VEEP: VEHICLE ECONOMY, EMISSIONS, AND PERFORMANCE PROGRAM

Donald A. Heimburger and
Marcia A. Metcalfe

Jet Propulsion Laboratory
California Institute of Technology

ABSTRACT

VEEP is a general-purpose discrete event simulation program being developed to study the performance, fuel economy, and exhaust emissions of a vehicle modeled as a collection of its separate components. It is written in SIMSCRIPT II.5. The purpose of this paper is to present the design methodology, describe the simulation model and its components, and summarize the preliminary results. Topics include chief programmer team concepts, the SDDL design language, program portability, user-oriented design, the program's user command syntax, the simulation procedure, and model validation.

I. INTRODUCTION

The Jet Propulsion Laboratory (JPL) of the California Institute of Technology has become significantly involved in the analysis of automotive vehicle systems and their components, such as transmissions and power sources [1, 2, 3]. By 1976 there was a clear need for a general-purpose simulation model to provide an institutional capability to analyze a wide variety of vehicles, using their separate chassis, heat-engine, transmission, and drive train components. A survey and analysis was made of the existing, well-known simulation programs to determine if the present and future requirements could be satisfied using an existing program [4, 5, 6, 7, 8]. Due to differing objectives, lack of portability and documentation, degree of specialization, or difficulty of modification, the decision was made to develop a structured model in a series of phases.

The first phase of development has the following objectives: (1) laying the basic framework for a general-purpose simulation program capable of an evolutionary expansion in future phases, (2) providing the capability to simulate the performance, fuel economy, and exhaust emissions of an automotive vehicle as a collection of its separate components, and (3) allowing the user to enter, list, delete, save on file, and recall all the component data in either a conversational or a batch mode. It was decided, with the user's interest in mind, to design a helpful, courteous program with built-in prompting, tutorial and debugging features. Additionally, the maintainability and portability of the program is a constant consideration in the design and code.

The Energy Research and Development Administration (ERDA) is sponsoring an activity at JPL referred to as the Automotive Technology Status and Projections (ATSP) project. The VEEP program development is one of several tasks comprising the ATSP effort. Since it is being produced in the public domain, the program code, complete documentation, and the user's guide will be sent to the Computer Software Management Information Center (COSMIC) in the prescribed NASA procedure to facilitate distribution to other interested users.

This paper will first explain the design methodology used by the Chief Programmer Team (CPT). Next, it will discuss the components of the simulation model which comprise the program's data structure and their high-level manipulation commands. Then the simulation procedure and report options will be addressed. Finally, the Phase I results will be reviewed, and planned Phase II activities will be presented.

II. MODEL DESIGN AND DEVELOPMENT METHODOLOGY

The CPT approach to program development is well-suited for constructing computer simulation models [9, 10]. The team assembled for the VEEP program development was also trained in design languages [11], "ego-less" programming, "Joint Programmer Reader Teams" [12], structured walkthroughs of both design and code [13], the use of internal and external program libraries (with a librarian) [14], and rate charting [15]. This section explains the methodology used to design and develop the VEEP program.

A. THE CHIEF PROGRAMMER TEAM

The Chief Programmer Team organization emphasizes the delegation of responsibilities through roles. The literature adequately documents these responsibilities, so they will not be detailed here. The team size has varied from two to seven persons and has included the following roles: the chief

*This paper presents the results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under Contract NAS7-100, sponsored by the National Aeronautics and Space Administration.

programmer, project manager, design leader, support programmer, backup programmer, program librarian, documentarian, automotive design engineer, automotive design consultant and user community representative. Some members have performed more than one role, but have clearly kept their activities separated.

Another characteristic of the chief programmer team is the handling of the internal and external libraries, and the use of standard project procedures. The internal libraries are the machine-readable design document and the source code listings. The program librarian is given standard procedures to update, maintain, list and distribute copies of both the design document and the source code.

B. THE DESIGN DOCUMENT

The Software Design Document (SDD) contains the model's necessary data requirements and English specifications of the simulation procedures. The team uses a computer program and an advanced high-speed, non-impact printer to generate new design documents weekly for each team member. The program used is the JPL-developed Software Design and Documentation Language (SDDL) Processor [16].

