# BEHAVIOR MODELING FRAMEWORK FOR EMBEDDED SIMULATION

**Amy Henninger, William Gerber, Ronald DeMara, Michael Georgiopoulos, and Avelino Gonzalez**
**University of Central Florida**
**Orlando, FL.**

## ABSTRACT

Although embedded training has become the preferred approach for training military forces, it is surrounded by a variety of technical challenges. The Inter-Vehicle Embedded Science and Technology (INVEST) Science and Technology Objective (STO) program explores technologies required to embed simulation in combat vehicles. One of these requirements is to provide a simulation environment in which computer generated forces, manned simulators, and live vehicles may interact in real-time. Unfortunately, providing this geographically distributed and untethered real-time interaction is severely limited by the communications requirements imposed by the need to convey large amounts of data between the respective players. By extending the concept of Distributed Interactive Simulation (DIS) dead-reckoning, a vehicle movement method, to the behavioral level, this limitation may be mitigated. The Vehicle Model Generation and Optimization for Embedded Simulation (VMGOES) project at the University of Central Florida is focusing on this aspect of the INVEST program. This paper presents the specifications and development process of VMGOES.

## ABOUT THE AUTHORS

**Amy Henninger** is a doctoral candidate in computer engineering at the University of Central Florida, a Research Fellow at U.S. Army STRICOM, and a recipient of the Ninth Annual I/ITSEC Scholarship. She has earned B.S. degrees in Psychology, Industrial Engineering, and Mathematics from Southern Illinois University, an M.S. in Engineering Management from Florida Institute of Technology, and an M.S. in Computer Engineering from UCF.

**William Gerber**, Lt. Col., U.S.A.F. (Ret.), is a Ph.D. student in computer engineering at the University of Central Florida and a Research Fellow at U.S. Army STRICOM. He has a B.S.E.S. degree in Astronautics and Engineering Sciences from the U.S.A.F. Academy, an M.S.E. in Nuclear Engineering from the University of California at Los Angeles and an M.S.Cp.E. in Knowledge-Based Systems from UCF.

**Ronald DeMara** is a full-time faculty member in the Electrical and Computer Engineering Department at the University of Central Florida. Dr. DeMara received the B.S.E.E.. degree from Lehigh University in 1987, the M.S.E.E. degree from the University of Maryland, College Park in 1989, and the Ph.D. degree in Computer Engineering from the University of Southern California, Los Angeles in 1992.

**Michael Georgiopoulos** is an Associate Professor at the Department of Electrical and Computer Engineering of the UCF. His research interests lie in the areas of neural networks, fuzzy logic and genetic algorithms and the applications of these technologies in cognitive modeling, signal processing and electromagnetics. He has published over a hundred papers in scientific journals and conferences.

**Avelino Gonzalez** received his bachelor's and master's degrees in Electrical Engineering from the University of Miami, in 1973 and 1974, respectively. He obtained his Ph.D. degree from the University of Pittsburg in 1979, also in Electrical Engineering. He is currently a professor in the Electrical and Computer Engineering Department at UCF, specializing in human behavior representation.

# BEHAVIOR MODELING FRAMEWORK FOR EMBEDDED SIMULATION

**Amy Henninger, William Gerber, Ronald DeMara, Michael Georgiopoulos, and Avelino Gonzalez**
**University of Central Florida**
**Orlando, Florida**

## INTRODUCTION

The combination of computer simulation and networking technologies has provided the U.S. military forces with an effective means of training through the use of Distributed Interactive Simulation (DIS). DIS is an architecture for building large-scale simulation models from a set of independent simulator nodes (Smith, 1992) that represent one or more entities in the battlefield simulation. By communicating over a network via a common protocol, these entities are able to exist simultaneously and interact meaningfully in the same virtual environment. Currently, however, the ability of live vehicles to interact with these simulated forces in the virtual world is constrained by the communication requirements needed for real-time interoperability. Eliminating or reducing this impediment would enhance military training in a number of ways. For example, it would diminish the costs associated with having live vehicles travel to maneuver ranges for live exercises. Also, by shifting more of the training to operational units, it would reduce the costs associated with the training schools. In essence, the military could rely less on formal school-house training, more on deployable training systems, and fundamentally make training more readily available on an "as-needed" basis.

