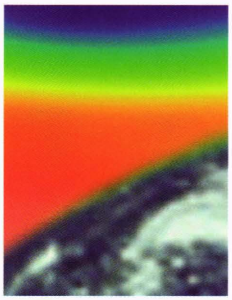


Operator Support Concepts for Tomahawk Strike Management

Mark D. LoPresto, Ann F. Pollack, John Florence, Robert C. Ferguson, and Ian E. Feldberg

Modeling and simulation play important roles in the exploration of future system capabilities. Plans for the next generation of Tomahawk include a strike management capability (the ability to control missiles after launch). The Laboratory has developed strike management conceptual and functional prototypes that have helped to refine system concepts and requirements, stimulate user feedback, and provide an environment for early algorithm development. The functional prototype has become a valuable tool for examining broader issues of weapon employment and has been incorporated into a distributed interactive simulation environment to help refine real-time control of precision strike weapons as system-level challenges grow in complexity.

INTRODUCTION

As the technical direction agent for Tomahawk, APL is helping the Navy to shape the direction of the cruise missile program. One example of this effort is the Battle Group Strike Warfare Coordination (BG STC) initiative, which is intended to improve overall strike effectiveness through better coordination of strike warfare systems. Tomahawk, which constitutes the U.S. Navy's long-range cruise missile arsenal, is the initial focus of this undertaking.

Originally conceived as a strategic system for delivering nuclear warheads, Tomahawk is now a conventional weapon system appropriate for tactical environments. Tactical environments necessitate greater responsiveness and adaptability to the dynamics of strike warfare, since target priorities are likely to change as battle damage accumulates and objectives shift. The opposing force's defenses, once alerted, may be moved, and new threats to the strike force may emerge to challenge routes of attack. Timely awareness of strike effectiveness and the ability to control how strike operations unfold are therefore important objectives. These objectives are prompting a change from the current fire-and-forget employment mode for Tomahawk.

The Tomahawk Baseline Improvement Program (TBIP) is developing capabilities for the next generation of Tomahawk to improve system response time and enhance employment flexibility. The upgrade, scheduled to reach operation by the turn of the century, will include two-way communications between missiles in flight and command and control nodes, as shown in Fig. 1. Communications from the missiles will provide real-time health and status reports and battle damage indications. Return communications will allow an operator to divert, or flex, missiles to alternate targets based on the effectiveness of previous missiles. It will be possible to judge aimpoint damage using single-

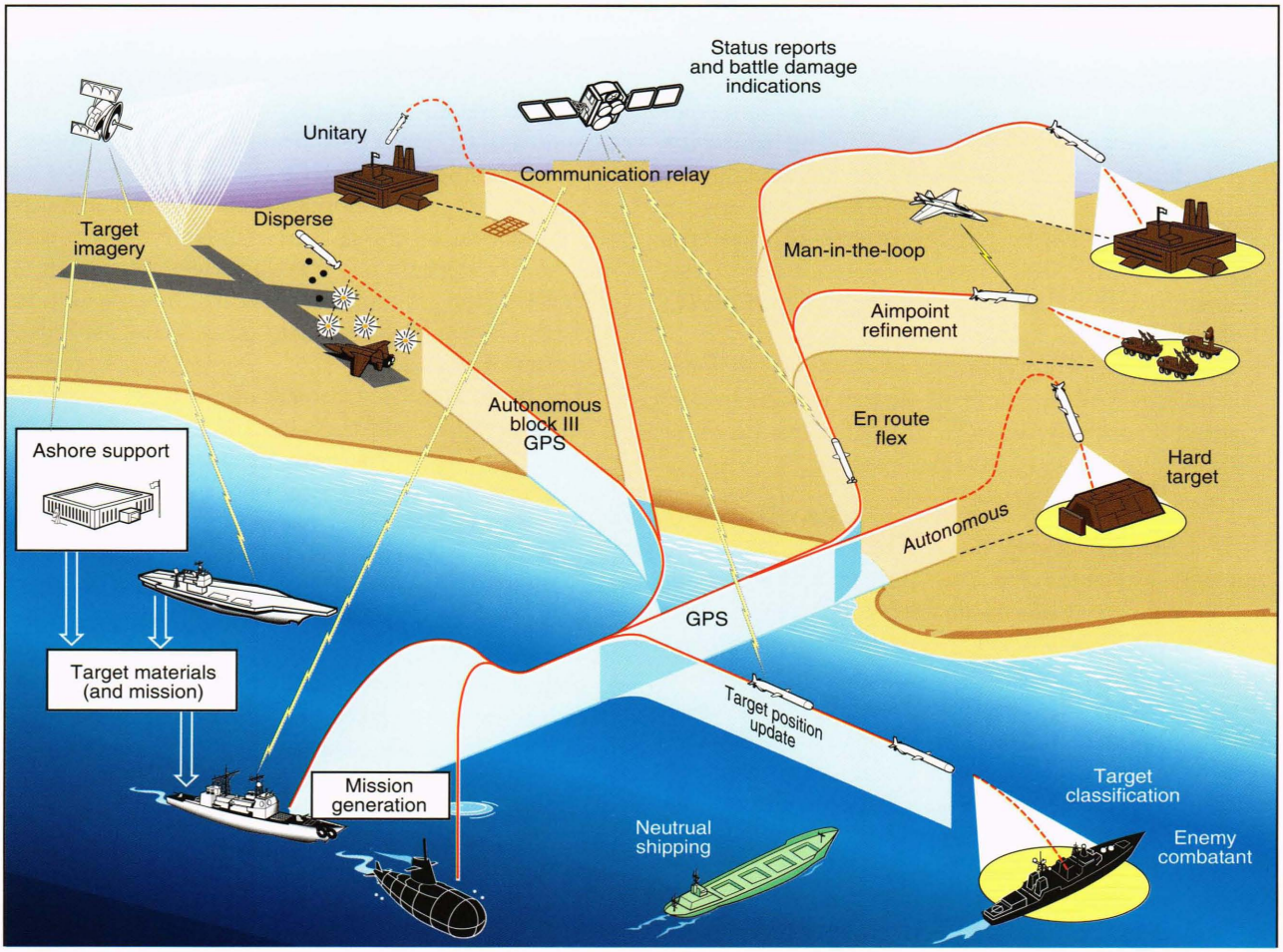


Figure 1. Tomahawk Weapon System Baseline IV concept of operations. (GPS = Global Positioning System.)

