffic intersections, a traffic network or an entire system.

To provide insights into the potential gain resulting from development of a code-generating program, we first built "by hand" GPSS/H modules for modeling a single intersection. The source code in the resulting GPSS/H model consists of about 4000 statements. After expansion of macros included among the 4000 statements, the compiled form of the model consists of about 7000 statements. More than 50% of these statements structure the output and specify the experiments to be carried out with the model. Some 10% to 15% of the statements are application dependent and, in the absence of an automatic code generator, would have to be developed manually for each new application. The proportion of statements used to specify various aspects of the model are summarized in Figure 3.

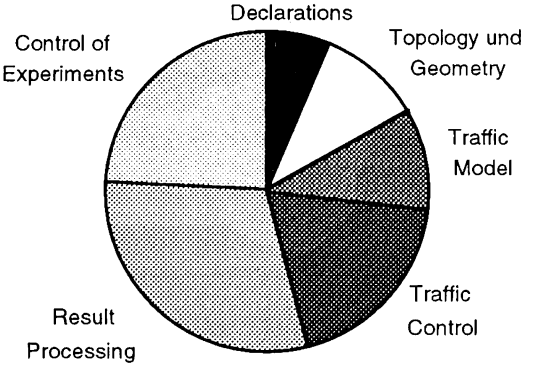

Figure 3: Relative Proportions of GPSS/H Model Statements in a Typical Traffic-Intersection Model

An important goal for the automatic code-generating software we are developing is to reduce the hand-tailored application-specific part of the code from about 10-15% of the model to about 0-1% of the model.

The GPSS/H modeling language has both strengths and weaknesses when used to write a program that will read records from a Final Path File (see Figure 2) and generate GPSS/H statements for a corresponding traffic simulation model. Excellent help is provided for this purpose in the form of GPSS/H entity-type Functions (S-Functions; Henriksen and Crain 1989). Our experi-

ence has shown that it is possible to define all needed data structures for a traffic system on the basis of the Final Path File, and to build the corresponding GPSS/H entity-type functions using a code-generating program which reads that file as input. In general, such a code generator could be written in any number of languages. GPSS/H has enough file and string processing capability, for example, to support the use of GPSS/H to write such a generator in an easy and natural way. Figure 4 provides a glimpse of aspects of such a code-generating program, and shows a sample of the resulting GPSS/H code generated via application of the program.

The two most notable weaknesses of GPSS/H when used to write a GPSS/H code generator are these:

✔ Absence of a control-statement SUBROUTINE capability;

✔ Lack within the control-statement language of a block structure which allows the use of local variables.

On the other hand, three relatively new features provided in GPSS/H Professional which have proved to be very useful in using GPSS/H to generate GPSS/H code are these:

✔ The INSERT control statement for including files;

✔ The SYSCALL statement for calling external programs;

✔ Statements for string and file processing, and the powerful features for combining numeric and symbolic identifiers (e.g. Entity Functions and the Standard Character Attribute SYM).

5 IMPLEMENTATION OF THE SIMULATOR IN PASCAL

As explained in Section 1, an important part of our research is to use different simulators to implement our model generating concept. In the last section we described the use of GPSS/H, and in this section we discuss our experience to date in the use of Turbo-Pascal.

Many simulation languages and simulation systems exist in the simulation world. Some of them are well tested, are widely and successfully used and are portable across different computer hardware. Why did we start to develop a new simulator? Traffic simulators, like communication network simulators, have to handle many events because hundreds of entities move through a traffic system at the same time. Our goal here is to create a very fast simulator, designed specifically for traffic simulations, which is not limited in model size.

