Assessment of A Human Computer Interface Prototyping Environment

Final Report

Delivery Order No. 16
Basic NASA Contract No. NAS8-39131

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

Contact Loretta A. Moore, Ph.D.
Principal Investigator
(205) 844-6330
moore@eng.auburn.edu

July 15, 1993

NASA
NATIONAL AERONAUTICS & SPACE ADMINISTRATION

Report Documentation Page

1. REPORT NO.	2. GOVERNMENT ACCESSION NO	3. RECIPIENT'S CATALOG NO

4. TITLE AND SUBTITLE

Assessment of a Human Computer Interface Prototyping Environment

5. REASON DATE

May 15, 1993

6. PERFORMING ORGANIZATION CODE

CSE, Auburn University

7. AUTHOR(S)

Loretta A. Moore, Ph.D.

8. PERFORMING ORGANIZATION REPORT NO

CSE 93-08

10. WORK UNIT NO.

Delivery Order No. 16

9. PERFORMING ORGANIZATION NAME AND ADDRESS

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

11. CONTRACT OR GRANT NO.

Basic NASA Contract No. NAS8-39131

13. TYPE OF REPORT AND PERIOD COVERED

Final Report
March 8-May 15, 1993

12. SPONSORING AGENCY NAME AND ADDRESS

National Aeronautics and Space Administration
Washington, D.C. 20546-0001
Marshall Space Flight Center, MSFC, AL 35812

14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

16. ABSTRACT

A Human Computer Interface (HCI) prototyping environment with embedded evaluation capability has been successfully assessed which will be valuable in developing and refining HCI standards and evaluating program/project interface development, especially Space Station Freedom on-board displays for payload operations. The HCI prototyping environment is designed to include four components: (1) a HCI format development tool, (2) a test and evaluation simulator development tool, (3) a dynamic, interactive interface between the HCI prototype and simulator, and (4) an embedded evaluation capability to evaluate the adequacy of an HCI based on a user's performance.

17. KEY WORDS (SUGGESTED BY AUTHORS)	18. DISTRIBUTION STATEMENT
Human Computer Interaction, Graphical User Interface Simulator	Unlimited

19. SECURITY CLASSIF. (OF THIS REPORT)	20. SECURITY CLASSIF. (OF THIS PAGE)	21. NO. OF PAGES	22. PRICE
Unclassified	Unclassified	27	

NASA FORM 1626 OCT 86

Assessment of A Human Computer Interface Prototyping Environment

Final Report

Delivery Order No. 16
Basic NASA Contract No. NAS8-39131

Loretta A. Moore, Ph.D.
Principal Investigator

July 15, 1993

Abstract

A Human Computer Interface (HCI) prototyping environment with embedded evaluation capability has been successfully assessed which will be valuable in developing and refining HCI standards and evaluating program/project interface development, especially Space Station Freedom on-board displays for payload operations. The HCI prototyping environment is designed to include four components: (1) a HCI format development tool, (2) a test and evaluation simulator development tool, (3) a dynamic, interactive interface between the HCI prototype and simulator, and (4) an embedded evaluation capability to evaluate the adequacy of an HCI based on a user's performance. The objectives of this research were to evaluate components 1, 2, and 4. The system which was chosen for empirical evaluation of this HCI prototyping environment was an automobile. Components of the dynamic prototyping environment were assessed in a two phase effort. During Phase I, the requirements for the automobile interface and simulator were developed, also during this phase prototypes of the interface and simulator were developed. During Phase II, evaluation criteria for the operation of the automobile prototype were developed.

ACKNOWLEDGEMENTS

We appreciate the assistance provided by NASA personnel, especially Mr. Joseph P. Hale whose guidance has been of great value. The following is an alphabetical listing of the team members who have participated on the project.

Principal Investigator:

Loretta A. Moore, Ph.D.

Graduate Research Assistants:

William Owen
Shannon Price
Yauwen Wang

The following trademarks are referenced in the text of this report.

Sammi is a trademark of Scientific Software-Intercomp.

PERCNET is a registered trademark of Perceptronics, Inc.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 Introduction

The objective of the project was to assess a Human Computer Interface (HCI) prototyping environment that includes: (1) an HCI format development tool, (2) a test and evaluation simulator development tool, and (3) the capability to provide a dynamic, interactive interface between the resultant prototyped HCIs and simulators, and (4) an embedded capability to evaluate the adequacy of an HCI based on a user's performance.

The HCI format development tool allows the designer to develop static displays dynamically. The test and evaluation simulator development tool will allow the functionality of the system to be implemented and will act as a driver for the displays. The dynamic, interactive interface will handle communication between the HCI prototyping environment and the simulation environment. The embedded evaluation tool will perform the evaluation of the Human-Computer Interaction in terms of specific evaluation measures.

The system which was chosen for empirical evaluation of this HCI prototyping environment was an automobile. An automobile was chosen because it has sufficient complexity and subsystems interdependencies to provide a moderate amount of operational workload. Further, potential subjects in the empirical studies would have a working understanding of an automobile's functionality, thus minimizing pre-experiment training requirements.

An extremely important aspect of the prototyping process is the ability to evaluate the adequacy of the prototyped HCIs. The goal of evaluation is to not only determine if an HCI is deficient, but point to specific aspects or attributes of the HCI that need improvement. Relevant evaluation data based on the user's interaction with the simulator through the HCI will be automatically collected and logged during the evaluation session. The system would then produce evaluation reports following the session.

Components of the dynamic prototyping environment were assessed in a two phase effort. During Phase I, the requirements for the automobile interface and simulator were developed, also during this phase prototypes of the interface and simulator were developed. During Phase II, evaluation criteria for the operation of automobile prototype were developed. A listing of the specific tasks which were performed are presented below.

(1) Requirements were developed for the automobile simulator.

(2) The automobile simulator was developed using PERCNET, a graphical modeling and knowledge-based simulation development environment.

(3) A Human Computer Interface (HCI) for operating the automobile simulator was developed using Sammi, a HCI development environment.

(4) Evaluation criteria for the operation of the automobile simulator were developed.

The products of this effort include an understanding of the preliminary requirements for a dynamic HCI prototyping environment and a test and evaluation simulator development tool, a prototype of an automobile using the test and evaluator simulator development tool, a human computer interface for the automobile prototype using the HCI development tool, and evaluation criteria for the operation of the automobile simulator. The following sections describe in detail the tasks which have been completed.

2.0 Simulator

Two common problems that arise in any system (computer system or otherwise) occur when the developer has an incomplete or inaccurate conception of system requirements and when users' needs are ignored as the central concern of development. A lack of understanding will lead to an error-ridden system that hinders user's productivity. If a computer system is developed to file tax returns can the system be considered an asset if calculations have to be double checked? The look and feel of a system are major factors when users evaluate a system. If users do not enjoy using a system, they will not use it effectively (if at all) regardless of the quality of the system.

Simulation of a system provides a means of studying a system with minimal expense or risk and an opportunity to study the effects of variations in the system. Thus, it facilitates a deeper understanding of the behavior of the system in that it tests the developer's own understanding of the problem at hand. If the developer overlooked certain possibilities or inaccurately represented certain aspects of the system, these should be apparent in the simulation. Expense and effort toward implementing a fault-ridden system would be reduced.

The effectiveness of a system depends to a great extent on the comfort of the user. If a user is unhappy with the feel of a system it is likely to impact the performance of the user. A simulation allows evaluation of various strategies for the operation of the system. That is, the developer is able to study the effectiveness of the user interface. If the user has problems with some portion of the interaction, the developer will be able to change that portion to assist the user.

2.1 Simulation Systems

A system to develop simulations to provide the support mentioned above can be described by the structure shown in Figure 1. This structure, or architecture, keeps the system to be modeled separate from the user interface to facilitate changes in the architecture. The user interface is able to send inputs to and accept outputs from the model. The environment may also have some effects on the model as is shown in the figure. Notice here that the simulation is actually composed of two models: a model of the system as well as a model of the environment.