The SDDL program was designed to facilitate communications during (and after) the software development process. It takes as input a card-image computer file containing SDDL directives and high-level design specifications. It produces as output a clear, unambiguous declaration of each component of the data structure and specific procedures to act upon the data structure to fulfill each of the program specifications.

The VEEP program's specifications are expressed in the form of a user command syntax that the program should act upon. The SDD is used to specify exactly which procedures should be performed to execute each allowed command. In each procedure of the SDD, the program clearly shows the flow of logic by: (1) automatically indenting lines (to denote selective conditions, looping and substructures), (2) supplying left arrows (showing loop escapes and and procedure exits), and (3) supplying right arrows with page numbers (for invocation of lower-level procedures). Figure 1 is a sample procedure from the VEEP SDD.

The SDD has been a successful communications medium for the automotive engineers, program designers, managers, sponsors, automotive industry simulation experts, and coders who each used the document for their particular requirements. In addition to the automatically generated table of contents, sections are included for a calendar of project events, memoranda, objectives, acknowledgements, reading conventions and future considerations. The SDDL processor automatically produces a simulation calling sequence diagram; it is also easy to obtain cross references on data structure items, procedure names, revision notices, update assignments, and footnotes. The SDD has been very useful to management in assessing the project's progress and status.

C. THE DESIGN DEVELOPMENT METHODOLOGY

The literature and existing programs that pertained to the simulation of automotive vehicles were reviewed. The team visited industry and government experts to discuss our objectives, their programs, available data and detailed modeling methods. The resulting information, together with the SDDL processor, made it possible to represent the main components of the data structure and the top level of the simulation procedure in the first SDD.

Each week the team met to discuss and submit revisions and refinements to procedures for which they were responsible. This top-down development approach has been used continually throughout the project. Technical subgroup discussions were used between weekly meetings to discuss design and modeling alternatives. Monthly review meetings were held for upper management. Structured walkthroughs of the evolving design and program code were often included in the management meetings.

D. THE DESIGN PHILOSOPHY

The VEEP program was conceived as an interactive program that could also be run in a batch mode. Flexible, easy-to-use commands were designed to accommodate persons with minimal computer familiarity [17]. By entering a "?", the user can get a list of available commands at different places in the program. He or she can also type HELP followed by any command name to get a brief tutorial about the command and an example of its usage.

The ENTER, LIST, SAVE, LOAD, DELETE, and CHANGE commands were designed to allow the user to manipulate the associated data for each of eight easily conceptualized components (explained in Section IV). (A complete external file database is created and maintained by the program.) The PROMPT command can be used to set program prompting at full, intermediate, brief, or none to accommodate and expedite a wide range of potential users.

