PATH RFP for 2002 - 2003

<table>
<thead>
<tr>
<th><strong>Project Title:</strong></th>
<th>Integration of a Commercial Manual Traffic Simulator with SmartAHS</th>
</tr>
</thead>
</table>
| **Principal Investigators:** | Professor Roberto Horowitz  
Dept. of Mechanical Engineering,  
University of California, Berkeley, CA 94720-1740  
Tel: 510-642-4675.  
Fax: 510-642-6163  
Email: horowitz@me.berkeley.edu |
| **Other Investigators:** | Professor Luis Alvarez  
Instituto de Ingenieria, UNAM  
Apdo. Postal 70-472, 04510 Coyoacán D.F. México  
Tel: +52-55-5622-8089,  
Fax: +52-55-5622-8090  
E-mail: alvar@pumas.iingen.unam.mx |
| **Submitting Unit:** | Institute of Transportation Studies, University of California at Berkeley |
| **Funding Requested:** | Year 1: $ 98,797  
Year 2: $ 101,158 |
| **Start Date:** | July 1, 2002 |
| **End Date:** | June 30, 2004 |
| **Program:** | ATMIS/AVCSS |
| **Topic Number:** | 2.3.2 |
| **Topic Name:** | Integration of Traffic Simulations with SmartAHS |
Summary

This proposal addresses topic 2.3.2 (Integration of Traffic Simulations with SmartAHS) in the PATH RFP for the 2002-2003 funding year. The aim of the project is to design and implement a seamless interface between a manual traffic microsimulator and SmartAHS, an automated traffic simulator. This interface will allow users to analyze the interactions between an Automated Highway System (AHS) and the surrounding non-automated arterials and highways. The proposed simulator will be a helpful tool for researchers studying problems such as the impact of the AHS on the existing traffic infrastructure, and the design of AHS entry and exit stations that support the required high volumes of traffic.

The task of connecting the two simulators is not a trivial one. The two systems under consideration, the manual network and the AHS, are coupled and can influence each other in a variety of ways. For example, traffic exiting the AHS directly influences the utilization of non-automated streets that absorb the outflow, and congestion on these streets could affect the AHS by spilling back over the exit station. The coupling between systems could also be indirect if AHS controllers use measurements from the urban infrastructure to regulate automated flows. We approach this difficulty by identifying the portions of the two systems that are coupled, and by creating redundant representations of these portions in both simulation environments. The approach is described in detail in the Methodology section.

In contrast to previous PATH projects in the area of AHS/manual simulation, the manual model will be selected from existing, well-known commercial simulators. The three products currently under consideration are CORSIM, VISSIM and Paramics. All three have been used extensively in the past by traffic engineers to simulate urban and freeway traffic operations. The candidates were selected in part because of their acceptance in the traffic engineering community, but also because they provide a facility for accessing and overriding their internal model parameters. In Paramics and VISSIM this is done with an Application Programming Interface, or API. In CORSIM with a Run-Time Extension, or RTE. These, as explained in the text of the proposal, will be essential to the design of the software interface.

The AHS will be simulated using SmartAHS. This tool, developed by PATH researchers, includes detailed models of all layers of the AHS control hierarchy, and is currently the most complete environment for testing AHS strategies. Our research group, having been involved in many of the extensions to SmartAHS, including the extended fault-tolerant control structure, link layer controllers, emergency vehicle maneuvers, and the SmartAHS/SmartCAP interface, includes experienced SHIFT programmers who are well-acquainted with the structure of the program. We believe that the combined simulator will be another useful extension to the capabilities of SmartAHS because it will allow future researchers to study many crucial AHS design and deployment issues, such as:

- the efficiency and safety of AHS entry and exit infrastructure and maneuver designs.
- protocols for the transfer of control from the human driver to the AHS, and vice versa.
- the impact of increased AHS flows on the existing network, and the response of the AHS control structure to congestion in surrounding manual arterials.
cooperative AHS/urban controllers.

Important features of the new simulator will be a unified interface through which the user will be able to configure the combined simulation, and a common graphical output window that simultaneously displays all manual and automated vehicles in the network.

The development of the combined simulator will be carried out in several stages. As mentioned earlier, we believe that either VISSIM, Paramics, or CORSIM would be a good candidate for interfacing with SmartAHS, but at this point in time, we cannot unequivocally recommend one simulator over the others. Thus, our first step will be to consult with experts and developers to determine which simulator is best-suited for the interface, in terms of capabilities and required API/RTE development. After we have made the selection, we will design the main components of the interface, namely the manual/automated transitions, input processor, simulation coordination algorithm and data exchange methods. Next, we will implement the interface, concluding with a test of a simplified AHS entry/exit structure in order to demonstrate functionality of the combined simulator. The main deliverables will be the SmartAHS/manual simulator itself, a final report which includes technical and design documentation, and a user’s manual. For more detailed descriptions of the development stages, tasks, and deliverables, please see Sections D.6 and E.
C Background

The combined simulator described in this proposal will build on two existing simulators. This section contains a brief description and comparison of the candidate manual microsimulators (MMSs), with an evaluation of each based in our own experience and on comments received from other researchers. It also includes reviews of a few complementary research projects within the PATH program, focusing primarily on SmartAHS and on work on entry designs.

C.1 Related Research

**Paramics** is a manual traffic microsimulator developed by Quadstone Limited. There are five main Paramics modules: Modeller, Processor, Analyzer, Monitor, and Programmer. The Modeller provides access to the simulation and 3-D visualization capabilities of Paramics. The Processor allows simulation runs to be configured or launched in batch mode. Display of statistical output is handled by the Analyzer. The Monitor keeps track of pollution levels. The Programmer tool enables customization of Paramics through an Application Programming Interface (API). Paramics employs three models for vehicle behavior, one for vehicle following, one for gap acceptance, and another for lane changing; individual driver characteristics are determined by adjusting the values of the “aggression” and “awareness” parameters. Paramics allows detailed modeling of roadway networks. Assorted control elements, such as traffic signals, freeway on-ramp meters, and variable message signs, can be simulated [13].

Favorable features of Paramics include its speed and scalability. Paramics can be scaled to any network size, from individual intersections to expansive roadway networks. The performance of the Paramics software has been documented; for a particular network of 1170 links and 106 zones, a benchmark test of Paramics Modeller 2.0 yielded a performance index (PI) of 17,000 on a Pentium II PC, 333 MHz clock speed, with 128MB main memory, running Windows NT. The performance index is the number of vehicles simulated (nV), multiplied by the number of minutes of simulation time that are possible in one minute (RT). Since PI was shown to be constant for large nV, this means that 10,000 vehicles would be simulated 1.7 times faster than real time for these particular test conditions [1, 17].

The Paramics Programmer allows users to customize aspects of the simulator through an API. In [9], the basic Paramics simulation loop is described as a series of functions that are invoked at each time step. For example, during a given time step, Paramics calls functions to release vehicles into the network, update the positions of existing vehicles, and update the status of traffic signals. The API allows the user to alter or replace any of the functions that comprise the simulation loop. The authors identify three classes of API functions: overload, override, and callback. Specifically,

- **Overload functions** allow interruption of the Paramics simulation, either at user-determined times, or in response to specific events.

- **Override functions** replace data or decision-variable values which will be used during the simulation. Given that Paramics updates the position of a vehicle based on its acceleration, which is determined by the Paramics car-following model, an override function could be used to replace the original acceleration value with one calculated by a user-specified model.
• **Callback functions** provide access to information that may be required by overload or over-ride functions, such as data from vehicles, links, and nodes.

A Dynamic Link Library (DLL) file is generated from the code that uses the APIs. The DLL is loaded into the simulation upon execution of Paramics, thus, DLLs are referred to as “plugins” for the simulation.

A variety of API functions have been developed for Paramics. Several examples are given in [9], including plugins that handle vehicle-actuated signal control, run-time tuning of vehicle model parameters, and customization of performance measures, as well as one that overrides the standard vehicle route choice procedure. In Paramics, vehicles normally select routes based on computed travel cost; the plugin allows routes to be assigned according to turning counts associated with each link, which is the same as the method used by CORSIM. Another plugin enables vehicles with user-specified types and speeds to be released at chosen locations in the network during specified time intervals. This aids in generating initial conditions in Paramics that match real-world data [10].

Interfacing SmartAHS with Paramics is feasible due to the Paramics API. In our estimation, the main advantages that either VISSIM and CORSIM have over Paramics are, respectively, complexity and extensive field validation of the traffic model. See Section D.6 for a comparison of features for each of the three manual simulators.