frame video images from a missile's terminal-imaging infrared seeker. (The term aimpoint in this article is synonymous with aiming point.)

The two-way communications capability permits real-time monitoring and control by a Tomahawk Strike Coordinator (TSC) or a delegated representative.[1] If necessary, the TSC will be able to reallocate resources by diverting missiles to alternate targets using an en route flex command to maintain coverage of critical targets, conserve missiles, and permit recovery from missile casualties. Such functions have been termed strike management.

Recognizing that proper utilization of such capabilities is critical to their success, the Program Executive Office for Cruise Missiles and Unmanned Aerial Vehicles (PEO[CU]) began to define operator support concepts through the BG STC initiative. A fundamental objective was to demonstrate that the new capabilities could be made manageable for operators. In response, the Laboratory created a computer environment for visualizing strike management through prototypes that present the look and feel of potential capabilities. We use the prototypes to obtain comments and suggestions from prospective operators on interface design, usability, and essential features. In addition, the computer environment provides a context for algorithm formulation. To demonstrate system-level implications of advanced strike and cruise missile concepts and capabilities, the Laboratory is integrating the products of this task into broader simulation environments.

DESIGNING PROTOTYPES

In software engineering, prototypes are especially useful for solidifying requirements (by the procuring agent), understanding design trade-offs (among the technical community), and influencing system design (by the eventual customer; in this case, the Fleet). Prototypes provide an avenue for uncovering and solving problems before schedule and cost become the main determinants of development. Prototypes can be evolved in four basic stages leading toward an operational system as follows:

1. **Conceptual prototypes**, which focus mainly on the human–computer interface, show the look and feel of

the eventual system; such prototypes may be automated.

2. **Functional prototypes** provide the internal organization (data structures and architecture) and incorporate candidate algorithms and processing behind the interface to give the system limited functionality.
3. **Fieldable prototypes** provide the major functions of the eventual system reliably enough to insert the prototype into an exercise or operational environment for initial evaluation. Some operational benefit is also obtained.
4. **Operational prototypes** provide virtually all functions in a robust and reliable system that is often an interim capability introduced in anticipation of future delivery through a formal program.

Prototypes supply useful information throughout their multistage development. They facilitate definition of technical characteristics through hands-on evaluation and permit user feedback before an investment is made in full-scale development. Customers can validate system requirements and usability by interacting with prototypes that approximate the look and feel of the eventual system. Questions arising from this process help to refine system concepts further. A mature prototype can serve as an interim solution that does not require a lengthy acquisition process to remedy shortfalls in fielded capabilities.

This article describes the application of conceptual and functional prototypes to Tomahawk strike management. The conceptual prototype models the look of operator support by simulating the content and presentation format of key information on a typical strike management display. The functional prototype conveys the feel of operator support by enabling options for the functional support layer behind displays (including data structures, functional flow, algorithms, decision support features, and controls for governing flow and applying decisions) to be explored.

DEVELOPMENT PHILOSOPHY

Early prototypes have helped to mature Tomahawk strike management concepts through a spiral development cycle. Top-level requirements were written in a system-level specification for the Tomahawk Weapon System Baseline IV. We began designing the prototypes by storyboarding ideas for strike management to produce a series of sample displays illustrating possible operator support concepts. After review, the ideas were transformed into an automated prototype that was initially implemented and evolved on the Macintosh and then progressed to the current 486-based system. Throughout development, the team concentrated on designing a flexible, easily extendable system. This philosophy accommodated changing requirements and allowed rapid modifications on the basis of sponsor and user feedback.

Object-oriented techniques were used to implement a flexible system that could readily be adapted and expanded to represent new ideas and concepts. Object-oriented techniques model the problem domain as a set of objects and the relationships among them. Objects are abstractions of things in the problem domain that relate to the system's responsibilities. A class is a set of objects that have a common structure and behavior. In general, classes are static, and objects are particular instances of classes.

The first advantage of object-oriented development is design stability. Although specific requirements may change, any Tomahawk strike management system will have objects for representing missiles, shooters, missions, and others. Object-oriented development also permits data encapsulation, since each object has an associated set of attributes and actions. In this manner, data are packaged or encapsulated with particular objects, thus helping to manage system complexity.

Object complexity evolves with the prototype. Initially, the prototype represented objects at a very high level with few details, which provided an overall perspective useful for refining capabilities and identifying new requirements. As the prototype unfolded, the objects afforded a framework for continued development, since they could easily be added, deleted, or modified.

In addition to employing object-oriented techniques, the development philosophy stressed code reuse, which was particularly important in the interface development. A commercial product (Symantec Think C) was used to provide the basic user interface objects on the Macintosh, such as windows, menus, and dialogue panels. Drawing on this existing code enabled the team to concentrate on implementing Tomahawk strike management objects. Code reuse allowed rapid progress on the specific problem domain within a generic interface framework.

