USER PERFORMANCE EVALUATION BASED ON A USER MODEL FOR A GRAPHICAL USER INTERFACE ENVIRONMENT

Technical Report 94-05

Yaowem Wang

Department of Computer Science and Engineering
Auburn University
Auburn University, AL 36849-5347

March 17, 1994

REPORT ABSTRACT

USER PERFORMANCE EVALUATION BASED ON A USER MODEL FOR A GRAPHICAL USER INTERFACE ENVIRONMENT

Yaowen Wang

Master of Computer Science and Engineering
December 16, 1993
(M.S., Auburn University, 1992;
B.Eng., China Textile University, 1982)

130 Typed Pages

Directed by Dr. Kai H. Chang

Graphical user interface technique has been used more and more in software design as the workstation and the window technologies continue to advance. The concerns of this project are the design and implementation of a user model and the user performance evaluation based on the user model for a graphical user interface environment. It monitors the human-computer interaction by interpreting the user actions under varies states. A simulated car model and its graphical user interface have been previously developed. This project is to evaluate the driver performance by using a user model. It also can be extended to evaluate the performance of other system components.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The field of human-computer interaction is at present attracting many researchers. One important reason for its popularity is that neglecting the human user in the design of computer systems is an expensive oversight. Another reason lies in the challenge of the problems themselves. A user interface is a mechanism which handles the human-computer interaction. Its appropriateness and efficiency can determine the system's overall performance. Recently, graphical user interface(GUI) technique has been adopted more and more in software design, as the workstation and the window technologies continue to advance. Its advantages include the ease of use and operation, and the clarity of the presentation. It also provides design and implementation flexibility. Graphical user interfaces are likely to become an industrial standard in the near future. Hence, it becomes an important subject in the research of human-computer interaction.

A major challenge, which arises from the inclusion of potential users in the design and development process, is to provide a realistic graphical user interface in the early design stage of the system. Using user model to evaluate

the performance of the graphical user interface is an approach that is studied only recently. A user model monitors the human-computer interaction that takes place between the graphical user interface and the simulator of a target system. The performance of the user is then evaluated by the user model. The evaluation results reflect the performance of different graphical user interfaces.

This project concerns with the design and implementation of a user model for a graphical user interface environment. A simulator for an automobile and its graphical user interface have existed. The user model created in this project is to evaluate the performance of the user on the simulator. It is anticipated that the user model working with the simulator allows a system designer to experiment and evaluate the performance of different graphical user interfaces. Therefore, a graphical user interface with the best performance can be found.

The user model can also be used to design so-called adaptive tutorial system in this GUI environment. Such system aims at adapting its communication to the current needs of the users, in which the relevant information is provided to the user's current model of the task to be finished.

II. LITERATURE REVIEW

EVALUATION APPROACH

Using a user model to evaluate a graphical user interface interests many researchers recently. However, the research on the evaluation of the user interface has been conducted for more than a decade. A verity of methods have been investigated to evaluate the quality of user interfaces under different environment from several aspects.

E.A. Edmonds[1981] used an evaluation process to build an adaptive program for a tutorial system. The evaluation result was fed back to the adaptive system. The attributes of the graphical user interface were then modified until the optimum design were achieved.

Ricky E. Savage and James K. Habinek[1984] wrote a article about achieving design and evaluation objectives for the graphical user interface through simulation. The first phase of the approach was to complete preliminary design and evaluation. The second phase was to perform redesign and reevaluation based on the results of the first phase. Analysis of the errors made by participants in the interface simulation studies led to the development of hypotheses concerning user choice behavior. For example, novice users normally had difficulty selecting menu options based on job

titles. Their finally designed interface appeared to accommodate both the novice user, through an extensive hierarchy of menus, and the experienced user, through a variety of shortcuts to system functions.

D. Meister[1987] proposed that expert systems be evaluated from a behavioral stand-point in a series of four tests. (1) Initial design testing compared the data base and the rules of thumb developed by the knowledge engineer against the data source and the human expert source of the rules. (2) Software testing determined whether the tested software provided reasonable solutions to problems. (3) Developmental testing refined system design by exposing the prototype system to test subjects. (4) Operational testing determined whether users could employ the system effectively. In the operational and developmental testing, the tests of the performance effectiveness, attribute and acceptability are included.

D.R. Sewell, N.D. Geddes, and W.B. Rouse[1987] described a methodology for conducting design evaluation. The methodology was based on the concept of presenting the design principles to operators who might employ the system. In this evaluation six pilots were interviewed with the possible application of the graphical user interface design to be used for jet fighter aircraft cockpits. They were presented each design feature alone and asked to rate the feature for viability, acceptability, validity, and

desirability. They were also asked to provide free protocol reactions to the design features. The features were redesigned again based on the evaluation. The result showed pilots found the new design viable, valid, and acceptable.

M. Paetau[1985] mentioned a "Human-computer Interface Laboratory" in which selected elements of user friendliness could be created in an rapid prototyping process and evaluated with social scientific methods. The general guideline of this research was the cognitive regulation of human action.

N. Hammond, *et al*, [1985] discussed the evaluation of the interface of a document processor-comparison of expert judgement and user observation. They explored evaluation through (1) assessment by human factors and (2) analysis of user performance. In their evaluation experiments, three pairs of researchers prepared reports on the interface of a document processor. Separately, five novice users were observed learning the system. The two evaluations generated overlapping but separable classes of information. User testing provided low-level information on procedural and conceptual difficulties, while experts provided a more integrated overview and hypotheses concerning the sources of problems.

Liam Bannon and Claire O'Malley[1985] discussed methods of evaluation, their strengths and weaknesses, in the context of a program developed to assist users in getting

quick access to information contained in the UNIX manual. They thought that one of the most difficult aspects of interface design was evaluating new or changed features of an interface.

G. Janke and K. Kohnert[1989] described how the development and psychological examination of an interface could be done by designing and analyzing state-transition-diagrams and equivalent Prolog-programs. This report was about a special aspect of the research on the planning behavior of novice programmers. The experiments showed that a user friendly interface provided the maximum support. Their findings may prospectively be applied within the field of intelligent computer-aided instruction.

EVALUATION CRITERIA

In the research on the evaluation of the graphical user interface, the critical point is to find the criteria for evaluation. Some researchers have done detailed work in this area.

S. Ravden and Grham Johnson[1989] discussed the evaluation of human-computer interfaces. They concluded nine criteria for evaluation: visual clarity, consistency, informative feedback, explicitness, appropriate functionality, flexibility and control, error prevention and correction, and user guidance and support. The evaluation process was to construct the evaluation task, to conduct the

evaluation for different user background and to analyze the results based on the criteria.

Andrew Monk[1987] illustrated the empirical evaluation of user interfaces. How and when to collect the behavior data is the critical point. The behavior data indicated how the system was used and where its strengths and weaknesses lied. He also used the statistical study to evaluate the behavioral data. The results of the study identified the significance of various behaviors. Significance was achieved if the probability of achieving a result is larger than some arbitrary results.

Thomas S. Tullis[1984] conducted his graphical user interface evaluation on screen. His result was the derivation of six measures that could be used to describe the spatial array of characters on any alphanumeric display: overall density, local density, number of groups, size of groups, number of items and layout complexity.

J. Kirakowski and M. Corbett[1989] discussed the evaluation process in four aspect. (1) Determining what was to be measured (hypothesis generation). (2) Selecting the evaluation procedure: naturalistic studies took advantage of existing conditions and was full of extraneous variables out of control by the investigator; the quasi-naturalistic study was quite intrusive; experimental studies were the scientific method par excellence. (3) Allocating subjects to conditions in a study: there were two basic types of

experimental design or ground-plan. In the independent subject type of design, subjects were studied in one experimental condition only; in the repeated measurement type of design, subjects enter more than one experimental condition. (4) The dependent variable: counting events, time measures, reaction time, questionnaires.

From the above literature review, it can be found that the evaluation of a user graphical interface involves the research in the field of artificial intelligence, targeted computer systems design, human-factors and cognitive ergonomics. The user model approach in this project evaluates the performances of a user while interacting with a graphical user interface.

III. PROGRAM DESIGN AND IMPLEMENTATION

USAGE OF A USER MODEL

A human user interacts with a computer system through user interface. A user model is to conceptualize the user of a system. The user model normally consists of two parts. One is the monitor which monitors the interaction between the human user and the computer system. The other is the evaluator which evaluates the user's actions.

The main task in the design of a user model is to design an evaluator. The evaluator is normally composed of a task model and an evaluation methodology. The task model possesses the typical procedural knowledge of how to operate the system. User performance is compared with task model. The difference between the user performance and task model signifies the lack of operation knowledge of the user or an inappropriate interface or system design.

When a user model is used to evaluate the performances of a user or a interface or a target system, the evaluation condition should be strictly controlled. The user, interface and the system can be considered as three independent elements in this human-computer system. If two of the three elements maintains the same, the user model can

evaluate the performance of the rest directly or indirectly. For example, several users operate individually on a system through the same interface. The performance of each user can be evaluated by a user model. If a user operates a same target system through different interfaces, the user model's result shows indirectly the performance of each interface.

THE GRAPHICAL USER INTERFACE ENVIRONMENT

The user model designed in this project runs under a graphical user interface environment(Figure 1). The target system in this environment is an existing car simulator. The user is supposed to be a driver. The graphical user interface is designed to have two panels. One is the information panel which gives out information coming from a test scenario. The other is the control panel by which a user can operate the simulated car. An existing graphical user interface is shown in Figure 2. The user, the graphical user interface and the simulator communicate with each other through a blackboard.

The simulated car with the user graphical interface has the following main properties,

1) The car is a manual shift. The shift gears consists of neutral, first, second, third, fourth, and reverse.

2) The headlights consists of parking lights, low beam, high beam, and fog lights. These lights are controllable.

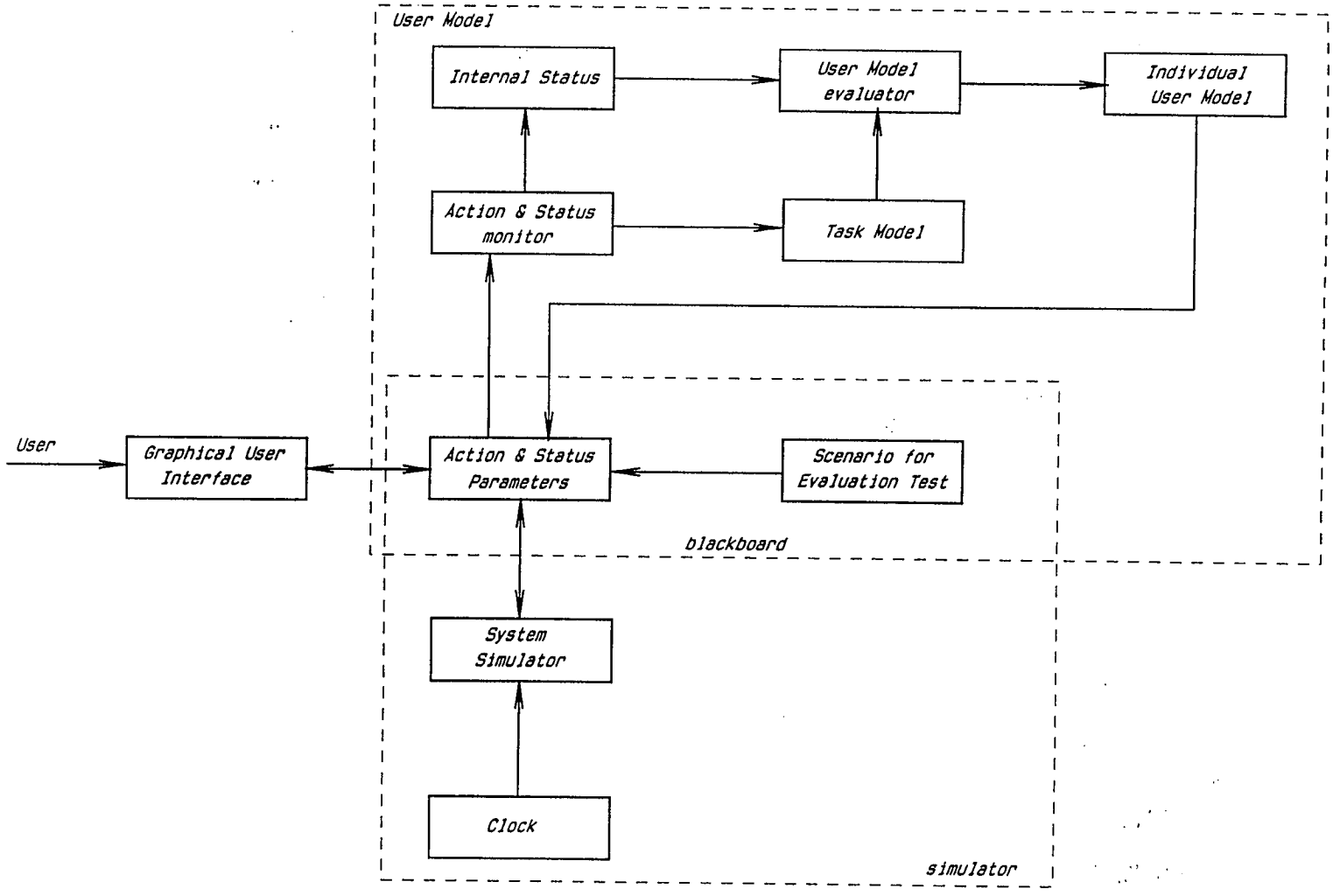

Figure 1.
The User Model in a GUI Environment

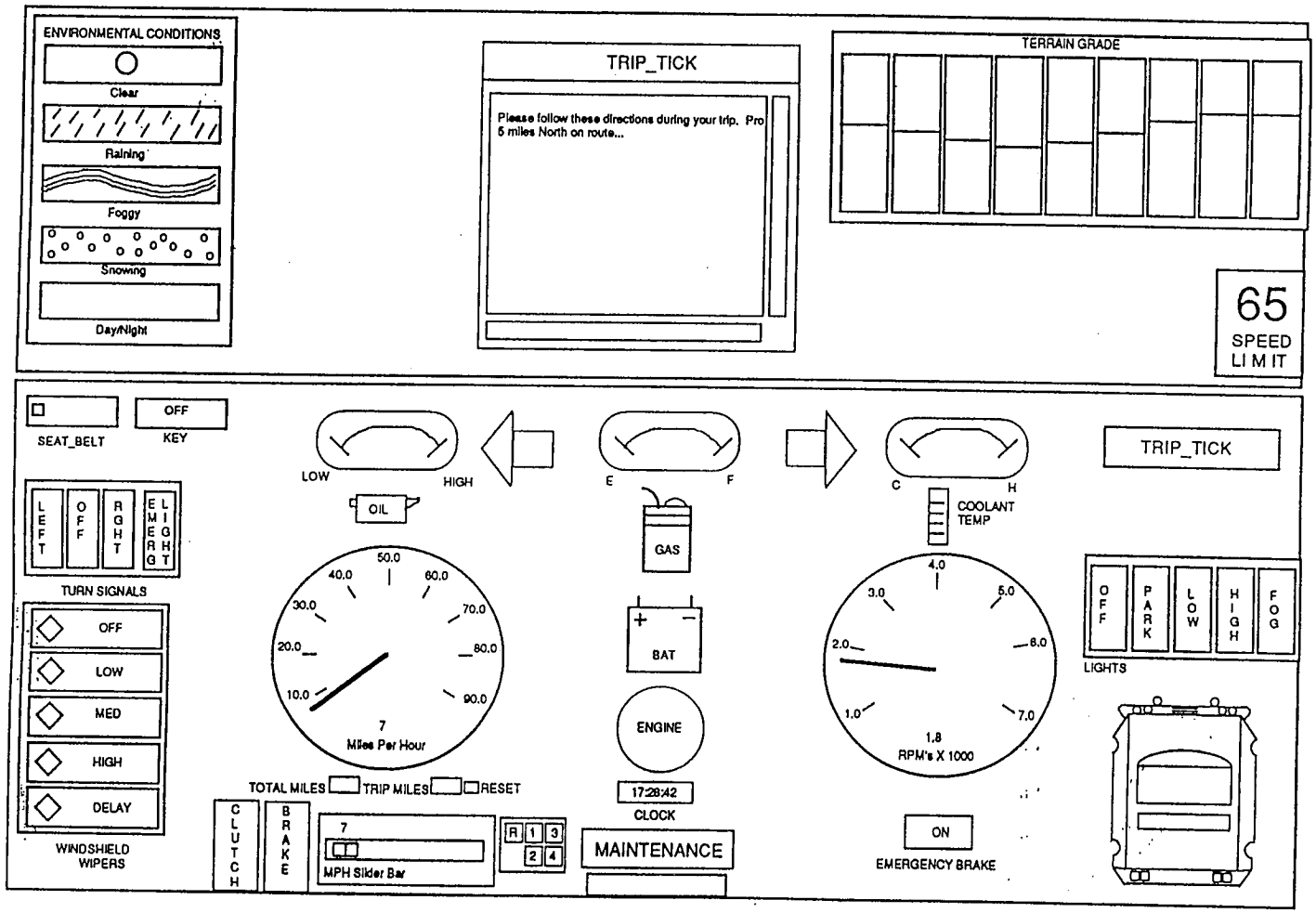

Figure 2.
The Developed Graphical User Interface Environment

3) The windshield wiper has low, medium, and high speeds.

4) The blinker consists of left, right and emergency flash lights.

5) The car has a brake, a gas pedal(accelerator), a cruise control, a speedometer, a mileage meter, etc.

6) The car can run at the maximum speed of 95 miles per hour.

During testing, a user operates this simulated car by manipulating the attributes on the control panel while reading the driving condition supplied on the information panel. The change of the driving condition is dependent on the testing scenario. Each testing scenario consists of a series of events [Appendix A] which may happen to a driver in reality. The user model monitors the user's actions, the car's status and driving environment continuously in a fixed time level. At the same time, the user model evaluates the performance of the user.

THE DESIGN OF THE USER MODEL

In order to monitor the user's actions, two groups of registers are used in the user model. One group is called the cur_status which records the input data at the current clock cycle. The other is named the pre_status which stores the data from the previous clock cycle. Comparing the cur_status to pre_status, the different status of the

maneuverable attributes in the control panel shows the user's current actions. Since each user's action can only be evaluated for correctness under a given driving environment, the user model also reads the current external driving environment from the blackboard in each clock cycle. The evaluation procedures in the user model are activated corresponding to the scenario. A user's actions in driving a car can be divided as the critical or non_critical actions. The critical actions are the actions which effects the driving. Turning off the lights in the night is a critical action which is evaluated as wrong action. The evaluations in this project are based on the critical actions. The non-critical action are the actions which do not effects the driving. For example, the user may release the accelerator for a moment as long as the car keeps running in a correct speed. This action should not be evaluated as a correct or wrong action.

PROCEDURE DESIGN

The designed user model contains two categories, environment category and driving category. The environment category is to evaluate the user's operations on accessories when the driving environment changes. The driving category is to evaluate the user's operations on maneuvering the car.

The Environment Category

Figure 3 is the structure chart of the environment category. The user model continously reads conditions of weather and time from the black board. Then it activates procedure time_day() or time_night() and procedure weather_clear(), weather_foggy, or weather_snow_rain depending on the conditions.

1. TIME

If the time is day, procedure day() is called. If the time is night, procedure night() is called (Figure 4).

Figure 5 shows the typical evaluation process of procedures in the environment category. The evaluation of user's operations is based on the response time and accuracy of the user's actions to finish a task. Three counters are used to record the result of each evaluation in both environment and driving categories. These counter are the expert_counter, normal_counter and the novice_counter, which show how many of user's actions are at the expert level or the normal level or the novice level. Therefore, the model of the user can be established. When a procedure is activated, it reads the status of accessories, such as blinker, wiper and light. If an accessory is in the wrong status and the user does not have an action within a time cycle, the time counter increases one. If a user has an action, the user's action is compared with the task model first. If the user action is wrong, the user model

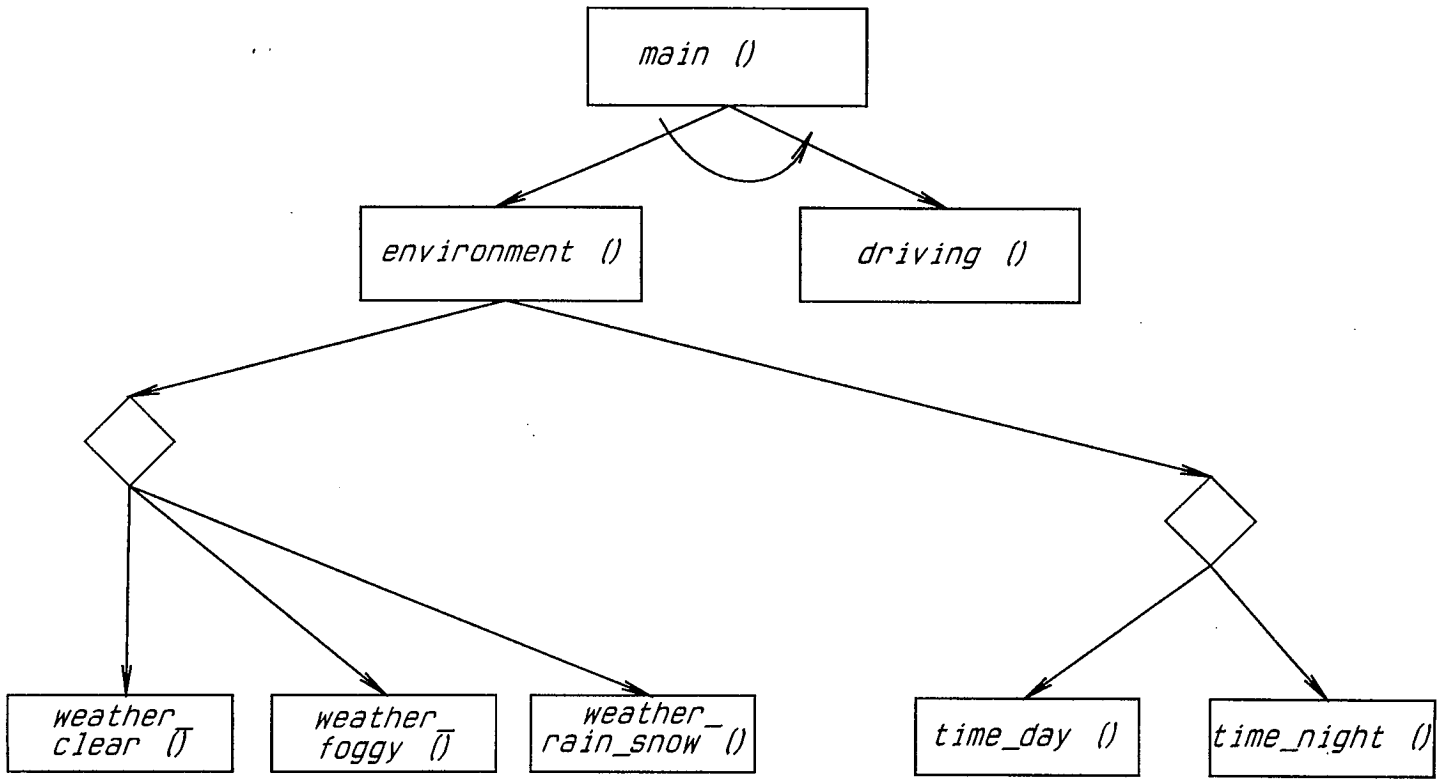

Figure 3.
Structure Chart of the Environment Category

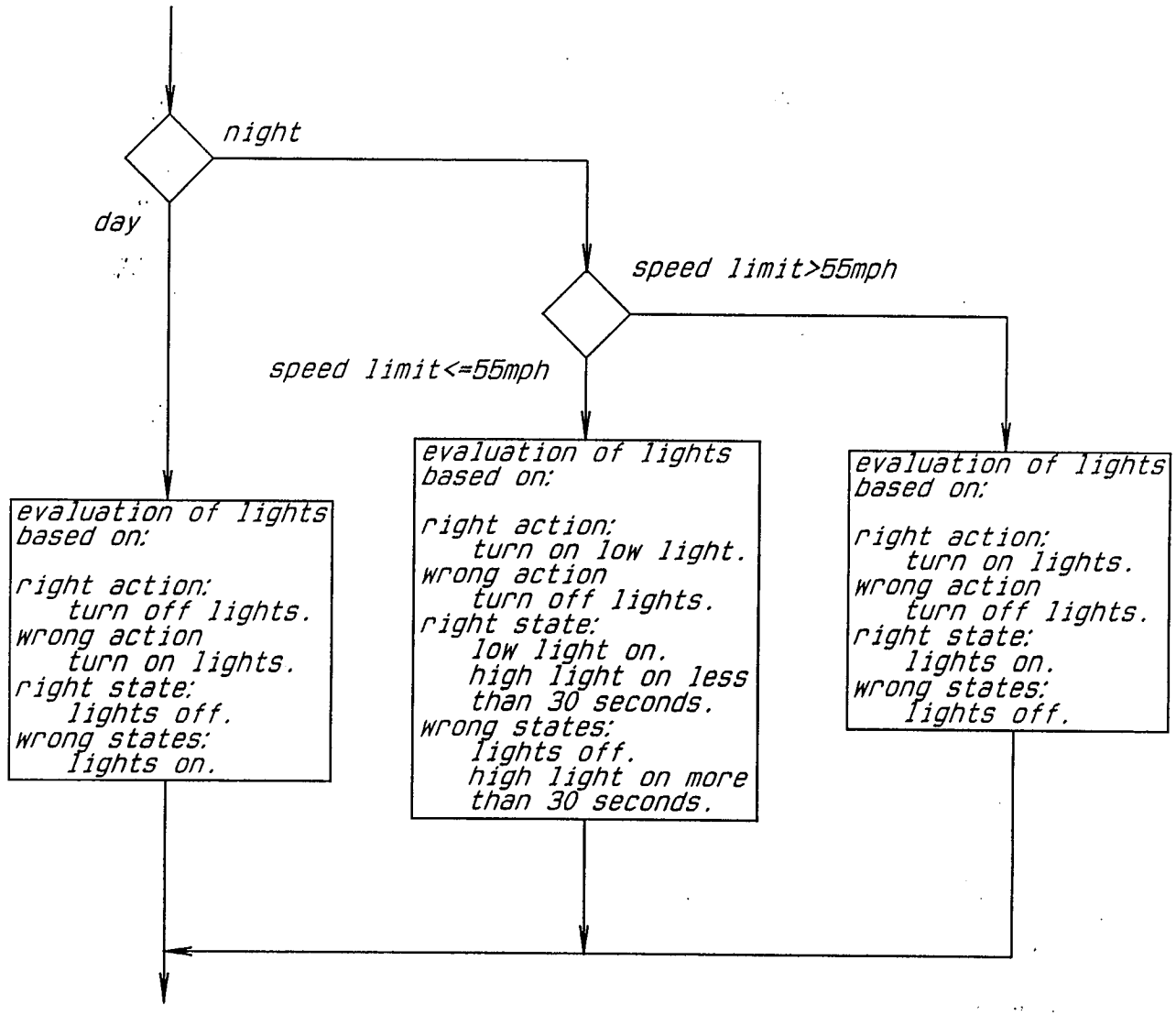

Figure 4.

