The APL Coordinated Engagement Simulation (ACES)

Michael J. Burke and Joshua M. Henly

The APL Coordinated Engagement Simulation (ACES) is being developed to analyze methods of executing engagements in which multiple units have capability against multiple threats. In such situations, coordination among the units may improve performance against the raid. ACES generates Monte Carlo results to evaluate alternative methods of engagement coordination. Key features affecting engagement outcome are represented at a level appropriate for Monte Carlo analysis. Unit behavior is modeled by functional decomposition to facilitate representing new or imagined system capabilities. Each unit acts independently on perceived reality from its own sensors and information received from other units via ACES' network models. Initial development has focused on Navy upper- and lower-tier ballistic missile defense. Future versions are planned to include land-, air-, and space-based units and to simulate multi-service and multi-mission scenarios.

INTRODUCTION

Theater Air and Missile Defense (TAMD) in the littoral environment is a complicated situation in which multiple units have capability against multiple threats. The overlap in system capabilities offers flexibility in engaging the threat, but lack of coordination among the units could result in more than one unit engaging the same threat. Because each unit can only engage a limited number of threats, overengagement on one threat can cause lost opportunity against other threats. A method of engagement coordination is therefore desired to decide which unit should engage which threat to best defeat the raid.

Alternative means of engagement coordination must be evaluated by comparing their relative performance against complex and varied TAMD scenarios. Computer simulation is the only feasible way to perform this task, but an appropriate simulation environment is required. Previous analysis used the CERT (Coordinated Engagements against Raids of TBMs) simulation,[1] which implemented a distributed engagement decision process in detail; however, its low-fidelity representations of other system functions limited the scope of analysis. A high-fidelity representation of the detect-control-engage process has been modeled in ARTEMIS[2] (APL Area/Theater Engagement Missile/Ship Simulation), but this simulation cannot currently address engagement coordination because the initial implementation focuses on a single ship. A simulation for engagement coordination analysis must adequately capture the detect-control-engage performance of each unit in the scenario *and* model the networks used to exchange information as well as the logic used to act on that information. The APL Coordinated Engagement Simulation (ACES) is being developed to address these needs.

The ACES logo (Fig. 1) symbolizes the multi-unit, multi-service behavior that will be represented within the simulation. The particular units depicted reflect ACES' initial focus on Tactical Ballistic Missile Defense (TBMD). Version one of the simulation is capable of analyzing Navy upper- and lower-tier TBMD. The multi-year development effort includes plans to model other mission areas.

A general understanding of ACES can be obtained by considering its inputs, outputs, and key features (Fig. 2). Analyst involvement is needed to get the desired inputs and to ensure that they are self-consistent and appropriate to the desired analysis outputs. Inputs can be chosen to reflect the capabilities of current systems or to explore parametric variations. Some ACES inputs, such as threat trajectories and interceptor performance, are generated by other simulations. These data could be generic in nature or based on high-fidelity, system-specific representations.

Figure 1. The ACES logo symbolizes the multi-unit, multi-service behavior that will be represented within it. The particular units depicted reflect the initial focus on Tactical Ballistic Missile Defense. Other mission areas are part of the multi-year development effort.