We gave our special simulator tailored features to handle the high volumes of specialized events that can occur under conditions characterizing traffic networks. We wanted to know whether it is possible to justify the amount of time needed to develop the basic routines for a special simulator and whether such a special simulator

```
         PUTSTRING  FILE=GPS,('     INTEGER    &IPATH        CURRENT NUMBER')
         PUTSTRING  FILE=GPS,('     INTEGER    &NPATH        TOTAL NUMBER')
         PUTPIC     FILE=GPS,LINES=2,(&NPATH,&NPATH)
         LET        &NPATH=*
  PATHS  FUNCTION    &IPATH,S*,F,L,S,C,XB,XH,XF,XL,B          ALL PATHS
  MFNGEN STARTMACRO              GENERATING OF FREE FORMAT
*                                FUNCTION FOLLOWER STATEMENTS
         LET &INBUF=''
         DO #A=1,#B
          LET       &INBUF=&INBUF||&CZAHL(#A)||&KOMMA||#C(#A)||'/'
          IF (LEN(&INBUF)>60)OR(#A=#B) THEN
           PUTPIC   FILE=GPS,(&INBUF)
*
           LET      &INBUF=''
          ENDIF                        LEN...
         ENDDO                            #A
         ENDMACRO
  MFNGEN MACRO       &IPATH,&NPATH,&PATHS      FUNCTION FOLLOWER STATEMENTS
```

(a) Some of the GPSS/H Statements in a GPSS/H Code-Generating Program

```
    INTEGER    &IPATH         CURRENT NUMBER
    INTEGER    &NPATH         TOTAL NUMBER
    LET        &NPATH=61
  PATHS  FUNCTION    &IPATH,S61,F,L,S,C,XB,XH,XF,XL,B          ALL PATHS
  1,SCCSN/2,SCCSN1/3,SCCSN2/4,SCCSS/5,SCCSS1/6,SCCSS2/7,SCGOS/8,SCPIS/
  9,SCCSNX/10,SCCSSX/11,SCGOSX/12,SCPISX/13,SCCSNG1/14,SCCSNR1/
```

(b) Part of the GPSS/H Code Produced by the GPSS/H Code Generator

Figure 4: An Example of GPSS/H Code Generation Using GPSS/H

will execute fast enough to support needed statistical experimentation in the time frame within which finalized decisions have to be made.

Turbo Pascal was chosen as the implementation language because in many cases other existing engineering tools for traffic planning are written in Turbo Pascal. Were this not so, another reasonable choice of implementation language would have been C.

The Pascal simulator we have built uses structured programming techniques and is based upon an event-oriented simulation approach. The steps carried out by the simulator are outlined in Figure 5.