Flowchart of Procedure night()

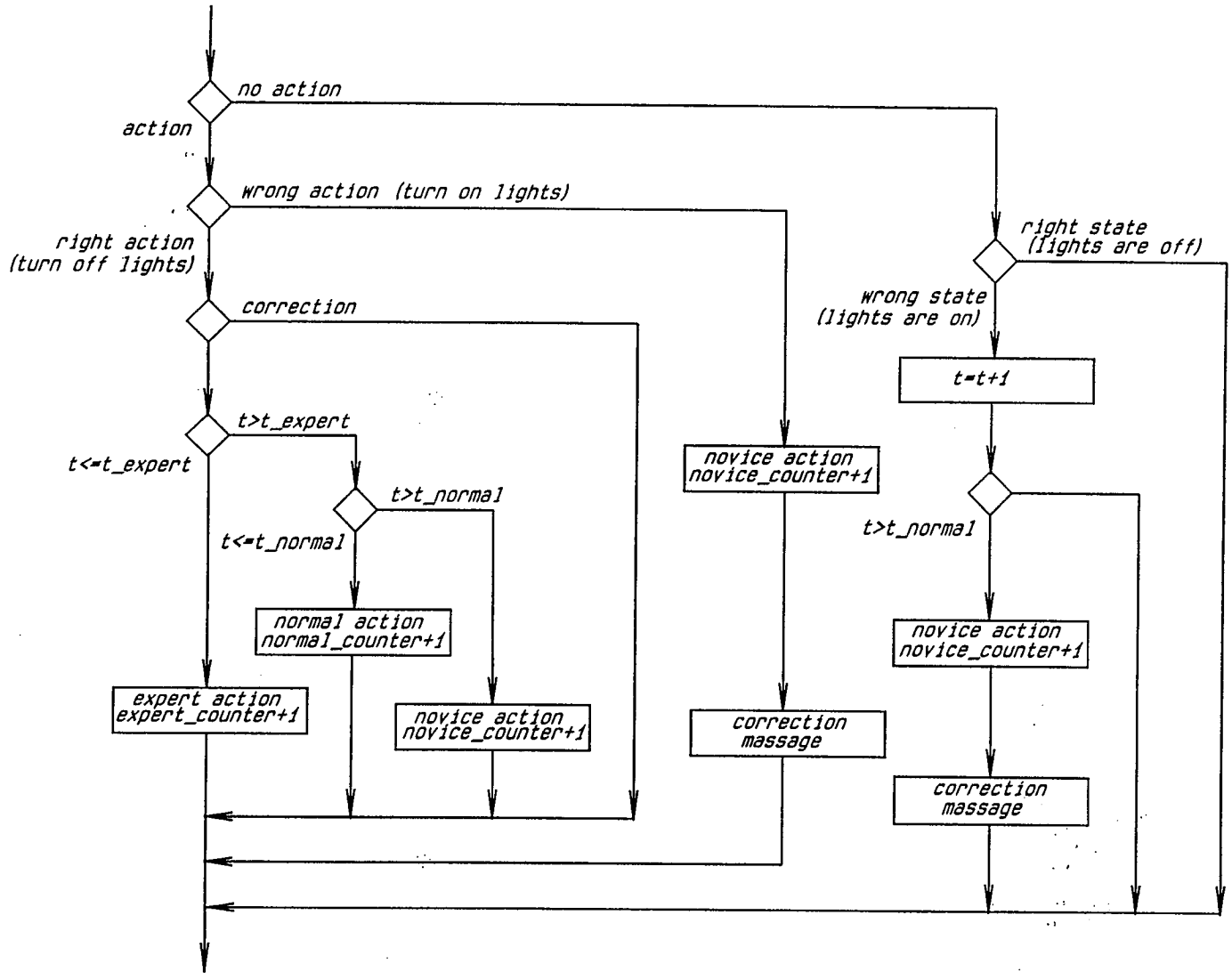

Figure 5.
The Evaluation Method in the Environment Category

considers the action at the novice level and records it in novice_counter. If the action is correct, the user model evaluates it further based on the response time of the action. If the response is quickly enough like expert users, the user model considers the action as a expert users' action and records it in the expert_counter. If the response time is in the range of normal users', the user model considers the action as a normal users' action and records it in the normal_counter. If the response time is longer than normal users', the user model considers the action as a novice users' action and records it in the novice_counter. A message may be sent out to tell a novice user about the improper status of the accessory. Procedure day() is an example.

The rule in the procedure day() is that the high and low light should be turned off during the day. If the lights are still on when the time comes from night to day, the time counter increases. When the lights are turned off, the response time in the time counter is compared with the experts' response time, normal users' response time and novice user's response time. The result tells at which level the user's action is. If a user does not turn off lights when the response time exceeds the normal user's response time, the user model rates the user at the novice level and sends user a message that lights are still on. If a user turns on a light in the day time, the user model

considers this action as a novice action and also sends user the above message. The rated user actions are recorded in the action counters corresponding to its levels.

Procedure night() works in the same way as Figure 5. It has the rules as follow,

The low light should be used and the high light should not be continuously used more than 30 seconds during the night at a speed zone less than and equal to 55 mile per hour (assuming not on the interstate road);

The low or high light should be used during the night at a speed zone larger than 55 mile per hour (assuming on the interstate road).

2. WEATHER

Procedure weather_clear() is called when the weather is clear. The procedure includes the rules,

The foggy light should be off in a clear day;

The wiper should be off in a clear day.

Procedure weather_foggy() is called when the weather is foggy. The procedure includes the rule that the foggy light should be on in a foggy weather.

Procedure weather_rain_snow() is called when it rains or snows. It includes several rules(Figure 6),

The wiper should be turned on at the delay or low when the car speed is at the range of larger than 15 and less than 25 mile per hour.

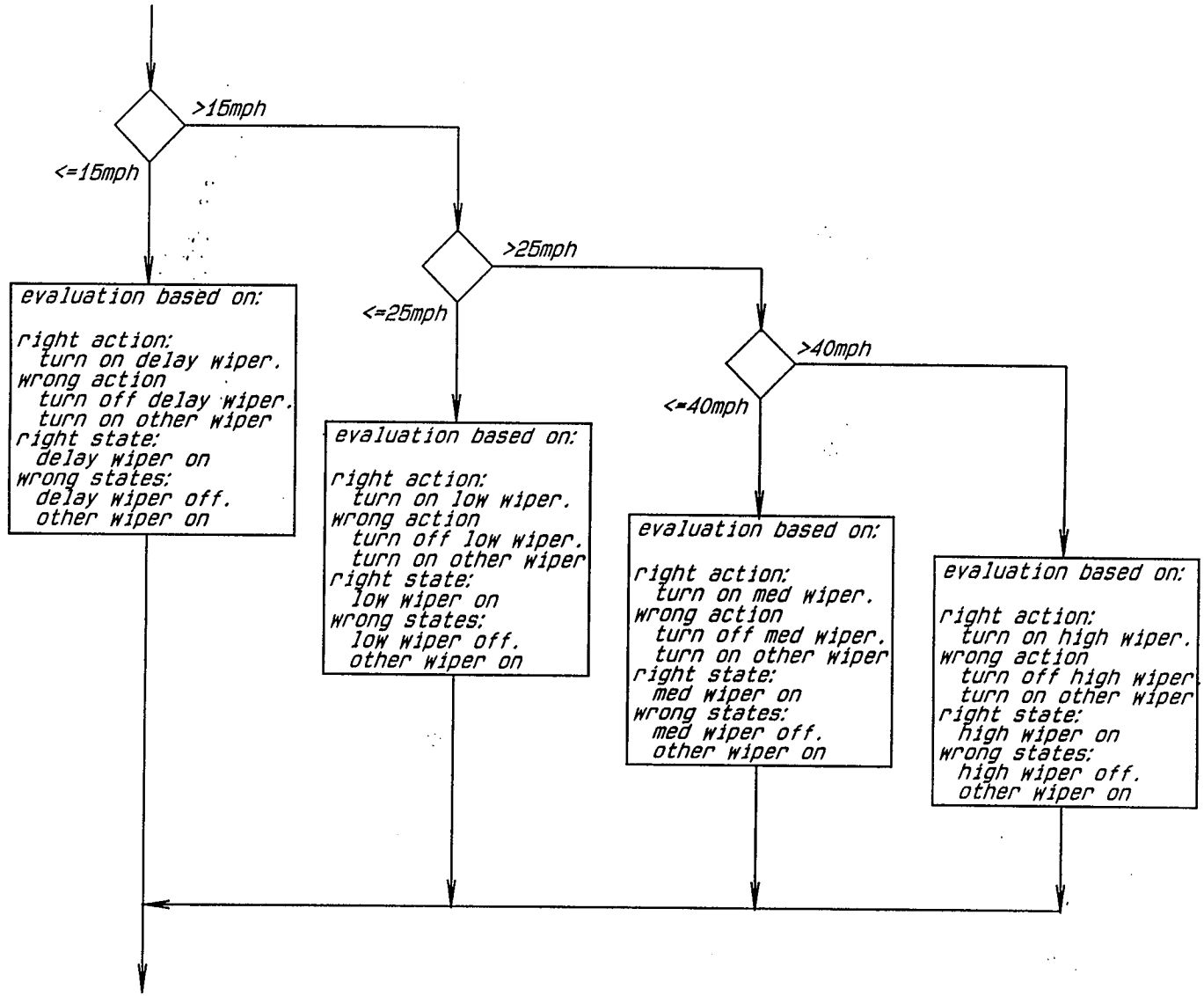

Figure 6.

Flowchart of Procedure weather_rain_snow()

The wiper should be at low or middle when the car speed is at the range of larger than 25 and less than 45 miles per hour.

The wiper should be at the middle or high when the car speed is larger then 45 mile per hour.

The rules are assumed that a medium rain or snow is coming down to the ground constantly.

The implementation of procedures weather_clear(), weather_foggy() and weather_rain_snow() works under the same principle as in Figure 5.

The Driving Category

In the driving category, the structure chart is shown in Figure 7. The user model reads the driving phase on the black board, then calls the correspondent evaluation procedures. The driving phase has the status of driving forward, driving backward, stopping car forward, stopping car backward and process emergency.

The typical model of an evaluation procedure for the driving category is shown in Figure 8. It normally needs a sequence of user's actions to operate a car to finish a task. For example, a driver has to push down off the clutch, change a gear, push down the accelerator pedal and put on the clutch in order to start moving the car. In this case, the user model stores the time of each action in its correspond register. After a task is finished, the user

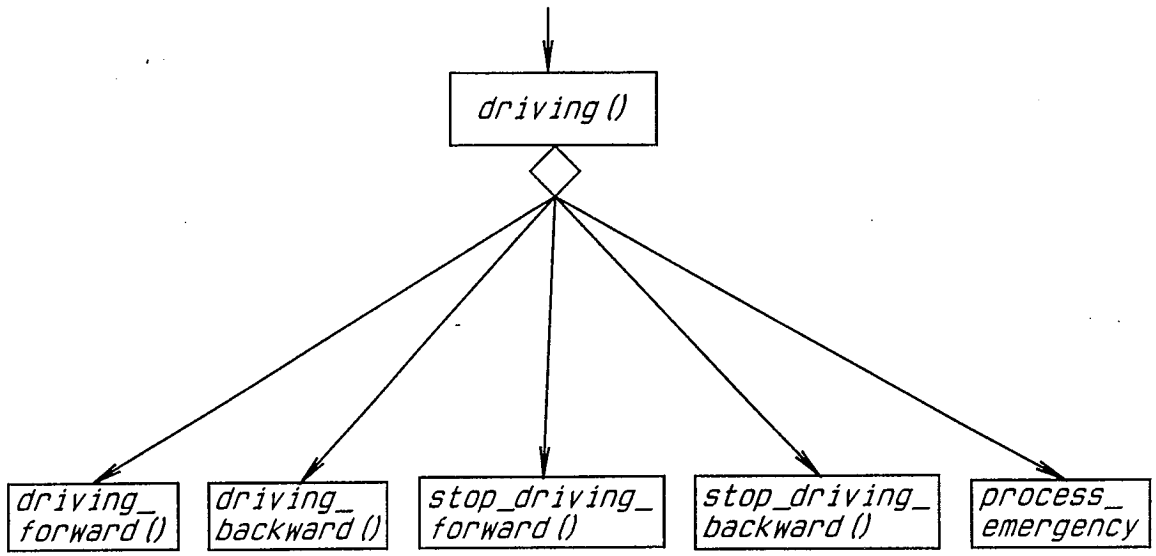

Figure 7.
Structure Chart of the Driving Category

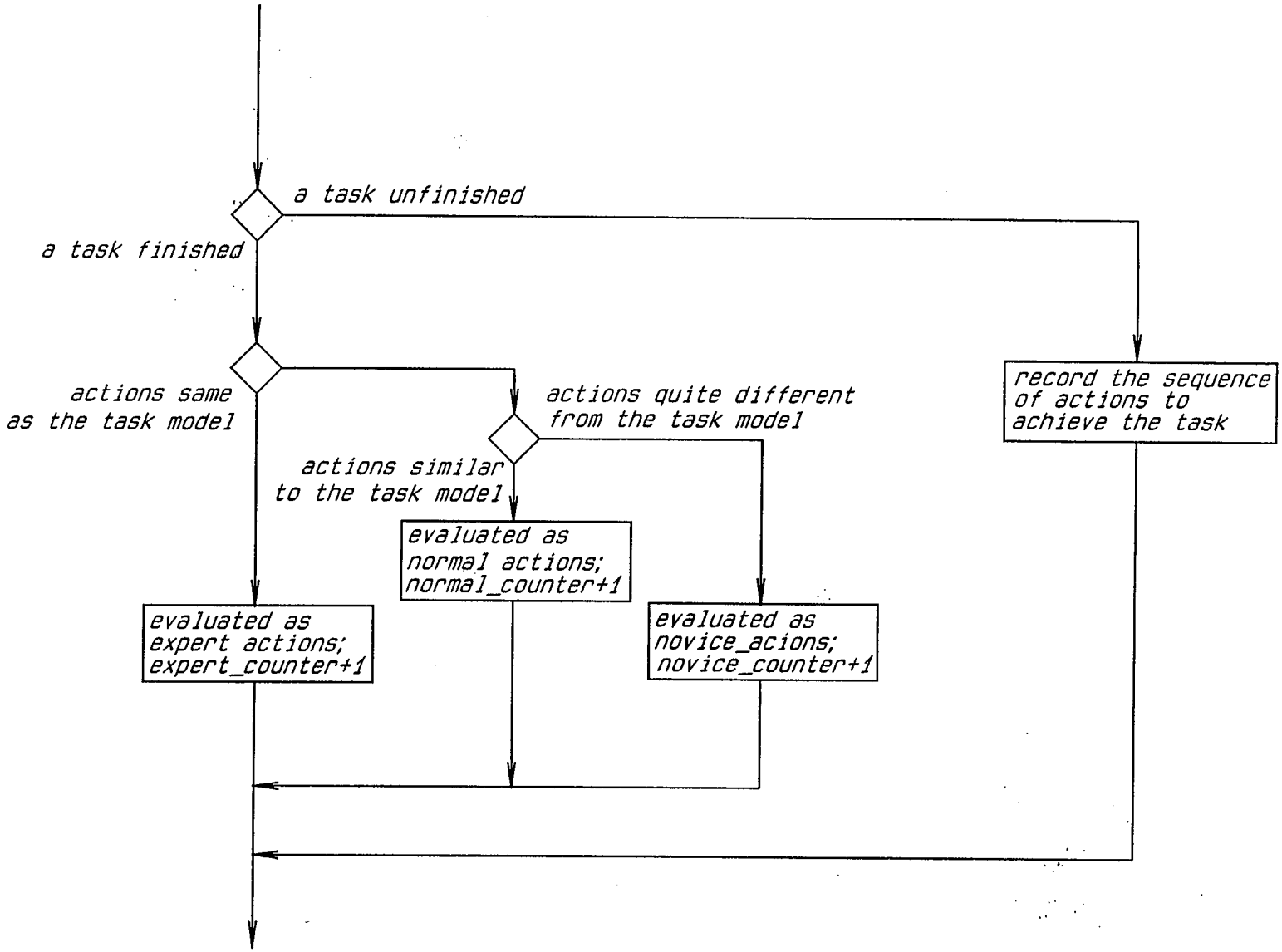

Figure 8.
The Evaluation Method in the Driving Category

model utilizes these times to find the sequence of the user's actions. Then the evaluator in the user model compares these sequence of the user's actions with the task model. If the user's actions are completely the same as the task model's, the actions are considered at the expert level. If the user's actions are a little different from the task model's but has no serious mistake, the user's actions are included into the normal user's actions. Otherwise, the user's actions are rated at the novice level. The total number of actions at each level are recorded by their corespondent counters.

1. DRIVING FORWARD

Figure 9 shows the structure chart of the driving forward phase. The start_engine() is called when the engine is not turned on. When the key is on and the car speed is zero, the procedure start_car() is activated. When the car starts moving, the procedure driving_forward() is called. Procedure turn_left() or turn_right is activated when the car is in turning left or turning right situation.

Procedure start_engine():

This procedure evaluates the user's performance before he starts the engine. The evaluation is done at the moment that the ignition key is at "on" continuously.

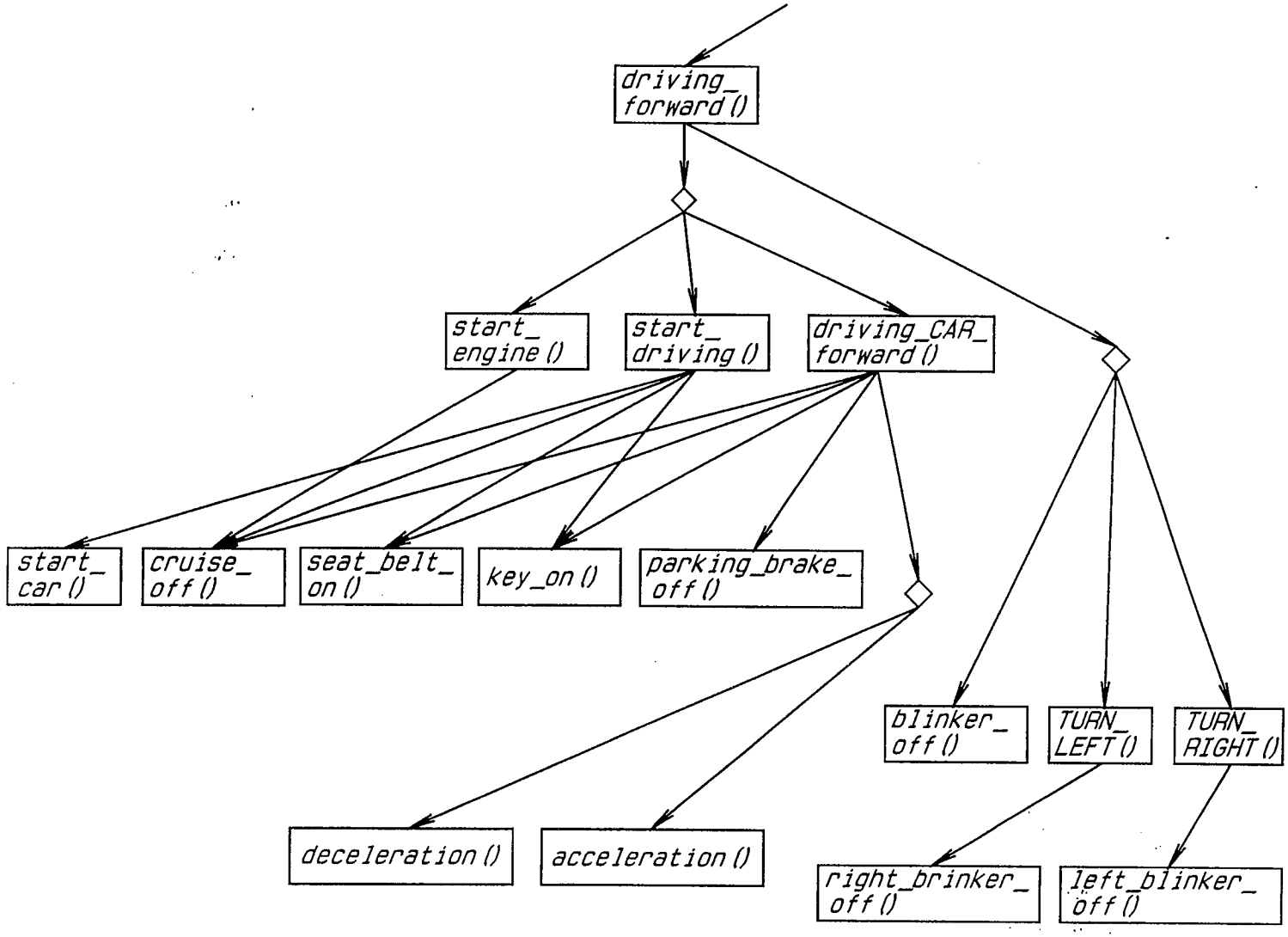

Figure 9.
Structure Chart of Procedure driving_forward()

The actions at the expert level obey the rules as follow:

The action sequence is,

a. button the seat belt;

b. push the clutch or change gear to n;

c. turn on the key.