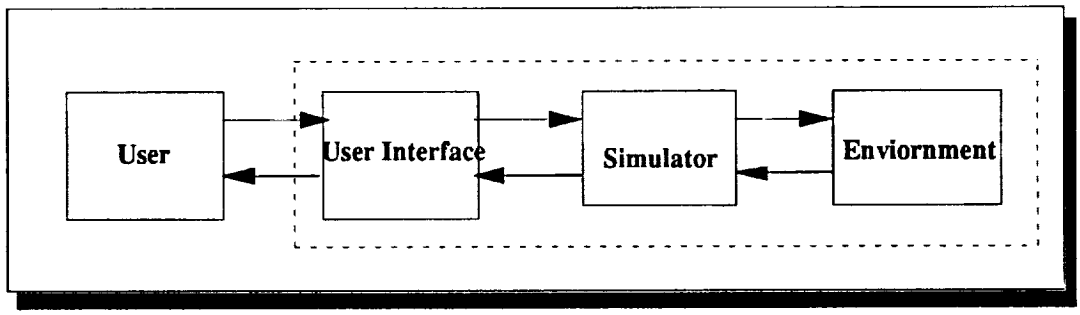

Figure 1 - Simulation System

2.2 Automobile Simulator

The simulation architecture was used in this research to model an automobile. Figure 2 demonstrates how an automobile system could be mapped onto the architecture described

above. The main component of the automobile is the engine which responds to inputs from the driver (e.g., the driver shifts gears or presses the accelerator pedal) and factors from the effects of the environment (e.g., climbing a hill causes a decrease in the speed of the car). The driver changes inputs to obtain desired performance results. If the car slows down climbing a hill, pressing the accelerator closer to the floorboard will counteract the effects of the hill.

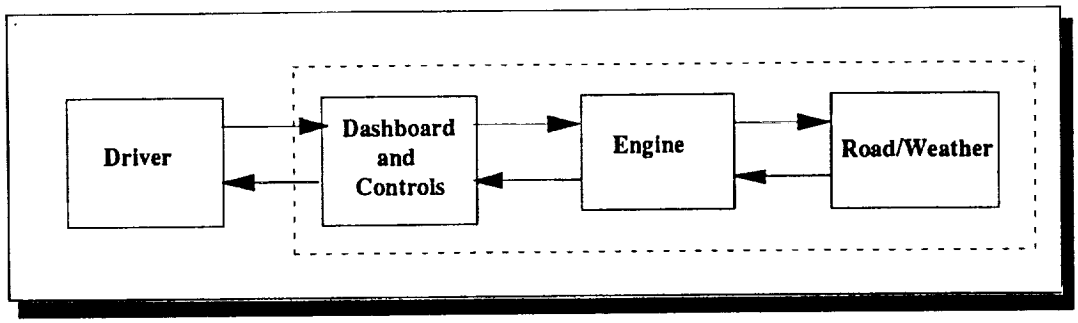

Figure 2 - Automobile Simulator

In developing a simulation, first it is important to decide what level of detail will be needed. For purposes of this research, a low fidelity simulator will be developed. For the automobile simulator the following questions were asked: Should the simulation represent one particular automobile or automobiles in general? Since automobiles are composed of thousands of parts, which of these parts are needed to provide an accurate simulation? It was necessary to list the major components of the automobile and examine each in terms of importance of the operation of the automobile both from the driver's perspective and the engine's.

The simulator which has been developed here has modeled automobiles in general. Various choices have been made and in most cases the decisions have been based on how common a particular feature is. For example, a fuel injection system for delivery of fuel to the engine would affect the performance of the automobile, but fuel injection is not an option for many cars and has not been modeled in the simulator. A survey of general automobile repair books has led to the list of the most important components of automobiles in general. This system remains low-fidelity, however, comments throughout the system description should leave no doubt that higher fidelity systems may be achieved as well.

2.3 Implementation

The software which has been used to implement the model is Percnet. Percnet uses modified Petri nets to model the system. Petri nets have been designed to model systems with interactive and concurrent components. Each component is an object with its own state information. The state of the component determines which actions, if any may be performed and may depend on previous actions of the component. Pictorially, Petri nets show systems of activities and events. Activities describe actions performed by the system. Activities are joined by events that occur during execution. Flow of control, represented by tokens passing through the system, passes from activities to events and vice versa. Before an event can fire, that is, pass control beyond the event at the next activity, all incoming arcs must have tokens. When this occurs, the event places tokens on all outgoing arcs passing control to activities. A sample Petri net is shown in Figure 3.

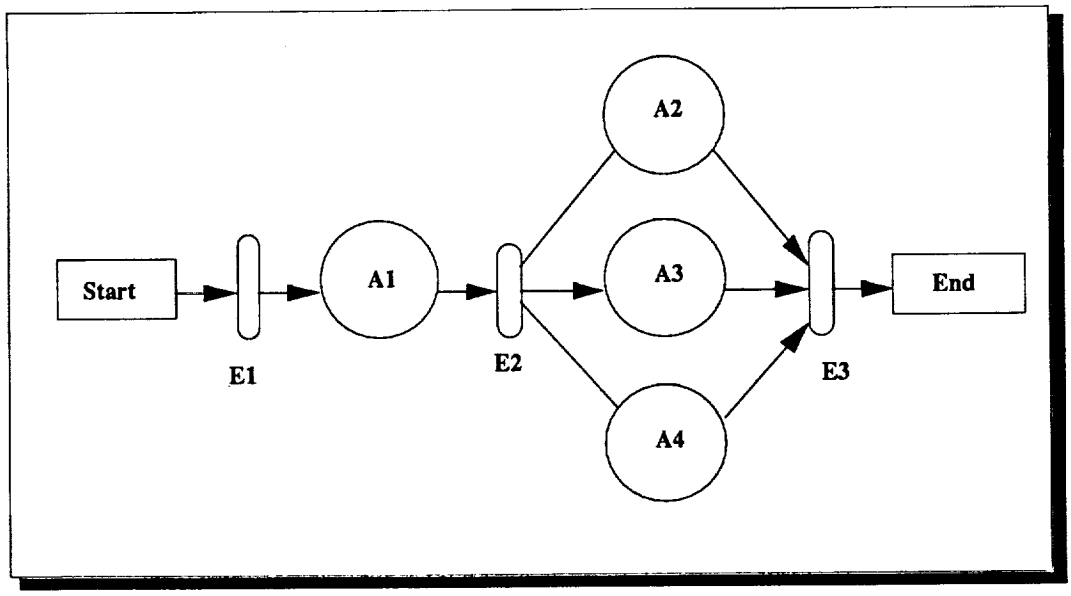

Figure 3 - Sample Petri Net

In this Petri Net, event E1 will occur immediately. One token is passed from E1 to activity A1. Next, A1 passes one token to event E2. E2 then passes tokens to activities A2, A3, and A4, which all occur concurrently. Finally, event E3 may only fire after activities A2, A3, and A4 have occurred.

Modified Petri nets combine a variant of Petri Nets with frames and rules. This combination allows modeling systems in a compact and straightforward way. Through Percnet, rules and functions can be added to the network which allows for more complex flow control. Returning to the example in Figure 3, if event E1 represents the starting of an automobile and the driver is required to be wearing his seat belt before ignition may occur then event E1 may contain a rule:

```
If (seat_belt == on)
then ignition
else do_nothing
```

In addition, the type of action to be performed for each activity may be described by a function or small program. If activity A2 represents the actions performed by the fuel pump in an automobile, those actions may be described by an expression:

```
gasoline = gasoline - (gasoline_consumption_rate)
```

Percnet models are "executable." After defining the system, a developer is allowed to run the simulation. Tokens are depicted flowing through the system. Percnet also provides a means of modeling the systems interactions with the user as well as with the environment. Using *scenarios* events external to the system may be defined. These events include occurrence information - time of occurrence, duration of event, and frequency of occurrence as well as effects of the action on the system. In the automobile system, Percnet's scenarios provided a way to model user and environmental inputs. The user action of bucking a seat belt may be defined as occurring at some particular time. The action of turning the key may occur after the seat belt buckling has occurred. It is important to note that Percnet currently does not

provide synchronization of activities and events with the real time clock. Time units are provided, but no relation is established between these units and actual time. Developers must ensure that time units are consistent throughout the system.