**CORSIM (CORridor SIMulation)** models traffic microscopically using a combination of two simulators: NETSIM, which handles urban networks, and FRESIM, a freeway modeler. CORSIM was developed and is maintained by the Federal Highway Administration (FHWA). In its latest incarnation, CORSIM is included as part of the Traffic Software Integrated System (TSIS) v.5.0, along with an input processor, TRAFED, and an output animator, TRAFVU; these tools are accessible through TShell, a graphical user interface. CORSIM accommodates a variety of traffic conditions, including moderate flow, heavily congested flow, and behavior caused by incidents. Complex highway network geometries can be modeled, and are represented using a link-node format. CORSIM enables simulation of various traffic control and management strategies, such as fixed-time or actuated signaling at intersections, freeway ramp metering control, and High-Occupancy Vehicle (HOV) lanes. Up to 40,000 vehicles (20,000 in NETSIM and 20,000 in FRESIM) can be simulated during any given time step [12, 15].

In this case, the enabling feature for the SmartAHS/manual simulator would be the CORSIM Run-Time Extension (RTE) package, which allows users to interface their own software with CORSIM. Through the RTE package, users can have their own code activated at preset times during each cycle of the CORSIM simulation, and can access and alter most of the internal CORSIM data structures. An RTE is a DLL that is first registered with TSIS and then called at specified times during execution of CORSIM [15]. An FHWA-funded project, “Evaluation of Real-Time Traffic Adaptive Control System (RT-TRACS) Prototypes” is described in [12]; one result of this project is a successful interface between CORSIM and an external RT-TRACS program. Through the interface, the RT-TRACS program receives vehicle information from CORSIM, then determines appropriate signal states and phase changes. The control decisions are passed back to CORSIM, which updates the states of its vehicles accordingly. Measurements of delay, travel time, and intersection service condition were collected on Reston Parkway in northern Virginia, both before and after the installation of the RT-TRACS prototype. The CORSIM simulation results matched well
with the field data.

CORSIM was selected as a candidate for the combined simulator due to its flexibility (via the RTE), widespread use, and validation against real-world data.

**VISSIM** (VISual SIMulation) is another microsimulation package, developed in Germany by PTV AG and distributed in the US by ITC (Innovative Transportation Concepts Inc.) [16]. Its original purpose was as an environment for solving various problems in urban traffic planning and management, but has recently been extended to reproduce freeway driving behaviors. The VISSIM simulation software consists primarily of two interacting modules: the traffic flow module and the traffic control module. The traffic flow module updates the position and velocity of each vehicle in the network according to the psycho-physical behavioral model developed by Wiedmann (1974), which replicates traffic dynamics by separately considering the driver’s perception and the physical capabilities of the vehicle. The outputs of the traffic flow module are detector calls. These are used by the traffic control module to determine the signal status for the network. The signal control logic used by this module can be fixed-time or traffic responsive, and is defined by the user with a programmable signal state generator (VAP). VISSIM provides a variety of text-based output files which are described in [20].

VISSIM developers claim that the simulator is capable of running a network containing about 80,000 vehicles in real time. This benchmark was obtained on a Pentium III 1000 MHz computer, with a simulation time interval of 1 sec, and with animation and signal control functions deactivated. Our research group has experience working with VISSIM and has in the past calibrated its model parameters to approximate freeway behaviors. We have also used VAP, VISSIM’s pseudo-language for coding signal control logics, to implement the ALINEA ramp metering algorithm.

The efficiency and traffic modeling capabilities of VISSIM, in addition to our previous experience working with the simulator, have lead us to consider it a good candidate for the project. Thomas Bauer, the president of ITC, has expressed interest in working with our group to develop the necessary APIs for the SmartAHS/manual interface.

**Comparisons of Manual Microsimulators**

In [4], VISSIM and CORSIM are compared in terms of their attributes, and also with respect to their performance in evaluating a set of design alternatives for a specified roadway network. VISSIM and CORSIM were found to differ significantly in their approaches to network modeling, car-following logic, gap acceptance modeling, signal modeling, animation features, and control over output data selection. A study was conducted on a network that includes a section of State Route (SR) 519 in downtown Seattle. Two classes of design alternatives, one involving a three-lane configuration and another involving a two-lane configuration for SR-519, were tested in each simulator. Care was taken to ensure that the same demands, control assumptions, and highway geometries were used in both VISSIM and CORSIM. For the same input conditions, travel times (both route-specific and system-wide) were found to differ between the simulation environments. System travel time was obtained by summing travel times over several routes in the network. Although the system travel times derived from CORSIM were on average 9% higher than those of VISSIM, the simulators were consistent in predicting that the three-lane scenarios would result in savings of about 10 minutes over the two-lane scenarios. The authors concluded that both CORSIM and VISSIM proved adequate for the SR-519 design evaluation, but that simulator selection should
be carried out on a case-by-case basis. They do not categorically recommend one simulator over the other.

Similar studies comparing Paramics with VISSIM or CORSIM were not found. The main deliverable of Task 1.1 of this project will be a comparative study of the three simulators, citing pros and cons and justifying our final selection.

C.2 Complementary Research in PATH

C.2.1 AHS Simulation

SHIFT / SmartAHS:
SmartAHS is the primary tool used by PATH researchers to simulate and evaluate AHS operations. Its basic version includes complete models of the coordination, regulation and physical layers of the AHS, as well as components for creating simple highway geometries [6, 11]. A number of PATH projects have extended SmartAHS capabilities to include link layer control strategies, degraded mode controllers, emergency vehicles, fault detection and handling algorithms, etc.

SmartAHS is written in the SHIFT programming language, also developed at PATH specifically for simulating highly complex hybrid systems such as the AHS. Under MOUs 311, 312, 373 and 383/4208, our research group has participated in extending the SmartAHS model, and has become very familiar both with the SHIFT language and the SmartAHS code (see Section F).

Two features of SHIFT are relevant to this project. First, a new version for PC platforms was released, and is available from [18]. Because most human driver models are PC-based, an important step in this project will be to transfer our SmartAHS code to the new PC-SHIFT. The other important feature is the SHIFT Foreign Function Interface (FFI), which enables external C functions to be called from within a SHIFT program, thus allowing users access and modify internal SHIFT variable values through customized code [2]. For more information on SmartAHS and SHIFT see [19, 18].

MOU 383/4208: Development of integrated meso/microscale traffic simulation software for testing fault detection and handling in AHS (R. Horowitz)
The objective of MOU 383/4208 is to integrate the PATH micro-simulation (SmartAHS) and meso-simulation (SmartCAP) software packages into a coherent programming platform for efficient simulations of a large scale AHS. The resulting integrated package will be capable of simulating vehicle faults at the micro level, and the impact of faults on the overall AHS response at the macro level. Because the final product of MOU 383/4208 will be an extended version of SmartAHS, it could be combined with the manual/AHS simulator proposed here, which would enable large scale simulation of the AHS and surrounding non-automated arterials.

C.2.2 Manual Traffic Simulators

MOU 369/4222: Human Driver Model Development (Delphine Delorme, Jim Misener)
The work of MOU 369/4222 is closely related to this proposal. Both share the common objective of building a simulator that simultaneously handles driver-controlled and automated traffics. The approach of MOU 369/4222 however has been to implement a cognitive model of human drivers
in SmartAHS. The selected model, COSMODRIVE, is well-suited for SHIFT because, similarly to the automated model, it is hybrid and highly complex. COSMODRIVE is composed of seven interacting modules, which combine tactical decision logics (discrete dynamics) with perceptual and environmental variables (continuous dynamics). In contrast to our project, MOU 369/4222 also focuses on calibrating the human driver model with real-world data, and using it to gain insight into the task of driving.

**MOU 359: Simulation of ITS on the Irvine FOT Area Using The Paramics Scalable Microscopic Traffic Simulator (Baher Abdulhai)**

This MOU had two main goals. One was to create a checklist of requirements with which to evaluate the usefulness of a given traffic simulator as a tool for testing ITS strategies. The requirements of the list, as reported in [3], fall into four categories: supply/control, demand/behavior, environmental aspects, and performance. The second objective of the research was to apply the checklist to Paramics 1.5, and to attempt to calibrate it using real data collected from a section of the I-405 freeway in Irvine. The findings of the project with respect to Paramics were that “it offers two very important and unprecedented features: high performance and scalability”. However, a series of recommendations to Paramics developers was presented that includes allowing the user to trace the position and velocity of a set of selected vehicles. This is a necessary capability for our research and could be implemented with an API. MOU 359 complements our project by providing a partial criteria for selecting a suitable manual simulator.

Below is a partial list of past and current projects within PATH dedicated to the development and use of the Paramics traffic simulator. Detailed information is not included due to the lack of available documentation for these projects.

**MOU 4106: Bay Area Simulation and Ramp Metering Study (Joy W. Dahlgren, Yonnel Gardes)**

This project proposes to develop a simulation method that will be used to evaluate the effects of new and existing traffic management strategies. The Paramics model is being applied to the I-680 freeway to investigate alternative strategies, including ramp metering.

