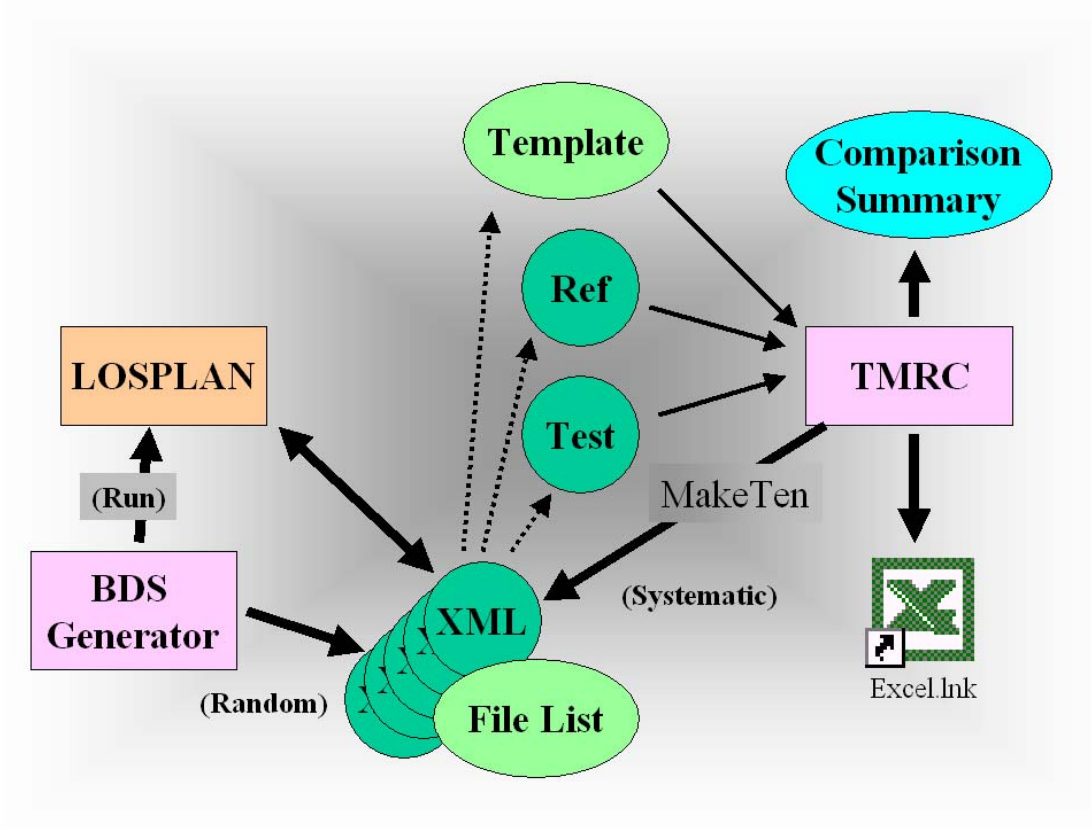
Final Report
Contract BB-235



Development of Benchmark Data And Procedures for Testing Traffic Operations Models

Prepared for
The Florida Department of Transportation
Systems Planning Office

By The University of Florida
Transportation Research Center

March 2002

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle	5. Report Date
Development of Benchmark Data and Procedures for Testing Traffic Operations Models	March 2002
	6. Performing Organization Code: 4910450459612
7. Author(s): Ken. Courage, Nan Zhao, Morya. Willis	8. Performing Organization Report No.: TRC-596-F
9. Performing Organization Name and Address: University of Florida, Department of Civil and Coastal Engineering, 124 Yon Hall / P.O. Box 116580, Gainesville, FL 32611-6580	10. Work Unit No. (TRAIS)
	11. Contract or Grant No.: BB-235
12. Sponsoring Agency Name and Address: Florida Department of Transportation, Research Management Center, 605 Suwannee Street, MS 30, Tallahassee, FL 32301-8064	13. Type of Report and Period Covered: Final: 7-9-97 to 3-31-02
	14. Sponsoring Agency Code

15. Supplementary Notes

Prepared in cooperation with the Federal Highway Administration

16. Abstract

Analytical models of traffic flow play an important role in the evaluation of all types of highway facilities. Accurate modeling of facility performance is essential to support decision-making related to planning, design and operation of freeways, arterials, intersections, etc. Most models are constructed on a framework of analytical and simulation concepts and have a very limited empirical basis. Lacking field data as "ground truth," it is very difficult to compare the results from different models and modeling approaches.

The Florida Department of Transportation has developed the LOSPLAN software, which provides a set of models for planning level analysis of various traffic facilities. All of the models are based on the level of service estimation procedures prescribed by the Highway Capacity Manual. The objective of the project described in this report was to develop tools and techniques for testing the LOSPLAN components.

Software tools were developed to create benchmark data sets for freeway systems, two-lane highways, multilane highways, and signalized arterials. Additional tools were developed to read and analyze the results from traffic models. The results were compared and observations were made on the similarities and differences between the software products that were tested. Observations were also made about internal relationships that were evident in specific models.

17. Key Words	18. Distribution Statement
Highway Capacity, Level of Service, Planning Models.	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA, 22161

19 Security Classif. (of this report)	20.Security Classif. (of this page)	21.No. of Pages	22 Price
Unclassified	Unclassified	135	

Acknowledgements

The research reported herein was carried out with support from the Florida Department of Transportation, and the US Department of Transportation. The work was performed by the University of Florida Transportation Research Center. Prof. Ken Courage coordinated the overall project effort. Significant technical support was provided by Ms. Nan Zhou and Dr. Morya Willis. The FDOT Technical Coordinator was Ms. Gina Bonyani.

Disclaimer

Development of Benchmark Data and Procedures
For Testing Traffic Operations Models

This report presents the results of a research project carried out by the University of Florida Transportation Research Center for the Florida Department of Transportation. The project includes several specific and somewhat independent research tasks determined by the Systems Planning office to be critical to its future efforts to promote uniform and defensible procedures for planning level assessment of performance on transportation facilities in Florida. Most of the tasks were completed previously, with the results presented in the following working papers:

1. Percent Turns from Exclusive Lanes: Preliminary Study Material
2. Car Following Model Description
3. Determination of the Proportion of Turns from Exclusive Lanes (PTXL) for Planning Purposes
4. Driver Comfort as a Level Of Service Criterion For Rural Freeways
5. Input Requirements for Benchmark Data Sets
6. Extension of the PTXL analysis to produce a recommended procedure for determining a global PTXL value for use in ART-TAB, analogous to the global value of g/C ratio
7. Traffic Model Markup Language (TMML) Data Dictionary for Signalized Intersections and Arterial Routes, with Sample Data Sets
8. An Improved Procedure for Determining an Equivalent Value of Cycle Length and g/C Ratio For Use by ART-PLAN in Evaluating Unsignalized Control on Arterial Streets
9. TMML Reader And Comparison Software: Program Specifications
10. Traffic Model Markup Language (TMML) Specifications
11. ARTPLAN User Interface Documentation
12. TMRC Program Documentation

This document presents the results of the final task, which represents the culmination of the project efforts to develop benchmark data sets for evaluation of the traffic models used by FDOT. This effort included the development of the Traffic Model Markup Language (TMML) for exchanging data between traffic models, and rendering their computational results. It also included the development of benchmark data set generators to create multiple data sets with parameters varied either randomly or systematically. Software utilities were developed to generate benchmark data sets and compare the results of traffic model computations using these data sets. Separate utility programs were developed for application to arterials, highways and freeways. The results of this task are presented in Working Paper 596-13.

CREATION AND APPLICATION OF BENCHMARK DATA FOR EVALUATION OF TRAFFIC MODELS

Working Paper 596-13

Prepared for
The Florida Department of Transportation
Systems Planning Office

By the University of Florida
Transportation Research Center

January 2002

TABLE OF CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

Model comparison and evaluation play an important role in traffic study today. Most models are constructed based on a framework of analytical and simulation concepts and have a very limited empirical basis. Benchmark data sets offer a more practical alternative to the problem of model comparison. With sufficient number of hypothetical data sets reflecting a wide range of conditions, it is possible to establish the similarities and differences between models, and to gain some insight into their merits and shortcomings. This paper will focus on the development and use of benchmark data sets for model evaluation purposes and making maximum use of XML in creating a common basis for traffic model evaluation.

1.1 PROBLEM STATEMENT

Empirical calibration of traffic models has been hindered by the difficulty (both effort and expense) of obtaining an adequate base of field data. Most models are constructed on a framework of analytical and simulation concepts and have a very limited empirical basis. Lacking field data as "ground truth," it is very difficult to compare the results from different models and modeling approaches.

Benchmark data sets offer a more practical alternative to the problem of model comparison. With sufficient numbers of hypothetical data sets reflecting a wide range of conditions, it is possible to establish the similarities and differences between models, and to gain some insight into their merits and shortcomings. So, as a pragmatic alternative to empirical validation and calibration, this paper will focus on the development and use of benchmark data sets for model evaluation purposes.

1.2 PROJECT OBJECTIVES

The Traffic Model Markup Language (TMML) was developed as a previous project activity. TMML is a fully compatible subset of the Extensible Markup Language (XML), which has evolved as a solidly entrenched information technology concept [1]. Several traffic model software products now use TMML for data storage and sharing. The goal of this study is to make maximum use of XML in creating a common basis for traffic model evaluation.

The specific project objectives are as follows:

1. To identify appropriate ranges of input data and operating parameters for testing models of freeway and surface street operations. Freeway facilities include basic freeway segments, merge/diverge areas and weaving sections. Surface streets include signalized intersections, signalized arterials and highways without any influence from traffic signals.
2. To generate a series of hypothetical data sets with randomized values of the input data elements in TMML format.
3. To identify two software products that model each of the facility types and to apply the benchmark data to identify their similarities and differences.
4. To formulate recommendations to guide future efforts in this area.

1.3 PROJECT TASKS

The following tasks were performed in support of these objectives:

1. The literature pertaining to TMML and to traffic modeling was reviewed to provide the basis for the remainder of the project activities. The results of this task are presented in Chapter 2;
2. Software tools were developed to create benchmark data sets for:
    - Freeway systems
    - Two-lane highways
    - Multilane highways
    - Signalized arterials
    - Signalized intersections

    The results of this task are presented in Chapter 3;
3. Software tools were developed to read and analyze the results from traffic models. The results of this task are also presented in Chapter 3
4. The results were compared and observations were made on the similarities and differences between the software products that were tested. Observations were also made about internal relationships that were evident in specific models. The results of this task are presented in Chapters 4, 5 and 6 for arterials, highways and freeways, respectively. The principal objective of each of these facility application chapters is to demonstrate the testing of the corresponding LOSPLAN component program as a planning level

implementation of the HCM procedures for level of service analysis on the specified facility.

5. Conclusions and recommendations were formulated and are presented in Chapter 7.

Three appendices are included in this report to provide a collection of "stand-alone" products produced during the course of this project. The complete Traffic Model Markup Language (TMML) specification is included as Appendix A. An addendum to the TMML specification covering the three LOSPLAN components described in this report is contained in Appendix B. Finally, Appendix C presents the full text of a paper developed by the project staff and submitted to the Transportation Research Board for presentation and publication. This paper has been accepted.

2 BACKGROUND DISCUSSION

This section presents the background and concepts required for an understanding of the details presented in the subsequent section. It begins with a discussion of traffic models in general, followed by the Highway Capacity Manual (HCM) procedures [1] as the traffic model upon which this project focuses. It then proceeds to the relevant software products that implement the HCM procedures, and concludes with an introduction to the Traffic Model Markup Language [2] developed as an activity of this project.

2.1 TRAFFIC MODELS

This section provides an overview of models used in traffic software related to highway capacity analysis and level of service [1].

Planners and engineers have different needs in regards to scope and level of detail. Planners focus on network performance in broad, general terms to understand interactions between supply and demand. Their main interest is at the level of land use impacts and transportation planning. On the other hand, engineers need to know how changes in the design of a specific facility or in the way it operates will affect its performance in terms of capacity, delays, queuing characteristics and other measures.

Compared to simulation models, planning models tend to focus on larger geographic areas with more links and nodes. The objective is to provide insights regarding network performance based on future traffic patterns and strategies for capacity enhancement and network improvement. Planning models represent traffic at a macroscopic level of detail and rely on equations to get relationships between flow parameters. Planning level analysis is characterized by the use of assumptions, approximations and default values to reduce the need for the detailed field data upon which operational level analyses are based. Figure 2-1 explains the differences among these perspectives and the models used to address the issues involved.

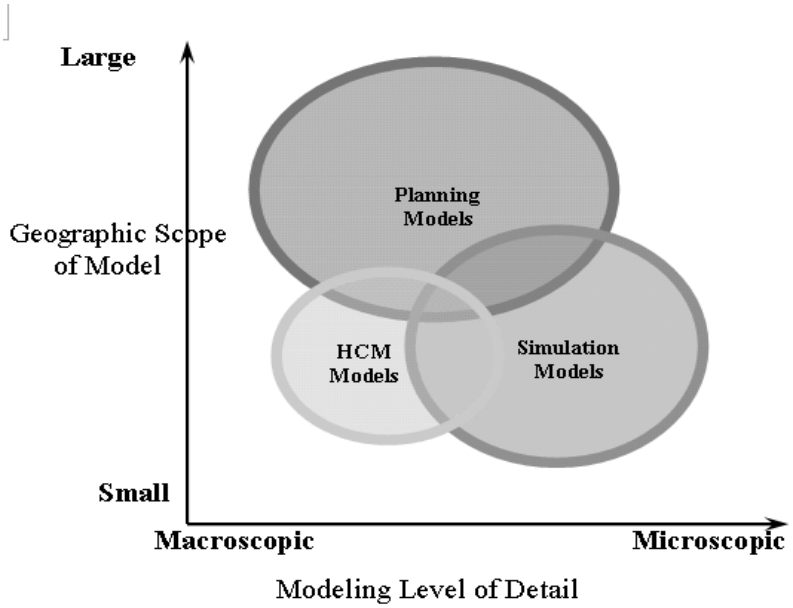


Figure 2-1. Traffic Model Categories

Traffic simulation models use numerical techniques on a digital computer to create a description of how traffic behaves over extended periods of time for a given transportation facility or system. As compared to empirical and analytical models, simulation models predict performance by stepping through time and across space, tracking events as the system state unfolds. Time can be continuous or discrete, and system state is a technical term that effectively describes the status or current condition of the system. Empirical models predict system performance on the basis of relationships developed through statistical analysis of field data, whereas analytical models express relationships among system components on the basis of theoretical considerations as tempered, validated, and calibrated by field data.

Traffic simulation models focus on the dynamic of traffic flow. They can represent a range of situations from a single facility to an entire network. Some implicit assumptions include interdependencies between the traffic objects, processing capabilities of the physical places and processing logic.

There are four fundamental descriptors commonly used in highway capacity-related simulation models: state variables, possible events, time-step logic and processing logic [1]. To describe a complete model, all four attributes must be specified, and in combination, they must represent a unified and consistent model.

2.2 THE HIGHWAY CAPACITY MANUAL PROCEDURES

HCM methodologies are somewhat in the middle of planning models and simulation models in both geographic scope and level of detail. Traffic demands are variable and time dependent, and the paths used by vehicles to traverse the network typically are sensitive to the capacity provided. HCM methodologies tend to focus on individual network elements: specific facilities or collections of facilities. Their intent is to assess the level of service provided by a particular facility with a given configuration and operational plan in response to the traffic flow accommodated. As a result, the geographic area being represented ranges from a single point to a small region. The HCM methods also represent traffic flows with variables that reflect the flow dynamics.

2.3 THE HIGHWAY CAPACITY SOFTWARE (HCS)

The HCS offers a full and faithful implementation of the HCM procedures. This software product has existed since the time of the 1985 edition of the HCM, which was the first edition that was developed with the recognition that the procedures would be implemented primarily by software. The current version of HCS implements the HCM 2000 edition. The HCS 2000 will be used in this project to represent the HCM procedures in comparison with other models and variations. Separate HCS Modules exist for each facility covered by the HCM. The modules of interest to this project include:

- Signalized intersections,
- Urban arterials,
- Two lane highways,
- Multilane highways,
- Basic freeway segments,
- Freeway ramps and
- Freeway weaving sections.

Results obtained from these modules will represent the performance indicators from the HCM.

2.4 THE LOSPLAN SOFTWARE

Highway capacity and quality of service can be viewed to exist at three planning levels: operational, conceptual and generalized [3]. Conceptual planning is applicable when there is a desire for a good estimate of a facility's LOS without doing detailed analyses. Unlike

generalized planning, when one evaluates an individual facility, the accuracy of analysis is more important than numerical consistency.

Planning level analysis is typically characterized by the use of assumptions, approximations and default values to reduce the need for the detailed field data upon which operational level analyses are based. FDOT has advanced the state of the art in planning level analysis by developing software that makes effective use of assumptions and approximations but also incorporates a unique structure that combines individual roadway elements into transportation facilities.

Collectively, the FDOT planning level software is grouped under the name LOSPLAN, which now includes three component programs: ARTPLAN, FREEPLAN, and HIGHPLAN, each of which implement the HCM analysis procedures for their respective facilities (arterials, freeways and highways). Each of these components will be described briefly in terms of its general operation and its specific relationship to this project. A more detailed description of LOSPLAN may be found in Reference [4].

2.4.1 ARTPLAN

ARTPLAN is the LOSPLAN component program that performs analysis on signalized arterial streets. The computations are based on the concepts contained in HCM [1] Chapter 10 and on the procedures prescribed by HCM Chapter 15. ARTPLAN performs a separate analysis for four different modes of travel, including automobiles, pedestrians, bicycles and buses. A maximum of nine segments, usually terminated by a signalized intersection may be included in each analysis. The automobile mode properties associated with each segment include:

- Segment length,
- Cycle length for the terminating signal
- g/C ratio for the terminating signal
- AADT
- Hourly volume
- Percent turns from exclusive lanes
- Arrival type
- Number of through lanes and
- Free-flow speed, that defaults to a value 5 mph greater than the posted speed.

The directional hourly volumes are computed from the AADT volumes using globally specified values of the K factor, D factor and peak-hour factor. The selected arterial class and area type determine default values for these factors. Other arterial inputs such as median type, existence of left turn lanes and arterial class are applied globally to the whole facility.

The screen display organization for the ARTPLAN automobile mode is illustrated in Figure 2-2. Separate data input/edit screens are provided for the overall facility data and for the segment-specific data. The results are also presented in two screens. The first displays the segment and arterial performance measures reflecting the currently entered data. The second displays the service volume tables for arterials with 1-4 through lanes in each direction. Note that the graphics presented in Figure 2-2 are intended to show the schematic organization of the screens, and are not legible at the level of detail required for a complete understanding of the data. Full size screen reproductions may be found in Reference [4].

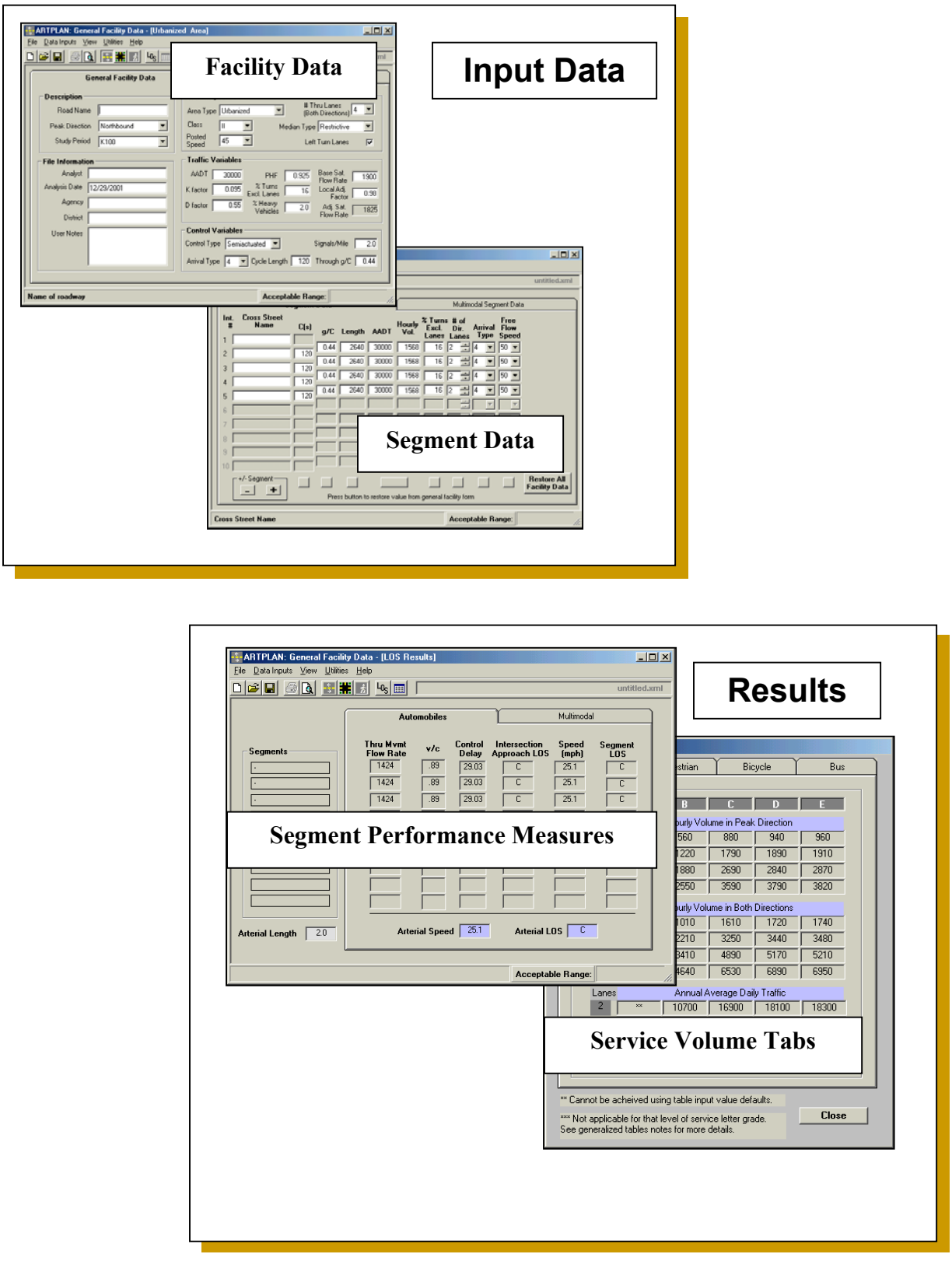


Figure 2-2. Screen Display Organization for ARTPLAN (Automobile Mode)

Supplemental screens are provided for the other modes of operation, as illustrated in Figure 2-3. The additional inputs requires for pedestrians, bicycles and buses include:

- Existence of a paved shoulder or bicycle lane,
- Outside lane width,
- Pavement Condition,
- Type of sidewalk/roadway separation,
- Existence of sidewalk/roadway protective barrier,
- Existence of obstacles to bus stop,
- Bus service frequency and
- Bus span of service.

Global default values entered on the multimodal facility data screen are transferred to the segment-specific multimodal screen, where each value may be edited to reflect the conditions on a specific segment. Because segments are often long and their properties are not always homogeneous for pedestrians (e.g., a sidewalk covering a portion of a segment) each segment may be divided into a maximum of three sub segments for pedestrians. Each sub segment may have different properties assigned.

The presentation of results is similar to the automobile mode. The multimodal segment results are presented on one screen, and the service volume tables are presented on separate screens for each mode. Service volumes are computed for the bicycle and pedestrian modes as a function of the number of through lanes. The transit level of service is a function of the bus service frequency and is independent of the number of through lanes on the facility.

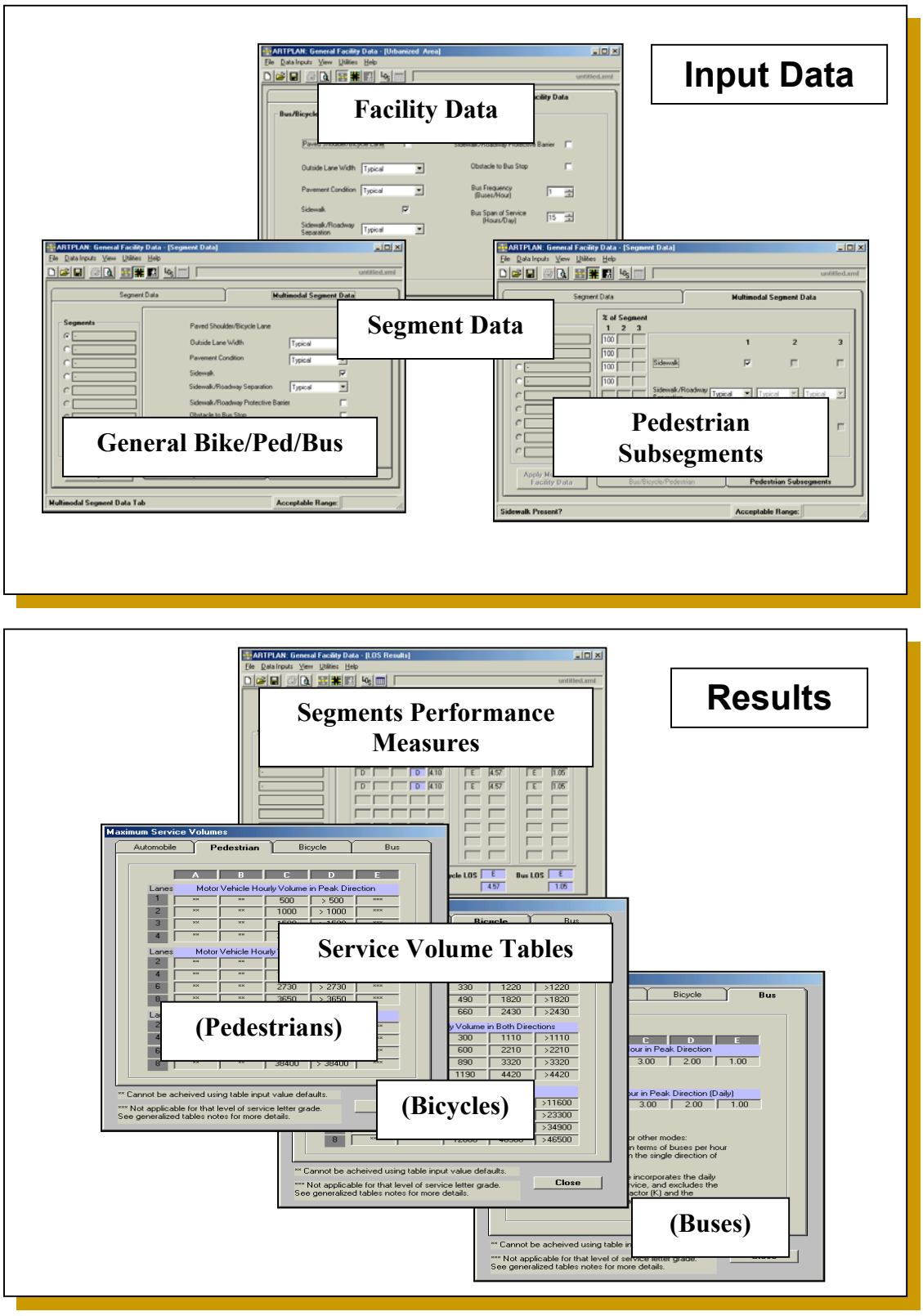


Figure 2-3. Screen Display Organization for ARTPLAN (Multimodal)

2.4.2 HIGHPLAN

HIGHPLAN is the LOSPLAN component program that performs analysis on two-lane and multilane highways. The computations are based on the concepts contained in HCM [1] Chapter 12 and on the procedures prescribed by HCM Chapter 20 for two-lane highways and 21 for multilane highways. HIGHPLAN is much simpler in concept than ARTPLAN because it deals only with the automobile mode and it does not break the facility into segments.

The simplicity of HIGHPLAN is evident in the screen display organization illustrated in Figure 2-4. HIGHPLAN has only two screens, one for facility data and LOS results, and another for the service volume tables. The input data include roadway and traffic variables, which are essentially the same as ARTPLAN to the extent that they apply (e.g., none of the signal operating parameters apply to open highways). The type of terrain (level or rolling) is required for both two-lane and multilane highways. Information on exclusive passing lanes and no passing zones is also required for two-lane highways:

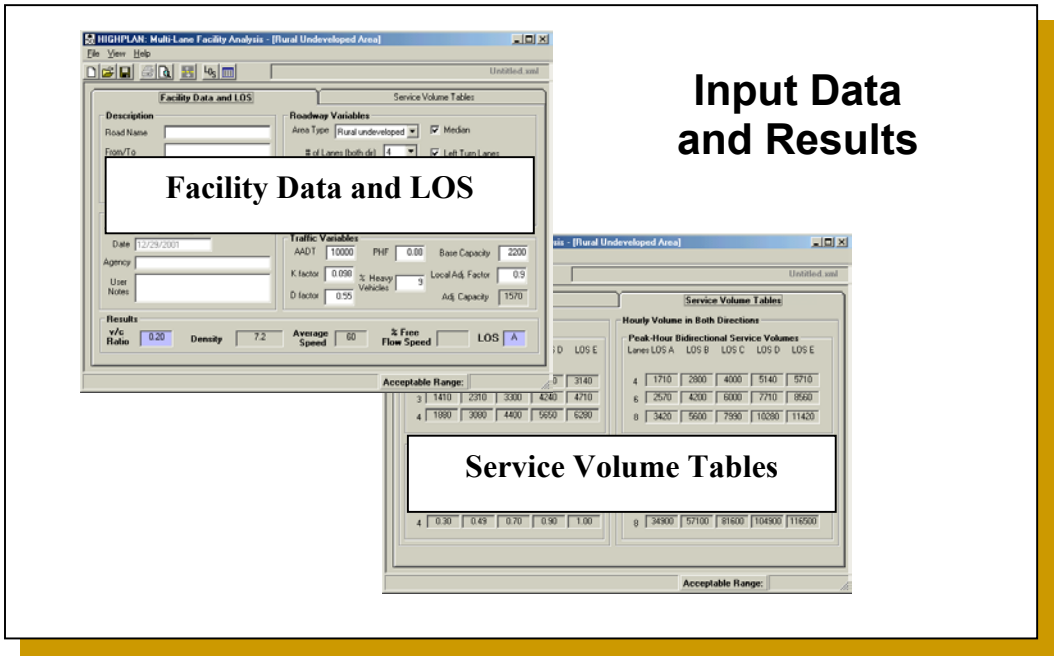


Figure 2-4. Screen Display Organization for HIGHPLAN

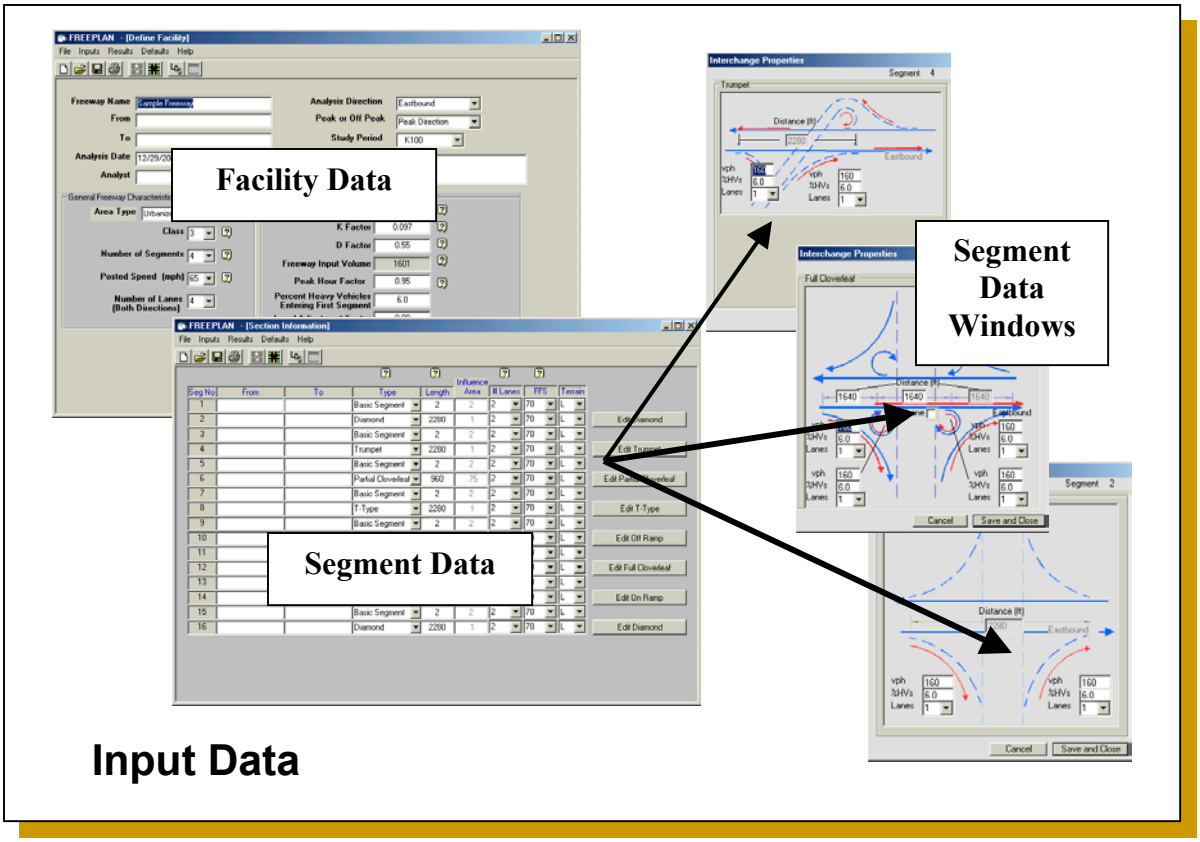
2.4.3 FREEPLAN

FREEPLAN is the LOSPLAN component program that performs analysis on freeway facilities. The computations are based on the concepts contained in HCM [1] Chapter 13 and on the procedures prescribed by HCM Chapter 23, 24 and 25 for basic freeway segments, freeway weaving and ramp operations, respectively. FREEPLAN deals only with the automobile mode, and is therefore able to avoid the complications of multimodal inputs, computations and results. It accommodates a maximum of 20 segments in each analysis.

Like ARTPLAN, the FREEPLAN input data are organized into separate screens for facility data and segment-specific data. The facility data are essentially the same as ARTPLAN, but the segment specific data are substantially different. The screen display organization for FREEPLAN is illustrated in Figure 2-5. Each FREEPLAN segment may be assigned to one of the following types:

- Basic freeway segment
- Various types of interchange
- Partial or full cloverleaf
- On ramp
- Off ramp

Each of the segment types has its own special data entry window because different segment types have slightly different data requirements. The display of results follows the conventional LOSPLAN scheme involving one screen for the segment and overall LOS results and a separate screen for the service volume tables.