To accomplish these objectives, the Department of Defense has recently initiated an effort to determine how embedded training and advanced simulation technologies could be used to overcome the obstacles surrounding this technology. One problem, for instance, is that in order for a driver of a live vehicle to train in a virtual domain, he must be able to traverse the artificial/virtual terrain. Correspondingly, he must be able to see the other live and virtual entities on the virtual battlefield and interact with them in real time. To accomplish this, the embedded training systems must sustain the transfer of massive volumes of data. Unfortunately, the networking and communications limitations of currently fielded systems make the transfer of this data using current DIS supported techniques a strenuous task.

Current forms of DIS dead-reckoning are viewed as vehicle movement methods that are used to reduce DIS packet traffic. By communicating a given vehicle's location, velocity and acceleration to other DIS simulators, the models residing on these simulators can predict the unperturbed near term physical location of the vehicle. In the event that this vehicle begins to deviate from its predicted path, the simulator responsible for creating the entity will send out an update of the vehicle's true location to the other simulators. Thus, the predictive utility of the dead-reckoning model is pivotal to the success of network traffic minimization.

The requirement to transfer enormous volumes of data coupled with the communication limitations of currently fielded systems makes using currently existing DIS methods an inadequate approach. Bahr and DeMara (1996) suggest that extending the concept of DIS dead reckoning to the behavioral level may reduce DIS traffic more than merely applying DIS dead reckoning to vehicle movement tasks. Figure 1 illustrates the DIS dead reckoning concept applied to embedded training and simulation. As indicated by Figure 1, this concept requires the distributed processing of multiple vehicle models because every live or simulated vehicle is represented by a model and every model is resident on every vehicle. The vehicle model (VM) serves to predict the actions of the vehicle it represents. When the actions of the vehicle are consistent with the actions predicted by the vehicle's model, all of the copies of that vehicle's model are correctly reflecting the live vehicle's actions. In this instance, the interaction between the other vehicles and the vehicle model in the virtual world is an accurate representation of the vehicles' interactions in the real world. However, if the actions of the vehicle are not consistent with the actions predicted by the vehicle's model, the copies of that vehicle's model

are not correctly reflecting the live vehicle's actions. In this instance, the interaction between the other vehicles and the vehicle model is not consistent with their real world interaction.

As indicated in Figure 1, a system that extends the DIS dead-reckoning concept to the behavioral level requires the identification of discrepancies between the behavior of an actual vehicle and that vehicle's model. The portion of this system that identifies and classifies these discrepancies is referred to as the Difference Analysis Engine (DAE) in Figure 1. By comparing the state of the vehicle model with the state of the actual entity, the DAE identifies whether discrepancies in the behavior as well as the position exist. If there are discrepancies, the DAE determines whether an update is necessary and what that update should be. The types of information provided by the DAE are specified in a future section of this paper.

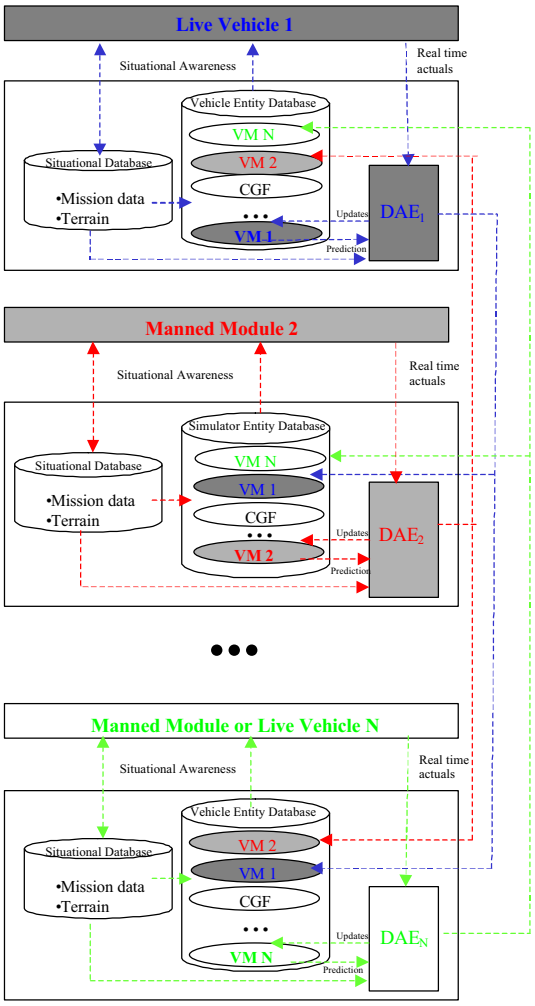


Figure 1. DIS Dead-Reckoning Approach Extended to Behavioral Level

This paper offers a framework for the development of the VM and the DAE. Also, this paper addresses the integration of the two components into the full system known as Vehicle Model Generation and Optimization for Embedded Simulation (VMGOES).