**MOU 4121: Implementations to the Demand Estimation and Subsection Analysis of the Microscopic Traffic Simulator-Paramics (Reinaldo Garcia)**

**MOU 4130: Dynamic OD Demand Estimation and Subsection Analysis of the Microscopic Traffic Simulator - Paramics (Reinaldo Garcia)**

**MOU 4142: Large Scale Traffic Simulation Through Parallel Computing (Henry Liu)**

**MOU 4143: Real-time Traffic Information Estimation Through On-line Simulation and Hybrid Data Fusion System for ATMS Applications (L. Chu)**
C.2.3 AHS interface design

The following paragraphs give a detailed description of an AHS entrance design and entry maneuver presented by D. Godbole in [5, 8]. The described configuration is the simplest of four possibilities suggested by Dr. Godbole, and is presented in detail for two reasons: first, it is used throughout the rest of the proposal as an example to illustrate certain aspects of the combined simulator, and second, its simplicity makes it a good candidate for the software testing phase of this project (Section E, Task 4).

**Work on Entry and Exit Maneuvers by D. Godbole**

All the AHS entry and exit designs presented in [5, 8] assume that manual and automated vehicles are segregated. In this proposal we focus on the entry configuration which requires the least modification to an existing highway facility.

The configuration is shown in Figure 1. As can be seen in the figure, this highway configuration consists of one automated lane (AL), where only automated vehicles are allowed, one transition lane (TL), for manual and automated vehicles, and two manual lanes (ML), for manual vehicles only. It should be noted that entry into and exit from the transition lane is restricted to specific locations (gates) by means of barriers, as shown in Figure 1. The figure also shows a check-in station inside the TL where the transition from manual to automated control takes place.

In Figure 1, manually controlled vehicles enter from the ML to the leftmost portion of the TL through a gate. The section of the TL where the AHS capabilities of the vehicle are tested is called the check-in station. In this design, vehicles pass through the check-in station and make the transition to automated control without stopping. Manual vehicles that are unsuccessful in making the transition return to the ML through a gate after the check-in station. Automated vehicles exiting the check-in station then proceed to a queue at the stop sign and await entry to the AL. The stop sign controller uses measurements from a road occupancy sensor on the AL to determine when space is available for new vehicles to enter. When a gap is found, the automated vehicle initiates an entry maneuver, either as a free-agent or by forming a pre-platoon.

A pre-platoon may enter the AL either as a separate platoon or behind a target AL platoon. In the latter case, a communication protocol is initiated between the lead vehicle of the pre-platoon and the target platoon. If the target platoon is not engaged in any other maneuver, the pre-platoon will initiate an entry maneuver in which it accelerates until it aligns itself with the target platoon.
at an entry gate, and then changes lanes into the AL. The entry maneuver concludes when the pre-platoon merges with the target platoon.

MOU 386 / MOU 4216: The AHS/Street interface effects of capacity concentrations on system performance (Randolph Hall)

The objective of the project is to determine how the interface between the AHS and the arterial street network could be designed to accommodate the high volume of AHS traffic, and evaluate ways to maximize the benefits that accrue from added capacity. The focus is on how to accommodate concentrations of capacity within narrow corridors or a limited number of interchanges, and its implications on the surrounding street system.

C.3 Problem Statement

One of the central arguments for full automation, put forth by Varaiya in [21], is the dramatic increase in the total capacity of the highway that would result from the platooning concept. In his article, Varaiya states that by organizing vehicles into 20-vehicle platoons, a four-fold increase in throughput is possible. This is an astounding estimate that has driven much of the subsequent research at PATH towards designing the necessary hardware and software to implement automated platooning. However, it is also true that these potential gains can only be realized if the increase in freeway capacity is accompanied by a comparable increase in throughput at the points of entry and exit to the AHS. Godbole makes this point in his thesis [8]:

"AHS entrances are narrow veins that feed the wide arteries of the automated lanes. If those veins get constricted or if they are too few in number, the arteries will be starved, and the AHS capacity will be underutilized."

Entries and exits are thus an important part of the AHS that so far have received relatively little attention, due in part to the absence of adequate methods for analyzing the effects of different entry and exit configurations. However, even if adequate entries and exits are designed, the usefulness of the AHS can only be assessed in terms of its effect on the entire system and on the travel times experienced by individual drivers. An AHS that substantially increases the discharge of a freeway, but as a consequence creates congestion in the surrounding arterials, cannot be considered an improvement.

We can therefore identify two important areas of investigation that will be aided by the introduction of an integrated simulation environment:

- Entry/exit designs. According to [8], the simple entrance configuration of Figure 1 leads to a 25% increase in the total capacity of the four-lane facility (8000 to 10000 veh/hr). Assuming that a typical onramp has a flow of 600 veh/hr, and that the 25% increase in capacity leads to a 25% increase in utilization, a simple calculation reveals that each such onramp will contribute 450 veh/hr to the automated lane. Considering that congestion usually forms near these onramps, where lane changes increase, the location of the check-in stations with respect to the existing highway onramps will be an important consideration. Similarly, the placement of check-out stations will depend on the behavior of vehicles near highway off-ramps.
• AHS deployment. The benefit of introducing automation at a particular location will depend on a wide variety of factors. These include the driving habits of the users and non-users and, as stated above, on the effect of the AHS on the existing network. These effects are very complicated and can only be adequately studied with computer simulations.

C.4 Proposed Solution

The final product of this project will be a composite environment that will permit realistic simulations of the combined AHS and manual traffic systems, and of the interactions that occur at the critical points of entry and exit. The new simulator will be built by combining SmartAHS with one of the three simulators described in Section C.1.

The entry configuration described in Section C.2.3 can be used to illustrate what is meant by the term interaction and some of the difficulties that arise in the design of a combined simulator. Consider an idealized situation where all manual vehicles that enter the check-in station are AHS-ready, and make the transition to automated successfully. In this case, there will never be a mixed flow of automated and manual vehicles. That is, a boundary can be defined, upstream of which all vehicles are manually driven, and downstream of which all are automated. However, an automated vehicle that has just crossed this boundary still interacts with the manual vehicle behind it, in the sense that it still affects its motion. Thus, a vehicle interacts with any other vehicle that is within its range of perception or influence, by radio or radar for automated vehicles, and by human sight for manual vehicles. The situation is more complicated if we allow the AHS control system to reject manual vehicles that are not AHS-ready. In this case, a stream of AHS vehicles interspersed with rejected manual vehicles may exist in the check-in station. Here the interaction is both perceptual and physical, since automated and manual vehicles can occupy the same physical space on the highway (i.e. collisions are possible). The approach described in the Methodology section relies on the distinction between these two types of interactions to define regions of the network where coupling between the AHS and the manual network occurs.

The development of the composite simulation tool will evolve on three fronts. First, a consistent methodology for connecting the two simulation environments without disrupting either of the underlying dynamic models is developed. For example, a manual vehicle following another manual vehicle should behave in exactly the same way as if it were following an automated vehicle; in addition, it should not be affected if the leader transitions to the automated state, as would happen when passing through the check-in station. The issues arising here are modeling issues and are addressed in Sections D.1, D.2, and D.3.

The second area of development will be in creating a consistent and efficient user interface for the combined simulator. The goals here are: (1) to design a method for managing inputs to the combined simulator, (2) to provide a unified graphical interface that simultaneously represents all vehicles in the network, and (3) to provide an interface for users to issue simple simulation control commands (e.g. “play”, “pause”). Approaches to these problems are discussed in Section D.4.

Finally, the third area of development will focus on establishing communication between the two simulators, and on coordinating their respective tasks. The difficulty here is that all manual simulators under consideration are PC/Windows based, and closed-sourced. The software interface will therefore rely heavily on the facilities provided by each software for modifying their contents, namely APIs and RTEs. Strategies for implementing the interface are discussed in Section D.5.
C.5 Impact on Existing PATH Projects

- SHIFT / SmartAHS.
The new simulator will allow SmartAHS users to create much more realistic situations than previously possible. Currently in SmartAHS, boundary conditions (i.e. inlet flows) are deterministic and set manually. Constraints on outgoing flows are not supported. By embedding the SmartAHS simulation in a larger and much more versatile manual microsimulator, a variety of scenarios can be recreated by using manually simulated boundary conditions. Another important benefit of embedding SmartAHS in the manual simulator will be the wealth of graphic and analysis tools that will become available to the user.

- MOU 383/4208:
Meso/micro level simulation of the AHS is an example of a project that will benefit directly from the new simulator for the reasons stated above.