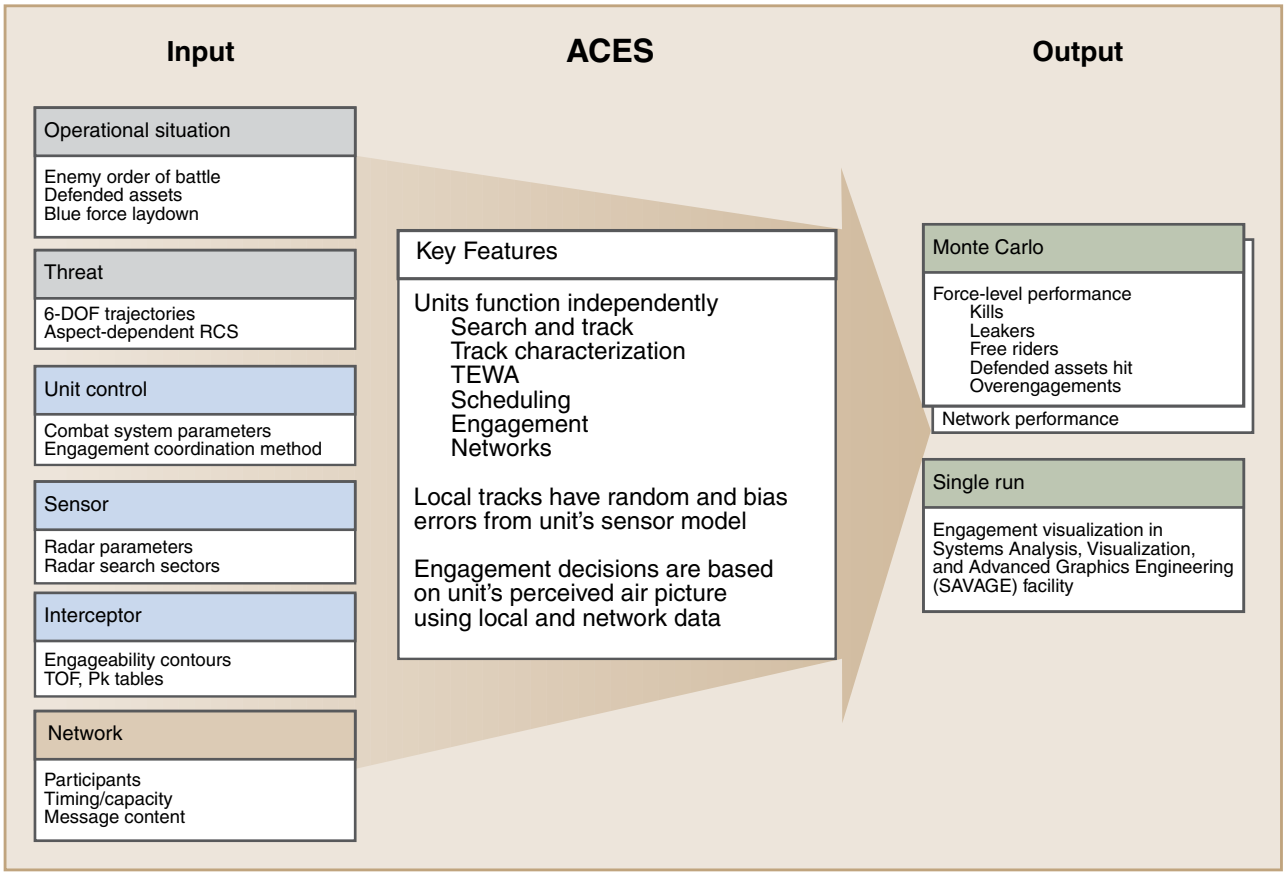

Figure 2. ACES inputs, outputs, and key features. Inputs in gray are associated with ground truth, blue with the units, and tan with networks. Outputs are shown in green. This color scheme also applies to Fig. 4.

ACES is primarily used to perform Monte Carlo analysis. Currently, three parameters are randomly varied across the Monte Carlo runs: the arrival time of the threats, the orientation of the ships, and the residual bias errors of the ships and networks. Statistics of force-level performance metrics can be used to evaluate the relative merits of different engagement coordination methods. Parameters that influence the engagement, such as network timing and bandwidth, can be varied to test the sensitivity of the results to these parameters. For a single run, ACES output includes the positions of physical objects as a function of time and logs of significant events, such as interceptor launches. These data allow a detailed analysis of the factors contributing to performance, and they have been used to visualize scenarios in the Systems Analysis, Visualization, and Advanced Graphics Engineering (SAVAGE) facility.

ACES STRUCTURE AND FUNCTION

ACES is a single-threaded application, written in C++ under the Linux operating system. The software is built around a generic framework for discrete event simulations.[3] A discrete event simulation advances time in increments from T_{now} to $T_{event(i)}$, where $T_{event(i)}$ is the time at which the next event (event i) is scheduled to occur. Events are kept in a priority queue, where they are sorted by T_{event}. If two or more events are scheduled to occur at the same time, fixed precedence rules determine their order. Most events in ACES, e.g., *Search, Track, ThreatAssessment*, and *Scheduling*, correspond to ACES functions. The simulation progresses through an event loop by retrieving event i from the event queue, updating every object in the simulation to $T_{event(i)}$, and processing event i. This pattern of `get next event, update objects,` and `process event` continues until there are no more events in the queue. Before starting this process, the simulation performs an initialization function in which initial events are added to the queue. During the event loop, new events may be added to the queue as a result of updating the states of simulation objects, or by processing the current event.