The parking brake or the brake is on.

When the user finishes the above actions, but does not obey the action sequence. The user's actions are considered at the normal user level unless b and c are reversed.

If user actions do not obey the rules at expert level or normal user level, those actions are considered at the novice level.

Procedure start_car():

This procedure evaluates the user's operations on starting the car. The evaluation is done at the moment that the speed changes from zero to non-zero.

The rules to evaluate the user's actions at the expert level is described as follow.

a. The brake is on when the parking brake is set off.

b. The clutch is off when the gear is changed to 1.

c. The brake is off when the accelerator is pushed to on or the clutch is put on.

d. The accelerator is on when the clutch is put on.

The normal user's actions are assumed to meet rules b, c and d. Otherwise, the user's actions are considered at the novice level.

Procedure driving_car_forward():

This procedure is to evaluate a user's skill of changing gear at different driving speeds. When the speed is 5 miles per hour more than the speed limit, call the deceleration() to evaluate the user's operations on decreasing the car speed. When the speed is 5 miles per hour less than the speed limit, call the acceleration() to evaluate the user's operations on increasing the car speed.

Procedure acceleration():

The flowchart of procedure acceleration() is shown in Figure 10. A user's actions at the expert level obey the following rules,

Gear 1 is changed to gear 2 at the speed range from 8 to 10 mile per hour;

Gear 2 is changed to gear 3 at the speed range from 15 to 25 mile per hour;

Gear 3 is changed to gear 4 at the speed range from 30 to 40 mile per hour.

If the user changes the gear at a speed which exceeds the above speed range not more than 5 mile per hour, the user's actions are considered at the normal user level.

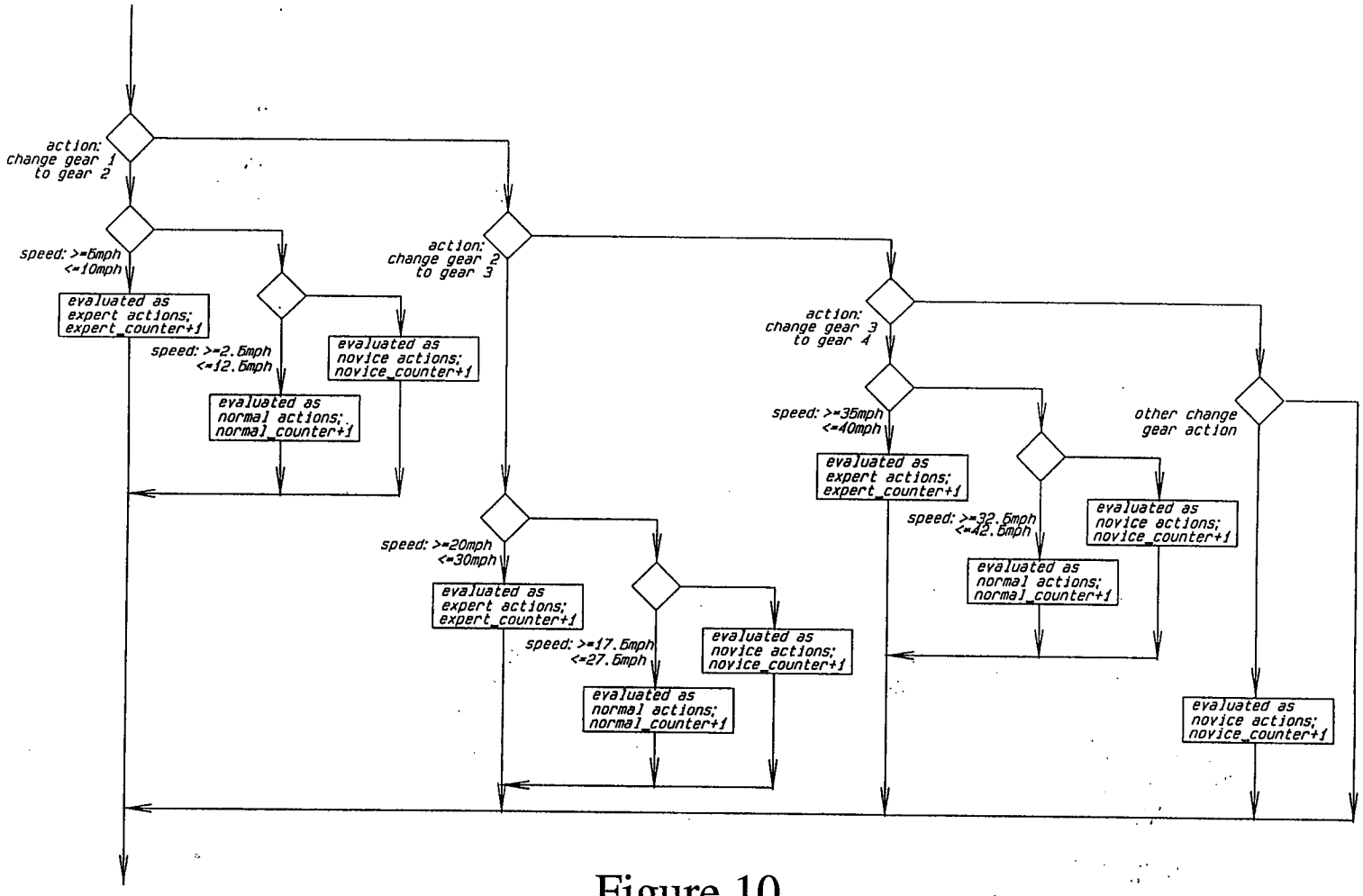

Figure 10.
Flowchart of Procedure acceleration()

If the changing gear speed exceeds the above speed range more than 5 mile per hour, the user's action is considered at the novice level.

Procedure declaration() works the same way as acceleration(). It contains the rules as shown in Figure 11.

Procedure turn_left():

The procedure turn_left() is called when the car turns to the left. It includes several rules as follow,

a. The left blinker should be turn on.

b. If the turning speed is in the range of plus or minus 5 miles per hour from the speed limit, the turning action is considered as an expert action; If the turning speed exceeds the expert speed range and is in the range of plus or minus 10 miles per hour from the speed limit, the turning action is considered as the normal user action; If the turning speed exceeds plus or minus 10 miles per hour from the speed limit, the turning action is considered as the novice action.

c. The blinker should be turned on about 100 feet before the turning location. If the blinker is turned on too earlier or later, the blinker operation is considered as a normal user action or a novice action.

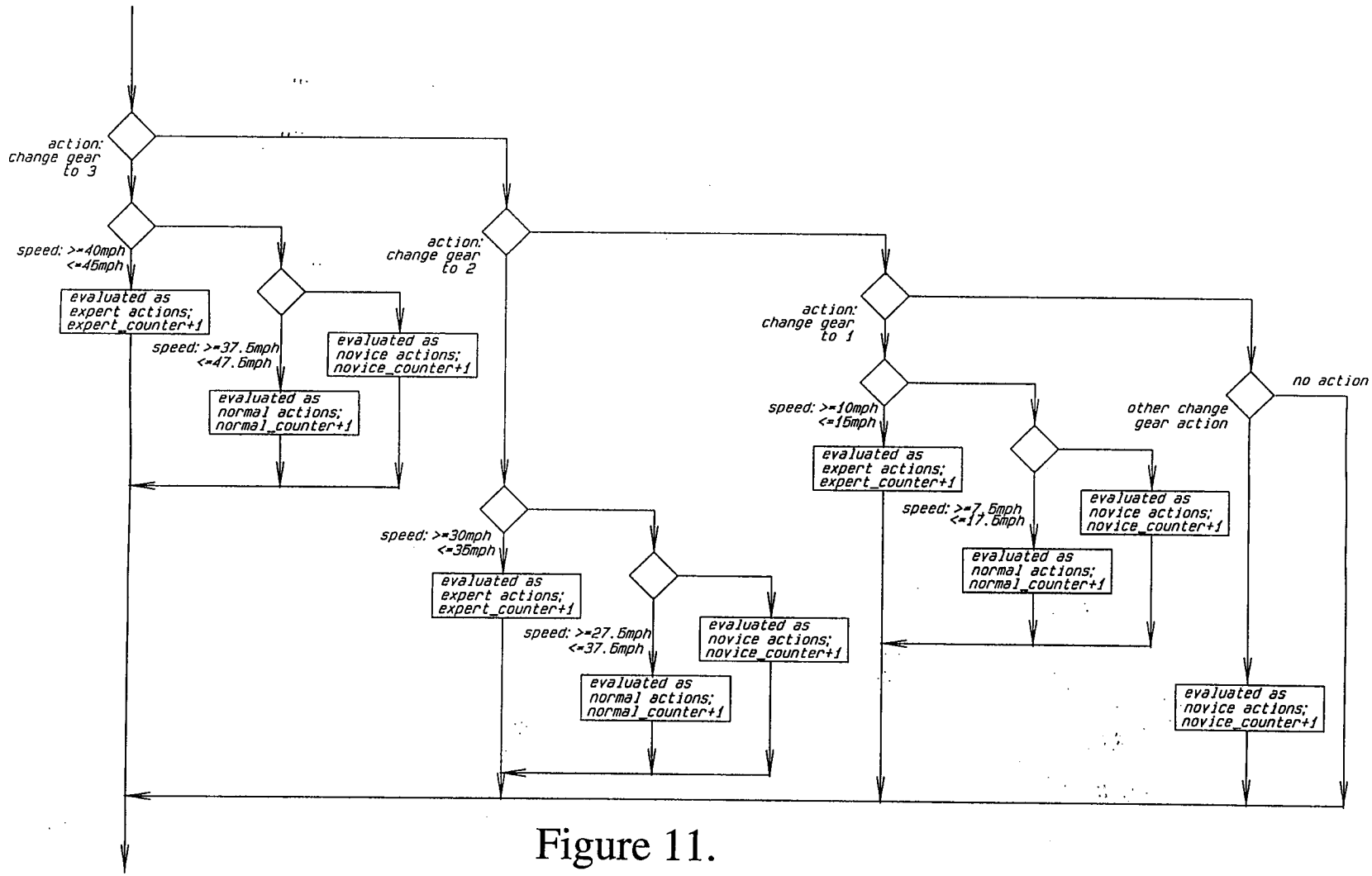

Figure 11.

Flowchart of Procedure deceleration()

d. The blinker should be turned off after the turning location.

The implementation flowchart is shown in figure 12.

Procedure turn_right() is called when the car turns to the right. The design principle is the same as the turning_left().

Other procedures used in the driving forward phase are key_on(), cruise_off(), seat_belt_on(), parking_brake_off(), blinker_off(), right_blinker_off(), and left_blinker_off(). These procedures implements the following rules, which works in the way shown Figure 5.

The cruise should be in "off" when the speed is less than 55 mile per hour(procedure cruise_off()).

The blinker should be at "off" when car goes straight(procedure blinker_off()).

The left blinker should be at "off" when car turns to right(procedure left_ blinker_off()).

The right blinker should be at "off" when car turns to left(procedure right_blinker_off()).

The user should not use parking brake while driving(parking_brake_off()).

The key should be at "on" (procedure key_on()).

The seat belt should be at "on"(procedure seat_belt_on().

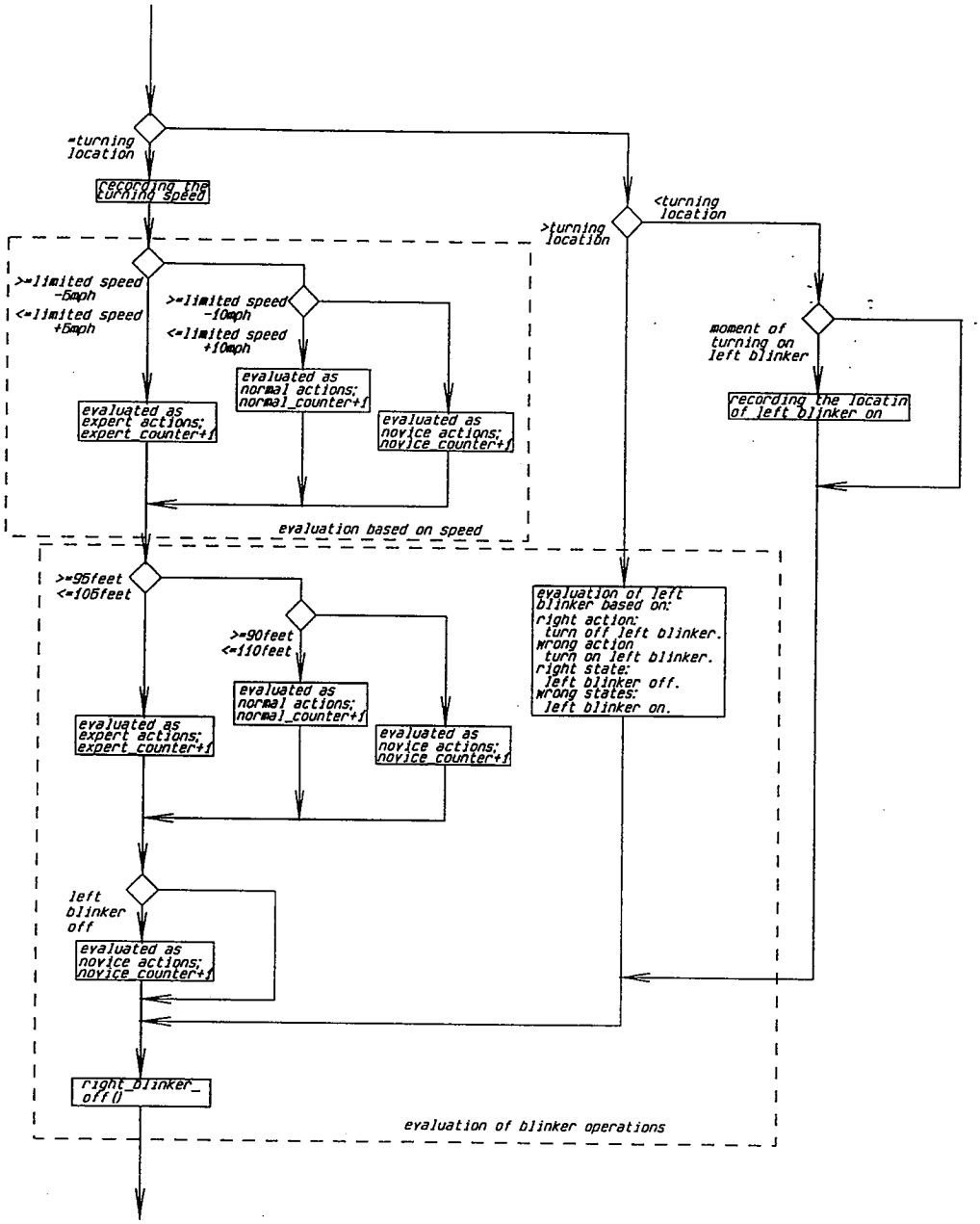

Figure 12.
Flowchart of Procedure turning_left()

2. DRIVING BACKWARD

Procedure driving_backward() is called when the test scenario tells the user to drive backward. The structure chart of this procedure is shown in Figure 13. Procedure deceleration() is activated if the forward speed is larger than 7 mile per hour. When the forward speed is reduced to less than 7 mile per hour, procedure stop_from_idle_speed is called. When the forward speed goes to zero, procedure backward_operation() is called.

Procedure backward_operation():

Since only gear r is used in the driving backward operation, it is simpler than the driving forward case. The correct operation should have an action sequence,

a. brake on;

b. clutch off;

c. gear to r;

d. brake off;

e. acceleration petal on;

f. clutch on.

If the user's actions have above sequence from a to f, the actions are rated as experts' actions.

If the user finishes above actions but only with a sequence action e before action f, the user's actions are evaluated at the normal user lever.

If the user's actions do not belong to expert level and

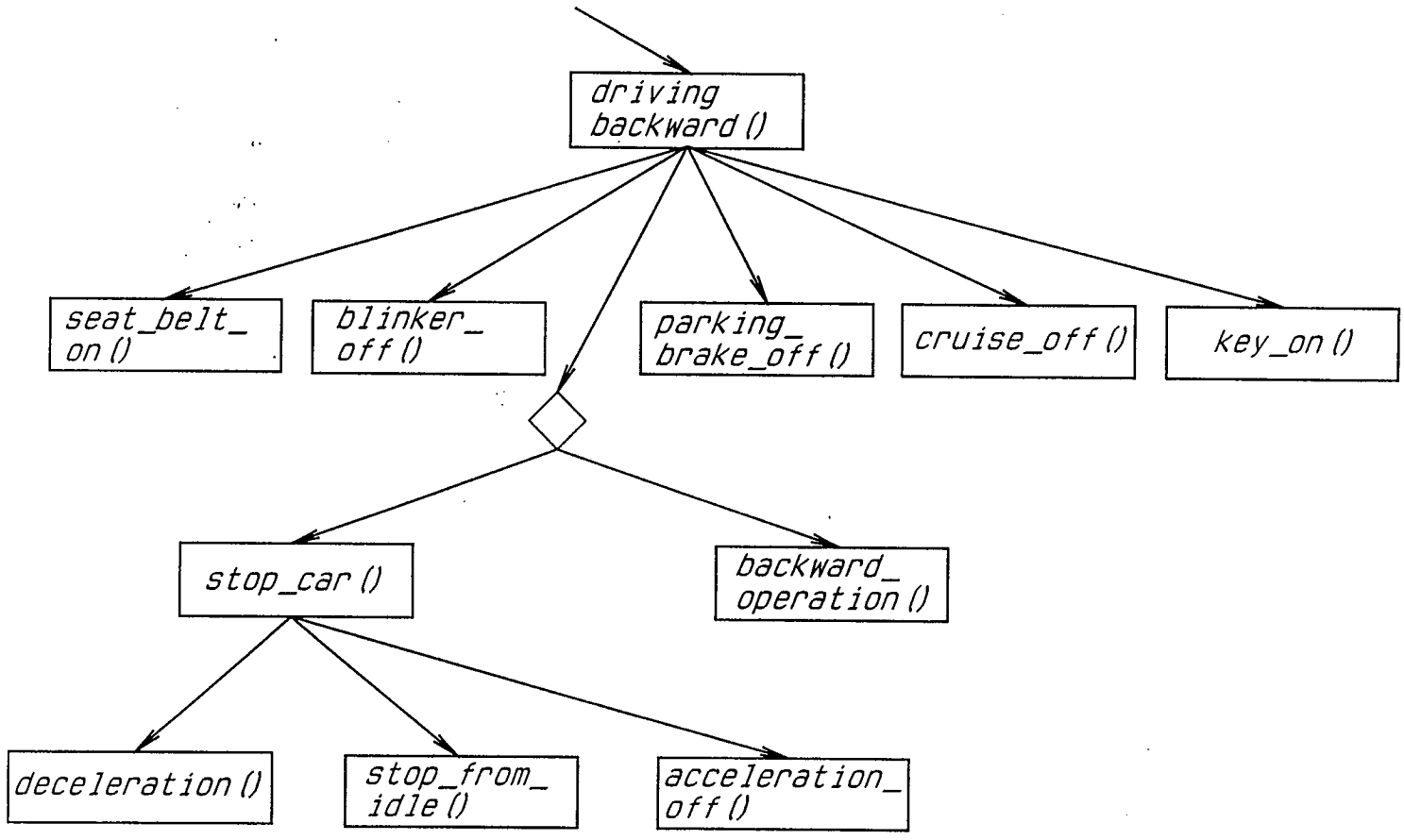

Figure 13.
Structure Chart of Procedure driving_backward()

normal level, the actions are rated at the novice level.

Procedure stop_from_idle():

This procedure evaluates the user's actions based on the following rules implemented in the same way as in Figure 5,

The clutch should not be off when the speed is larger than 3 miles per hour;

The clutch should be off when the speed is less than 3 miles per hour;

Other rules implemented in the driving backward phase are listed as follow,

The cruise should be in "off" status(procedure cruise_off()).

The acceleration pedal should be at "off" when the forward speed is reduced(procedure acceleration_off()).

The blinker should be at "off"(procedure blinker_off()).

The user should not use parking brake (parking_brake_off()).

The key should be at "on" (procedure key_on()).

The seat belt should be at "on"(procedure seat_belt_on().

3. STOP DRIVING FORWARD

When a user is told to stop his car, he should reduce the speed to zero. Figure 14 shows the structure chart of procedure stop_driving_forward(). When the speed has not been reduced to zero, procedure deceleration() and stop_from_idle() are called to evaluate the user's actions. When the speed equals to zero, procedure parking_action is called if parking is required. Procedure turn_left() or turn_right() is activated to evaluate the user's operations when the car turns left or right.

Procedure parking_action():

The procedure parking_action() evaluates the user performance of parking the car. The evaluation is done at the moment that the driver puts the clutch on and is based on the following rules,

The action sequence at the expert level is,

a. parking brake on;

b. brake off;

c. key off.

d. gear at 1 or r;

e. clutch on.

If a user finishes all above actions and meets the sequence of action c before action e, the user's actions are rated at the normal level.

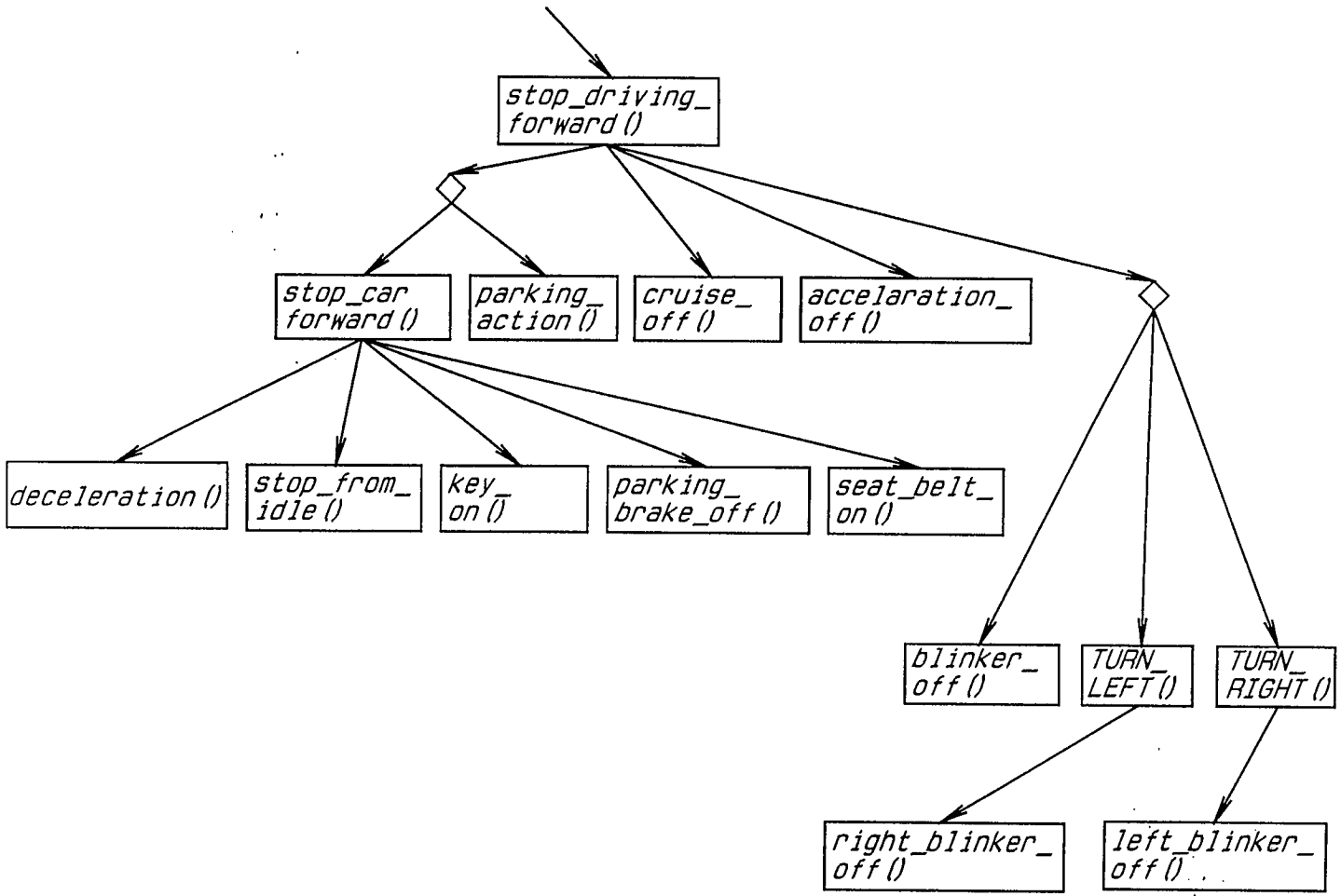

Figure 14.
Structure Chart of Procedure stop_driving_forward()

If a user's actions do not meets the rules for the expert level and normal level, the user's actions are rated to the novice level.