- Paramics:
In the case that Paramics is selected as the manual simulator, our project will contribute new APIs which may be useful to other researchers using this simulation environment. Current projects involving Paramics include MOUs 4106, 4121, 4130, 4142 and 4143.

C.6 Project Outcomes and Further Research

The final product of this project will be a working interface between SmartAHS and a manual microsimulation environment, suitable for testing interactions between an AHS and the surrounding manual traffic network. Our objective is to construct the simulator in a manner that is independent of any particular AHS design. To demonstrate functionality of the interface, including the software domain definitions, automated/manual transitions, user interface, simulation coordination, and data exchange algorithms, we will test the software using simplified entry/exit scenarios. For example, we could use a version of the design described in D. Godbole’s thesis [8], modified so that it lacks complex maneuver protocols and control laws (such as the automated-vehicle entry maneuver and platoon lane change), since these components not necessary for validating the SmartAHS/MMS interface. To simulate in detail any of the entry/exit maneuvers referenced in Section C.2.3, further enhancements to SmartAHS will be required and are considered future work.

D Methodology

In this section we describe the overall structure of the combined simulator, and features within each of the mentioned commercial simulators that will enable us to realize the design.

One of the main considerations in developing the software interface will be flexibility; the aim of the project is not to implement a particular entry/exit infrastructure design, but rather to create an environment with which to test and compare a variety of possibilities. With this in mind, we begin the section with a generalized definition of regions within the network that will be controlled by the automated and non-automated simulation tools. This classification is flexible enough to accommodate many situations in which manual and automated vehicle systems interact, including
the entry and exit designs of [8] (described in Section C.2.3). The classification will be used to define the domains of the two simulators, and the minimum amount information that must be exchanged between the two.

D.1 Software domains

The software domain of each simulator is the region of the network that it manages. Thus the software domain of SmartAHS must contain the entire AHS and the domain of the manual microsimulator (denoted MMS in the remainder of this proposal) must contain the entire non-automated network. However, this is not sufficient because it does not consider the coupling of the two systems that occurs at the boundaries between the AHS and the urban network. In this section we present a methodology for defining software domains that reduces the computational effort and the amount of information exchanged between simulators, without affecting the results of the simulation.

The network, including all urban streets and highways under consideration, is divided into four types of regions: manual regions (MAN), automated regions (AUT), regions of coexistence (CO), and buffer regions (BUF). These are defined as follows:

1. MAN regions are portions of the network used only by manually driven vehicles.
2. AUT regions are used only by automated vehicles.
3. CO regions can be shared (physically) by both types of vehicles. The check in station in Figure 1 is an example of a region of coexistence.
4. BUF regions are sections of length $\Delta$, where $\Delta$ is the maximum communication or sensor range of any automated or manual vehicle. That is, the maximum over all radar ranges, LAN communication device ranges, and human perceptual thresholds. The perceived environment of any vehicle is thus contained in a circle of radius $\Delta$ around the vehicle.
5. BUF regions are segregated, and of two types: ABUF and MBUF. Only manual vehicles are allowed in MBUF regions, and only automated in ABUF regions. These buffer regions will be used to isolate the regions of coexistence from MAN and AUT regions, so that all vehicles in CO regions are outside the range of influence of all vehicles in AUT and MAN regions, and vice-versa.

Referring to the terminology used in the Problem Statement, CO regions are areas of physical interaction between vehicles, whereas BUF regions are areas of perceptual but not physical interaction. The classification is schematically depicted in Figure 2.

Connections between regions (represented in Figure 2 with arrows) are established when traffic flows between them. In order to isolate MAN and AUT regions from CO regions, only the following four connection sequences are allowed:
MAN → MBUF → CO
AUT → ABUF → CO
CO → MBUF → MAN
CO → ABUF → AUT.  \hspace{1cm} (1)

By using only allowed connection configurations (i.e. by placing the appropriate buffers between MAN or AUT regions and CO regions), it is assured that the motion of vehicles in MAN and AUT regions is not influenced by vehicles in CO regions. Connections not included in the list can usually be reduced to one of the allowed configurations. These specifications should be observed by the user when coding the network in the MMS input interface. One of the first steps prior to the simulation will be to check that none of the rules have been violated, since this could cause a failure in the program. For example, a CO → AUT connection, without the required intermediate buffer, could cause a manual vehicle in the CO to not see an automated vehicle that is within its range of vision, because it is in the AUT. If a violation in the rules is found, the user will be prompted and asked to revise the connection sequence.

We assume that definitions 1 and 2 (MAN and AUT regions) are enforced by some means within the individual environments: e.g. physical barriers, traffic rules, AHS controllers. They are required, as explained below, to ensure consistency of the manual and automated models across the region boundaries.

In our entry configuration example (Figure 1), described in Section C.2.3, a legal (although conservative) partition can be defined as shown in Figure 3, and described below:
• AUT-I: Consists of the automated lane and the acceleration lane, after the stop sign. The requirement for AUT regions (definition 2) is satisfied if the manual simulator can prevent all non-automated vehicles from passing point A in the transition lane.

• MAN-I: All manual lanes and a portion of the transition lane upstream of the check-in station.

• CO-I: The regions where automated and non-automated vehicles coexist, consisting of the transition lane between the check station and point A.

• MBUF-I: A distance $\Delta$ upstream of the check station.

• MBUF-II: A gap of length $2\Delta$ at the manual re-entry gate.

• ABUF-I: The stretch of the transition lane between point A and the stop sign.

The example is illustrated schematically in Figure 4.

The motivation behind this segmentation of the network is that it provides a consistent methodology for defining the domains of each of the two simulators: as is shown in Figure 2, the manual simulator is responsible for the union of all MAN, BUF and CO regions, while the automated simulator (SmartAHS) manages the union of all AUT, BUF and CO regions. The intersection of the two domains (i.e. all BUF and CO regions) represents information that must be shared, and is denoted the INT region (intersecting region).

\[
\text{SmartAHS domain} = \text{AUT} \cup \text{CO} \cup \text{BUF} \\
\text{MMS domain} = \text{MAN} \cup \text{CO} \cup \text{BUF} \\
\text{INT} = \text{CO} \cup \text{BUF} = \text{SmartAHS domain} \cap \text{MMS domain} \tag{2}
\]

![Figure 4: Simulation regions](image)

By following the rules previously indicated for defining the software domains, we can avoid violations in the car-following and lane change dynamics arising from vehicles not seeing everyone in their range of perception/communication. For example, it eliminates the possibility of a lead vehicle disrupting the car following dynamics of its trailing vehicle by crossing a boundary of the simulation domain of its follower. To see this, suppose that the lead vehicle is automated and the trailing vehicle is manually driven. Since the leader is crossing a simulation boundary, the only possibility is that it is going from an ABUF region to an AUT, thereby leaving the domain of the
manual simulator (Definition 1 removes the possibility of the automated vehicle leaving the domain of the automated simulator since it is not allowed into MAN regions). Because the trailing manual vehicle is not allowed into the ABUF region, the vehicles must be separated by a distance of at least $\Delta$. The manual vehicle therefore cannot see the automated vehicle and is unaffected by its removal from the manual simulation environment. The opposite case, where the leader is manual and the follower is automated, is analogous. Cases in which both are of the same type do not present a problem because they are represented in the same simulation domain.

The next two sections focus on the regions of the highway where automated and manual vehicles interact. This, as stated above, is the intersection of the two simulation domains given by the union of BUF and CO regions. The traffic state in these portions of the highway must be represented in both simulation environments. Two important issues are addressed: how vehicles in one simulator are seen by the other, and how vehicles make the transition from automated to manual, and vice-versa.

**D.2 Vehicle representations in the INT**

Defining the software domains as indicated in the previous section guarantees that the perceived environment of every automated vehicle will consist of other automated vehicles and of manual vehicles in the intersection of the software domains. For convenience, we have denoted this region, composed of CO and BUF regions, as the INT region (intersecting region). Since any given automated vehicle can see all other automated vehicles in its range of perception (since they all reside in SmartAHS), what remains is to represent the manual vehicles of the INT in SmartAHS. An analogous task pertains to the representation of automated vehicles in the MMS.

The representation of manual vehicles in the SmartAHS environment is embodied by a SHIFT object that we call the *ghost vehicle*. This component is essentially a simplified AHS vehicle type, with all AHS related controllers and communication devices removed. However, it will be capable of performing four basic functions: (1) to provide a *radar target* for automated vehicles, (2) to reject any attempts by automated vehicles to establish radio communications, (3) to carry any driver behavior and vehicle information that may be needed by the control system to assess whether it is fit to enter the AHS, and (4) to carry a vehicle ID with which to relate back to its counterpart in the MMS. The position and velocity of the ghost vehicles will not be computed by SmartAHS, but directly set by the manual simulator through the combination of an API and the SHIFT foreign function interface. By using the simplified type, instead of the standard AHS type, we avoid a large amount of unnecessary computation. Details on how the ghost vehicle state will be overridden are given in Section D.3.