The aim of the architectural philosophy was to achieve the flexibility and growth potential emphasized in the software development philosophy through a modular approach. Thus, the logical functions of the prototype were separated into two elements: the strike management display and the simulation driver. The strike management display approximated the system to be deployed, and the simulation driver provided the inputs and responses for stimulating the human–computer interface. In an operational system, real-world inputs would replace the driver. To mirror this discrete functionality in the prototype architecture, separate processors were used for the two elements (Fig. 2). Interaction between the elements was through messages across a network. This architecture was designed to enhance modularity and set the groundwork for developing an operational system.

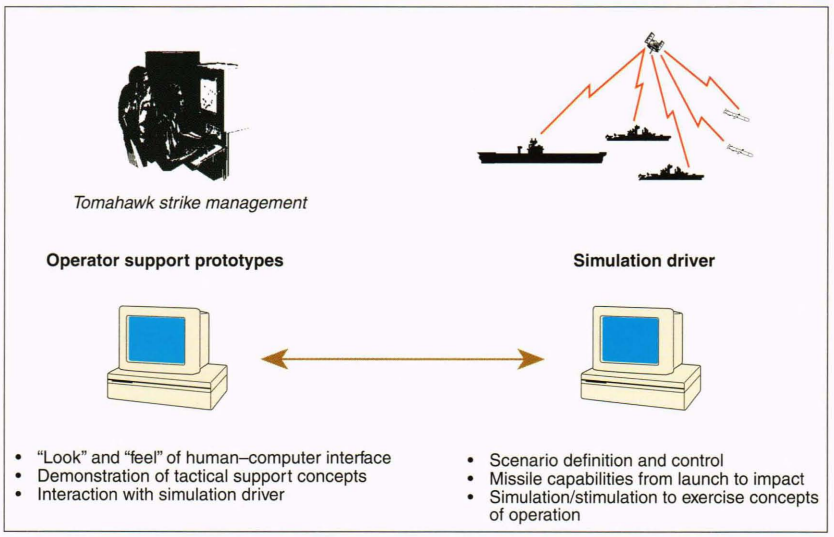


Figure 2. Top-level hardware and functional architecture of the operating support environment.

CONCEPTUAL PROTOTYPE

The initial conceptual prototype consisted of a series of computer screen viewgraphs illustrating operator displays for each of the anticipated strike management functions. Figure 3 presents a sample display. Concept refinement occurred through weekly meetings with the Navy sponsor. After only a few such meetings, the prototype displays were firm enough to permit work to begin on the computer implementation of the functional prototype.

The displays in the conceptual prototype are simple and intuitive. Graphics are used to convey information on a gross scale, and text and tables are employed to organize and present fine details. The geographic display presents the top-level information required to monitor the strike, allowing the operator to retrieve more detailed information if needed. A consistent color scheme is used to present essential strike information. Green is used as a positive indication (e.g., receipt of a missile status message) and blue denotes completion (i.e., the missile has reached the target). Nonstandard conditions are presented in red and yellow, depending on the degree of urgency or effect on the overall strike plan.

The conceptual prototype illustrates a hands-off strike management style. As long as the strike proceeds nominally, results are presented on the display in green and blue. The TSC monitors progress but is not

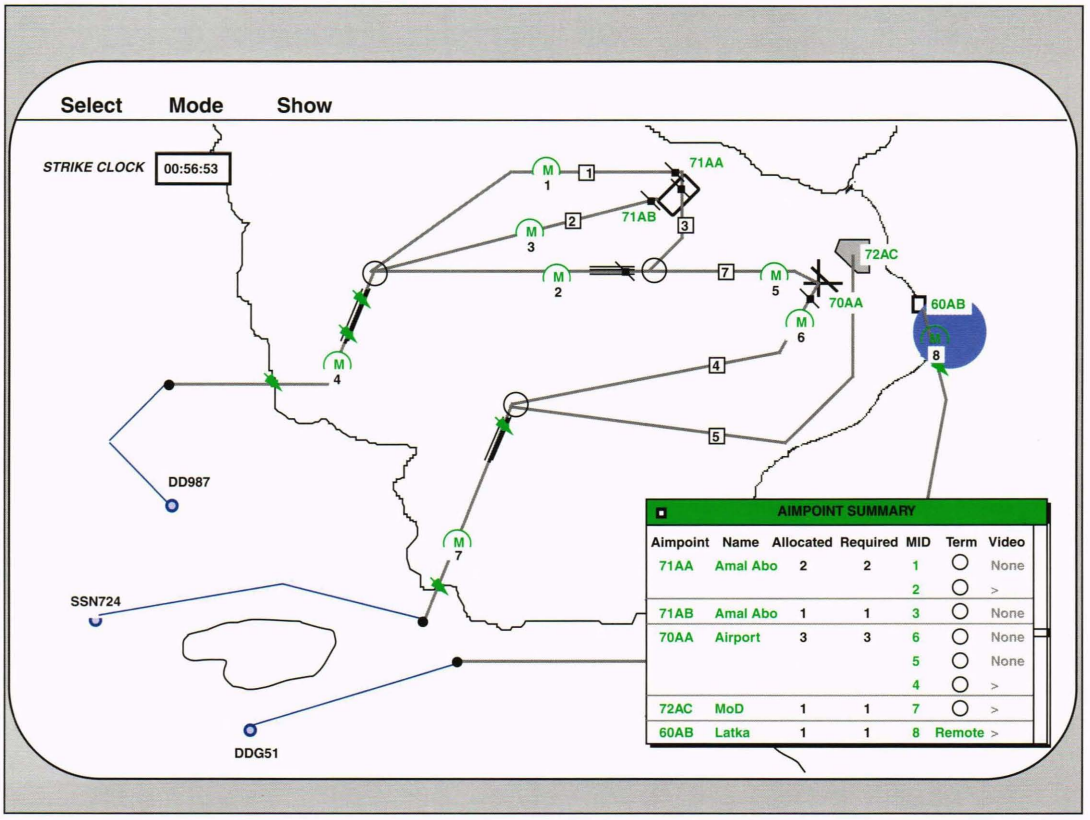


Figure 3. Strike management display for the conceptual prototype. (MID = missile identification.)

required to take action. Intervention is required only when there is a deviation from the plan, causing a yellow or red display warning to appear. Such anomalous conditions invoke system recommendations that prompt action from the TSC. Suggested actions are limited to a few well understood choices. This strike management style reinforces a goal of the prototype development effort: to demonstrate a straightforward user interface.