The stop_driving_forward() also implements the following rules by calling the correspondent procedures,

The cruise should be in "off" status(procedure cruise_off()).

The acceleration pedal should be at "off"(procedure acceleration_off()).

The blinker should be at "off" when car goes straight(procedure blinker_off()).

The left blinker should be at "off" when car turns to right(procedure left_ blinker_off()).

The right blinker should be at "off" when car turns to left(procedure right_blinker_off()).

The user should not use parking brake while driving(parking_brake_off()).

The key should be at "on" until the speed is down to zero(procedure key_on()).

The seat belt should be at "on" until the speed is down to zero(procedure seat_belt_on().

4. STOP DRIVING BACKWARD

Figure 15 shows the structure chart of procedure

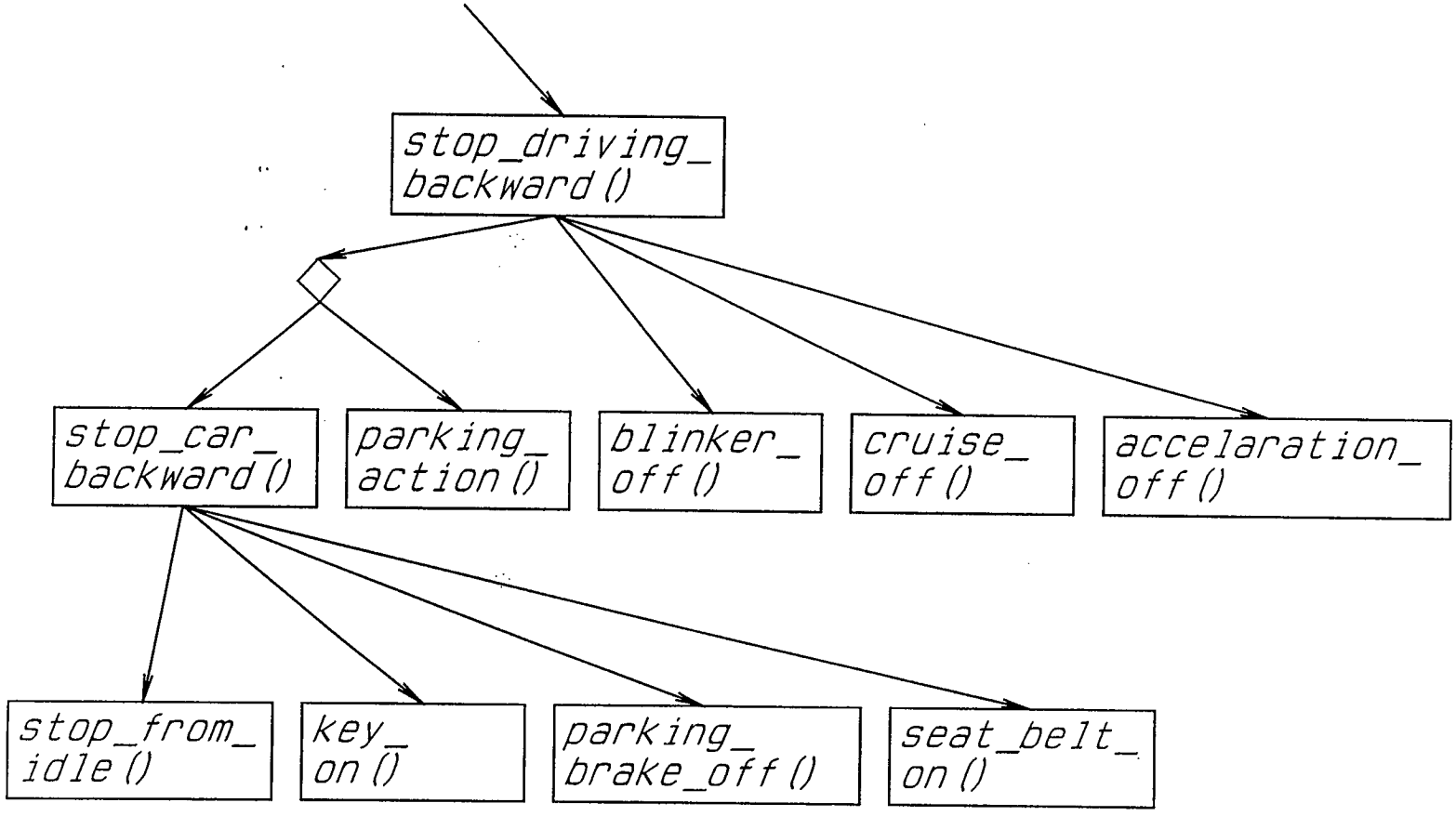

Figure 15.
Structure Chart of Procedure stop_driving_backward()

stop_driving_backward(). When car does not stop, procedure stop_from_idle() is called to evaluate the user's stopping car operations. When car stops, procedure parking_action() is called to evaluate the user's parking car operations if parking is required. The other rules in stop_driving_ backward(), implemented by their corespondent procedures(Figure 14), are listed as follow,

The cruise should be in "off" status(procedure cruise_off()).

The blinker should be at "off"(procedure blinker_off()).

The acceleration pedal should be at "off"(procedure acceleration_off()).

The user should not use parking brake while driving(parking_brake_off()).

The key should be at "on" until the speed is down to zero(procedure key_on()).

The seat belt should be at "on" until the speed is down to zero(procedure seat_belt_on().

5. PROCESS EMERGENCY

Procedure process emergency() works when the car is in a emergency situation, such as flat tire, loss of wheel, failure of steering, and car skidding. The procedure evaluates the user's reactions to the above situations. It

calls several basic procedures(Figure 16) to see if the user obeys the following rules,

The cruise should be at "off" status(procedure cruise_off()).

The emergency light should be at "on"(procedure emergency_light_on()).

The acceleration pedal should be at "off"(procedure acceleration_off()).

The user should not use brakes at the speed larger than 7 mile per hour and should use brakes at the speed less than 7 mile per hour(Figure 17). The implementation procedure is emergency_action(), which works in the same principle as the Figure 5.

The key should be still at "on" until the speed is down to zero(procedure key_on()).

The seat belt should be still at "on" until the speed is down to zero(procedure seat_belt_on().

When car stops, the correct parking actions are needed(procedure parking_action()).

THE RESULT OF EVALUATION

In the evaluation process, three counters are used to record the number of operations at different levels. The expert counter records the total number of the expert operations. The normal counter records the total number of

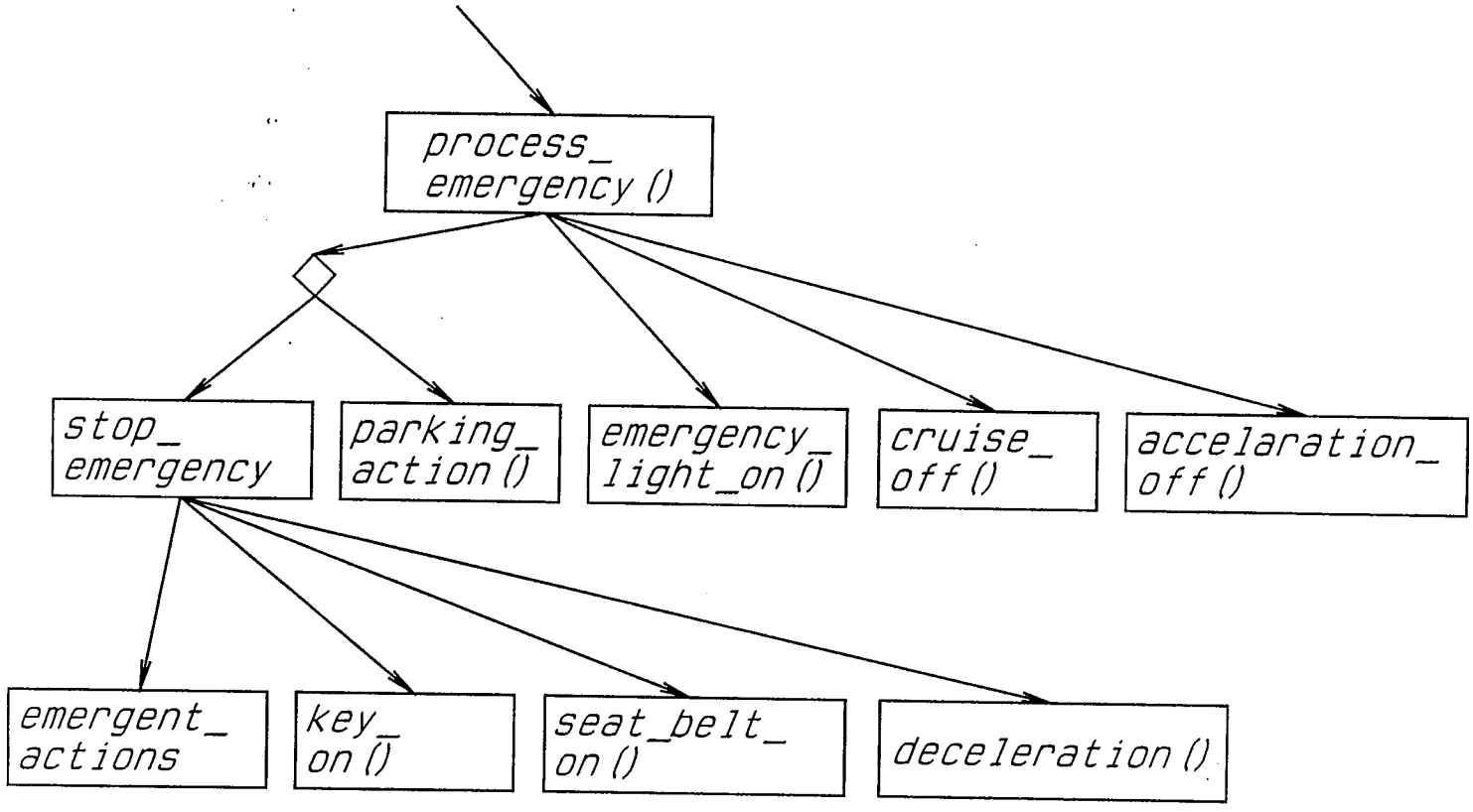

Figure 16.
Structure Chart of Procedure process_emergency()

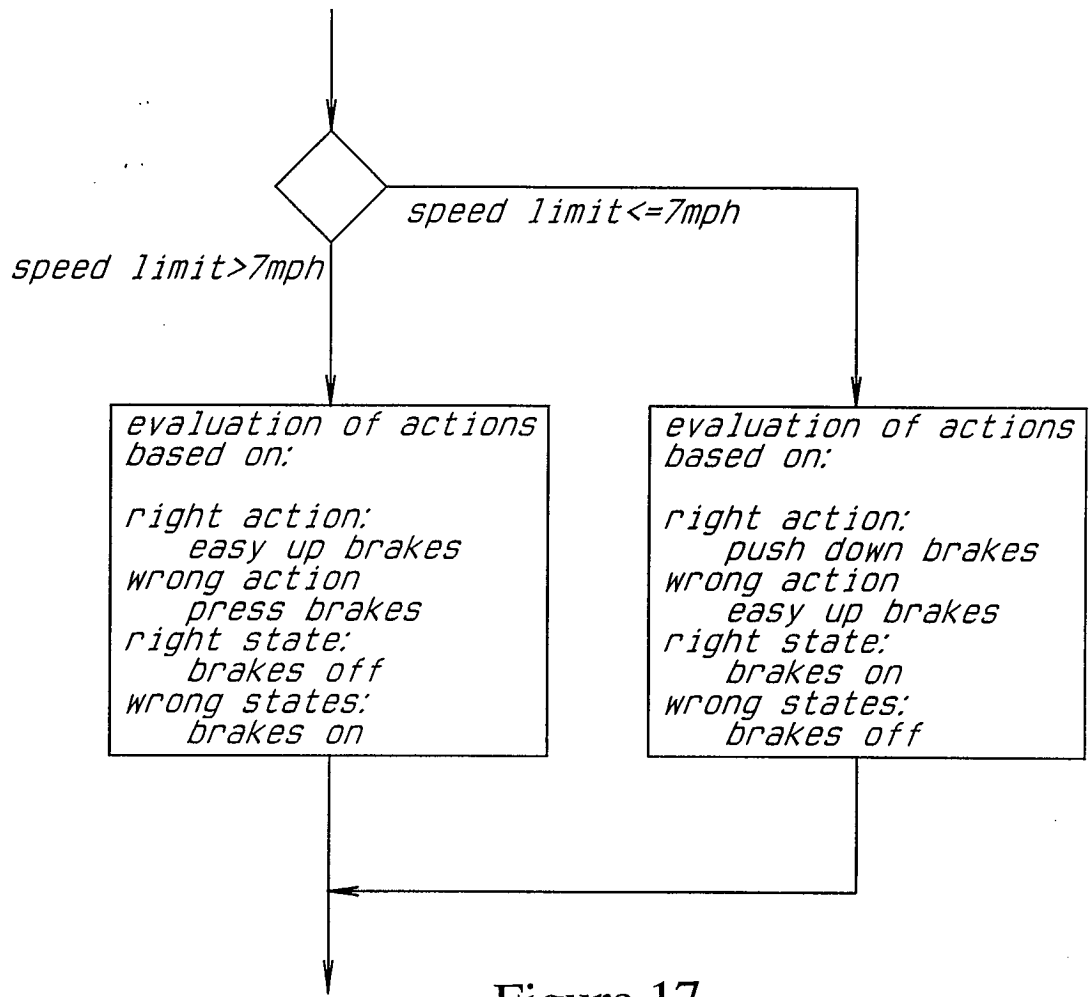

Figure 17.

The Flowchart of Procedure Emergency_action()

the normal operations. The novice counter records the total number of the novice operations. An expert level operation is the operation that is perfect for finishing a task. A normal level operation is the operation that is correct for driving but not optimal. The novice operation is the operation that is wrong or unskillful for driving. The user model evaluates the user's performance based on the qualities of user's majority actions. The numbers in the three counters are used as the raw scores, which are converted to the total score of a user. One count in the expert counter earns one honor point for the total score, one count in the normal counter does not earn a honor point, and one count in the novice counter losses one honor point. The conversion equation is,

The total score = 2 x number in the expert counter + 1 x number in the normal counter - 1 x number in the novice counter

The raw scores are also converted to the average score which is expressed by

The average score = The total score / the total number of operations

The total score tells the absolute score in each

evaluation, which represents the final result of the user model. The average score shows the user model based on the user's actions in real time. If the average score is larger than 1 in a particular time, it is supposed that the user is an expert user because the majority of the user's actions are expert level actions. If a user has an average score between 0 and 1, the majority of the user's actions are normal user level actions and the user is considered as a normal user. If a user has a negative average score, the majority of user's action are novice level actions and the user is considered as a novice user.

IV. RUNNING EXAMPLES

The user model communicates with the graphical user interface environment through parameters on the black board. These parameters, representing car status and driving environment, are listed in the following,

accel -- the car accelerator; status: from scale 0 to 10.

brake -- the car brake; status: off or on.

blinker -- the car blinker; status: off, left, right, emergency.

clock -- the clock recording the exact time of program running.

clutch -- the car clutch; status: off or on

cruise -- the car cruise control system; status: off or on.

distance -- the driving mileage.

gear -- the car gear; status: 1, 2, 3, 4, or r.

key -- the car key; status: off or on.

light.fog -- the car fog light; status: off or on.

light.high -- the car high light; status: off or on.

light.low -- the car low light; status: off or on.

phase -- the car driving phase; status: driving forward, driving backward, stop driving forward,

stop backward or emergency.

direction -- the car driving direction; status: straight, left turn or right turn.

parking_brake -- the car parking brake (the emergency brake); status: off or on.

seat_belt -- the car safety belt; status: off or on.

speed -- the car current driving speed.

speed_limit -- the speed limit in a speed zone.

time -- the current time; status: day or night.

wiper -- the car wiper; status: delay, low, medium, or high.

weather -- the weather; status: clear, foggy, rain or snow.

msg -- the message to a user.

THE RUNNING BACKGROUND OF THE USER MODEL

In order to run the user model in a GUI environment, a testing scenario must be used. The scenario is designed to combine different events listed in Appendix A. Each event in the scenario sends two messages to the black board. One message is read by the GUI and the other by the user model. The procedures in the user model are activated according to the events in the scenario. Here is an example of using the testing scenario.

When a user drives from a private drive to a street, the message read by GUI:

"Yield sign 300 feet away";

the message read by the user model:

driving phase = stop_forward();

When a user sees a massage from GUI that "Yield sign 300 feet away", the user should slow speed. The user model evaluates the user's operation of reducing speed using procedure stop_forward().

When the user drives close to the Yield sign,

massage read by GUI:

"A car is passing.";

massage read by the user model:

driving phase = stop_forward();

The user model evaluates the user's operation on a complete stop.

When there is no car ahead,

massage read by GUI:

"no car is passing";

massage read by the user model:

driving phase = forward();

The user model evaluates the user's operation on starting driving.

When the user drives on the street

massage read by GUI:

"you are driving on urban street";

massage read by the user model:

driving phase = forward();

The user model evaluates the user's operation on acceleration to the speed limit.

THE RUNNING EXAMPLES

The following examples show that all the rules in the user model are implemented in procedures. The user model program runs repeatedly at a fixed time interval. When a procedure in the user model is activated by the test scenario, the procedure reads user's actions and car status from the black board to evaluate user's actions. In the running test, each running cycle is recorded. A running cycle is one iteration of the user model program. The program running time is represented by running cycles. The time interval between two of any user's actions should be larger than or equal to one running cycle. A sample output file is shown in Appendix C. It is assumed that one running cycle is needed by experts to finish a task and two running cycles by normal users in some testing events,

expert_time = 1 running cycle;

normal_time = 2 running cycle.

The running test shows how the user model evaluates the user performance under various driving situations. The change of driving phase and the driving environment on the blackboard depends on the design of the test scenario. The change of the car status are caused by the user operations. Procedures in the user model are activated, corresponding to the driving phase and the driving environment, to evaluate the user's performance based on the car status,

Example 1:

Evaluation of Driving Forward

The driving phase is driving forward from cycle 0 to 33. Procedure start_engine() is activated from cycle 1 to 3. Under the initial condition, the user's action sequence is: a. fastening the seat belt; b. pushing the clutch off; c. turning the ignition key on. These actions are rated at the normal user level. Procedure start car() is called from cycle 3 to 7. The user's actions are rated as normal user's actions. At the cycle 7, the left blinker is turned on. This action is evaluated by procedure blinker_off() as a novice user's action. At the same time, a warning massage is sent to the user. From cycle 7 to cycle 17, procedure acceleration() is activated. The user changes gears at different speeds. The evaluation result tells that these actions belong to the expert's actions.

The seat belt is turned off at cycle 18 and the parking brake is turned on at cycle 19. These two actions are rated

at the novice level by procedures seat_belt_on() and parking_brake_off().

From cycle 30 to 41, the user is informed to turn the car to the left at the turning location of 10 miles. The turning speed limit is 25 mile per hour. The user turns the right blinker on at cycle 36 instead of the left blinker. This action is rated as a novice action by procedure right_blinker_off(). The user corrects this action by turning the left blinker on at cycle 38, at which time the driving distance is 9 mile. Since the left blinker is turned on too early, this action is also rated as a novice action at cycle 39. The user reduces the speed by changing gears to pass the turning location. The shifting gear to 3 is rated as an experts' action at cycle 36 and the shifting gear to 2 is rated as a novice user's action at cycle 39 by procedure deceleration(). The user's turning speed is 25 miles per hour which is rated as an experts' action by procedure turn_left(). The blinker is turned off at cycle 40. This action is rated as a normal user's action because the response time is not quick enough.

Evaluation of Driving with Change of Environment

The time is night from cycle 21 to 27. The user turns on the low light at cycle 21 as soon as the night comes. This action is rated by procedure night() as an experts' action at cycle 22. The fog light is turned on at cycle 22. This wrong action is a novice's action rated by

weather_clear(). The high light is turned on for a short period of time from cycle 22 to cycle 24. The user model considers that the user knows how to use high light correctly on the local road. The user model does not rate this actions. The high light is turned on from cycle 25 at the speed limit 60 mile per hour, which is assumed on the interstate road. Since the low light is on, this action is not rated for a score. All lights are turned off as the time changes to day at cycle 28. These actions are rated by procedure day() as experts' actions.

The weather is foggy from cycle 27 to 28. The fog light is turned on as soon as the weather changes to foggy. This action is evaluated as an experts' action by procedure weather_foggy(). The weather is raining from cycle 29 to 32. The user turns on the wiper to the low at cycle 29, then adjust the wiper to the high at cycle 32 to correspond to the current speed 50 mile per hour. The first action is rated as a normal user's action and the second action is rated as experts' action by procedure weather_rain_snow(). The user model allows the user to adjust the wiper to the right position in a short period time. The wiper is turning off as soon as the weather changes to clear at cycle 33. This action is evaluated as an experts' action by procedure weather_clear().

Evaluation of Stopping Drive Forward

Beginning from cycle 42 to the end, the user is

informed to stop the car. The user releases accelerator, pushes brake on and changes the gear to 1. These actions are rated as experts' actions at cycle 43 by procedure deceleration(). The user turns on the cruise at cycle 44, which is rated as a novice action by procedure cruise_off(). The user pushes the clutch off when the speed reduces to 1 mile per hour. This action is rated as an experts' action at cycle 48 by procedure stop_from_idle(). When the speed goes to zero, the user puts the parking brake on and releases the normal brake. These actions are rated as experts' actions at cycle 48 by procedure parking_action(). The user then turns off the key and finally puts the clutch on. Procedure parking_action() rates these actions as experts' action at cycle 52.

The total score obtained in this example is 22 and the average score 0.79. The average score shows the user is a normal user.

Example 2:

Evaluation of Processing Emergency

In this example, cycle 0 to 17 are the same as the above example. Beginning from cycle 18 to 29, the driving phase is emergency. The user turns on the emergency light immediately after he is told that his car is in emergency, such as loss of a wheel. This action is rated as an experts' action by procedure emergency_light_on(). At cycle

20, the user tries to use the brake to stop his car. This action is rated as a novice action by procedure emergent_action(). At cycle 22, the user pushes the accelerator in hurry, this action is also rated as a novice action by acceleration_off(). After the user reads the user model's warning messages, he corrects his wrong action. The user reduces the speed by changing gears. Because the user shifts the gears at the correct speeds, actions of changing gears are rated as experts' actions at cycles 24 and 27 by procedure deceleration(). When the speed slows down to 3 mile per hour at cycle 28, the user uses the brake. This action is rated as an experts' action by procedure emergent_action(). The user turns off the emergency light at cycle 29. The user model evaluates this action as a novice action by procedure emergency_light_on().

Evaluation of Driving Backward

The user is informed to drive backward at cycle 22. The forward speed at cycle 22 is already zero. The user changes the gear to "r" and starts to drive backward. These actions are rated as normal user's actions at cycle 33 by procedure driving backward().