The approach on the manual side is a bit different. In contrast to SmartAHS vehicles, simulated manual vehicles are assumed to be computationally cheap. The effort of creating a simplified manual vehicle type, analogous to the ghost vehicle of SmartAHS, is therefore not necessarily justified by the resulting gain in simulation speed. In addition, lack of access to the MMS source code means that this would have to be implemented through an API, which could further affect the speed of the simulation. Alternatively, all vehicles in the MMS, whether automated or not, will be represented with a manual vehicle object. Lists will be used to keep track of which vehicles are manual and which automated. The task of representing automated vehicles in the MMS is then
solved with an API that assigns a set of positions and velocities (prescribed by SmartAHS) to all the vehicles in the automated list.

Vehicle ID :  

| 1 | 2 | 3 | 4 |

SmartAHS :  

MMS :  

Figure 5: Vehicle representations in the INT

Figure 5 shows the four representations described above for vehicles in the INT region. It stresses the fact that all vehicles in this region are represented twice: once in each simulation environment. The aforementioned list that identifies automated vehicles in the MMS would, in this case, contain vehicle IDs 1 and 4.

D.3 Manual / Automated transitions

Any entry or exit design, and specifically those described in Section C.2.3, requires vehicles to switch from one type to another. For example, any entry maneuver will involve converting manual vehicles to automated ones in both simulation environments. Thus, the combined simulator must not only be able to simulate both vehicle types, but also the transfer of control between a human driver and the on-board computer. The methods described above for representing one vehicle type in another environment suggest how this can be done. In the case of a manual vehicle becoming automated, we assume that before the transformation takes place, the manual vehicle exists in the MMS, but is also represented by a ghost vehicle in SmartAHS. In the MMS, the transformation is simple: the vehicle’s ID is added to the list of automated vehicles in the INT region. In SmartAHS, the ghost vehicle is removed from the simulation and a full AHS vehicle type is created in its place. The inverse happens when the transformation is from automated to manual.

We assume that the command for a vehicle to switch types comes from the AHS control system, residing in SmartAHS. In our AHS entrance example (Figure 1), the check station logic is modeled by SmartAHS, and applied to the ghost vehicle representing a manual vehicle requesting access to the AHS. If the manual vehicle meets the AHS specifications, SmartAHS issues an event that triggers the manual/automated transition. This command is then conveyed to the manual simulator by the data exchange interface described in Section D.5.

D.4 User interfaces : input management and graphics

The main motivations for constructing a user interface are to eliminate data redundancy, to establish a method for processing simulation control commands, and to facilitate data interpretation by providing a common graphical output. In the first case, we want to ensure that common inputs (inputs that must be known to both simulators), e.g. the geometry of the AHS lanes, only need to be entered once by the user. In the second case, we want commands such as “pause simulation after 3 clock ticks”, to be routed to a common controlling process (described in Section D.5).
Thirdly, the user should be able to observe both manual and automated vehicles in a single output window. Thus, in this section, we identify and discuss the main features of the user interface: input management and graphical output.

It is important to note that the input-editing and graphical output capabilities of VISSIM, Paramics, and CORSIM are much more advanced than those of SHIFT/SmartAHS. Each MMS allows highway topology information to be entered graphically; for instance, in Paramics, model road geometry can be built on top of a Bitmap image of the study area. In our proposed design, we want to retain the favorable input/output features of the manual simulator. In particular, the user should be able to specify the entire network geometry through the MMS input editor, and view all vehicles (automated and manual) via the MMS graphical display.

D.4.1 Input management

In either the manual simulator or SmartAHS, inputs can be categorized according to how they are used. Some information, such as link lengths, lane widths, and loop detector locations, help to define the network layout. Another class of inputs are used to configure the simulation, e.g. which data to collect, total simulation time, and time step size. For each of the MMSs and SmartAHS, the user must supply system inflows (traffic demands) as a function of time. The inputs mentioned up to this point can be specified before the simulation is run. By contrast, other inputs are provided at run-time and affect the execution of the simulation; e.g., using the SHIFT debugger, a SmartAHS simulation can be paused or advanced by a user-specified number of time steps.

For the proposed simulator design, we want to develop a method for handling common inputs, i.e. those that are required by both simulation environments. Common inputs could fall into any of the categories mentioned above, for example,

- **Highway topology**: all of the highway topology in the SmartAHS domain (the AUT ∪ CO ∪ BUF region described in Section D.1) must be known to both simulators. SmartAHS must have this information for simulation purposes, but we want to allow the user to design all parts of the network using the MMS input tools, thus ensuring that all roads and barriers are visible in a single output window. Therefore, the MMS must also know the geometry of the region, including the portions that are purely automated.

In addition, we would like other roadside features of the AHS, such as the stop sign controller and check-in station of Figure 1, to be displayed in the graphical output. This presents a problem, since these components do not currently have a representation in any of the manual simulators. One of our goals is to determine suitable “placeholder” components in the MMS that (1) indicate the locations of AHS roadside elements in the common graphical display and (2) are inert, i.e. do not interact with the MMS simulation. We would choose placeholders from the pre-existing components in the MMS; for example, the AHS stop sign could be represented by an inactive manual simulator stoplight.

- **Traffic demands**: this information may or may not be shared between simulators, depending on the network geometry. If all inlet flows consist of manual traffic, then the user would specify all demands using the MMS input tools. However, the network could be designed to have one or purely automated inflows, which would result if, for example, the portion of
highway shown in Figure 1 constituted the entire region of simulation, thus requiring the
inflow to the AL lane to consist of automated vehicles. In this case, we still prefer to have
the user enter all flows using the MMS input methods, but the automated inlet flows must be
identified and communicated to SmartAHS.

- **Simulation configuration parameters**: some parameters, such as time step size, are relevant
to both simulators. We will determine a method for entering configuration parameters that
is convenient for the user; as in the previous cases, this may involve entering values through
the standard MMS interface and transferring this information to SmartAHS.

- **Simulation control commands**: unlike the previous examples, these inputs are provided while
the simulation is running. In this case, the input method will depend on the design chosen
for the overall simulation control structure, discussed in Section D.5. As explained in that
section, there are several entities that could ultimately serve as the controlling process, i.e.,
the program that coordinates execution of the combined simulator. For example, if Smar-
tAHS controls execution of the MMS, it would be natural to process run-time commands
through the SHIFT debugger.

To handle common information that is specified before execution of the simulation program,
we propose to develop a *secondary input file* and a *secondary input processor*. The secondary
input file will contain all the AHS component information in a SmartAHS-compatible format. It
will also define the correspondence between the AUT, BUF, and CO regions and the equivalent
parts of the MMS topology. This could be done by relating the labels AUT, BUF, CO to their
associated link IDs in the MMS. Locations of automated inflows could also be coded in this file.
The secondary input processor will (1) combine information from the secondary file and the MMS
topology file to generate a SmartAHS-formatted topology file; (2) extract relevant simulation con-
figuration from the MMS (e.g. time step size) and place it in a SmartAHS input file; (3) create
additional SmartAHS input files containing specifications for all the AHS roadside components
and boundary flows; (4) use the AHS component information to write placeholder objects into
the MMS topology file, so that AHS elements are visible in the graphical output.

In the first stage of topology definition, the user will design the network layout using the input
facilities of the MMS. Detectors must be placed at certain boundaries between MAN-MBUF
regions, since as described in Section D.2, information of manual vehicles in the MBUF regions
will be communicated to SmartAHS. The portions of freeway that will eventually be represented
in SmartAHS (corresponding to the AUT, BUF, and CO regions of Figure 3) must be chosen in a
manner that is geometrically compatible with SmartAHS. Compatibility is a concern, considering
that while each MMS accommodates a wide range of highway topologies, the range of allowed
SmartAHS geometries is relatively limited. In SmartAHS, highways are specified as an intercon-
ected series of straight or curved sections of fixed width, as described in the SmartPath User’s
manual [7].

For inputs that must be specified prior to run time, the main steps of the input methodology,
summarized in Figure 6, are as follows:

1. User enters the network topology, traffic demands, and other configuration information using
   the standard MMS interface. Files containing this information are produced by the MMS.
2. In a secondary input file, the user specifies portions of the MMS topology that correspond to AUT, BUF, and CO regions, and which (if any) of the network inflows are purely automated. The user also defines AHS roadway components not supported by the MMS interface (e.g. stop sign controller, check-in station) in the secondary text file.

3. The secondary input file is fed to the secondary input processor. The input processor verifies that selected links can be represented in SmartAHS, and that connectivity information satisfies the rules of Section D.1. If an error is detected, the processor informs the user of the mistake, and asks the user to revise either the topology specification or the secondary input file.