FUNCTIONAL PROTOTYPE

The modular architecture of the prototype design environment allows the functional components to be developed in parallel with conceptual prototypes. This modularity promotes rapid prototype development. The functional prototype consists of the strike management user interface for monitoring and controlling missiles and a simulation driver for generating missile messages and responding to strike management commands.

The Simulation Driver

The simulation driver is an important component in the functional prototype. Without the driver, the utility of the functional prototype would be limited to providing the appearance of operator interfaces without the feel. With the driver, the functional prototype offers an environment that stimulates realistic response from the operator support prototype.

Early driver development was based on the Tomahawk Object-oriented Trade-off Evaluation Model (TOTEM), which APL created as a tool for exploring advanced Tomahawk concepts. It included a graphical user interface (GUI) that permitted parameters to be modified easily for defining a scenario and analyzing different options. With a GUI, the operator controls the computer by manipulating icons on the computer screen rather than entering text instructions on a command line. Such directly manipulatable, user-friendly interfaces are familiar and intuitive for Macintosh, DOS Windows, and X Windows users. Figure 4 illustrates a TOTEM scenario development screen. The object-oriented domain model allowed the program to be adjusted easily for exploring potential new capabilities. Because of this inherent flexibility, TOTEM was a logical starting point for the simulation driver and served as the basic domain framework and interface to build upon.

The primary functions of the simulation driver are to stimulate the strike management system with representative missile messages and to mimic the response of missiles to strike management commands. Although

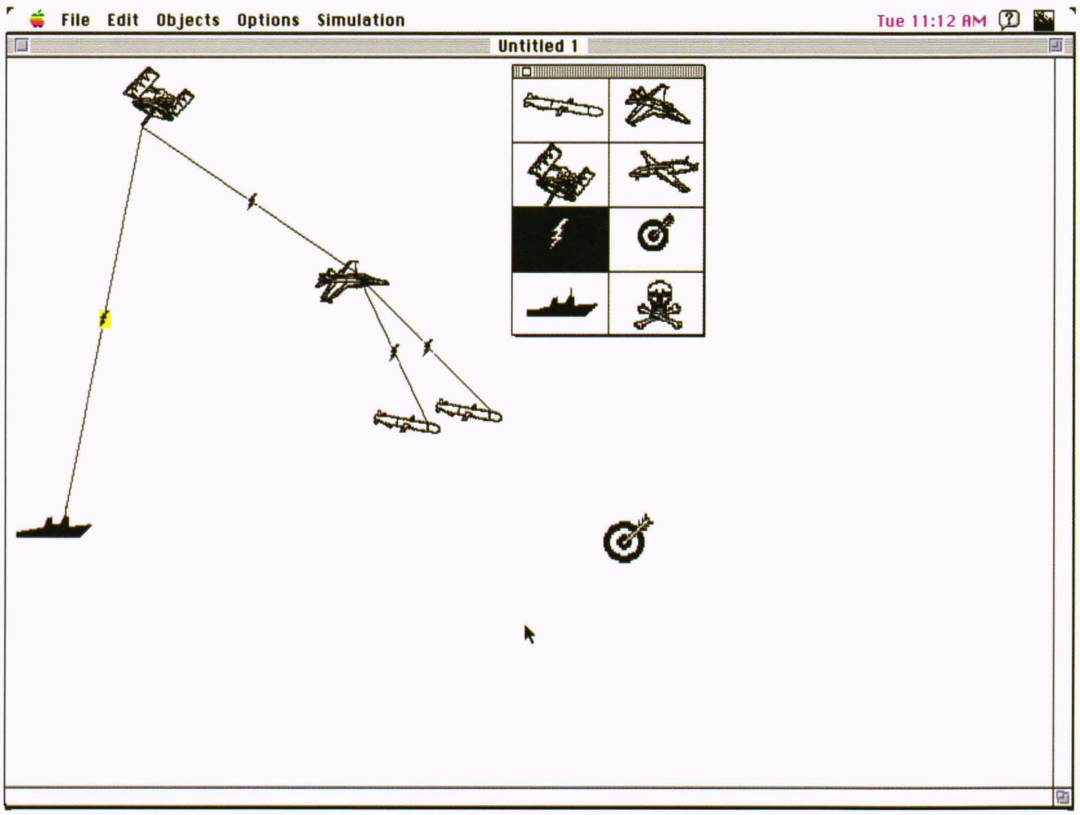


Figure 4. Scenario development screen for the Tomahawk Object-oriented Trade-off Evaluation Model (TOTEM).

the driver simulates missile actions from launch to impact, the emphasis is on message exchange and processing. All communication between the driver and the strike management module is through message passing. In the Macintosh-based prototype, messages were passed over a network utilizing the Apple Program-to-Program Communications Toolbox, Apple Events, and the Apple Event Manager. Apple Events uses Apple's message-passing protocol for program-to-program communications. Apple Events provided a built-in mechanism for exchanging data between the strike management module and the driver, simulating the transmission of messages between the Tomahawk Strike Manager and missiles in flight. In the NEXTSTEP version of the functional prototype, message passing was achieved using Unix remote procedure calls in a client–server architecture.