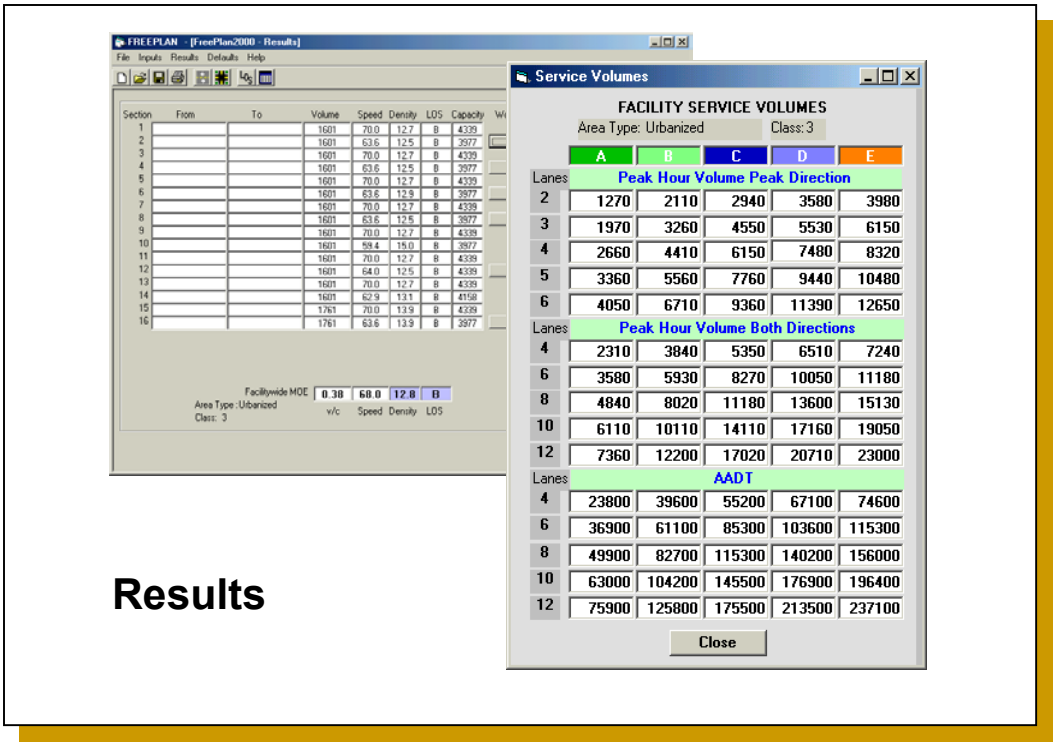


Figure 2-5. Screen Display Organization for FREEPLAN

2.5 THE ROLE OF XML AND TMML IN DATA EXCHANGE

A wide range of traffic control system models are used by engineers and planners throughout the world. Although most models deal with the same or at least more or less similar input data, they are unable to share information or communicate with each other because of different data formats. This problem costs users a lot of effort in dealing with various formats of input information as well as analysis itself. The Extensible Markup Language (XML) is catching on as a means of transferring data between two systems or users who deal with the same data, but in different formats [5]. This research, including data generation, operation, and comparison, is based on the XML format data storage and interchange. A brief review of some basic concepts of XML language naturally becomes a starting point.

For those who are not familiar with the term, XML (eXtensible Markup Language) is a method of encoding data in a text file where each data item is identified with a tag. It offers a method of producing "self-describing data" which, by definition, is free of arbitrary formatting constraints and proprietary controls.

Compared with HTML (Hyper Text Markup Language), XML provides developers with the ability to create and manipulate their own tags and create pages that are as elegantly presented as they are structured [6]. The "X" in many computer-oriented acronyms denotes "extended." In this case, it denotes "extensible," which differs from "extended" in the sense that you have to provide your own extensions. For example, a tag called <Volume> would be considered foreign to an HTML document. It would be quite acceptable in XML, but would not be useful unless its significance had been previously established [2].

There have been specific vocabularies developed for statistics, mathematics, chemistry, and many other disciplines as well as for traffic control system modeling. The TMML (Traffic Model Markup Language), a branch of XML in the area of traffic control system modeling has been developed to facilitate sharing of data among traffic modeling software products. TMML is a fully XML-compatible markup language prescribing the class structure and data element tag names required to represent traffic model data in a "self-describing" format [2]. The principal applications of TMML include exchanging data between traffic model software products and facilitating the compilation and presentation of results.

Most traffic models deal with similar input and output data. For most software products, it should be possible to import or export a large part of any data set with minimal processing logic.

On the other hand, each software product has unique definitions and structures for representing the data. Therefore, it is not possible to create a universally understood TMML specification that would accommodate all software products directly. Each product that offers TMML connectivity will require a programmer interface document, which identifies the data elements that are recognized and any conditions that apply to their interpretation.

TMML not only found its way in storing, saving and exchanging data for specific modeling problems, but also in presenting data in an embellished tabular format and comparing the results from different data sets. This additional benefit of using TMML (XML based) data format is called the extensible style sheet. EXtensible Style sheet Language (XSL) can be used to transform the plain text format of XML into custom report formats. The capabilities of a Web browser can then be exploited to handle the display and printing duties. Furthermore, users that prefer to develop their own interface for the input data can interface directly with the computational engine of an XML based data format application.

The TMML language is defined in terms of a collection of data structures that describe the properties of the objects associated with traffic carrying facilities. TMML provides the structure and vocabulary to completely define a data set for various facilities and software products [2]. It has been applied extensively to software products that analyze the performance of several facility types, including freeways, urban and rural highways, and signalized arterials. Many of the software products implement various chapters of the Highway Capacity Manual (HCM) (TRB, 2000) [1].

TMML is fully described in the Traffic Model Markup Language Specification [2], which includes a detailed specification for the structure and vocabulary for all classes of data. The specification includes a list of recognized abbreviations intended to reduce the size of the XML tags that describe the data elements. TMML was created following the general principles of style described in IEEE Standard 1489 (IEEE, 2000) [2].

It is important that each model's definition of every data item be understood in the data transfer process. The rules to identify and assign specific tags to data items that are commonly used by traffic models are:

1. TMML is a fully XML compatible markup language intended for transferring data between traffic model applications and for creating output data in a format that is easily rendered by office productivity products.

2. All TMML data files shall have the XML file name extension, so that they will be recognized by common software applications. A prologue containing XML processing instructions and data type declarations shall precede the root tag. The root tag shall be:

<TMML Facility = "*FacilityType*">

3. The TMML language is defined in terms of a collection of document type definition (DTD) files that describe the structure and vocabulary to completely define a data set for various facilities and software products.

4. The order of presentation of classes or elements within a class shall not be prescribed for a specific software product. It must be possible to present the classes and elements within a class in any order as long as the class structure is maintained. The same class may appear more than once in a file (e.g. to separate input and output data). A given data item may appear only once in any class otherwise an irresolvable ambiguity will be created.

The following are some examples of accepted formats for tags.

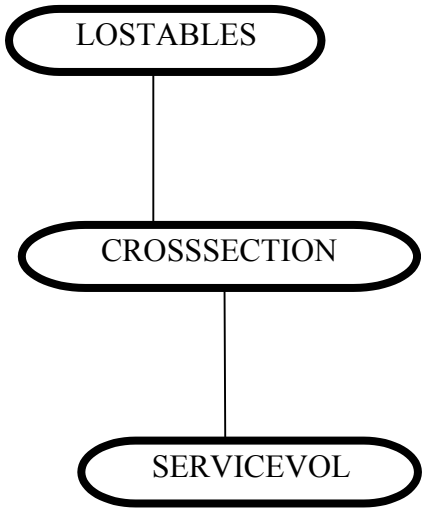
```
<TMML Facility="TwoLane">
<HIGHWAY>
        <AreaType>Rural undeveloped</AreaType>
        <Class_HCM>1</Class_HCM>
</HIGHWAY>
</TMML>
```

TMML is intended to be open-ended with respect to its design to encourage a wide range of software development for using and exchanging data between traffic model software products. However, each piece of TMML-compliant software will have its own limitations with respect to the number of intersections, approaches, movements, etc. that it will accommodate and the range of elements and tags that it will recognize. Specific software products may also impose rules regarding structure and interpretation of the various tags. Missing data elements are treated as null values; numerical data will be set to zero and null strings will be set for character data. Class or data element tags that are not recognized by a specific program are ignored.

Data elements will apply to the class in which they are placed. For example, The <FreeFlowSpeed> tag in the ARTERIAL class would logically be interpreted as a default value applied to all movements on all approaches at all intersections. A different Free Flow Speed value placed in a lower class would logically override the default value from the higher level.

Structure for TMML arterial files shall conform to the hierarchy shown in Table 2-1. This structure provides the framework for present and future traffic model software.

Table 2-1. TMML Structure for the LOSTABLES Class



Class Attribute	Values
Mode (Arterials only)	Auto, Bicycle, Pedestrian or Transit
Lanes	1 to 6 (each direction)
LOS	A to E

3 RESEARCH METHODOLOGY

3.1 SOFTWARE TOOLS

In order to deal with a large number of hypothetical data sets feasibly and efficiently, several software programs have been developed to support the comparison process; and TMML serves as a general database in the communication among these programs. For this research, the BDS (Benchmark Dataset Generator) was developed to generate XML format based data sets. The TMRC (Traffic Model Results Comparator) will be used to compare two separate output XML files intelligently. To display the methodology used in this research, the LOSPLAN Package (from the Florida Department of Transportation) is used as the first model and the HCS is used as the second model.

3.1.1 Benchmark Dataset Generators

Being the main product, as well as a useful tool of this research, the BDS (Benchmark Dataset Generator) has been developed through the use of visual basic language and supported by XML technique. The BDS is composed of three parts designed to serve different facilities: APBDS (ARTPLAN Benchmark Data Sets generator), HPBDS (HIGHPLAN Benchmark Data Sets generator) and FPBDS (FREEPLAN Benchmark Data Sets generator). Their major functions are summarized in Table 3-1.

Table 3-1. BDS Components

BDS	Corresponding Program	XML File Facilities Type	Naming Rules (## is a sequentially assigned number)
APBDS	ARTPLAN	Single Signalized Intersection Arterial	AP_BDS##.XML
HPBDS	HIGHPLAN	Multilane Roadway Two-lane Roadway	HP_BDS##.XML
FPBDS	FREEPLAN	Basic Freeway Section Merging or Diverging Section Weaving Section Toll Plaza	FP_BDS##.XML

These three components have a similar user interface and logic for generating data sets. Their uniform features will be described in the following paragraphs in addition to the detail information that is a unique aspect of each program.

The core arithmetic of the BDS program is to generate a list of random numbers within their corresponding valid rages and according to specific distribution rules. The output data sets would be saved in an XML format.

The random numbers are generated by the Visual Basic random-number generator (RND) function, which gives a number uniformly distributed between 0.0 and 1.0 each time it is called. The RND function uses the same initial seed number initially, and thereafter uses the last generated number as a seed value for the next number. To avoid getting repeating random values, a RANDOMIZE statement may used before the first RND function to get a return value from the TIMER function as an initial new seed value.

To get a value for a particular parameter within its valid range of the corresponding traffic model, equations need to be used. For example, AADT (Average Annual Daily Traffic) is uniformly distributed between 5000 and 50000. The random AADT formula would be:

AADT = 5000 + Int(45000 * RND( ) + 0.5)

(INT function is used to delete the decimal fraction.)

Some parameters, such as the K Factor, are not uniformly distributed within their valid range. It is more likely that they follow the normal or Poisson distribution in the real situation. In this case a sectional uniform distribution is used to approximate the real situation. Consequently, two random numbers are needed to generate each value with the first one used to decide in which section it falls. The following is an example showing the approximation of the normal distribution of K factor:

20% uniform distributed between 0.06 and 0.09

60% uniform distributed between 0.09 and 0.10

20% uniform distributed between 0.10 and 0.15

Two random numbers generated by RND function are denoted as R1 and R2;

If R1 is less than 0.2, K factor will fall into the first section and

KFactor = 0.06 + Int(30 * R2 + 0.5) * 0.001

If R1 is greater than 0.2 and less than 0.8, K factor will fall into the second section and

KFactor = 0.09 + Int(10 * R2 + 0.5) * 0.001

If R1 is greater than 0.8, K factor will fall into the third section and

KFactor = 0.1 + Int(50 * R2 + 0.5) * 0.001

To make the program more powerful, parameters can be designated as randomized or fixed by the user. If a variable is specified as a fixed value, an input box will be displayed and a particular value can be input by user.

Some detailed information about user's interface, the valid range and distribution of all the parameters in these programs are described in Chapter 4, 5, and 6 by facility type.

3.1.2 TMRC

The TMRC (Traffic Model Results Comparison) program is another important tool used in this research. It was developed to compare the contents of two TMML files for the same facility type.

One of the files will represent a standard reference against which the other file will be tested. TMRC is useful for comparing the results from two different software products or different versions of the same product applied to the same input data. It is also used in testing the sensitivity of the results to variations in the input data. Each TMRC run appends a line to a text file, which may be imported into a spreadsheet for analysis and plotting.

The TMRC user interface is shown in Figure 3-1. The type of facility to be used needs to be selected in order to begin a comparison run. The output in the Comparison Level box can be designated as "All Items" to show a complete result or "Differences Only" to minimize the output size. Any difference less than the given tolerance would be overlooked by the program and would not be included in a "Difference Only" output.

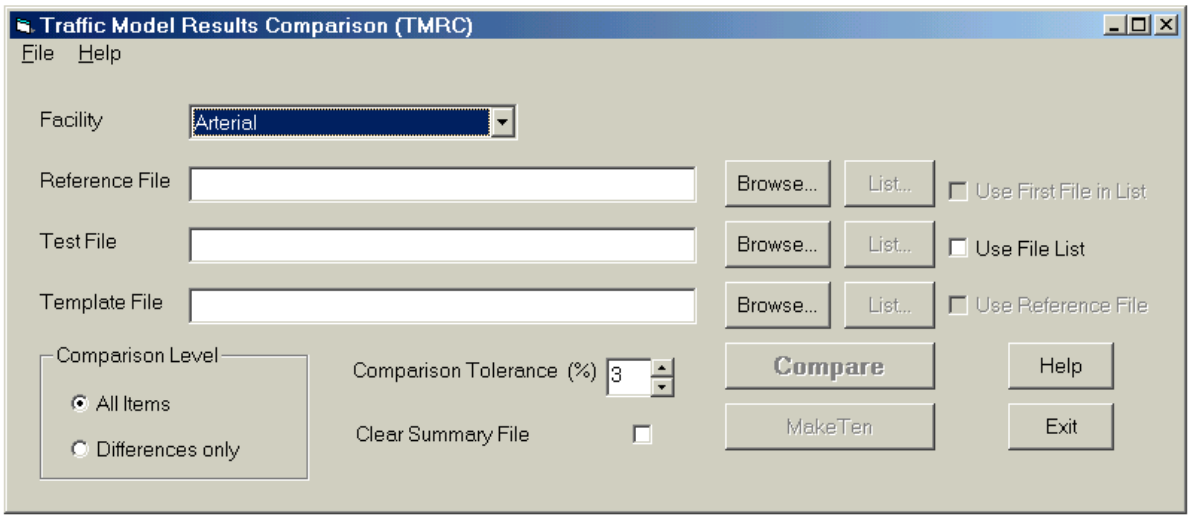


Figure 3-1. Traffic Model Results Comparison Main Menu Screen

Usually a template file is needed to provide more information and structure for the comparison. The template is a TMML file with a structure that should be identical to the files being compared. In the template you can specify the elements you want to focus on, to look for differences between the results of two different runs, either of the same model or of two different models. If the reference file is chosen as a template file, all of the reference file's data elements that appear in the test file will be compared.

Normally, the actual values contained in the data elements in a template are ignored, but the program can read some special instructions if added in those places.

The labels in the output file for corresponding parameters can be specified at that point followed by an exclamation point (!). The default labels would be the corresponding tag names with a space before each upper case letter.

For example, a template file tag named <BaseCapPerLn> would be labeled as "Base Cap Per Ln" by default in the comparison output. It could be converted to "Base Capacity" if the information between the tags were replaced as follows:

<BaseCapPerLn>**Base Capacity!**<BaseCapPerLn>

TMRC also facilitates the comparison of multiple runs by placing information in a separate text file for importing into a spreadsheet. When doing multiple comparisons, the test file should be a file list (a text file contains a list of paths to XML files) instead of a single XML file. To store the output in a spreadsheet format, specific data element values in the template file need to be replaced with the code: **@nhhhhhhhhh**, where

- **@** indicates that this particular field should be included in the spreadsheet;
- **n** (a number between 1 and 99) specifies the spreadsheet column in which the data will be placed;
- **hhhhhhhhh** specifies the column heading;

For example, you want to compare capacity values in 10 XML files and place the values in the first column of a spreadsheet with "CAPACITY" as the heading. The following information should be added in the template:

<Capacity>**@1CAPACITY**<Capacity>

Another important function of TMRC is to create 10 data sets based on one XML file, with a systematic variation in some of the data elements. The rules for varying the data element values are contained in the special markup instructions that must appear immediately after the

line that contains the tag for the data element to be varied. An XML comment line in the following format will be recognized as a Make 10 instruction by the program:

<!-- MAKE10 @ n "###" --> where

- n is a single digit between 0 and 9, indicating the number of specific data set in which the value is to appear and
- ### is the value to insert in place of the value given in the original data set.

The original data set will be considered as number zero, and data set numbers 1 through 9 will be created from the MAKE10 instruction. If the value of n is zero, the number (###) will be taken as an increment for subsequent data sets instead of an absolute number.

This feature is explained in the following example:

Assume that the traffic volume in the original data set is 1000 vph, and you want to create nine more data sets with the volume incremented by 100vph in each file: The value of 1000 vph would be specified in the original data set by the XML element line:

<Volume>1000<Volume>. By inserting the following line: **<!-- MAKE10 @ 0 "100" -->**, nine more data sets will be created with the volume incremented as specified. If an unequal increment were desired, it would be necessary to use nine MAKE10 instructions with the absolute value specified in each instruction.

This function is very useful when testing the sensitivity of the results to variations in each parameter. Detailed examples of multiple file comparisons will be provided in later sections of this report.

3.2 TESTED MODELS

LOSPLAN package and the HSC, the first and second models respectively, are used in this research as good samples to explain the methodology of comparing and evaluating two models using BDS and TMRC. Both of these models use the TMML format for data storage and interchange. This facilitates direct result comparison to the corresponding parts that deal with arterial, highway or freeway analysis in the other model.

LOSPLAN is the FDOT's planning level software package, which includes the component software programs ARTPLAN, HIGHPLAN, and FREEPLAN [7]. Due to some unique characteristics and philosophical differences of opinion by the FDOT, LOSPLAN incorporates a number of concepts and calculations that differ significantly from the basic procedures in the HCM. Initial defaults are given, which reflect the most common conditions and facilities that are

encountered in the State of Florida. Values specified by the user can be substituted for all of the defaulted items. A general introduction of LOSPLAN's features are covered in Chapter 2.4

3.3 OVERALL PROCEDURE

Basically the procedure can be divided into two parts: basis establishment and the sample test. The first part is to establish a model comparison and evaluation system that includes a series of hypothetical data generators and a result comparator. In the second part, two models are identified and their similarities and differences are tested.

In the basis establishment segment, appropriate ranges of input data and operating parameters are determined and the software tools including BDS (Benchmark Data Set generator) and TMRC (Traffic Model Results Comparison) are developed to support the overall research.

BDS consists three components: APBDS for signalized arterials and signalized intersections; HPBDS for highway facilities including two-lane and multi-lane roadways; and FPBDS for freeway facilities including basic freeway segments, merge/diverge areas and weaving sections.

The sample test portion is made up of two types of tests: a performance test and a sensitivity test. The first portion tests the overall performance of the models while the second compares the sensitivity of the results to each parameter. LOSPLAN and HCS were chosen as the two models in this research. They both support the XML format and incorporate models of each of the facility types mentioned above so that the entire system can be tested using this example. Additionally, both examples are deterministic models. An examination of their internal calculations will allow for the comparison results to be checked. In the performance test, the BDS data generator generated numbers of random data sets. These data sets would be run by corresponding LOSPLAN and HCS components. The output LOS, density or average travel speed will be compared by TMRC and a rough conclusion can be made intuitively from the results. In the sensitivity test, a more in-depth investigation of the data sets is needed. More empirical data sets with partial fixed values need to be generated based on those data sets. The variation rate in results will be tested according to the change of one particular parameter.

Figure 3-2 illustrates the comparison methodology:

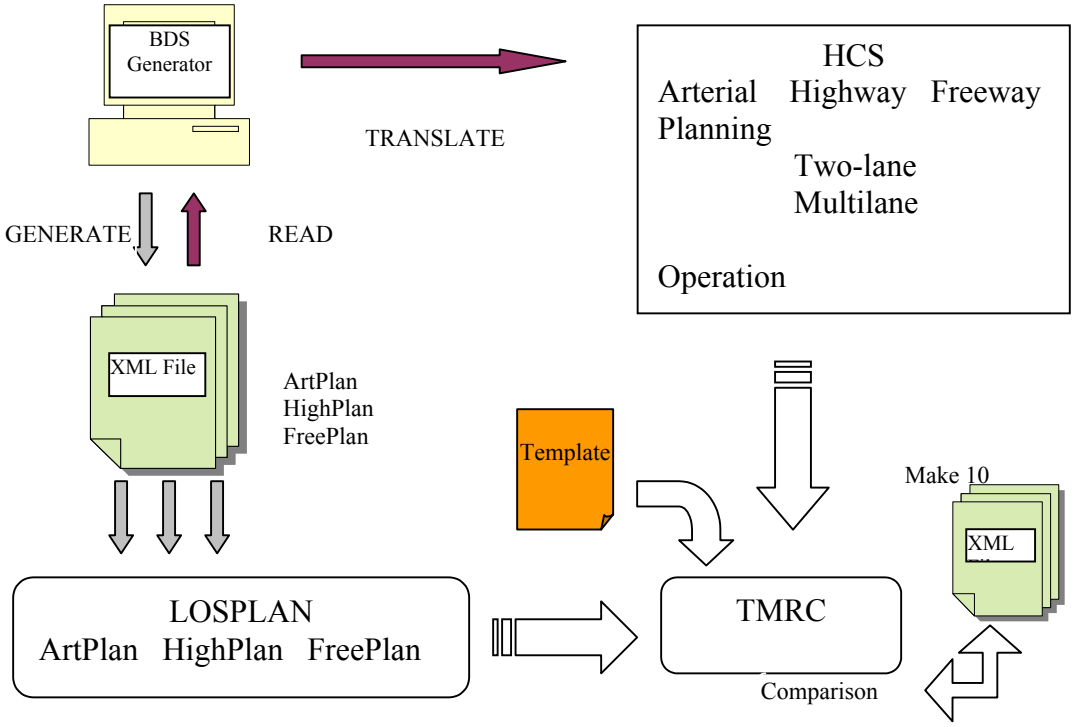


Figure 3-2. Overall Comparison Procedures

4 APPLICATION TO ARTERIAL FACILITIES

This chapter describes the application of the benchmark data generation and analysis methodology presented in Chapter 3 to signalized urban arterials. The principal objective of this part of the study is to demonstrate the testing of ARTPLAN as a planning level implementation of the HCM procedures for arterial level of service analysis. The main topics include the development of a software tool for generating Arterial Benchmark Data Sets, and the performance and sensitivity tests that were carried out using the benchmark data.

4.1 TMML STRUCTURE FOR ARTERIALS

The TMML class structure for arterial facilities [2] modeled by ARTPLAN is depicted in Figure 4-1. The individual data elements within each class are identified in Appendices A and B.

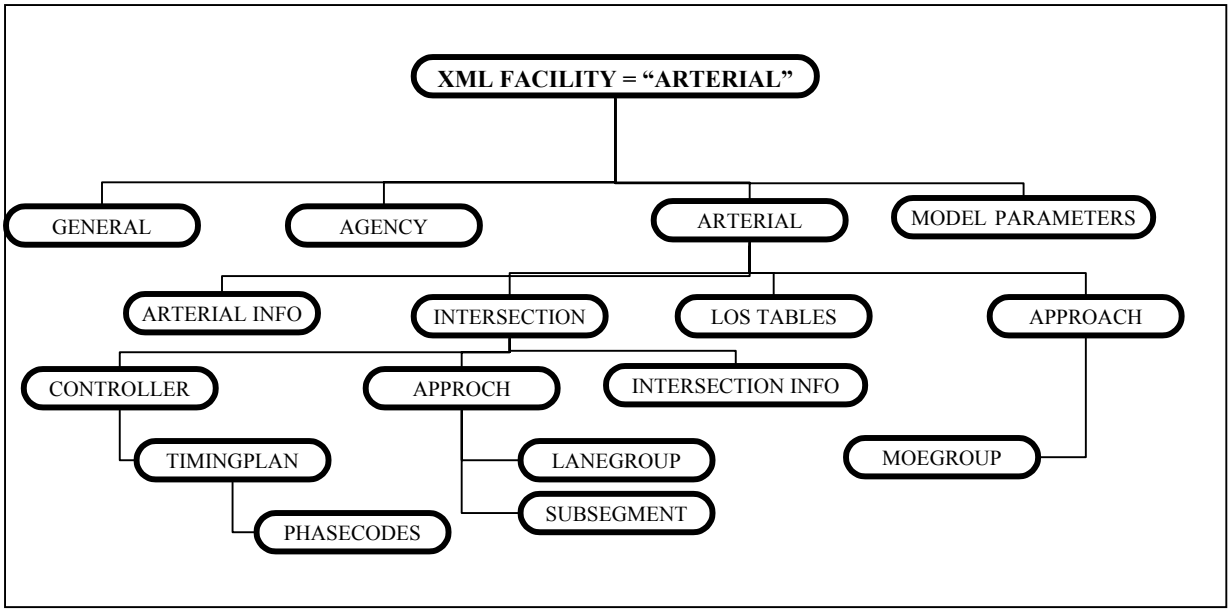


Figure 4-1. TMML Structure for Arterials

4.2 THE ARTERIAL BENCHMARK DATA SET GENERATOR (APBDS)

Figure 4-2 shows the user's interface of APBDS. Up to 200 data sets can be generated each run; and data files are named as the prefix followed by a number. Before data generation process, some parameters in the data sets need to be assigned by user. Click on the Assign Properties button, parameters including: Number of Intersections, Area Type and Control Type will be show in the Properties Assignment Table (Figure 4-3).

Clicking on the Select Variables button will open the Variable Selection Table showing in Figure 4-4. The system will generate arbitrary numbers for the parameters that are selected as random. If a parameter is left in the fixed mode (the default value), the system will automatically provide a fixed value in a text box. The user can edit this value and the corresponding valid range will be displayed in the lower right corner of the screen when the text box is chosen. The parameters shown on the left side of the Variable Selection Screen (such as volume, g/C Ratio, Posted Speed) can be randomized individually. To make the hypothetical data more close to the real situation, only one value generated for each facility parameter and random adjustment factors are created for segments. So the segment values, though randomly generated, are still consistent with one another in the whole facility.

If Volume is selected as a random variable in APBDS, the program will generate arbitrary numbers for AADT, KFactor and DFactor and the product represents the random value of volume. So the valid range for volume is between 150 and 7500 vphpl according to the range of the three parameters in ARTPLAN.

The other function of APBDS is to read in an XML file in ARTPLAN structure and transfer it to the HCS- Multilane or Two-lane structure. Multiple files can be translated at the same time and the letters "_hcs" will be added to the original file name to denote it as a translated file.

Before the data set generation process begins, several of the parameters in the sets need to be assigned by the user. These parameters include: Number of Intersections, Area Type and Control Type.

- The parameters located on the left side of the Variable Selection Screen, (such as volume, g/C Ratio, Posted Speed, etc.) can be selected as randomized by either facility or segment. To make the hypothetical data relate more closely to real situations, only one value is generated for each facility parameter and random adjustment factors are created for each segment. So the segment values, though randomly generated, are still consistent with one another in the whole facility.
- If Volume is selected as a random variable in APBDS, the program will generate random numbers for AADT, KFactor and DFactor and the product represents the random value of volume. So the valid range for volume is between 150 and 7500 vphpl according to the range of the three parameters in ARTPLAN.

The following steps are involved in creating and analyzing the Benchmark Data Sets:

1. Identify appropriate ranges and distributions of input data and operating parameters for testing signalized arterial or intersection operations. Table 4-1 shows the valid range and distributions of each parameter in the data sets. The parameters in ARTPLAN are divided into four groups: fixed, assigned by user, randomized by arterial, and randomized by intersections.

Table 4-1. Valid Range and Distribution in APBDS

A. Fixed Value

PARAMETERS	RANGE	FIXED VALUE
Road name		"BDS Road"
Peak Direction	N, S, E, W	"Northbound"
Study Period	K30, K100, K5-6, Kp/d, Kother	"K100"
Sidewalk/Roadway Separation	Adjacent, Typical, Wide	Typical
Sidewalk/Roadway Protective Barrier	True, False	False
Obstacle to Bus Stop	True, False	False
Bus Frequency (Buses/hour)	0 ~ 12	5
Bus Span of Service (hours /day)	0 ~ 24	15

B. User assigned parameters:

PARAMETERS	RANGE
Area type	Urbanized, Transitioning/Urban, Rural developed
Control Type	Pretimed, Semi-actuated, Actuated
Number of Intersections	1 ~ 10

C. Randomized by Arterial:

PARAMETERS	RANGE	DISTRIBUTION
Class	1, 44~56 2, 34~44 3, 31~35 4, 25~31	Urbanized 3: 50% 4: 50% Transitioning/Urban 2: 60% 3: 20 % 4: 20% Rural developed 1: 40% 2: 60%
Median type	None Nonrestrictive Restrictive	Uniform Distributed
Left turn lanes	True, False	True False Urbanized: 50% 50% Transitioning/Urban: 80% 20% Rural developed: 100%
K Factor	0.06 ~ 0.15	20% uniform distributed 0.06 to 0.09 60% uniform distributed 0.09 to 0.10 20% uniform distributed 0.10 to 0.15
D Factor	0.5 ~ 1	75% uniform distributed 0.5 to 0.6 25% uniform distributed 0.6 to 1.0
PHF	0.75 ~ 1	25% uniform distributed 0.75 to 0.85 75% uniform distributed 0.85 to 1.0
% Heavy Veh	0 ~ 25	75% uniform distributed 0 to 10 25% uniform distributed 10 to 25
Base Sat Flow Rate	1400 ~ 2000	25% uniform distributed 1400 to 1700 65% uniform distributed 1700 to 1900 10% uniform distributed 1900 to 2000
Local Adj. Factor	0.75 ~ 1	Fixed = 1.0
Paved Shoulder/bike Lane	True, False	Urbanized: 90% True Transitioning/Urban: 70% True Rural developed: 30% True
Outside Lane Width	Narrow, Typical, Wide, Specify Width	Narrow 25% Typical 50% Wide 25%
Pavement Condition	Undesirable, Typical, Desirable	Undesirable 25% Typical 50% Desirable 25%
Sidewalk	True, False	Urbanized: 80% True Transitioning/Urban: 50% True Rural developed: 30% True

D. Randomized by intersection:

PARAMETERS	RANGE	DISTRIBUTION	BY INTERSECTION
Posted speed	30, 35, 40, 45	Class 1: 45 Class 2: 40 80%; 35 20% Class 3: 35 80%; 30 20% Class 4: 30	Arterial value adds a number randomly chosen from -10, -5, 0, 5, 10
Through lanes (both direction)	2, 4, 6, 8	2 4 6 8 Urbanized: 10% 40% 40% 10% Transitioning/Urban: 20% 60% 20% Rural developed: 50% 40% 10%	Arterial value adds a number randomly chosen from -1, 0 ,+1 in each direction
Cycle Length	60 ~200	10% uniform distributed 60 to 70 75% uniform distributed 70 to 120 15% uniform distributed 120 to 200	Arterial Value times X X ~ (0.75, 1.25) uniform distributed
Signals/Mile	0.5 ~ 8	Urbanized: 6.4 ~ 8 Transitioning/Urban: 3.2 ~ 8 Rural developed: 0.5 ~ 4.8	Arterial Value times X X ~ (0.75, 1.25) uniform distributed
Through g/C	0.2 ~ 0.7	0.2 ~ 0.4 10% 0.4 ~ 0.55 80% 0.55 ~ 0.7 10%	Arterial value adds X X ~ (-0.1,0.1), Uniform distributed
AADT	5000 ~ 50000	Uniform 5000 to 50000 per lane (integer)	(Volume including AADT , K, D) Arterial Value times X X ~ (0.75, 1.25) uniform distributed
% Turns Excl Lane	0 ~ 45	75% uniform distributed 0 to 20 25% uniform distributed 20 to 45	Arterial Value times X X ~ (0.75, 1.25) uniform distributed
Arrival Type	1, 2, 3, 4, 5, 6	1 10% 2 20% 3 40% 4 15% 5 10% 6 5%	No Rule

2. APBDS was developed in Visual Basic to create XML formatted data sets. The features of BDS programs are described in Chapter 3.1 Software Tools.
3. In step 3, the XML data sets created in step 1 are translated to accommodate the format of the HCS-ARTERIAL (Planning Level). The translation rules are mapped into APBDS and the translated file will be designated as the original file name followed by "_hcs".
4. The files are processed through ARTPLAN and the HCS-ARTERIAL respectively in step 4 and the results saved in XML format.
5. In step 5, multiple comparisons are done to the file list according to the template designed in the instructions discussed earlier in Chapter 3.1 Software Tools, and the output is saved in a spreadsheet format text file.
6. Step 6, analyze the comparison output and derive conclusions.

Sensitivity Test Part:

7. Set all parameters to reasonable fixed values by observing the output of the performance test. Save the file as a basic XML file.
8. Open the basic XML file in a text editor. Generate data sets based on single parameter changes (10 versions for each parameter.) Save the data sets by groups.

   The test parameters with their respective ranges are itemized as follows:

   - AADT: distributed between 5000 and 50000, a range starts from 5000 to over saturated (v/c>1) which is 37500 in this data set are tested;
   - Free Flow Speed: ARTPLAN ranges from 25 to 60 by increments of 5; and HCS ranges from 25 to 55 with different class levels;
   - Number of Lanes: 1 to 5 in one direction are tested;
   - Signal Spacing: distributed between 660 feet, which is 8 signals per mile, and 10560 feet, which is 0.5 signals per mile;
   - Cycle Length: 10 second increments starting at 60 seconds to going to 200 seconds;
   - g/C Ratio; from 0.2 to 0.7
9. Translate these files to the HCS-Multilane format as shown in step 3;
10. Create new templates that include a single test parameter and Delay, Average Travel Speed.
11. Do multiple comparisons by groups and save the results to a spreadsheet.
12. Analyze the output and derive conclusions.