4. If no errors are detected, the input processor adds AHS placeholder components to the MMS topology file, and generates the following files in SmartAHS format: an input file defining SmartAHS roadside elements, a file containing any common configuration data that were entered through the MMS interface, an AHS topology file, and a register of purely automated boundary flows.

![Figure 6: Input methodology](image)

**Figure 6: Input methodology**

### D.4.2 Graphical output

All of the manual simulators under consideration have more advanced graphical capabilities than SHIFT, thus we plan to use the MMS graphics processor to display all vehicles, both automated and manual, in the network. Roadside elements, such as the lane barriers and check-in station in Figure 1, will also be displayed in the MMS graphical output.

Displaying AHS vehicles in the MMS graphical output should be fairly straightforward. A API and foreign-function based data exchange method will be developed to transmit to the manual simulator, at every time step, the position of every automated vehicle in the network (for more
detail, see Section D.5). This could be facilitated, as suggested in Section D.2, by using the pre-existing manual vehicle type as a shell for SmartAHS dynamics, and keeping a register or list of all vehicles in the system that are currently automated. Vehicles belonging to the “automated” list will receive their position updates from SmartAHS. Every automated vehicle will thus have a dual representation in SmartAHS and in the manual simulator, as described in Section D.2.

D.5 Software interface: Windows implementation and data exchange

In this section we describe the dynamic elements of the combined simulator, namely the coordinated execution of SmartAHS and the MMS, and the methods used to exchange data between the simulators.

One simplifying assumption is that, in both simulators, the state of each object at time \( t + \Delta t \) depends only on past (i.e. step \( t \) and earlier) states and inputs, and not on predictions of the future states of any of the vehicles. This would lead naturally to a design described later in this section, wherein SmartAHS- and MMS-simulated regions are updated sequentially. According to Tom Simmerman of ITT industries, this basic assumption does not hold for CORSIM, since its internal model is partially based on a prediction of the future state of the leading vehicle. If the leading vehicle were controlled by SmartAHS, this could lead to problems if CORSIM happens to predict a vehicle state that is very different from the one supplied by SmartAHS. However, as described in Section C.1, an override API function can be used to bypass the car-following model in Paramics. Part of Task 1.1 will be to determine whether or to what extent MMS vehicle behavior would be affected by having SmartAHS overwrite the states of certain vehicles.

Another assumption is that the MMS time step can be chosen as an integer multiple of the SmartAHS update interval, or vice versa. For simplicity, we would prefer \( \Delta t \) to be the same in both environments, but some flexibility can be allowed if different time scales are preferred. As an example, a typical SmartAHS time step is 0.05s, whereas the suggested time step for VISSIM is \( 4 \times 0.05s = 0.2s \). In this case, the “run SmartAHS for \( \Delta t \)” task in Figure 7 will be interpreted as “run SmartAHS for 4 time clicks”.

The required tasks for the composite simulator are to:

1. Launch both programs
2. Carry out configuration tasks, e.g. load topology files, allow user to set length of simulation run and specify other parameters.
3. Coordinate execution of simulators, process run-time commands, and manage data exchange, e.g. execute the simulation loop shown in Figure 7.

As part of this project, we will identify which kinds of dynamic (time-varying) data must be exchanged between simulators. For the design described in Sections D.1, D.2, and D.3, we expect that, at every time step,

- Positions of all manually-driven vehicles in the MBUF and CO zones will be transmitted from the MMS to SmartAHS;
• Positions of all automated vehicles in SmartAHS will be transmitted to the MMS. This is a requirement of the graphical output design described in Section D.4;

• If certain event-based data are available, they will be communicated from the MMS to SmartAHS or vice versa. There are two cases in which vehicles cross the boundary of a simulation environment: (1) Manual vehicles entering the SmartAHS domain by crossing a MAN-MBUF interface are sensed by detectors in the MMS and communicated to SmartAHS, and (2) AHS vehicles entering the MMS domain by crossing an AUT-ABUF interface are detected by SmartAHS and communicated to the MMS. Also, manual/automated transitions triggered by SmartAHS need to be conveyed to the MMS.

We want to determine the simplest and most efficient means of realizing a run-time interface between the simulators. A number of design options can be considered; for example, data exchange and simulation control tasks could be (1) managed by an external batch file (e.g., a VisualBasic script); (2) managed by APIs of the manual simulator; (3) managed by SmartAHS, through the use of the Foreign Function Interface (FFI), an interface that allows SmartAHS to invoke C-language programs; (4) distributed between SmartAHS and the MMS.

Some constraints must be considered when designing an interface between SmartAHS and an external program. Joel VanderWerf of the PATH research program has provided us with helpful insights; for instance, he states that it is possible to “drive” a SHIFT simulation using an external program. That is, we can control the flow of execution of SmartAHS by communicating with the SHIFT command-line debugger through a one-way pipe. However, it is not possible to implement a two-way pipe with the SHIFT debugger, that is, we cannot use the debugger commands to extract data from SmartAHS at specified times. Although the debugger can be used to print SmartAHS vehicle data to the screen, the timing of the printout cannot be controlled. This is because SHIFT simulations write data to a buffer, and the buffer may be flushed to the screen at an arbitrary time, e.g. after the simulation is over. In light of Dr. VanderWerf’s advice, it seems that the best way to exchange data with a running SHIFT simulation is to use the FFI.

An additional task will be to determine an appropriate medium for the data exchange. One way would be to pass information between programs using text files, but for large amounts of data,
this method could prove to be slow. An alternative is to use socket-based communications. A 
socket is an endpoint of a two-way communications channel that can be used to pass data between 
applications residing on the same machine, or between machines connected through a network [14].

As an example, a possible realization of the simulation control loop of Figure 7 would be to 
interface SmartAHS with Paramics, using SmartAHS to both control the flow of execution and 
initiate data exchange, with the help of sockets. The main steps of the algorithm would then be:

1. SmartAHS opens a socket with Paramics; commands Paramics to ad-

   vance $\Delta t$ and pause. This could be achieved by invoking a Paramics overload func-
   tion (described in Section C.1), which is an API that allows interruption of the Paramics 
simulation.

2. SmartAHS advances $\Delta t$ and pauses.

3. SmartAHS invokes foreign functions that carry out data exchange 

   with Paramics. Information would again be interchanged over sockets, making use of 
   Paramics override functions to import SmartAHS data into Paramics. Data required by Smar-
   tAHS, such as manual vehicle locations in the CO zone, could be extracted from Paramics 
   by callback functions.

4. Return to first step.

   Similar constructs to the overload/override/callback functions could be most likely be imple-

   mented through the CORSIM RTE, which allows custom code to be executed during each COR-
   SIM simulation cycle. Alternatively, we could use the VISSIM API, but further consultation with 
   ITC will be needed to assess the state of their API development. While the SHIFT restrictions 
   indicate that SmartAHS should manage the exchange of data, additional research will be required 
   to determine the best design for the overall simulation control loop.

**D.6 Development Stages**

The project will consist of three development stages, described below.

**Stage 1: Software selection and design**

During this stage, we will select the manual microsimulator for integration with SmartAHS, and 
settle on a method for implementing the software link between the two environments. In this 
proposal, we have identified three candidates, VISSIM, Paramics, and CORSIM, that are favored 
due to their modeling features, flexibility, and/or widespread use. However, in our investigations 
of their capabilities and through communications with users and technical staff, none has emerged 
as a clear best choice. The following table summarizes some of the findings:
<table>
<thead>
<tr>
<th></th>
<th>PROS</th>
<th>CONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VISSIM</td>
<td>A sophisticated driver model. Good graphical capabilities. Developer’s support.</td>
<td>API not fully developed and without documentation.</td>
</tr>
<tr>
<td>CORSIM</td>
<td>Extensive field validation. Existing RTE for connecting other programs. Inexpensive.</td>
<td>Less advanced graphical capabilities. No technical support for the RTE.</td>
</tr>
<tr>
<td>Paramics</td>
<td>Speed and scalability. Ongoing API development in PATH. Advanced graphics.</td>
<td>Low modeling complexity.</td>
</tr>
</tbody>
</table>

Through further consultation with expert users and developers, we will evaluate and compare the manual simulators based on the following criteria:

1. **Input/output features:** How easily can the input and output data formats be modified to accommodate our design?

2. **Existing APIs:** Can we use any previously developed APIs to help realize the interface?

3. **Validation against real-world data:** To what extent has the simulator been tested against actual traffic measurements? Has it been shown to be accurate?

4. **Efficiency:** How does the simulation-to-real time ratio vary with the number of vehicles in the simulation? Is one MMS clearly more efficient than the others?