E. THE COMPUTER PROGRAM

The chief programmer, support programmer, and backup programmer are translating the English-like design into executable computer code. The programming language selected was SIMSCRIPT II.5 [18]. The simulation model data is easily represented using the "world view" of entities, their attributes and sets. Each basic component in the model was coded as an entity. The data that described that component was the entity's attributes. Whereas most of the attributes are scalar data values, some are pointers to tabular data. When optional or irregularly spaced data (such as in the dynamometer map) made tabular representation unsuitable, sets were used to associate the smaller aggregations of data with their owning entity. The selection of SIMSCRIPT II.5 allows the program to be used on computers manufactured by International Business Machines, Sperry UNIVAC, Honeywell, and Control Data Corporation. SIMSCRIPT II.5 is also available on several computer time-sharing systems.

```
LINE
188    PROCEDURE TO SIMULATE_DRIVING_CYCLE                                    PAGE 76
189                                                                           DAH*
190      NOW GET_THE_VEHICLE_INITIAL_CONDITIONS AND SIMULATION.TIME . . . . . !HK! > (108)
191      * YIELD: VEHICLE.STATE: (CURRENT)
192      *        SIMULATION.TIME (GLOBAL)
193      *        SIMULATION FLAG
194
195      IF SIMULATION FLAG = "NOGO"
196    <----EXIT_PROCEDURE FOR FURTHER USER INTERACTION
197      ENDIF
198
199      LOOP UNTIL THE DRIVING CYCLE SEGMENTS ARE ENDED (ANYTHING BUT A SEGMENT CARD ENDS THE SCHEDULE)
201
202        NOW DEFINE_THE_DRIVING_CYCLE_STEP AND END CONDITIONS . . . . . >(109)
203        * GIVEN:   VEHICLE.STATE: (CURRENT),
204        * YIELD:   STEP.CONTROL.SPECIFICATIONS:
205
206        SELECT SIMULATION TYPE FOR STEP.CONTROL.SPECIFICATIONS: SIMULATION.TYPE
208
209        CASE 1   "SPECIFIED ACCELERATION"
210          (USED FOR CONSTANT ACCELERATION, CONSTANT VELOCITY,
211          VELOCITY-TIME)
212          NOW PERFORM_SPECIFIED_ACCELERATION . . . . . >( 77)
213          * GIVEN:   VEHICLE.STATE: (CURRENT)
214          *          STEP.CONTROL.SPECIFICATIONS:
215          * YIELD:   VEHICLE.STATE: (CURRENT) UPDATED
216
217        CASE 2   "SPECIFIED PERCENT PEDAL"
218          NOW PERFORM_SPECIFIED_PERCENT_PEDAL . . . . . >( 80)
219          * GIVEN:   VEHICLE.STATE: (CURRENT)
220          *          STEP.CONTROL.SPECIFICATIONS:
221          * YIELD:   VEHICLE.STATE: (CURRENT) UPDATED
222
223        CASE 3   "SPECIFIED ACCELERATION LIMIT AND HOLD THROTTLE"
224          NOW PERFORM_ACCELERATION_TO_SPECIFIED_LIMIT . . . . . >( 81)
225          * GIVEN:   VEHICLE.STATE: (CURRENT)
226          *          STEP.CONTROL.SPECIFICATIONS:
227          * YIELD:   VEHICLE.STATE: (CURRENT) UPDATED
228
229          IF THE STEP IS NOT ENDED
230
231            NOW PERFORM_SPECIFIED_PERCENT_PEDAL . . . . . >( 80)
232            * GIVEN:   VEHICLE.STATE: (CURRENT)
233            *          STEP.CONTROL.SPECIFICATIONS:
234            * YIELD:   VEHICLE.STATE: (CURRENT) UPDATED
235          ENDIF
236
237        ENDSELECT SIMULATION TYPE
238
239      REPEAT UNTIL DRIVING CYCLE SEGMENTS ARE ENDED
240
241      NOW PRINT_STANDARD_REPORTS . . . . . >(157)
242      * GIVEN:   TYPE = 3 (DRIVING CYCLE).
243
244    END_PROCEDURE
```

FIGURE 1. VEEP SDD PROCEDURE

The program is being structured modularly with maintainability in mind, since maintenance can cost upward of 70% of a program's total lifetime cost [19]. The routines are being coded and tested in a top-down, structured manner. Each module can easily be changed and replaced. Tracing and testing options are being built into the program as each new routine is added to the internal code library.

The program has a high-level parsing capability designed to make expansions relatively simple. Additional commands, components, and options can be added to the program's vocabulary, allowing unlimited evolution of the basic Phase I framework.

III. MODEL COMPONENTS

The data for this model was conceptualized into eight (8) logical components:

(1) Vehicle: the collection of vehicle components which are to be simulated.
(2) Chassis: the body and running gear of the vehicle.
(3) Engine: the vehicle's power source.
(4) Transmission: the means of providing gearing and coupling between the vehicle's engine and its drive power train.
(5) Drive power train: the means by which power from the transmission is transferred to the vehicle's drive wheels.
(6) Driving schedule: the specifications for vehicle state at various points in time.
(7) Environment: the environmental conditions.
(8) Simulation specifications: the various time parameters and simulation conditions.

In modeling, the components of a system are represented by abstracting the attributes of importance to the simulation. Each of the eight logical components was represented as a SIMSCRIPT entity, whose first attribute is its name. The user can specify a unique name to identify all the data for any component, and he may also enter as much free-format textual description as is desired.

A. VEHICLE

The vehicle is the conceptual data structure that allows the user to group together a set of vehicle components for simulation. The four basic components of a vehicle are its chassis, engine, transmission, and drive power train (Figure 2). A brief discussion of each follows:

1. Chassis

The chassis is the physical entity upon which the other vehicle components are "mounted." It is characterized by such attributes as test weight, frontal area, drag coefficient, wheelbase, drive wheel (front or rear), etc. Additionally, the entity also has a set called "other losses", where the user can enter other torque or power losses. The loss data can be in the form of a table or an equation.