4.3 AN ARTPLAN BDS EXAMPLE

The following step-by-step detailed example will illustrate how to apply benchmark data to test the average travel speed-signal spacing relationship of arterials and compare the ARTPLAN results with HCS-Arterial results.

1. Run APBDS from Windows. The APBDS Main Menu Screen will appear (see Figure 4-2). Click on the button to open the Assign Properties Table.

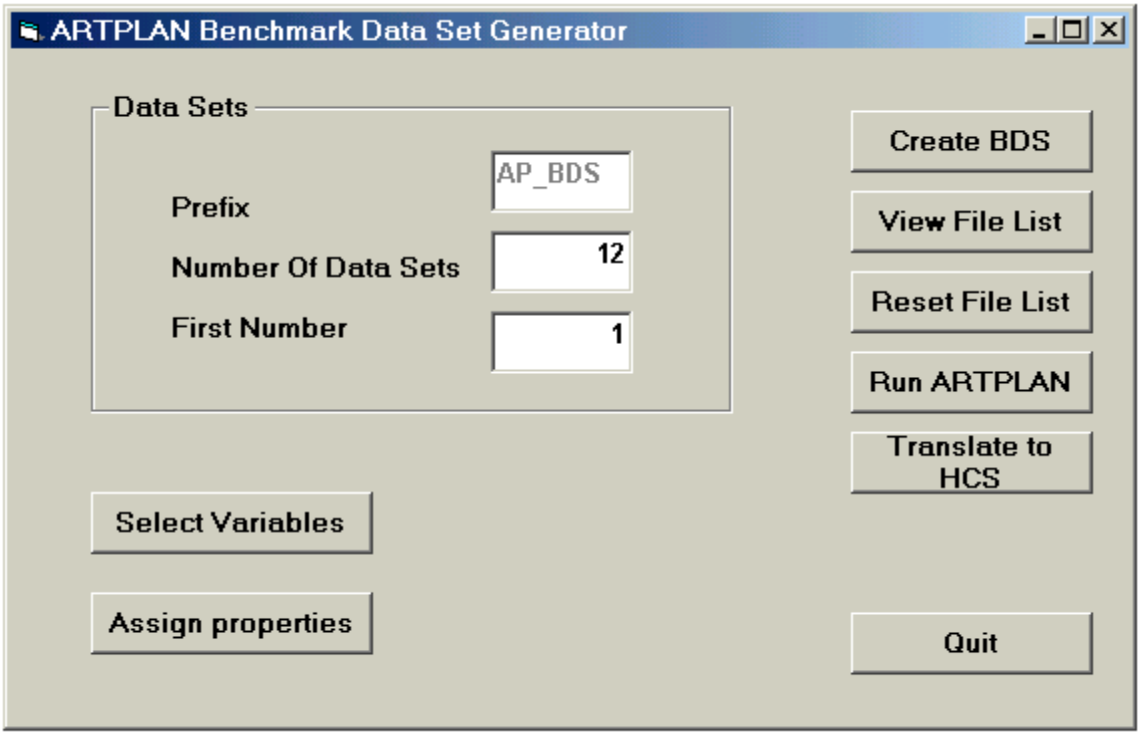


Figure 4-2. APBDS Main Menu Screen

2. Figure 4-3 is the Properties Assignment Menu Screen for APB DS. In the Area Type section, select Rural Developed. In the Control Type section, select Pretimed. Enter 2 as the Number of Intersections and click "OK" to return to the APBDS Main Menu Screen.

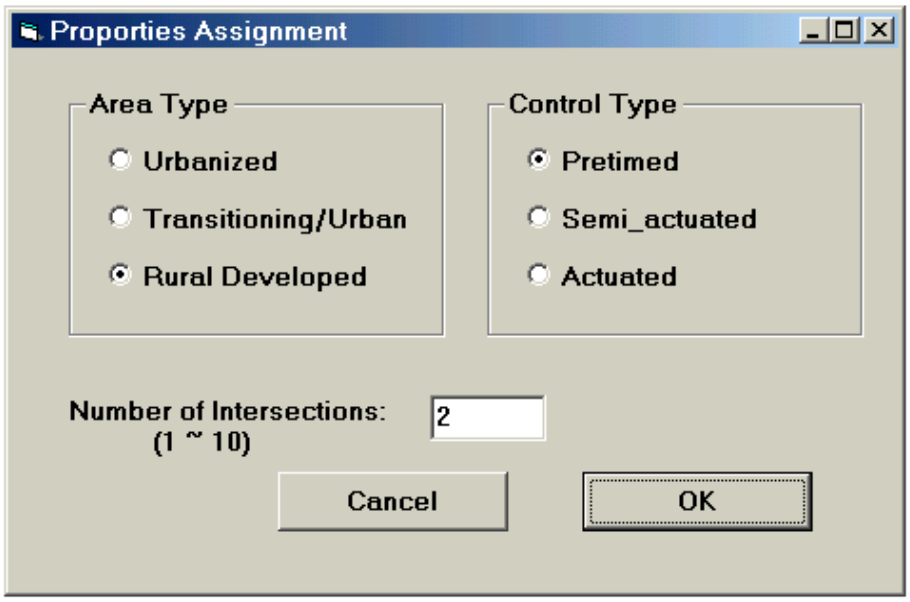


Figure 4-3. Properties Assignment Screen for APBDS

3. At the APBDS Main Menu Screen (Figure 4-2), click on the Select Variables button. This will open the table shown in Figure 4-4. Set Signal Per Mile to be Random by Facility by checking the box in the Variable Selection table. Leave the remaining variables as fixed. Click the OK button to return to the APBDS Main Menu Screen.

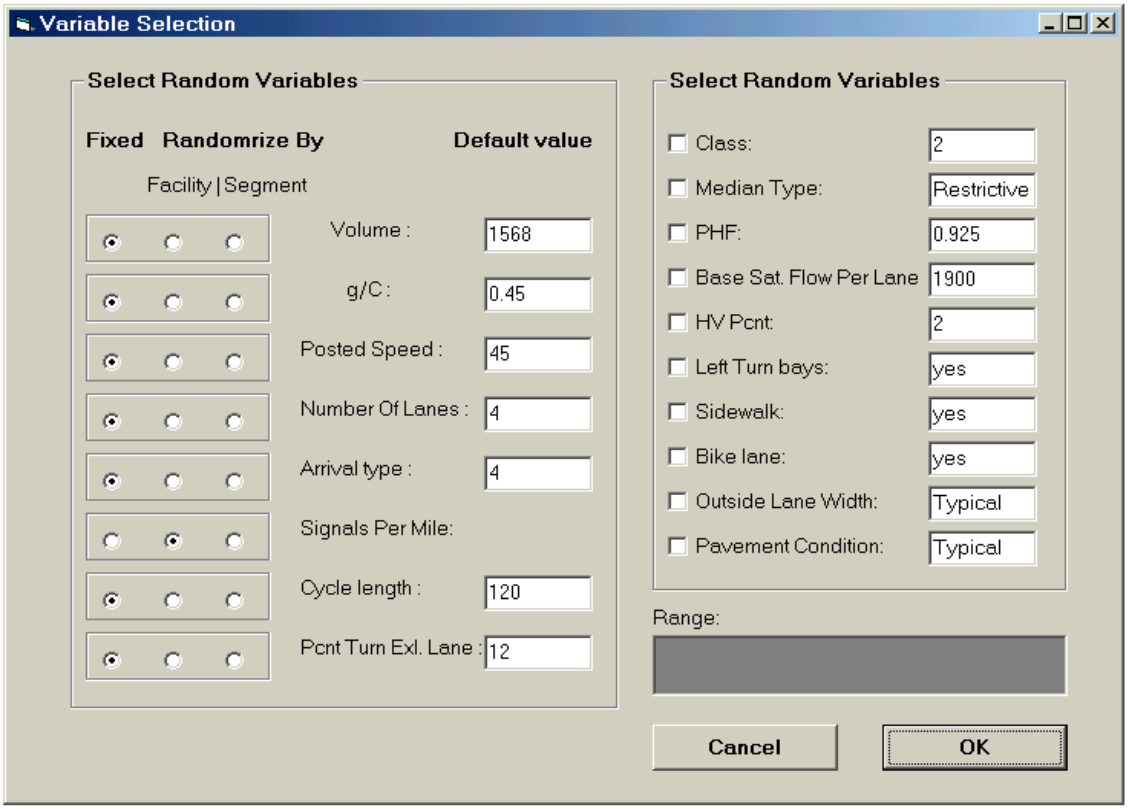


Figure 4-4. APBDS Variable Selection Table

4. At the APBDS Main Menu Screen (Figure 4-2) enter 12 as the Number of Data Sets and 1 for the First Number selection. Click on the Reset File List button. The screen will appear as it does in Figure 4-5 with only the first typed line showing. Close this screen (File-Close.) At the APBDS Main Menu Screen click on the Create BDS button to generate the data sets.
5. To check the names and paths of the newly created data sets, click on the View File List button at the APBDS Main Menu Screen (Figure 4-2). The Benchmark Data File List will appear as shown in Figure 4-5. The data files are stored in the BDS folder under the APBDS directory. The file lists are saved under APBDS.txt. Samples of the File List and Data Sets are shown in Figure 4-5 and Figure 4-6.

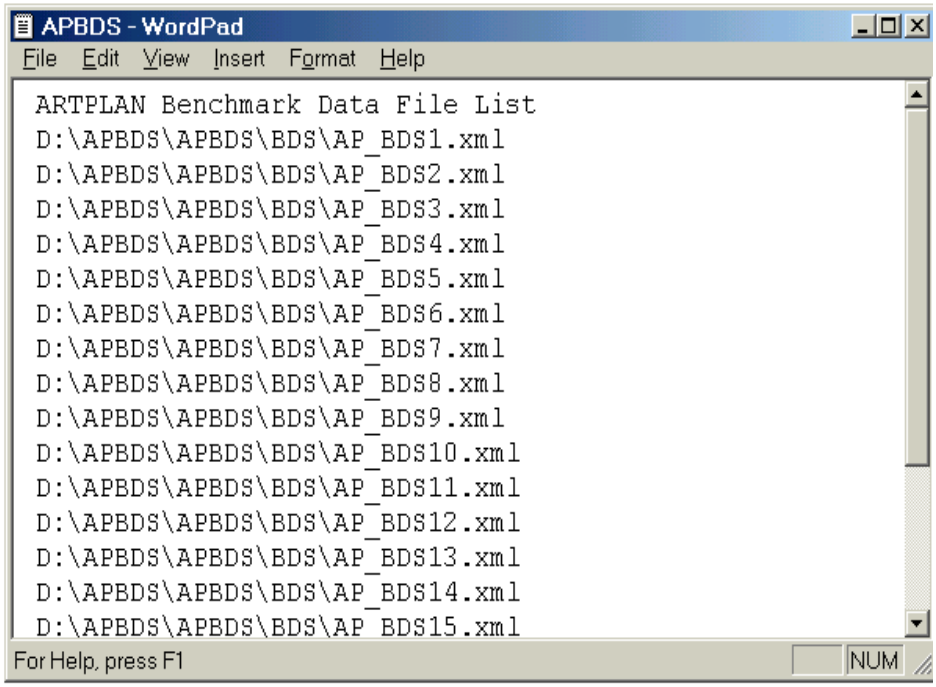


Figure 4-5. File List in APBDS

```
<TMML Facility="Arterial">
- <AGENCY>
    <AgencyName>UF-TRC</AgencyName>
    <CityState>Not Specified</CityState>
  </AGENCY>
- <MODELPARAMETERS>
    <AnalysisType>Operations</AnalysisType>
    <PeriodHr>0.25</PeriodHr>
  </MODELPARAMETERS>
- <GENERAL>
    <Date>12/18/2001</Date>
    <PeriodID>K100</PeriodID>
    <Analyst>BDS</Analyst>
  </GENERAL>
- <ARTERIAL ID="1">
  - <ARTERIALINFO>
      <ArterialName>Benchmark Road</ArterialName>
      <FwdDirection>Northbound</FwdDirection>
      <AreaType>Rural Developed</AreaType>
      <ArterialClass_HCM>2</ArterialClass_HCM>
      <PostedSpeed>45</PostedSpeed>
      <MedianType>Restrictive</MedianType>
      <AADT>30000</AADT>
      <KFactor>.095</KFactor>
      <DFactor>.55</DFactor>
      <DDHV>1568</DDHV>
```

Figure 4-6. Sample of Data Sets Generated by APBDS

6. At the APBDS Main Menu Screen (Figure 4-2) click on the Run ARTPLAN button. The 20 generated data sets in the list will automatically be processed through ARTPLAN, one by one.
7. Next, (at the APBDS Main Menu Screen) click on the Translate to HCS button. An Open File screen will appear. Highlight the files generated in step 4 (shown in Figure 4-5) and click the Open button. These 20 files will be translated to the HCS structure automatically. The translated files will be stored in the same directory with the original file names followed by the letters "_HCS" and the APBDS Main Menu Screen will appear. Close the APBDS program.
8. Open the HCS program and run the translated data sets in HCS-Arterial (planning).
9. Open the TMRC program and run the Traffic Model Results Comparison (TMRC) (Figure 4-7). For the box labeled Facility, select Arterial. Select Use File List and type in the path and name of the APBDS list file. Check the box Use First File in List.

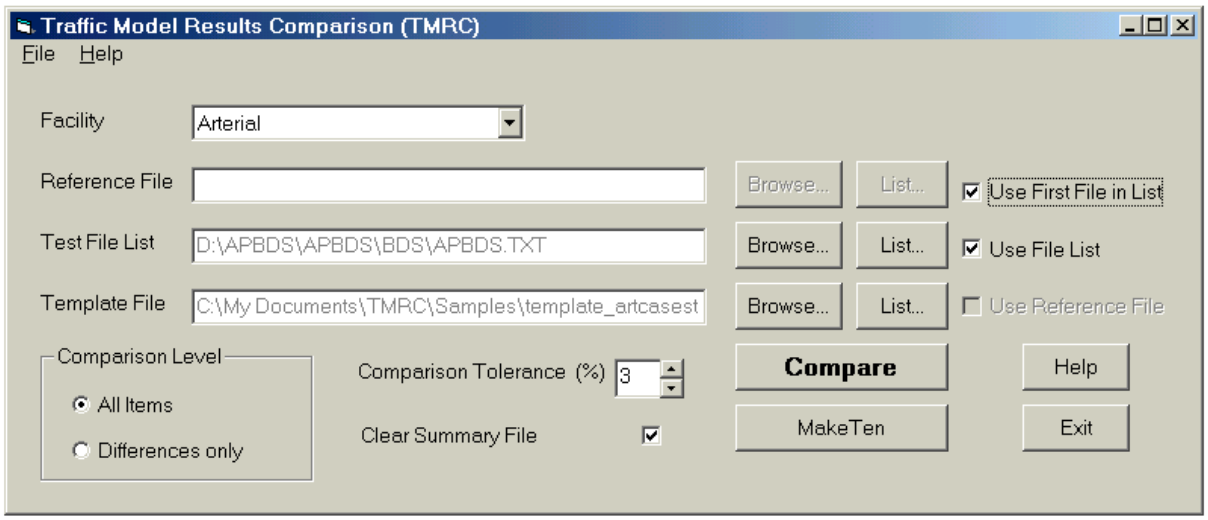


Figure 4-7. TMRC Main Menu Screen

10. Copy one of the data files to create a template according to the rules described in Chapter 3.1.2. Still working in the TMRC Main Menu Screen, type the path and name of the newly created file in the Template File box. Check the box for Clear Summary File and click on the Compare button to start multiple file comparisons. The template file is shown in Figure 4-8.

```
<?xml-stylesheet type="text/xsl" href="D:\APBDS\APdefault.xsl"?>
 <TMML Facility="Arterial">
   <ARTERIAL ID= "1">
      <APPROACH ID= "forward">
         <MOEGROUP ID= "forward">
            <AvgTravelSpeed>@02AverageSpeed</AvgTravelSpeed>
            <LOS>@03LOS</LOS>
         </MOEGROUP>
      </APPROACH>
      <INTERSECTION ID= "2">
         <APPROACH ID= "forward">
            <LANEGROUP ID= "T">
               <LinkLength>@01SignalSpacing</LinkLength>
            </LANEGROUP>
         </APPROACH>
      </INTERSECTION>
   </ARTERIAL>
 </TMML>
```

Figure 4-8. TMRC Template File

11. The results of the comparisons will be saved in a spreadsheet format in a file labeled TMRCSum.txt under the working Directory of TMRC (see Figure 4-9.)

```
SignalSpacing, AverageSpeed, LOS
1.82424, 43.79, B
0.3589, 27.81, B
1.19848, 40.81, B
2.95852, 46.26, B
0.26648, 24, B
0.73788, 36.3, B
1.80682, 43.73, B
0.91913, 38.49, B
0.44318, 30.46, B
0.77992, 40.66, B
0.77992, 36.88, B
0.54716, 32.99, B
1.9036, 44.04, B
1.9036, 44.04, B
0.56364, 33.33, B
1.76667, 37.84, B
1.76667, 43.59, B
0.68598, 35.53, B
0.61383, 34.31, B
```

Figure 4-9. TMRC Summary File

12. Once the Summary File is saved, use Excel to open the file and create a Density-Flow chart for the ARTPLAN data.

13. Repeat steps 9 through 12 for the translated HCS data files. The combined results are shown in Figure 4-10

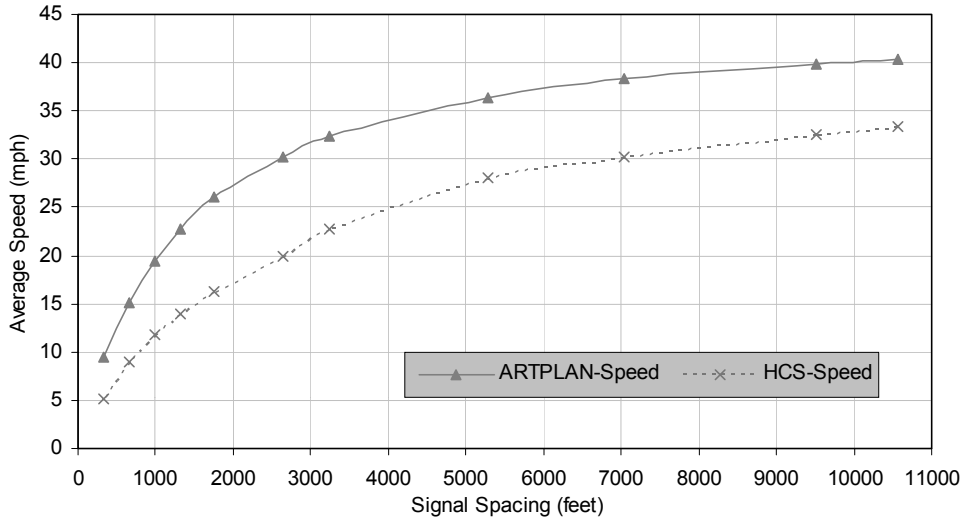


Figure 4-10. Average Travel Speed for Different Signal Spacing

4.4 ARTERIAL TESTS AND RESULTS

4.4.1 Performance Test Findings

17 data sets are randomly generated to compare the results given by ARTPLAN and HCS-Arterial (Planning). In almost all of the situations covered by these data sets, ARTPLAN gives the average travel speeds as two times the HCS response.

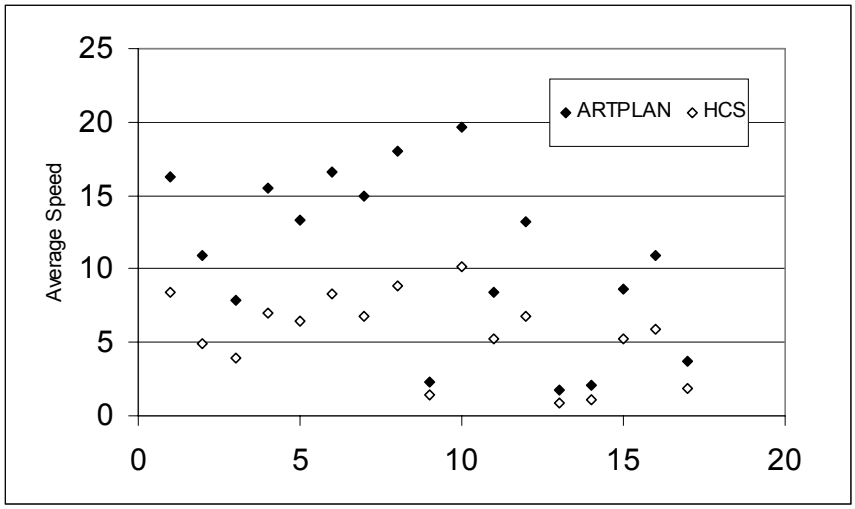


Figure 4-11. Performance Test for Arterial Facilities

4.4.2 Sensitivity Test Findings

These are some of the questions answered in the test and a comparison of the results provided by the two models:

- How much does an AADT affect delay and average travel speed?
- What happens to the average speed when free flow speed varies?
- How is the average speed affected by signal spacing?
- How will the delay and average speed vary for different cycle length, g/C ratio?
- How does the number of lanes affect the delay and average speed?

A major difference in the input files of the two models is that ARTPLAN accepts above parameters entered by segments while HCS-Arterial (Planning) does not. To eliminate this difference and minimize the data size, only one link is examined. This link connects two intersections with a type 4 arrival (defined as moderately dense platoon arriving in the middle of the green phase or a dispersed platoon containing 40 to 80 percent of the lane group volume arriving throughout the green phase.) from upstream. Table 4-2 gives the parameter settings for the Sensitivity Test:

Table 4-2. Default Values in Sensitivity Test

Median Type	Nonrestrictive	Base Sat. FR	1900
K	0.095	D	0.55
Num of lanes	4	Heavy vehicle %	2
Free Flow Speed	45	% Exclusive Lane	16
Signals/mile (Spacing)	32(2640)	Local Adj. Factor	1
PHF	0.925	Arrival type	4
Left Turn Lanes	Yes	Cycle Length	120
Thru g/C	0.45	Control Type	Pretimed
AADT	10000		

4.4.2.1 Volume

Altering any of the following parameters in either or both models can change the volume: AADT, K Factor, D Factor, PHF. AADT was chosen as the single variable in the test in order to obtain more recognizable relations. There were 14 data sets generated and run through both ARTPLAN and HCS-Arterial (Planning).

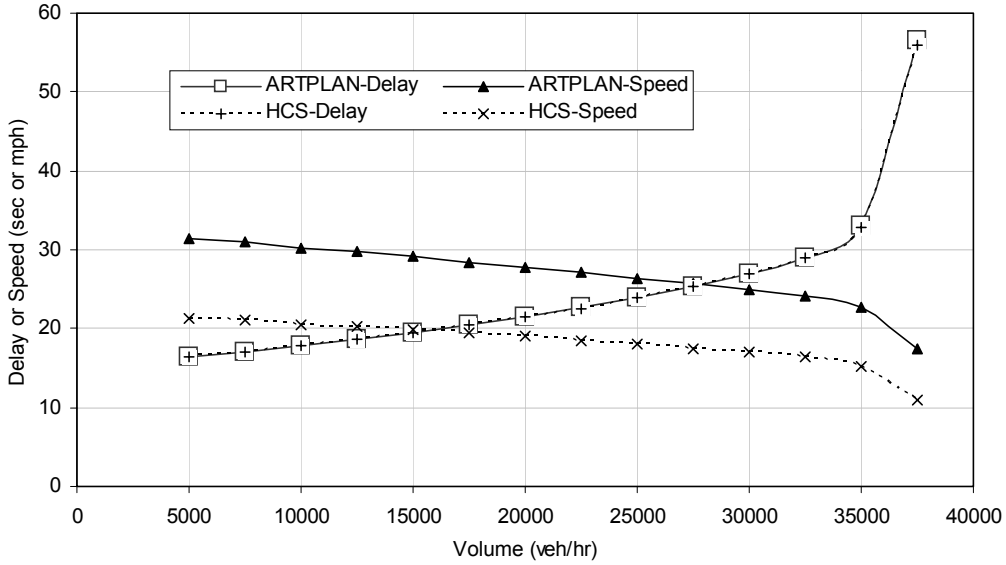


Figure 4-12. Delay & Average Travel Speed for Different Volumes

As can be seen in Figure 4-12, the lines that represent Delay from both ARTPLAN and HCS overlap. These results show that ARTPLAN uses the HCS method for control delay calculation. The average travel speed calculated by ARTPLAN is much higher than the one from the HCS. That explains why ARTPLAN always comes out with a better LOS than HCS, since LOS mostly depends on travel speed for arterials. Both delay and average speed have a sudden change when the v/c ratio closes to 1.0.

4.4.2.2 Free Flow Speed

The average travel speed (Figure 4-13) varies with both control delay and free flow speed. From the analysis above we know that the two models use the same method in control delay calculation. The Average Speed-FFS relationship in ARTPLAN is quite close to a simple linear relationship; while in HCS the curve is smoother, which represents a logarithmic relationship.

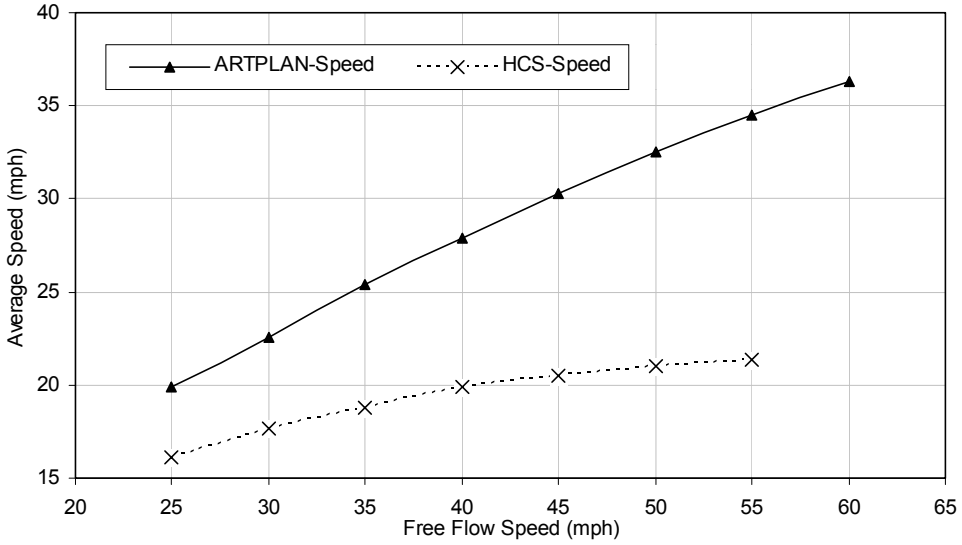


Figure 4-13. Average Travel Speed for Different Free Flow Speeds

4.4.2.3 Signal Spacing

Figure 4-14 shows the average travel speed influenced by signal spacing. A range from 660 feet, which is 8 signals per mile, to 10560 feet, which is 0.5 signals per mile, has been tested. The curve representing the ARTPLAN results shifts significantly above the curve from HCS data. The average speed increases dramatically for smaller signal spacing and decreases when the spacing becomes greater in ARTPLAN. While in the HCS model, the whole curve is smoother than ARTPLAN. Again, ARTPLAN gives relatively higher LOS. The data reflects that for lower spacing, average speed is more sensitive to signal spacing in the ARTPLAN model.

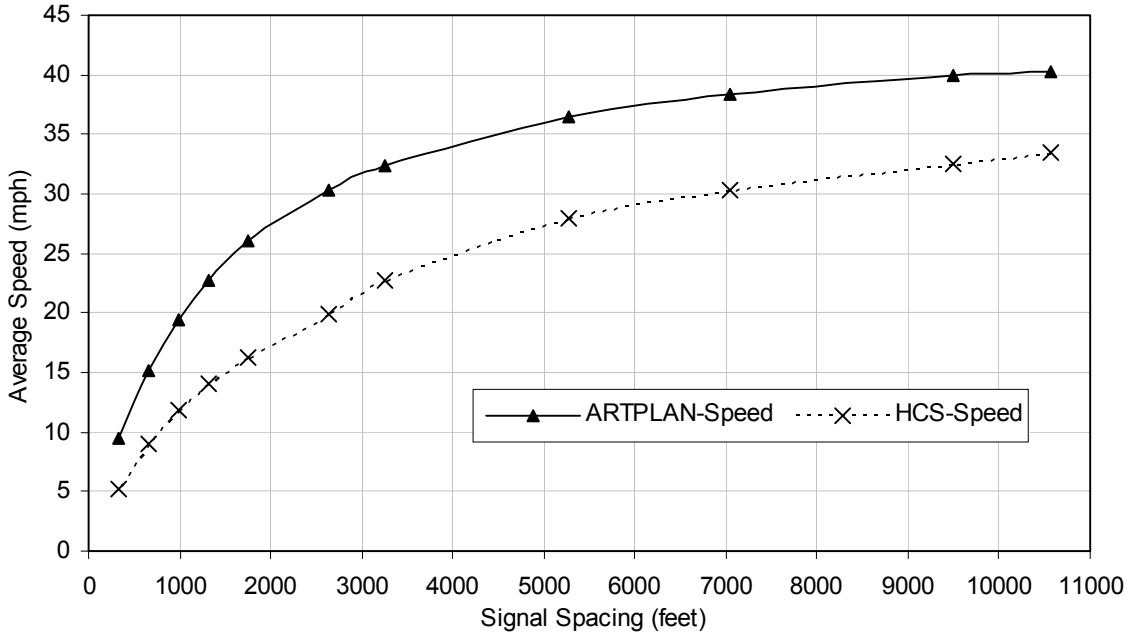


Figure 4-14. Average Travel Speed for Different Signal Spacing

4.4.2.4 Cycle Length

Figure 4-15 presents the relationship between delay, average speed and cycle length.

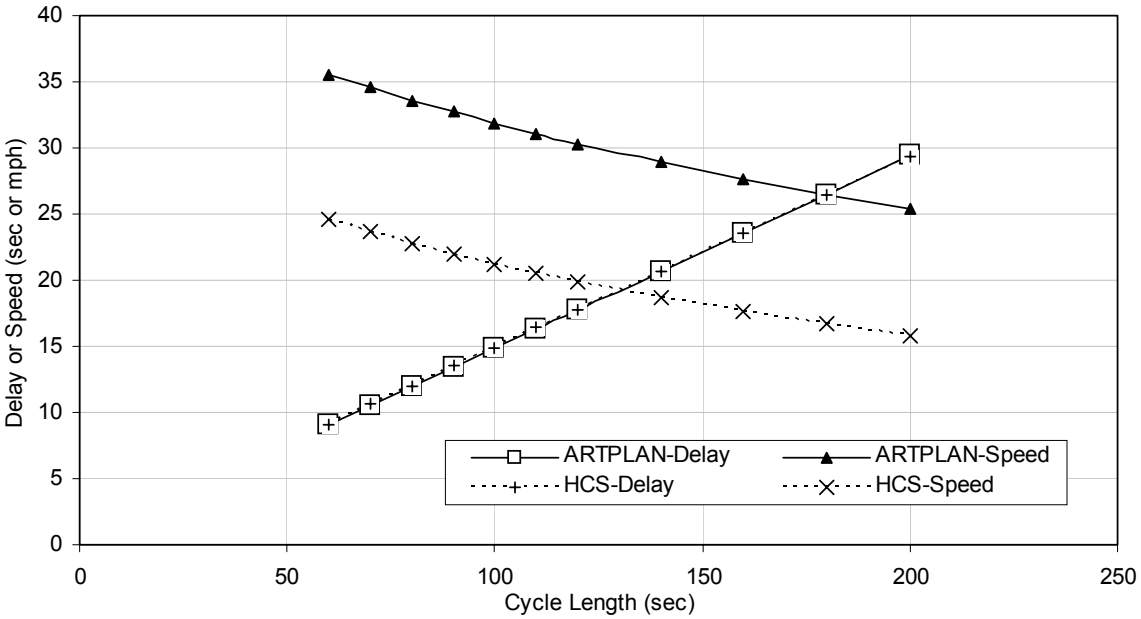


Figure 4-15. Average Travel Speed for Different Cycle Length

4.4.2.5 g/C Ratio

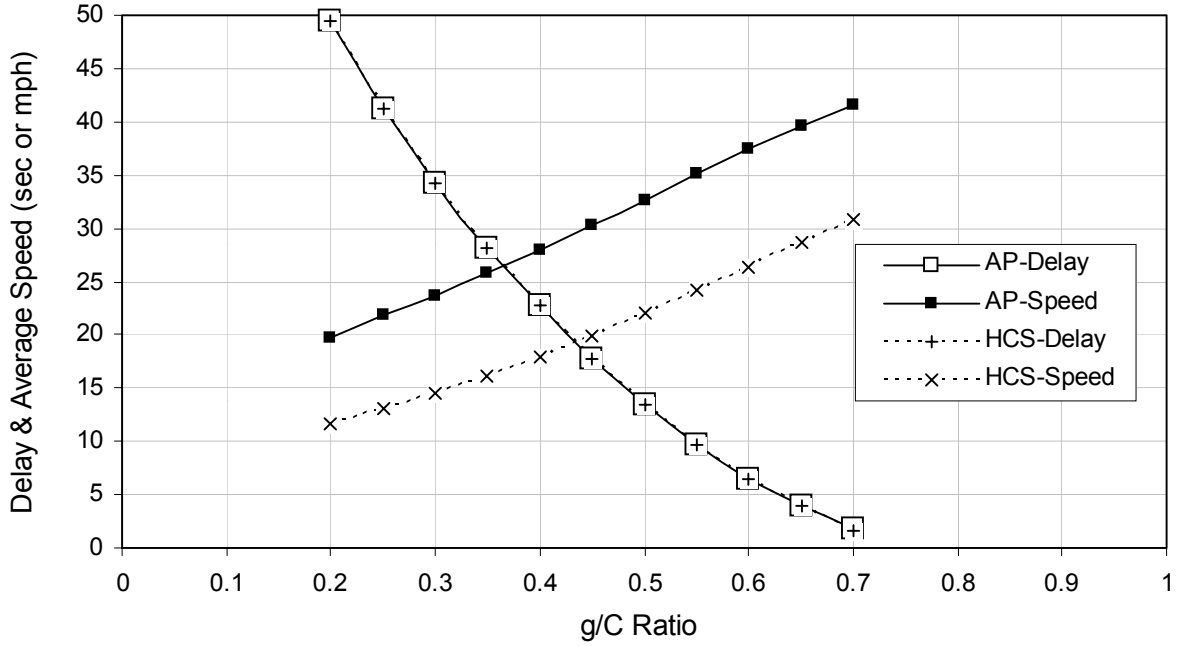


Figure 4-16. Delay and Average Speed for Different g/C Ratio

Delay-g/C Ratio and Average Speed-g/C Ratio relationships for the two tested models are shown in Figure 4-16. Again, the delay curve for both models overlapped. The ARTPLAY speed curve shifted about 8 to 10 mph above the HCS speed curve.