Finally, Percnet provides several simulation analysis options including workload profiles, time-baseed performance profiles, and a means of viewing all simulation data. These options could be useful for evaluation of user's interactions with the system.

Figure 4, shows how Percnet could be used to model an automobile. Variables should be viewed as global variables recorded on a blackboard. Scenarios (user and environment) and the automobile may retrieve or modify information on the blackboard. An example is the calculation of speed. The user chooses a gear and presses the accelerator pedal. The engine calculates the speed based on the gear and the amount the accelerator pedal is depressed. The speed is posted on the blackboard and is fetched by the user interface for display on the dashboard.

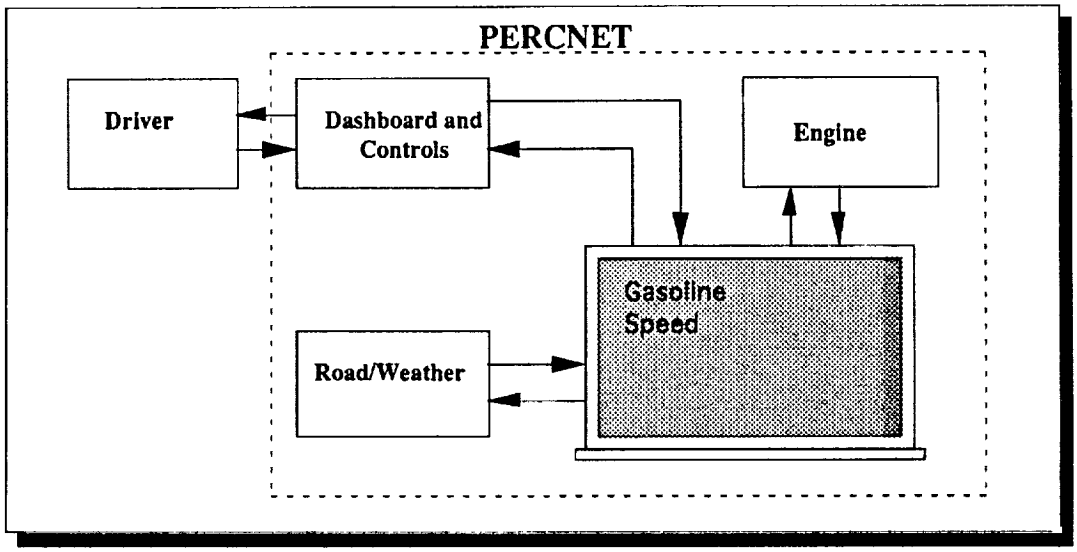

Figure 4 - Percnet Structure for Automobile Example

2.4 Assumptions and Details

As stated previously, the automobile simulator implemented using Percnet, models common automobile operation without focusing on one particular type of automobile; however, some basic assumptions have been made abut the automobile. These are:

10 gallon gas tank
4 quarts oil capacity
6 units coolant capacity
5 gears (reverse, neutral, first, second, third, fourth)
manual transmission (clutch and manual shifting)
driver must be wearing seatbelt before ignition

In modeling the components of the engine, the model focuses on the most important parts and tries to maintain the same level of detail for each. Modeling the complete electrical

system could have included objects representing each wire and sensor, but such a level of detail is not necessary for the purposes of this research. This level of detail would be provided in a high fidelity simulator. Some suggested extensions to the low fidelity simulator which has been developed are provided at the conclusion of this section.

The major components of the engine modeled are:

fuel pump	distributor	battery
oil pump	spark plugs	alternator
water pump	starter	fan

The condition of these components is modeled using a boolean variable indicating that either they are functioning or they are not. The boolean variables are then used as conditions within events occurring during the simulation. For example,

```
if (fuel_pump_ok && (gasoline > 0.0))
then gasoline = gasoline - gasoine_use_constant
else stop_running
```

This means that as long as the fuel pump is working and there is gasoline, the engine consumes gas; otherwise, the engine stops.

Figure 5, shows the top level Petri Net of the automobile simulator. The majority of this net shows the actions of the driver and engine components up to the ignition of the engine. More detail is shown (and will be explained) in subsequent subnetworks of the model.

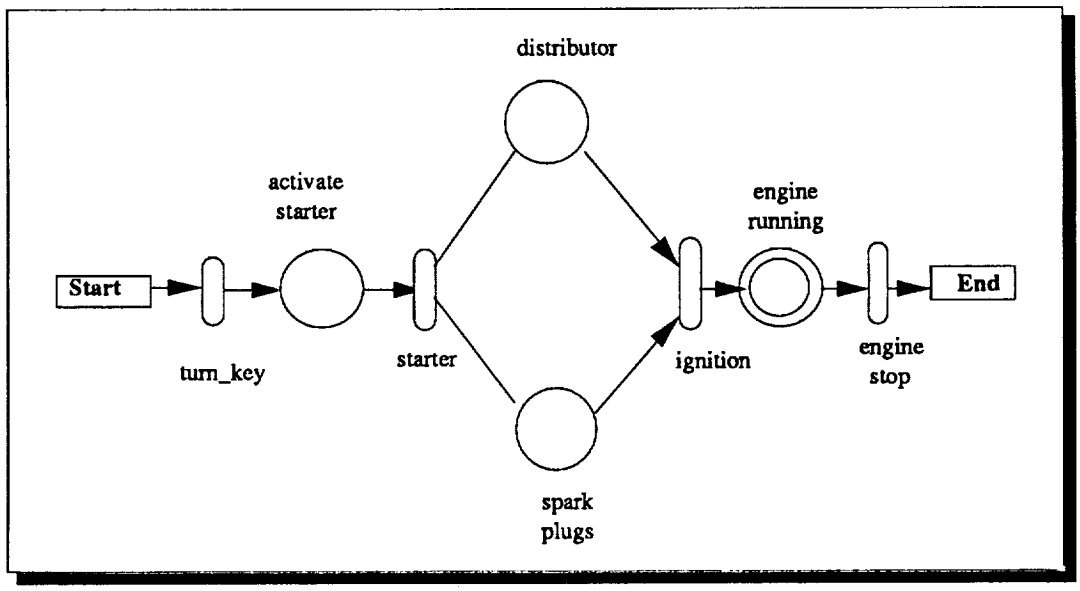

Figure 5 - Top-Level Petri Net of the Automobile Simulator

The starter is the component that is activated by the turning of the key. Before that starter can begin working, however, the key should be turned on, the driver must be wearing his/her seat belt, the car must be in neutral, and the battery must have a sufficient charge to start the starter. In rule form, the pre-condition for activating the starter are:

if (key_on && seatbelt_on && (gear == neutral) && battery_has_charge)
then activate_starter
else stop running

When all three preconditions are true, the starter is activated and control advances to the right in the Petri net.

Once the starter has been activated, it must do its part to start the automobile. The starter allows electricity to flow into the distributor where it is channeled into the spark plugs. As long as the starter is functioning, the distributor and spark plugs are activated. In rule form this is:

if starter_ok
then distributor & spark_plugs
else stop_running

Finally, as long as the spark plugs and distributor are working properly and there is gasoline, the spark from the spark plugs ignites the gasoline mixture in the engine and ignition is achieved.

if (spark_plugs_ok) && (distributor_ok) && (gasoline > 0.0)
then ignition
else stop running

Now that ignition has been accomplished, the engine is running. The concentric circles representing the engine_running activity in Figure 5 indicate that the state is shown in a subnet. This subnet is shown in Figure 6.