Evaluation of Stopping Drive Backward

When the backward speed is 4 mile per hour at cycle 36, the user is told to stop the car. The user releases the accelerator to zero, pushes on the brake and pushes off the clutch. These actions are rated as expert's actions at

cycle 38 by procedure stop_from_idle(). When the speed goes to zero, the user parks the car. The parking actions are evaluated as expert's actions by procedure parking_action() at cycle 39 and 41, which is the same as in example 1.

The total score in this running expert is 19 and the average score is 1. The average score shows that the user is a normal user but has better performance than the user in example 1.

The total score in this running example is 19 and the average score is 1. The average score shows that the user is a normal user but has better performance than the user in example 1.

V. CONCLUSIONS

This user model is supposed to be used under a graphical user interface environment. In the design of the user model, the following facts are considered to evaluate a user's performance,

- the mental objects
- the properties of these objects
- the mental operations
- the conditions for using the operations
- the effects of the operations.

The mental objects are represented by blinkers, lights, brake, gear, etc. These objects have certain properties. For example, brake is used for reducing speed. The mental operations includes both actual operations which the user takes to finish a task and "thought operations" in which the user decides what operations to take. In this GUI environment, he has to decide whether to use the delay, low, median or high wiper before the user turns on the wiper in rain. The conditions for using the operation are linked to the system running environment. For instance, the user

should turn on light during the night and turn off the light in the day time. The effects of the operations is related to the user's knowledge of operating an object. That means, a user should know that the effect of using brake would not stop the car immediately but would loss control of the car when a car is in emergency such as loss of a wheel.

The implementation of evaluation procedures is based on the characteristics of driving a car. The accuracy and the response time of user's operations are the criteria in evaluation. During the evaluation process, a group of actions to finish a task is evaluated as the expert actions, the normal user actions or the novice actions. The expert level actions finish a task in high quality; the normal user actions achieve the goal of a task correctly; the novice actions are the unskillful or wrong operations. The user model is calculated based on evaluations completed.

The structure of this user model is a network approach which is a collection of procedures and tasks and contains hierarchy and sequence information. The advantages of this design are:

1. It is quite intuitive and self explanatory for the procedure/task sequences and hierarchy.

2. It is general enough to accommodate a wide variety of situations that will be necessary in evaluating user's performance in complex human-machine systems.

3. The network paradigm encourages top-down modeling

and allows use of existing libraries of models and procedures.

4. It may be used at different levels of evaluation from high-level mission performance to low-level button-pushing tasks.

5. Procedures require less investment of analyst time to obtain useful results; moreover, and not insignificantly, this network approach can easily be comprehended by higher management.

The disadvantages of this approach exists in:

1. Because of it generality, highly interacting procedures lead to levels of complexity that make checkout and validation very difficult. The results depends on the quality of the supporting data.

2. The identification and development of subprocedures and task models in procedures are often not unique. This may lead to an inadequate model if important tasks or events are not evaluated.

3. Libraries of commonly used procedures can be included in the network. This can be an advantage if the libraries contain assumptions and procedures that are appropriate but a disadvantage if they do not.

4. The evaluation in procedures considers some operations among tasks, which may not be the user possible operations.

Given the current state of the user model design, is

the methodology ready to be an integral part of the system design process? Although the methodology has a number of admitted weakness, it also has the ability to make a number of unique contributions to the process of system engineering.

By using the user model early in the designing process, a formal means is provided for considering the impact of user performance on the range of design issues that must be confronted while there is still time to resolve them. An early modeling effort can provide quantitative and qualitative analysis that allow design trade-off studies to include a verity of human performance factors along with other system variables. This process forces consideration of the assumptions and design decisions which underlie assertions that the system will work with available personnel.

In all, there are compelling reasons to believe that user modeling efforts should be regularly advocated and used as a regular part of the design process for large-scale human-machine systems.

VI. REFERENCES

Bannon, L. and Claire O'Malley, "Problems in Evaluation of Human-computer Interfaces", 1985, Human-Computer Interaction-Interact'84, North-Holland, Amsterdam, pp.709-713.

Bosser, T., "The Evaluation of Learning Requirements of IT System", Cognitive Engineering in the Design of Human-Computer Interaction and Expert System, edited by G. Salvendy, Elesevier, Oxford, pp. 45-52.

Edmonds, E.A., "Adaptive Man-Computer Interfaces", 1981, Computing Skills and the User Interface, Academic Press, London, pp.395-397.

Hammond, N. *et al*, "Evaluation The Interface of A document Processor: A comparison of Expert Judgement And User Observation", 1985, Human-Computer Interaction-Interact'84, North-Holland, Amsterdam, pp.725-729.

Janke, G and K. Kohnert, "Interface Design of A Visual Programming Language: Evaluating Runable Specifications", 1989, Man-Computer Interaction Research, North-Holland, Amsterdam, pp.567-581.

Kirakowski, J. and M. Corbett, Effective Methodology for the Study of HCI, 1990, North-Holland, Amsterdam, pp.167-198.

Meister, D., "Behavioral Test and Evaluation of Expert System", 1987, Cognitive Engineering in the Design of Human-Computer Interaction and Expert System, edited by G. Salvendy, Elesevier, Oxford, pp. 537-550.

Monk, A., "Empirical Evaluation of User Interfaces", 1987, Human-Computer Interaction, Morgan Kaufmann Publishers, Inc., pp.135-143.

Paetau, M., "The cognitive Regulation of Human Action as a Guideline for Evaluating the Man-Computer Dialogue", 1985, Human-Computer Interaction-Interact'84, North-Holland, Amsterdam, pp.731-741.

Ravden, S. and G. Johnson, Evaluation Usability of Human-Computer Interfaces, 1988, Ellis Horwood Limited, Chichester

Savage, R.E. and J.K. Habinek, "A Multilevel Menu-Driven User Interface: Design and Evaluation Through Simulation.", 1984, Human Factors In Computer System, Ablex Publishing Corporation, New Jersey, pp.165.

Sewell, D.R., N.D. Geddes, and W.B. Rouse, "Initial Evaluation of An Intelligent Interface for Operators of Complex Systems", 1987, Cognitive Engineering in the Design of Human-Computer Interaction and Expert System, edited by G. Salvendy, Elesevier, Oxford, pp.551-558.

Tullis, S. T.,"A System for Evaluating Screen Formats: Research and Application", 1984, Advances in Human-computer interaction, Volume 2, Ablex Publishing Co., New Jersey, pp.215-286.

APPENDICES

A. THE EVENTS USED IN SCENARIOS DESIGN

B. SOURCE CODE OF THE USER MODEL

C. INPUT AND OUTPUT FILES OF RUNNING EXAMPLES

APPENDIX A.

THE EVENTS USED IN THE SCENARIO DESIGN

The events used in the scenario design are to test a user's knowledge of the traffic regulation and experience of driving a car under different conditions. The following are examples of the testing events which are based on Alabama Traffic Regulation 1993,

1. Speed limit zone: When a user drives a car into a speed limit zone, the user should drive under the speed limit. The speed limits are 30 mile per hour in an urban area, 55 mile per hour in other location and 65 mile per hour where post.

2. Construction warning: When a user is informed a road construction area 1500 feet ahead, the user should drive under the speed limit of 30 mile per hour to pass the construction area.

3. Sharp turn: A user should drive under the speed limit 25 to 15 mile per hour at a sharp turn.

5. Informational or Guidance: The Informational or guidance board tells the user where is gas station, so that user can plan to use gas to avoid running out of gas at the middle of road. It also tells the user where to change his path to reach his travel destination.

7. School, hospital zone: car should slow down to the speed limit.

8. Complete stop: A user should completely stop his car at

a. red light

b. stop sign

c. school bus stop

d. railroad having red light

e. emergency car passing

.....

9. Yield: A user should slow speed to under 7 mile per hour when approaches the yield sign. If no traffic, the user can get through. Otherwise, the user should make a complete stop. The yield sign are

a. Yield signal

b. Continuous yellow flash light

c. from private drive to public street

.....

10. Blowout of tire: When blowout of tire happens, a user should turn on emergency light, not push brake, easy the accelerator to let the speed go down slowly, brake at speed of 7 mile per hour.

11. Loss of wheel: When loss of wheel happens, a user should turn on emergency light, not push brake, easy up on the accelerator to let the speed go down slowly, brake at speed of 7 mile per hour.

12. Steering failure: When steering fails, a user should turn on emergency light, not push brake, easy the

accelerator to let the speed go down slowly, brake at speed of 7 mile per hour.

13. Overheat: When overheat happens, a user should stop driving for a while.

14. Environment condition: A user should use wipers and lights to deal with environment conditions, such as clear, raining, snowing, foggy, night.

15. Terrain road: The user should adjust the gears and accelerator to the desired speed.

16. Car skidding: User should turn on emergency light, not push brake, easy the accelerator to let the speed goes down slowly, brake at speed of 7 mile per hour.

APPENDIX B.