## SCOPE OF MODEL

Frequently, DIS simulations use computer controlled combatants known as Computer Generated Forces (CGFs) to populate the battlefield. The behavior of a CGF may be generated by a human operator assisted by software, in which case the class of CGF is referred to as a semi-automated force (SAF) or generated completely by software, in which case the class of CGF is referred to as an autonomous force (AF). The behaviors generated by CGFs are based on doctrine and represent a wide variety of tasks with a reasonable level of detail. Because these behavioral models are fashioned entirely by doctrine, they emulate standard procedures that are acquired from declarative knowledge (i.e., manuals and interviews) and provide a range of *feasible* behavior. However, these models of behavior provide no representation for the 1) implicit knowledge or 2) intrinsic performance characteristics that make "live entities" unique from one another. For example, the current CGF behavioral models used in DIS exercises may simulate the movement of a vehicle to a given location by some standard movement model, but they do not "individualize" that movement method by either assigning or simulating human performance characteristics (e.g., tendency to hug the side of the road, propensity to maintain speed above speed limit, etc.) to it. Thus, behavioral models fashioned entirely by doctrine are often characterized as yielding responses that are "canned", "predictable", or "too perfect". However, the fact that these behaviors are "canned" or "preprogrammed" in no way suggests that these behaviors are simplistic. Prevalent SAF systems have integrated hundreds of thousands of lines of code to successfully emulate the command and control hierarchy of a military unit and its operation on the battlefield. By providing a variety of planned behaviors (e.g., "Conduct a Tactical Road

March", "Attack By Fire", "Service Station Resupply", etc.), situational awareness and assessment, and reactive behaviors (e.g., "Breach a Minefield", "Call for Indirect Fire", "Actions on Contact", etc.), they have successfully provided suitable friendly and enemy forces to populate the battlefield.

The models to be used in this project are conceptually similar to CGF models, but they are distinguishable by the addition of human performance characteristics in the model. In other words, whereas a CGF may *emulate* the selection of a vehicle's cover and concealment position, extending the DIS dead-reckoning concept to the behavioral level requires the *prediction* of the vehicle's actual cover and concealment position. This necessary increase in detail for the VM coupled with the research oriented nature of this project, limits the initial efforts for VMGOES to an exercise smaller in scope than one may find in a typical DIS exercise.

The exercise used in VMGOES centers around a Blufor M1A2 tank platoon or section performing a Tactical Road March and executing an Actions on Contact task in response to a potential enemy threat (i.e., an Opfor T-72 platoon, section, or vehicle). A variety of control parameters can be modified by the VMGOES model users. This allows the users to more fully exercise the model to evaluate its ability to generalize. These parameters are categorized in two groups: (1) task parameters and (2) operational parameters. These parameters and their permissible ranges are defined below.

Task parameters that may be changed by the evaluators are expressed by task. These tasks include Tactical Road March and tasks related to Actions on Contact maneuvers.

**Tactical Road March Parameters**

Tactical Road March parameters that may be modified include the route and march rate.

***Route -*** may be defined within the constraints of the assumptions/conditions (listed under Assumptions section).

***March rate -*** must be defined within the acceptable limits of the march rates delimited in simulation.

**Actions on Contact Parameters**

Rules of engagement is the only parameter that may be modified to influence this task.
***Rules of engagement -*** may be initialized as free, tight, or hold to either all or none of the Blufor M1A2 entities.

**Operational Parameters**

The following operational parameters may be changed in a VMGOES exercise:

***Terrain -*** area where scenario is executed within constraints of the Assumptions section

***Blufor Unit Size -*** tank section or tank platoon

***Opfor Unit Size -*** single vehicle, tank section or tank platoon, and

***Opfor Unit Location -*** positioning (location and direction) of Opfor unit

**Assumptions**

Lastly, the following conditions/assumptions will apply to the exercises considered by VMGOES:

1. Terrain does not include bodies of water (e.g., lakes, rivers, swamps or ponds).
2. Model does not simulate Command Overrides, Fragmented Orders, or other externally initiated changes in orders.
3. Opfor (T-72 vehicles) operate according to defaulted behavior of simulation unless specified otherwise for a scenario.
4. Blufor units should begin exercise on route, have heading directed towards end of route, and be oriented closely parallel to its position on the route.
5. Manned module always represents the lead tank (i.e., platoon leader).
6. There will be no modifications to terrain (e.g., obstacles or minefields).
7. M1A2 may not initiate calls for support (e.g., indirect fire).
8. The section of terrain east of Barstow Road and west of Hill 720 in the NTC-0101 terrain database will be used for development and tests.
9. The simulation's environmental factors (e.g., weather, tactical smoke, etc.) will not change during a scenario.