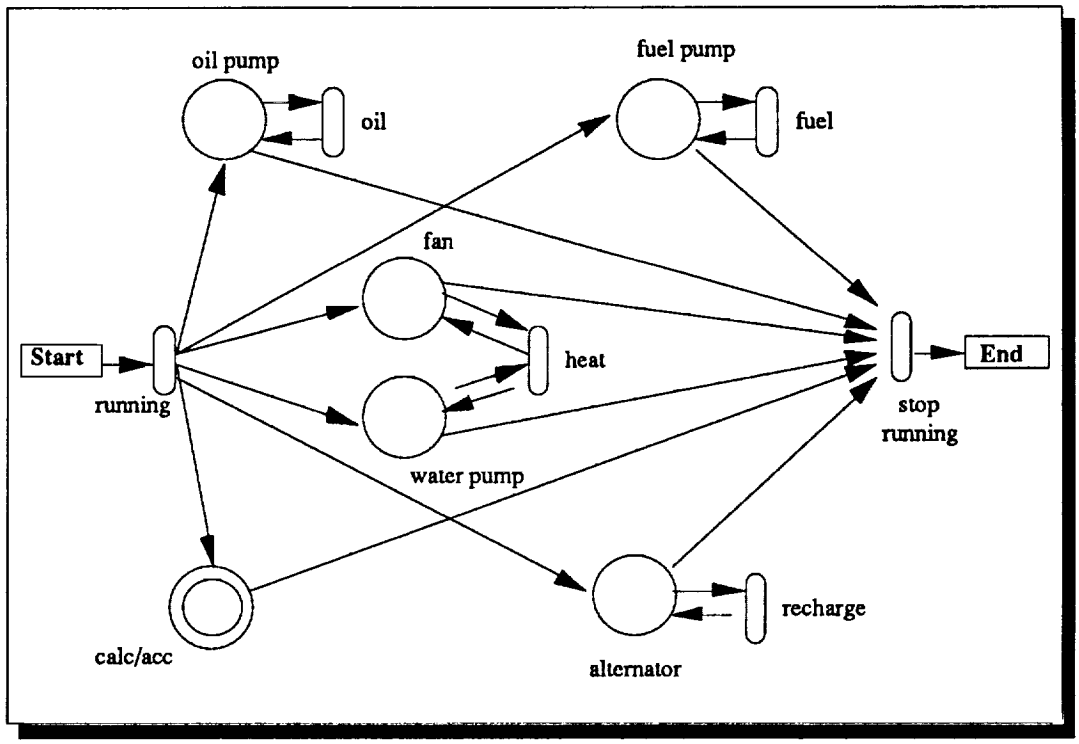

Figure 6 - Engine Running Subnet

The "running" event shown in Figure 6 initializes the rpm value to 1000 (which has been chosen as the value for when the automobile is idling). Next, several events begin to occur concurrently. The automobile begins to consume gasoline, pump oil, maintain the charge on the battery, and regulate the engine temperature. Each of the components of the automobile are examined in more detail in the following sections, and a complete list of rules is presented in Appendix A.

Fuel Pump. As mentioned previously, as long as the fuel pump is functioning and there is gasoline present, the fuel pump pumps fuel into the engine where it is consumed. Actually, what occurs in the engine is that the gasoline is combined with air and forced into the engine where it is compressed and ignited. The ignition of this gasoline/air mixture provides power to the engine. This power is manifested in the turning of the flywheel. The turning of the flywheel is then transferred to the wheels through the transmission system. The speed of the wheels depends on the speed of the engine. Therefore, the rate of fuel consumption increases as the speed and rpms increase. The fuel pump activity samples the current speed and rpms to calculate the amount of fuel consumed by the engine. If at any time the fuel pump ceases to operate or the engine consumes all of the fuel, the engine stops.

Alternator. The battery provides electricity to the engine through the spark plugs (the spark initiates the ignition of the fuel mixture) as well as to the radio and other accessories inside the passenger compartment. To prevent the battery from running down, the alternator recharges the battery. With age, the battery may lose its ability to hold the charge sent from the alternator, but as long as the alternator operates the battery receives a charge.

Water Pump. The ignition (or explosion) occurring over and over in the engine generates extraordinary amounts of heat in the engine. Therefore, the engine has a system to regulate its own temperature. The coolant in the radiator and in various hoses is present to reduce the temperature of the engine. If the coolant does not circulate through the engine, there is no cooling effect. The water pump pumps the coolant through the cooling system (which includes a condenser that cools down the coolant). This pumping which constantly absorbs heat from the engine (via hoses that circulate through the main parts of the engine) and then cools the engine (via coolant cooled by the condenser) helps to keep engine temperature at acceptable levels.

Fan. In addition to the water pump, a small fan in the front of the engine compartment draws air through the grill to provide extra cooling. Unlike the water pump, the fan only operates if the engine temperature rises above a threshold value (typically around 250 degrees). The fan also operates if the air conditioner is running no matter what the engine temperature is.

Oil Pump. Oil lubricates the engine components. Many of the metal components are in constant motion against each other separated by a thin layer of oil. Without oil, the friction of components touching each other would cause the engine to stop and most likely result in serious damage to the engine. The oil pump circulates oil through the engine. If, at any time, the oil pump ceases to operate or all of the oil leaks out of the engine, the engine will stop. The automobile does not actually consume oil at all; the only way for the oil level to drop is for oil to leak out.

Calc/Acc. The subnet "calc/acc" shown in Figure 6 depicts calculations performed by the simulator reflecting the performance of the automobile as well as the effects of some "less essential" features of the automobile. Figure 7 shows the details of this subnet.

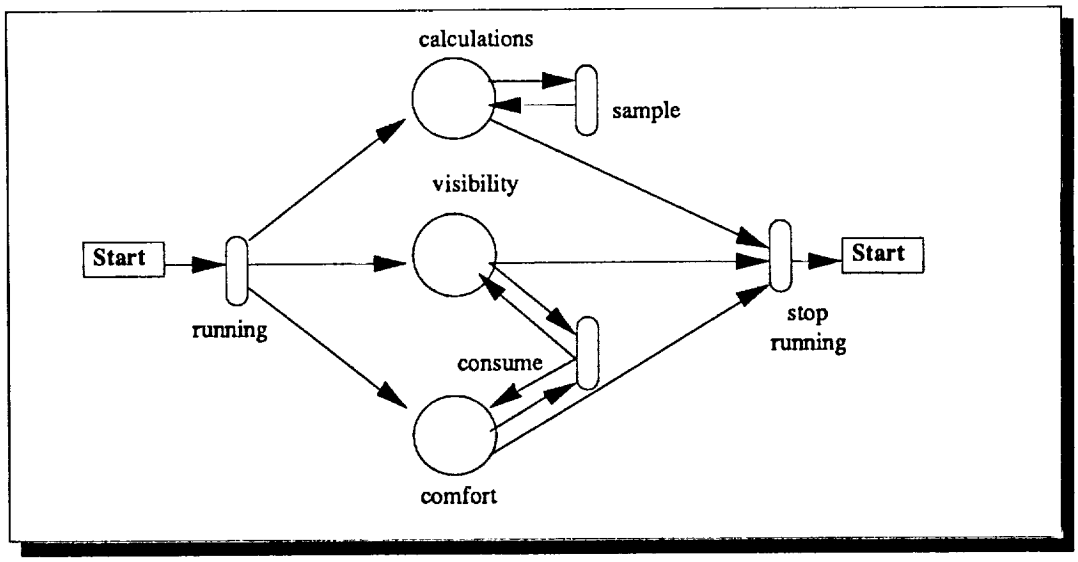

Figure 7 - Calc/Acc Subnet

Calculations. Three values best reflecting the performance of the automobile are computed in the calculations activity - speed, rpm, and engine temperature. Both speed and rpms are dependent on the current gear and the amount of throttle given. The relationship between speed and rpm can be seen in the graph of Figure 8. Notice that the values for each relate to the current gear. The amount of throttle needed to attain a given speed and rpm value in each gear is also shown. In addition, the speed and rpm calculations factor in the effects of the terrain (i.e., uphill and downhill) and the amount of brake the driver is delivering.