SOURCE CODE OF THE USER MODEL

```
#include <stdio.h>
#include <string.h>

FILE *fpr, *fpw;
/*****************************************************************************
*  Parameters  defined  in  program.
*****************************************************************************/

#define        n            0    /* gear at neutrol */
#define        on           1    /* button status */
#define        off          2    /* gear at neutrol */
#define        left         3    /* blinker left light on */
#define        right        4    /* blinker right light on */
#define        r            5    /* gear at rear */
#define        emergen      6    /* both left and right blinkers on*/

#define        clear        11   /* driving environment */
#define        rain         12
#define        foggy        13
#define        snow         14
#define        day          15
#define        night        16

#define        forward      21   /* driving forward process evaluation*/
#define        backward     22   /* driving backward process evaluation*/
#define        stop_forward 23   /* stop the car moving forward evaluation*/
#define        stop_backward 24  /* stop the car moving backward evaluations */
#define        emerg        25   /* emergency driving evaluation*/

#define        acc          31   /* acceleration process evaluation*/
#define        deacc        32   /* deceleration process evaluation*/
#define        left_turn    33   /* left_turn driving */
#define        right_turn   34   /* right_turn driving */
#define        straight     35   /* straight driving */

#define        wdelay       41   /* wiper at delay status */
#define        wlow         42   /* wiper at low status */
#define        wmed         43   /* wiper at meddle status */
#define        whigh        44   /* wiper at high status */

/*****************************************************************************
*  Parameters  read  from  blackboard.
*****************************************************************************/

typedef struct
{
  int        accel;       /* acceleration pedal scaling from 0 to 10 */
  int        brake;       /* status: on or off */
  int        blinker;     /* status: off, right, left or emergen */
  int        clock;       /* clock time */
  int        clutch;      /* status: off or on */
```

```
    int         cruise;         /* status: off or on */
    int         direction;      /* status: straight, right or left turn*/
    int         distance;       /* number from mileage meter */
    int         gear;           /* status: n,1,2,3,4,r */
    int         key;            /* status: on or off */
    struct {
            int fog;            /* fog light status: on or off */
            int high;           /* low light status: on or off */
            int low;            /* high light status: on or off */
    } light;
    int         phase;          /* status: start engine, start driving,
                                    drive forward, drive backward
                                    stop car, drive under emergency */
    int         parking_brake; /* status: on or off */
    int         seat_belt;      /* status: on or off */
    int         speed;          /* number in speed meter */
    int         speed_limit;    /* road speed limit */
    int         time;           /* status: day or night */
    int         turning_location; /* the turning location (unit: miles) */
    int         wiper;          /* status: delay, low, med or high */
    int         weather;        /* status: clear, snow, rain, or foggy */

} external_status;

external_status pre_status, cur_status;  /* read in status from blackboard.
                                            pre_status recordes last status.
                                            cur_status reads the current status. */


/*****************************************************************************
*  Counters and flags assisting program running.
*****************************************************************************/

int  time_counter           = 0;   /* time counter */
int  time_high_counter      = 0;   /* time counter */
int  weather_counter        = 0;   /* time counter */
int  emergency_counter      = 0;   /* time counter */
int  driving_counter        = 0;   /* time counter */
int  weather_counter_1      = 0;   /* time counter */
int  blinker_counter        = 0;   /* time counter */

int  weather_speed_flag     = 0;
int  driving_flag           = 0;
int  emergency_flag         = 0;
int  weather_flag           = 0;
int  time_high_flag         = 0;
int  time_flag              = 0;
int  weather_flag_1         = 0;
```

```
/*****************************************************************************
*  Parameters used to record internal status.
*****************************************************************************/

int  accel_time;          /* record the time of acceleration on */
int  brake_time;          /* record the time of braking */
int  blinker_location;    /* record the time of blinker on */
int  brake_off_time;      /* record the time of braker off */
int  clutch_off_time;     /* record the time of clutch off */
int  clutch_on_time;      /* record the time of clutch on */
int  gear_1_time;         /* record the time of changing to gear 1 */
int  gear_r_time;         /* record the time of changing to gear r */
int  key_time;            /* record the time of turning on key */
int  key_off_time;        /* record the time of turning off key */
int  park_brake_time;     /* record the time of park_brake on or off */
int  seat_belt_time;      /* record the time of fasten seat belt */
int  speed_change;        /* acceleration or deceleration */
int  turning_speed;       /* record the user's turning speed */


/*****************************************************************************
*  Output parameters from the user model.
*****************************************************************************/

int   expert_counter     = 0;   /* expert skill records of driver */
int   normal_counter     = 0;   /* normal skill records of driver */
int   novice_counter     = 0;   /* novice skill records of driver */
int   total_score        = 0;   /* total score representing user model */
float average_score      = 0;   /* average score representing user model */

char    msg_accel[50];            /* information massage to the user */
char    msg_blinker[50];
char    msg_brake[50];
char    msg_cruise[50];
char    msg_fog_light[50];
char    msg_key[50];
char    msg_lights[50];
char    msg_operation[50];
char    msg_parking_brake[50];
char    msg_seat_belt[50];
char    msg_wiper[50];

/*****************************************************************************
*  Procedure clear_msg(): clearing old messages.
*****************************************************************************/

void clear_msg(){

        strcpy(msg_accel, "");
```

```
        strcpy(msg_blinker, "");
        strcpy(msg_brake,"");
        strcpy(msg_cruise,"");
       strcpy(msg_fog_light,"");
        strcpy(msg_key,"");
        strcpy(msg_lights,"");
        strcpy(msg_operation,"");
       strcpy(msg_parking_brake,"");
        strcpy(msg_seat_belt,"");
        strcpy(msg_wiper,"");
}

/*****************************************************************************
* Procedure display(): displaying the results of each cycle during the
* testing.
*****************************************************************************/

void display(){

    fprintf(fpw, "\ncycle %d\n \n", pre_status.clock);


    fprintf(fpw, "accel %d          ", pre_status.accel);


    if (pre_status.blinker == left)
            fprintf(fpw, "blinker left\t");
    else if (pre_status.blinker == right)
            fprintf(fpw, "blinker right\t");
    else
            fprintf(fpw, "blinker off\t");


    if (pre_status.brake == off)
            fprintf(fpw, "brake off\t");
    else
            fprintf(fpw, "brake on\t");


    fprintf(fpw, "clock %d          ", pre_status.clock);


    if (pre_status.clutch == off)
            fprintf(fpw, "clutch off\n");
    else
            fprintf(fpw, "clutch on\n");


    if (pre_status.cruise == off)
            fprintf(fpw, "cruise off\t");
    else
            fprintf(fpw, "cruise on\t");
```

```
if (pre_status.direction = = right)
        fprintf(fpw, "direction right\t");
else if (pre_status.direction = = left)
        fprintf(fpw, "direction right\t");
else
        fprintf(fpw, "direction straight\t");


fprintf(fpw, "distance %d mile    \t", pre_status.distance);


fprintf(fpw, "gear %d\n", pre_status.gear);


if (pre_status.key = = on)
        fprintf(fpw, "key on    \t");
else
        fprintf(fpw, "key off   \t");


if (pre_status.light.low = = on)
        fprintf(fpw, "low light on\t");
else
        fprintf(fpw, "low light off\t");


if (pre_status.light.fog = = on)
        fprintf(fpw, "fog light on\t");
else
        fprintf(fpw, "fog light off\t");


if (pre_status.light.high = = on)
        fprintf(fpw, "high light on\n");
else
        fprintf(fpw, "high light off\n");


if (pre_status.parking_brake = = on)
        fprintf(fpw, "parking brake on\t");
else
        fprintf(fpw, "parking brake off\t");


if (pre_status.phase = = forward)
        fprintf(fpw, "phase forward\t");
else if (pre_status.phase = = backward)
        fprintf(fpw, "phase backward\t");
else if (pre_status.phase = = stop_forward)
        fprintf(fpw, "phase stop_forward\t");
else if (pre_status.phase = = stop_backward)
```

```
        fprintf(fpw, "phase stop_backward\t");
else
        fprintf(fpw, "phase emergency\t");


if (pre_status.seat_belt = = on)
        fprintf(fpw, "safty belt on\t");
else
        fprintf(fpw, "safty belt off\t");


fprintf(fpw, "speed %d mph\n", pre_status.speed);


fprintf(fpw, "limit_speed %d mph   \t", pre_status.speed_limit);


if (pre_status.time = = day)
        fprintf(fpw, "time day\t");
else
        fprintf(fpw, "time night\t");


fprintf(fpw, "turning location %d mile\n", pre_status.turning_location);


if (pre_status.weather = = clear)
        fprintf(fpw, "weather clear\t");
else if (pre_status.weather = = snow)
        fprintf(fpw, "weather snow\t");
else if (pre_status.weather = = rain)
        fprintf(fpw, "weather rain\t");
else
        fprintf(fpw, "weather fog\t");


if (pre_status.wiper = = wdelay)
        fprintf(fpw, "wiper delay\t");
else if (pre_status.wiper = = wlow)
        fprintf(fpw, "wiper low\t");
else if (pre_status.wiper = = wmed)
        fprintf(fpw, "wiper medium\t");
else
        fprintf(fpw, "wiper high\t");


fprintf(fpw, "expert %d\t   ", expert_counter);

fprintf(fpw, "normal %d\t", normal_counter);

fprintf(fpw, "novice %d\n", novice_counter);
```

```
    total_score = 2*expert_counter + normal_counter - novice_counter;

    if (total_score != 0)
            average_score = (float) total_score/((float) (expert_counter+
            normal_counter + novice_counter));

    fprintf(fpw, "total_score %d \t", total_score);

    fprintf(fpw, "average_score %0.2f\n \n", average_score);

    if (strlen(msg_accel) > 5)
       fprintf(fpw, "msg_accel: %s\n", msg_accel);

    if (strlen(msg_blinker) > 5)
       fprintf(fpw, "msg_blinker: %s\n", msg_blinker);

    if (strlen(msg_brake) > 5)
       fprintf(fpw, "msg_brake: %s\n", msg_brake);

    if (strlen(msg_cruise) > 5)
       fprintf(fpw, "msg_cruise: %s\n", msg_cruise);

    if (strlen(msg_fog_light) > 5)
       fprintf(fpw, "msg_fog_light: %s\n", msg_fog_light);

    if (strlen(msg_key) > 5)
       fprintf(fpw, "msg_key: %s\n", msg_key);

    if (strlen(msg_lights) > 5)
       fprintf(fpw, "msg_lights: %s\n", msg_lights);

    if (strlen(msg_operation) > 5)
       fprintf(fpw, "msg_operation: %s\n", msg_operation);

    if (strlen(msg_parking_brake) > 5)
       fprintf(fpw, "msg_parking_brake: %s\n", msg_parking_brake);

    if (strlen(msg_seat_belt) > 5)
       fprintf(fpw, "msg_seat_belt: %s\n", msg_seat_belt);

    if (strlen(msg_wiper) > 5)
       fprintf(fpw, "msg_wiper: %s\n", msg_wiper);

}


/*****************************************************************************
* Input parameters to establish a user model.
*****************************************************************************/

int     expert_time = 1;     /* the expert's response time */
int     normal_time = 2;     /* the normal user's response time */
```

```
/*****************************************************************************
* Initialize the status to run the user model.
*****************************************************************************/

void initiate(){

    pre_status.accel = 0;
    pre_status.blinker = off;
    pre_status.brake = off;
    pre_status.clock = 0;
    pre_status.clutch = on;
    pre_status.cruise = off;
    pre_status.distance = 0;
    pre_status.direction = straight;
    pre_status.gear = 1;
    pre_status.key = off;
    pre_status.light.low = off;
    pre_status.light.high = off;
    pre_status.light.fog = off;
    pre_status.parking_brake = on;
    pre_status.phase = forward;
    pre_status.seat_belt = off;
    pre_status.speed = 0;
    pre_status.speed_limit = 30;
    pre_status.time = day;
    pre_status.turning_location = off;
    pre_status.weather = clear;
    pre_status.wiper = off;

}


/*****************************************************************************
* read the status from the blackboard.
*****************************************************************************/

void read_data(){

        fscanf(fpr, "%d", &cur_status.blinker);

        fscanf(fpr, "%d", &cur_status.brake);

        fscanf(fpr, "%d", &cur_status.clock);

        fscanf(fpr, "%d", &cur_status.clutch);

        fscanf(fpr, "%d", &cur_status.cruise);

        fscanf(fpr, "%d", &cur_status.direction);

        fscanf(fpr, "%d", &cur_status.distance);
```

```
        fscanf(fpr, "%d", &cur_status.gear);

        fscanf(fpr, "%d", &cur_status.key);

        fscanf(fpr, "%d", &cur_status.light.low);

        fscanf(fpr, "%d", &cur_status.light.high);

        fscanf(fpr, "%d", &cur_status.light.fog);

        fscanf(fpr, "%d", &cur_status.parking_brake);

        fscanf(fpr, "%d", &cur_status.phase);

        fscanf(fpr, "%d", &cur_status.seat_belt);

        fscanf(fpr, "%d", &cur_status.speed);

        fscanf(fpr, "%d", &cur_status.speed_limit);

        fscanf(fpr, "%d", &cur_status.time);

        fscanf(fpr, "%d", &cur_status.turning_location);

        fscanf(fpr, "%d", &cur_status.weather);

        fscanf(fpr, "%d", &cur_status.wiper);
}

/*************************************************************************
* Procedure weather_clear(): In the clear weather, wiper should be off.
* When the clear weather comes in, the user should turn off the wiper
* immediately. Evaluation is based on the user's response time. If the
* wiper is turned on in the clear weather, novice counter adds one and
* a massage "The wiper is on" will be given. If the user does not
* response within the normal user's response time, The massage "The
* wiper is on." will also be given. The abrove rule is also applied to
* turn off the fog light.
*************************************************************************/

void weather_clear(){

 /* Turn off the wiper.
 If the wiper is turned off,
 evaluation is based on the user's response time. */

        if  (pre_status.wiper != off && cur_status.wiper == off) {
        if (weather_flag != 1) {
            if (weather_counter <= expert_time)
                expert_counter++;
            else if (time_counter <= normal_time)
```

```
            normal_counter++;
          else
            novice_counter++;
        }
          weather_counter = 0;
          weather_flag = 0;
          strcpy(msg_wiper, "");
      }

/* If the wiper is still on, the time counter(weather_counter).
And a message will be sent out dependent on conditions.*/

      else if (pre_status.wiper != off && cur_status.wiper != off) {
        weather_counter++;
          if (weather_counter > normal_time || weather_flag == 1)
            strcpy(msg_wiper,"The wiper is on.");
      }

/* If the wiper is turned on from "off" status,
the novice_counter will be increased by one and a massage will send out. */

      else if (pre_status.wiper == off && cur_status.wiper != off)  {
          novice_counter++;
          strcpy(msg_wiper,"The wiper is on.");
          weather_flag = 1;
      }

/* Turn off the fog light */

      if (pre_status.light.fog == on && cur_status.light.fog == off){
      if (weather_flag_1 != 1)
          if (weather_counter_1 <= expert_time)
            expert_counter++;
          else if (weather_counter_1 <= normal_time)
            normal_counter++;
          else
            novice_counter++;
        strcpy(msg_fog_light, "");
        weather_counter_1 = 0;
        weather_flag_1 = 0;
      }

      else if (pre_status.light.fog == on && cur_status.light.fog == on){
        weather_counter_1++;
          if (weather_counter_1 > normal_time || weather_flag_1 == 1)
            strcpy(msg_fog_light, "The foggy_light is on.");
      }
      else if (pre_status.light.fog == off && cur_status.light.fog == on){
          novice_counter++;
          strcpy(msg_fog_light, "The foggy_light is on.");
          weather_flag_1 = 1;
}     }
```

```
/*****************************************************************************
* Procedure weather_foggy(): When it is foggy, the foggy light should be
* turned on. Evaluation is based on the user's response time of turning off
* light. The weather_counter counts the needed response time. If the
* response time is more than the normal user's response time, a massage
* "The light is off." will be given. If the foggy light is turned off
* in this foggy weather, the massage will also be given.
*****************************************************************************/

void weather_foggy() {

 /* When the wiper is turned on,
 Evaluation is based on the user's response time. */

        if (pre_status.light.fog == off && cur_status.light.fog == on){
        if (weather_flag != 1)
            if (weather_counter <= expert_time)
                expert_counter++;
              else if (weather_counter <= normal_time)
                normal_counter++;
              else
                novice_counter++;
          strcpy(msg_fog_light, "");
          weather_counter = 0;
          weather_flag = 0;
        }

/* If the fog light is still off, the time counter(weather_counter)
   increases one. A massage sends out if the user's response time
   exceeds the normal user's response time or if the fog light
   is turned off from "on" status. */

        else if (pre_status.light.fog == off && cur_status.light.fog == off){
          weather_counter++;
              if (weather_counter > normal_time || weather_flag == 1)
                strcpy(msg_fog_light, "The foggy_light is off.");
        }
        else if (pre_status.light.fog == on && cur_status.light.fog == off){
              novice_counter++;
              strcpy(msg_fog_light, "The foggy_light is off.");
              weather_flag = 1;
        }
}
```

```
/*************************************************************************
* Procedure time_day(): Both the low light and high light should be
* off during the day time. Evaluation is based on the response time and
* the correctness of the user's action.
* *************************************************************************/

void time_day(){

        if (pre_status.light.low == on
           && cur_status.light.low == off
           && cur_status.light.high == off){
         if (time_flag != 1)
           if (time_counter <= expert_time)
              expert_counter++;
            else if (time_counter <= normal_time)
              normal_counter++;
            else
              novice_counter++;
            strcpy(msg_lights, "");
            time_counter = 0;
            time_flag = 0;
       }

        else if (pre_status.light.high == on
              && cur_status.light.high == off
              && cur_status.light.low == off){
       if (time_flag != 1)
           if (time_counter <= expert_time)
              expert_counter++;
            else if (time_counter <= normal_time)
              normal_counter++;
            else
              novice_counter++;
            strcpy(msg_lights, "");
            time_counter = 0;
            time_flag = 0;
      }

       else if ((pre_status.light.low == on
              && cur_status.light.low == on)
              || (pre_status.light.high == on
              && cur_status.light.high == on)) {
         time_counter++;
            if (time_counter > normal_time || time_flag == 1)
              strcpy(msg_lights,"The light is on.");
       }

       else if ((pre_status.light.low == off
              && cur_status.light.low == on)
              || (pre_status.light.high == off
              && cur_status.light.high == on)){
```

```
                novice_counter++;
                strcpy(msg_lights,"The light is on.");
                time_flag = 1;
            }
}


/*****************************************************************************
* Procedure time_night(): when the driving speed is more than 55mph, it is
* assumed driving on freeway. Both low and high light can be used. When
* the speed lower than 55 mph, it is assumed driving on local roads.
* Only the low light can be used and the high light can be used for a
* short period of time(for instance, less than 30 seconds.
*****************************************************************************/

void time_night()
{
    if (cur_status.speed_limit > 55 )   /* driving on freeway */ {
        if ((pre_status.light.low == off && cur_status.light.low == on
            && cur_status.light.high == off) ||
              (pre_status.light.high == off && cur_status.light.high == on
              && cur_status.light.low == off)){
          if (time_counter != 1 )
              if (time_counter < expert_time)
                expert_counter++;
              else if (time_counter < normal_time)
                normal_counter++;
              else
                novice_counter++;
          strcpy(msg_lights,"");
          time_counter = 0;
          time_flag = 0;

       }

        else if (pre_status.light.low == off && cur_status.light.low == off
              && pre_status.light.high == off && cur_status.light.high == off) {
          time_counter++;
              if (time_counter > normal_time || time_flag == 1)
                strcpy(msg_lights,"The light is off.");
       }

        else if ((pre_status.light.low == on && cur_status.light.low == off
                && cur_status.light.high == off) ||
              (pre_status.light.high == on && cur_status.light.high == off
                && cur_status.light.low == off)){
            novice_counter++;
            strcpy(msg_lights,"The light is off.");
            time_flag = 1;
        }

    }
```

```
if (cur_status.speed_limit <= 55) {

        if (pre_status.light.low == off && cur_status.light.low == on){
        if (time_flag != 1)
            if (time_counter < expert_time)
                expert_counter++;
              else if (time_counter < normal_time)
                normal_counter++;
              else
                novice_counter++;
              strcpy(msg_lights, "");
              time_counter = 0;
              time_flag = 0;
        }

        else if (pre_status.light.low == off && cur_status.light.low == off){
          time_counter++;
              if (time_counter > normal_time || time_flag == 1)
                strcpy(msg_lights,"The low_light is off.");
        }

        else if (pre_status.light.low == on && cur_status.light.low == off){
              novice_counter++;
              strcpy(msg_lights,"The low_light is off.");
              time_flag = 1;
        }

/* When the high light is on, the program counts the time in which the high
   light is on. The evaluation is based on the the time. The high can be
   turned on for a short period of time. */

        if (pre_status.light.high == on && cur_status.light.high == off){
        if (time_high_flag !=1)
            if (time_high_counter < expert_time)
                expert_counter++;
              else if (time_high_counter < normal_time)
                normal_counter++;
              else
                novice_counter++;
              strcpy(msg_lights,"");
              time_high_counter = 0;
              time_high_flag = 0;
        }
        else if (pre_status.light.high == on && cur_status.light.high == on){
          time_high_counter++;

              if (time_high_counter > normal_time || (time_high_flag == 1 &&
                time_high_counter > normal_time))
              strcpy(msg_lights,"The high light is on.");
        }
        else if (pre_status.light.high == off && cur_status.light.high == on){
              time_high_flag = 1;
```

```
            time_high_counter++;
          }

      }
}


/*****************************************************************************
* Procedure weather_snow_rain(): This procedure is used for both the rain
* and snow weather. The wiper should be used when it rains or snows.
* Depending on the driving speed, the delay, low, med of high is used
* respectively. The rain or snow is assumed to be constant amount.
* Evaluation is based on the response time and the correctness of the
* user's actions.
*****************************************************************************/

void weather_snow_rain(){

/* When speed <= 15 mph, the wiper should be in the deley. */

          if (pre_status.wiper != wdelay && cur_status.wiper == wdelay){
            if (weather_flag != 1)
                if (time_counter < expert_time && cur_status.speed <= 15)
                  expert_counter++;
                else if (time_counter < normal_time)
                  normal_counter++;
                else
                  novice_counter++;
          else
              if (weather_speed_flag ==1)
                  novice_counter++;
                strcpy(msg_wiper, "");
                weather_counter = 0;
                weather_flag = 0;
                weather_speed_flag = 0;
          }

/* When speeds > 15 mph and <= 25 mhp, the wiper should be in
   the low. */

          if (pre_status.wiper != wlow && cur_status.wiper == wlow){
            if (weather_flag != 1)
                if (cur_status.speed > 15 && cur_status.speed <= 25 &&
              weather_counter < expert_time)
                    expert_counter++;
                else if (weather_counter < normal_time)
                  normal_counter++;
                else
                  novice_counter++;
            else if (weather_speed_flag == 1)
                  novice_counter++;
```

```
            strcpy(msg_wiper, "");
            weather_counter = 0;
            weather_flag = 0;
        weather_speed_flag = 0;
      }

/* When the speed > 25 mph and <= 40 mph, the wiper should
   be in the meddle. */

      if (pre_status.wiper != wmed && cur_status.wiper == wmed){
        if (weather_flag != 1)
            if (weather_counter < expert_time && cur_status.speed > 25
               && cur_status.speed <= 40)
              expert_counter++;
            else if (weather_counter < normal_time)
              normal_counter++;
            else
              novice_counter++;
        else if (weather_speed_flag == 1)
              novice_counter++;
            strcpy(msg_wiper, "");
            weather_counter = 0;
            weather_flag = 0;
        weather_speed_flag = 0;
      }

/* When speed > 40, the wiper should be in the high. */

      if (pre_status.wiper != whigh && cur_status.wiper == whigh){
        if (weather_flag != 1)
            if (weather_counter < expert_time && cur_status.speed > 40)
              expert_counter++;
            else if (weather_counter < normal_time && cur_status.speed>20)
              normal_counter++;
            else
              novice_counter++;
        else if (weather_speed_flag ==1)
            novice_counter++;

            strcpy(msg_wiper, "");
            weather_counter = 0;
            weather_flag = 0;
        weather_speed_flag = 0;
      }

/* When the wiper is off or at the wrong status, information will be
   sent to the user. */

      else if (pre_status.wiper == off && cur_status.wiper == off){
        weather_counter++;
            if (weather_counter > normal_time || weather_flag == 1)
              strcpy(msg_wiper,"The wiper is off.");
```

```
    }

    else if (pre_status.wiper != off && cur_status.wiper == off){
          novice_counter++;
          strcpy(msg_wiper,"The wiper is off.");
          weather_flag = 1;
    }

    if (pre_status.wiper == wdelay && cur_status.wiper == wdelay){
     if (cur_status.speed > 15)
          weather_counter++;
     else
          weather_counter = 0;

   if (weather_counter > normal_time)
     {    strcpy(msg_wiper,"The wiper is not at the correct status.");
          weather_flag = 1;
          weather_flag = 1;
     }
   }

    else if (pre_status.wiper == wlow && cur_status.wiper == wlow){
     if (cur_status.speed < 15 || cur_status.speed > 25)
          weather_counter++;
     else
          weather_counter = 0;

   if (weather_counter > normal_time)
     {    strcpy(msg_wiper,"The wiper is not at the correct status.");
          weather_flag = 1;
     weather_speed_flag =1;
     }
}

   else if (pre_status.wiper == wmed && cur_status.wiper == wmed){
     if (cur_status.speed < 25 || cur_status.speed > 40)
          weather_counter++;
     else
          weather_counter = 0;

   if (weather_counter > normal_time)
     {    strcpy(msg_wiper,"The wiper is not at the correct status.");
          weather_flag = 1;
     weather_speed_flag = 1;
     }
}

   else if (pre_status.wiper == whigh && cur_status.wiper == whigh){
     if (cur_status.speed < 40)
          weather_counter++;
     else
          weather_counter = 0;
```

```
            if (weather_counter > normal_time)
              {    strcpy(msg_wiper,"The wiper is not at the correct status.");
                   weather_speed_flag = 1;
              weather_flag = 1;
              }
          }
  }


/*************************************************************************
* Procedure turn_left(): The blinker should turned on 100 feet before the
* turning location. The driving speed is recorded at the turning location.
* Evaluation is based on comparing the turning speed with the speed limit.
* When the car passing the turning location, the blinker has to be turned
* off.
*************************************************************************/

void right_blinker_off(){

          if (pre_status.blinker == right
              && cur_status.blinker != right)
             strcpy(msg_blinker, "");
            else if (pre_status.blinker == right
                     && cur_status.blinker == right)
                    strcpy(msg_blinker, "The right blinker is on.");
            else if (pre_status.blinker != right
                     && cur_status.blinker == right){
                  novice_counter++;
                  strcpy(msg_blinker, " The right blinker is on.");
          }
}




void turn_left() {

                  right_blinker_off();

    if (pre_status.distance < cur_status.turning_location
            && cur_status.distance >= cur_status.turning_location){
        turning_speed = cur_status.speed;      /*record the turning speed.*/

          if (blinker_location < (cur_status.turning_location-0.004)
            && blinker_location > (cur_status.turning_location-0.01))
                  expert_counter++;
          else if (blinker_location < (cur_status.turning_location-0.0045)
            && blinker_location > (cur_status.turning_location-0.015))
                  normal_counter++;
          else                    /*rating based on blinker_on location*/
                  novice_counter++;
```

```
        if (turning_speed <= (cur_status.speed_limit+1)
            && turning_speed > (cur_status.speed_limit - 5))
          expert_counter++;
        else if (turning_speed < (cur_status.speed_limit+5)
                && turning_speed > (cur_status.speed_limit-10))
          normal_counter++;
        else
          novice_counter++;   /*rating based on speed*/
    }


    if (pre_status.blinker != left
                        && cur_status.blinker == left)
        blinker_location = cur_status.distance;
                        /*record the blinker on location.*/

    if (pre_status.distance > cur_status.turning_location)
        if (pre_status.blinker == left
                        && cur_status.blinker == left)
          { blinker_counter++;
            if (blinker_counter>normal_time)
            strcpy(msg_blinker, "The blinker is on");
          }
        else if (pre_status.blinker == left
                        && cur_status.blinker != left){
          if (blinker_counter < expert_time)
              expert_counter++;
          else if (blinker_counter < normal_time)
              normal_counter++;
          else
              novice_counter++;      /*rating based on turning off time*/
          strcpy(msg_blinker, "");
          blinker_counter = 0;

        }
}


/*****************************************************************************
* Procedure turn_right(): Its principle is same as the procedure
* turn_right().
*****************************************************************************/

void left_blinker_off(){

        if (pre_status.blinker == left
            && cur_status.blinker != left)
          strcpy(msg_blinker, "");
        else if (pre_status.blinker == left
                && cur_status.blinker == left)
              strcpy(msg_blinker, " The left blinker is on.");
```

```
        else if (pre_status.blinker != left
                && cur_status.blinker == left){
            novice_counter++;
            strcpy(msg_blinker, " The left blinker is on.");
        }

}


void turn_right() {

            left_blinker_off();

    if (pre_status.distance < cur_status.turning_location
            && cur_status.distance >= cur_status.turning_location){
        turning_speed = cur_status.speed;

        if (blinker_location < (cur_status.turning_location-0.004)
            && blinker_location > (cur_status.turning_location-0.01))
                expert_counter++;
        else if (blinker_location < (cur_status.turning_location-0.0045)
            && blinker_location > (cur_status.turning_location-0.015))
                normal_counter++;
        else
                novice_counter++;

        if (turning_speed <= (cur_status.speed_limit+1)
            && turning_speed > (cur_status.speed_limit - 5))
            expert_counter++;
        else if (turning_speed < (cur_status.speed_limit+5)
                && turning_speed > (cur_status.speed_limit-10))
            normal_counter++;
        else
            novice_counter++;}

    if (pre_status.blinker != right
                        && cur_status.blinker == right)
        blinker_location = cur_status.distance;

    if (pre_status.distance > cur_status.turning_location)
        if (pre_status.blinker == right
                        && cur_status.blinker == right)
            { blinker_counter++;
             if (blinker_counter>normal_time)
             strcpy(msg_blinker,"blinker is on");
            }
        else if (pre_status.blinker == right
                        && cur_status.blinker != right){
            if (blinker_counter < expert_time)
                expert_counter++;
            else if (blinker_counter < normal_time)
                normal_counter++;
```

```
            else
                novice_counter++;
            strcpy(msg_blinker, "");
            blinker_counter = 0;
        }

}


/*****************************************************************************
* Procedure acceleration_off(): if the acceleration is not off, the
* novice action counter adds one.
*****************************************************************************/

void acceleration_off() {

        if (pre_status.accel != 0
            && cur_status.accel == 0)
          strcpy(msg_accel, "");
        else if (pre_status.accel != 0
                && cur_status.accel != 0)
              strcpy(msg_accel, "The gas is added into engine.");
        else if (pre_status.accel == 0
                && cur_status.accel != 0){
            novice_counter++;
            strcpy(msg_accel, " The gas is added into engine.");
        }
}


/*****************************************************************************
* Procedure-blinker_off(): When driving straight, blinker lights should
* be turned off.
*****************************************************************************/

void blinker_off() {

        strcpy(msg_blinker, "");
        if (pre_status.blinker != off
            && cur_status.blinker == off)
          strcpy(msg_blinker, "");
        else if (pre_status.blinker != off
                && cur_status.blinker != off)
              strcpy(msg_blinker, "The blinker is not off.");
        else if (pre_status.blinker == off
                && cur_status.blinker != off){
            novice_counter++;
            strcpy(msg_blinker, " The left blinker is not off.");
        }

}
```

```
/*****************************************************************************
* Procedure emergency_light_on(): When an accident happens to the car, the
* emergency light should be turned on.
*****************************************************************************/

void emergency_light_on() {

        if (pre_status.blinker != emergen
           && cur_status.blinker == emergen){
        if (emergency_flag != 1)
            if (emergency_counter <= expert_time)
                expert_counter++;
              else if (weather_counter <= normal_time)
                normal_counter++;
              else
                novice_counter++;
          strcpy(msg_blinker, "");
          emergency_counter = 0;
          emergency_flag = 0;
       }
        else if (pre_status.blinker != emergen
                 && cur_status.blinker != emergen){
          emergency_counter++;
              if (emergency_counter > normal_time || emergency_flag == 1)
                strcpy(msg_blinker, "The emergency_light is off.");
        }
        else if (pre_status.blinker == emergen
                 && cur_status.blinker != emergen){
              novice_counter++;
              strcpy(msg_blinker, "The emergency_light is off.");
              emergency_flag = 1;
        }
}


/*****************************************************************************
* Procedure seat_belt_on(): if the safty belt is off, the novice action
* counter adds one.
*****************************************************************************/

void seat_belt_on() {

       if (pre_status.seat_belt == off
          && cur_status.seat_belt == on)
         strcpy(msg_seat_belt, "");
       else if (pre_status.seat_belt == off
                && cur_status.seat_belt == off)
              strcpy(msg_seat_belt, "The seat belt is off.");
       else if (pre_status.seat_belt == on
                && cur_status.seat_belt == off){
            novice_counter++;
            strcpy(msg_seat_belt, "The seat belt is off.");
```

```
        }
}


/*************************************************************************
* Procedure key_on(): if the key is off, the novice action counter
* adds one.
*************************************************************************/

void key_on() {

        if (pre_status.key == off
            && cur_status.key == on)
          strcpy(msg_key, "");
        else if (pre_status.key == off
                  && cur_status.key == off)
                strcpy(msg_key, "The key is off.");
        else if (pre_status.key == on
                  && cur_status.key == off){
              novice_counter++;
              strcpy(msg_key, " The key is off.");
        }
}


/*************************************************************************
* Procedure cruise_off(): if cruise is on, the novice action counter
* adds one.
*************************************************************************/

void cruise_off() {

        if (pre_status.cruise == on
            && cur_status.cruise == off)
          strcpy(msg_cruise, "");
        else if (pre_status.cruise == on
                  && cur_status.cruise == on)
                strcpy(msg_cruise, "The cruise is on.");
        else if (pre_status.cruise == off
                  && cur_status.cruise == on){
              novice_counter++;
              strcpy(msg_cruise, " The cruise is on.");
        }
}
```

```
/*************************************************************************
* Procedure parkig_brake_on(): if the parking brake is on, the novice
* action counter adds one.
*************************************************************************/

void parking_brake_off() {

        if (pre_status.parking_brake == on
           && cur_status.parking_brake == off)
          strcpy(msg_parking_brake, "");
        else if (pre_status.parking_brake == on
                 && cur_status.parking_brake == on)
               strcpy(msg_parking_brake, "The parking_brake is on.");
        else if (pre_status.parking_brake == off
                 && cur_status.parking_brake == on){
              novice_counter++;
              strcpy(msg_parking_brake, " The parking_brake is on.");
        }
}


/*************************************************************************
* Procedure start_engine(): It records the time when the user fastens
* the seat belt, operates on clutch and brake, and turns on the key.
* The evaluation is based on the sequence of these operations.
*************************************************************************/

void start_engine() {

          cruise_off();

   if (pre_status.seat_belt == off && cur_status.seat_belt == on)
        seat_belt_time = cur_status.clock;
   /* record the time of selt belt operation. */

   if (pre_status.clutch == on && cur_status.clutch == off)
        clutch_off_time = cur_status.clock;
   /* record the time of clutch operation. */

   if (pre_status.brake == off && cur_status.brake == on)
        brake_time = cur_status.clock;
   /*record the time of brake operation. */

   if (pre_status.key == off && cur_status.key == on){
        if (cur_status.gear == n || cur_status.clutch == off) {
            if (seat_belt_time <= clutch_off_time &&
                 seat_belt_time <= brake_time &&
                 cur_status.clutch == off &&
                 cur_status.seat_belt == on &&
                 (cur_status.brake == on ||
                 cur_status.parking_brake == on) &&
                 cur_status.blinker == off)
```

```
                    expert_counter++;
            else if
                (cur_status.gear == n &&
                (cur_status.parking_brake == on ||
                cur_status.brake == on) &&
                cur_status.seat_belt == on &&
                cur_status.blinker == off)
                    expert_counter++;
            else if
                (cur_status.seat_belt == on
                && (cur_status.brake == on
                || cur_status.parking_brake == on))
                normal_counter++;
            else
                novice_counter++;
                strcpy(msg_operation, "");
        }
          else {
            strcpy(msg_operation,"Engine can not be started. Try again");
            novice_counter++;
        }
    }
        /* rating the driver's skill. */

}


/*****************************************************************************
* Procedure start_car(): this procedure records the sequence of the user's
* operations from starting engine to car moving. The evaluation is based on
* comparing the operations with the task model.
*****************************************************************************/

void start_car(){


    if (pre_status.parking_brake == on && cur_status.parking_brake == off)
        park_brake_time = cur_status.clock;
        /*record parking brake off time*/

    if (pre_status.clutch == on && cur_status.clutch == off)
        clutch_off_time = cur_status.clock;
        /*record clutch off time*/

    if (pre_status.brake == off && cur_status.brake == on)
        brake_time = cur_status.clock;
        /*record brake on time */

    if (pre_status.gear != 1 &&
        cur_status.gear == 1 &&
        cur_status.clutch == off
        && cur_status.brake == on){
```

```
            expert_counter++;
            normal_counter++;}
            /*rating during change gear*/

        if (pre_status.brake == on && cur_status.brake == off)
            brake_off_time = cur_status.clock;
            /*record the brake off time*/

        if (pre_status.accel == 0 && cur_status.accel != 0)
            accel_time = cur_status.clock;
            /*record push acceleration time*/

        if (pre_status.clutch == off
            && cur_status.clutch == on
            && cur_status.gear == 1){ /*rating at this step*/
            clutch_on_time = cur_status.clock;
            if (cur_status.brake == off && cur_status.parking_brake == off){
            if ( brake_time <= park_brake_time
                && brake_time <= clutch_off_time
                && brake_off_time <= accel_time
                && accel_time  <= clutch_on_time)
                    expert_counter++;
            else
            if (brake_off_time <= accel_time
                && accel_time  <= clutch_on_time)
                    normal_counter++;
            else
                    novice_counter++;
            strcpy(msg_operation,"");
            }
            else{
                novice_counter++;
                strcpy(msg_operation,"");
            }
        }
}


/*****************************************************************************
* Procedure acceleration(): This procedure finds if the user changes
* the gear in the correct speed range during acceleration. The number of
* unIt in the procedure is mile per hour.
*****************************************************************************/

void acceleration (){

        if (pre_status.gear != 2 && cur_status.gear == 2)
            if (pre_status.gear != 1)
                novice_counter++;

        if (pre_status.gear !=3 && cur_status.gear == 3)
            if(pre_status.gear !=2)
```

```
        novice_counter++;

    if (pre_status.gear !=4 && cur_status.gear == 4)
        if(pre_status.gear !=3)
          novice_counter++;

    if (pre_status.clutch == off && cur_status.clutch == on
    && cur_status.gear == 2)
        if (cur_status.speed >= 5 && cur_status.speed <= 10)
            expert_counter++;
        else if (cur_status.speed >= 2.5 && cur_status.speed <= 12.5)
            normal_counter++;
        else
            novice_counter++;
        /* Evaluating the user's actions when gear is changed to 2. */

    else
    if (pre_status.clutch == off && cur_status.clutch == on
        && cur_status.gear == 3)
        if (cur_status.speed >= 20 && cur_status.speed <= 25)
            expert_counter++;
        else if ( cur_status.speed>= 17.5 && cur_status.speed <= 27.5)
         normal_counter++;
        else
            novice_counter++;
        /* Evaluating the user's actions when gear is changed to 3. */

    else
    if (pre_status.clutch == off && cur_status.clutch == on
         && cur_status.gear == 4)
        if (cur_status.speed >= 35 && cur_status.speed <= 40)
            expert_counter++;
        else if ((cur_status.speed >= 32.5 && cur_status.speed <= 42.5))
         normal_counter++;
        else
            novice_counter++;
        /* Evaluating the user's actions when gear is changed to 4. */


}

/*****************************************************************************
* Procedure deceleration(): This procedure finds if the driver changes
* the gear in the correct speed range during deceleration.
*****************************************************************************/

void deceleration(){

    if (pre_status.clutch == off && cur_status.clutch == on
        && cur_status.gear == 3)
         if (cur_status.speed >=40 && cur_status.speed <= 45)
            expert_counter++;
```

```
        else if (cur_status.speed >= 37.5 && cur_status.speed <= 47.5)
            normal_counter++;
        else
            novice_counter++;
       /* Evaluating the user's actions when gear is changed to 3. */

    else
    if (pre_status.clutch == off && cur_status.clutch == on
        && cur_status.gear == 2)
        if (cur_status.speed >= 30 && cur_status.speed <= 35)
            expert_counter++;
        else if (cur_status.speed >= 27.5 && cur_status.speed <= 37.5)
            normal_counter++;
        else
            novice_counter++;
       /* Evaluating the user's actions when gear is changed to 2. */

    else
    if (pre_status.clutch == off && cur_status.clutch == on
        && cur_status.gear == 1)
        if (cur_status.speed >=10  && cur_status.speed <= 15)
            expert_counter++;
        else if
            (cur_status.speed >= 7.5 && cur_status.speed <= 17.5)
            normal_counter++;
        else
            novice_counter++;
       /* Evaluating the user's actions when gear is changed to 2. */
}


/*****************************************************************************
* Procedure stop_from_idle(): It evaluates the user's performance
* when the user stops the car from 7 mph.
*****************************************************************************/

void stop_from_idle() {

   if (cur_status.speed < 3){
        if (pre_status.clutch == off && cur_status.clutch == on)
        novice_counter++;
   }
   /* When speed<3mph, clutch should not be on again. */

        if (pre_status.speed != 0 && cur_status.speed == 0
            && (cur_status.gear == 1 || cur_status.gear == r)) {
        if (cur_status.clutch == off
            && cur_status.brake == on
            && cur_status.accel == 0
            && cur_status.parking_brake == off
        ){
            expert_counter++;
```

```
            normal_counter++;
          }
          else
            novice_counter++;
    }
}



/*****************************************************************************
* Procedure parking_action(): This procedure evaluates the user's
* parking operations.
*****************************************************************************/

void parking_action(){

   if (pre_status.parking_brake == off
        && cur_status.parking_brake == on){
         if (cur_status.brake == on)
           expert_counter++;
         else
         normal_counter++;
         park_brake_time = cur_status.clock;
   }
   if (pre_status.clutch == on && cur_status.clutch == off)
        clutch_off_time = cur_status.clock;
   if (pre_status.brake == off && cur_status.brake == on)
        brake_time = cur_status.clock;
   if (pre_status.gear != 1 && cur_status.gear == 1
     && cur_status.clutch ==off
     && (cur_status.brake == on
     || cur_status.parking_brake == on))
        gear_1_time = cur_status.clock;
   if (pre_status.gear != r && cur_status.gear == r
     && cur_status.clutch ==off
     && (cur_status.brake == on
     || cur_status.parking_brake == on))
        gear_r_time = cur_status.clock;
   if (pre_status.brake == on && cur_status.brake == off)
     brake_off_time = cur_status.clock;
   if (pre_status.key == on && cur_status.key == off)
     key_off_time = cur_status.clock;
   if (pre_status.clutch == off && cur_status.clutch == on){
     clutch_on_time = cur_status.clock;
    if ((cur_status.parking_brake == on
         || cur_status.brake == on)
         && cur_status.accel == 0)
     if (
            (clutch_off_time <= gear_r_time
           || clutch_off_time <= gear_1_time)
           && key_off_time <= clutch_on_time
```

```
            && (cur_status.gear == r || cur_status.gear == 1))
            expert_counter++;
        else if (key_off_time <= clutch_on_time
             && (cur_status.gear == r
             || cur_status.gear == 1))
            normal_counter++;
        else
            novice_counter++;

      else
          novice_counter++;
    }}
/*****************************************************************************
* Procedure backward_operation(): Evaluation on the user'action is based
* on the driving back operations.
*****************************************************************************/

void backward_operation(){

    if (pre_status.parking_brake == on
        && cur_status.parking_brake == off)
          if (cur_status.brake == on)
            expert_counter++;
          else
            normal_counter++;
    /* When the car stops, brake should be on before the parking
      brake is off. */

    if (pre_status.clutch == on && cur_status.clutch == off)
          clutch_off_time = cur_status.clock;
    if (pre_status.brake == off && cur_status.brake == on)
          brake_time = cur_status.clock;
    if (pre_status.gear != r && cur_status.gear == r)
        if (cur_status.clutch == off)
          if (driving_flag != 1){
            if (cur_status.brake == on
            || cur_status.parking_brake == on)
                expert_counter++;
            else
          normal_counter++;
                gear_r_time = cur_status.clock;
                strcpy(msg_operation, "");
                driving_flag = 0;
        }
        else {
            novice_counter++;
            strcpy(msg_operation, "The clutch is on.");
            driving_flag = 1;
        }
      /*rating the skill of changing gear to r. */

   if (pre_status.brake == on && cur_status.brake == off)
```

```
            brake_off_time = cur_status.clock;
        if (pre_status.accel == 0 && cur_status.accel > 0)
            accel_time = cur_status.clock;
        if (pre_status.clutch == off && cur_status.clutch == on){
            clutch_on_time = cur_status.clock;
            if (cur_status.brake == off
                && cur_status.parking_brake == off
                && cur_status.gear == r){
            if (driving_flag != 1)
             if (
                brake_time <= clutch_off_time
              && clutch_off_time <= gear_r_time
              && brake_off_time <= accel_time
              && accel_time  <= clutch_on_time)
              expert_counter++;
            else if
              (
              accel_time  <= clutch_on_time)
              normal_counter++;
            else
                novice_counter++;
                strcpy(msg_operation,"");
                driving_flag = 0;
            }
           else {
                novice_counter++;
                strcpy(msg_operation,"The brake is on.");
                driving_flag = 1;
           }
          } /* rating the skill of backward driving. */
}


/*****************************************************************************
* Procedure emergent_action: When the user is informed that the car is in
* emergency, he should stops immediately. The evaluation is based on the
* the user's action to process this situation.
*****************************************************************************/

void emergent_action(){
            strcpy(msg_brake, "");
  if (cur_status.speed > 7) {
    if ((pre_status.parking_brake == off
      && cur_status.parking_brake == on)
      || (pre_status.brake == off
      && cur_status.brake == on))
     {
            novice_counter++;
            strcpy(msg_brake, "Should not use the brake.");
      }

    if ((pre_status.parking_brake == on
```

```
    && cur_status.parking_brake == on)
    || (pre_status.brake == on
    && cur_status.brake == on))
        strcpy(msg_brake, "Should not use the brake.");

   if ((pre_status.parking_brake == on
    && cur_status.parking_brake == off)
    || (pre_status.brake == on
    && cur_status.brake == off))
        strcpy(msg_brake, "");
  /* When speed>7mph, brakes should not be used. */
 }
 else {
        if ((pre_status.brake == off
          && cur_status.brake == on)
          ||(pre_status.parking_brake == off
          && cur_status.parking_brake == on)){
          if (driving_counter != 1 )
              if (driving_counter < expert_time)
                  expert_counter++;
                else if (driving_counter < normal_time)
                  normal_counter++;
                else
                  novice_counter++;
            strcpy(msg_brake,"");
            driving_counter = 0;
            driving_flag = 0;

        }

          else if (pre_status.brake == off
             && cur_status.brake == off
                && pre_status.parking_brake == off
                && cur_status.parking_brake == off) {
            driving_counter++;
                if (driving_counter > normal_time)
                  strcpy(msg_brake,"The brake is off.");
          }

          else if ((pre_status.brake == on
                    && cur_status.brake == off
                    && cur_status.parking_brake == off) ||
                  (pre_status.parking_brake == on
                  && cur_status.parking_brake == off
                    && cur_status.brake == off))
                driving_flag = 1;
          /*The brake should be used when speed<7mph.*/
 }

}
```

```
/*****************************************************************************
* Procedure driving_forward(): It evaluates the user's performance when the
* the user drives car forward.
*****************************************************************************/

void start_driving(){

                    start_car();
                    key_on();
                    seat_belt_on();
                    cruise_off();

}

void driving_car_forward(){

                parking_brake_off();
                    key_on();
                    seat_belt_on();
                    if (cur_status.speed < 50)
                      cruise_off();

                if (cur_status.speed > (cur_status.speed_limit+5))
                        speed_change = deacc;

                if (cur_status.speed < (cur_status.speed_limit-5))
                        speed_change = acc;

                if (cur_status.speed < (cur_status.speed_limit+5)
                && cur_status.speed >= (cur_status.speed_limit+5)) {
                        speed_change = deacc;
                        novice_counter++;
                }

                if (cur_status.speed < (cur_status.speed_limit-5)
                && cur_status.speed >= (cur_status.speed_limit-5)) {
                        speed_change = acc;
                        novice_counter++;
                }

                switch(speed_change){
                        case acc: acceleration();
                                  break;
                        case deacc: deceleration();
                                    break;
                }

}

void driving_forward(){
```

```
    switch(cur_status.direction){

    case left_turn:      turn_left(); break;
    case right_turn:     turn_right(); break;
    case straight:         blinker_off(); break;

    }

                        if (cur_status.key == off
                          ||(pre_status.key == off &&
                              cur_status.key == on))
                             start_engine();

                        else if ((cur_status.key == on
                                 ||(pre_status.key == off &&
                                    cur_status.key == on)) &&
                          cur_status.speed == 0)
                             start_driving();

                        else if (cur_status.key == on
                                 && cur_status.speed >0)
                             driving_car_forward();

}


/*****************************************************************************
* Procedure stop_driving_forward(): It evaluates the user's performance
* when the user tries to stop a car moving forward.
*****************************************************************************/

void stop_car_forward(){

                        deceleration();
                        stop_from_idle();
                      key_on();
                        parking_brake_off();
                        seat_belt_on();

}

void stop_driving_forward(){
                        if (cur_status.speed >0
                              ||(pre_status.speed>0
                              &&cur_status.speed==0))
                                stop_car_forward();

                        else if (cur_status.speed == 0
                              && pre_status.speed== 0 )
                                parking_action();

                        acceleration_off();
```

```
                    cruise_off();

                    switch(cur_status.direction){

                        case left_turn:      turn_left(); break;
                        case right_turn:     turn_right(); break;
                        case straight:       blinker_off(); break;

                    }}
/*****************************************************************************
* Procedure driving_backward(): It evaluates the user's performance when
* the user drives a car backward.
*****************************************************************************/

void stop_car(){

            deceleration();
            stop_from_idle();
            acceleration_off();

}

void driving_backward(){

            seat_belt_on();
            blinker_off();
            cruise_off();
            key_on();
           parking_brake_off();

            if (cur_status.speed != 0 && cur_status.gear != r)
            stop_car();
            else
            backward_operation();

}

/*****************************************************************************
* Procedure stop_driving_backward(): It evaluates the user's performance
* when the user tries to stop car moving backward.
*****************************************************************************/

void stop_car_backward(){

                  stop_from_idle();
                key_on();
                  parking_brake_off();
                  seat_belt_on();

}
```

```
void stop_driving_backward(){

                    if (cur_status.speed >0
                        ||(pre_status.speed>0
                        &&cur_status.speed==0))
                          stop_car_backward();

                    else if (cur_status.speed == 0
                        && pre_status.speed== 0 )
                          parking_action();

                    acceleration_off();
                    cruise_off();
                    blinker_off();

}



/*****************************************************************************
* Procedure process_emergency(): It evaluates the user's performance when
* the user drives under emergency situation.
*****************************************************************************/

void stop_emergency(){

                    emergent_action();
                    key_on();
                    seat_belt_on();

}

void process_emergency(){

                    if (cur_status.speed >0
                        ||(pre_status.speed>0
                        &&cur_status.speed==0)) {
                          stop_emergency();
                          deceleration(); }

                    else if (cur_status.speed == 0
                        && pre_status.speed== 0 )
                          parking_action();

                    acceleration_off();
                    cruise_off();
              if (pre_status.phase != emerg){
                        emergency_flag = 0;
                        emergency_counter = 0;
```

```
                    }
                    emergency_light_on();
}


/*****************************************************************************
* Procedure driving(): It evaluates the user's performance on manipulating
* car on the road.
*****************************************************************************/

void driving(){

    switch(cur_status.phase) {

    case forward:          driving_forward();
                        break;

    case stop_forward:     stop_driving_forward();
                        break;

    case backward:         driving_backward();
                       break;

    case stop_backward:    stop_driving_backward();
                        break;

    case emerg:            process_emergency();
                        break;
    }
}


/*****************************************************************************
* Procedure environment(): This procedure evaluates the user's actions
* with the change of weather and time.
*****************************************************************************/

void environment(){

    switch(cur_status.weather) {

    case clear: if (pre_status.weather != clear){
                    weather_flag = 0;
                    weather_counter = 0;
                weather_flag_1 = 0;
                    weather_counter_1 = 0;
                }
                weather_clear();
                break;
```

```
case foggy: if (pre_status.weather != foggy){
                weather_flag = 0;
                weather_counter = 0;
            }
            weather_foggy();
            break;

case rain:  if (pre_status.weather != rain ||
                pre_status.weather != snow){
                weather_flag = 0;
                weather_counter = 0;
                weather_speed_flag = 0;
            }
            weather_snow_rain();
            break;

case snow:  if (pre_status.weather != snow ||
                pre_status.weather != rain){
                weather_flag = 0;
                weather_counter = 0;
            weather_speed_flag = 0;
            }
            weather_snow_rain();
        break;

}

switch(cur_status.time) {

case day:       if (pre_status.time != day){
                    time_flag = 0;
                    time_counter = 0;
                    }
                    time_day();
                    break;

case night:     if (pre_status.time != night){
                    time_flag = 0;
                    time_counter = 0;
                    }
                    time_night();
                    break;

}
}
```

```
/*****************************************************************************
* Procedure main(): It reads data from file data.in and write the result
* to file.
*****************************************************************************/

void main ()
{


  fpr = fopen("data.in","r");
  fpw = fopen("data.out","w");


        initiate();
        display();

 while(fscanf(fpr, "%d", &cur_status.accel) != EOF){

        read_data();
        clear_msg();
        environment();
        driving();
        pre_status = cur_status;
        display();

 }
        close(fpr);
        close(fpw);

}
```