5. **Features of vehicle model:** How do the vehicle behavioral models compare in terms of complexity and accuracy?

Once a manual simulator is selected, we will be able to decide on a method for coordinating the two simulation environments and exchanging data, based on the interfacing capabilities of the chosen simulator. Also, having decided on a manual model, we will be able to compile a list of all input data required by the combined simulator and settle on a format for the extended input files. Deliverables completed during this stage will be (1) a comparative study of the three simulators, citing pros and cons and justifying our final selection, and (2) a complete description of the structure of the combined simulator, which will be made available in a report.

**Stage 2: Implementation** : Figure 8 illustrates the various software elements that will be created during the second stage of development. In the figure, boxes represent programs or routines and arrows represent information pathways or text files. The components that must be developed are the following:

- **Vehicle representations and manual/automated transitions:** The facilities needed to represent automated vehicles in the MMS will be created, including an API that overrides manual vehicle dynamics with SmartAHS-derived positions and velocities. A SmartAHS ghost vehicle representation for manual vehicles in the INT region will be implemented. Manual/automated transition procedures will be coded into SmartAHS.
Coordinating routine: Depending on the results of the first stage, this box may be implemented within one of the two simulators, or as a separate entity. Its functions will be to coordinate the evolution and exchange of information between the two simulators, and interpret user simulation control commands.

Exchange of information: The arrows connecting each of the two simulators to the coordinating routine box represent information routes that will be implemented using APIs, RTEs, foreign functions, Windows sockets, etc. depending on the MMS selected in Stage 1.

The deliverable for this stage will be a preliminary version of the simulator that, while not fully tested, contains the main modules described above.

Stage 3: Testing:
Once all elements of the combined simulator have been created and assembled, the final task will be to test it using the entry example described in this proposal. The objective of this exercise will be to ensure that the different components of the new simulator function correctly. Thus, the effort will be mostly focused on debugging and demonstrating the user interfaces and software.
coordination algorithms, and less on verifying and calibrating the models. The results of this stage will be (1) a tested version of the simulator (2) a final report including all design information and technical documentation for the simulator, and (3) a User’s Manual, which will provide steps for setting up simulations and obtaining results, and will include the entrance configuration as an illustrating example. The last deliverable will consist of a workshop, where we will present the the SmartAHS/manual simulator, and explain its use, features, and benefits to the traffic research community.

E Research Plan and Deliverables

The proposed project has a planned two year duration. A concise statement of the tasks to be performed is given below.

Task 1: Selection and evaluation of a manual simulator.

- Task 1.1: A comparative study of the three options will be carried out using the criteria of Section D.6. Documentation of the study and findings will be provided.

- Task 1.2: Acquisition and installation of the selected simulator. This task includes becoming familiar with details of the simulator not considered in the study, yet relevant to the project. An example is the syntax of the API language.

Task 2: Completion of the simulator design based on this selection.

- Task 2.1: Design of the input file structure and methods for encoding and processing common time-invariant inputs. Design of common output representation (graphics and text files) (Section D.4).

- Task 2.2: Design of the simulation coordination algorithm (Section D.5). This task will rely heavily on the results from Task 1.

- Task 2.3: Design of dynamic data exchange methods. This task includes a specification of exactly what data needs to be exchanged and a software solution for doing it (Section D.4).

- Task 2.4: Design of dual vehicle representations and manual/automated transition protocols (Sections D.2 and D.3).

Task 3: Implementation of the design. The main deliverable of this task will be a beta-version of the simulator.

- Task 3.1: Coding of the input processor.

- Task 3.2: Coding of data exchange channels (APIs, RTEs, etc.).
- **Task 3.3:** Coding the coordination routine. This module will consist of a finite state machine included in SmartAHS, the MMS, or as a separate component, depending on the results of Task 1.

- **Task 3.4:** Coding of dual vehicle representations and manual/automated transition procedures.

- **Task 3.5:** Assembly of the various components and connection to the coordinating routine.

**Task 4:** Testing and documentation.

The final task will be to set up and run a scenario that tests the functionality of all the components of the combined simulator. We will consider a simple topology consisting of a one-lane linear AHS with upstream and downstream transition stations. This test will be documented and presented as an example in the User Manual and final report.

### Milestones

<table>
<thead>
<tr>
<th>List of tasks</th>
<th>Start date</th>
<th>End date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>7/1/2002</td>
<td>8/31/2002</td>
</tr>
<tr>
<td>1.2</td>
<td>9/1/2002</td>
<td>10/31/2002</td>
</tr>
<tr>
<td>2.1</td>
<td>11/1/2002</td>
<td>12/31/2002</td>
</tr>
<tr>
<td>2.2</td>
<td>11/1/2002</td>
<td>12/31/2002</td>
</tr>
<tr>
<td>2.3</td>
<td>11/1/2002</td>
<td>12/31/2002</td>
</tr>
<tr>
<td>2.4</td>
<td>11/1/2002</td>
<td>12/31/2002</td>
</tr>
<tr>
<td>3.1</td>
<td>1/1/2003</td>
<td>3/31/2003</td>
</tr>
<tr>
<td>3.2</td>
<td>4/1/2003</td>
<td>6/30/2003</td>
</tr>
<tr>
<td>3.3</td>
<td>7/1/2003</td>
<td>9/30/2003</td>
</tr>
<tr>
<td>3.4</td>
<td>4/1/2003</td>
<td>6/30/2003</td>
</tr>
<tr>
<td>3.5</td>
<td>10/1/2003</td>
<td>12/31/2003</td>
</tr>
<tr>
<td>4</td>
<td>1/1/2004</td>
<td>6/30/2004</td>
</tr>
</tbody>
</table>

### Deliverables

<table>
<thead>
<tr>
<th>Deliverables</th>
<th>Date of completion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Comparative study of the 3 simulators justifying final selection.</td>
<td>8/31/2002</td>
</tr>
<tr>
<td>2. Documentation of the completed simulator design.</td>
<td>12/31/2002</td>
</tr>
<tr>
<td>3. Untested version of the simulator</td>
<td>12/31/2003</td>
</tr>
<tr>
<td>4. Tested version of the simulator</td>
<td>6/30/2004</td>
</tr>
<tr>
<td>5. Final report including technical documentation.</td>
<td>6/30/2004</td>
</tr>
<tr>
<td>7. A workshop presentation.</td>
<td>6/30/2004</td>
</tr>
</tbody>
</table>
Qualifications of Principal Investigator, Key Researchers and Collaborators

Roberto Horowitz, Ph.D., received the B.S. degree with highest honors in mechanical engineering from the University of California at Berkeley in 1978 and the Ph.D. degree in 1983. In 1982 he joined the Department of Mechanical Engineering at the University of California at Berkeley. He is currently a Professor in the same department where he teaches and conducts research in the areas of adaptive, learning, nonlinear and optimal control, and dynamic systems, with applications to many mechatronics devices including micro-electromechanical systems, disk file systems, photocopying and printing machines and robots, as well as in the areas of the control of complex and hierarchical systems and intelligent vehicle and highway systems.

Dr. Horowitz is an associated faculty member of the Berkeley Sensor & Actuator Center (BSAC), a member of the Computer Mechanics Laboratory (CML), a principal investigator of the Partners for Advance Transportation and Highways (PATH) research program of University of California and CALTRANS, and a faculty member of the National Storage Industry Consortium (NSIC). He is also principal investigator of DARPA Project: High-Bandwidth, High-Accuracy MEMS Micropositioners for Magnetic Disk Drives.

Dr. Horowitz has been or is currently principal investigator of the following PATH research projects:

- MOU 383: Development of integrated meso/microscale traffic simulation software for testing fault detection and handling in AHS.

Dr. Horowitz has been and is currently involved in the formulation, design, testing and verification of many maneuvers and control laws for PATH’s AHS normal and degraded mode control architectures. As part of this work, Dr. Horowitz analyzed the link and network layers of the PATH architecture and synthesized decentralized and distributed control laws for these systems which were proven to be stabilizing. These control algorithms have been also tested extensively using AHS simulation tools. Dr. Horowitz has also been involved with the design, analysis and safety verification of the hybrid system formed by the coordination/regulation layers of the current PATH AHS control architecture.
Luis Alvarez, Ph.D., obtained a B.S. in Mechanical Engineering and an M.S. in Electrical Engineering from the National Autonomous University of Mexico in 1981 and 1988, respectively. He obtained a Ph.D. in Mechanical Engineering from the University of California at Berkeley in 1996. He is currently Professor at the Institute of Engineering of the National Autonomous University of Mexico. Professor Alvarez worked as a graduate research assistant and as a post-doctoral researcher with Prof. Roberto Horowitz in many issues associated with the PATH AHS control architecture. He was a researcher in PATH’s MOU-135, MOU-238/310, MOU-287/311 and MOU-288/312 projects, where he was involved in the design of feedback based control laws, stabilizing link layer controllers, emergency vehicle access and control, and in the implementation of the fault handling module for a fault tolerant AHS.