4.4.2.6 Number of Lanes

Figure 4-17 illustrates the relationship between the number of through lanes and average speed. These curves, similar to the speed-spacing curves, show that the influence of the number of lanes to both delay and average speed decreases when the lane number increases.

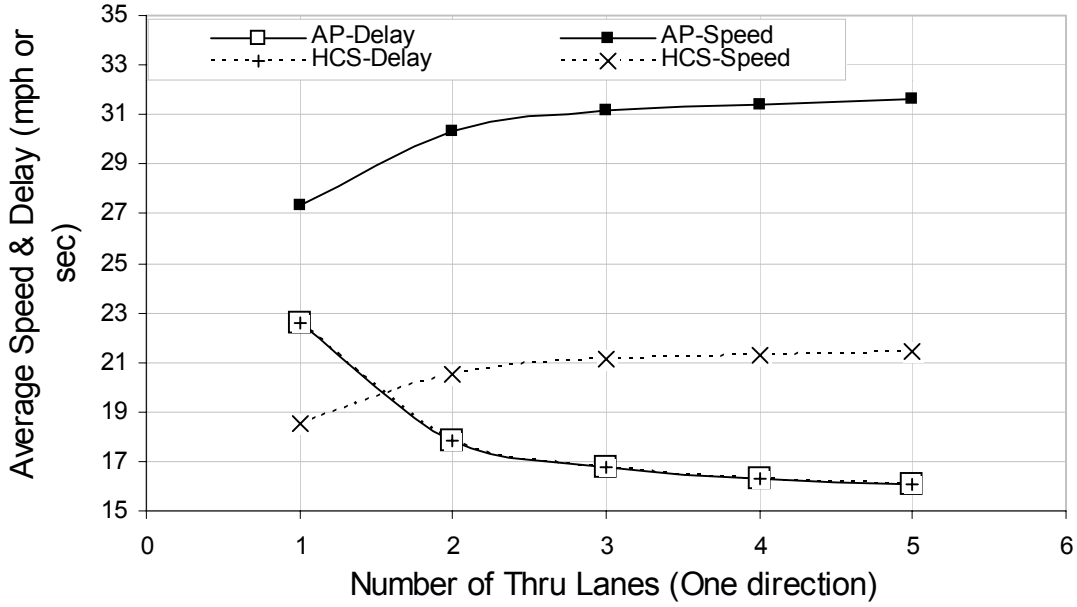


Figure 4-17. Delay and Average Speed for Different Number of Lanes

All of the comparison results from both ARTPLAN and HCS prove that the two models use the same method in delay calculation [1]; and a similar method but different measurements in average speed calculation. The differences in the results can be explained by the internal relationships that are evident in the two models. ARTPLAN uses a different running timetable than the one used in the Highway Capacity Manual, Chapter 15 [1] [8]. The estimated FFS used by the two models is also different:

Urban Street Class	FFS in ARTPLAN	FFS in HCS
1	55, **50**, 45	
2	45, **40**, 35	45, **40**, 35
3	**35**, 30	**35**, 30
4	35, **30**, 25	35, **30**, 25

In Table 4-3, the values shown in parentheses are the HCS values.

Table 4-3. Running Time Per Mile Table in ARTPLAN & HCS

Class	1					2				3			4		
FFS	55	50	45	40	35	45	40	35	30	35	30	25	35	30	25
Average Segment Length	Running Time per Mile														
0.05											**227**	**265**		(227)	(265)
0.10								**145**	**155**	**165** (145)	**180** (155)	**220**	(165)	(180)	(220)
0.15								**135**	**141**	**140** (135)	**150** (141)	**180**	(140)	(150)	(180)
0.20			**109**	**115**	**125**	(109)	(115)	**128** (125)	**134**	**130** (128)	**140** (134)	**165**	(130)	(140)	(165)
0.25	(97)	(100)	**104** (104)	**110**	**119**	(104)	(110)	**120** (119)	**127**	**122** (120)	**132** (127)	**153**	(122)	(132)	(153)
0.30	(92)	(95)	**99** (99)	**102**	**110**	(99)	(102)	(110)							
0.40	(82)	(86)	**94** (94)	**96**	**105**	(94)	(96)	(105)							
0.50	(73)	(78)	**88** (88)	**93**	**103**	(88)	(93)	(103)							
1.00	(65)	(72)	**80** (80)	**90**	**103**	(80)	(90)	(103)							

5 APPLICATION TO HIGHWAY FACILITIES

This chapter describes the application of the benchmark data generation and analysis methodology presented in Chapter 3 to two-lane and multilane highways arterials. The principal objective of this part of the study is to demonstrate the testing of HIGHPLAN as a planning level implementation of the HCM procedures for highway level of service analysis. The main topics include the development of a software tool for generating Freeway Benchmark Data Sets, and the performance and sensitivity tests that were carried out using the benchmark data.

5.1 TMML STRUCTURE FOR HIGHWAYS

The TMML class structure for highway facilities is depicted in Figure 5-1. The individual data elements within each class are identified in Appendices A and B.

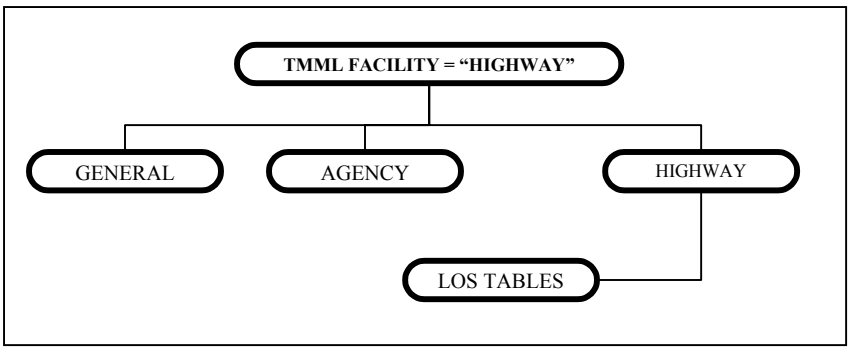


Figure 5-1. TMML Structure for Highways

5.2 THE HIGHWAY BENCHMARK DATA SET GENERATOR

Figure 5-2 shows the user's interface of HPBDS. Up to 200 data sets can be generated each run; and data files are named as the prefix followed by a number. Clicking on the Select Variables button will open the Variable Selection Table showing in Figure 5-3. The system will generate random numbers for the checked parameters. If a parameter is left unchecked, the system will automatically provide a fixed value in the text box. The user can edit this value; and the corresponding valid range will be displayed in the lower right corner when clicking in the text box. The other function of HPBDS is to read in an XML file in HIGHPLAN structure and transfer it to the HCS- Multilane or Two-lane structure. Multiple files can be translated at the same time and a "_hcs" will be added to the original file name to denote the translated files.

The following steps are involved in creating and analyzing the Benchmark Data Sets:

1. Identify the appropriate ranges and distributions of the input data and operating parameters for testing multilane and 2-lane roadway operations. Table 5-1 shows the valid ranges and distributions of the parameters in random data sets. Some fixed values are also included in the table.
2. HPBDS was developed in Visual Basic to create XML formatted data sets. The features of BDS are described in Chapter 3.1 Software Tools. In the performance test in HPBDS, all parameters are chosen as randomized.
3. In step 3, the XML data sets created in step 1 are translated to accommodate the format of HCS-Multilane (or Two-lane, since HCS Two-lane doesn't support the XML format at this time, only the multilane facility has been tested in this research). Due to the nuances in structure and tag names in the two XML files, the data sets generated by BDS cannot be read by HCS directly. The translation rules are mapped into HPBDS. Any XML file in HIGHPLAN structure can be read into the program and rewritten in HCS structure with "_hcs" added to the original file name to indicate the translated file.
4. The files are processed through HIGHPLAN and HCS-Multilane respectively in step 4 and the results saved in XML format.
5. In step 5, multiple comparisons are done to the file list according to the template designed in the instructions discussed earlier in Chapter 3.1 Software Tools, and the output is saved in a spreadsheet format text file.
6. In step 6, analyze the comparison output using statistic methodologies and derive conclusions.

Table 5-1. Valid Range and Distribution in HPBDS

VARIABLES	RANGE	DISTRIBUTION
Area type	Urbanized Transitioning/Urban Rural developed Rural undeveloped	Two Lane Roads: 40% Rural undeveloped 20% Rural developed 20% Transitioning/Urban 20% Urbanized Multilane Roads: 25% for each category
Class (Two-lane only)	1 2	75% Class 1 25% Class 2
Posted Speed	40, 45, 50, 55, 60, 65	Uniformly distributed among 40, 45, 50, 55, 60, 65
Number of lanes (Both Directions)	2, 4, 6, 8	Tow-lane Roads: 100% 2 lanes Multilane Roads: 40% 4 lanes 40% 6 lanes 20% 8 lanes
Road name		Fixed = "BDS Road"
Peak Direction	N, S, E, W	Fixed = "Northbound"
Study Period	K30, K100, K5-6, Kp/d, Kother	Fixed = "K100"
AADT	1000 ~ 100000	Uniform 1000 to 35000 per lane (integer)
K Factor	0.06 ~ 0.20	20% uniform distributed 0.06 to 0.09 60% uniform distributed 0.09 to 0.10 20% uniform distributed 0.10 to 0.20
D Factor	0.5 ~ 1	75% uniform distributed 0.5 to 0.6 25% uniform distributed 0.6 to 1.0
PHF	0.75 ~ 1	25% uniform distributed 0.75 to 0.85 75% uniform distributed 0.85 to 1.0
%Heavy Veh	0 ~ 25	75% uniform distributed 0 to 10 25% uniform distributed 10 to 25
Base capacity	1400 ~ 2400	Two-lane / Multilane Rural undeveloped 1700 / 2300 Rural Developed 1900 / 2100 Urban/Transitioning 1700 / 2200 Urbanized 1700 / 2000
Local Adj. Factor	0.8 ~ 1.0	75% Uniformly distributed 0.9 ~ 1.0 25% Uniformly distributed 0.8 ~ 0.9
Terrain	Level, Rolling	75% level 25% Rolling
Passing Lane Spacing (Two-lane Only)	0 ~ 30	75 % zero 25% uniformly distributed 3 to 15 miles (Integer)
Prop no passing Zone (Two-lane Only)	0 ~ 100	Uniformly distributed 0 to 100

Sensitivity Test Part:

7. Set all parameters to reasonable fixed values by observing the output of the performance test. Save the file as a basic XML file.
8. Open the basic XML file in a text editor. Generate data sets in HPBDS based on single parameter changes (10 versions for each parameter.) This configuration will test the sensitivity to that particular parameter. Save the data sets by groups.

   Parameters to be tested with their ranges are itemized as follows:

   - Volume: distributed between 30 and 20000, valid range in the HCS-Multi lane 0 to 999999;
   - PHF: distributed between 0.25 to 1, valid range in the HCS-Multilane 0.25 to 1;
   - Number of Lanes: 4, 6, 8 in both direction, valid value in the HCS-Multilane 2 or 3 for one direction (4-lane roadways cannot be tested in this research);
   - Free Flow Speed: 45, 50, 55, 60, 65, 70, valid range in the HCS-Multilane 45 to 60 (multilane roadways with a posted speed of 60 or 65 cannot be tested in this research);
   - Heavy vehicle percentage: distributed between 0 to 25, valid range in the HCS-Multilane 0 to 25;

9. Translate these files to the HCS-Multilane format as in step 3;
10. Create new templates that include a single test parameter and Density, LOS;
11. Do multiple comparisons by groups and save the results to a spreadsheet.
12. Analyze the output and derive conclusions.

5.3 A HIGHPLAN BDS EXAMPLE

The following step-by-step detailed example will illustrate how to apply benchmark data to test the density-flow rate relationship of multilane highways and compare the HIGHPLAN results with the Highway Capacity Manual, Chapter 12 results.

1. Run HPBDS from Windows. The HPBDS Main Menu Screen will appear (see Figure 5-2). Select Multilane as the Facility choice and click on the Select Variables button.

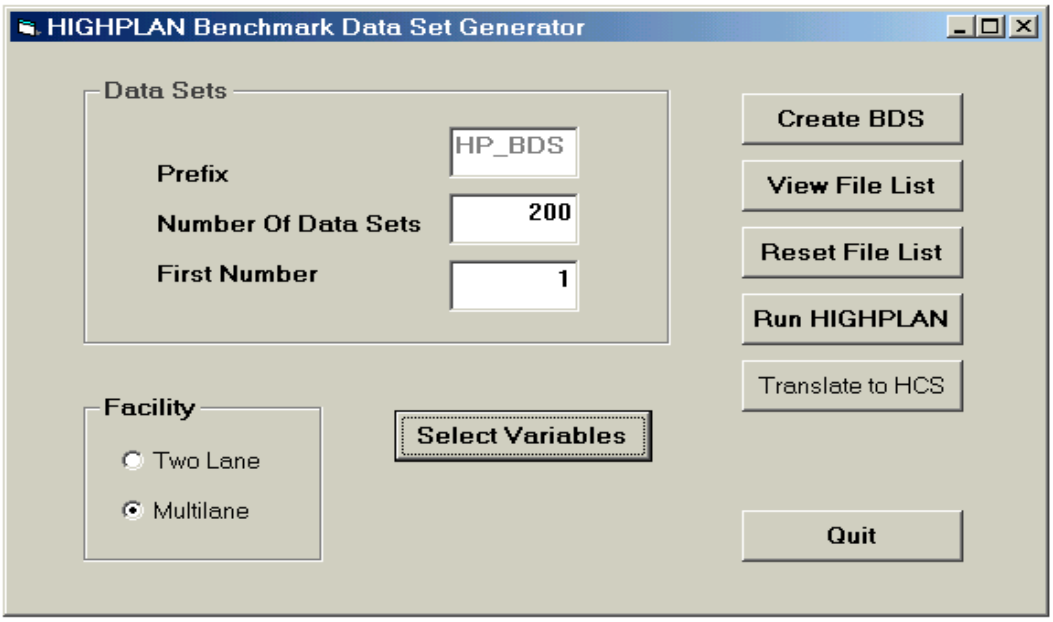


Figure 5-2. HPBDS Main Menu Screen

2. Set all of the variables to be random by checking the boxes in the variable selection table (see Figure 5-3) and click on "OK" to return to the main screen.

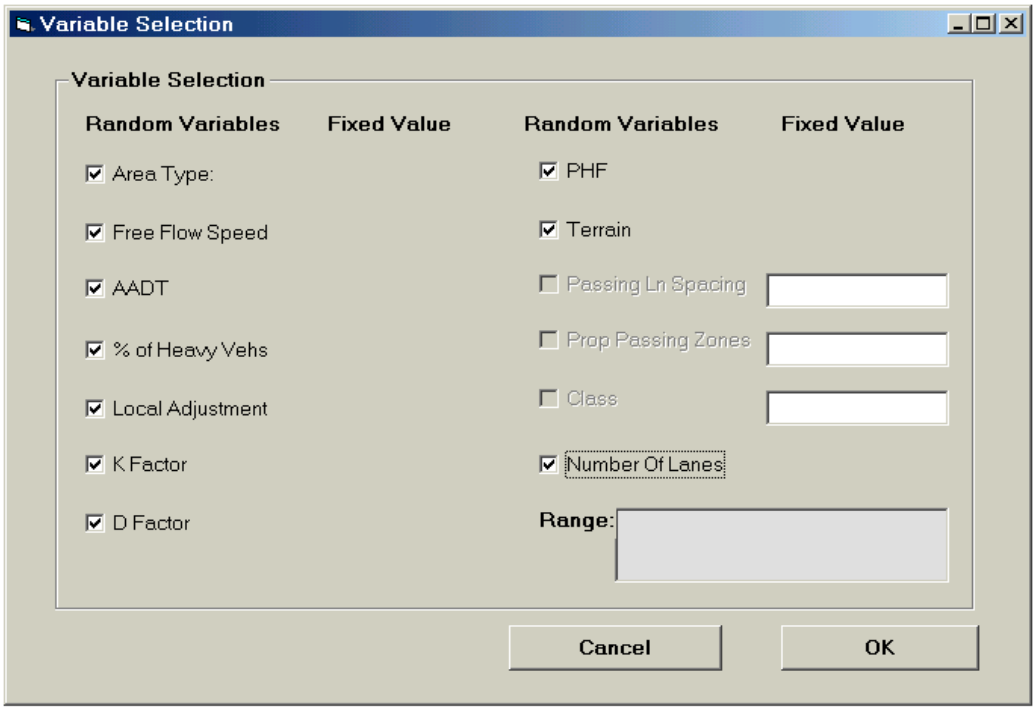


Figure 5-3. HPBDS Variable Selection Table

3. At the HPBDS Main Menu Screen, type the Number of Data Sets as "200" and the First Number as "1". Click on the Reset File List button. The screen will appear as it does in Figure 5-3 with the line HIGHPLAN Benchmark Data File List. Close this screen (File-Close.) then click on the Create BDS button to generate the data sets.
4. To check the names and paths of the newly created data sets, click on the View File List button at the APBDS Main Menu Screen (Figure 5-2). The Benchmark Data File List will appear as shown in Figure 5-4. The data files are stored in the BDS folder under the HPBDS directory. The file lists are saved under HPBDS.txt. Samples of the File List and Data Sets are shown in Figure 5-4 and Figure 5-5.

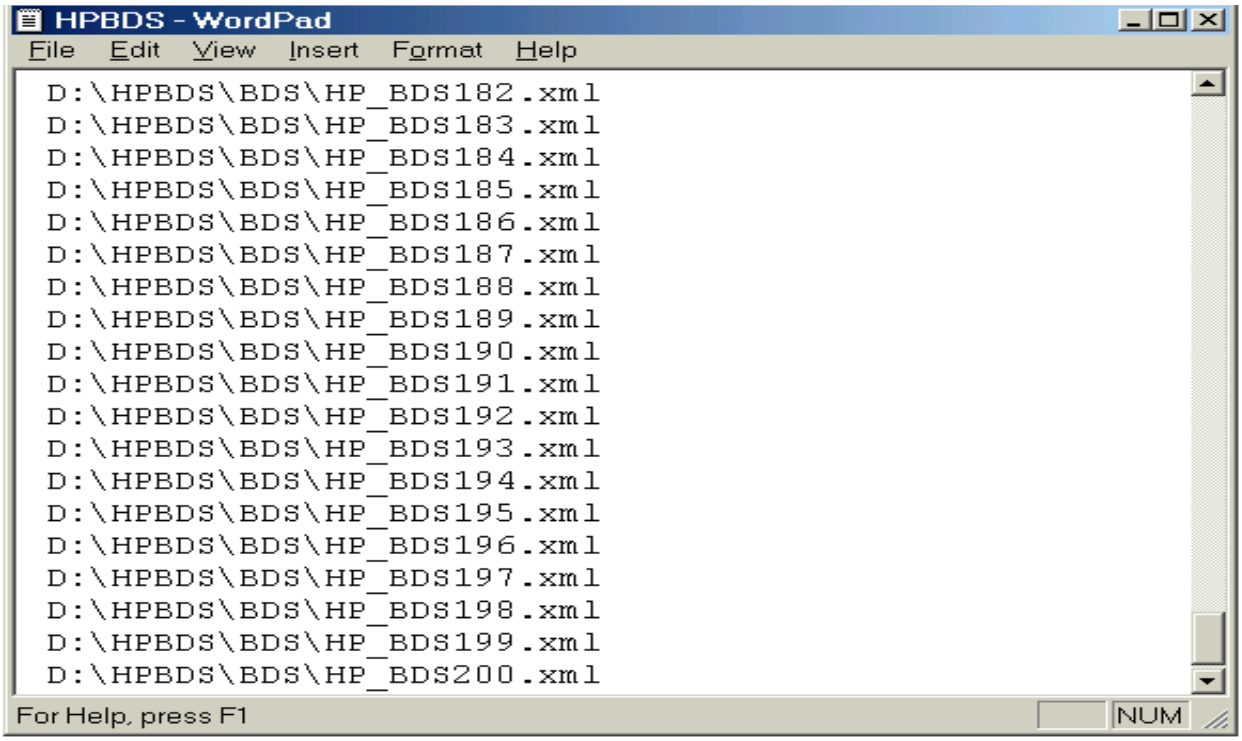


Figure 5-4. File List in HPBDS

5. At the HPBDS Main Menu Screen (Figure 5-2) click on Run HIGHPLAN button. The 200 data sets in the list will automatically be processed through HIGHPLAN one by one.

```
<?xml version="1.0" ?>
<TMML Facility="MultiLane">
- <GENERAL>
    <PeriodID>K100</PeriodID>
    <Comment>D:\HPBDS\BDS\HP_BDS1.xml</Comment>
    <Analyst>BDS</Analyst>
    <Agency>UFTRC</Agency>
    <District>Not Specified</District>
  </GENERAL>
- <HIGHWAY>
    <RoadName>Benchmark Road</RoadName>
    <FromTo>Not Specified</FromTo>
    <AreaType>Urbanized</AreaType>
    <NumberOfLns>6</NumberOfLns>
    <PostedSpeed>60</PostedSpeed>
    <FreeFlowSpeed>65</FreeFlowSpeed>
    <PassingLnSpacing>15</PassingLnSpacing>
    <PropPassingZone>20</PropPassingZone>
    <FwdDirection>Northbound</FwdDirection>
    <AADT>16500</AADT>
    <HVPcnt>1</HVPcnt>
    <LocalAdj>.86</LocalAdj>
    <BaseCapPerLn>2000</BaseCapPerLn>
    <AdjCapPerLn>1720</AdjCapPerLn>
    <KFactor>.091</KFactor>
```

Figure 5-5. Sample of Data Sets Generated by HPBDS

6. Open the TMRC program and run the Traffic Model Results Comparison (TMRC) (Figure 5-6). For the box labeled Facility, select Multilane. Select Use File List and type in the path and name of the HPBDS list file. Check the box Use First File in List.

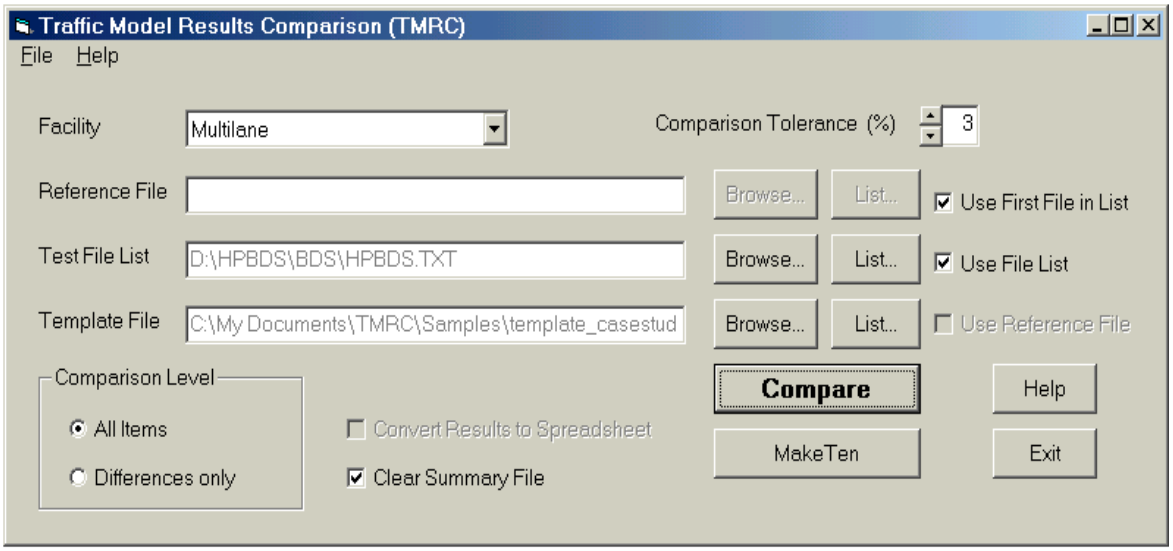


Figure 5-6. TMRC Main Menu Screen

7. Copy one of the data files to make a template according to the rules described in Chapter 3.1.2. Still working in the TMRC Main Menu Screen, type in the path and name of the newly created file in the Template File box. Check the box for Clear Summary File and click on the "Compare" button to start multiple file comparisons. The template file is shown in Figure 5-7.

```
<?xml version="1.0" ?>
    <TMML Facility="MultiLane">
    <HIGHWAY>
      <AreaType>@01Areatype</AreaType>
      <NumberOfLns>@02NumberofLn</NumberOfLns>
      <FreeFlowSpeed>@03FFS</FreeFlowSpeed>
      <AADT>@04AADT</AADT>
      <HVPcnt>@05HvPcnt</HVPcnt>
      <LocalAdj>@06LocalAdj</LocalAdj>
      <BaseCapPerLn>@07BaseCap</BaseCapPerLn>
      <KFactor>@08KFactor</KFactor>
      <DFactor>@09DFactor</DFactor>
      <PHF>@10PHF</PHF>
      <Terrain>@11Terrain</Terrain>
      <LOS>@12LOS</LOS>
      <VCRatio>@13VCRatio</VCRatio>
      <Density>@14Density</Density>
  </HIGHWAY>
  </TMML>
```

Figure 5-7. TMRC Template File

8. The results of the comparisons will be saved in a spreadsheet format in a file labeled TMRCSum.txt under the working Directory of TMRC (see Figure 5-8).

```
Areatype, NumberofLn, FFS, AADT, HvPcnt, LocalAdj, BaseCap, KFa
VCRatio, Density
Rural developed, 8, 45, 33489, 1, .804, 2100, .08, .56, .84, R(
Transitioning/Urban, 6, 50, 28906, 4, .978, 2200, .129, .548,
Transitioning/Urban, 8, 55, 21926, 0, .999, 2200, .105, .565,
Rural developed, 6, 70, 2783, 23, .951, 2100, .098, .781, .81,
Transitioning/Urban, 6, 55, 15245, 10, .951, 2200, .155, .554,
Transitioning/Urban, 4, 70, 15545, 3, .983, 2200, .068, .525,
Rural developed, 4, 60, 13448, 9, .962, 2100, .091, .525, .92,
Transitioning/Urban, 4, 55, 33189, 8, .973, 2200, .088, .953,
Urbanized, 6, 60, 31003, 23, .999, 2000, .094, .793, .94, Roll
Transitioning/Urban, 6, 60, 2184, 12, .988, 2200, .068, .609,
.7166666
Rural undeveloped, 8, 45, 19440, 21, .993, 2300, .099, .816, .
Rural undeveloped, 6, 50, 10063, 6, .962, 2300, .097, .6, .78,
Rural undeveloped, 6, 70, 12760, 7, .945, 2300, .093, .542, .9,
Rural developed, 4, 55, 22825, 0, .894, 2100, .092, .919, .92,
```

Figure 5-8. TMRC Summary File

9. Once the Summary File is saved, use Excel to open the file and create a Density-Flow Chart to compare the HCM [1] Chapter 12 data. The results are shown in Figure 5-9.

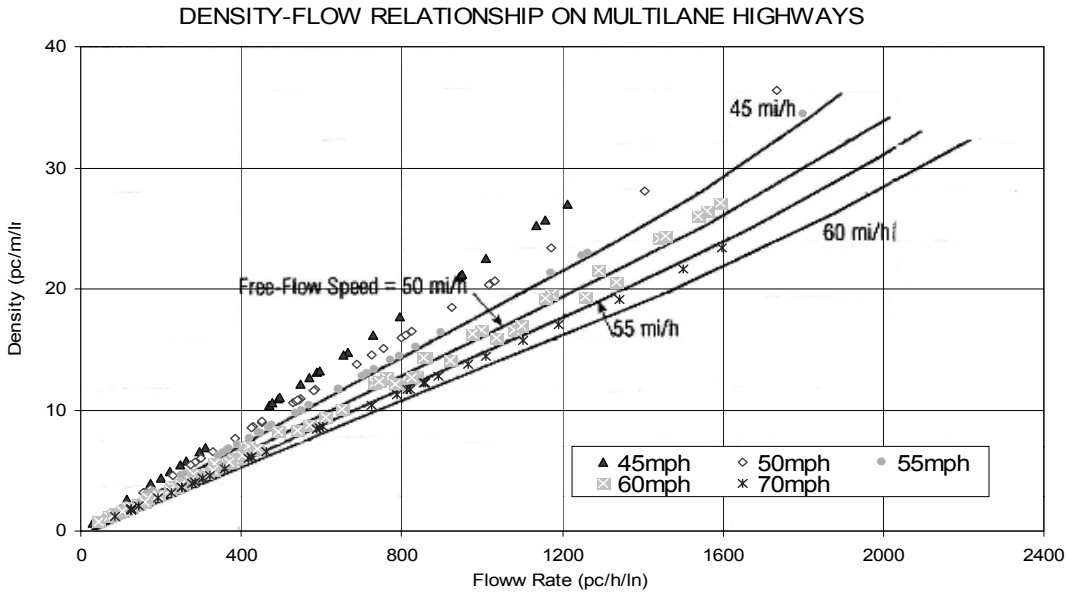


Figure 5-9. Density-Flow Relationship on Multilane Highway from HIGHPLAN & HCM

10. Analyze the summary data using SAS to obtain the equations.

Equations from SAS:

FFS= 45: Density = -0.0033 + 0.0222 FR

FFS= 50: Density = -0.4368 + 0.0209 FR

FFS= 55: Density = -0.1811 + 0.0186 FR

FFS= 60: Density = -0.9269 + 0.0183 FR

FFS= 65: Density = -0.0064 + 0.0154 FR

5.4 HIGHWAY TESTS AND RESULTS

5.4.1 Performance Test Findings

One of the first tasks to be undertaken is to compare the LOS given by the two models. 200 random data sets are generated and run by HIGHPLAN and the HCS-Multilane respectively to cover a wide range of conditions. Approximately 90% (182) of the results received the same output LOS from the two models. When the results were different, HIGHPLAN was more likely to give a lower LOS value than the HCS-Multilane. Table 5-2 shows the information of the first 10 data sets.

Table 5-2. Performance Test for Highway Facilities

1	AADT	Hv Pcnt	Local Adj.	Base Cap.	K Factor	D Factor	PHF	Terrain	HIGHPLAN			HCS-Multilane	
									LOS	V/C	D	D	LOS
55	22916	0	0.86	2300	0.183	0.55	1	level	C	0.61	25.4	21.8	C
65	11293	1	0.87	2000	0.157	0.539	0.9	rolling	A	0.21	6.4	5.9	A
50	22477	0	0.89	2100	0.06	0.544	1	level	A	0.11	10.9	8.0	A
50	25598	16	0.91	2300	0.1	0.536	0.9	level	B	0.31	14.5	10.6	A
55	11094	23	0.82	2100	0.103	0.691	0.9	rolling	A	0.28	10.6	5.5	A
45	22059	10	0.9	2200	0.16	0.547	1	level	B	0.29	14.2	14.9	B
55	9682	6	0.81	2100	0.098	0.578	0.9	rolling	A	0.14	5.3	3.7	A
50	10846	10	0.99	2200	0.093	0.516	0.7	level	A	0.19	8.2	7.1	A
55	31573	6	0.88	2000	0.095	0.533	0.9	level	B	0.37	13.5	10.9	A
45	21674	8	0.9	2100	0.142	0.561	0.9	level	C	0.54	25.3	20.4	C

Figure 5-10 explicates the difference in densities given by the two models for the same date sets. The densities given by HIGHPLAN lie above the densities given by the HCS. As the densities increase, the differences become more obvious.

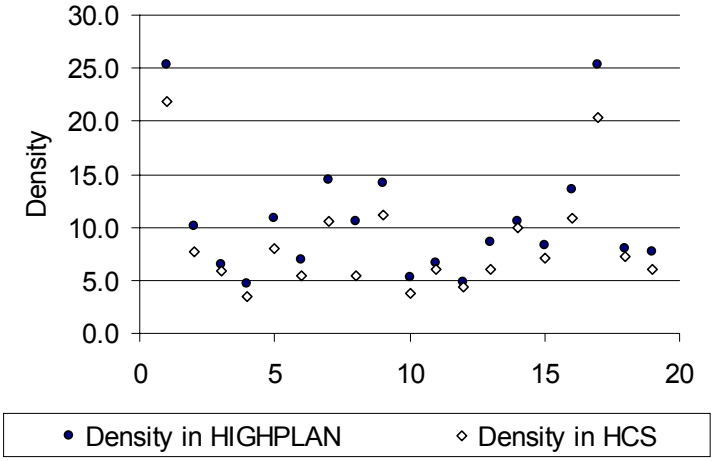


Figure 5-10. Performance Test for Highway Facilities

5.4.2 Sensitivity Test Findings

To test how the density is affected by volume the following parameters are fixed at reasonable values: Free Flow Speed = 55, Heavy Vehicle % = 5, Peak Hour Factor = 0.85. Volume ranges from 52 to 3135 vehicles/hour (AADT: 1000 to 60000) were tested. 46 data sets were generated and run through both HIGHPLAN and the HCS-Multilane (Operation). As can be seen in the Figure 5-11, the lines that represent the results from HIGHPLAN for 4 lane and 6 lane roadways, respectively, indicate higher densities than those from the HCS-Multilane for the same data sets. It is more likely to indicate a linear relationship between density and volume when the volume per lane is lower than 800.

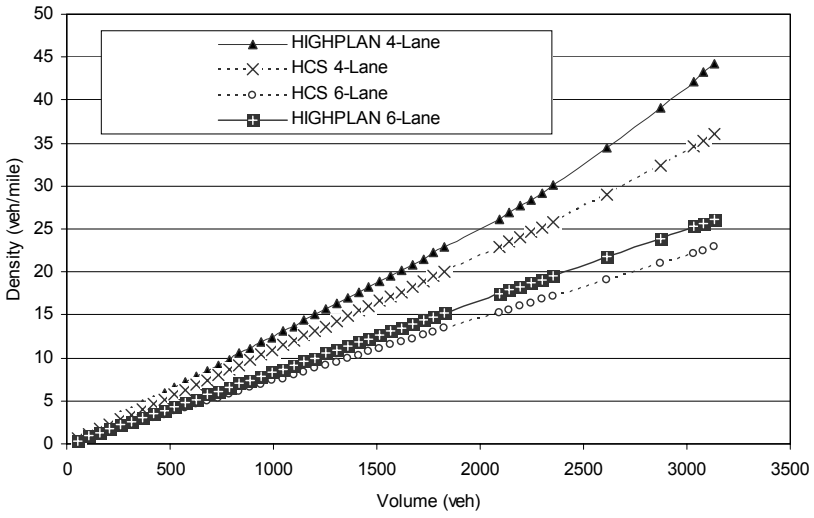


Figure 5-11. Density-Flow Rate Relationship

Figure 5-12 shows the relationship of free flow speed and density. The volume is set to be 1045 vehicles/hour, which is equal to an AADT of 20000. The Free Flow Speed is incremented by 5 miles/hour from 45 to 70 in the HIGHPLAN and by 1 mile/hour from 45 to 60 in the HCS-Multilane. Again, the HIGHPLAN lines for 4-lane and 6-lane roadway give higher densities than the HCS. The differences between the results are almost constant since the four lines are parallel in the figure.