Figure 3 illustrates one aspect of ACES' object-oriented structure. Physical objects become more specialized as inheritance moves from left to right in the figure. An object inherits attributes and behaviors from its left, and may add new attributes and behaviors so as to more precisely define its purpose. For example, because all *Units* are capable of having *Sensors* and *Networks*, every *Ship* can have these attributes as well. Physical objects can be friendly (blue), hostile (red), or neutral (gray). Currently, red units are simply the land-based launchers that fire red missiles. The fact that missiles originate from launchers may seem insignificant, but in the future this behavior will allow air-launched threats to be launched from an aircraft. An Anti-Ship Cruise Missile will not just appear in the simulation environment, it will be launched from a platform, which itself may or may not be in track. Future versions of ACES may allow red units to have the same functionality as blue units.

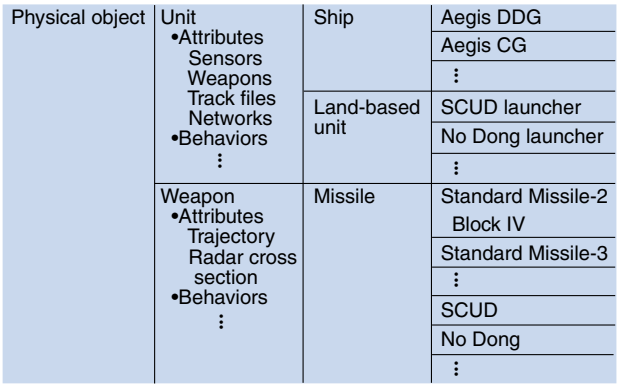

Figure 3. Physical objects provide an example of ACES' object-oriented structure. Units and weapons are two types of physical objects. Units can have attributes such as sensors, weapons, track files, and networks.

The focus of this article is the ACES unit, the fundamental entity that—through the use of its own resources—determines the tactical situation, makes engagement decisions, and executes engagements. The initial development has used the Aegis Weapon System as the representation of unit functional behavior. The Aegis-like implementation of some of the functions is discussed to provide the reader with an example of the intended scope of the function and, in some cases, the level of detail at which the function is implemented. ACES functions and data structures are listed at the top of Fig. 4.

There is a strong correspondence between these functions and ACES events, but the relationship is not one to one. The sequence in which the functions are listed in Fig. 4 is essentially the order of precedence of the events. The intent is that ACES functions, data structures, and data exchange mechanisms are flexible enough to allow various systems to be represented within the framework of the model.

The interactions between ACES functions and data structures are illustrated in Fig. 4. The top-level functions within the unit appear at the top of the diagram, arranged from left to right to follow the logical flow of an engagement from detection to engagement. The *TEWA* function is broken down into its subfunctions to properly depict the interactions with the data structures. The *Networks* straddle the bottom of the diagram. Some of the network functionality resides on a particular unit and some of it is common among the units. The networks are the only means of data exchange between units, and network participation can vary from unit to unit. The data structures serve

ACES functions :

Search and Track
- Sensor control
- Sensor detection
- Local track management
- Sensor coordination

Track Characterization
- Categorize track
- Perform cooperative identification
- Perform noncooperative identification
- Type objects
- Cluster objects

Threat Evaluation and Weapon Assignment (TEWA)
- Threat assessment
- Engageability
- Select local weapon
- Select force engagement

Scheduling
- Schedule local engagement

Engagement
- Initialize launch
- Interceptor flyout and midcourse guidance
- Terminate engagement
- Assess kill

Networks
- TDL (time division multiple access data link) data exchange
- TDL correlation
- SBN (sensor-based network) data exchange
- SBN correlation

ACES data structures:

Local Track File
Track states observed by local sensor; each track has a local track number.

Remote Track File
Track states received via the TDL network. Each track has a remote track number. A unit broadcasts data if it has reporting responsibility for that remote track number.

Composite Track File
Track states formed by combining data received via the SBN network. Each track has a composite track number. Units broadcast local data correlated to the composite track.

Threat Priority List
Tracks currently assessed to be a threat and their priority.

Candidate List
Engageability data for threats that are engageable by local weapons.

Force Engagement Schedule
Data required to schedule and execute an engagement. May contain entries from other units to support distributed engagement decisions. Data from other units are received via one of the networks.