The driver is an event-stepped simulation with a variable execution rate. Simulation events are inserted into an event list in sequence of execution order. Events are removed from the top of the event list at the user-specified execution rate unless constrained by the performance of the computing platform. The driver manages event processing alongside user input handling and display processing within the application, allowing the user to alter the execution rate before or during simulation.

Simulation events include missile events, threat events, target events, and Global Positioning System (GPS) events. Each missile is initialized with a launch event. After the missile begins flight, a booster separation event is scheduled, and so on, through each sequential mission event. Anomalous missile events can be generated at random, under user control, or based on proximity and state of threats. These events allow the strike management operator support prototype to handle difficult scenarios.

Strike Management Operator Support

The strike management user interface for the functional prototype was developed using Apple's Hypercard application program, which was then replaced by SuperCard (from Silicon Beach). Hypercard and SuperCard, both of which support Apple Events, are innovative applications for programming on the Macintosh. They supported the rapid growth cycle of the prototypes with their graphical user interface development and message-passing capabilities.

Hypercard and SuperCard work similarly, using Macintosh's HyperTalk interpreted programming language. The SuperCard environment was selected because of its color display support. Graphical entities on the screen represent objects in the program that interact through the exchange of messages. When an object receives a message, it may perform some service, sometimes using data passed with the message. In addition, the object may send out its own message or pass the original message farther along a hierarchical message-passing structure to be handled by the program or the operating system.

Strike management interface development started with the set of viewgraphs that constituted the conceptual prototype. The basic display was a geographic view of the area of interest that served as the default window when the program was started. This display inspired new and expanded features, such as pop-up information windows that remained open while the user clicked on an object, pull-down menus for setting up and tailoring the display, and special keyboard equivalents for common user actions.

The initial interface development phase, although short, was probably the most productive. The features of SuperCard made it easy to dispense with mundane but potentially time-consuming tasks such as implementing window management services and handling mouse tracking and clicks. Instead, developers concentrated on introducing features relevant to Tomahawk strike management. Attention focused on how strike management would work, not just how it would look.

Four top-level strike management functions (strike preview, strike monitoring, strike control, and strike assessment) were drawn from the Tomahawk Weapons System Baseline IV System Specification, the document specifying the next generation of Tomahawk. How these functions could be performed became apparent through using the rudimentary strike management display. The geographic display was well suited to strike preview (simulating the strike before execution) and strike monitoring (viewing strike progress during execution). The information necessary to support strike control (diverting missiles in flight) and strike assessment (gauging strike success from reported results), however, was not clearly presented on a geographic display. Early attempts at performing these functions with the preliminary prototype led to additional displays to support these functions.

In the initial implementation, the strike control function was triggered by a reported missile failure. Strike plans require a specific number of missiles to be allocated to each aimpoint. If a missile failure results in the quantity of missiles allocated to an aimpoint falling below the number required, a set of actions is needed to recover from the failure. Recovery is accomplished by diverting missiles or launching backup missiles, or by doing both, to restore complete aimpoint coverage.

In the Macintosh version, the recovery from a missile failure was scripted; the system response was hard coded and only made sense if a certain missile failed before a certain point in the strike. The failure manifested itself on the display through changes in the

colors of key objects (e.g., the missile icon turned from green to red) and audible alerts. The failure appeared to cause the system to calculate and rank possible recovery options.

It was difficult, however, to show recovery options on the same geographic display that was indicating the current state of the strike. A key concern was that the TSC would confuse recommended actions with the actual state. Instead, separate dialogue panels and alert boxes were used to present recovery options unambiguously to the TSC, as shown in Fig. 5. In the Strike Preview mode, the TSC can fail a missile at will to practice recovery procedures. When an actual strike is being monitored, the failure has to be initiated at the simulation driver and passed to the strike management module via Apple Events.

The TSC performs strike assessment by interpreting the missile status messages and viewing terminal imagery (when available) for any battle damage indications. Throughout the interface development, but in particular when considering strike assessment, it became apparent that the real focus of the TSC should be aimpoint coverage rather than individual missile health and progress. The geographic display is not well suited to this aimpoint view of the strike. For example, aimpoint location on the geographic display does not necessarily convey aimpoint priority. Consequently, in addition to the geographic display, the prototype provides an aimpoint summary window that draws the attention of the TSC to aimpoint coverage. The aimpoint summary consists of a table of prioritized aimpoints (as determined from the strike plan). For each aimpoint, the table shows the missiles (by missile identification number) heading to that aimpoint, the quantity of missiles required, and the number of currently allocated missiles. The table also provides access to terminal imagery when it arrives at the strike management module.