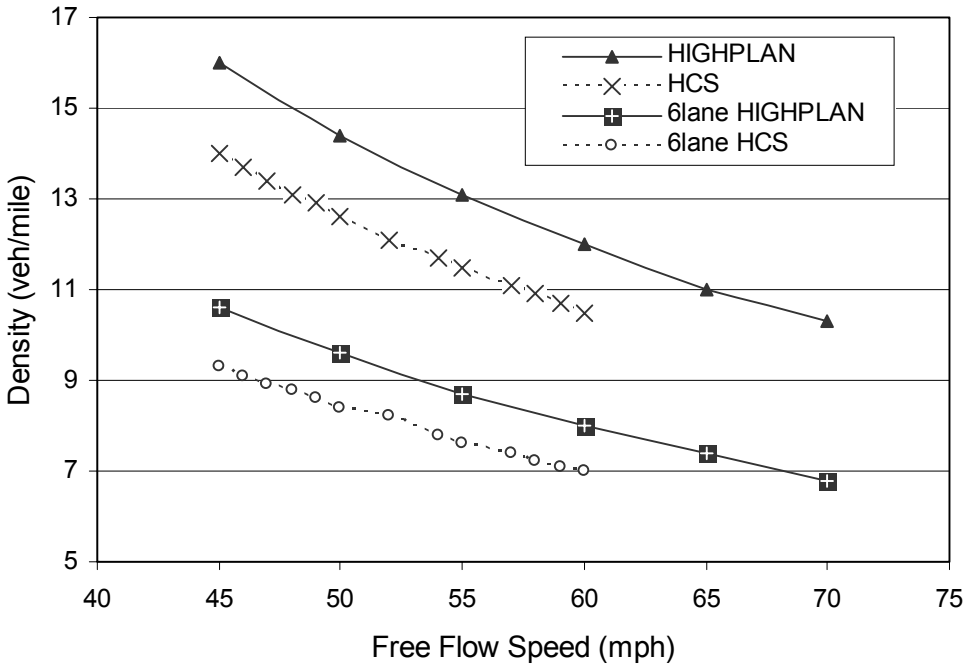


Figure 5-12. Density-FFS Relationship for Highway Facilities

Passing Lane Spacing-%FFS Relationship of two-lane roadways is shown in Figure 5-13. The %FFS decreases when the Passing Lane Spacing increases and it approaches *to the value* when Passing Lane Spacing is 0, which represents a no passing lane.

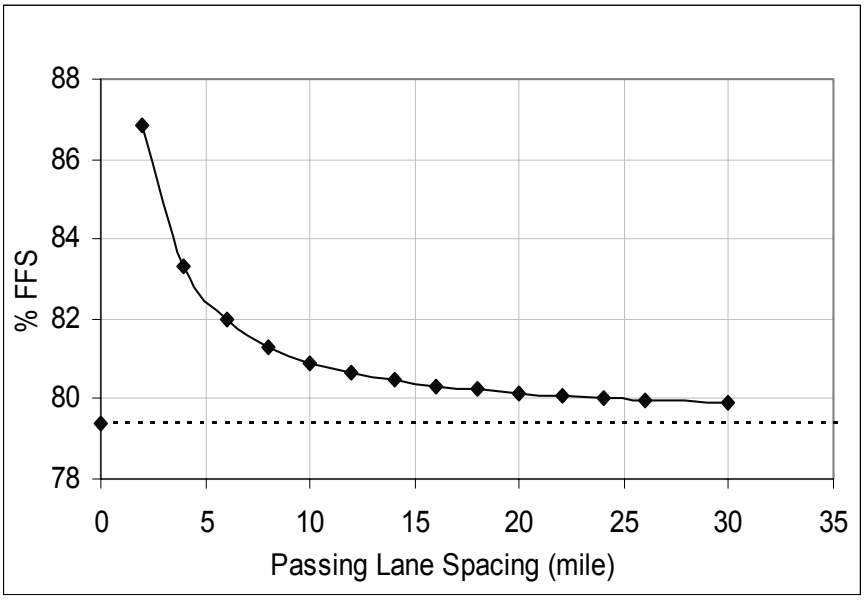


Figure 5-13. Passing Lane Spacing-%FFS Relationship of Two-Lane Roadways

The relationships between Passing Lane Spacing (PLS) and the threshold flow rate of each level of service were also tested and the results are shown in Figure 5-14. Volume values drop down dramatically when PLS is less than 5 miles and approaching to a straight line that represents the volume while in a no passing zone.

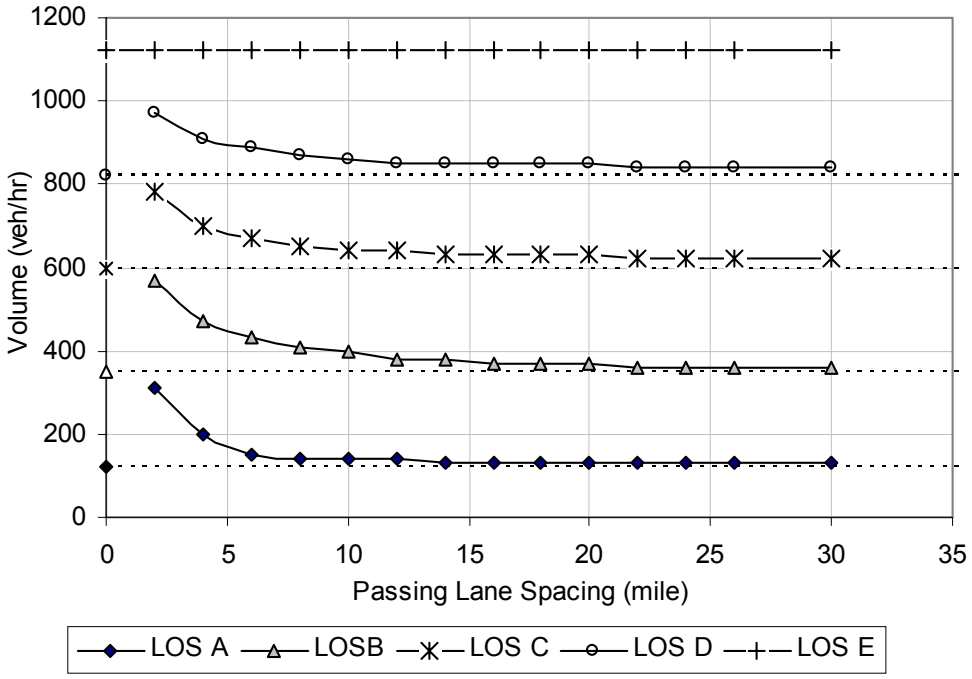


Figure 5-14. Passing Lane Spacing and the Threshold Flow Rate

To check the results of the comparisons above, look at the internal relationships that are evident in the two models.

The determination of free-flow speed in the HCS is based on field measurements or applying reductions to the ideal FFS [1]. Since HIGHPLAN is used for conceptual planning purposes, the posted speed for a highway plus 5 mph is used as an estimation of FFS.

A primary simplifying technique used by HIGHPLAN was the concentration on the through movement of traffic, while the impact of side road and left turning movements are handled generically [10]. Essentially, the concept is to get left turning vehicles out of the through traffic stream and reflect some level of detriment.

HIGHPLAN has chosen to use a local adjustment factor instead of $f_{p,}$ which is used in the HCS to account for potential capacity reduction due to unfamiliar drivers. The local adjustment factor is intended to account for all implicit capacity reducing effects that the analyst may consider important.

6 APPLICATION TO FREEWAY FACILITIES

Figure 6-1 shows the user's interface of the FPBDS. Up to 200 data sets can be generated on each run. The data files created are named as the prefix followed by a number. The Number of Segments (1~20) can be typed in the appropriate box on the FPBDS Main Menu Screen. Before the data set generation process begins, the types of each segment need to be assigned by user. Still at the FPBDS Main Menu Screen, click on the Assign Segment Type button to open the Segment Type Assignment Screen (Figure 6-2). Some combinations of segment types are not valid in FREEPLAN, but the data set generator is not presently able to check for this. The "Translate to HCS" button will not be enabled unless the Number of Segments is equal to 1.

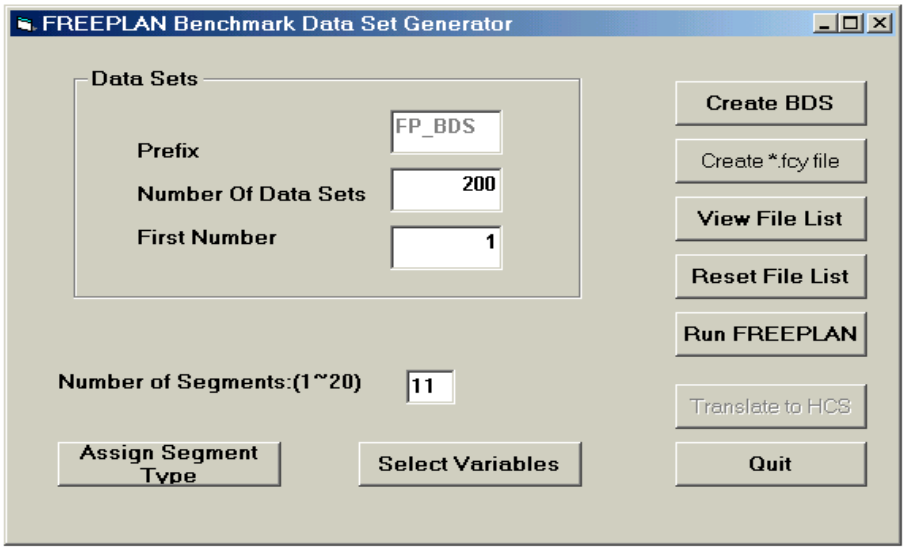


Figure 6-1. FPBDS User's Interface Screen

Clicking on the Select Variables button will open the Variable Selection Screen shown in Figure 6-3. The system will generate arbitrary numbers for the parameters that are selected as random. If a parameter is left in the fixed mode (the default value), the system will automatically provide a fixed value in a text box. The user can edit this value and the corresponding valid range will be displayed in the lower right corner of the screen when the text box is chosen. The parameters shown on the top left side of the Variable Selection Screen (Section Length, Posted Speed, Number of Lanes and Terrain) can be randomized individually. The bottom left portion of the screen shows the parameters for Ramp Variables (Ramp VPH, HvPct and Number of Ramp Lanes). These variables are capable of being randomized by Facility, Segments or Ramps with the default values being designated as On Ramps or Off Ramps. To make the hypothetical data

relate more closely to real situations, only one value is generated for each facility parameter and random adjustment factors are created for each segment. So the segment values, though randomly generated, are still consistent with one another in the whole facility.

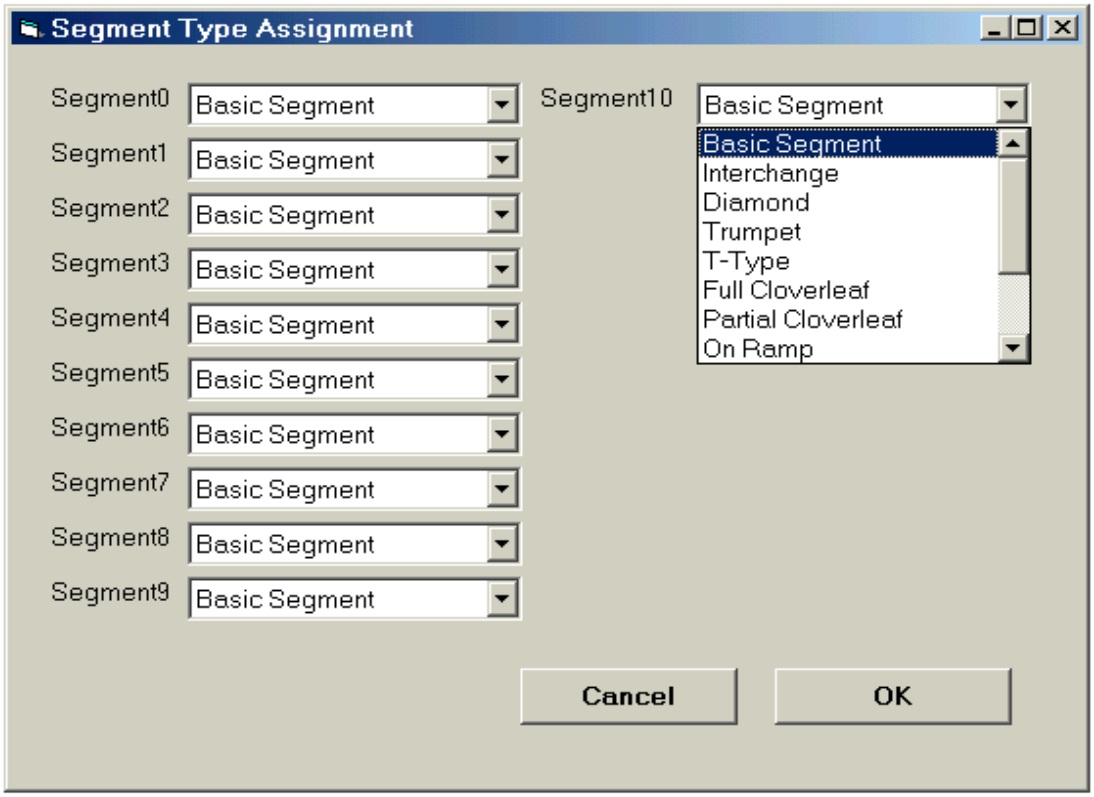


Figure 6-2. Segment Type Assignment Screen

The other function of the FPBDS program is to read in an XML file in ARTPLAN structure and transfer it to the HCS-Multilane or Two-lane structure. Multiple files can be translated at the same time and the letters "_hcs" will be added to the original file name to denote it as a translated file.

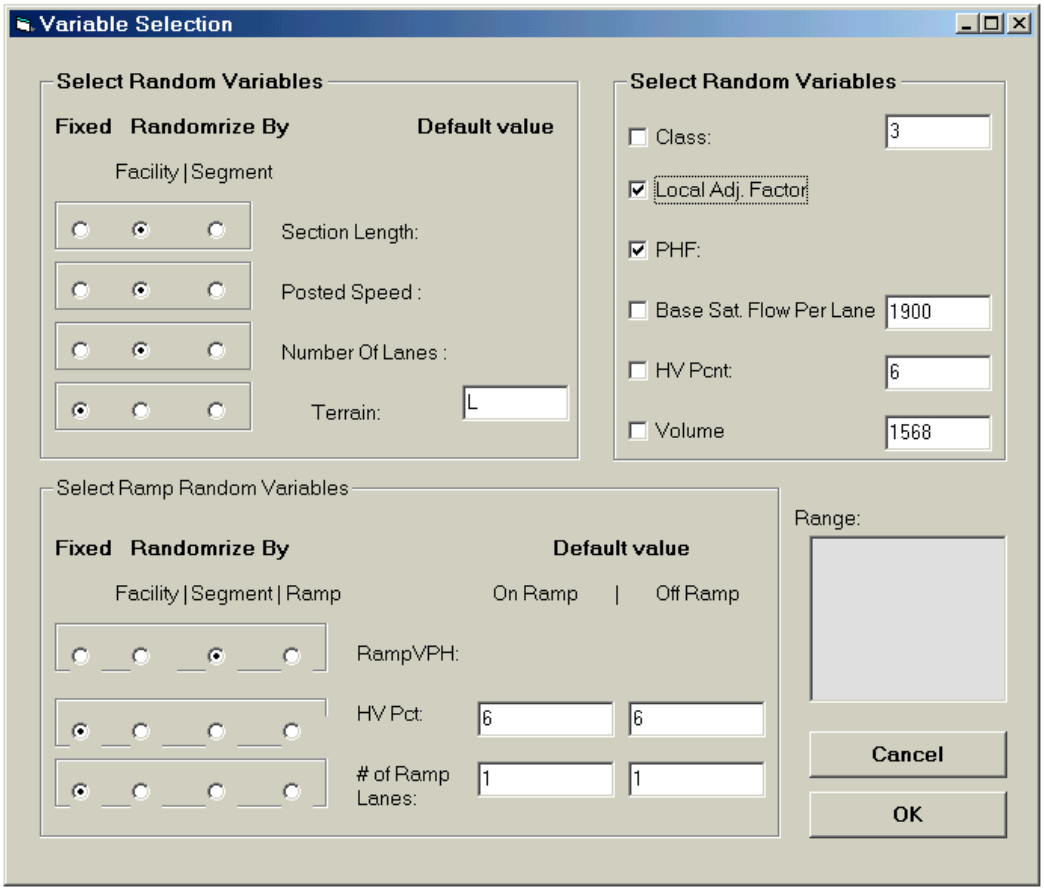


Figure 6-3. Variable Selection Screen

6.1 TMML STRUCTURE FOR FREEWAYS

The TMML class structure for freeway facilities is depicted in Figure 6-4. The individual data elements within each class are identified in Appendices A and B.

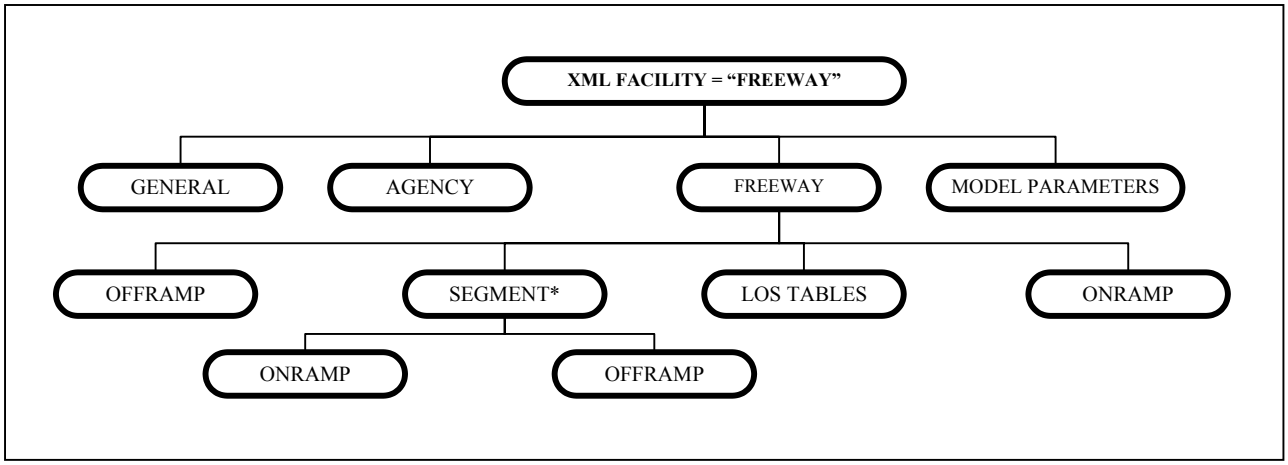


Figure 6-4. TMML Structure for Freeways

6.2 THE FREEWAY BENCHMARK DATA SET GENERATOR

The number of segments and each segment type are the first things that need to be assigned before the generation process can begin. Certain combinations of segment types are not valid in FREEPLAN, but at this time, the data generator is not able to check for this inconsistency. The "Translate to HCS" button will not be enabled unless the number of segments is equal to 1.

Table 6-1 shows the valid range and distributions of parameters in FPBDS. Fixed parameters are the same as those in APBDS. Because FREEPLAN is still under development and HCS-Freeway does not support XML format, the comparison of freeway facilities will not be covered in this paper. However, simple test for basic freeway segment will be done to show that the whole system works for freeways as well.

Table 6-1: Valid Range and Distribution in FPBDS

A. Properties Assigned by User

VARIABLES	RANGE	DISTRIBUTION
Number of sections	1~20	Assigned
Type	Basic Segment Interchange Diamond Trumpet T-Type Full (Partial) Cloverleaf On (Off) Ramp Major merge (Diverge) Toll Plaza	Assigned

B. Parameters Randomized by Facility:

VARIABLES	RANGE	DISTRIBUTION
Class	1 to 4	1 to 4 uniform distributed
AADT	5000 ~ 30000	Uniform 5000 to 300000 (integer) BY INTERSECTION: (Volume including AADT, K, D) Freeway Value times X X ~ (0.75, 1.25) uniform distributed
K Factor	0.07 ~ 0.12	20% uniform distributed 0.07 to 0.09 60% uniform distributed 0.09 to 0.10 20% uniform distributed 0.10 to 0.12
D Factor	0.52 ~ 1.0	75% uniform distributed 0.52 to 0.6 25% uniform distributed 0.6 to 1.0
PHF	0.75 ~ 1.0	25% uniform distributed 0.75 to 0.85 75% uniform distributed 0.85 to 1.0
% Heavy Vehicles	0 ~ 25	75% uniform distributed 0 to 10 25% uniform distributed 10 to 25
Local Adj. Factor	0.75 ~ 1	Uniform 0.75~1.0

C. Parameters Randomized by Sections:

VARIABLES	RANGE	DISTRIBUTION	By SECTION (RAMP)
Posted speed	50, 55, 60, 65,70	Class 1: 70 Class 2: 70 80% 65 20% Class 3: 55 10% 60 40% 65 40% 70 10% Class 4: 55	Freeway value adds a number randomly chosen from -10, -5, 0, 5, 10
Number of Lanes (one direction)	2, 3, 4, 5, 6, 7	4 6 8 10 12,14 30% 30% 20% 10% 10%	Freeway value adds a number randomly chosen from -1, 0, +1 in each direction
Section Length	Basic section >200 Interchange 2000~7200	Uniform distributed	Uniform distributed
Terrain	Level (L), Rolling(R), Mountainous (M)	Uniform distributed	Uniform distributed
Ramp VPH	50 to 1000	Uniform distributed	Uniform distributed
Ramp HvPct	0 to 25	Uniform distributed	Uniform distributed
Ramp Lanes	1 or 2	Uniform distributed	Uniform distributed

6.3 A FREEPLAN BDS EXAMPLE

The following step-by-step detailed example will illustrate how to apply benchmark data sets to test the average passenger car speed-flow rate relationship of basic freeway segments and compare the FREEPLAN results with the HCM Chapter 13 results.

1. Run FPBDS from Windows. The FPBDS Main Menu Screen will appear (see Figure 6-5). Set the Number of Segments: (1~20) as 1. Click on the button to open the Assign Segment Type (Figure 6-6).

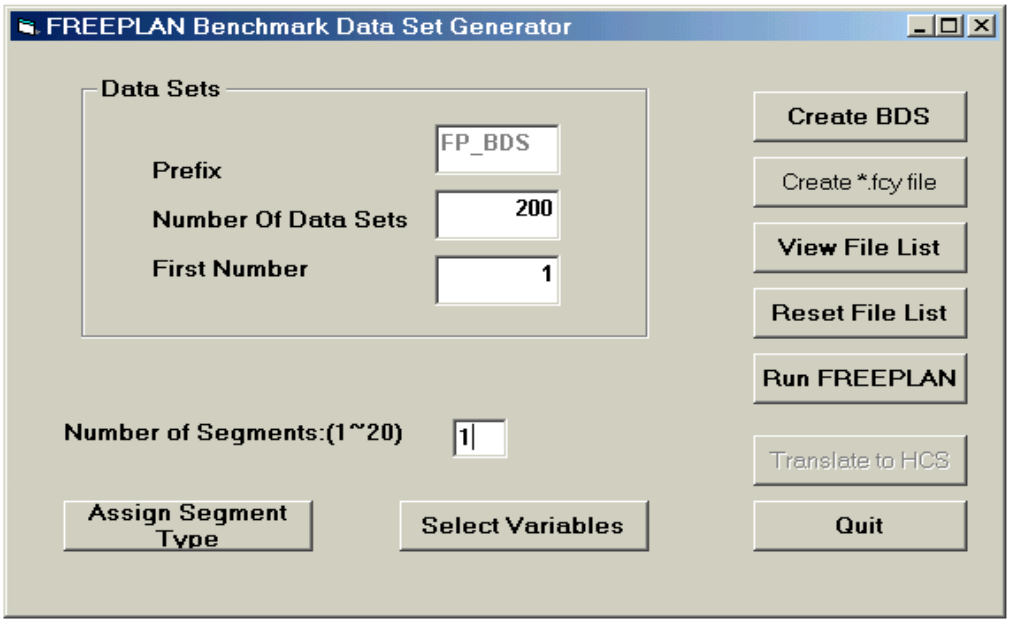


Figure 6-5. FPBDS Main Menu Screen

2. Choose Basic Segment for the Segment0 type and click the "OK" button to return to the FPBDS Main Menu Screen.

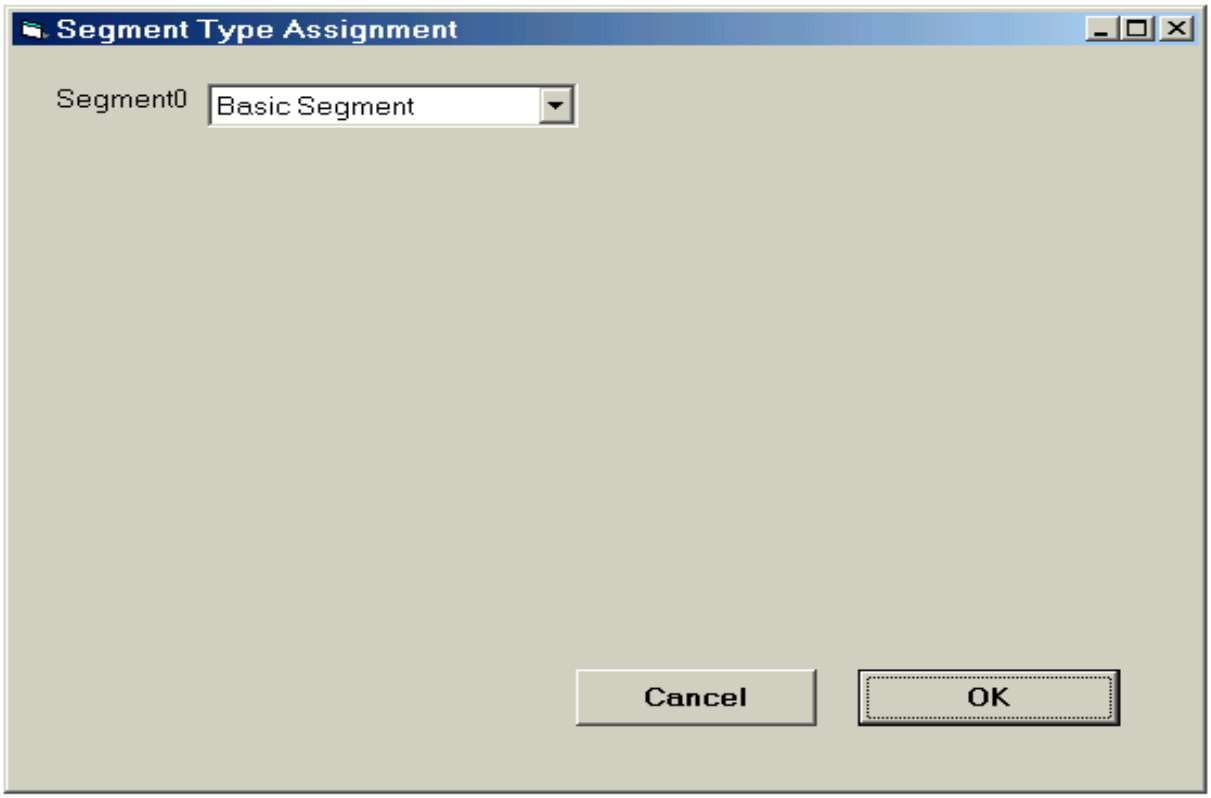


Figure 6-6. FPBDS Segment Type Assignment Table

3. Click on the Select Variables button at the FPBDS Main Menu Screen to open the Variable Selection Screen (Figure 6-7). In the Select Random Variables box on the upper right side of the screen, set the Volume to be random by checking the box. Type in the fixed values for remainder of the variables (Class through Hv Pcnt). Click the "OK" button to return to the FPBDS Main Menu Screen (Figure 6-5).

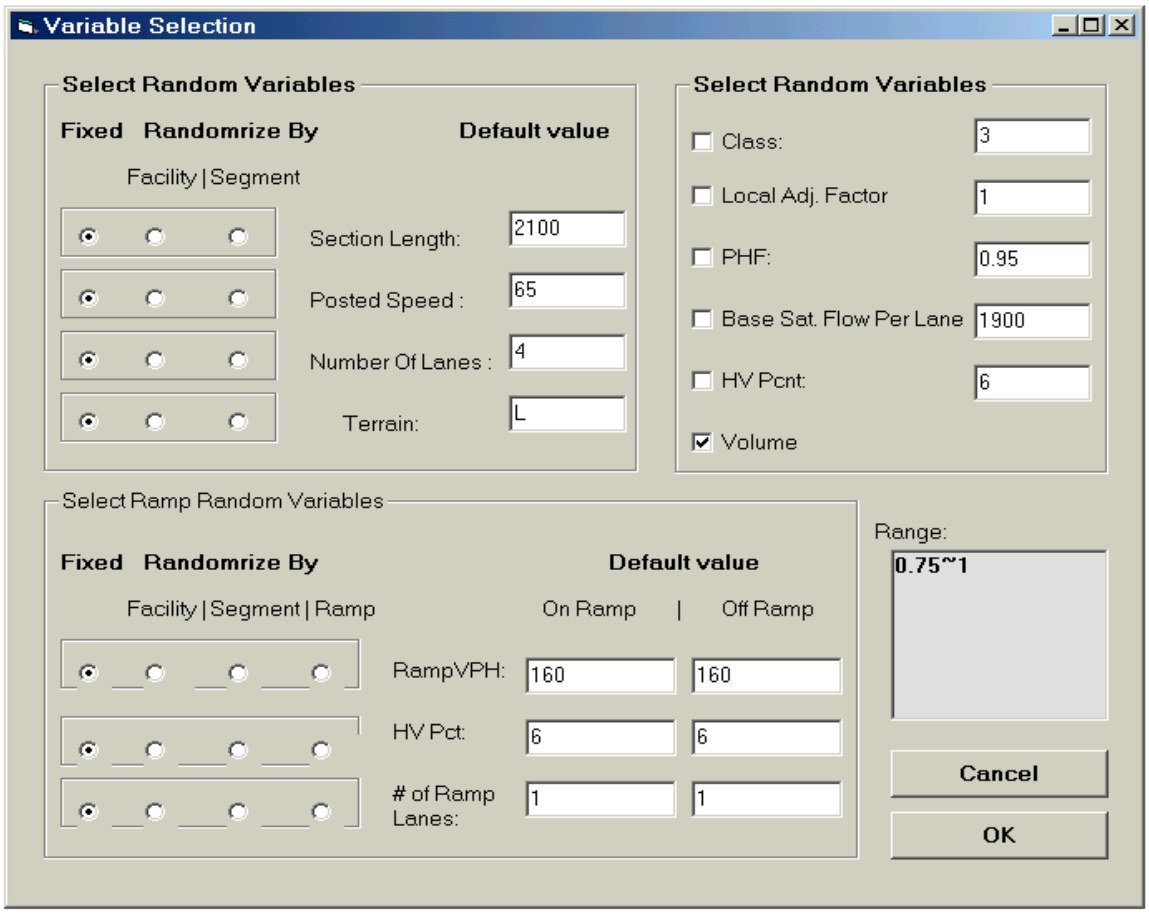


Figure 6-7. FPBDS Variable Selection Table

4. At the FPBDS Main Menu Screen, input the Number of Data Sets as "1". Click on the Reset File List button. The screen will appear as it does in Figure 6-8. Close this screen (File-Close.) At the FPBDS Main Menu Screen click on the Create BDS button to generate the data sets.

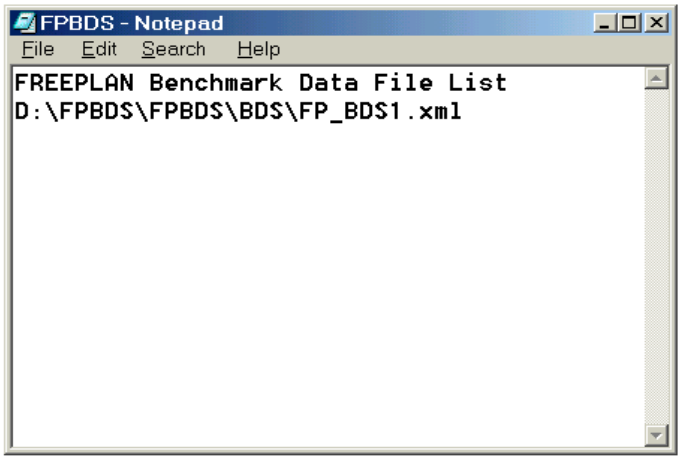


Figure 6-8. File List in FPBDS

5. To check the names and paths of the newly created data sets, click on the View File List button at the FPBDS Main Menu Screen (Figure 6-5). The Benchmark Data File List will appear as shown in Figure 6-8. The data files are stored in the BDS folder under the FPBDS directory. The file lists are saved under FPBDS.txt. Samples of the File List and Data Sets are shown in Figure 6-8 and Figure 6-9.

```
<?xml version="1.0" ?>
- <TMML Facility="Freeway">
  - <AGENCY>
      <Agency>UF-TRC</Agency>
    </AGENCY>
  - <GENERAL>
      <Date>12/17/2001</Date>
      <Analyst>BDS</Analyst>
      <PeriodID>K100</PeriodID>
      <ProgramVersion>1.01</ProgramVersion>
      <Comment>D:\FPBDS\FPBDS\BDS\FP_BDS4.xml</Comment>
    </GENERAL>
  - <FREEWAY>
      <FacilityName>Benchmark Road</FacilityName>
      <To>Not Specified</To>
      <From>Not Specified</From>
      <FwdDirection>Northbound</FwdDirection>
      <PeakdirectionYN_FP>Peak Direction</PeakdirectionYN_FP>
      <FreewayClass_FP>3</FreewayClass_FP>
      <NumberofSegments>1</NumberofSegments>
      <PostedSpeed>65</PostedSpeed>
      <AADT>9932</AADT>
      <KFactor_PLN>0.09</KFactor_PLN>
      <DFactor_PLN>0.54</DFactor_PLN>
      <DDHV>483</DDHV>
```

Figure 6-9. Sample of Data Sets Generated by FPBDS

6. Open the TMRC program and run the Traffic Model Results Comparison (TMRC) (Figure 6-10). For the box labeled Facility, select Freeway. In the Reference File box type in the path and name of the generated date file.