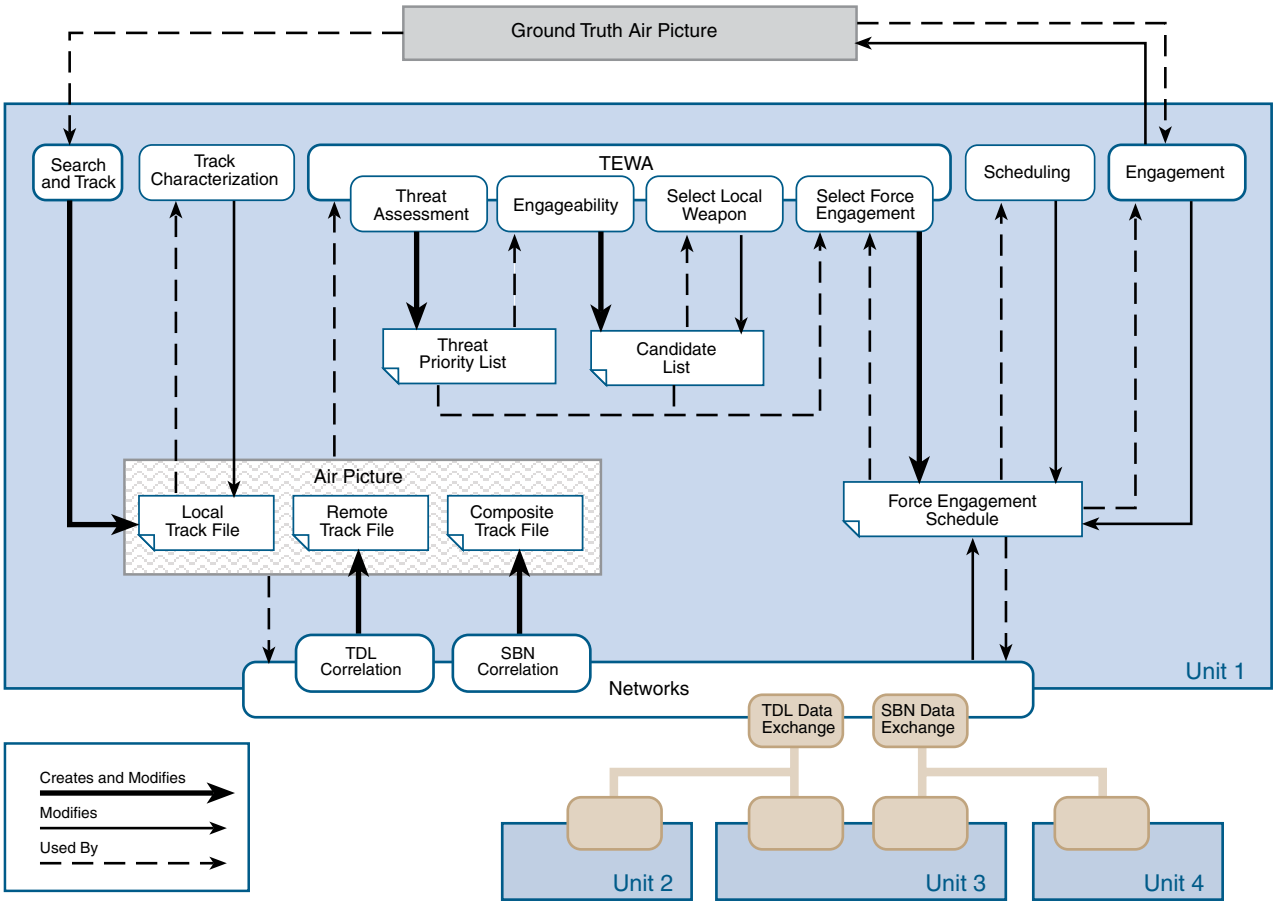

Figure 4. Interactions between ACES functions and data structures.

as the only means of interaction between the functions on a particular unit. Figure 4 indicates whether the function creates data entries, modifies them, or simply uses them. Each function is executed periodically, but the period used for each function can differ. The functional breakdown is identical on each unit, but the behavior can be unique.

The *Search and Track* function, discussed in the article by Bates et al., this issue, uses the position and orientation of the threat objects to calculate the signal-to-noise ratio of the radar return from a search waveform. The probability of detection is obtained using this ratio. A random draw is compared with the probability to determine if the threat transitions to track. If successful, a local track number is created and a track state is written to the *Local Track File*. The local track state is the sum of the actual track state and random and bias errors. Random and bias errors are also used to generate the track's covariance matrix. The *Local Track File* provides imperfect knowledge of the air picture because of the limited spatial extent of each unit's search sector, different timing for the search events, and the track state errors. Different units track different threats at different times and have different track states for threats they hold in common.