2. Engine

The engine is the power source of the vehicle. It is characterized by such attributes as maximum horsepower, maximum revolutions per minute (rpm), idle rpm, idle fuel flow, and both minimum and maximum torque boundaries (as a function of speed). The engine's dynamometer map represents the torque, flow of fuel, and exhaust emission as a function of its speed and throttle setting. In addition to a set of "other losses" (which are user-specified), there is also a provision for start-up losses.

3. Transmission

The transmission is the vehicle component which provides a means of gearing and coupling between the engine and the drive power train. The user has two alternatives as to the type of transmission the vehicle may have: (1) direct gearing, which has one gear, or (2) discrete gears, where the number of gears is specified by the user. The coupling type can be either a torque converter (for an automatic transmission) or a clutch (for a manual transmission). Like all other vehicle components, the transmission has a set of "other losses" which may be entered by

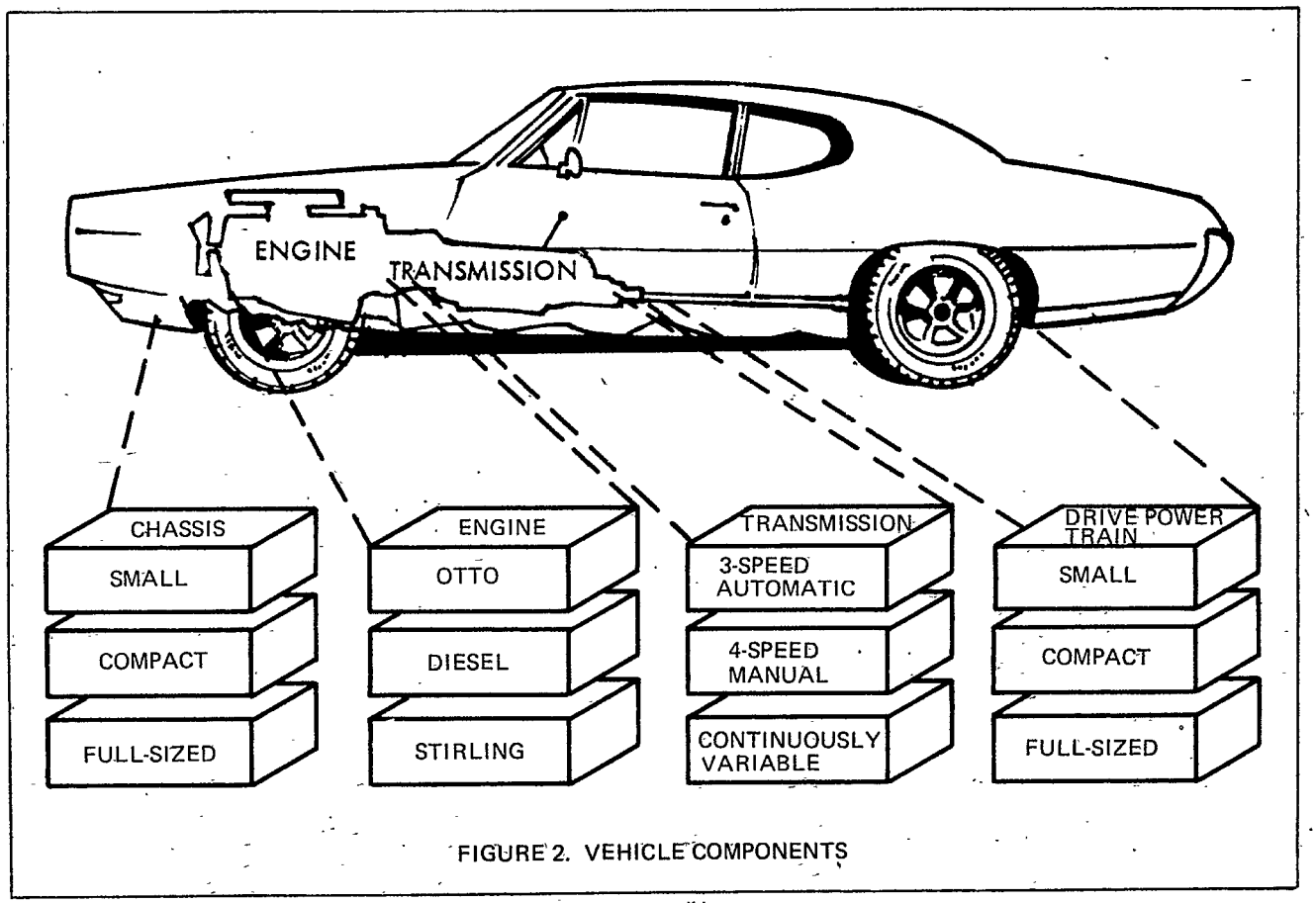

FIGURE 2. VEHICLE COMPONENTS

the user.