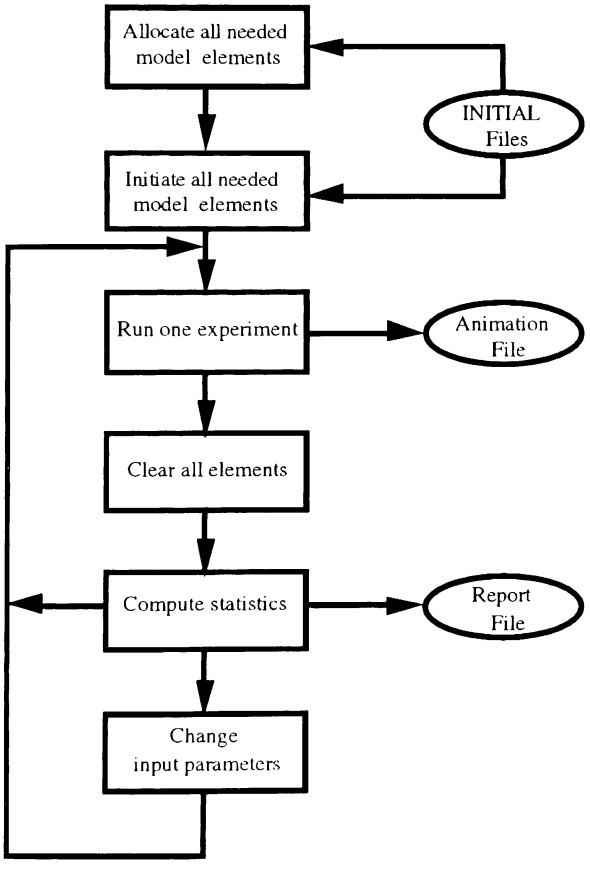

Figure 5: Processing Steps Carried Out by the Simulator Implemented in Pascal

In one approach, a traffic simulation model could be written in Pascal for each new application. Such an approach is not of interest here, however, because our objective is to dramatically reduce the amount of time needed to create and validate new traffic system modeling applications. The Final Path File of Figure 1 describes the structure of the specific traffic system of current interest. Using information read from this file, the Pascal simulator creates the elements used to produce a corresponding traffic simulation model. The model elements consist solely of dynamically allocated data, resulting in a customized traffic simulation model produced without the need to compile any Pascal code.

To provide some insights into the construction of the Pascal simulator, consider how paths are handled. Paths provide a mechanism for the guided movement of entities. A path is determined by two basic types of elements: edges and nodes. Figure 6 is an extract from the Pascal Type definition component of the Pascal simulator which shows the attributes of edges and nodes.

```
t_path  = RECORD
 edge : ^t_edge     ; edge of the path
 node : ^t_node     ; node for the path
end;

t_edge = RECORD
 ident :            ; path identification
 next_edge :        ; handling all edges
 locked_ent_clas :  ; path sometimes locked
 destination :      ; reachable exit points
 home_path :        ; path including this edge
 length :           ; length of the path
 next_node :        ; next node
 pred_node :        ; predecessor node
 signs :            ; traffic sign on path
 cross_edges:       ; crossing edges
 move_queue:        ; handle moving vehicles
 wait_queue:        ; handle waiting vehicles
end;

t_node = RECORD
 ident :            ; node identification
 next_node :        ; handling all nodes
 stop_by_lights:    ; stopping signal lights
 input_edges :      ; incoming edges
 output_edges :     ; outgoing edges
 owner :            ; blocked by an entity
end;
```

Figure 6: An Example of Pascal Type Definitions for the Edges and Nodes Making Up a Path

The Final Path File and other input data files have to be read two times. Memory allocation for each element and insertion of this element into the internal handler are steps carried out in the first pass. Each element is identified in this pass and pointers are established. The needed remaining information is developed in the second pass, during which the elements are connected via pointers and initial events are created.

The run-time system is highly event oriented. There are four types of events that can be scheduled for future

occurrence: move_end (to indicate that movement of an entity along an edge has been completed); stop_end (to indicate that an entity has stopped moving); signal_switch (to indicate that the setting of a signal has been changed); and create (corresponding to creation of a new entity).

Special features for collecting statistics have also been implemented in the Pascal simulator. Statistical procedures are automatically called after every replication of the simulation and after all replications have been completed for the specified experimental design. In addition, all aspects of the model are re-initialized between consecutive replications.

6 AN APPLICATION OF THE GPSS/H AND PASCAL SIMULATORS

The development of the new GPSS/H and Pascal traffic simulation tools was carried out with the active involvement of several German traffic engineering companies. One of these companies had been contracted to plan and install a traffic-light control system for an intersection in Schoenebeck, a town near Magdeburg, Germany. Figure 7 shows a layout of the intersection.