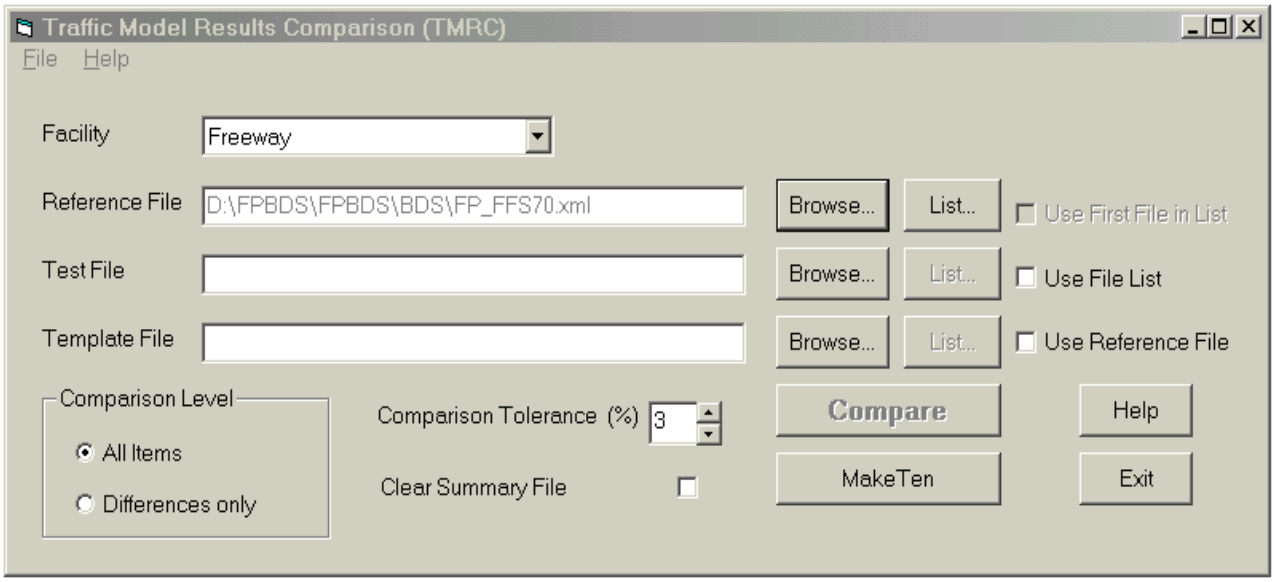


Figure 6-10. TMRC Main Menu Screen

7. Click on the List button at the end of the Reference File line to open the reference file in WordPad and type in the line.

```
<!-- MAKE10 @ 0 "7000" -->
```

Close the file (File-Close-Save [Yes]) and click on the Make Ten button at the TMRC Main Menu Screen (Figure 6-10). 10 files will be generated with the parameter AADT increasing by 7000 each time. These files will automatically be processed through FREEPLAN. A file list named make10list will be created in the TMRC working directory. (Figure 6-11.)

```
<?xml version="1.0"?>
<TMML Facility="Freeway">
   <AGENCY>
      <Agency>UF-TRC</Agency>
   </AGENCY>
   <GENERAL>
      <Date>12/17/2001</Date>
      <Analyst>BDS</Analyst>
      <PeriodID>K100</PeriodID>
      <ProgramVersion>1.01</ProgramVersion>
      <Comment>D:\FPBDS\FPBDS\BDS\FP_BDS4.xml</Comment>
   </GENERAL>
   <FREEWAY>
      <FacilityName>Benchmark Road</FacilityName>
      <To>Not Specified</To>
      <From>Not Specified</From>
      <FwdDirection>Northbound</FwdDirection>
      <PeakdirectionYN_FP>Peak Direction</PeakdirectionYN_FP>
      <FreewayClass_FP>3</FreewayClass_FP>
      <NumberofSegments>1</NumberofSegments>
      <PostedSpeed>65</PostedSpeed>
      <AADT>9932</AADT>
<!-- MAKE10 @ 0 "7000" -->
      <KFactor_PLN>0.09</KFactor_PLN>
```

Figure 6-11. TMRC Make10 Instruction File

8. Return to the TMRC Main Menu Screen as shown in Figure 6-12. Check the box for "Use File List". In the Test File List box, type in the path and name of the make10list file. Check the box for "Use First File in List."

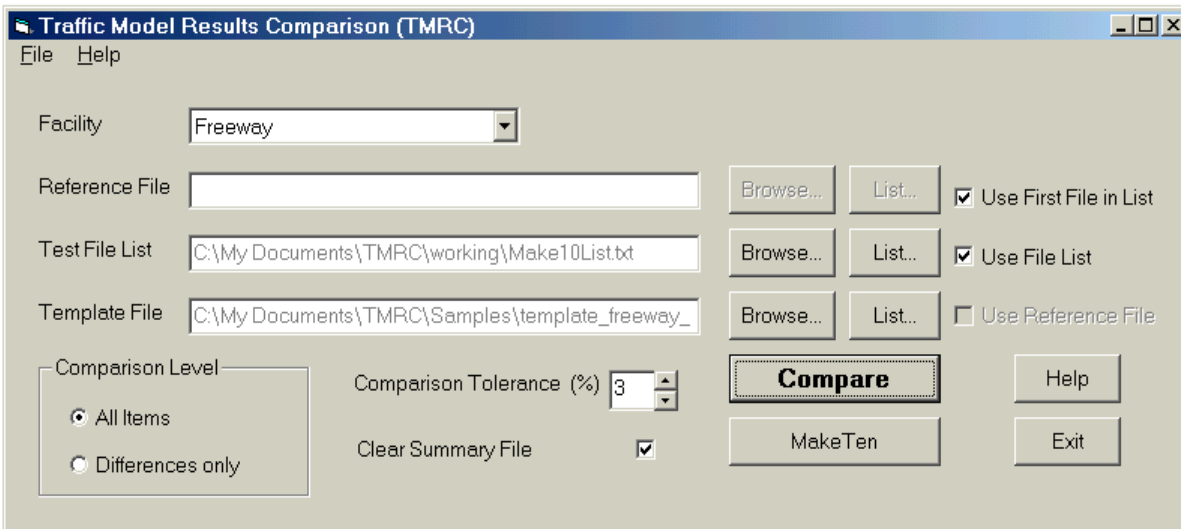


Figure 6-12. TMRC Main Menu Screen after Make Ten Process

9. Copy one of the data files to create a template according to the rules described in Chapter 3.1.2. Still working in the TMRC Main Menu Screen, type the path and name of the newly created file in the Template File box. Check the box for Clear Summary File and click on the Compare button to start multiple file comparisons. The template file is shown in Figure 6-13.

```
<!--  Freeway Description  -->
<FacilityName>Benchmark Road</FacilityName>
<To>Not Specified</To>
<From>Not Specified</From>
<FwdDirection>Northbound</FwdDirection>
<NumberofSegments>1</NumberofSegments>
<!--  Global Freeway data  -->
<PeakdirectionYN_FP>Peak Direction</PeakdirectionYN_FP>
<FreewayClass_FP>3</FreewayClass_FP>
<PostedSpeed>65</PostedSpeed>
<AADT>@01AADT</AADT>
<KFactor_PLN>K</KFactor_PLN>
<DFactor_PLN>D</DFactor_PLN>
<DDHV>@02DDHV</DDHV>
<PHF>@05PHF</PHF>
<HVPct>@04HvPct</HVPct>
<NumberOfLns>@03NumOfLanes</NumberOfLns>
<LocalAdjFactor>1</LocalAdjFactor>
<!--  Freeway Results  -->
<Speed>@07AverageSpeed</Speed>
<Density>@06Density</Density>
<LOS>@09LOS</LOS>
<VCRratio>@08VCRatio</VCRratio>
<!--  Default data for all on ramps  -->
```

Figure 6-13. TMRC Template File

10. The results of the comparisons will be saved in a spreadsheet format in a file labeled TMRCSum.txt under the working Directory of TMRC (see Figure 6-14).

TMRCSum - WordPad

File Edit View Insert Format Help

```
AADT, DDHV, NumOfLanes, HvPct, PHF, Density, AverageSpeed, VCRatio, LOS
16932, 823, 2, 6, .95, 6.373609, 70.00001, .1858969, A
23932, 1163, 2, 6, .95, 9.006691, 70, .2626952, A
30932, 1503, 2, 6, .95, 11.63977, 70, .3394934, B
37932, 1843, 2, 6, .95, 14.27286, 70, .4162917, B
44932, 2184, 2, 6, .95, 16.91368, 70, .4933158, B
51932, 2524, 2, 6, .95, 19.55015, 69.98788, .570114, C
58932, 2864, 2, 6, .95, 22.29563, 69.6365, .6469122, C
65932, 3204, 2, 6, .95, 25.36031, 68.48911, .7237105, C
72932, 3544, 2, 6, .95, 29.00956, 66.22717, .8005087, D
```

Figure 6-14. TMRC Summary File

11. Once the Summary File is saved, use Excel to open the file and create an Average Speed-Flow Rate Relationship chart for the FREEPLAN data. Results are shown in Figure 6-15.

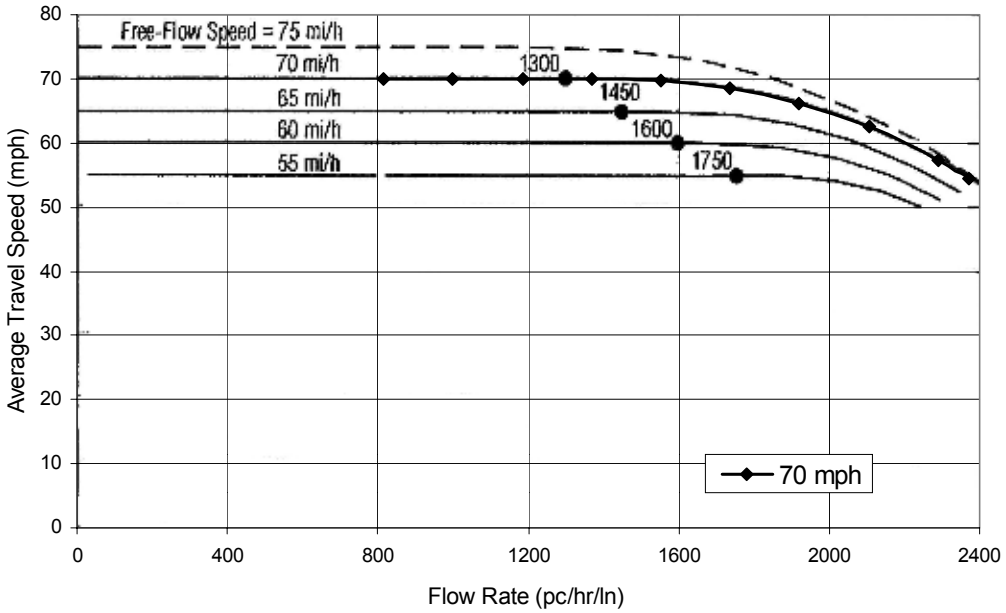


Figure 6-15. Average Speed-Flow Rate Relationship for Basic Freeway Segments

6.4 FREEWAY TESTS AND RESULTS

The density-flow relationship calculated by FREEPLAN shows results similar to the ones designed in the HCM Chapter 13 (Figure 6-16). However, FREEPLAN tends to gives higher densities when the flow rate increases.

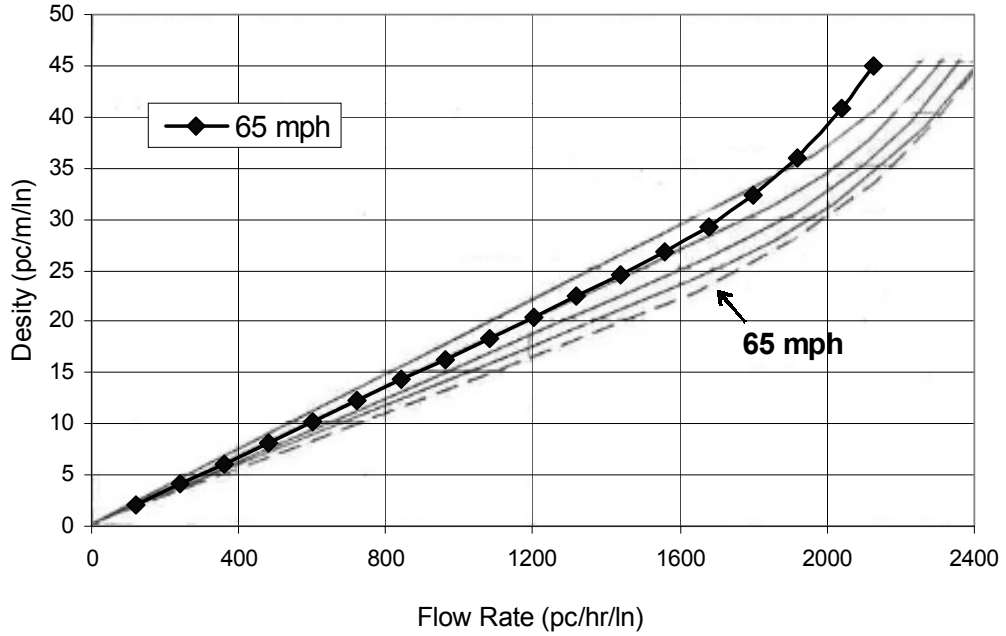


Figure 6-16: Density-Flow Relationships for Basic Freeway Segments

7 CONCLUSIONS AND RECOMMENDATIONS

This report presents the development and application of a model comparison and evaluation system based on benchmark data methodology and XML technique. Within the limits of the study, the following conclusions and recommendations are offered.

7.1 CONCLUSIONS

- Benchmark datasets methodology offers a practical alternative to the problem of model comparison. With a large number of data sets reflecting a wide range of conditions, it is quite efficient and reliable in establishing the similarities and differences between models, and to gain some insight into their merits and shortcomings.
- The tested model, LOSPLAN, is well suited to its intended application, which is planning level analysis of Arterial (and signalized intersection), Highway (two-lane & multilane), and Freeway (basic segment, merging, diverging, waving & interchange). It maintains fidelity to the HCM procedures to the extent that Florida conditions will allow and the users will accept.

7.2 RECOMMENDATIONS

- Due to the inconsistency in structures and tags, TMML does not really serves as a competent means of exchanging data between two systems now. Translations have to be made to support one model to read the TMML file from another. Some work needed to be done to take full advantage of XML based data storage and interchange.
- Some random generated data sets give combinations of the parameters, which will never happen in reality. Most times these data sets cannot be recognized by traffic models and will cause errors. It is recommended in future work that a diagnostic model can be developed to check the data sets.

REFERENCES

1. Laurent, S. S. XML A Primer (Second Edition)
2. Highway Capacity Manual. TRB, National Research Council, Washington, D.C., 2000.
3. Prassas, E. S., D. Mcleod. Arterial Planning Methodology – Concept, Implementation, and Experience. TRB Record—No. 1678.
4. Kim, J. T., K. G. Courage. Development and Application of A Traffic Model Data Structure Based on XML. No. C4-3411.
5. Prassas, E. S., D. Mcleod. Freeway Facility Planning Methodology: Concept and Implementation. TRB Record—No. 1678.
6. Prassas, E. S. Improving the Running Times in Highway Capacity Manual Table 11-4; Related Observations on Average Travel Speed. TRB Record—No. 1678.
7. Washburn, S. S., K. G. Courage and D. Mcleod. Adoption of the HCM2000 for Planning Level Analysis of Two-Lane and Multilane Highways in Florida
8. Quality/Level of Service Handbook (2001 Draft). Florida Department of Transportation.
9. LOSPLAN Documentation
10. XML-Based Specification for Traffic Software Data Interchange

APPENDICES

APPENDIX A

Traffic Model Markup Language (TMML)

Draft Specification
July 31, 2000

Transportation Research Center
University of Florida

TRAFFIC MODEL MARKUP LANGUAGE (TMML)

Without going into a long history, the eXtensible Markup Language (XML) is becoming increasingly popular as a method for:

- Exchanging data between incompatible data base management systems;
- Presenting data in a manner that can be readily absorbed by office productivity software; and
- Facilitating the presentation of data on the Internet.

For those who are not familiar with the term, XML is a method of encoding data in a text file whereby all data items are identified with their own tags. XML offers one method of producing "self-describing data" which, by definition, is free of arbitrary formatting constraints and proprietary controls.

As chronicled in a recent Scientific American article (recommended reading), XML is catching on rapidly as a means of transferring data between two systems or users who deal with the same data, but in different formats [1]. There have been specific vocabularies developed for statistics, mathematics, chemistry, and many other disciplines. As a further indication of widespread recognition, an edition of "XML for Dummies" may now be found in bookstores.

XML is a logical extension of HTML, the universal Internet language, with its own context-specific vocabulary. The "X" in many computer-oriented acronyms denotes "extended." In this case, it denotes "extensible," which differs from "extended" in the sense that you have to provide your own extensions. For example, a tag called <Volume> would be considered foreign to an HTML document. It would be quite acceptable in XML, but would not be useful unless its significance had been previously established. The purpose of TMML is to create a set of such tags and define their meaning.

In the traffic engineering field, the Traffic Model Markup Language (TMML) has been proposed to facilitate sharing of data between traffic modeling software products. As envisioned here, TMML will be a fully XML-compatible markup language prescribing the class structure and data element tag names required to represent traffic model data in a "self-describing" format.

Traffic models deal with similar input and output data. Therefore, for most software products, it should be possible to import or export a large part of any data set with minimal processing logic. On the other hand, each software product has unique definitions and structures for representing the data. Therefore, it is not possible to create a universally understood TMML specification that would accommodate all software products directly. The specification described herein covers all of the data commonly used by a set of publicly developed software products that deal with signalized intersections and arterial systems.

To avoid setting rigid and prescriptive requirements, alternatives that encompass a broad range of current practice are offered. No software product is likely to recognize all of the data elements contained in this specification. Each product that offers TMML connectivity will require a

programmer interface document, which identifies the data elements that are recognized and any conditions that apply to their interpretation.

The purpose of this specification is to identify and assign specific tags to data items that are commonly used by traffic models. It is not intended to be a dictionary of terms and definitions. TMML simply provides the schema for transferring data between traffic models. It is not intended as the foundation of a universal database. It is important that each model's definition of each data item be understood in the data transfer process.

No specific units or system of units are implied in any of the tags that constitute the TMML vocabulary. For example, total delay may be expressed in vehicle-seconds, vehicle-minutes or vehicle-hours, depending on the individual software product. It is essential that programmers understand the interpretation that each software product places on all data items.

Rules, Conventions and Guidelines

The following rules, conventions and guidelines constitute a preliminary specification:

1. TMML is a fully XML compatible markup language intended for transferring data between traffic model applications and for creating output data in a format that is easily rendered by office productivity products.

2. All TMML data files shall have the XML file name extension, so that they will be recognized by common software applications. (e.g., Microsoft Office, Word Perfect, Internet Explorer, Visual FoxPro and programming languages such as Visual Basic, C++, J++, etc). A prologue, containing XML processing instructions and data type declarations shall precede the root tag. The root tag shall be <TMML Facility = "*FacilityType*">.

3. The TMML language will be defined in terms of a collection of document type definition (DTD) files that describe the structure and vocabulary to completely define a data set for various facilities and software products. Pending further developments within the profession, Mc*Trans* will serve as a repository for the collection of DTD files that define TMML. A list of accepted abbreviations for tags, as discussed later, will also be included in the definition of TMML.

4. The TMML class and data element structure for arterials and intersections is described in detail in Attachment 1

5. The Traffic Software Data Dictionary (TSDD) shall be the authoritative reference for the vocabulary and tags identifying classes and attributes. Where a required element is contained in the TSDD, it shall be implemented with the same format, including spelling, capitalization, etc. Where a required element is not contained in the TSDD, it shall be created following the general principles of style as used by the TSDD. Class elements shall be represented in upper-case characters (e.g., APPROACH). Data elements within each shall be represented as a concise series of connected words with lower-case characters and initial capitals (e.g., MedianWidth).

6. Abbreviations (e.g., MedWdth) should be avoided. However, abbreviations and symbolic representations are appropriate when they are widely used and recognized (e.g., PHF), and when they are associated with the terminology defined in the literature for a specific model. A list of abbreviations is contained in Schedule A. Abbreviations should be applied consistently, and Schedule A should be consulted when new tag names are created.

7. IEEE Standard 1489 [2] shall be followed in the development of names for classes and attributes that are not represented in the TSDD.

8. The XML format is well suited to the storage and retrieval of data for traffic model software. It is expected that XML will be used extensively for this purpose. Data for specific models may include the results of intermediate computations in addition to inputs and outputs (e.g., the supplemental worksheets for the HCM signalized intersection procedure). These data items should not be subject to the need to conform to any standards, nor should they be considered as a part of the TMML specification. The tag names for such items shall end with the underscore character "_" followed by an optional string to indicate the specific model to which they apply (e.g., EL1_HCM). This will provide for an XML-compliant representation of a full data set without requiring extensive coordination of the TMML vocabulary. The underscore character shall not be used for any other purpose. It is the only non-alphabetic character allowed in XML tags.

9. The order of presentation of classes or elements within a class shall not be prescribed for a specific software product. It must be possible to present the classes and elements within a class in any order as long as the class structure is maintained. The same class may appear more than once in a file (e.g., to separate input and output data). A given data item may appear only once in any class otherwise an unresolvable ambiguity will be created.

10. The rules of XML permit data items to be represented either as attributes of a defined class, or as elements within the class; e.g.:

```
<APPROACH Direction = "EB">
                or
<APPROACH>
    <Direction>EB</Direction>
etc.
```

    For purposes of TMML, metadata should be represented as an attribute and all other data items should be assigned their own tags. In the above example, the direction would be metadata because it is the only information that distinguishes one approach from another. In other words, it is "data about the data," as opposed to information per se. All class tags that appear more than once in a data set (e.g. APPROACH) shall include an attribute value that serves as an index to identify where the data item should be stored when it is read.

11. Each class tag shall have a maximum of one attribute to facilitate reading and parsing of data. That attribute shall distinguish the specific instance of the class from all other instances of the same class (e.g., INTERSECTION ID = “3”).

References

1. Scientific American, May, 1999, pp.89-93, Jon Bosak and Tim Bray, “XML and the Second-Generation WEB.”

2. IEEE P1489, Institute of Electrical and Electronics Engineers, Inc., “Draft Standards for Data Dictionaries for Intelligent Transportation Systems—Part 1 Functional Area Data Dictionaries.”

Schedule A: Recognized Abbreviations for TMML

AADT	Average Annual Daily Traffic
Adj	Adjusted or adjustment
ADT	Average Daily Traffic
Avg	Average
AWSC	All-Way Stop Control
BOQ	Back of Queue
BW	Progression bandwidth
Cap	Capacity
DDHV	Directional Design Hour Volume
DHV	Design Hour Volume
DI	Disutility Index
EB	Eastbound
Eff	Effective
EW	East-West
Excl	Exclusive
FDW	Flashing Don't Walk
Fwd	Forward.
GCRatio	The ratio of effective green time to cycle length
HOV	High-occupancy Vehicle
Hr	Hour(s)
HV	Heavy vehicles (eg., trucks and buses)
ID	Identification
LG	Lane group
Ln	Lane
LOS	Level of Service
LT	Left turn
Max	Maximum
Min	Minimum
MOE	Measure(s) of effectiveness
Movt	Movement
NB	Northbound
NS	North-South
Pct	Percent
Ped	Pedestrian(s)
PF	Delay adjustment factor for progression quality as defined in the HCM
PHF	Peak Hour Factor
PI	Performance Index
PPV	Persons per vehicle
Prop	Proportion
Pros	Progression opportunities (T7F)
Rev	Reverse or reversed
RT	Right turn
RTOR	Right Turn on Red

SatFlow	Saturation Flow Rate
SB	Southbound
Sec	Second(s)
Thru	Through
TMC	Turning movement count
TT	Travel time seconds, minutes, etc
TWSC	Two-way Stop Control
UnAdj	Unadjusted
VCRatio	The volume to capacity ratio of a movement, lane group, approach, etc.
Veh	Vehicle
TD	Travel Distance (veh-mi, veh-km, etc.)
Vol	Volume or flow rate
WB	Westbound
YN	Indicates a binary (Yes or No) condition when appended to the end of a tag

Attachment 1: TMML Structure and Notation for Arterial Data

The structure for TMML arterial files shall conform to the following hierarchy:

```
<TMML Facility = "Arterial">
<GENERAL>
</GENERAL>
<AGENCY>
</AGENCY>
<ARTERIAL ID=#>
        <INTERSECTION ID=(# or All>
                <TMC Begin=(Time)>
                </TMC>
                <CONTROLLER>
                </CONTROLLER>
                <TIMINGPLAN Phase=(# or All>
                        <PHASECODES Approach=(# or Code)
                        </PHASECODES>
                </TIMINGPLAN>
                <APPROACH ID=(#, Code or All)>
                        <LANEGROUP ID=(#, Code or All)>
                        </LANEGROUP>
                        <LANE ID=(#, Code or All)>
                        </LANE>
                        <MOVEMENT ID=(#, Code or All)
                        </MOVEMENT>
                        <ODMATRIX>
                        </ODMATRIX>
                </APPROACH>
        </INTERSECTION>
        <MOEGROUP ID=(# or Code)
        </MOEGROUP>
<MODELPARAMETERS>
</MODELPARAMETERS>
</ARTERIAL>
</TMML>
```

This structure provides the framework for present and future traffic model software. At this time, TMML processing software exists only for the Arterial and Intersection facilities. Specific examples of processing software and the specific limits that they impose are described in separate programmer interface documents.

TMML is intended to be open-ended with respect to its design to encourage a wide range of software development for using and exchanging data between traffic model software products. However, each piece of TMML-compliant software will have its own limitations with respect to the number of intersections, approaches, movements, etc. that it will accommodate and the range

of elements and tags that it will recognize. Specific software products may also impose rules regarding structure and interpretation of the various tags. Current software conforms generally to the following notation and guidelines:

1. Missing data elements are treated as null values. Internal defaults will be applied if appropriate. Otherwise numerical data will be set to zero and null strings will be set for character data.

2. Class or data element tags that are not recognized by a specific program are ignored.

3. Data elements will apply to the class in which they are placed. The significance of elements that apply to sub approach classes (MOVEMENT, LANE and LANEGROUP) that are placed at a higher level may require interpretation by the individual programs. For example, <Vol> (a recognized abbreviation for "volume") placed at the APPROACH level would logically be interpreted as the volume for the whole approach. On the other hand, <SatFlowPerLn> at the approach level would logically be interpreted as a value which applies to each lane on the approach. (Note that "SatFlow" and "Ln" are both defined as recognized abbreviations in Schedule A). The <PHF> tag in the ARTERIAL class would logically be interpreted as a default value wihch applies to all movements on all approaches at all intersections. A different PHF value placed in an APPROACH class would logically override the default value from the higher level.

4. To avoid ambiguities, data items that apply to all instances of a child class within a given parent class should be placed in the child class using the “All” attribute. For example, suppose the yellow time is constant at 4 seconds throughout the whole arterial. According to the TMML structure, the <Yellow> element belongs in the TIMINGPLAN class, and has no logical interpretation outside of that class. The representation that conforms fully to the TMML structure would be:

```
<ARTERIAL>
    <INTERSECTION ID="All"
        <TIMINGPLAN Phase="All"
            <Yellow>4</Yellow>
            etc.
```

   A given program could choose to interpret a TIMINGPLAN tag placed directly in the ARTERIAL class as applying to all intersections, or it could reject the tag as non-conforming. It could even interpret a <Yellow> tag in this class as applying globally. This is an example of a detail that must be covered in the Programmer Interface Document.

5. The class placement of program outputs and measures of effectiveness (MOEs) will generally have a different significance than the placement of input data. For example, <ControlDelay> placed at the MOVEMENT level would naturally apply to the specific movement. However, when placed at the INTERSECTION level, it would logically apply to the intersection as a whole. The “All” attribute should have little or no application to program outputs.

6. When sub-approach level MOEs appear at higher levels, it may be necessary to clarify their interpretation. The interpretation alternatives are

    - An overall average of the lower level measures (e.g., control delay, sec/veh)
    - The sum of all lower level measures (e.g., total delay, veh-hrs)
    - The critical value among the lower level measures (e.g., v/c ratio)

    In most cases the nature of the MOE will suggest a logical meaning for higher level representations, but there may be a need to clarify this detail in some cases (e.g., LOS).

7. One of the major problems of standardization is the interpretation of non-numeric data, especially multiple choice alternatives. Fortunately, there are very few of these data types. Multiple choices have been eliminated from TMML wherever possible by using a set of binary (YN) choices. Conventions have been suggested for all multiple-choice specifications. Where alternatives are described by text, the representation "F+", where "F" is the first character in the string, has been used to indicate that any string beginning with F (non-case sensitive) would be interpreted in the intended manner. For example the <ControlMode> may be specified according to the following mutually exclusive choices:

P+	Pretimed
A+	Actuated
S+	Semi-actuated
2+	2 way stop control
4+	4 way stop control

    This avoids the need for precise formulation and spelling in multiple-choice character strings.

8. Numeric representation of multiple choices has been used only for program-specific data items and in each case the underscore character has been applied. For example, <LTTreatment_AAP> gives five mutually exclusive choices of left-turn treatments specified numerically as defined and used within the AAP. This type of representation will normally be used for program-specific data storage and retrieval instead of inter-model data transfer.

The data elements for each arterial and intersection traffic model class will now be described in detail. A table will be provided for each class that identifies the tag names with brief descriptions where necessary. When a specific tag name is self-explanatory no description should be necessary. Descriptions are also omitted for model specific data with tag names containing the underscore character.

Description of TMML Classes and Data Elements

The GENERAL Class

This class contains elements that describe the file itself, as opposed to the traffic data within the file. It is good practice to place this class at the top of the file, because it identifies the source of the data, units of measurement etc. Processing software should not require the GENERAL class to be at the beginning, but should make a preliminary pass to establish the parameters of the data.

GENERAL (No attribute)	
FileName	File Name
Program	Program Name
Version	Program Version
Units	Units (U+ or M+)
Date	Date That the File Was Generated
District	Label
ProjectID	Label
Comment	Label
Analyst	Label
City	Label
PeriodID	Label

The AGENCY Class

This class contains a collection of elements that describe the user of the program. Different programs will use these elements to different extents.

AGENCY (No Attribute)	
RegistrationCode	All elements in this class are labels, used in different combinations by various programs
UserName	
AgencyName	
Address1	
Address2	
Address3	
City	
State	
Country	
PostalCode	
Phone	
Fax	
Email	

The ARTERIAL Class

All of the data items for an arterial facility will be contained in this class. If the processing software accommodates more than one arterial, the attribute value indicates which arterial is being described. While network analysis programs such as PASSER 4, TRANSYT-7F and CORSIM accommodate multiple arterials, the scope of current processing software is limited to a single arterial. Therefore, the attribute value for the <ARTERIAL> tag must be "1."

ARTERIAL ID=(Arterial #)	
ArterialName	
FwdDirection	Direction of increasing intersection sequence number
RevDirection	Direction of decreasing intersection sequence number
ArterialClass_HCM	As defined by HCM
SignalsPerMile	
KFactor	Ratio of peak-hour volume to daily volume
DFactor	Proportion of two-way traffic in the peak (heavier)direction
DHV	Design hour volume
DDHV	Directional design hour volume
NumberOfIntersections	
NumberOfSections	
TotalLength	
ADT	Average daily traffic
AADT	Average annual daily traffic
PI	
BWEfficienciencyFwd	BWEfficiency = Progression band width / cycle length
BWEfficiencyRev	
BWEfficiencyAvg	
BWAttainabilityFwd	BWAttainability = Progression band width / Shortest green time
BWAttainabilityRev	
BWAttainabilityAvg	
InterferenceFwd	Interference and Pros are defined in the TRANSYT-7F documentation
InterferenceRev	
ProsFwd	
ProsRev	
ProsAvg	
WeightedPros	
OversaturatedLinks	
QueueSpillback_T7F	
PctTimeJammed_T7F	

The INTERSECTION Class (Child of ARTERIAL)

The Facility = "Arterial" processing software is limited to a single arterial facility. Use attribute "All" to set default values for all intersections in subsequent child classes.

INTERSECTION ID=(Intersection #)	
CrossStreetName	
IntersectionName	
EWStreetName	
NSStreetName	
ControlMode	P+, A+, S+, 2+, 4+
CriticalX_HCM	Weighted average g/C ratio for the critical movements
CriticalY_HCM	Sum of the flow (v/s) ratios for the critical movement of each phase
MinDelayCycle_P4	The cycle length for isolated delay minimization, disregarding progression
NodeNumber	
XCoordinate	Referenced to an origin at the western limit
YCoordinate	Referenced to an origin at the southern limit
ProgressionGroup_T7F	As defined by TRANSYT-7F
BasicLayout_AAP	As defined by the AAP
Orientation_AAP	N+, E+ indicating NS or EW

The CONTROLLER Class (Child of INTERSECTION)

This class contains all of the data elements that apply to the controller as a whole. There is no attribute for this class because there is only one controller per intersection. Some of the data elements (e.g., CycleLength) may apply to all intersections. If the processing software does not recognize the CONTROLLER class as a child of the ARTERIAL class, it will be necessary to create an INTERSECTION class with ID = "All" in the ARTERIAL class.

CONTROLLER (No Attribute)	
CycleLength	
Phase1Movt	The entire phase plan can be derived from this for dual ring control
Offset	
OffsetReference	Reference Phase # or First green, Simultaneous green, Barrier (F+, S+, B+)
Standard	Nema or 170, if dual Ring
CoordinatedYN	
YieldPoint	NEMA Definition
NumberOfRings	1 or 2 (I.e., single or dual ring) This will control the subsequent processing of the timing plan data

NSOverlapYN	If Overlap is not allowed, then both opposing left turns must have the same phase lengths. Applies to single-ring operation only
EWOverlapYN	
SOP_AAP	Standard operating plan used in Florida
PhaseRev12	The phase reverse feature is a non-NEMA extension that provides for lagging left turns. It is found on many controllers and is recognized by CORSIM
PhaseRev34	
PhaseRev56	
PhaseRev78	

The TIMINGPLAN Class (Child of INTERSECTION)

This class contains all of the information required to describe one phase of the timing plan at an intersection.

TIMINGPLAN Phase = (Phase # or All)	
Green	Specified interval times
Yellow	
AllRed	
PedWalk	
PedFDW	
PhaseTimePct	Sum of all intervals in the phase
PhaseTimeSec	
MinGreenTime	
MaxGreenTime	
VehRecall	Recall to Min or Max
PedRecallYN	
ForceOffTime	
DetectorLength	
DetectorSetback	
DetectorDelay	
DetectorCarryover	
PermittedMovts	(List)
ProtectedMovts	(List)
CoordinatedYN	

The PHASECODES Class (Child of TIMINGPLAN)

The representation of signal phasing differs widely among traffic modeling software products. This class provides options to accommodate the schemes used by the common products. It may be omitted if the <PermittedMovt> and <ProtectedMovt> lists are supplied in the TIMINGPLAN class. Since this information is very specific to approaches and phases, it has no logical interpretation in other classes.