Improving the Functional Prototype

The functional prototype, consisting of the integrated simulation driver and strike management user interface, was demonstrated to the sponsor in July and August 1993, just 4 months after the effort began. On the basis of sponsor feedback, APL set out to build a more robust prototype for demonstrating TBIP capabilities to Navy operational personnel. We developed

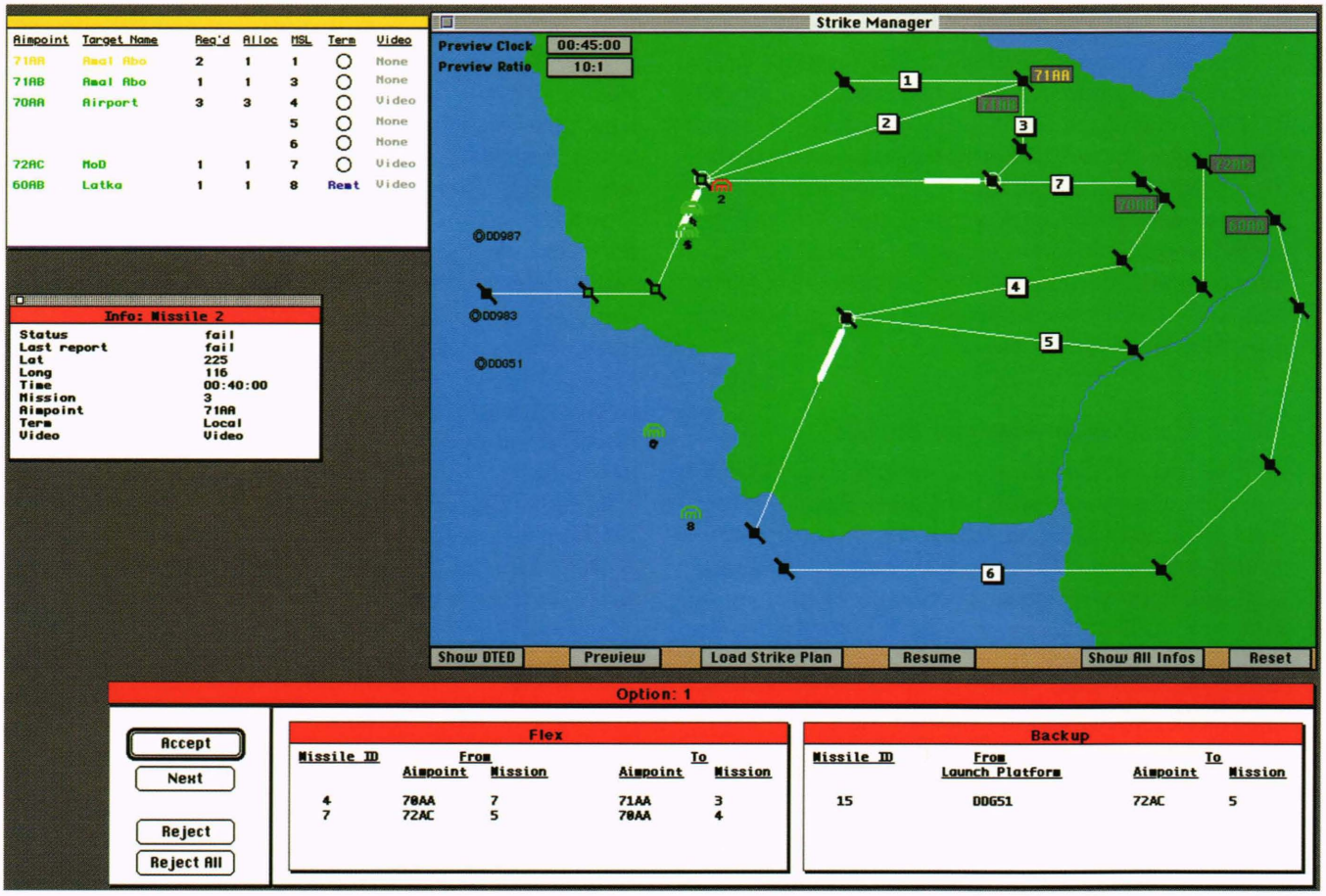


Figure 5. Sample strike management display using SuperCard with a Macintosh. (Alloc = allocated, MSL = missile, Remt = Remote, DTED = Digital Terrain Elevation Database, ID = identification.)

the new prototype using an IBM-compatible PC. The NEXTSTEP operating system was selected because it provides an object-oriented graphical user interface and supports development of object-oriented applications. The support structure includes sophisticated tools for building graphical user interfaces to programs incorporating Objective C, which, like C++, is an object-oriented programming language that can execute standard C source code.

The transition to NEXTSTEP was relatively straightforward. Although SuperCard is not an object-oriented development environment, it exhibits several characteristics of an object-oriented system, such as encapsulation, modularity, and message passing. Much of the application structure developed for SuperCard remained valid in the NEXTSTEP version. The behavior of key objects and their interactions with other objects in the application were well known. In short, the analysis and much of the existing design directly carried over to NEXTSTEP; most of the revamping effort concentrated on programming the interface. Figure 6 shows an example of the current strike management user interface based on the IBM PC.

The NEXTSTEP revision offered the opportunity to incorporate a growing list of enhancements and alternative displays. Suggestions for improvements, extensions, and new capabilities frequently emerged from demonstrations. Alternative (i.e., nongeographic) displays for tracking missile progress and the ability to fail any missile at any point in the strike were two suggestions high on the list of frequently requested capabilities. These and other desired capabilities required developing decision aids for integration into the prototype.