APPENDIX C.

INPUT AND OUTPUT FILES OF RUNNING EXAMPLES

RUNNING EXAMPLE ONE

The Input File:

```
0 2 2 1 1 2 35 0 1 2 2 2 2 1 21 1 0 30 15 2 11 2
0 2 2 2 2 2 35 0 1 2 2 2 2 1 21 1 0 30 15 2 11 2
0 2 2 3 2 2 35 0 1 1 2 2 2 1 21 1 0 30 15 2 11 2
0 2 1 4 2 2 35 0 1 1 2 2 2 1 21 1 0 30 15 2 11 2
0 2 1 5 2 2 35 0 1 1 2 2 2 2 21 1 0 30 15 2 11 2
1 2 2 6 2 2 35 0 1 1 2 2 2 2 21 1 0 30 15 2 11 2
1 3 2 7 1 2 35 0 1 1 2 2 2 2 21 1 0 30 15 2 11 2
1 2 2 8 2 2 35 0 1 1 2 2 2 2 21 1 9 30 15 2 11 2
1 2 2 9 2 2 35 1 2 1 2 2 2 2 21 1 9 30 15 2 11 2
1 2 2 10 1 2 35 1 2 1 2 2 2 2 21 1 9 30 15 2 11 2
1 2 2 11 2 2 35 1 2 1 2 2 2 2 21 1 24 30 15 2 11 2
1 2 2 12 2 2 35 1 3 1 2 2 2 2 21 1 24 30 15 2 11 2
1 2 2 13 1 2 35 1 3 1 2 2 2 2 21 1 24 30 15 2 11 2
1 2 2 14 2 2 35 1 3 1 2 2 2 2 21 1 38 50 15 2 11 2
1 2 2 15 2 2 35 1 4 1 2 2 2 2 21 1 38 50 15 2 11 2
1 2 2 16 1 2 35 1 4 1 2 2 2 2 21 1 38 50 15 2 11 2
1 2 2 17 1 2 35 1 4 1 2 2 2 2 21 1 45 50 15 2 11 2
1 2 2 18 1 2 35 1 4 1 2 2 2 2 21 2 45 50 15 2 11 2
1 2 2 19 1 2 35 1 4 1 2 2 2 1 21 1 45 50 15 2 11 2
1 2 2 20 1 2 35 1 4 1 2 2 2 2 21 1 45 50 15 2 11 2
1 2 2 21 1 2 35 1 4 1 1 2 2 2 21 1 45 50 16 2 11 2
1 2 2 22 1 2 35 1 4 1 1 1 1 2 21 1 45 50 16 2 11 2
1 2 2 23 1 2 35 1 4 1 1 1 2 2 21 1 45 50 16 2 11 2
1 2 2 24 1 2 35 1 4 1 1 2 2 2 21 1 50 60 16 2 11 2
1 2 2 25 1 2 35 1 4 1 1 1 2 2 21 1 50 60 16 2 11 44
1 2 2 26 1 2 35 1 4 1 1 1 2 2 21 1 50 60 16 2 11 2
1 2 2 27 1 2 35 1 4 1 1 1 2 2 21 1 50 60 16 2 13 2
1 2 2 28 1 2 35 1 4 1 2 2 1 2 21 1 50 60 15 2 13 2
1 2 2 29 1 2 35 1 4 1 2 2 1 2 21 1 50 60 15 2 12 42
1 2 2 30 1 2 33 1 4 1 2 2 1 2 21 1 50 60 15 2 12 42
1 2 2 31 1 2 33 1 4 1 2 2 1 2 21 1 50 60 15 2 12 42
1 2 2 32 1 2 33 1 4 1 2 2 2 2 21 1 50 60 15 2 12 44
1 2 2 33 1 2 33 1 4 1 2 2 2 2 21 1 50 60 15 2 11 2
1 2 2 34 2 2 33 1 4 1 2 2 2 2 21 1 45 60 15 2 11 2
1 2 2 35 2 2 33 1 3 1 2 2 2 2 21 1 40 25 15 2 11 2
1 4 2 36 1 2 33 1 3 1 2 2 2 2 21 1 40 25 15 2 11 2
1 2 2 37 2 2 33 1 3 1 2 2 2 2 21 1 40 25 15 2 11 2
1 3 2 38 2 2 33 9 2 1 2 2 2 2 21 1 33 25 15 10 11 2
1 3 2 39 1 2 33 11 2 1 2 2 2 2 21 1 25 25 15 10 11 2
1 3 2 40 1 2 33 11 2 1 2 2 2 2 21 1 25 2 15 0 11 2
1 2 2 41 2 2 33 11 2 1 2 2 2 2 21 1 15 2 15 0 11 2
0 2 1 42 2 2 35 11 1 1 2 2 2 2 23 1 15 2 15 0 11 2
0 2 1 43 1 2 35 11 1 1 2 2 2 2 23 1 15 2 15 0 11 2
0 2 1 44 1 1 35 11 1 1 2 2 2 2 23 1 7 2 15 0 11 2
0 2 2 45 1 2 35 11 1 1 2 2 2 2 23 1 7 2 15 0 11 2
0 2 2 46 1 2 35 11 1 1 2 2 2 2 23 1 2 2 15 0 11 2
```

0 2 1 47 2 2 35 11 1 1 2 2 2 2 23 1 1 2 15 0 11 2
0 2 1 48 2 2 35 11 1 1 2 2 2 2 23 1 0 2 15 0 11 2
0 2 1 49 2 2 35 11 1 1 2 2 2 1 23 1 0 2 15 0 11 2
0 2 2 50 2 2 35 11 1 1 2 2 2 1 23 1 0 2 15 0 11 2
0 2 2 51 2 2 35 11 1 2 2 2 2 1 23 1 0 2 15 0 11 2
0 2 2 52 1 2 35 11 1 2 2 2 2 1 23 1 0 2 15 0 11 2

The Output File:

cycle 0

accel 0 blinker off brake off clock 0 clutch on
cruise off direction straight distance 0 mile gear 1
key off low light off fog light off high light off
parking brake on phase forward safty belt off speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 0 novice 0
total_score 0 average_score 0.00

cycle 1

accel 0 blinker off brake off clock 1 clutch on
cruise off direction straight distance 0 mile gear 1
key off low light off fog light off high light off
parking brake on phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 0 novice 0
total_score 0 average_score 0.00

cycle 2

accel 0 blinker off brake off clock 2 clutch off
cruise off direction straight distance 0 mile gear 1
key off low light off fog light off high light off
parking brake on phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 0 novice 0
total_score 0 average_score 0.00

cycle 3

accel 0 blinker off brake off clock 3 clutch off
cruise off direction straight distance 0 mile gear 1
key on low light off fog light off high light off
parking brake on phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 1 novice 0

total_score 1 average_score 1.00

cycle 4

accel 0 blinker off brake on clock 4 clutch off
cruise off direction straight distance 0 mile gear 1
key on low light off fog light off high light off
parking brake on phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 1 novice 0
total_score 1 average_score 1.00

cycle 5

accel 0 blinker off brake on clock 5 clutch off
cruise off direction straight distance 0 mile gear 1
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 1 novice 0
total_score 1 average_score 1.00

cycle 6

accel 1 blinker off brake off clock 6 clutch off
cruise off direction straight distance 0 mile gear 1
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 1 novice 0
total_score 1 average_score 1.00

cycle 7

accel 1 blinker left brake off clock 7 clutch on
cruise off direction straight distance 0 mile gear 1
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 2 novice 1
total_score 1 average_score 0.33

msg_blinker: The left blinker is not off.

cycle 8

accel 1 blinker off brake off clock 8 clutch off
cruise off direction straight distance 0 mile gear 1

key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 9 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 2 novice 1
total_score 1 average_score 0.33

cycle 9

accel 1 blinker off brake off clock 9 clutch off
cruise off direction straight distance 1 mile gear 2
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 9 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 2 novice 1
total_score 1 average_score 0.33

cycle 10

accel 1 blinker off brake off clock 10 clutch on
cruise off direction straight distance 1 mile gear 2
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 9 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 1 normal 2 novice 1
total_score 3 average_score 0.75

cycle 11

accel 1 blinker off brake off clock 11 clutch off
cruise off direction straight distance 1 mile gear 2
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 24 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 1 normal 2 novice 1
total_score 3 average_score 0.75

cycle 12

accel 1 blinker off brake off clock 12 clutch off
cruise off direction straight distance 1 mile gear 3
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 24 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 1 normal 2 novice 1
total_score 3 average_score 0.75

cycle 13

accel 1 blinker off brake off clock 13 clutch on
cruise off direction straight distance 1 mile gear 3
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 24 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 2 normal 2 novice 1
total_score 5 average_score 1.00

cycle 14

accel 1 blinker off brake off clock 14 clutch off
cruise off direction straight distance 1 mile gear 3
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 38 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 2 normal 2 novice 1
total_score 5 average_score 1.00

cycle 15

accel 1 blinker off brake off clock 15 clutch off
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 38 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 2 normal 2 novice 1
total_score 5 average_score 1.00

cycle 16

accel 1 blinker off brake off clock 16 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 38 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 3 normal 2 novice 1
total_score 7 average_score 1.17

cycle 17

accel 1 blinker off brake off clock 17 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 45 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 3 normal 2 novice 1

total_score 7 average_score 1.17

cycle 18

accel 1 blinker off brake off clock 18 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase forward safty belt off speed 45 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 3 normal 2 novice 2
total_score 6 average_score 0.86

msg_seat_belt: The seat belt is off.