4. Drive Power Train

The drive power train is the means by which the power from the engine and the transmission is transferred to the drive wheels. It is typified by such attributes as tire rolling radius, wheel inertia (for each wheel), and rolling resistance. It also has an optional set of "other losses."

B. DRIVING SCHEDULE

The driving schedule is the mode of operation under which the vehicle is to be simulated (or "driven"). There are three alternatives from which to choose: (1) steady state, (2) maximum performance, and (3) Federal driving cycle (Figure 3). The user can also enter a particular driving schedule of his own. Each mode has associated with it a set of initial conditions and preselected reports.

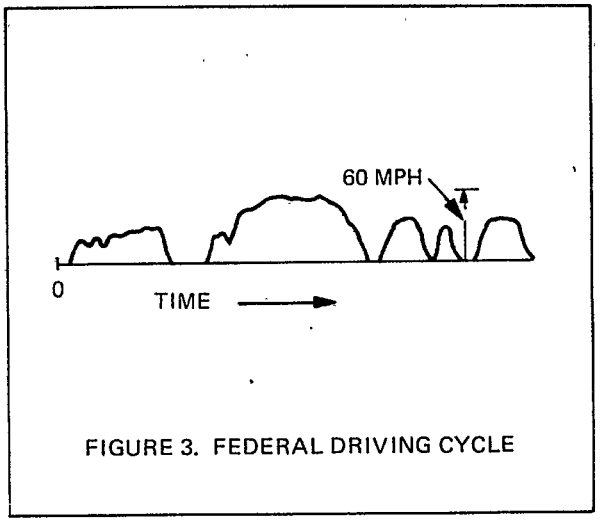

FIGURE 3. FEDERAL DRIVING CYCLE

C. ENVIRONMENT

The environment contains the environmental conditions under which the vehicle is to be simulated. It includes such data as the headwind velocity and road gradient (both functions of distance), ambient air pressure, ambient air temperature, and ambient air humidity.

D. SIMULATION SPECIFICATIONS

As implied by its title, simulation specifications include the specifications under which the simulated vehicle is to be "driven." The user may specify that the vehicle is to be simulated under dynamometer conditions or road load conditions. Other attributes of this entity include the normal simulation time increment, the wide-open-throttle time increment, and time increment while shifting.

IV. COMPONENT MANIPULATION COMMANDS

To facilitate easy handling of large amounts of data, six (6) manipulation commands are provided for high-level model management capabilities. These commands enable the user to associate an entire data structure with a single unique descriptive name. The manipulation commands, their syntax, and a brief description of each one's function and usage follows:

A. THE ENTER COMMAND

[ENTER / CREATE*] component

The ENTER (CREATE) command allows the user to enter all the necessary data to describe any of the eight model components. When this command is used, the program first checks for a current component (one in core) of the same type. If one is found, the user is given the opportunity to (1) delete it from core, (2) save it on an external file and then delete it from core, or (3) abort the ENTER command. If there is no current component (or if the user deletes or saves the current component) the program then prompts for all necessary information.

B. THE LIST COMMAND

[LIST / DISPLAY* / SHOW*] component (NAME(S) / CATALOG)

This command allows the user to list the information associated with a specified entity. When used, the program first checks that a component of the specified type is current. If it is, all attributes of that component are listed. Otherwise, a message that the component is not current is sent to the user. Optionally, the user can list only the current component name (or vehicle component names) as well as a catalog listing of all previously saved entities of the specified type.

C. THE SAVE COMMAND

[SAVE / FILE*] component

The SAVE (FILE) command allows the user to save all the information about a specified component with its identifying name on an external file. When used, the program first checks for an already saved entity with the same name. If one is found, the save operation is aborted with a message to the user. If no other component with the current name exists on file, the current component is saved; and a message to this effect is printed for the user.