10. Tactical Road March tasks may only be assigned to terrain where the road is observable.

## MODELING PARADIGM

To develop the vehicle model, VMGOES is using a machine learning technique known as Learning by Observation (Gonzalez, et al, 1998). This technique facilitates the development of intelligent, computational models of human behavior. Although a relatively new concept in the discipline of machine learning, Learning by Observation has been successfully used in a variety of highly complicated, real-world tasks. Pomerlau (1992), for example, used Learning by Observation in the development of an Autonomous Land Vehicle In a Neural Network (ALVINN). In this project, Pomerlau trained a neural network to drive a vehicle through a one-lane road under ideal environmental conditions. Moreover, this network was able to generalize its training to perform satisfactorily in two-lane as well as in dirt roads, and under adverse environmental conditions (snow, rain, etc.). Expanding on this work, Pomerlau et al. (1994) have developed “smart” vehicles as part of Advanced Research Projects Agency's (ARPA's) Unmanned Ground Vehicle (UGV) program, intended to reduce the need for human presence in hazardous situations. These vehicles are capable of driving themselves at speed up to 55 mph for distances of over 90 miles on public roads. Moreover, they are capable of driving both during the day and night, driving on a variety of roads, avoiding obstacles, and even performing parallel parking.

With respect to the vehicle models in this project, Learning by Observation will be used to learn human decision making skills (e.g., reactive transitions, route planning, selection of cover and concealment) and low-level human control strategies (e.g., route following, scanning, etc.). Ultimately, these behaviors will be learned through the observation of a human expert tank commander. However, as a preparatory step, VMGOES is currently developing a VM prototype by learning these behaviors through the observation of a ModSAF M1A2 entity. In other words, instead of using a human as the expert whose behavior is learned, VMGOES is initially using a ModSAF entity as the “expert” whose behavior is learned. This prototype model is referred to as $VM_{ModSAF}$. It is anticipated that this prototype work will assist in the identification of technical issues that may arise in the second phase of the study and ultimately, will increase the likelihood of successful results in the final system.

## VMGOES DEVELOPMENT

In addition to using ModSAF to supply the M1A2 "expert" entity from which $VM_{ModSAF}$ will learn, the ModSAF system is being used to provide the simulated environment in which the vehicle models interact. This allows VMGOES researchers to focus on the development of accurate behavior-based models as opposed to the implementation issues pertaining to vehicle simulation (e.g., physical modeling, weapons modeling, network interface, etc). The VM is embedded in ModSAF and receives input data consistent with sensory information obtainable from the controls and display systems resident in an M1A2. The VM output contains the commands and parameters needed to control the vehicle's motion and weapons' execution. The DAE, alternatively, runs as a separate process and receives input from both the vehicle model and the (live or simulated) master vehicle's interface. In both the $VM_{ModSAF}$ and the vehicle model derived from the human expert ($VM_{MM}$), this interface supplies sensory information and dead-reckoning type data pertinent to the given model's behavior. This is also true of the interface to the (live or simulated) master vehicle. Once it receives these inputs, the DAE identifies whether discrepancies exist between the vehicle models and the master vehicle and sends out the necessary updates.
The updates sent out by the DAE contain one or more of following four types of information: 1) position, orientation and other basic dead-reckoning information 2) model parameter information, 3) behavior enumeration, or 4) action enumeration. Examples of updates containing these types of information are provided in the following section.

### Software Engineering Model

As illustrated in Figure 2, VMGOES has adopted an Incremental Model (Schach, 1993) software engineering process. Using this type of model, the product is designed, implemented, integrated, and tested as a series of thirteen incremental builds, where each build is represented by a numbered circle. Also, each build consists of code pieces that interact together to provide a specific

functional capability. For the most part, the stages down the left-hand column of Figure 2 represent the VM builds and the stages across the bottom two rows represent the DAE builds as they are integrated with the VM. Of the two rows representing the DAE builds, the top row represents the development cycle of VMGOES using a ModSAF entity as the "expert", and the bottom row represents the development cycle of the VMGOES using the human expert. Finally, the integration of the two components (VM and DAE) occurs at the intersection of the column and rows.