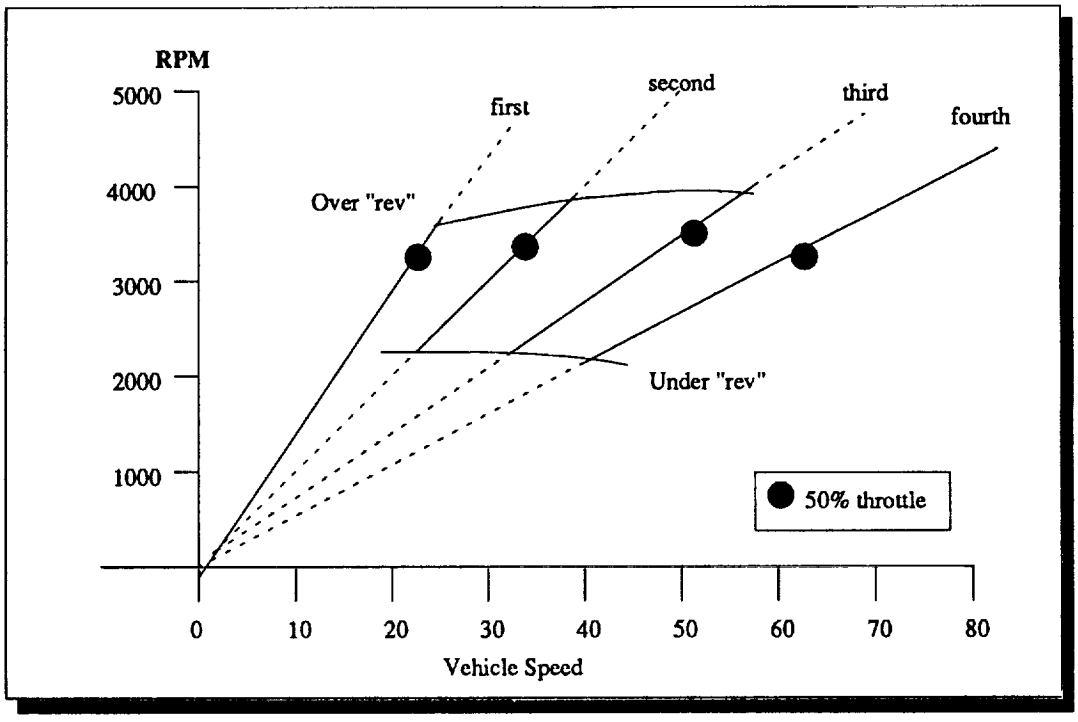

Figure 8 - RPMs vs Speed for each Gear

Engine temperature should remain between 180 and 250 degrees and is dependent on the current speed and rpms. Notice that in this situation there may be several independent nodes in the simulator affecting one value. Engine temperature may be reduced by the water pump and the fan. If the speed and rpms are changing then the engine temperature is constantly changing in the calculations node as well. Such an approach shows how the backboard may be used to effectively model the inner workings of a system (particularly an interactive and concurrent one).

Visibility. Another value returned to the user interface is visibility - a measure of the clarity of the windshield. Earlier the separation of simulator and environment was stressed. Again, it is important to see that visibility is reduced by the environment (from rain, snow, or darkness) and is improved by the automobile's accessories - wipers, defrost, and headlights (low and high beams). Visibility may be viewed as a percentage of a perfectly clear view. One hundred percent visibility would be the value on a clear day with a clean windshield. The blackboard is again used to obtain an accurate simulation of events. The environment may reduce the visibility. If the user takes action to increase visibility (e.g., turning on windshield wipers in the rain or snow or turning on headlights at night) the visibility node increases visibility.

The accessories which improve visibility also drain energy from the battery as they operate. It is a minimal amount while the automobile is operating, but if left on after the engine has stopped running (when the alternator is not able to do its recharging duties), the battery will lose its charge.

Comfort. Several extra accessories are also part of the car and include heater, air conditioner, and radio. The heater and air conditioner change the passenger compartment temperature and use energy from the battery. The radio only consumes the battery's power.

Engine Stop. The Petri Net representing the automobile passes from the ignition portion to the engine running state and remains in the running state until some condition causes the engine to stop running. It is important to see that the system may be modeled in such a way that any from a list of events will pass control from the running state to the stopped state. The conditions are:

```
if ((gasoline == 0.0) || (temperature > 300) || (key == 0) ||
    (rpm < 1200 && gear != 0) || (rpm < 800 && gear == 0) ||
    (!fuel_pump_ok) || (!oil_pump_ok) || (!spark_plug_ok) ||
    (!alternator_ok) || (battery <10 volts))
then engine_stop
```

The engine will stop running if the engine runs out of gas or runs out of oil; the temperature rises above a certain threshold; the key is turned off; the engine stalls (when the automobile is in some gear and the rpms fall below a threshold amount); the battery loses its charge; or the fuel pump, oil pump, spark plugs or alternator fail.

2.5 Simulation

As mentioned previously, actual simulation of the automobile requires development of scenarios which will model user inputs as well as environmental conditions. User inputs consist of throttle, brake, gears, and the accessories mentioned above. One disadvantage of using Percnet to model user actions as that user actions are no longer random. The actions to be performed must be planned in advance including the time in the simulation that the action will occur.

The environment scenarios include weather (rain and snow), brightness (day and night), road conditions (angle of incline or decline), and random failures (e.g., oil leak). Each of these environmental events has an effect on some blackboard value(s). In this way, the environmental conditions are "factored into" the performance calculations.

2.6 Future Enhancements

The next step toward a more complete and effective simulation is to facilitate the communication of a system modeling tool like Percnet with an interface modeling tool like Sammi. If the two could be made to communicate (as shown in Figure 9), the simulation would be much more accurate and informative. Both goals of simulation (analysis of systems and interfaces) could be achieved.

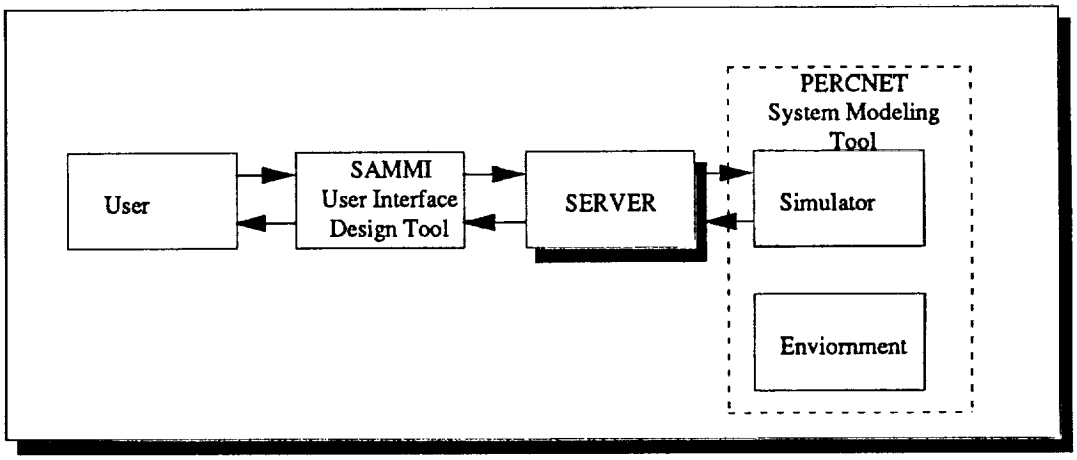

Figure 9 - Ideal Simulation Architecture

Possible extensions to the simulator include expansion of the model and refinement (providing a more detailed description). Expansion could include new features such as cruise control, fuel injection and maintenance assistance. Perhaps a more helpful maintenance assistance feature would provide an on-board computer capable of basic engine troubleshooting (e.g., suggesting to the driver that changing the air or fuel filter could clear up sudden acceleration problems).