PHASECODES Approach = (# or Code)	
LTCode_HCS	P=Pretimed, A=Actuated
ThruCode_HCS	
RTCode_HCS	
PedCode_HCS	X = Peds moving in conflict with right turn
SignalIndication_AAP	L,T,R for protected movements, G for permitted left, U for unopposed
ControlCode_TRF	(1-9) Per TSIS Manual RT 36

The APPROACH Class (Child of INTERSECTION)

This class contains the approach-level data for each of four approaches to an intersection accommodated by current processing software.

APPROACH ID = (# or Code)	
ArterialClass_HCM	
LTTreatment_AAP	0=None, 1=Permitted, 2=Protected, 3=Prot/Perm, 4=Not opposed
LTPermittedYN	
LTProtectedYN	
LTBayLength	
RTTreatment_AAP	0=None, 1=No RTOR, 2=RTOR,3= LT overlap, 4=Free
RTBayLength	
AddedLnLength	Length of Added nearside thru lane (Short lane)
DroppedLnLength	Length of dropped farside thru lane (Short lane)
FreeFlowSpeed	
RunningSpeed	
RunningTime	
Section	
QueueClearance_P4	
PctTurnExclLn	Percentage of the total volume that turns from exclusive lanes.
GradePct	
ParkingLeftYN	
ParkingManeuversLeft	
ParkingRightYN	
ParkingManeuversRight	
PedVol	
PedSpeed	
PedDistance	
PedMinTime	
MidBlockEntry	
ThruLns_HCS	
RTORYN	
RTORVol	
Jumpers	Number of left turn vehicles released at the beginning of green
Sneakers	Number of left turn vehicles released at the end of green
ThruCap	
MedianWidth	
UpstreamNode	Node numbers used by CORSIM. Required information for each approach
OpposingNode	
NodeAhead	
NodeToLeft	
NodeToRight	

ReceivingLns	Number of receiving lanes
D2I_HCM	
CrossWalkWidth	Width of crosswalk

The attribute code for the approach may be:

- A number in the range 1 to 4 (indicating NB, SB, EB or WB)
- A directional designation (N+, S+, E+, W+) identifying the northbound, southbound, eastbound or westbound approach, respectively. For data storage purposes, the northbound approach will be designated as number 1, the southbound as 2, etc.

A directional designation beginning with the letter F or R, identifying the forward or reverse arterial direction. If this designation is used, the arterial directions must be specified by a <FwdDirection> or <RevDirection> tag under the <ARTERIALINFO> class. This option is intended for use in arterial planning applications in which the cross street operations are not modeled.

Sub-Approach Classes

Different modeling process and software products employ different structures to partition the traffic flow on an approach to an intersection. Three different sub-approach classes have been created to accommodate all of the traffic models:

The LANEGROUP Class (Child of APPROACH)

A lane group is a set of one or more lanes carrying a homogeneous mixture of movements. Three lane groups may be included on each approach in current software. The code designating the lane group may be:

- A number in the range 1 to 3
- The corresponding code L+, C +or R+, designating the left, center or right lane group, respectively.

Lane groups will normally be used for models that represent homogeneous flows on individual links (e.g, TRANSYT, HCM, etc.).

The MOVEMENT Class (Child of APPROACH)
Three movements may be included on each approach in the <MOVEMENT> class. The code designating a movement may be either a number in the range 1 to 3, or the corresponding code letter L+, T+, or R+.

The LANE Class (Child of APPROACH)

Some traffic models deal approach flow on a lane-by-lane basis. No current TMML processing software recognizes individual lane movements, so this class is reserved for future expansion and

development. Note the <Lane1Reference> tag in the MODELPARAMETERS class. This element indicates whether the lanes are numbered from left to right or from right to left.

Since most of the sub-approach data elements are common to all of the sub-approach classes, they are presented in a single table.

Sub-Approach Classes	
MOVEMENT ID = (# or Code)	
LANEGROUP ID = (# or Code)	
LANE ID = (# or Code)	
Movts	Movements permitted in the class (Combination of LTR)
Vol	
UnAdjVol	
AdjVol	
Peak15Vol	
PHF	
IdealSatFlow	
SatFlowPerLn	
NumberOfLns	
LnWidth	
SatFlow	
AdjSatFlow	
HVPct	
ArrivalType	
UnitExtension_HCM	
InitialQueue	
StartUpLostTime	
EffGreenExtension	
NumberOfBuses	
PropLT	
PropRT	
fW_HCM	
fHV_HCM	
fP_HCM	
fBB_HCM	
fA_HCM	
fG_HCM	
fLU_HCM	
fLT_HCM	
fRT_HCM	
VCRatio	
GCRatio	
VSRatio	
UniformDelay	
PF_HCM	Progression Factor as defined by the HCM

Cap	Capacity
D2K_HCM	
IncrementalDelay	
OtherDelay	
ControlDelay	
LOS	LOS based on control delay
SegmentLOS	LOS based on average speed in the segment
CriticalYN	
PctLTUsingSharedLn	
PctRTUsingSharedLn	
MaxQueueReach	
AvgQueueReach	
MaxQueueAccumulation	
AvgQueueAccumulation	
QueueStorage	
FuelConsumption	
SLTAdjustment_T7F	
ExtEffGreenAdj_T7F	
PermittedMaxFlowRate	
ConflicingMovt1_T7F	
PctConflicing1_T7F	
ConflicingMovt2_T7F	
PctConflicing2_T7F	
LinkLength	
AvgTravelSpeed	
RunningTime	
UnitTT	Sec/veh, Min/veh, etc.
TotalTT	Veh-sec, Veh-min, etc.
TotalTD	Veh-mi, veh-km, etc.
UniormfDelay	
IncrementalDelay	
ControlDelay	
UnitDelay	Same units as UnitTT
TotalDelay	Same units as TotalTT
PassengerHrDelay	
UniformStops	
UniformStopsPct	
RandomStops	
RandomStopsPct	
Stops	
StopsPct	
MinPhaseTime	
LinkNumber	

GrowthFactor	Decimal multiplier for traffic volume
AvgVehSpace	
EffGreen	
EffRed	
OpposingVol	
FuelConsumption	
OperatingCost	
ApproachVolPct	Percent of approach volume in sub-approach class

The TMC Class (Child of Intersection)

Turning movement counts (TMCs) are ideally suited to TMML representation because the same basic data (i.e., a set of traffic volumes representing a specified movement during a specified period of time) are now stored in innumerable ways. At this point, manual reentry is the most common method of transferring data from one format to another. An established TMML representation could facilitate the transfer of TMC data from, for example, a TMC database to a traffic model software product such as the Arterial Analysis Package.

The TMC data are presented for each time period within the <TMC> class. The "Begin" attribute for this class indicates the time (military) at which the count begins. The element tags identify the specific movements.

TMC Begin = (Time)	
<NBLT	
<NBThru>	
<NBRT>	
<SBLT>	
<SBThru>	
<SBRT>	
<EBLT>	
<EBThru>	
<EBRT>	
<WBLT>	
<WBThru>	
<WBRT>	

The order in which these class tags are presented above is arbitrary, and no specific order is required. Other class and element tags could be added to fully represent the data acquisition capabilities of available TMC hardware (e.g., a fourth movement on each approach).

It is anticipated that TMML files with TMC data will usually be separate files produced by the TMC data reduction software, and combined with other tags included in this specification. These files will be imported by traffic model processing software and the information will be merged with the other traffic data.

The ODMATRIX Class (Child of APPROACH)

Some traffic models (e.g., TRANSYT and CORSIM) recognize the origin-destination characteristics on an approach. This class has been provided to specify those characteristics.

ODMATRIX (No attribute)	
LT2LT	O-Dmatrix: follows TSIS definition based on exiting volume.
Thru2LT	
RT2LT	
LT2Thru	
Thru2Thru	
RT2Thru	
LT2RT	
Thru2RT	
RT2RT	

The MOEGROUP Class (Child of ARTERIAL)

Some programs report MOEs by grouping that does not conform to the standard TMML class structure (e.g., average travel speed for through traffic in one direction of an arterial route). This class has been provided to accommodate such groupings. The group numbers or codes are subject to individual software product definitions.

MOEGROUP ID = (# or Code)	
GroupID	Program-specific identification of the group
Any MOE data elements that are normally applied within recognized classes	

The MODELPARAMETERS Class (Top Level, applied globally)

This class contains all of the model specific parameters, calibration factors and run control instructions required to execute specific software products.

MODELPARAMETERS (No attribute)	
TimingUnits	
FromIntersection	
ToIntersection	
MinCycle	
MaxCycle	
CycleIncrement	
CycleStepSearch_T7F	
FinalStepSearch_T7F	
TargetVCRatio	
StopPenalty	
PlatoonDispersionFactor_T7F	
DoubleCycleThreshold_T7F	
FuelMultiplier_T7F	
BOQFactor_T7F	
InflationFactor	
FuelCost	
LinkMOE_T7F	
AvgPPV	Vehicle occupancy (persons per vehicle)
PIDefinition_T7F	
DIDefinition_T7F	
ProsWeight_T7F	
ProsFactorFwd_T7F	
ProsFactorRev_T7F	
AllowLeadLagYN	
Ln1Reference	L+ or R+, indicates whether lanes are numbered from left or right
OptimizeSequenceYN	Run instructions to optimize sequence, splits or offsets
OptimizeSplitsYN	
OptimizeOffsetsYN	
FreeSpeedFactor_AAP	Factor to convert running speed to free speed
AnalysisType	Identifies model-specific options
PeriodHr	Period length: (Hours)
PeriodMinutes	Period length: (Minutes)
AreaType_HCM	Area type as defined by the HCM (CBD or Other)
MasterIntersection	The master intersection offset is zero by definition

APPENDIX B

TRAFFIC MODEL MARKUP LANGUAGE (TMML)

ADDENDUM FOR LOSPLAN COMPONENT PROGRAMS

ARTPLAN
FREEPLAN
HIGHPLAN

Prepared for
The Florida Department of Transportation
Systems Planning Office

By the University of Florida
Transportation Research Center

January 2002

LOSPLAN DATA STRUCTURE AND INTERFACE REQUIREMENTS

The LOSPLAN package is a set of software tools developed and used by the Florida Department of Transportation to conduct planning level analyses to assess the performance of signalized intersections, arterial routes, highways and freeways throughout Florida. A new version of LOSPLAN has been developed to add multimodal functionality and to enhance the user interface. The total package contains analysis procedures for:

Signalized arterials
Signalized intersections
Freeways
Two-lane and multi-lane highways

Previous versions of FDOT's planning level software offered two categories of programs named according to their function. The "PLAN" programs (e.g, ARTPLAN) produced an estimate of the level of service based on specific inputs. The "TAB" programs (e.g., ARTTAB) produced tables of service volumes for a variety of conditions. The purpose of the LOSPLAN software is to combine the PLAN and TAB functions into a single product.

This document describes the detailed structure for loading, saving and exchanging the required input and output data.

Proposed Data Structure

To facilitate loading and saving of data, as well as data exchange between programs, a common data storage scheme has been employed. The mechanism proposed in this document is based on the Traffic Model Markup Language (TMML), a fully XML-compliant method consisting of a structure and vocabulary designed specifically for traffic data. A complete specification for TMML has been developed to encompass several of the most commonly used traffic analysis models [1]. This document describes the manner in which the LOSPLAN data structure fits into the general concept of TMML. It also creates the additional class and data element tags required to accommodate all of the LOSPLAN data.

A specific data structure based on TMML is proposed for each facility type. The structure and vocabulary of the existing TMML specification has been extended to accommodate the new LOSPLAN functionality with minimal modifications. The following discussion assumes at least a rudimentary knowledge of XML, plus a general familiarity with the current TMML specification as outlined in Reference 1.

The LOSTABLES Class

A new data class must be added to accommodate the LOS tables that were previously produced by the TAB programs. The LOS Tables are facility specific and present the following information for LOS A through E

Peak-hour service volumes for the peak direction
Peak-hour service volumes for both directions
Maximum AADT for both directions
Maximum peak-hour v/c ratios for the peak direction

The LOSTABLES class serves as a container for this information. There are no data elements at the class level. Instead, two sub classes (child classes) have been created to contain the information for the peak direction and both directions, respectively. The information for the four data items identified above is presented within those two sub classes.

The ARTERIAL Facility Structure

It was necessary to add one new XML class for arterials to deal with the division of arterial segments into subsegments for bicycle, pedestrian and transit purposes. Another class was added to represent all of the information that appears in the LOS tables previously generated by the "Tab" programs. About a dozen new data element tags were also created to accommodate the additional input and output requirements for these three modes of travel.

The data class structure, which includes the new "SUBSEGMENT" class is presented as follows:

```
<TMML Facility = "Arterial">
<GENERAL>
</GENERAL>
<AGENCY>
</AGENCY>
<ARTERIAL ID=#>
        <INTERSECTION ID=# >
                <CONTROLLER>
                </CONTROLLER>
                <APPROACH ID= (Code)>
                        <LANEGROUP ID="T")>
                        </LANEGROUP>
                        <SUBSEGMENT ID=# >              (New data Class)
                        </SUBSEGMENT>
                </APPROACH>
        </INTERSECTION>
        <MOEGROUP ID=(Code)>
        </MOEGROUP>                                     (New definition)
        <MODELPARAMETERS>
        </MODELPARAMETERS>
        <LOSTABLES Lanes =#>                            (New data class)
                <PEAKDIRECTION LOS=(A-E)>
                </PEAKDIRECTION>
                <BOTHDIRECTIONS LOS=(A-E)>
                </BOTHDIRECTIONS>
        </LOSTABLES>
</ARTERIAL>
</TMML>
```

The TMML specification provides for the MOEGROUP class, which is intended for grouping measures of effectiveness (MOEs) according to program-specific rules. This class will be used in ARTPLAN to aggregate the bicycle, pedestrian and transit MOEs over multiple contiguous

segments. This capability is important for bicycle and pedestrians because their trips often do not cover the entire arterial route. It is important to transit because the character of transit operations may change within the arterial route.

The HIGHWAY Facility Structure

The highway facility encompasses both two lane and multilane highways. The structure is much simpler than the arterial facility because there are no segments, signal control parameters or multimodal features to accommodate. The LOSTABLES class was added to all facilities to represent of the information that appears in the LOS tables previously generated by the "Tab" programs. A few new data element tags were also created to accommodate the additional input and output requirements for this facility type.

The proposed HIGHWAY facility structure is presented as follows:

<TMML Facility = attribute>, where attribute = "TwoLane" or "MultiLane"
<GENERAL>
</GENERAL>
<AGENCY>
</AGENCY>
<HIGHWAY ID="1">
(Input data elements)
</HIGHWAY>
<LOSTABLES Lanes =#>
<PEAKDIRECTION LOS=(A-E)>
</PEAKDIRECTION>
<BOTHDIRECTIONS LOS=(A-E)>
</BOTHDIRECTIONS>
</LOSTABLES>
</TMML>

Since there are no links to be grouped for analysis purposes, the MOEGROUP class is not used in representing the highway facility.

The FREEWAY Facility Structure

The freeway facility structure is more complicated than the highway structure because it must accommodate segments. But it is still simpler than the arterial structure because of the lack of signal control and multimodal features. The LOSTABLES class was also added to this facility to represent all of the information that appears in the LOS tables previously generated by the "Tab" programs. A few new data element tags were also created to accommodate the additional input and output requirements for this facility type.

The proposed FREEWAY facility structure is presented as follows:

```
<TMML Facility = “Freeway”>
<GENERAL>
</GENERAL>
<AGENCY>
</AGENCY>
<FREEWAY ID=”1”>
        (Input data elements)
        <SEGMENT ID = “#”>  (20 segments max. Segment 0 applies to the whole facility)
                <ONRAMP ID=”#”>  (Max 2 on-ramps per segment)
                </ONRAMP>
                <OFFRAMP ID=”#”>  (Max 2 off-ramps per segment)
                </OFFRAMP>
        </SEGMENT>
</FREEWAY>
<LOSTABLES Lanes =#>
        <PEAKDIRECTION LOS=(A-E)>
        </PEAKDIRECTION>
        <BOTHDIRECTIONS LOS=(A-E)>
        </BOTHDIRECTIONS>
</LOSTABLES>
</TMML>
```

Description of TMML Classes and Data Elements

The proposed scheme will now be described in detail using the existing TMML specification as a base. The classes and data elements that are redundant to LOSPLAN have been removed and the additional classes and elements required to accommodate the multimodal expansion have been incorporated. This scheme covers only the arterial and Highway facilities for now, and must be extended to the other components of the LOSPLAN package.

The GENERAL Class

This class contains elements that describe the file itself, as opposed to the data within the file. It is good practice to place this class at the top of the file, because it identifies the source of the data, units of measurement etc. The LOSPLAN programs should not require the GENERAL class to be at the beginning, but should make a preliminary pass to establish the parameters of the data. No data elements have been added to the GENERAL class to accommodate LOSPLAN.

GENERAL (No attribute)	
FileName	File Name
Program	[Always **LOSPLAN**]
Version	Program Version
Units	Units (U+ or M+) [Always U until a metric version is created]
Date	Date That the File Was Generated
District	Label
Comment	Label
Analyst	Label
PeriodID	Label

The AGENCY Class

This class contains a collection of elements that describe the user of the program. LOSPLAN uses only one element to describe the agency.

AGENCY (No Attribute)	
AgencyName	All elements in this class are labels, used in different combinations by various programs

The ARTERIAL Class

All of the data items for an arterial facility will be contained in this class. ARTPLAN accommodates only one arterial, therefore the attribute value for the <ARTERIAL> tag must be "1."

ARTERIAL ID=(Arterial #)	
ArterialName	
FwdDirection	Direction of increasing intersection sequence number
RevDirection	Direction of decreasing intersection sequence number
ArterialClass_HCM	As defined by HCM
SignalsPerMile	
KFactor	Ratio of peak-hour volume to daily volume
DFactor	Proportion of two-way traffic in the peak (heavier)direction
DDHV	Directional design hour volume
NumberOfIntersections	
TotalLength	
AADT	Average annual daily traffic

The INTERSECTION Class (Child of ARTERIAL)

Because of the planning level character of ARTPLAN, the intersection class contains very little data. Note that the X and Y coordinate data (part of the TMML specification) have remained in this class to accommodate possible future GIS applications. The coordinates are not currently recognized by ARTPLAN.

INTERSECTION ID=(Intersection #)	
CrossStreetName	
ControlMode	P+, A+, S+, 2+, 4+
XCoordinate	Referenced to an origin at the western limit
YCoordinate	Referenced to an origin at the southern limit

The CONTROLLER Class (Child of INTERSECTION)

This class contains all of the data elements that apply to the controller as a whole. There is no attribute for this class because there is only one controller per intersection. Since ARTPLAN does not deal with the details of traffic control all of the elements except one have been removed from this class.

CONTROLLER (No Attribute)	
CycleLength	

The APPROACH Class (Child of INTERSECTION)

This class contains the approach-level data for each of four approaches to an intersection. ARTPLAN uses only the "Forward" and "Reverse" approaches.

APPROACH ID = (# or Code)	
LTBayLength	
FreeFlowSpeed	
PctTurnExclLn	Percentage of the total volume that turns from exclusive lanes.
ParkingRightYN	HCM provides for left and right parking, but ARTPLAN uses only the left
Additional ARTPLAN Tags	
PedSection	For grouping pedestrian segments (5 sections max)
BikeSection	For grouping bicycle segments (5 sections max)
BusRouteSegment	For grouping bus segments (5 route segments max)
PedLOS	LOS for pedestrians
BikeLOS	LOS for bicycles
BusLOS	LOS for transit

The arterial directions must be specified by a <FwdDirection> or <RevDirection> tag under the <ARTERIAL> class.

Sub-Approach Classes

Different modeling process and software products employ different structures to partition the traffic flow on an approach to an intersection. Three different sub-approach classes have been created to accommodate all of the traffic models:

The LANEGROUP Class (Child of APPROACH)

A lane group is a set of one or more lanes carrying a homogeneous mixture of movements. Three lane groups may be included on each approach in current software. The code designating the lane group may be:

- A number in the range 1 to 3
- The corresponding code L+, C +or R+, designating the left, center or right lane group, respectively.

Lane groups will normally be used for models that represent homogeneous flows on individual links (e.g., TRANSYT, HCM, etc.).

The MOVEMENT Class (Child of APPROACH)
Three movements may be included on each approach in the <MOVEMENT> class. The code designating a movement may be either a number in the range 1 to 3, or the corresponding code letter L+, T+, or R+. ARTPLAN does not use the MOVEMENT class.

The LANE Class (Child of APPROACH)

Some traffic models deal approach flow on a lane-by-lane basis. No current TMML processing software recognizes individual lane movements, so this class is reserved for future expansion and development.

Since most of the sub-approach data elements are common to all of the sub-approach classes, they are presented in a single table.

Sub-Approach Classes	
MOVEMENT ID = (# or Code)	
LANEGROUP ID = (# or Code)	
LANE ID = (# or Code)	
Movts	Movements permitted in the class (Combination of LTR)
Vol	
PHF	
SatFlowPerLn	
NumberOfLns	
HVPct	
ArrivalType	
UnitExtension_HCM	
InitialQueue	

NumberOfBuses	
VCRatio	
GCRatio	
ControlDelay	
LOS	LOS based on control delay
SegmentLOS	LOS based on average speed in the segment
LinkLength	
AvgTravelSpeed	
RunningTime	
UnitTT	Sec/veh, Min/veh, etc.
TotalTT	Veh-sec, Veh-min, etc.
TotalTD	Veh-mi, veh-km, etc.
UnitDelay	Same units as UnitTT (Generally Control delay + other delay)
TotalDelay	Same units as TotalTT
GrowthFactor	Decimal multiplier for traffic volume

The SUBSEGMENT Class (Child of APPROACH)

This class has been added to accommodate ARTPLAN's need to break a roadway segment into discrete subsegments for bicycle and pedestrian analysis. Most of the new data elements for bicycles, pedestrians and transit are applied at the subsegment level.

SUBSEGMENT ID=#	
HVPct	Percent heavy trucks
NumberOfBuses	Buses per hour
ParkingRightYN	Is on-street parking permitted?
Additional ARTPLAN Tags	
PctOf Segment	Percent of the approach segment included in the subsegment
MedianType	Restrictive, Non-restrictive, None
BikeLnYN	
OutsideLnWidth	Narrow, Standard, Wide
PavementCondition	New, Typical, Terrible
SidewalkYN	
SidewalkSeparation	Adjacent, Typical, Wide
SidewalkToBusYN	Is there a clear, short path to the bus stop?

The MOEGROUP Class (Child of ARTERIAL)

Some programs report MOEs by grouping that does not conform to the standard TMML class structure (e.g., average travel speed for through traffic in one direction of an arterial route). This class has been provided to accommodate such groupings. The group numbers or codes are subject to individual software product definitions.

ARTPLAN uses the MOEGROUP class to aggregate MOE's for bicycles, pedestrians and buses over specified segment groups. MOE groups for bicycles and pedestrians are called "sections." MOE groups for buses are called "route segments" for consistency with transit terminology. Each segment of the arterial must be assigned to a specific MOE group. ARTPLAN accommodates up to five MOE groups for each mode. The MOE group codes consist of a character that identifies the mode (B, P, T for bicycles, pedestrians and transit, respectively) followed by a number between 1 and 5. So, for example, the MOE group code "B2" would indicate the second MOE group for bicycles.

MOEGROUP ID = (# or Code)	
GroupID	Program-specific identification of the group
Any MOE data elements that are normally applied within recognized classes	

The MODELPARAMETERS Class (Top Level, applied globally)

This class contains all of the model specific parameters, calibration factors and run control instructions required to execute specific software products.

MODELPARAMETERS (No attribute)	
PeriodHr	Period length: (Hours)
PeriodMinutes	Period length: (Minutes)

The HIGHWAY Class (Top Level in HIGHPLAN)

This class applies to highway facilities only. All of the data items for the highway (with the exeption of the LOS tables) will be contained in this class. LOSPLAN accommodates only one highway, therefore the attribute value for the <HIGHWAY> tag must be "1."

HIGHWAY ID=# (Only 1 highway is accommodated by LOSPLAN)	
Name	
Tags common to the ARTERIAL class	
FwdDirection	Direction of increasing intersection sequence number
KFactor	Ratio of peak-hour volume to daily volume

Dfactor	Proportion of two-way traffic in the peak (heavier) direction
FreeFlowSpeed	
PHF	
NumberOfLns	Always in each direction in the data. May require conversion
RunningSpeed	
AreaType	
LOS	
AADT	Average annual daily traffic
GrowthFactor	Decimal multiplier for traffic volume
SatFlowPerLn	Adjusted saturation flow rate per lane (ASFR on the FoxPro screen)
New Tags for LOSPLAN	
MedianYN	Existence
LTBaysYN	Existence
PropPassingZone	Proportion of Passing zone on highway
PropPassingLn	Proportion of passing lane on highway
PostedSpeed	Posted speed (mph)

The FREEWAY Class (Top Level in FREEPLAN)

This class applies to freeway facilities only. All of the data items for the freeway (with the exeption of the LOS tables) will be contained in this class and its child classes. The current version of LOSPLAN accommodates only one freeway, therefore the attribute value for the <FREEWAY> tag must be "1."

FREEWAY ID="#" (Only one freeway accommodated in FREEPLAN)

FacilityName	Name of Freeway
To	Starting Point
From	Ending Point
FwdDirection	(North, South, East or West)
PeakDirectionYN_FP	Is this the peak direction? (Boolean)
FreewayClass_FP	Class of freeway by HCM definition
NumberofSegments	Max 20 segments
PostedSpeed	
AADT	
KFactor_PLN	
Dfactor_PLN	
DDHV	
PHF	
HVPct	
NumberOfLns	

The SEGMENT Class (Child of FREEWAY)

This class contains all of the information related to a specific segment of the freeway. FREEPLAN accommodates a maximum of 20 segments. Segment zero represents the default values for all segments.

SEGMENT ID="#"	
To	Starting point of segment
From	Ending point of segment
SegmentType	Basic segment, on-ramp, off-ramp, interchange, etc.
TypeCode_FP	??
Length	Segment length
NumberOfLns	
FFSpeed	
Terrain	Level or rolling
AuxLns	Number of auxiliary lanes
BetweenLength_FP	??
BetweenAUXLaneYN_FP	??
Distance0_FP	??
Distance1_FP	??
Distance2_FP	??

The ONRAMP Class (Child of FREEWAY or SEGMENT)

This class contains the properties of a specific on-ramp. FREEPLAN accommodates two on-ramps per segment. If the class tag appears as a child of the FREEWAY class, the data are considered as global defaults and the attribute is ignored. If it appears as a child of the SEGMENT class, the data will apply only to that class. If no attribute is given to indicate which of the two ramps is represented, or if the segment type only contains one ramp, then the data will apply to the first (or only) ramp.

<ONRAMP ID="1 or 2">

Volume	Ramp Volume
HVPct	Percent Heavy Vehicles
NumberOfLns	Number of lanes on ramp

The OFFRAMP Class (Child of FREEWAY or SEGMENT)

This class contains the properties of a specific off-ramp. FREEPLAN accommodates two off-ramps per segment. If the class tag appears as a child of the FREEWAY class, the data are considered as global defaults and the attribute is ignored. If it appears as a child of the SEGMENT class, the data will apply only to that class. If no attribute is given to indicate which of the two ramps is represented, or if the segment type only contains one ramp, then the data will apply to the first (or only) ramp.

<OFFRAMP ID="1 or 2">
Volume Ramp Volume
HVPct Percent Heavy Vehicles
NumberOfLns Number of lanes on ramp

The LOSTABLES Class (Top Level, Facility MOE)

The LOSTABLES class is used as a container for all of the tabular output of all of the LOSPLAN programs. The attribute value indicates the number of lanes in each direction (i.e, the row in the LOS table) and establishes the internal storage location for the data. There are no data elements at the LOSTABLES level. All of the tabular data will placed within a child class.

LOSTABLES Mode=(A,B,P,T) Only the first letter is significant (e.g., Autos Automobiles, Bikes, Bicycles, Peds, Pedestrian, etc.)	
(No data elements)	

The CROSSSECTION Class (Child of LOSTABLES)

The CROSSSECTION class is used as a container for all data that applies to a particular roadway cross section. The attribute value indicates the number of lanes in each direction (i.e, the row in the LOS table) and establishes the internal storage location for the data. There are no data elements at the CROSSSECTION level. All of the tabular data will placed within a child class.

This class applies only to the "Automobile" mode. If the class tag is omitted, one lane will be assumed in each direction. Thus, the CROSSSECTION class is used only for automobile mode data on multilane highways.

CROSSSECTION Lanes=(1,2,3,4) to indicate the applicable number of lanes in each direction.	
(No data elements)	

The SERVICEVOL Class (Child of LOSTABLES of CROSSSECTION)

This class contains the hourly service volumes for the peak direction, both directions and AADT. The attribute establishes the LOS (i.e., the column in the LOS table)

SERVICEVOL LOS=(A-E)	
PeakDirection	Peak service volume for the specified LOS
BothDirections	Bi-directional service volumes for the specified LOS
AADT	AADT service volumes for the specified LOS

References

1. Traffic Model Markup Language Preliminary Specification. University of Florida Transportation Research Center, July 2000.

Example TMML Representation Of The LOS Tables For A Full Multimodal Data Set Describing A Multilane Arterial:

Note: "###" represents the numerical data

```
<LOSTABLES Mode="Auto">
        <CROSSSECTION Lanes =  "2">
                <SERVICEVOL LOS="A"
                        <PeakDirection>###<PeakDirection>
                        <BothDirections>###<BothDirections>
                        <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="B"
                        <PeakDirection>###<PeakDirection>
                        <BothDirections>###<BothDirections>
                        <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="C"
                        <PeakDirection>###<PeakDirection>
                        <BothDirections>###<BothDirections>
                        <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="D"
                        <PeakDirection>###<PeakDirection>
                        <BothDirections>###<BothDirections>
                        <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="E"
                        <PeakDirection>###<PeakDirection>
                        <BothDirections>###<BothDirections>
                        <AADT>###<AADT
                </SERVICEVOL>
        </CROSSSECTION>
        <CROSSSECTION Lanes =  "3">
                <SERVICEVOL LOS="A"
                        <PeakDirection>###<PeakDirection>
                        <BothDirections>###<BothDirections>
                        <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="B"
                        <PeakDirection>###<PeakDirection>
                        <BothDirections>###<BothDirections>
                        <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="C"
```

```
                    <PeakDirection>###<PeakDirection>
                    <BothDirections>###<BothDirections>
                    <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="D"
                    <PeakDirection>###<PeakDirection>
                    <BothDirections>###<BothDirections>
                    <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="E"
                    <PeakDirection>###<PeakDirection>
                    <BothDirections>###<BothDirections>
                    <AADT>###<AADT
                </SERVICEVOL>
            </CROSSSECTION>
            <CROSSSECTION Lanes =  "4">
                <SERVICEVOL LOS="A"
                    <PeakDirection>###<PeakDirection>
                    <BothDirections>###<BothDirections>
                    <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="B"
                    <PeakDirection>###<PeakDirection>
                    <BothDirections>###<BothDirections>
                    <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="C"
                    <PeakDirection>###<PeakDirection>
                    <BothDirections>###<BothDirections>
                    <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="D"
                    <PeakDirection>###<PeakDirection>
                    <BothDirections>###<BothDirections>
                    <AADT>###<AADT
                </SERVICEVOL>
                <SERVICEVOL LOS="E"
                    <PeakDirection>###<PeakDirection>
                    <BothDirections>###<BothDirections>
                    <AADT>###<AADT
                </SERVICEVOL>
            </CROSSSECTION>
    </LOSTABLES>
    <LOSTABLES Mode="Bicycle">
            <SERVICEVOL LOS="A"
                <PeakDirection>###<PeakDirection>
```

```
            <BothDirections>###<BothDirections>
            <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="B"
            <PeakDirection>###<PeakDirection>
            <BothDirections>###<BothDirections>
            <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="C"
            <PeakDirection>###<PeakDirection>
            <BothDirections>###<BothDirections>
            <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="D"
            <PeakDirection>###<PeakDirection>
            <BothDirections>###<BothDirections>
            <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="E"
            <PeakDirection>###<PeakDirection>
            <BothDirections>###<BothDirections>
            <AADT>###<AADT
        </SERVICEVOL>
</LOSTABLES>
<LOSTABLES Mode="Pedestrian">
        <SERVICEVOL LOS="A"
            <PeakDirection>###<PeakDirection>
            <BothDirections>###<BothDirections>
            <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="B"
            <PeakDirection>###<PeakDirection>
            <BothDirections>###<BothDirections>
            <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="C"
            <PeakDirection>###<PeakDirection>
            <BothDirections>###<BothDirections>
            <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="D"
            <PeakDirection>###<PeakDirection>
            <BothDirections>###<BothDirections>
            <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="E"
```

```
                <PeakDirection>###<PeakDirection>
                <BothDirections>###<BothDirections>
                <AADT>###<AADT
        </SERVICEVOL>
</LOSTABLES>
<LOSTABLES Mode="Transit">
        <SERVICEVOL LOS="A"
                <PeakDirection>###<PeakDirection>
                <BothDirections>###<BothDirections>
                <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="B"
                <PeakDirection>###<PeakDirection>
                <BothDirections>###<BothDirections>
                <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="C"
                <PeakDirection>###<PeakDirection>
                <BothDirections>###<BothDirections>
                <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="D"
                <PeakDirection>###<PeakDirection>
                <BothDirections>###<BothDirections>
                <AADT>###<AADT
        </SERVICEVOL>
        <SERVICEVOL LOS="E"
                <PeakDirection>###<PeakDirection>
                <BothDirections>###<BothDirections>
                <AADT>###<AADT
        </SERVICEVOL>
</LOSTABLES>
```

APPENDIX C

Development of an XML-Based Specification for Traffic Model Data Interchange

Kenneth G. Courage
kcour@ce.ufl.edu

Scott S. Washburn
swash@ce.ufl.edu

Jin-Tae Kim
kimjint@ufl.edu

Department of Civil and Coastal Engineering
University of Florida
511 Weil Hall, PO Box 116588
Gainesville, FL 32611-6588
352-392-7575 (phone)
352-392-3394 (fax)

Submitted for Presentation at the 81st Annual Meeting of the Transportation Research Board, Washington, D.C., 2002

ABSTRACT

The proliferation of traffic software programs on the market today has resulted in many very specialized programs, aimed at analyzing one or two specific items within a transportation network. Consequently, traffic engineers usually find themselves utilizing multiple programs on a single project, which has ironically resulted in a new inefficiency for the traffic engineer. Most of these programs deal with the same core set of data, for example, physical roadway characteristics, traffic demand levels, and traffic control variables. However, most of these programs have their own special format for saving data files. Therefore, these programs are not able share information directly or communicate with each other because of incompatible data formats. Thus, the traffic engineer is often faced with manually re-entering common data from one program into another. In addition to the inefficiency issue, this also creates additional opportunities for data entry related errors to be made.