Other data in the *Local Track File* come from the *Track Characterization* function, which applies various tests to determine the track's category, primary identification (ID), and type. For example, the track may be declared to be of category *space-ballistic missile*, ID *hostile*, and type *separating*. *Track Characterization* subjects space tracks to further tests to determine if multiple objects originate from the same launch event. Any method may be used, but one created for Aegis will be described. Objects that are physically close are marked as belonging to the same cluster. One object from each cluster is designated as the primary object. Clusters may also be linked if other tests indicate that they originate from the same launch event. One of the primary objects from the linked clusters is chosen as the guidance track. Only the guidance track is used by subsequent functions. The goal is to choose one track per launch event for both TEWA and track reporting.

Local track data (guidance tracks) may be sent to other units in the simulation using either of two network models, which are discussed by McDonald et al. elsewhere in this issue. Units may participate in either network, both networks, or neither. Both networks allow the user to specify timing and bandwidth. The TDL carries the data used to create each unit's *Remote Track File*. The TDL correlation process attempts to determine if a local track and a remote track represent the same physical object. If the tracks correlate, reporting responsibility rules are used to determine if the local track is reported over the TDL network. Ideally, a single unit—the one with the highest-quality data—reports on a given remote track number. The SBN carries the data used to produce each unit's *Composite Track File*. If a local track correlates to a composite track, the local data are reported over the SBN network with the associated composite track number. All units with data correlated to a given composite track number report their data. All data received with a given composite track number are combined to produce the composite track. Another difference between the networks is that different bias errors are added to remote tracks and composite tracks to simulate the performance of each network's sensor registration (gridlocking) process.

Each of the track files gives an estimate of the air picture. The *Local Track File* estimates the air picture from one unit's perspective. The *Remote Track File* and *Composite Track File* both give a force-wide perspective because they contain tracks for objects detected by any unit in the force. In ACES, a track from any of these track files can be subjected to the *TEWA* function. In some cases only local data will be used. In others, remote or composite data can be used. The source of the air picture can even vary from function to function. For example, *Threat Assessment* may be performed using a remote track for an object that is not held locally, but *Engageability* may require local data. In this case, the local unit will perform a cued acquisition to establish a local track on the object.

The TEWA process is explained here assuming that local track data are the source for the air picture and that the track in question is a hostile space track. *Threat Assessment* for ballistic missiles begins by extrapolating the observed track state to predict the impact point. The covariance of the track is used to calculate an elliptical area of uncertainty around this predicted impact point. Defended asset locations are also specified as elliptical regions. The unit's Defended Asset List contains the locations of the defended assets and their priorities, a numerical indication of how important they are to defend. The impact point's area of uncertainty is compared to the ellipse associated with each defended asset. If there are any intersections, the track is assessed to be a threat and assigned a threat priority equal to the highest priority of the assets threatened by that track. This information is written to the *Threat Priority List*.

The *Engageability* function determines if any of the unit's weapons have capability against the threat. This amounts to determining if the extrapolated path of the threat passes through the volume in space that can be reached by the interceptor. If the trajectory *does* pierce the volume, the two endpoints of the intersection are used to calculate the earliest and latest time-to-intercept. Interceptor time-of-flight data are used to estimate the corresponding launch times. If it is not too late to launch, the threat is engageable by the interceptor and the engageability data are written to the *Candidate List*.

If the threat is engageable by more than one weapon in the unit's arsenal, the *Select Local Weapon* function uses a set of criteria to select the "best" weapon. A variety of criteria could influence the decision, such as doctrine, inventory, predicted effectiveness, intercept time, etc. This logic is not required in ACES version one because ships are modeled with only one weapon type.

Select Force Engagement is a function unique to ACES. Its existence anticipates that the threat might be engageable by multiple units. If the unit is alone, or acts like it is alone, this function is trivial: the unit selects itself to shoot. In a more general case, the *Force Engagement Schedule* (FES) contains entries from every unit that can engage the threat. Nonlocal *FES* entries are the result of engagement coordination messages exchanged via the communication network. The *Select Force Engagement* function then chooses among the possible shooters. It executes periodically and provides a means for a local unit to change its engagement status based on the reported capabilities of other units in the theater. Possible implementations of this function are discussed in the article by Moskowitz et al., this issue.