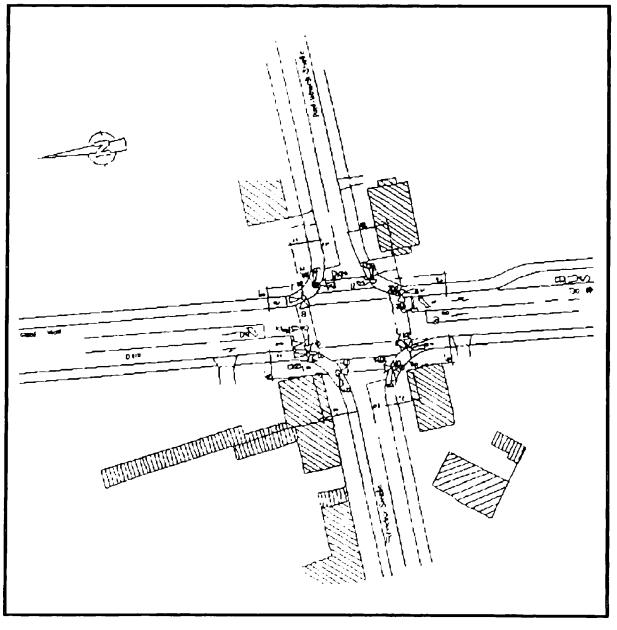

Figure 7: An Intersection in the Town of Schoenebeck, Germany

Using the GPSS/H and Pascal traffic simulators, the intersection of Figure 7 was modelled by making use of the types and numbers of model elements summarized in Figure 8.

Type of Model Element	No. of Elements
Traffic load situations	6
Traffic light control programs	3
Paths	61
Cars per hour	100-2000
Pedestrians per hour	0-5000
Traffic lights	10
Detectors	8

Figure 8: Types and Numbers of Elements Used to Model the Intersection of Figure 7

Figure 9 shows a view of the Proof Animation Layout File corresponding to the Figure 7 intersection and used to generate the Final Path File of Figure 1 and other input data files. The Proof Animation Layout File

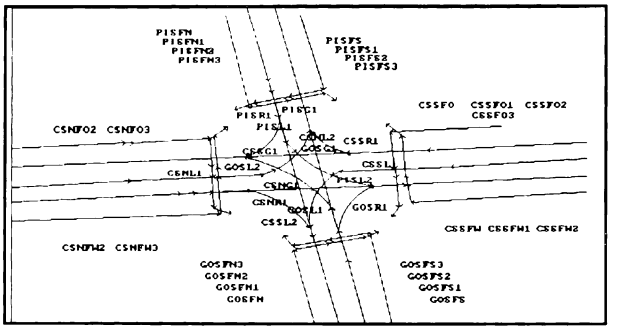

Figure 9: The Proof Animation Layout File for the Intersection of Figure 7

not only provides a means for supplying geometric specifications for the simulator, as pointed out in Section 3, but is also instrumental in supporting post-simulation animation of traffic simulations. In our experience, animation itself is a very valuable validation tool for traffic simulations. It is our judgment that whenever a change is made in the structure proposed for a traffic system, the corresponding animation of the changed model should be viewed carefully. In this context, animation is useful for the following purposes:

- ✔ Detect collisions of entities;
- ✔ Analyze the reasons for traffic jams;
- ✔ Explain the model to various users;
- ✔ Establish credibility for the model.

Figure 10 shows a snapshot of an animation of the Figure 7 intersection when it is in heavy use by pedestrian traffic. It is an attribute of so-called robust models that they are capable of handling extreme conditions such as this.

Moving pictures (animation) can be impressive and advantageous, but they are not usable for the ultimate evaluation and analysis of simulation results. The

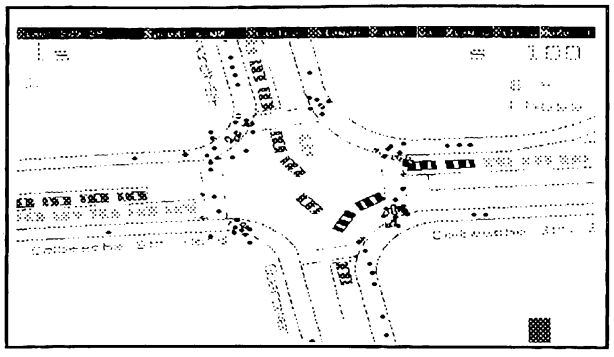

Figure 10: An Animation Snapshot
of the Figure 7 Intersection
(small squares are pedestrians and larger rectangles are cars; the snapshot has been highly compressed to fit this column, with a resulting loss of crispness in the details)

results of experiments must be extracted from a huge amount of data by means of statistical methods. Figure 11 shows an example of statistical analysis and comparison of waiting times at the Figure 7 intersection for automobiles under a range of traffic-load conditions. The generation of graphs of the type in Figure 11 is easily accomplished with the use of spreadsheet software and simulator report files.