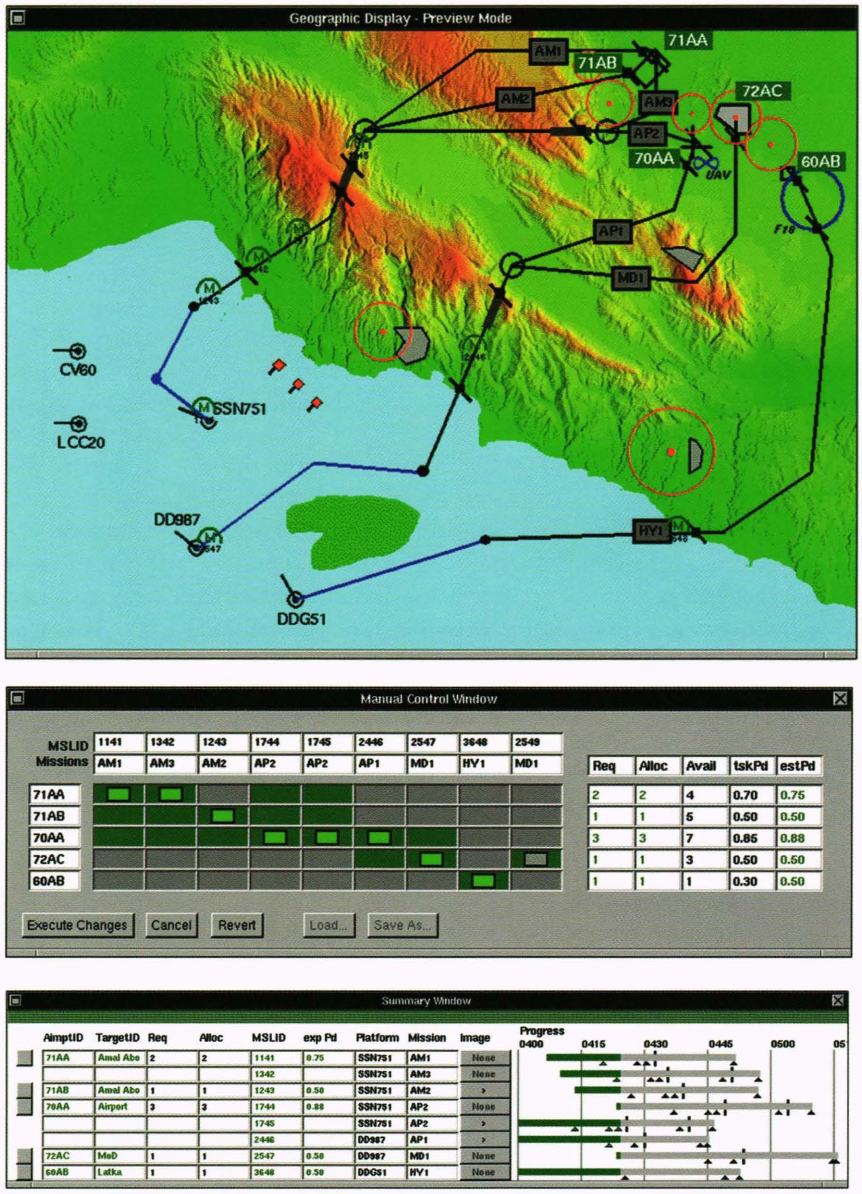


Figure 6. Sample strike management display using NEXTSTEP with an IBM PC. (AimptID = aimpoint identification, MSLID = missile identification, exp Pd = expected probability of damage, RAP = reporting action point, Req = required, Alloc = allocated, Avail = available, tskPd = task probability of damage, estPd = estimated probability of damage.)

Incorporating working decision aids into the prototype offers several advantages, such as prompt feedback to help refine complicated heuristics, information for use in determining data structure design (operator decisions are quite complex and can require intricate data structures), and the possibility of translating lessons learned from early algorithm implementation directly into the final product. In addition to lessons learned, the software itself can be incorporated in the final system. The Laboratory adhered to strict ANSI C standards when implementing the three decision aids introduced into the prototype. These were developed using a Macintosh Quadra 800 and subsequently ported to the IBM PC for integration with the rest of the system.

The first decision aid, aimpoint coverage, computes actions necessary to reallocate missiles to assure proper coverage of all aimpoints. The algorithm is triggered when the quantity of missiles designated for an aimpoint is less than the number required. This situation can arise from a missile failure, a previous reallocation, or an increase in required missiles based on an operator's assessment of previous damage. Each of these situations requires the system to consider different factors.

In responding to a failure, the system must consider if a backup missile should be launched or whether one of the missiles en route can be redirected to cover the failed missile's aimpoint. Clearly, a recovery solution that conserves resources is desirable (i.e., a solution using the minimum number of backup missiles). If an aimpoint is not sufficiently damaged, missiles headed toward lower-priority aimpoints may need to be redirected, depending on timing constraints. This action can deprive other aimpoints of coverage and set a cascade of missile redirections in motion. Eventually, backup missiles may need to be launched to cover an aimpoint, including the original underdamaged aimpoint.

The aimpoint coverage algorithm assists the operator in reaching a decision by enumerating all of the appropriate responses to the given situation. Each response is ranked using a series of heuristics and presented to the operator for consideration. The implemented heuristics are simple but effective in ranking solutions and include the following:

1. A failure recovery solution in which no backup missiles are launched is better than a solution in which backup launches are required (conserve resources).
2. A failure recovery solution requiring few missile diversions is better than a solution requiring many diversions (minimize the required number of reallocation moves).
3. A failure recovery solution requiring that a missile be diverted from a low-priority aimpoint is better than a solution in which a missile is diverted from a higher-priority aimpoint (divert missiles from low-priority aimpoints).

The aimpoint coverage algorithm is being expanded to reallocate missiles in response to positive damage assessments. In this case, to maximize the probability of meeting strike objectives, the system will seek to redirect missiles still flying to the destroyed aimpoint, which will eliminate the need to consider launching backup missiles. A key issue is how to redistribute the extra missiles. Some possibilities are to divert all extra missiles to the highest-priority aimpoint available, to distribute missiles evenly across all remaining available aimpoints, or to redistribute missiles based on a weighted priority of remaining available aimpoints (e.g., divert two missiles to the highest-remaining-priority aimpoint and one to the remaining second-highest-priority aimpoint).

Although the aimpoint coverage algorithm works well for the limited scenario used to demonstrate the prototype, the number of missiles and diversion and backup options in an operational system can be several orders of magnitude greater. In an operational environment, any practical algorithm may need to be able to generate and rank many possible solutions. The Laboratory is investigating other methods, such as genetic algorithms* and fuzzy logic,† for the next generation of aimpoint coverage algorithms.