The *Scheduling* function uses the data on the *FES* to create the launch time for the shot. A trial planned launch time is estimated and subjected to a number of tests. One test ensures that the number of missiles simultaneously in flight does not exceed a specified maximum value. To satisfy this test, the planned time to launch can be adjusted by moving it later in time, as long as the latest time to launch is not exceeded. Another test compares the number of simultaneous radar discrimination events to a specified maximum value. Shots failing this test will not be scheduled. Shots that can be scheduled have their final planned time to launch written to the *FES*, and their engagement status is set to *SCHEDULED*. Scheduling occurs periodically to react to changes in the air picture.

The *Engagement* function performs a variety of tasks to execute an engagement, from launch to kill assessment. Launch initialization reconciles the planned time to launch on the *FES* with launcher availability to determine the actual launch time. When the interceptor is launched, the shot's engagement status is changed to *IN_PROGRESS*. Using some engagement coordination schemes, status changes cause an engagement status message to be sent to the other units participating in the network. Additional random draws determine if the midcourse guidance is successful and if the interceptor fails in flight. If such a failure were to occur, it would do so at a randomly chosen time. If the interceptor survives until the actual time to intercept, the engagement is terminated.

The outcome of the engagement is determined by probability draws representing the likelihood of designating the lethal object and the likelihood that the interceptor will kill the object. These probabilities are user specified. They could be constant values such as zero or one for diagnostic runs, or system-specific threshold or objective values for trade studies. They could be tables of values estimated by high-fidelity missile simulations for more detailed analysis. The object state is modified to reflect the chosen outcome. The unit's perception of the outcome, determined by the *Assess Kill* function, depends on this outcome but is not identical to it. A table of probabilities is used to map the possible outcomes to the possible kill assessment results. ACES units operate on perceived reality from detection through kill assessment.

The functions in Fig. 4 have been described in the context of a single ship. It is important to reiterate that this functionality, or any subset of it, can be created for any number of units operating in a theater, enabling engagement coordination methods to be modeled and evaluated.

SUMMARY

ACES provides a suitable environment to evaluate engagement coordination for situations in which multiple units have capability against multiple threats. The generic structure and function of ACES' unit representation allows the same framework to be used to model a variety of TAMD participants. The detect-control-engage performance of each unit is modeled with medium fidelity, representing key features that affect engagement outcome. Each unit can have unique behavior and act independently on its perceived air picture. Network models allow the air picture fidelity, network timing, and bandwidth to be varied. Reasonable execution time permits Monte Carlo analysis to be performed, as well as single runs for visualization and detailed analysis. ACES is a powerful, flexible tool for evaluating TBM engagement coordination performance, and its utility will grow as it is expanded to model multi-mission scenarios.

REFERENCES

[1]Sommerer, S., *TBM Engagement Coordination Model*, ADS-00-054, JHU/APL, Laurel, MD (Aug 2000).
[2]Pollack, A. F., and Chrysostomou, A. K., "ARTEMIS: A High-Fidelity End-to-End TBMD Federation," *Johns Hopkins APL Tech. Dig.* **22**(4), 508–515 (2001).
[3]Horstmann, C. S., *Mastering Object-Oriented Design in C++*, John Wiley & Sons, Inc., New York (1995).

ACKNOWLEDGMENTS: The development of ACES is supported by the Office of Naval Research and the Program Executive Office Theater Surface Combatants, Navy Theater Wide Division. The authors would like to acknowledge the contributions of their fellow ACES team members: S. R. Allen, C. W. Bates, J. F. Engler, R. J. Gassler, B. L. Holub, S. A. Hyer, E. M. McDonald, S. Moskowitz, B. L. Paulhamus, R. A. Phillippi, and K. E. Shafer.

THE AUTHORS

MICHAEL J. BURKE received a bachelor's degree in mechanical engineering with a minor in mathematics in 1980, and a master's degree in mechanical and aerospace engineering in 1982, both from the University of Delaware. He has continued his education with postgraduate studies at Catholic University of America and the University of Maryland at College Park. Mr. Burke worked for the David Taylor Research Center and for Noise Cancellation Technologies Inc. before coming to APL in 1995. He is currently in the Air Defense Systems Engineering Group of ADSD. His e-mail address is michael.burke@jhuapl.edu.

JOSHUA M. HENLY is a computer scientist in the Air Defense Systems Engineering Group of ADSD. He received a B.S. in computer science from Drexel University in 1998, and an M.S. in computer science from The Johns Hopkins University in 2001. Since joining APL in 1998, he has written or managed several software simulations for Navy Ship Self-Defense and Theater Air Missile Defense. Mr. Henly's current interests include distributed computing and developing graphical user interfaces for software simulations. His e-mail address is joshua.henly@jhuapl.edu.