D. THE LOAD COMMAND

[LOAD / RECALL*] component component.name

The LOAD (RECALL) command allows the user to load all the data for a previously saved component from an external file. When used, the program first checks if a component of the specified type is

* Indicates a synonym
[] Indicates a required choice
() Indicates an optional choice

currently in core. If one is, the user is instructed explicitly to delete the current component before loading another of the same type. If there is no current component in core, the catalog is checked for the specified component name. If found, it is loaded and a message to this effect is sent to the user. If the user-specified component name is not found in the catalog, the LOAD command is aborted, and the user is informed accordingly.

E. THE DELETE COMMAND

DELETE component

This command allows the user to delete the current component from core. When used, the program first checks that the specified component currently exists in core. If it does, the component is deleted; and the user is informed accordingly. If one is not found, the DELETE command is aborted and the user receives a message to this effect.

F. THE CHANGE COMMAND

CHANGE component attribute.name TO value

The CHANGE command allows the user to change values for selected attributes of components. When used properly, a verification message is sent to the user. Appropriate warnings are issued whenever there is improper usage.

V. SIMULATION PROCEDURE

The program allows the user to simulate the vehicle in four different modes: steady state, maximum performance, a prescribed driving schedule, or any user-specified driving schedule. A different type of report will automatically be generated for each mode. A REPORT command will allow the user to obtain detailed traces and additional reports. At the time of this writing, the simulation portion of the program is not yet operational. Therefore, the following discussion pertaining to each of the simulation modes is based on the model design.

A. STEADY-STATE SIMULATION

The VEEP program automatically performs a verification of model completeness prior to beginning any of the various simulation modes. The steady-state simulation mode then causes the program to execute the following actions for a series of constant speeds (e.g., 10, 20...80 mph): (1) the initial conditions are examined to check for an infeasible situation, (2) if the conditions are feasible, the required power is calculated at the drive wheels, the transmission output shaft, the transmission input shaft and finally at the engine (deducting all losses for each component), and (3) the standard report information is printed, including fuel consumption, exhaust emission, and rotational speeds at several points in the vehicle.

B. MAXIMUM PERFORMANCE SIMULATION

After verifying model completeness, the maximum performance simulation mode causes the program to iterate over the following steps: (1) the vehicle is set to an initial condition (if feasible), (2) the vehicle is simulated in a wide-open-throttle condition, calculating the output delivered at the engine, the transmission output shaft, and finally at the drive wheels in small time increments until the final conditions are reached, and (3) the standard report information is printed, including 0-60 mph elapsed time, 1/4 mile elapsed time, and 4 and 10 second distances.

C. DRIVING SCHEDULE SIMULATION

The completeness verification is followed by setting the vehicle to an initial condition (if feasible). Then a commonly used set of prescribed velocities (Figure 3) at points in time is used as the driving force to schedule a series of accelerations. Based on the magnitude of the acceleration, the program can perform a wheels-to-engine calculation, an engine-to-wheels calculation, or a combination of both to achieve the desired vehicle state. The model will "move" the vehicle along the prescribed velocities (or as close to them as the vehicle can perform) until the schedule is ended. Then a standard report is printed that includes fuel economy, exhaust emissions, energy consumption and apportionment, and other pertinent engineering statistics.

D. USER-SPECIFIED DRIVING SCHEDULE

The operation of this simulation mode is identical to that of the driving schedule simulation mode above, except that the user specifies his own set of segments rather than using the provided schedule. Each segment can be specified in terms of constant velocity, acceleration, percent pedal, or acceleration to a limit and then holding a constant percent pedal. The end conditions for each segment can be defined by time (absolute or relative), distance (absolute or relative), passing distance, or velocity.

VI. RESULTS TO DATE

At the time of this writing, results from the simulation were not available. They will, however, be presented at the conference. It appears that the Phase I objectives, included in the Introduction, will be met. Current plans include sending to COSMIC the VEEP program source code, the final design document, the program documentation, the program user's guide, and a Society of Automotive Engineers paper on the engineering methodology used.

The SDD, in combination with the CPT and top-down design approaches, has proven to be an effective communications medium so often lacking in the software development process. Both the design and code had the principal objectives of being readable, understandable, and manageable [20], since the VEEP program will most likely outlive several generations of programmers.