cycle 19

accel 1 blinker off brake off clock 19 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake on phase forward safty belt on speed 45 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 3 normal 2 novice 3
total_score 5 average_score 0.62

msg_parking_brake: The parking_brake is on.

cycle 20

accel 1 blinker off brake off clock 20 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 45 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 3 normal 2 novice 3
total_score 5 average_score 0.62

cycle 21

accel 1 blinker off brake off clock 21 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light on fog light off high light off
parking brake off phase forward safty belt on speed 45 mph
limit_speed 50 mph time night turning location 2 mile
weather clear wiper high expert 4 normal 2 novice 3
total_score 7 average_score 0.78

cycle 22

accel 1 blinker off brake off clock 22 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light on fog light on high light on
parking brake off phase forward safty belt on speed 45 mph
limit_speed 50 mph time night turning location 2 mile
weather clear wiper high expert 4 normal 2 novice 4
total_score 6 average_score 0.60

msg_fog_light: The foggy_light is on.

cycle 23

accel 1 blinker off brake off clock 23 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light on fog light off high light on
parking brake off phase forward safty belt on speed 45 mph
limit_speed 50 mph time night turning location 2 mile
weather clear wiper high expert 4 normal 2 novice 4
total_score 6 average_score 0.60

cycle 24

accel 1 blinker off brake off clock 24 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light on fog light off high light off
parking brake off phase forward safty belt on speed 50 mph
limit_speed 60 mph time night turning location 2 mile
weather clear wiper high expert 4 normal 2 novice 4
total_score 6 average_score 0.60

cycle 25

accel 1 blinker off brake off clock 25 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light on fog light off high light on
parking brake off phase forward safty belt on speed 50 mph
limit_speed 60 mph time night turning location 2 mile
weather clear wiper high expert 4 normal 2 novice 5
total_score 5 average_score 0.45

msg_wiper: The wiper is on.

cycle 26

accel 1 blinker off brake off clock 26 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light on fog light off high light on
parking brake off phase forward safty belt on speed 50 mph

limit_speed 60 mph time night turning location 2 mile
weather clear wiper high expert 4 normal 2 novice 5
total_score 5 average_score 0.45

cycle 27

accel 1 blinker off brake off clock 27 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light on fog light off high light on
parking brake off phase forward safty belt on speed 50 mph
limit_speed 60 mph time night turning location 2 mile
weather fog wiper high expert 4 normal 2 novice 5
total_score 5 average_score 0.45

cycle 28

accel 1 blinker off brake off clock 28 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light on high light off
parking brake off phase forward safty belt on speed 50 mph
limit_speed 60 mph time day turning location 2 mile
weather fog wiper high expert 6 normal 2 novice 5
total_score 9 average_score 0.69

cycle 29

accel 1 blinker off brake off clock 29 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light on high light off
parking brake off phase forward safty belt on speed 50 mph
limit_speed 60 mph time day turning location 2 mile
weather rain wiper low expert 6 normal 3 novice 5
total_score 10 average_score 0.71

cycle 30

accel 1 blinker off brake off clock 30 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light on high light off
parking brake off phase forward safty belt on speed 50 mph
limit_speed 60 mph time day turning location 2 mile
weather rain wiper low expert 6 normal 3 novice 5
total_score 10 average_score 0.71

cycle 31

accel 1 blinker off brake off clock 31 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light on high light off
parking brake off phase forward safty belt on speed 50 mph
limit_speed 60 mph time day turning location 2 mile
weather rain wiper low expert 6 normal 3 novice 5
total_score 10 average_score 0.71

cycle 32

accel 1 blinker off brake off clock 32 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 50 mph
limit_speed 60 mph time day turning location 2 mile
weather rain wiper high expert 7 normal 3 novice 5
total_score 12 average_score 0.80

cycle 33

accel 1 blinker off brake off clock 33 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 50 mph
limit_speed 60 mph time day turning location 2 mile
weather clear wiper high expert 8 normal 3 novice 5
total_score 14 average_score 0.88

cycle 34

accel 1 blinker off brake off clock 34 clutch off
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 45 mph
limit_speed 60 mph time day turning location 2 mile
weather clear wiper high expert 8 normal 3 novice 5
total_score 14 average_score 0.88

cycle 35

accel 1 blinker off brake off clock 35 clutch off
cruise off direction straight distance 1 mile gear 3
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 40 mph
limit_speed 25 mph time day turning location 2 mile
weather clear wiper high expert 8 normal 3 novice 5

total_score 14 average_score 0.88

cycle 36

accel 1 blinker right brake off clock 36 clutch on
cruise off direction straight distance 1 mile gear 3
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 40 mph
limit_speed 25 mph time day turning location 2 mile
weather clear wiper high expert 9 normal 3 novice 6
total_score 15 average_score 0.83

msg_blinker: The right blinker is on.

cycle 37

accel 1 blinker off brake off clock 37 clutch off
cruise off direction straight distance 1 mile gear 3
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 40 mph
limit_speed 25 mph time day turning location 2 mile
weather clear wiper high expert 9 normal 3 novice 6
total_score 15 average_score 0.83

cycle 38

accel 1 blinker left brake off clock 38 clutch off
cruise off direction straight distance 9 mile gear 2
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 33 mph
limit_speed 25 mph time day turning location 10 mile
weather clear wiper high expert 9 normal 3 novice 6
total_score 15 average_score 0.83

cycle 39

accel 1 blinker left brake off clock 39 clutch on
cruise off direction straight distance 11 mile gear 2
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 25 mph
limit_speed 25 mph time day turning location 10 mile
weather clear wiper high expert 10 normal 3 novice 8
total_score 15 average_score 0.71

cycle 40

accel 1 blinker left brake off clock 40 clutch on
cruise off direction straight distance 11 mile gear 2
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 25 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 10 normal 3 novice 8
total_score 15 average_score 0.71

cycle 41

accel 1 blinker off brake off clock 41 clutch off
cruise off direction straight distance 11 mile gear 2
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 15 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 10 normal 4 novice 8
total_score 16 average_score 0.73

cycle 42

accel 0 blinker off brake on clock 42 clutch off
cruise off direction straight distance 11 mile gear 1
key on low light off fog light off high light off
parking brake off phase stop_forward safty belt on speed 15 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 10 normal 4 novice 8
total_score 16 average_score 0.73

cycle 43

accel 0 blinker off brake on clock 43 clutch on
cruise off direction straight distance 11 mile gear 1
key on low light off fog light off high light off
parking brake off phase stop_forward safty belt on speed 15 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 11 normal 4 novice 8
total_score 18 average_score 0.78

cycle 44

accel 0 blinker off brake on clock 44 clutch on
cruise on direction straight distance 11 mile gear 1
key on low light off fog light off high light off
parking brake off phase stop_forward safty belt on speed 7 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 11 normal 4 novice 9

total_score 17 average_score 0.71

msg_cruise: The cruise is on.

cycle 45

accel 0 blinker off brake off clock 45 clutch on
cruise off direction straight distance 11 mile gear 1
key on low light off fog light off high light off
parking brake off phase stop_forward safty belt on speed 7 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 11 normal 4 novice 9
total_score 17 average_score 0.71

cycle 46

accel 0 blinker off brake off clock 46 clutch on
cruise off direction straight distance 11 mile gear 1
key on low light off fog light off high light off
parking brake off phase stop_forward safty belt on speed 2 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 11 normal 4 novice 9
total_score 17 average_score 0.71

cycle 47

accel 0 blinker off brake on clock 47 clutch off
cruise off direction straight distance 11 mile gear 1
key on low light off fog light off high light off
parking brake off phase stop_forward safty belt on speed 1 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 11 normal 4 novice 9
total_score 17 average_score 0.71
cycle 48

accel 0 blinker off brake on clock 48 clutch off
cruise off direction straight distance 11 mile gear 1
key on low light off fog light off high light off
parking brake off phase stop_forward safty belt on speed 0 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 12 normal 5 novice 9
total_score 20 average_score 0.77

cycle 49

accel 0 blinker off brake on clock 49 clutch off
cruise off direction straight distance 11 mile gear 1
key on low light off fog light off high light off
parking brake on phase stop_forward safty belt on speed 0 mph

limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 13 normal 5 novice 9
total_score 22 average_score 0.81

cycle 50

accel 0 blinker off brake off clock 50 clutch off
cruise off direction straight distance 11 mile gear 1
key on low light off fog light off high light off
parking brake on phase stop_forward safty belt on speed 0 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 13 normal 5 novice 9
total_score 22 average_score 0.81

cycle 51

accel 0 blinker off brake off clock 51 clutch off
cruise off direction straight distance 11 mile gear 1
key off low light off fog light off high light off
parking brake on phase stop_forward safty belt on speed 0 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 13 normal 5 novice 9
total_score 22 average_score 0.81

cycle 52

accel 0 blinker off brake off clock 52 clutch on
cruise off direction straight distance 11 mile gear 1
key off low light off fog light off high light off
parking brake on phase stop_forward safty belt on speed 0 mph
limit_speed 2 mph time day turning location 0 mile
weather clear wiper high expert 13 normal 6 novice 9
total_score 23 average_score 0.82

RUNNING EXAMPLE TWO

The Input File:

```
0 2 2 1 1 2 35 0 1 2 2 2 2 1 21 1 0 30 15 2 11 2
0 2 2 2 2 2 35 0 1 2 2 2 2 1 21 1 0 30 15 2 11 2
0 2 2 3 2 2 35 0 1 1 2 2 2 1 21 1 0 30 15 2 11 2
0 2 1 4 2 2 35 0 1 1 2 2 2 1 21 1 0 30 15 2 11 2
0 2 1 5 2 2 35 0 1 1 2 2 2 2 21 1 0 30 15 2 11 2
1 2 2 6 2 2 35 0 1 1 2 2 2 2 21 1 0 30 15 2 11 2
1 3 2 7 1 2 35 0 1 1 2 2 2 2 21 1 0 30 15 2 11 2
1 2 2 8 2 2 35 0 1 1 2 2 2 2 21 1 9 30 15 2 11 2
1 2 2 9 2 2 35 1 2 1 2 2 2 2 21 1 9 30 15 2 11 2
```

1 2 2 10 1 2 35 1 2 1 2 2 2 2 21 1 9 30 15 2 11 2
1 2 2 11 2 2 35 1 2 1 2 2 2 2 21 1 24 30 15 2 11 2
1 2 2 12 2 2 35 1 3 1 2 2 2 2 21 1 24 30 15 2 11 2
1 2 2 13 1 2 35 1 3 1 2 2 2 2 21 1 24 30 15 2 11 2
1 2 2 14 2 2 35 1 3 1 2 2 2 2 21 1 38 50 15 2 11 2
1 2 2 15 2 2 35 1 4 1 2 2 2 2 21 1 38 50 15 2 11 2
1 2 2 16 1 2 35 1 4 1 2 2 2 2 21 1 38 50 15 2 11 2
1 2 2 17 1 2 35 1 4 1 2 2 2 2 21 1 45 50 15 2 11 2
0 2 2 18 1 2 35 1 4 1 2 2 2 2 25 1 45 50 15 2 11 2
0 6 2 19 1 2 35 1 4 1 2 2 2 2 25 1 40 50 15 2 11 2
0 6 1 20 1 2 35 1 4 1 2 2 2 2 25 1 35 50 15 2 11 2
0 6 2 21 1 2 35 1 4 1 2 2 2 2 25 1 35 50 15 2 11 2
1 6 2 22 2 2 35 1 4 1 2 2 2 2 25 1 35 50 15 2 11 2
0 6 2 23 2 2 35 1 2 1 2 2 2 2 25 1 35 50 15 2 11 2
0 6 2 24 1 2 35 1 2 1 2 2 2 2 25 1 35 50 15 2 11 2
0 6 2 25 2 2 35 1 2 1 2 2 2 2 25 1 25 50 15 2 11 2
0 6 2 26 2 2 35 1 1 1 2 2 2 2 25 1 25 50 15 2 11 2
0 6 2 27 1 2 35 1 1 1 2 2 2 2 25 1 25 50 15 2 11 2
0 6 1 28 2 2 35 1 1 1 2 2 2 2 25 1 5 50 15 2 11 2
0 2 1 29 2 2 35 1 1 1 2 2 2 2 25 1 0 50 15 2 11 2
0 2 1 30 2 2 35 1 1 1 2 2 2 2 22 1 0 50 15 2 11 2
0 2 1 31 2 2 35 1 5 1 2 2 2 2 22 1 0 50 15 2 11 2
1 2 2 32 2 2 35 1 5 1 2 2 2 2 22 1 0 50 15 2 11 2
1 2 2 33 1 2 35 1 5 1 2 2 2 2 22 1 0 50 15 2 11 2
1 2 2 34 1 2 35 1 5 1 2 2 2 2 22 1 1 50 15 2 11 2
3 2 2 35 1 2 35 1 5 1 2 2 2 2 22 1 5 50 15 2 11 2
0 2 1 36 1 2 35 1 5 1 2 2 2 2 24 1 4 50 15 2 11 2
0 2 1 37 2 2 35 1 5 1 2 2 2 2 24 1 3 50 15 2 11 2
0 2 1 38 2 2 35 1 5 1 2 2 2 2 24 1 0 50 15 2 11 2
0 2 1 39 2 2 35 1 5 1 2 2 2 1 24 1 0 50 15 2 11 2
0 2 1 40 2 2 35 1 5 2 2 2 2 1 24 1 0 50 15 2 11 2
0 2 1 41 1 2 35 1 5 1 2 2 2 1 24 1 0 50 15 2 11 2

The Output File:

cycle 0

accel 0 blinker off brake off clock 0 clutch on
cruise off direction straight distance 0 mile gear 1
key off low light off fog light off high light off
parking brake on phase forward safty belt off speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 0 novice 0
total_score 0 average_score 0.00

cycle 1

accel 0 blinker off brake off clock 1 clutch on
cruise off direction straight distance 0 mile gear 1
key off low light off fog light off high light off

parking brake on phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 0 novice 0
total_score 0 average_score 0.00

cycle 2

accel 0 blinker off brake off clock 2 clutch off
cruise off direction straight distance 0 mile gear 1
key off low light off fog light off high light off
parking brake on phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 0 novice 0
total_score 0 average_score 0.00

cycle 3

accel 0 blinker off brake off clock 3 clutch off
cruise off direction straight distance 0 mile gear 1
key on low light off fog light off high light off
parking brake on phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 1 novice 0
total_score 1 average_score 1.00

cycle 4

accel 0 blinker off brake on clock 4 clutch off
cruise off direction straight distance 0 mile gear 1
key on low light off fog light off high light off
parking brake on phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 1 novice 0
total_score 1 average_score 1.00

cycle 5

accel 0 blinker off brake on clock 5 clutch off
cruise off direction straight distance 0 mile gear 1
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 1 novice 0
total_score 1 average_score 1.00

cycle 6

accel 1 blinker off brake off clock 6 clutch off
cruise off direction straight distance 0 mile gear 1
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 1 novice 0
total_score 1 average_score 1.00

cycle 7

accel 1 blinker left brake off clock 7 clutch on
cruise off direction straight distance 0 mile gear 1
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 0 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 2 novice 1
total_score 1 average_score 0.33

msg_blinker: The left blinker is not off.

cycle 8

accel 1 blinker off brake off clock 8 clutch off
cruise off direction straight distance 0 mile gear 1
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 9 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 2 novice 1
total_score 1 average_score 0.33

cycle 9

accel 1 blinker off brake off clock 9 clutch off
cruise off direction straight distance 1 mile gear 2
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 9 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 0 normal 2 novice 1
total_score 1 average_score 0.33

cycle 10

accel 1 blinker off brake off clock 10 clutch on
cruise off direction straight distance 1 mile gear 2
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 9 mph
limit_speed 30 mph time day turning location 2 mile

weather clear wiper high expert 1 normal 2 novice 1
total_score 3 average_score 0.75

cycle 11

accel 1 blinker off brake off clock 11 clutch off
cruise off direction straight distance 1 mile gear 2
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 24 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 1 normal 2 novice 1
total_score 3 average_score 0.75

cycle 12

accel 1 blinker off brake off clock 12 clutch off
cruise off direction straight distance 1 mile gear 3
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 24 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 1 normal 2 novice 1
total_score 3 average_score 0.75

cycle 13

accel 1 blinker off brake off clock 13 clutch on
cruise off direction straight distance 1 mile gear 3
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 24 mph
limit_speed 30 mph time day turning location 2 mile
weather clear wiper high expert 2 normal 2 novice 1
total_score 5 average_score 1.00

cycle 14

accel 1 blinker off brake off clock 14 clutch off
cruise off direction straight distance 1 mile gear 3
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 38 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 2 normal 2 novice 1
total_score 5 average_score 1.00

cycle 15

accel 1 blinker off brake off clock 15 clutch off
cruise off direction straight distance 1 mile gear 4

key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 38 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 2 normal 2 novice 1
total_score 5 average_score 1.00

cycle 16

accel 1 blinker off brake off clock 16 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 38 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 3 normal 2 novice 1
total_score 7 average_score 1.17

cycle 17

accel 1 blinker off brake off clock 17 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase forward safty belt on speed 45 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 3 normal 2 novice 1
total_score 7 average_score 1.17

cycle 18

accel 0 blinker off brake off clock 18 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase emergency safty belt on speed 45 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 3 normal 2 novice 1
total_score 7 average_score 1.17

cycle 19

accel 0 blinker off brake off clock 19 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase emergency safty belt on speed 40 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 4 normal 2 novice 1
total_score 9 average_score 1.29

cycle 20

accel 0 blinker off brake on clock 20 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase emergency safty belt on speed 35 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 4 normal 2 novice 2
total_score 8 average_score 1.00

msg_brake: Should not use the brake.

cycle 21

accel 0 blinker off brake off clock 21 clutch on
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase emergency safty belt on speed 35 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 4 normal 2 novice 2
total_score 8 average_score 1.00

cycle 22

accel 1 blinker off brake off clock 22 clutch off
cruise off direction straight distance 1 mile gear 4
key on low light off fog light off high light off
parking brake off phase emergency safty belt on speed 35 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 4 normal 2 novice 3
total_score 7 average_score 0.78

msg_accel: The gas is added into engine.

cycle 23

accel 0 blinker off brake off clock 23 clutch off
cruise off direction straight distance 1 mile gear 2
key on low light off fog light off high light off
parking brake off phase emergency safty belt on speed 35 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 4 normal 2 novice 3
total_score 7 average_score 0.78

cycle 24

accel 0 blinker off brake off clock 24 clutch on
cruise off direction straight distance 1 mile gear 2
key on low light off fog light off high light off

parking brake off phase emergency safty belt on speed 35 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 5 normal 2 novice 3
total_score 9 average_score 0.90

cycle 25

accel 0 blinker off brake off clock 25 clutch off
cruise off direction straight distance 1 mile gear 2
key on low light off fog light off high light off
parking brake off phase emergency safty belt on speed 25 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 5 normal 2 novice 3
total_score 9 average_score 0.90

cycle 26

accel 0 blinker off brake off clock 26 clutch off
cruise off direction straight distance 1 mile gear 1
key on low light off fog light off high light off
parking brake off phase emergency safty belt on speed 25 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 5 normal 2 novice 3
total_score 9 average_score 0.90

cycle 27

accel 0 blinker off brake off clock 27 clutch on
cruise off direction straight distance 1 mile gear 1
key on low light off fog light off high light off
parking brake off phase emergency safty belt on speed 25 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 5 normal 2 novice 4
total_score 8 average_score 0.73

cycle 28

accel 0 blinker off brake on clock 28 clutch off
cruise off direction straight distance 1 mile gear 1
key on low light off fog light off high light off
parking brake off phase emergency safty belt on speed 5 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 6 normal 2 novice 4
total_score 10 average_score 0.83

cycle 29

accel 0 blinker off brake on clock 29 clutch off
cruise off direction straight distance 1 mile gear 1
key on low light off fog light off high light off
parking brake off phase emergency safty belt on speed 0 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 6 normal 2 novice 5
total_score 9 average_score 0.69

msg_blinker: The emergency_light is off.

cycle 30

accel 0 blinker off brake on clock 30 clutch off
cruise off direction straight distance 1 mile gear 1
key on low light off fog light off high light off
parking brake off phase backward safty belt on speed 0 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 6 normal 2 novice 5
total_score 9 average_score 0.69

cycle 31

accel 0 blinker off brake on clock 31 clutch off
cruise off direction straight distance 1 mile gear 5
key on low light off fog light off high light off
parking brake off phase backward safty belt on speed 0 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 7 normal 2 novice 5
total_score 11 average_score 0.79

cycle 32

accel 1 blinker off brake off clock 32 clutch off
cruise off direction straight distance 1 mile gear 5
key on low light off fog light off high light off
parking brake off phase backward safty belt on speed 0 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 7 normal 2 novice 5
total_score 11 average_score 0.79

cycle 33

accel 1 blinker off brake off clock 33 clutch on
cruise off direction straight distance 1 mile gear 5
key on low light off fog light off high light off
parking brake off phase backward safty belt on speed 0 mph
limit_speed 50 mph time day turning location 2 mile

weather clear wiper high expert 7 normal 3 novice 5
total_score 12 average_score 0.80
cycle 34

accel 1 blinker off brake off clock 34 clutch on
cruise off direction straight distance 1 mile gear 5
key on low light off fog light off high light off
parking brake off phase backward safty belt on speed 1 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 7 normal 3 novice 5
total_score 12 average_score 0.80

cycle 35

accel 3 blinker off brake off clock 35 clutch on
cruise off direction straight distance 1 mile gear 5
key on low light off fog light off high light off
parking brake off phase backward safty belt on speed 5 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 7 normal 3 novice 5
total_score 12 average_score 0.80

cycle 36

accel 0 blinker off brake on clock 36 clutch on
cruise off direction straight distance 1 mile gear 5
key on low light off fog light off high light off
parking brake off phase stop_backward safty belt on speed 4 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 7 normal 3 novice 5
total_score 12 average_score 0.80

cycle 37

accel 0 blinker off brake on clock 37 clutch off
cruise off direction straight distance 1 mile gear 5
key on low light off fog light off high light off
parking brake off phase stop_backward safty belt on speed 3 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 7 normal 3 novice 5
total_score 12 average_score 0.80

cycle 38

accel 0 blinker off brake on clock 38 clutch off
cruise off direction straight distance 1 mile gear 5
key on low light off fog light off high light off
parking brake off phase stop_backward safty belt on speed 0 mph

limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 8 normal 4 novice 5
total_score 15 average_score 0.88

cycle 39

accel 0 blinker off brake on clock 39 clutch off
cruise off direction straight distance 1 mile gear 5
key on low light off fog light off high light off
parking brake on phase stop_backward safty belt on speed 0 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 9 normal 4 novice 5
total_score 17 average_score 0.94

cycle 40

accel 0 blinker off brake on clock 40 clutch off
cruise off direction straight distance 1 mile gear 5
key off low light off fog light off high light off
parking brake on phase stop_backward safty belt on speed 0 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 9 normal 4 novice 5
total_score 17 average_score 0.94

cycle 41

accel 0 blinker off brake on clock 41 clutch on
cruise off direction straight distance 1 mile gear 5
key on low light off fog light off high light off
parking brake on phase stop_backward safty belt on speed 0 mph
limit_speed 50 mph time day turning location 2 mile
weather clear wiper high expert 10 normal 4 novice 5
total_score 19 average_score 1.00