Dr. Alvarez continues to collaborate with Professor Horowitz on several PATH projects. During the last three summers he has been a visiting scholar at the Institute of Transportation Studies.
Roberto Horowitz

Professor of Mechanical Engineering
Vice Chair of Graduate Study
6139 Etcheverry Hall
University of California, Berkeley CA 94720-1740
Tel: (510)642-4675. Fax: (510) 642-6163.
E-mail: horowitz@me.berkeley.edu

Field of Specialization

Adaptive, learning, nonlinear and optimal control, with applications to many mechatronics devices including Micro-ElectroMechanical Systems (MEMS), disk file systems, photocopying and printing machines and robots. Control of complex and hierarchical systems. Intelligent vehicle and highway systems (IVHS).

Education

B.S. Mechanical Engineering, University of California at Berkeley, 1977.
Ph.D. Mechanical Engineering, University of California at Berkeley, 1983.

Experience

Acting Assistant Professor, University of California, 9/82 - 6/83.
Assistant Professor, 6/83 - 7/89
Associate Professor, 7/89 - 7/94
Professor, 7/94 - present.
Vice Chair of Graduate Study, Department of Mechanical Engineering, 7/01 - present.

Societies

Member, American Society of Mechanical Engineers (ASME).
Member of IEEE.

Awards

NSF Presidential Young Investigator Award
IBM Young Faculty Development Award.
Japanese Foreign Research Fellowship.

Recent IVHS Publications


Luis Alvarez

Professor of Mechanical Engineering
Institute of Engineering
National Autonomous University of Mexico (UNAM)

Education

Ph.D. in Mechanical Engineering, University of California, Berkeley.
Grade conferred: December 1996.

Relevant Experience

- June 1999 - August 1999: University of California, Berkeley
  Visiting scholar. Institute of Transportation Studies.
- June 1998 - August 1998: University of California, Berkeley
  Visiting scholar. Institute of Transportation Studies.
- August 1996 - December 1997: University of California, Berkeley
  Postdoctoral researcher. Institute of Transportation Studies.
- January 1995 - August 1996: University of California, Berkeley
  Research Assistant, Institute of Transportation Studies.
- September 1987 - to present: Instituto de Ingeniería, Universidad Nacional Autónoma de México (National Autonomous University of Mexico), México. Professor.

Recent Publications


<table>
<thead>
<tr>
<th></th>
<th>7/02-6/03</th>
<th>7/03-6/04</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Other Direct Costs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misc Equipment and Supplies</td>
<td>$990</td>
<td>$1,062</td>
<td>$2,052</td>
</tr>
<tr>
<td>Computer Software</td>
<td>$1,000</td>
<td>$1,000</td>
<td>$2,000</td>
</tr>
<tr>
<td><strong>TOTAL OTHER DIRECT COSTS</strong></td>
<td>$1,990</td>
<td>$2,062</td>
<td>$4,052</td>
</tr>
<tr>
<td><strong>TOTAL DIRECT COSTS</strong></td>
<td>$89,815</td>
<td>$91,962</td>
<td>$181,777</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>MTDC</th>
<th>MTDC</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Indirect Costs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.0% of Modified Total Direct Costs</td>
<td>$89,815</td>
<td>$91,962</td>
<td>$8,982</td>
</tr>
<tr>
<td><strong>TOTAL AMOUNT REQUESTED</strong></td>
<td>$98,797</td>
<td>$101,158</td>
<td>$199,955</td>
</tr>
</tbody>
</table>

1. Salary rate as of 10/01.
2. Salary rate adjusted to reflect a projected 2% cost of living increase effective every October 1st.
3. Salary rate adjusted to reflect a projected 3.5% cost of living increase effective every October 1st.
4. Summer salary for faculty
5. Only the first $25,000 of each subcontract is subject to indirect costs.
I Resources

Most of the resources are requested for the category of personnel. Prof. Horowitz will charge one summer month for each year of this project. Luis Alvarez will charge two months of the academic year for each year of this project. Two graduate student researchers and a 25% time post-doctoral student will be appointed to the project. Expenses are included for a computer and commercial software, which are needed to carry out the development and testing of the interface. Two out-of-state trips will be used to present research at academic conferences. An additional amount has been requested for miscellaneous supplies.
### Progress Reports

<table>
<thead>
<tr>
<th>Project Title:</th>
<th>MOU 373/Task Order 4207: Development and Implementation of a Vehicle-Centered Fault Diagnostic and Management System for the Extended PATH-AHS Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principal Investigator:</td>
<td>Professors R. Horowitz, K. Hedrick and M. Tomizuka</td>
</tr>
<tr>
<td>Funding Allocated:</td>
<td>$130,000</td>
</tr>
<tr>
<td>Funding Spent to Date:</td>
<td>$130,000</td>
</tr>
<tr>
<td>Starting Date:</td>
<td>September 1999</td>
</tr>
<tr>
<td>End Date:</td>
<td>December 2001</td>
</tr>
</tbody>
</table>

**In the space below, describe research progress relative to original research plan. Explain any deviations from plan.**

The following tasks were executed in this project:

1. Nonlinear observers/filters were designed and tested in simulations to detect the previously unconsidered faults.
2. Residual processor has been extended and tested in simulations to identify the unconsidered faults in MOU 312.
3. The fault management scheme has been redesigned to handle the faults in the regulation layer level without degraded model maneuvers. Integration between fault detection and identification schemes and the management system has been implemented and tested in the SmartAHS simulator using SHIFT.
4. An extended fault management scheme has been designed to include detecting and handling the “soft” faults. A tire/road friction estimation scheme has been designed and tested in simulations. Such an estimation scheme can be combined with the emergency braking controllers to enhance the safety and performance of both manual and automated vehicles.
5. A fault diagnostic scheme, both longitudinal and lateral, has been implemented and tested on vehicles in RFS. The experimental results verify the validity of the design.

**In the space below, list all project deliverables completed to date (research reports, software, publications, etc.).**

**Software:**

1. MATLAB software for simulation and testing of nonlinear observer/filters for fault diagnostics and residual processor for fault identification.
2. MATLAB software for simulation and testing of friction estimation and emergency braking controller.
3. Real-time implementation of fault diagnostics and identification of longitudinal and lateral control systems.
4. SmartAHS simulation software for integration of fault detection and identification and fault handling systems for longitudinal vehicle control systems.

**Publications:**

5. X. Claeyx, J. Yi, R. Horowitz and C. Canudas de Wit. *A new 3D tire/road friction model for vehicle simulation*, in proceedings of the 2001 ASME IMECE.
In the space below, describe research progress relative to original research plan. Explain any deviations from plan.

The following tasks were executed in this project:
1. SmartAHS capabilities have been extended. An extended communication structure, a simplified sensor architecture, and a simplified set of regulation-layer components have been developed. Under certain conditions, the simplified set produces identical results as the full components, and increases simulation speed 4 to 5 fold.
2. The interface between the mesosimulator SmartCAP and microsimulator SmartAHS has been designed and implemented in SHIFT. A SmartAHS component was created to schedule and monitor all aspects of the interface between SmartCAP and SmartAHS. The SmartCAP activity model was extended to include platooning and join/split maneuvers.
3. The integrated meso/microscale simulation software for stationary regions has been developed and tested for different scenarios. A MATLAB-based visual interface was created which allows the user to view both levels of the simulation simultaneously, by including both macro (curves) and micro (vehicles) data in the same plot. It uses MATLAB’s GUI capabilities so that users can easily manipulate (e.g. zoom, rotate, change) the output.
4. A redefinition of the project objectives led to the replacement of the final task, which was originally the development of a moving interface between SmartCAP and SmartAHS. It was decided that a more direct and useful application of the meso/micro simulator would be in analyzing the interaction of an AHS with connecting manual traffic arterials at high-volume AHS entry and exit locations. One microscale manual traffic simulator, along with a macroscale simulator, will be selected, respectively, to model manual traffic near to and far away from the AHS entrances and exits. We have examined several microscopic and macroscopic models of manual traffic, including VISSIM and the cell transmission model, and are considering them as candidates for integration with the meso/micro simulator.

In the space below, list all project deliverables completed to date (research reports, software, publications, etc.).

Software:
1. Extended SmartAHS simulation software including link, coordination, regulation, and physical layers and extended communication structure, and simplified sensor structure.
2. SHIFT-compatible version of extended SmartCAP macro-simulator.
3. Integrated meso/microscale simulation software for stationary region.

Reports:

Publications:
References