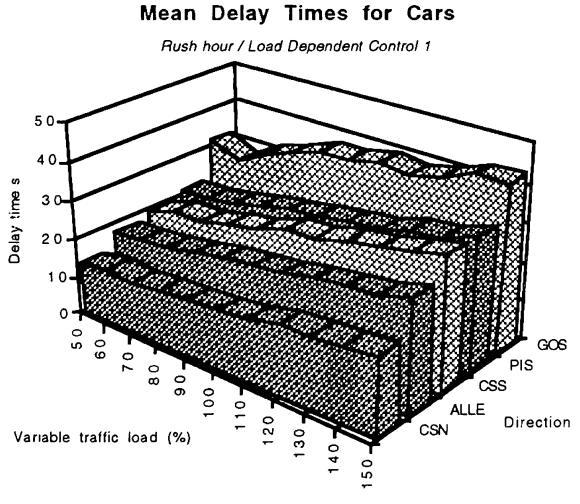

Figure 11: Average Car Waiting Times As a Function of Traffic Intensity at the Intersection of Figure 7

7 A SUMMARY OF EXPERIENCES AND FINDINGS TO DATE

Our experiences to date in the development and use of the GPSS/H- and Pascal-based traffic simulators can be summarized as follows:

1. The concepts we have implemented have been applied in the rigorous modeling of two traffic intersections in Germany, one involving 10 traffic lights, as described in Section 6, with the other being a larger scale system involving 33 traffic lights and associated vehicle traffic (cars, trucks and buses), street cars, bicyclists and pedestrians. The modeling of these intersections has been judged successful in the sense of uncovering problems in existing or proposed traffic-control procedures and finding improved traffic-control schemes.
2. These successful applications demonstrate the feasibility of basing traffic-model generators in part on graphic representations of the system to be modeled, e.g. extracting information from Proof Animation Layout Files to support model generation. The use of Proof Animation Layout Files has the additional follow-on benefit of supporting realistic post-simulation animation of the simulated traffic system.
3. Generation of the traffic models is additionally based on computerized recognition of all geometrical and topological relations and of the various objects involved in controlling the traffic system.
4. Our approach uses file-interfacing techniques to achieve compatibility with traditional traffic engineering software systems and thereby take advantage of traditional sources of traffic-engineering information.
5. Our concepts and methods for working with graphical representations and taking traditional traffic engineering information into account are general to the extent that they seamlessly accommodate two quite different approaches for creating customized micro-level traffic models: one approach based on an existing general purpose simulation language (GPSS/H); the other approach based on a model-specific simulation engine written in a programming language (Pascal).
6. The GPSS/H-based and the Pascal-based approaches are both easy to use and produce consistent results.
7. Our methodology substantially reduces the amount of engineering effort required to create micro-level traffic models with details faithful to the real system.
8. In our usage experience, the computer time required to carry out simulations with the traffic models generated under either approach are modest enough to invite comprehensive statistical investigation of system performance in practice. For example, the application reported in Section 6 required under 20 CPU seconds per replication on a 486-based PC, with the duration of each replication being 1 simulated hour. The simulators need to be optimized with respect to their run-time performance, however, before it will be appropriate to carry out rigorous timing studies with them.

Work on and with the GPSS/H- and Pascal-based traffic simulators is ongoing. For example, there is room

for refinements in some of the underlying algorithms used to handle the detailed movement of traffic in such areas as the modeling of car-following and lane-switching. And additional experience gained in practical application of the simulators will no doubt produce insights that will lead to enhancements in the usability and utility of the methodology.