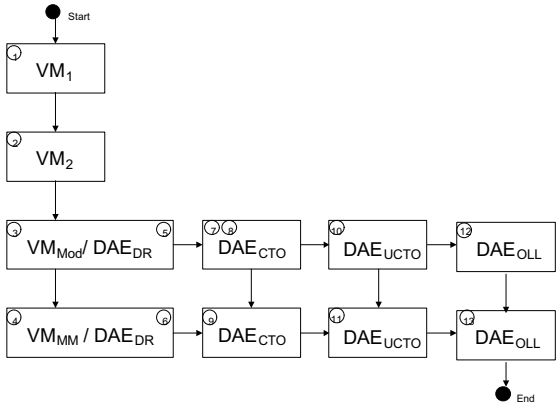


Figure 2. VMGOES Development Model

The thirteen VMGOES model builds presented in Figure 2 are enumerated and defined below according to their functional divisions. Builds 1 and 2 can be described as:

1. Train $VM_1$ to replicate reactive behavior context transitions by observing a ModSAF M1A2 CGF entity. In this build the reactive behaviors are enumerated as a part of the training set.

2. Train $VM_2$ to replicate reactive behavior context transitions by observing a ModSAF M1A2 CGF entity. In this build the reactive behaviors are not enumerated as a part of the training set.

In both Builds 1 and 2, a ModSAF M1A2 entity serves as the "expert" from which knowledge is acquired. Also, both models resulting from these builds focus on the acquisition of knowledge pertaining to unit level reactive behaviors. $VM_1$ and $VM_2$ are both trained with data containing sensory information (input). The difference, however, is that the output data used to train $VM_1$ includes the reactive behavior enumeration, whereas the $VM_2$ model does not have access to this enumeration. As a result, $VM_2$ must additionally employ some strategy to infer the reactive behavior type that should be associated with a given input vector or cluster of vectors. This second build better reflects the actual task at hand: to learn behaviors from a *human* expert. In other words, since the human expert will not be verbalizing or enumerating his behavior, the methodology for developing the final models in this project must be capable of inferring what that behavior is. It is anticipated that methods developed in Build 2 will assist in meeting this requirement.

Builds 3 and 4 can be described as:

3. Train $VM_{ModSAF}$ to replicate context transitions and actions for VMGOES test scenarios by observing ModSAF M1A2 CGF entity.

4. Train $VM_{MM}$ to replicate context transitions and actions for VMGOES test scenarios by observing a human expert in a M1A2 Manned Module.

The functionality provided by both Builds 3 and 4 is specific to the VMGOES test scenarios as defined in the VMGOES Requirements Document. These are briefly described in the Model Scope section of this paper. The difference between Builds 3 and 4 is that Build 3 uses a ModSAF entity as the "expert", whereas Build 4 uses a human expert. As previously discussed, it is anticipated that modeling strategies learned by the VMGOES team in Build 3 will be useful in the development of the final vehicle model ($VM_{MM}$) developed in Build 4.

Builds 5 and 6 can be described as:

5. Integrate $VM_{ModSAF}$ with DAE dead-reckoning control ($DAE_{DR}$) to evaluate $VM_{ModSAF}/DAE_{DR}$ for VMGOES test scenarios.

6. Integrate $VM_{MM}$ with $DAE_{DR}$ to evaluate $VM_{MM}/DAE_{DR}$ for VMGOES test scenarios.

Builds 5 and 6 are simply integration checkpoints in the development cycle. In both of these builds, the vehicle models are being integrated with the

DAEs' basic dead-reckoning control mechanism. This will enable the DAE to update the vehicle model position or orientation with basic dead-reckoning type parameters, in the event that the VM has deviated from the path pursued by the master vehicle.

Builds 7, 8, and 9 can be described as:

7. Train DAE context transition override control ($DAE_{CTO}$) of $VM_{ModSAF}$ to recognize context transitions for VMGOES test scenarios. In this build, reactive behavior transitions are supplied as part of the training set.

8. Train $DAE_{CTO}$ of $VM_{ModSAF}$ to recognize context transitions for VMGOES test scenarios. In this build, reactive behavior transitions are not supplied as part of the training set.

9. Train $DAE_{CTO}$ context transition override control ($DAE_{CTO}$) of $VM_{MM}$ to recognize context transitions for VMGOES test scenarios.