Refinement to the simulator would represent each engine component more exactly. New components, including the suspension system, carburetor, and exhaust system would be introduced as well. A truly high-fidelity model could be generated using automotive engineering theory.

Basically, the quality of the simulator is limited only by the developer's knowledge of the system. Any type of interface may be developed and tested using an interface development tool. This tool will be connected to a system modeling tool, such as Percnet providing complete analysis of a system early in development and allowing valuable savings in effort and resources.

3.0 Human-Computer Interface

3.1 Implementation of the Automobile Interface

The user interface for the automobile simulator was implemented using Sammi, a graphical user interface prototyping environment. Sammi combines the functions of a graphical user interface with full network communication support, providing both client/server and peer-to-peer communication options. The format editor of Sammi was used to develop the static automobile display. The display's functionality was tested by executing it within the runtime environment and connecting it to a random server and a test database through the applications programming interface (API). Once the architecture of Percnet is modified to allow interprocess communication, Sammi will communicate with Percnet, sending and receiving messages and commands, through a server written with Sammi's API. The next section describes the details of the automobile interface.

3.2 Detailed Description

This section provides a description of the interface which was developed for the automobile simulator. The description begins with the windshield which is in the top portion of the user interface shown in Figure 10.

Environment. The environmental conditions are shown on the left hand side of the windshield. It is a large rectangle containing five smaller rectangles. The smaller rectangles represent the following conditions: clear, raining, foggy, snowing, day, and night. If it is a clear day, then a representation of the sun shining is shown in the Clear rectangle; if it is a rainy day, then a representation of rain is shown in the Raining rectangle; if it is a foggy day, then a representation of fog is shown in the Foggy rectangle; and if it is snowing, then a representation of snow is shown in the Snowing rectangle. It is possible that a representation of two of the first four rectangles might be shown at the same time. Possible combinations include: Clear and Raining, Raining and Foggy, Foggy and Snowing, and Clear and Snowing. The Day/Night rectangle indicates what time of day it is.

Trip Tick. The trip tick is a pop-up window that gives the directions for the user to follow.

Terrain. The terrain grade can be found on the right side of the windshield. This larger rectangle contains nine bar charts that together represent the slope of the terrain. The bar charts are set up in sequence so that when new terrain heights are reached, the old terrain heights all shift one bar chart to the left. When the automobile begins to approach a hill, for example, the bar chart on the far right would indicate so by rising higher than the bar chart to its immediate left. The closer the automobile got to the hill, the further that representation of the bar would move to the left. In the same manner, if the automobile began to approach a down hill slope, the bar chart to the far right would indicate this by being lower than the bar charts to its left.

Speed Limit. The speed limit can be found on the lower right hand corner of the windshield.

The control panel (the bottom portion of Figure 10) will be described next.

Seat Belt. The seat belt button has a small square that is indented and red in color to indicate that the belt is not presently being worn. When the user clicks on this button, the indentation will raise and turn grey indicating that the seat belt is being worn.

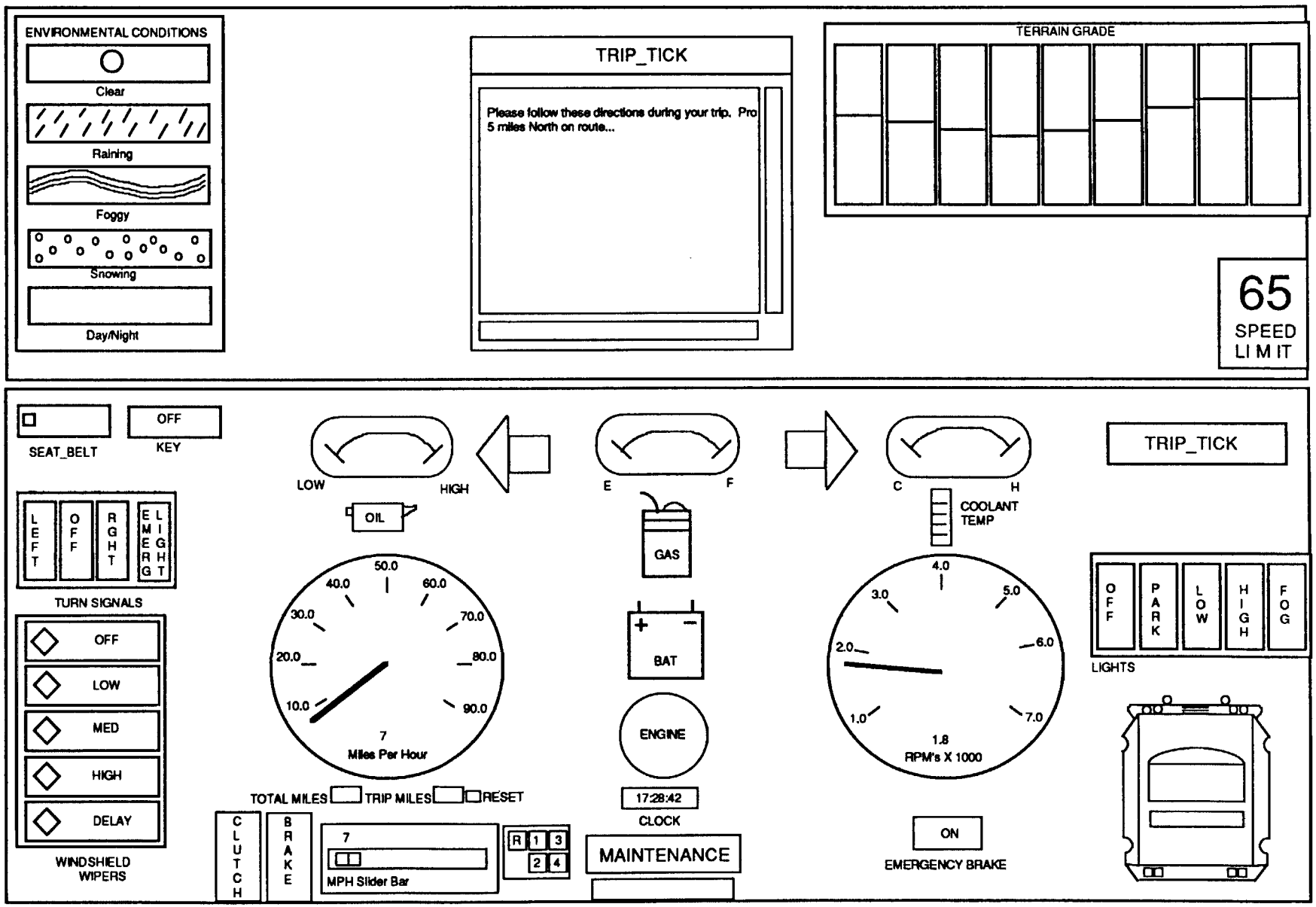

FIGURE 10 - Automobile Interface

Key. The key button is raised and says OFF. When the user clicks on this button, it will indent and say ON. To turn the key back off the user clicks on the button again.

Oil Gauge. The oil gauge gives a graphical representation of the level of oil in the automobile. When the oil level is high, the picture and the word "OIL" beneath the gauge are black. When the oil is at a medium level, the picture and the word are blue. When the oil level is low, the picture and the word flash red.

Turn Signal Indicators. The turn signal indicators (arrows) are connected to the turn signal buttons. The left arrow flashes red when the left turn signal is activated, the right arrow flashes red when the right turn signal is activated, and both flash when the emergency light button is activated.