8 CONCLUSIONS

The use of geometric oriented data sources is a practicable and efficient way to provide traffic-system simulation models with the geometric and topological data needed to generate models that have a high degree of correspondence with the real system. Use of the Layout File inherent in Proof Animation provides one mechanism for bringing this about. The Layout File has two roles to play in this regard:

1. It provides topological and geometrical data to the traffic model;
2. It supports a realistic post-simulation animation of the simulated traffic system.

It is both possible and advantageous to extract topological and geometric data from available machine-readable sources of such information and to store these data in an interface file formatted in a manner designed to be usable by one or more simulators.

Preliminary results with our prototypes show that it is possible to develop vehicle traffic simulators with favorable runtime performance characteristics. On this basis it is feasible to carry out appropriate statistical experimentation in timely fashion to identify approaches that improve the performance of traffic control systems.

Finally, we note that the techniques developed here in the area of layout-based model generators (using traditional engineering drawings as the basis for computer-supported model generation and follow-on animation) are not restricted to traffic systems as an application area, but have potential broader-based applicability as well.

ACKNOWLEDGEMENTS

The final form of this paper has benefitted substantially from suggestions made by Robert C. Crain, James O. Henriksen and Douglas S. Smith, of the Wolverine Software Corporation.

REFERENCES

Henriksen, J. O., and R. C. Crain. 1989. *GPSS/H Reference Manual*, Third Edition. Annandale, Virginia: Wolverine Software Corp. (Translated into German by Ines Kuhrau and available under the title *GPSS/H Referenzhandbuch*.)

Henriksen, J. O., D. T. Brunner, and N. J. Earle 1992. *Using Proof Animation* (with Student Proof Animation on an included disk). Annandale, Virginia: Wolverine Software Corporation.

Lieberman, E. B. 1991. Integrating GIS, Simulation and Animation. In: *Proceedings of the 1991 Winter Simulation Conference*, eds. B. L. Nelson, W. D. Kelton, and G. M. Clark, 771-775. La Jolla, California: The Society for Computer Simulation.

Lieberman, E. B., and B. Andrews. 1990. The Role of Interactive Graphics when Applying Traffic Simulation Models. In: *Proceedings of the 1990 Winter Simulation Conference*, eds. O. Balci, R. P. Sadowski, and R. E. Nance, 753-758. La Jolla, California: The Society for Computer Simulation.

Lorenz, P. 1993. Bilder und Modelle Ampelgesteuerter Verkehrskreuzungen. *Tagungsband des Achten Symposium Simulationstechnik*, 359-362. Berlin: Vieweg Verlag.

Reitman, J. 1971. *Computer Simulation Applications*. Wiley Interscience, New York, NY.

Smith, D. S., D. T. Brunner, and R. C. Crain. 1992. Building a Simulator with GPSS/H. In *Proceedings of the 1992 Winter Simulation Conference*, eds. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson, 357-360. La Jolla, California: Society for Computer Simulation.

Walper, R. 1994. *Generierung von Modellen der Lichtsignalsteuerung fuer den Strassenverkehr*. Masters Thesis. Magdeburg, Germany: Department of Computer Simulation and Graphics.

AUTHOR BIOGRAPHIES

PETER LORENZ is a Professor in the Department of Computer Simulation and Graphics at The University of Magdeburg. He has over 20 years of experience in the field of simulation. A recent research interest of his is picture-based generation of simulation models, using both real and abstract pictorial system representations.

THOMAS J. SCHRIBER is a Professor of Computer and Information Systems at The University of Michigan. The author of *An Introduction to Simulation Using GPSS/H* (Wiley, 1991), he teaches, does research and consults in the area of discrete-event simulation.

THOMAS SCHULZE is an Associate Professor in the Department of Computer Simulation and Graphics at The University of Magdeburg. His research interests include modeling methodology, public systems modeling and simulation output analysis. He is an active member of ASIM, the simulation society for Germany, Austria and the German-speaking part of Switzerland.