As chronicled in a recent *Scientific American* article, the Extensible Markup Language (XML) is catching on rapidly as a means of transferring data between two systems or users who deal with the same data, but in different formats. There have been specific vocabularies developed for statistics, mathematics, chemistry, and many other disciplines. This paper attempts to add the traffic-modeling field to the ever-expanding list of XML community members. This paper introduces the *Traffic Model Markup Language* (TMML) as a resource for traffic model data representation, storage, rendering and exchange. The TMML structure and vocabulary are described and examples of their use are presented.

Introduction

The prevalent use of traffic modeling and analysis software programs has revolutionized how traffic engineers do their job and the efficiency with which they do it. With advances in computer hardware and software technology, computer programs continue to become more sophisticated and versatile. The proliferation of traffic software programs on the market today has also resulted in many very specialized programs, aimed at analyzing one or two specific items within a transportation network. Consequently, traffic engineers usually find themselves utilizing multiple programs on a single project.

This now common practice of using multiple software programs in the analysis of a single facility has ironically resulted in a new inefficiency for the traffic engineer. Most of these programs deal with the same core set of data, for example, physical roadway characteristics, traffic demand levels, and traffic control variables. However, most of these programs have their own special format for saving data files. Therefore, these programs are not able share information directly or communicate with each other because of incompatible data formats. Thus, the traffic engineer is often faced with manually re-entering common data from one program into another. In addition to the inefficiency issue, this also creates additional opportunities for data entry related errors to be made. Some software developers have tried to address this issue by writing conversion routines, either for exporting their data format to other programs, or for reading other program formats into their program. This is very inefficient from a developer's standpoint, as a conversion routine needs to be developed for each different program format (possibly both an import and an export routine).

To facilitate the development of more efficient traffic software, and to reduce the potential for data coding errors, what is needed is a common data file format with which all programs are

compatible. This paper describes the specification of such a format for use in the traffic-modeling arena. It also provides some specific examples of the use of this format. The objective is to describe a successful application of an industry standard to traffic model data exchange as a basis for further discussion of its general applicability to traffic-related software. This is not an attempt to cover the entire field of traffic-related software, nor is it an attempt to propose a new standard.

Overview of THE Extensible Markup Language (XML)

The problem of communication incompatibility is not unique to traffic modeling software programs. In fact, compatibility struggles within the financial transaction arena (e.g., E-Commerce) led the World Wide Web Consortium (W3C) to develop a standard for an electronic communication model that could be adopted by any organization [W3C, 2000]. The developed communication model/format was termed XML, for 'Extensible Markup Language'. Although initially targeted at electronic retailers, the language is readily extensible, as the name implies, to any specific discipline vocabulary. XML is simply a method for putting structured data into a text file. XML defines a set of rules and guidelines for producing text files that are easy to generate and read (by a computer), and are unambiguous.

As chronicled in a recent *Scientific American* article (Bosak et al, 1999), the Extensible Markup Language (XML) is catching on rapidly as a means of transferring data between two systems or users who deal with the same data, but in different formats. There have been specific vocabularies developed for statistics, mathematics, chemistry, and many other disciplines. All future indications are that XML is here to stay. There are programming journals heavily or exclusively dedicated to XML, XML is the data access standard in .NET (Microsoft's software development package successor to Visual Studio), and XML is supported in Office XP, to name just a few.

XML is actually a close relative of HTML (HyperText Markup Language). HTML is the standard for data display and formatting within web browsers (e.g., Internet Explorer, Netscape Navigator). The fundamental difference between the two is simple, but significant. Like HTML, the XML structure consists of tags (words bracketed by '<' and '>') and attributes (of the form name = "value"). But while HTML specifies what each tag and attribute means, and often how the text between them will look in a browser, XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it. To further clarify, an example HTML and XML statement will be shown and then the differences between the two will be explained.

HTML: UF-Transportation Research Center
XML: <AADT>25700</AADT>

The interpretation of the HTML statement by a web browser is that the text "UF-Transportation Research Center" should be displayed in bold print. While the XML statement is basically identical in format to the HTML statement, the difference is that the tag name, <AADT>, describes the type of data bounded by the tags. Therefore, the '27500' is interpreted as being the amount of Annual Average Daily Traffic. Whereas the HTML structure is such that the tag name indicates how to display the item between the tags, the XML format tag name describes the type of data between the tags.

Just like a web browser application must "know" the meaning of any tag names that an HTML document author wants to use, an application that uses an XML formatted document must know the definitions of the tags contained within it. The purpose of the scheme described in this paper is to create a set of *such* tags and define their meaning. Specifically, the Traffic Model Markup Language (TMML) has been developed to facilitate sharing of data between traffic modeling software products. TMML is a fully XML-compatible markup language prescribing the class structure and data element tag names required to represent traffic model data in a "self-describing" format. The principal applications of TMML include exchanging data between traffic model software products and facilitating the compilation and presentation of results.

Development of the TMML Specification

The underlying principle that guided the development of TMML is the desire for a scheme that can be used by non-computer professionals. The XML language is used in the "big leagues" of E-commerce and other areas where simplicity is not an issue. On the other hand, transportation system modeling is done by specialists in the transportation field (practitioners, researchers, students, etc.) and not by specialists in the computer field. The development of a scheme that is beyond the technical level of most of its users would defeat the purpose of TMML.

Nevertheless, office productivity software is moving towards XML compatibility, and new "XML-aware" applications and utilities are continuing to reach the market. So it is absolutely essential that TMML be fully XML compliant. These two requirements, standardization and simplicity, have been combined to develop TMML as a less formal but more structured subset of XML.

The TMML language is defined in terms of a collection of data structures that describe the properties of the objects associated with traffic carrying facilities. TMML provides the structure and vocabulary to completely define a data set for various facilities and software products. It was developed at the University of Florida Transportation Research Center and has been applied extensively to software products that analyze the performance of several facilities, including freeways, multilane and two-lane highways, and signalized arterials. Many of these software products implement various chapters of the Highway Capacity Manual (HCM) (TRB, 2000).

TMML is fully described in 'Traffic Model Markup Language Specification' (UF-TRC, 2000), which includes a detailed specification for the structure and vocabulary for all classes of data. The specification includes a list of recognized abbreviations intended to reduce the size of the XML tags that describe the data elements. TMML was created following the general principles of style described in IEEE Standard 1489 (IEEE, 2000).

An example of the TMML structure applied to a signalized arterial is presented in Figure C-1. Only the class tags are shown in this figure. Classes represent categories under which logically related data items are grouped. For example, the CONTROLLER class might contain the cycle length, the offset, the number of rings, and the like, with each of these represented by a separate data element tag. TMML files can be somewhat lengthy, and a complete description of all of the data elements is beyond the scope of this paper.

Some of the principles and guidelines that govern the TMML language include:

- The root tag is always <TMML> with an attribute defining the type of facility that the data represents (freeway, arterial, etc.).
- Each class tag has a maximum of one attribute to facilitate reading and parsing of data. That attribute distinguishes the specific instance of the class from all other instances of the same class (e.g., INTERSECTION ID = "3").
- The order of presentation of classes or elements within a class is not prescribed. It is possible to present the classes and elements within a class in any order as long as the class structure is maintained.
- The same class may appear more than once in a file (e.g., to separate input and output data). A given data item may appear only once in any class.
- Class elements are represented in upper-case characters (e.g., APPROACH). Data elements within each are represented as a concise series of connected words with lower-case characters and initial capitals (e.g., MedianWidth).
- Brevity is not an essential feature of XML, nor is it encouraged by the IEEE standard. Words in all tags are generally spelled out in full, except when they have a recognized abbreviation defined in the specification (e.g. "Movt" for "movement").
- No specific units or system of units are implied in any of the tags that constitute the TMML vocabulary. For example, total delay may be expressed in vehicle-seconds, vehicle-minutes or vehicle-hours, depending on the individual software product. It is essential that programmers understand the interpretation that each software product places on all data items.

The current TMML specification covers freeways, two-lane highways, multilane highways, arterials and intersections, each of which is represented by its own chapter in the HCM.

To avoid setting rigid and prescriptive requirements, alternatives that encompass a broad range of current practice are offered. No software product is likely to recognize all of the data elements contained in the TMML specification. Each product that offers TMML connectivity will require a programmer interface document (analogous to a document type definition (DTD) in XML jargon) that identifies the data elements that are recognized and any conditions that apply to their interpretation.

Description of TMML Classes

Each of the classes represented in Figure C-1 provides a "container" for the data elements that describe the properties of the class. A brief description of each of the classes that represent an arterial facility now follows.

- The *GENERAL Class* contains elements that describe the file itself, as opposed to the traffic data within the file.
- The *AGENCY Class* contains a collection of elements that describe the user of the program.
- The *ARTERIAL Class* contains all of the sub-classes and data items for an arterial facility.

For network analysis programs such as PASSER 4, TRANSYT-7F and CORSIM that can accommodate multiple arterials, the attribute value indicates which arterial is being described.

The *INTERSECTION Class* contains all of the sub-classes and data items required to represent each intersection. The attribute value must indicate the specific intersection to which the data elements apply.

The *CONTROLLER Class* contains all of the data elements that apply to the controller as a whole. This class is a child of the INTERSECTION class. There is no attribute because there is only one controller per intersection.

The *TIMINGPLAN Class* contains all of the information required to describe one phase of the timing plan at an intersection. The applicable signal phase must be specified as an attribute.

The *PHASECODES Class* provides options to accommodate the signal phasing schemes used by the common signal analysis software. The representation of signal phasing differs widely among traffic modeling software.

The *APPROACH Class* establishes the specific approach to the intersection. The approach designation must be included as an attribute.

The *LANEGROUP Class* establishes a subset of a specific approach. A lane group is a set of one or more lanes carrying a homogeneous mixture of movements. Lane groups will normally be used for models that represent homogeneous flows on individual links (e.g., TRANSYT, HCM, etc.).

The *MOVEMENT Class* identifies a specific movement (left, through, right) as a child of the APPROACH class. The specific movement must be designated as an attribute.

The *LANE Class* also establishes a subset of a specific approach. Some traffic models deal with approach flow on a lane-by-lane basis. The attribute must indicate the applicable lane number. A separate data element indicates whether the lane numbering is from left to right or right to left.

The *TMC Class* represents turning movement counts (TMCs) that supply traffic volumes as input data. TMCs are ideally suited to TMML representation because the same basic data (i.e., a set of traffic volumes representing a specified movement during a specified period of time) are now stored in innumerable ways. At this point, manual reentry is the most common method of transferring data from one format to another. An established TMML representation could facilitate the transfer of TMC data from, for example, a TMC database to a traffic model software product such as the Arterial Analysis Package. The TMC data are presented for each time period within the <TMC> class. The "Begin" attribute for this class indicates the time (military) at which the count begins. The element tags identify the specific movements.

The *ODMATRIX Class* stores the proportion of traffic entering a link as left, through and right turn movements that leave the link as left, through and right turn movements. Some traffic models (e.g., TRANSYT and CORSIM) recognize the origin-destination characteristics on an approach. This class has been provided to specify those characteristics.

The *MOEGROUP Class* provides for specialized grouping of measures of effectiveness (MOEs). Some programs report MOEs by grouping that does not conform to the standard TMML class structure (e.g., average travel speed for through traffic in one direction of an arterial route). This class has been provided to accommodate such groupings. The group numbers or codes are subject to individual software product definitions.

The *MODELPARAMETERS Class* contains all of the model specific parameters, calibration factors and run control instructions required to execute specific software products.

The *LOSTABLES Class* is used as a container for all of the tabular output of all of the LOSPLAN programs.

Reading and Writing TMML files

While of little or no concern to traffic software end-users, software program developers must be concerned with the details of actually creating XML files and subsequently reading them into the program. As already mentioned, and as seen Figure C-1, XML files are nothing more than a typical text file. Given this simplicity, simple text output routines were developed for several applications that are discussed later in this document. Tag names concatenated with the corresponding data item are written to the text file in the desired/appropriate sequence. Reading XML formatted routines is a little more complex than writing them. The file is read line-by-line, with all the contents of a line contained in a single text string. Since this string consists of a combination of the tag names and data items, the text string must be parsed to separate the tag from the data item.

A different approach to the 'brute force' method described above relies on XML processing functions developed by Microsoft Corporation. The routines reside in an external function library (MSXML.dll) that get called remotely from another application. The traffic software program just simply passes arguments (i.e., the tag name and data item) to the appropriate routine (either reading or writing) in the external library and it performs all the processing. While this method is easier, and reduces the amount of code in the traffic software program, it has a couple of drawbacks: 1) the programmer is constrained by the capabilities of the library routines; that is, the program cannot perform any reading or writing schemes that are not within the scope of the library routines; and 2) updates to the libraries based upon revised XML standards are dependent on whether Microsoft stays in business. However, a number of other vendors have developed, or are developing, XML reading and writing libraries, including many that are freely distributed.

Current Applications

TMML has already found its way into several traffic model applications. Its main functions have been storing, saving and exchanging data for specific modeling problems, presenting data in an embellished format and comparing the results from different data sets. The current usage of TMML is illustrated in Figure C-2.
The application programs that employ TMML in one form or another include:

The Highway Capacity Software (HCS 2000), which implements the facility-analysis

procedures prescribed in the HCM.

TRANSYT-7F, version 9 (McT7F9), the macroscopic simulation program for performing signalized arterial analysis and optimization.

The LOSPLAN suite of programs (ARTPLAN, FREEPLAN, HIGHPLAN), which is used by the Florida Department of Transportation for statewide planning purposes. The LOSPLAN programs provide a set of simplified HCM-based procedures for assessing the performance of all of the facility types indicated on Figure C-2. The simplification was achieved through the use of assumptions, approximations and default values. Visual LOS also generates tables of service volumes for LOS A through E, as defined in the HCM.

The Arterial Analysis Package (AAP2K), which provides access to several of the most commonly used traffic control system design and evaluation programs, including the -PASSER series, TRANSYT-7F, and CORSIM. The AAP reads and writes TMML files for an arterial facility. It also imports traffic counts, for which a TMML class has already been defined. Routines have been developed to extract the signal timing plans from PASSER and TRANSYT for importing into the AAP.

The Signal Operations Analysis Package (SOAP), which performs signal timing design and evaluation for a signalized intersection. The current version, SOAP2K, uses the computational engine of the HCS to ensure fidelity to the HCM performance analysis procedures. It provides the HCS with a design and optimization capability for developing optimal pretimed designs and for determining optimal settings for actuated control parameters. SOAP exchanges data with the HCS using the TMML file for the "intersection" facility type.

ARTPREP, which is a component program for AAP2K. ARTPREP is simply a front-end data input user interface for arterial analysis. The computational engine is the HCS, but is transparent to the user. The user only interfaces with ARTPREP, but when the user activates the LOS calculations for their inputs, ARTPREP exports the data in TMML format to the HCS, which performs the computations, and then returns the TMML file with the results embedded, for display in ARTPREP.

The Traffic Model Results Comparison (TMRC) program, which reads any two TMML files from the same facility type, and provides an intelligent comparison of their contents. TMRC is useful for comparing the results from two different software products (or different versions of the same product) applied to the same input data. It is also useful for evaluating the sensitivity of the results to variations in the input data. Each TMRC run appends a line to a text file that may be imported into a spreadsheet for analysis and plotting.

At this point TMML is more of a concept than a system. Its physical properties are limited to a set of loosely connected modules that process read, write and process the data. Most of these routines are embedded in traffic modeling software products.

Additional Benefits of XML-based Data Interchange Format

Some of the more potentially significant, yet subtle, features of XML that will prove to be very beneficial in the long run include the following:

- W3C controls the XML technology standard: As its name implies, the W3C is a consortium of over 500 commercial, governmental and educational members. This ensures that the standards are developed in the best interest of society, and are not driven solely by the special interests of one company or organization.
- XML is platform independent: Since the XML specification is pure text, it is a platform-neutral data format. The same XML file can be used to communicate with otherwise disparate systems, whether they be based on Microsoft Windows, Mac OS, Unix, etc.
- XML is supported by many vendors: With its fast growing support, users are not tied to specific vendors like Microsoft, or Sun, or IBM, etc.
- XML is license-free: Anyone can use XML, and no royalty fees are required for its use.
- XML is easy to read: It is both human- and machine-readable. Software development history has shown that the more comprehensible a standard, the more successful it will become. The easier it is to develop standards-compliant software, the more plentiful and less expensive the tools will become.
- XML is convenient and simple: There is nothing complicated about XML, yet its application possibilities are virtually limitless. Sometimes simple is better.

Some may argue that storing data in a text file structure is not the most efficient approach. With the computational power and storage capabilities of the modern computer, this argument holds little validity anymore. Furthermore, from a development standpoint, using a text format alleviates the difficulty of encoding and/or decoding complex proprietary formats of other programs that a developer may choose to make their program "compatible" with. It is rare these days to find a program that performs a perfect conversion of its data format to that of another program. Anyone that has transferred a document from Corel WordPerfect to Microsoft Word, or vice versa, can probably especially relate.

Interfacing with External Computational Engines

Implementation of TMML offers the opportunity to directly link custom data input user interfaces with standardized computational procedures. The Florida Department of Transportation specifies the use of its ARTPLAN software program for the determination of signalized arterial level of service. In its previous version, it was implemented as a LOTUS 1-2-3 spreadsheet. As LOTUS 1-2-3 began to lose some of its market share, some traffic engineering consultants developed their own spreadsheet implementation of ARTPLAN, and even stand-alone executable programs in some cases. This created a problem for the DOT in that they did not want to force firms to use LOTUS that had another spreadsheet program preference, yet they were concerned that custom developed programs may not be exact computational clones of their LOTUS spreadsheet. The new version of ARTPLAN, and a companion utility program, TMRC (described earlier), solve this problem in two ways. First, ARTPLAN, now a stand-alone

executable program, implements TMML as its data interchange format. This allows the FDOT to standardize it as the computational engine for arterial level of service estimation. This allows consultants the ability to still develop their own user interface, but pass the input data in TMML format to ARTPLAN for data validation and computations, and then return the results in TMML format to the user interface. This way, the FDOT can be assured that the custom program conforms to the computational requirements. Second, if a consultant wishes to develop a completely self-contained program (i.e., user-interface and computation engine), they can demonstrate computational compliancy through the use of the TMRC. By running FDOT standardized test data sets through their custom program and comparing the results to those obtained with FDOT's ARTPLAN, compliancy can be verified.

This capability also presents an opportunity for more direct linking of existing programs. For example, whereas one program can convert its custom input data format to that of another, to be run separately in the other program by the user, the first program can communicate directly with the computational engine of the second program through TMML, and provide its own summary of the results. If the first program writes its input data in the pre-specified TMML format of the other program, the second program can perform the data validation and generate the results, and the user of the first program never has to deal directly with the second program. Given that a number of traffic software program users are always curious as to how the touted "HCM-compatible" results of one program actually compare to those of say HCS, this method makes that comparison very straightforward.

Rendering TMML for Display and Printing

While XML is human readable, its native display format is not the most convenient for repetitive review. A complimentary development to XML has been that of the Extensible Stylesheet Language (XSL). XSL is an XML-based language for describing how to display XML formatted data in an HTML format for display in a web browser. This technique offers a very productive method of producing much more elegant output reports from any XML file. One advantage of XSL is that it easily lends itself to user-customization, and the creation of special reports that meet the needs of a particular user. Furthermore, with Web browser support of XSL, the programming overhead required for software developers to implement custom report formats for viewing and printing has been greatly reduced. For example, FDOT's new version of ARTPLAN uses an XSL template to display a custom report format directly in the program using an embedded Web browser control. This Web browser control handles the displaying and printing of the report, exactly like a stand-alone Web browser would, thus alleviating the developer of this normally arduous programming task. An example of an ARTPLAN file displayed in its native TMML format and the same file displayed in a custom report format (through an XSL transformation) directly within the program is shown in Figure C-3 and Figure C-4 respectively.

Data Processing

The new version of Microsoft's suite of office programs, XP, offers full XML reading and writing capability. Other office productivity software tools are expected to follow suit. This simplifies the process for importing results data from a traffic analysis program for post-processing, as you can import data directly into applications such as MS Word or MS Excel.

It is important to note that, while XML is an excellent medium for data exchange, it is rarely suitable as a medium for data storage. It is very inefficient for storage because of the high "overhead" imposed by the need to store the tags in their entirety. It also lacks the functions of sorting, query etc. found in data base managers. On the other hand, a growing number of data base managers offer XML import/export capabilities.

Summary

This paper has described TMML as resource for traffic model data representation, storage, rendering and exchange. The TMML concepts have been described and examples of their use have been presented.

TMML offers one approach to the adaptation of XML for these purposes. The question is not *should* XML be used for traffic model data, but *how will* XML be used for traffic model data. TMML is an attempt to answer that question with a specific example.

TMML has built a solid footing in the traffic-modeling field. It has established a strong client base in the number of software products that utilize it. It is fully open in structure and can be expanded to include an unlimited number and variety of other software products and purposes. What is lacking now is a wider recognition and adoption by the traffic modeling community. Hopefully, this paper will further that cause.

References

1. Bosak, J. and T. Bray. XML and the Second-Generation WEB. Scientific American, May 1999, pp.89-93,

2. IEEE. Draft Standards for Data Dictionaries for Intelligent Transportation Systems - Part 1 Functional Area Data Dictionaries. IEEE P1489/D0.1.2, Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ, USA. 1999.

3. Transportation Research Board. Highway Capacity Manual. Washington D.C., USA. 2000.

4. University of Florida Transportation Research Center. Traffic Model Markup Language (TMML) Specification, Working Draft. Gainesville, FL, USA. 2000.

5. World Wide Web Consortium, Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, Oct. 6, 2000. http://www.w3.org/TR/2000/REC-xml-20001006. Accessed July 25, 2001.

```
<TMML Facility = "Arterial">
<GENERAL>
</GENERAL>
<AGENCY>
</AGENCY>
<ARTERIAL ID=#>
<INTERSECTION ID=(# or All>
                                                                    <TMC Begir
            <CONTROLLER>
            </CONTROLLER>
            <TIMINGPLAN Phase=(# or All)>
                  <PHASECODES Approach=(# or Code)>
                  </PHASECODES>
            </TIMINGPLAN>
            <APPROACH ID=(#, Code or All)>
                  <LANEGROUP ID=(#, Code or All)>
                  </LANEGROUP>
                  <LANE ID=(#, Code or All)>
                  </LANE>
                  <MOVEMENT ID=(#, Code or All)
                  </MOVEMENT>
                  <SUBSEGMENT ID=# >
                  </SUBSEGMENT>
                  <ODMATRIX>
                  </ODMATRIX>
            </APPROACH>
      </INTERSECTION>
      <MOEGROUP ID=(# or Code)>
      </MOEGROUP>
      <MODELPARAMETERS>
      </MODELPARAMETERS>
      <LOSTABLES Lanes =#>
            <PEAKDIRECTION LOS=(A-E)>
            </PEAKDIRECTION>
            <BOTHDIRECTIONS LOS=(A-E)>
            </BOTHDIRECTIONS>
      </LOSTABLES>
</ARTERIAL>
</TMML>
```

Figure C-1. TMML Class Structure for Signalized Arterials

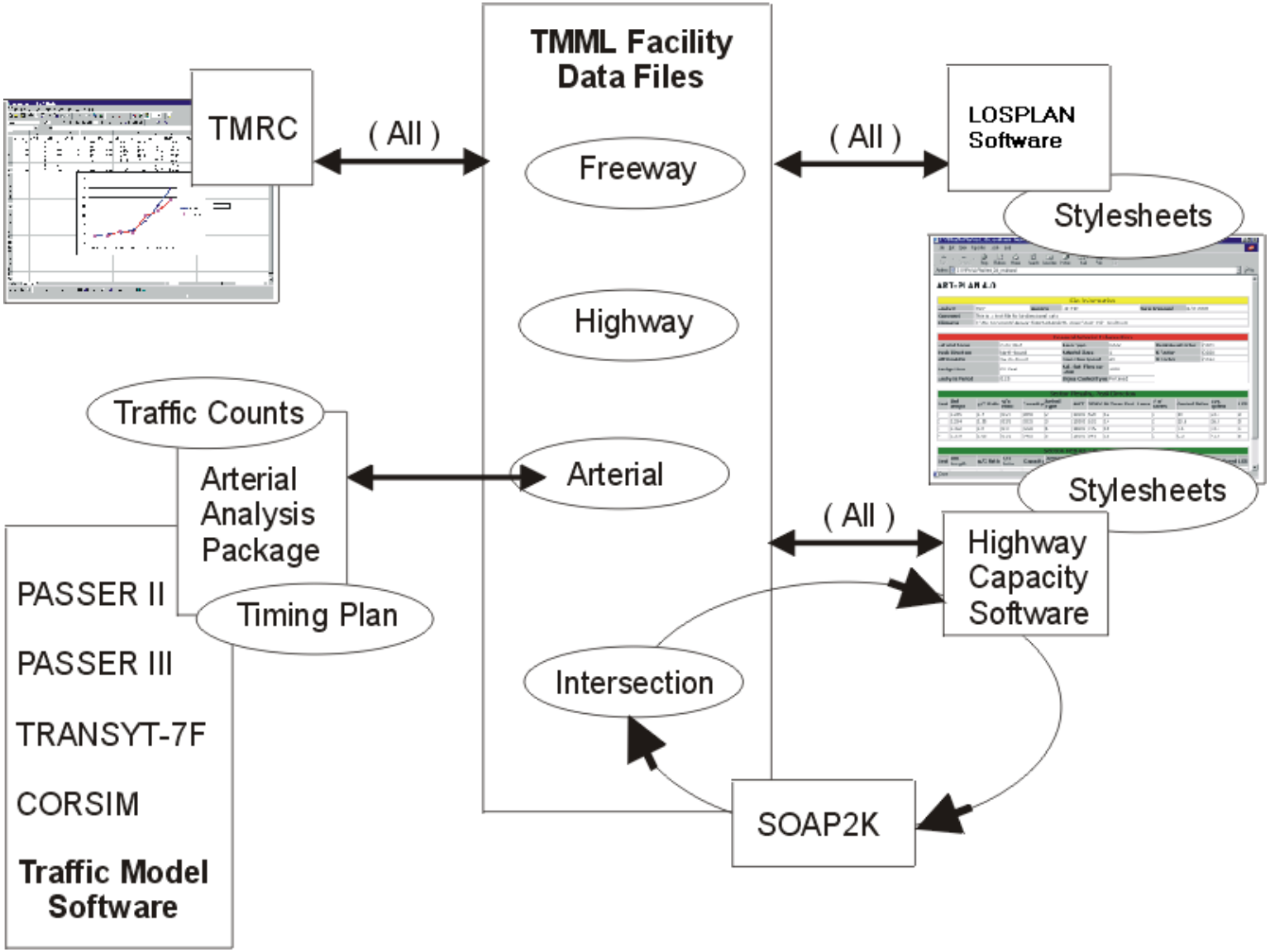


Figure C-2. Data Flow in Current TMML-Based Applications

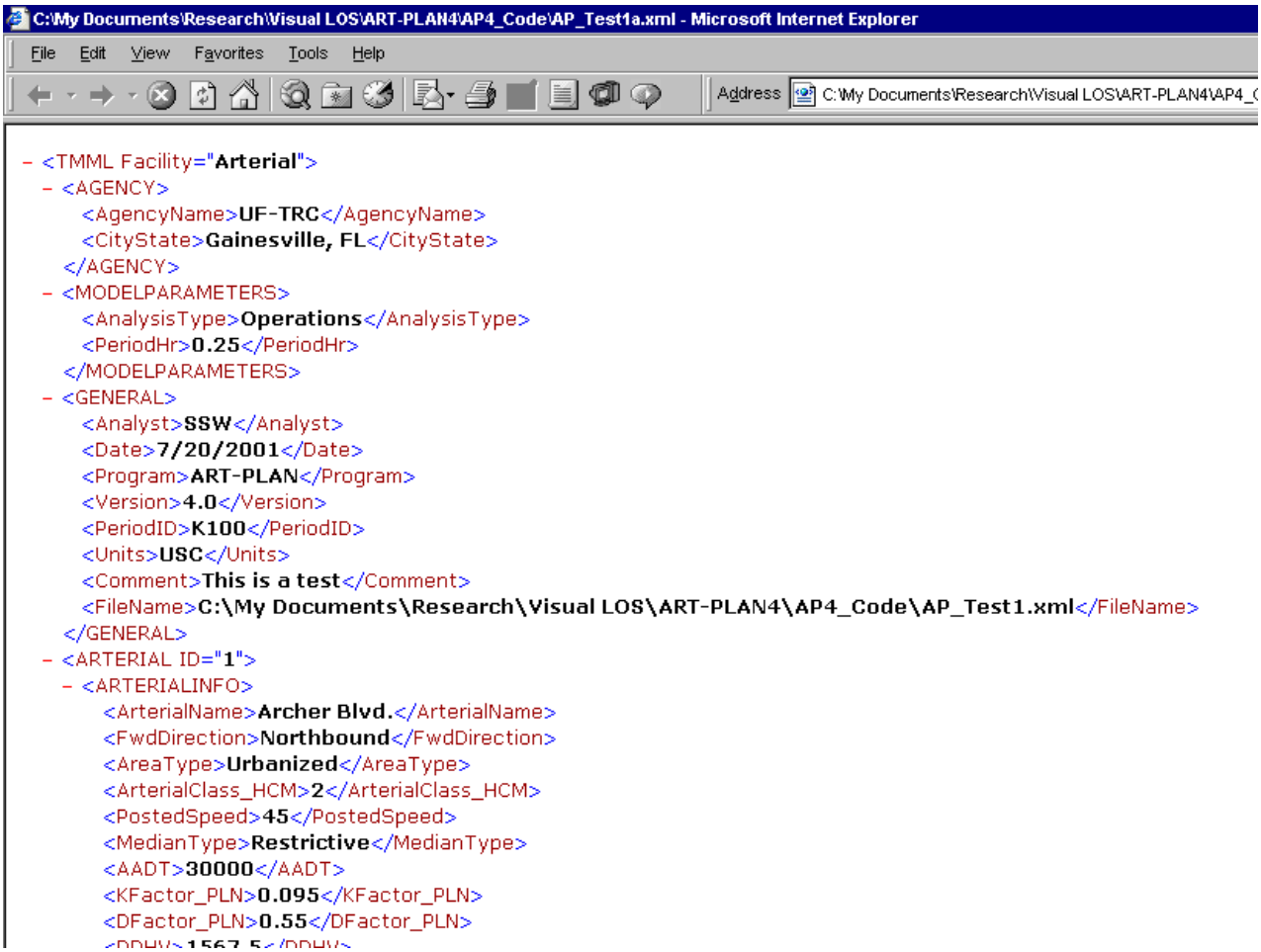


Figure C-3. Example TMML Display Formats

Native TMML Display Format in Internet Explorer Web Browser

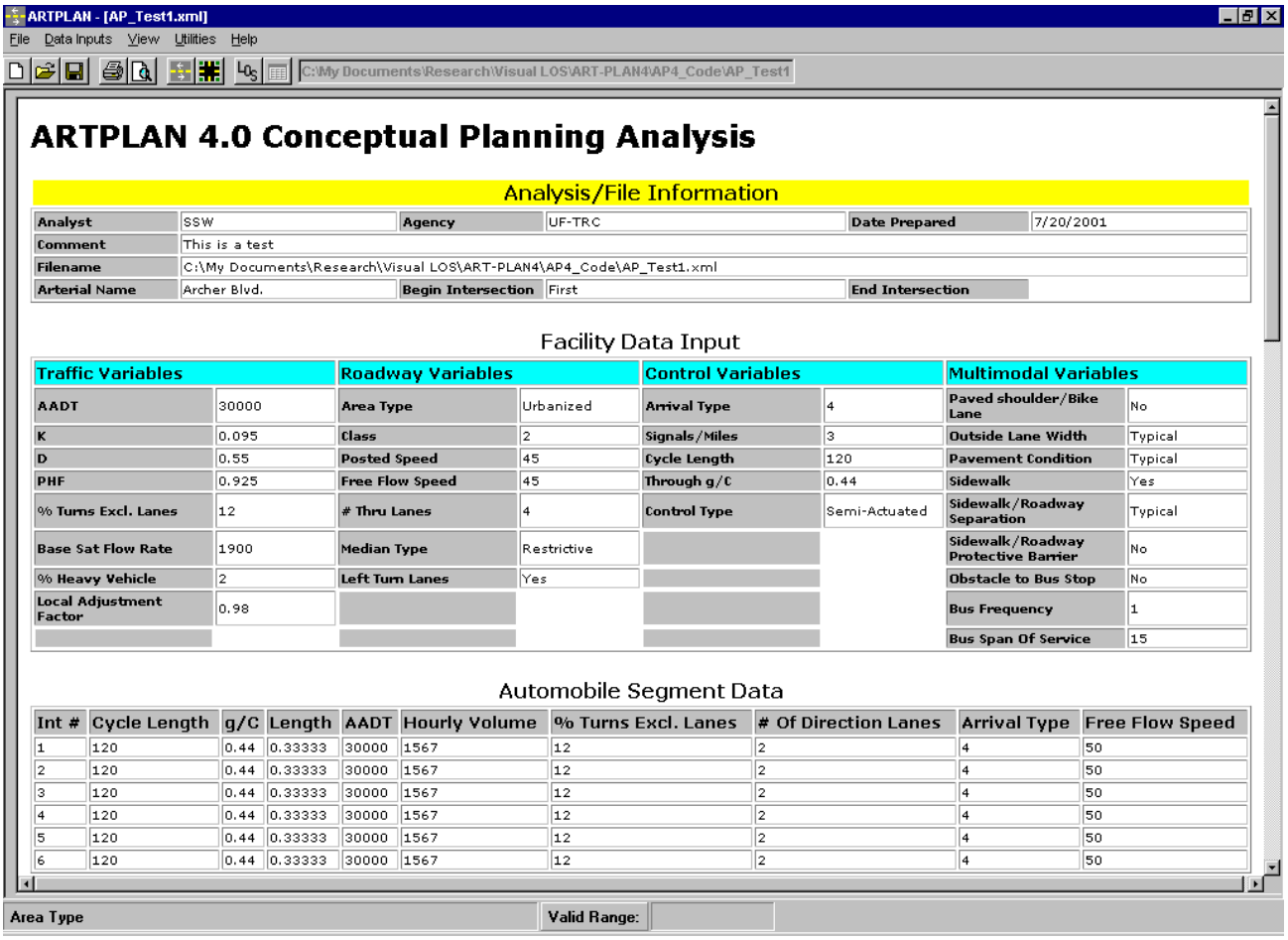


Figure C-4. Example TMML Display Formats
Custom Report Display in ARTPLAN Through XSL Transformation