Fuel Gauge. The fuel gauge is located to the right of the oil gauge and the right turn signal arrow. When the gas level is high, the picture and the word "GAS" in the picture beneath the gas gauge are black. When gas is at a medium level, the picture and the word are blue. When the gas is low, the picture and the word flash red. There is a small square right beneath the gauge (not shown in Figure 10) which shows the numeric value of the gauge at any given time.

Coolant Temperature Gauge. To the right of the gas gauge and the right turn signal arrow is the coolant temperature gauge. Below the gauge is a picture and the words "Coolant Temp."; these items behave in a manner similar to the oil and gas gauges.

Trip Tick. To the right of the gas gauge is the trip tick button. When activated by clicking on it with the left mouse button, the Trip Tick Text window pops up in the middle of the Windshield. The user may cancel the window at any time by clicking on the cancel button at the bottom left of the trip tick text window. After two minutes, the text window automatically cancels itself.

Turn Signal Buttons. Going to the left hand side of the control panel you can find the turn signal buttons. These start out in the off position which is indicated by the indentation of the OFF button. These four buttons (Left, Off, Right, and Emerg. Lights) are set up in such a way that only one may be active at any one time. These buttons are connected to the left and right turn signal indicators (arrows). If the Left button is active, the left arrow flashes; if the Right button is active, the right arrow flashes; and if the Emerg. Lights button is active, both the left and right arrows flash red. The Off button returns the arrows to the original non-flashing state.

Windshield Wiper Control Buttons. Below the turn signal buttons are the windshield wiper control buttons. The control buttons include: Off, Low, Med, High, Delay. These buttons start out with the Off button selected, which is indicated by the indented diamond which is turned red in color. Only one of the buttons may be selected at any one time. If the user clicks on a non-active button, it will become active and the former active one will automatically become inactive, as indicated by the raised diamond which is grey in color.

Speedometer. The large meter in the middle left of the window represents the speed of the car. The meter is currently connected to the MPH Slider Bar and gears at the bottom left hand side of the control panel. The user controls the speed by clicking on the clutch button so that it is active (indented). Then the user selects the gear desired by clicking on that gear. When it indents, the user may release the clutch by clicking on it again so that it is inactive (raised). Then the user may slide the slide bar to the right to gain speed. To slow down, the user clicks on the brake button. To put the automobile in neutral, the user activates the clutch

and then clicks on whichever gear is activated so that all gears are raised. The Speedometer updates the speed every second and displays the speed two ways: by the meter's needle and in numerical form towards the bottom middle of the meter.

Mileage. Between the speedometer and the MPH Slider Bar are two integer displays: the odometer and the trip meter. The odometer registers total distance traveled, and the trip meter is used to measure distance traveled per trip. To the right of the trip meter display is a reset button that will reset the trip miles to zero when it is clicked

Battery Display. To the right of the speedometer and below the gas gauge is the Battery Display. This display will flash red if there is a problem with the battery.

Engine. Below the battery is a circle representing the engine. It will flash red if there is a problem with the engine.

Time. Beneath the engine circle, there is a real time clock display.

Maintenance. Below the clock display is the Maintenance button. When the Maintenance button is clicked, a popup maintenance window appears directly above it in the Control Panel overlapping the Battery, Engine, and Clock displays. The user can select one of three options: Gas, Oil, or Coolant. The user clicks on the "ok" button after they have selected an option, or to cancel the maintenance menu, the user can click on the cancel button. The maintenance menu will automatically disappear if nothing is selected within 60 seconds. If gas is selected, then the gas gauge will show that the gas tank has been filled up; if oil is selected, then the oil gauge shows that the oil has been filled; and if coolant is selected, a small message indicating that coolant has been added is shown in a small text window under the Maintenance button.

Tachometer. The large meter in the middle right of the window represents the engine speed in revolutions per minute (RPMs). Right now, this is not connected in such a way that it would reflect the RPMs based on the speed of the car and the current gear.

Emergency Brake. Beneath the RPM meter is the Emergency Brake button. This button starts out in the On position, as is indicated by the indentation of the button. The user can turn the emergency brake off by clicking on the button. In the off position the button is raised.

Lights. The final part of the Control Panel is the Lights button and the Light Indicator display. There are five buttons for lights (Off, Park, Low, High, Fog) which are located to the right of the RPM meter on the Control Panel. These buttons start out with the Off button selected as indicated by the slight indentation of that button. When the driver selects another button, it will indent and the former indented one will automatically raise. If the Park button is selected, the appropriate parking lights on the picture of the automobile will turn orange indicating that the parking lights are on. If the Low button is selected, the parking lights come on, the outer headlights on the automobile picture turn yellow, and the rear night lights turn red to indicate that these lights are on. If the High button is selected, all the lights that would come on if the the Low button was selected are indicated, and the center highlights turn yellow. If the fog button is selected, all the lights indicated if the High button was selected are indicated, and the fog lights on the front end of the car turn yellow. Selecting the Off button will turn the circles back to the background color of white.

4.0 Evaluation

Evaluation provides a means of objectively assessing a design. It allows the designer/developer to verify user and system performance against requirements, to assess the performance of the user interface dialogue, and to provide data to the iterative design process. Evaluation consists of a static and a dynamic evaluation component. Evaluation of the static component will involve assessing the displays to determine whether or not they are in compliance with standards such as the Flight Human Computer Interface (HCI) standards. The dynamic component (which is addressed in this research) will consist of evaluation of the interaction between the user and the system.

Evaluation of the dynamic component should be accomplished by conducting usability studies. Given a functional prototype and some tasks that can be accomplished on that prototype, the designers should observe how users interact with the prototype to accomplish those tasks in order to identify improvements for the next design iteration. Measurable evaluation parameters should include: time to learn to use the system (i.e., training time measurement - how much time it takes to reach a particular level of proficiency), speed of task performance (or time to complete representative tasks), rates and types of errors made by users, retention over time, and subjective satisfaction. The load demands of the work situation might also need to be assessed, as well as, whether or not there was effective operator planning. For example, whether or not the user carried out the operation, carried out the operation as efficiently as possible, used wrong commands, used too many keystrokes, or received numerous help and error messages. The system should also be evaluated to determine which features of the system were used or not used effectively. For example, the number of times a help or explanation screen was requested will give the designer/developer some indication of which features of the system should be enhanced and which should be eliminated. Analysis of the errors and types of errors encountered will assist in redesign of the screens and dynamic data objects.

There are several techniques which should be considered for the collection of this data. They include embedded evaluation techniques, observation, and subjective satisfaction measures. The embedded evaluation technique includes a capture/playback component and an analysis component. The Capture feature captures a user's session and saves this information to a log. This log can later be "played" back or analyzed. Reports such as the frequency of each error message, menu-item selection, dialog-box appearance, help invocation, form-field usage, etc., are of benefit in order to redesign the user interface.

Specific criteria for evaluation of the interface to the automobile simulator can be found in Appendix B.

5.0 Conclusions

There is no current tool which allows for screen design, simulation, and evaluation. The most advanced tools for rapid prototyping which do not require programming experience are called UIMS (User Interface Management Systems). This term is used to describe software tools that enable designers to create a complete and working user interface without having to program in a traditional programming language. However, the users have to use a programming language to implement additional features such as database search, network communication, or scientific computation. An environment was investigated here which allows for development, simulation, and evaluation of designs.

REFERENCES

Bass, Len and Coutaz, Joelle (1991). Developing Software for the User Interface. Reading, Massachusetts: Addison-Wesley Publishing Company.

Hoyt, Wade A. (1981). Reader's Digest Complete Car Care Manual. New York, New York: Reader's Digest Press.

Payne, James A. (1982). Introduction to Simulation - Programming Techniques and Methods of Analysis. New York, New York: McGraw-Hill, Inc.