VII. PLANNED PHASE II ACTIVITIES

The major thrust of the scheduled Phase II activity will be adding the capability to simulate electric and hybrid vehicles. Additional model components will include the electric motor, a controller, and a storage device. Proposed enhancements include: (1) a driving schedule more appropriate for the simulation of an electric vehicle, (2) simulation procedures that utilize the new model components, (3) an expanded selection of report options, and (4) the addition of a PLOT command that will aid in output verification and understanding.

An important on-going activity of the VEEP program development is validation, verification, and fine tuning. This task, although begun in Phase I, will continue throughout the lifetime of the VEEP program.

ACKNOWLEDGEMENTS

The development of the VEEP program has been a team effort. The task benefitted from the contributions of the following people: T. Barber, J. Bevan, R. Chamberlain, H. Hubbard, S. Jacobs, H. Kleine, G. Klose, R. Norton, R. Stephenson, and T. Vanderbrug.

REFERENCES

1. Finegold, J. G. "Hydrogen: Primary or Supplementary Fuel for Automotive Engines," SAE Paper No. 760609, presented at 1976 West Coast Meeting of SAE, San Francisco, California., Aug. 9-12, 1976.

2. Hoehn, F. W., Baisley, R. L., and Dowdy, M. W. "Advances in Ultralean Combustion Technology Using Hydrogen Enriched Gasoline," paper presented at 10th Inter-Society Energy Conversion Engineering Conference, University of Delaware, Newark, Delaware, Aug. 17-22, 1975.

3. Jet Propulsion Laboratory, Should We Have a New Engine?, SP 43-17, Jet Propulsion Laboratory, Pasadena, California, August 1975.

4. Beachley, N. H., and Frank, A. A. Increased Fuel Economy in Transportation Systems by Use of Energy Management, Report No. DOT-TST-75-2, December 1974.

5. Hwang D. N. "Fundamental Parameters of Fuel Economy and Acceleration," SAE Transactions 1969, SAE Paper No. 690541, pp. 1963-1977.

6. Waters, W. C. "General Purpose Automotive Vehicle Performance and Economy Simulation," SAE Transactions 1972, SAE Paper No. 720043, pp. 216-233.

7. Malliaris, A. C., Withjack, E., and Gould, H. "Simulated Sensitivity of Automotive Fuel Economy, Performance and Emissions," SAE Paper 760157, presented at Automotive Engineering Congress and Exposition, Detroit, Michigan, Feb. 23-27, 1976.

8. Porjes, H. S., Speisman, C., and Young, P. H. "An Advanced Computer Program for Determining Vehicle Emissions and Fuel Economy Within a Surface Street Network," Proceedings of the 1975 Winter Simulation Conference, December 1975, pp. 497-501.

9. Baker, F. T. "Chief Programmer Team Management of Production Programming," IBM Systems Journal 11, 1 (1972), pp. 56-73.

10. Baker, F. T., and Mills, H. D. "Chief Programmer Teams," Datamation 19, 12 (December 1973), pp. 58-61.

11. Van Leer, P. "Top-Down Development Using a Program Design Language," IBM Systems Journal 15, 2 (1976), pp. 155-172.

12. Weinberg, G. M. The Psychology of Computer Programming, Van Nostrand Reinhold, New York, 1971.

13. Yourdon, E., How To Manage Structured Programming, Yourdon, Inc., New York, N. Y., 1976.

14. McGowan, C. L., and Kelly, J. R. Top-Down Structured Programming Techniques, Petrocelli/Charter, New York, N. Y., 1975.

15. Snyder, T. R. "Rate Charting," Datamation 22, 11 (November 1976), pp. 44-47.

16. Kleine, H. SDDL - Software Design and Documentation Language, Publication 77-24, Jet Propulsion Laboratory, Pasadena, California, May 15, 1977.

17. Martin, J. T. Design of Man-Computer Dialogues, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

18. Kiviat, P. J., Villaneuva, R., and Markowitz, H. M. The SIMSCRIPT II.5 Programming Language, CACI, Inc., Los Angeles, California, 1975.

19. Stay, J. F. "HIPO and Integrated Program Design," IBM System Journal 15, 2 (1976), pp. 143-154.

20. Kleine, H., and Morris, R. V. "Modern Programming: A Definition +," SIGPLAN Notices 9, 9 (September 1974), pp. 14-17.