In general, these builds focus on the context transition override control of the DAE. This control mechanism allows the DAE to update the VM's enumerated behavior type and is used when the DAE identifies the behavior/context of the live vehicle as being different from the behavior enumerated by the vehicle's model. For example, if the VM is performing a "Withdraw" and the DAE determines that the live vehicle is performing an "Assault", the DAE directs the VM to change its behavior to an Assault.

Specifically, Builds 7 and 8 are using ModSAF as the "expert" and Build 9 is using a human in a manned module as the expert. Additionally, Build 7 and 8, like Builds 1 and 2, are distinguishable by the availability of behavior enumerations in the output of the training set.

Builds 10 and 11 can be described as:

10. Train DAE unrecognized context transition override control ($DAE_{UCTO}$) of $VM_{ModSAF}$ to recognize unrecognizable context transitions for VMGOES test scenarios.

11. Train $DAE_{UCTO}$ of $VM_{MM}$ to recognize unrecognizable context transitions for VMGOES test scenarios.

Builds 10 and 11 focus on providing the DAE with the capability to completely control the vehicle model, when the DAE is unable to recognize what the live vehicle is doing. Again, Build 10 uses the ModSAF M1A2 entity as the "expert" and Build 11 uses a human expert.

Lastly, Builds 12 and 13 can be described as:

12. Develop procedures for refining previously trained $DAE_{UCTO}/VM_{ModSAF}$ off-line (adjust $DAE_{UCTO}/VM_{ModSAF}$ parameters or contexts) for VMGOES test scenarios.
13. Develop procedures for refining previously trained $DAE_{UCTO}/VM_{MM}$ off-line (adjust $DAE_{UCTO}/VM_{MM}$ parameters or contexts) for VMGOES test scenarios.

Once VMGOES becomes functional, it will have access to more observational data. Those data may help further explain behavior. Builds 12 and 13 capitalize on this fact by providing a mechanism to capture those data and refine the vehicle models.

## SUMMARY

This paper described a modeling framework for the development of a system designed to reduce the communications bandwidth required for an inter-vehicle embedded simulation exercise. This system includes a behavior model of the vehicle in the exercise and a difference analysis engine tasked with keeping that model synchronized with its live counterpart. Presently, the vehicle model and the DAE are being developed by using a ModSAF M1A2 entity as the "expert" from which knowledge is acquired. Future endeavors include efforts to apply the lessons learned from this phase of the study to the elicitation of knowledge from a human expert.

## ACKNOWLEDGEMENTS

This work was sponsored by the U.S. Army Simulation, Training, and Instrumentation Command as part of the Inter-Vehicle Embedded Simulation and Technology (INVEST) Science and Technology Objective (STO), contract N61339-98-K-0001. That support is gratefully acknowledged.

## BIBLIOGRAPHY


Bahr,H.A. and DeMara, R.F., (1996). A Concurrent Model Approach to Reduced Communication in Distributed Simulation, Proceedings of the 15th Annual Workshop on Distributed Interactive Simulation, Orlando, FL.

Gonzalez, A.J., Georgiopoulos, M., DeMara, R.F., Henninger, A, and Gerber, W., (1998). Automating the CGF Model Development and Refinement Process by Observing Expert Behavior in a Simulation. In Proceedings of the 8th Conference in Computer Generated Forces and Behavior Representation, Orlando, FL: University of Central Florida Institute for Simulation and Training.

Pomerlau, D.A., (1992). Neural Network Perception for Mobile Robot Guidance, Ph.D. Dissertation, School of Computer Science, Carnegie Melon University, Pittsburg, PA.

Pomerlau, D., Thorpe, C., Longer, D., Rosenblatt, J.K., and Sukthankar, R., (1994). AVCS Research at Carneghie Mellon University. Proceedings Of Intelligent Vehicle Highway Systems America 1994 Annual Meeting, p. 257-262.

Schach, S.R., (1993). Software Engineering. Aksen Associates Incorporated Publishers, Boston, MA.

Smith, S., and Petty, M. (1992). Controlling Autonomous Behavior in Real-Time Simulation. In Proceedings of the Second Conference in Computer Generated Forces and Behavior Representation. Orlando, FL: University of Central Florida Institute for Simulation and Training.

**This document is an author-formatted work. The definitive version for citation appears as:**

A.E. Henninger, W. Gerber, R. F. DeMara, M. Georgiopoulos, and A. J. Gonzalez, "Behavior Modeling Framework for Embedded Simulation," in *Proceedings of the 1998 Interservice/Industry Training, Simulation and Education Conference (I/ITSEC'98)*, Orlando, Florida, U.S.A., November 30 – December 3, 1998.