Perceptronics User's Manual. (1992). Woodland Hills, California: Perceptronics, Inc.

Peterson, James L. (1981). Petri Net Theory and the Modeling of Systems. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Sammi Users Guide. (1992). Houston, Texas: Kinesix Corporation.

Shneiderman, Ben. (1992). Designing the User Interface: Strategies for Effective Human-Computer Interaction Design. Reading, Massachusetts: Addison-Wesley Publishing Company.

Appendix A - Rule Base

Global Variables

Maintenance
gasoline, oil coolant

Calculations
gas_consumption = function of rpm, speed, gas_use_constant
speed = function of gear throttle, brake, incline
rpm = function of gear, throttle, brake, incline
direction = (reverse, none, forward)
engine temperature = function of rpm, coolant, fan

Driver Inputs
throttle (%)
brake (%)
gear (neutral, reverse, first, second, third, fourth)
key (on/off)
wipers, lights
A/C and setting, heater and setting

Components

fuel_pump_ok	starter_ok
water_pump_ok	distributor_ok
oil_pump_ok	spark_plugs_ok
alternator_ok	fan_on
battery_ok	

Environment
brightness - % (day =0.0%, night = 100%)
frost - %
compartment_temperature, outside_temperature
visibility = function of windsheild debris (e.g., rain or snow), brightness (e.g., day or night)
incline = angle -> (+=uphill, -=downhill)

Ignition
Initial: engine_stop

if key and (battery>10) and seat_belt and (gear == neutral)
then activate_starter
else engine_stop

if activate_starter and starter_ok
then distributor
    spark_plugs
else engine_stop

if distributor_ok and spark_plugs_ok and (gasoline > 0.0)
then ignition
else engine_stop

Engine Running

```
while (!stop_running)

        gas = gas - f(rpm, speed, gas_use_constant)

        if battery < maximum_charge
        then battery - f(battery_capability_to_maintain_charge)

        if temp > 260
        then  fan = on
              temperature = temperature - f(fan_ capability_to_cool)
        else  fan = off

        if (water_pump_ok)
        then  temperature = f(rpm, coolant)
        else  temperature = f(coolant)

        if wipers
        then  battery = battery - 0.00125
              windsheild_debris = windsheild_debris -  70%

        if  low_beams
        then  battery = battery - 0.00125
              if (brightness < 50%)
              then brightness = brightness + 10%

        if  high_beams
        then  battery = battery - 0.00125
              if (brightness < 50%)
              then brightness = brightness + 20%

        if  defrost
        then  battery = battery - 0.00125
              frost = frost - 10%

        if A/C
        then  battery = battery - (0.00125 * setting)
              compartment_temp = compartment_temp - (setting * 10%)

        if  heater
        then  battery = battery - (0.00125 * setting)
              compartment_temp = compartment_temp + (setting * 10%)

        if radio
        then battery = battery - 0.00125

        if  (!fuel_pump) or (gas == 0) or (!oil_pump) or (oil==0) or (!alternator) or
            (temp > 300) or ((rpm < 800) and (gear != neutral)) or (key == off) or
            (!spark_plugs)
        then  stop_running = True

end_while
engine_stop
```

Driver Actions (Inputs)

Add/Reduce Throttle
Add/Reduce Brake
Press clutch
Shift Gears (first, second, third, fourth, neutral, reverse)
Turn Key
Put on seatbelt
Depress emergency brake
Turn on: wipers, lights, turn signal indicators, defrost, A/C, heater, radio, etc.
Perform maintenance actions

Maintenance Actions

Add gas, oil, coolant

Items which need to be Monitored by Driver

Environment
    Day/Night (Brightness)
    Weather (Clear, Raining, Snowing, Foggy)

Terrain
    Incline (Uphill or Downhill)

Speed Limit

Trip Tick

Automobile
    Oil, Gas, Coolant Temperature, Battery, RPMs, Speed, Clock, Milage, &
    Compartment Temperature

Possible Extension - Trouble Monitor (interacts with dashboard)

```
if (temp > 275)
then temp_too_high_message
     if (rpm > 6000)
     then Suggest "Shifting to a higher gear or slowing down"
     else Suggest "Stop car, wait 30 minutes, add coolant"

if (gas < 1 gallon)
then low_on_gas_message

if (oil < 0.25 gallon)
then low_on_oil_message

if (battery < 10 volts)
then battery_low
```

Troubleshooting
Driver would enter symptoms and system would present probable causes

Appendix B - Criteria for Evaluation of the Automobile Interface

Task Model
The task model establishes what the driver should do to obtain certain results.

1. To start the engine
   a. button the seat belt
   b. push the clutch
   c. depress the brake
   d. turn on the key

2. To start driving the car forward
   a. free emergence brake
   b. shift to first gear
   c. free the brake
   d. push accelerator until speed is equal to 7 mph

3. To accelerate the car from 25 mph to 65 mph
   a. push the clutch, change gear to 2 at the speed of 8 to 10 mph, push accelerator
   b. push clutch, change the gear to 3 at the speed of 15 to 25 mph, push accelerator
   c. push clutch, change gear to 4 at speed of 30 to 40 mph, push accelerator to 65 mph

4. To decelerate the car from 65 mph to 7 mph
   a. slide down the accelerator, push clutch, change gear to 3 at the speed of 30 to 40 mph
   b. slide down the accelerator, push clutch, change gear to 2 at the speed of 15 to 25 mph
   c. slide down the accelerator, push clutch, change gear to 1 at the speed of 8 to 10 mph

5. To stop the car
   a. push clutch
   b. push brake

6. To back up the car
   a. stop car
   b. push clutch, change gear to reverse
   c. free clutch, free brake, push accelerator to a speed less than 7 mph

7. To park the car
   a. slow speed to less than 7 mph
   b. stop car
   c. put the park brake on

8. Turn right or left
   a. turn on right or left indicator on at least 100 feet before the turning point
   b. slow speed to limited range until the turning point
   c. turn off the blinker and accelerate when passing the turning point

9. Driving uphill
   a. keep speed in the limited range:
      push accelerator || push clutch, change to lower gear, and push accelerator

10. Driving downhill
    a. keep speed in the limited range:
       slide down accelerator || push clutch and change to higher gear.

11 Environmental Conditions
   a. Night: turn on low headlights
   b. Clear: shut all lights off
   c. Foggy: Turn fog lights on
   d. Raining or Snowing: Turn on wipers

Events which might occur during the simulation

1. Speed limit zone: 30 mph in an urban area, 55 mph highways, & 65 on the interstate

2. Construction warning: Road Construction 1 mile, speed limit 30 mph

3. Sharp turn: speed limit 15 mph

4. Road slippery when wet, ice, and snow

5. Information or Guidance: it tells the driver where the next gas station is, so that the driver can plan to get gas to avoid running out of gas in the middle of the trip; it also tells the driver where to change the path to reach his travel destination.

6. Railroad crossing: car should slow down when approaching the railroad

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

8. Complete stop
   a. red light
   b. stop sign
   c. school bus stop
   d. railroad having red light
   e. emergency car passing

9. Yield: when the driver sees the yield sign, the trip tick should also tell the driver whether or not there is traffic. The driver should slow his speed when approaching the yield sign. If there is no traffic the driver can proceed. Otherwise he/she should make a complete stop.
   a. yield signal
   b. continuous yellow or flashing light
   c. from private drive to public street

10. Overheat: stop for a while.

11. Environmental conditions:

12. Terrain: adjust the gears and accelerator to the desired speed.

13. Brake Failure: driver should turn on emergency light, use the emergency light, ease up on the accelerator.

14. Car skidding: driver should turn on emergency light, not push brake, ease off the accelerator, brake

Design of Evaluation Scenario

The scenario will give the driver different driving conditions. The driver's reaction will be recorded. The driver's response will be compared to the task model