The second decision aid, strike effectiveness, allows an operator to monitor the progress of a strike using the probability of target damage as a measure of overall strike effectiveness. By monitoring the strike progress, corrective actions can be made to compensate for those areas of the strike that are not meeting objectives. For example, if the current probability of damage to a given target is below the specified level (perhaps owing to failures or diverted missiles), the user can divert another missile from a lower-priority aimpoint to compensate. The conditional probability of damage to an aimpoint is calculated on the basis of the expected level of damage to an aimpoint from a single missile and the number of missiles going to a given aimpoint. The expected level of damage from a single missile is mission dependent and is calculated before strike execution.

The strike effectiveness algorithm dynamically computes the conditional probability of damage for each aimpoint as the strike progresses. For example, if a missile fails, the algorithm updates the conditional probability of damage values for all aimpoints. The updated values are presented to the operator for evaluation using the now familiar color scheme. Values that meet or exceed tasking are shown in green, nonzero values below the specified damage level are shown in yellow, and null values are shown in red. A null value will occur when no missiles are headed to an aimpoint. This decision aid allows the operator to evaluate a strike's effectiveness quickly during execution.

The third decision aid, hit probability, computes the probability that a Tomahawk will hit designated structures in the target area. It is designed for both collateral damage evaluation and target hit maximization. The algorithm accepts as input a series of disjoint, simple polygons specified by their vertices. These polygons represent buildings and other structures in the target area as viewed from above. The algorithm also accepts as input the parameters of a bivariate normal probability distribution, which denote a missile's most likely termination point and containment ellipses in two dimensions. To compute the probability of hit, the algorithm approximates the integral of the distribution function over the series of input polygons. The distribution function will generally be highly dependent on a missile's run-in heading and dive angle. Using the probabilities computed by the algorithm, an operator can determine the best run-in heading and dive angle

* Genetic algorithms are heuristic search algorithms based on mechanisms of evolution and natural selection. For further information, see the article by Best and Sanders on genetic algorithms in this issue.

† Fuzzy logic is a mathematical approach for simulating human-like reasoning and control. Consult Ref. 2 for further information.

for a missile going to a specified aimpoint to minimize the probability of collateral damage and maximize the probability of hitting the desired target. The Laboratory is developing a general algorithm for computing the hit probability that uses more sophisticated parallelepipeds, instead of polygons, to model the target area.

The three decision aids developed for the functional prototype provide the operator with vital assistance in controlling and monitoring strike progress. Both the aimpoint coverage and strike effectiveness aids have been integrated with the prototype. An independent program implements the hit probability algorithm. Eventually the hit probability decision aid will be incorporated into the functional prototype and may continue to be developed as a stand-alone system for use with the current Tomahawk arsenal.

These decision aids set the foundation for additional refinements that will allow a deeper exploration of operator support concepts. For example, we expanded the degree of operator control over the flex capability to permit examination of the level of control appropriate across a range of operational conditions. The initial control method for the aimpoint coverage decision aid used positive control in which the functional prototype computed and recommended actions but did not execute without explicit operator approval. A manual control capability to allow the TSC to divert any missile to any alternate mission available to that missile has been added (see the middle window in Fig. 6). Under manual control, the system does not provide any recommendations; it merely issues missile commands according to the direction of the operator.

Progressively more automatic control modes will also be introduced. In passive control, the prototype will compute and recommend strike actions and then automatically carry out the highest-ranking action unless overruled by the operator. This level of control might be appropriate for recovery options to high-priority aimpoints or when the missile is close to the branch point between two missions.

The final level of control is automatic control. In this mode, the strike management prototype will continuously evaluate and execute strike actions to optimize some measure of effectiveness without any operator intervention. One such optimization might be to minimize the mean square error between tasked and expected probability of damage, weighted by aimpoint priority (i.e., for each aimpoint i), as follows:

$$\min \mathrm{E}\{w(i)\,[PDE(i) - PDT(i)]^2\}\,,$$

where

$PDE(i)$ is the probability of damage expected at aimpoint i,

$PDT(i)$ is the probability of damage tasked at aimpoint i,

$w(i)$ is a weighting factor based on the priority of aimpoint i, and

E is the expectation operator.

The same method of control would not have to govern all scenarios. A strike control doctrine could invoke various methods of control for different situations. For example, control by negation might apply to recovery from missile failure, whereas positive control would apply to all other situations. Reallocation situations, other than missile failure, might arise through assessments by the TSC. The operator might decide to flex all missiles away from an aimpoint on the basis of damage seen in missile imagery. The prototype would compute flex actions to redistribute missiles to remaining aimpoints. Under positive control, the TSC would then select the option to execute.

Both the Macintosh and NEXTSTEP strike management prototypes process a limited scenario incorporating relatively few TBIP missiles, launch platforms, and targets. The three decision aids were developed to support this limited scenario. Nonetheless, these algorithms, as noted earlier, may provide a basis for devising and implementing expanded capabilities that will account for the full range of operational functions and considerations (e.g., a modified aimpoint coverage algorithm will need to account for large strikes consisting of a mix of current generation missiles and advanced TBIP missiles).

The expansion of strike control capabilities in the prototype will enable our sponsors to address other challenges facing future Tomahawk employment. Components of the functional prototype are being integrated into a demonstration of Tomahawk satellite communications to be conducted at the Laboratory during 1995. The demonstration will show two-way communications between strike management and missiles represented by the simulation driver. An important objective of this demonstration is to define techniques for dynamically assigning missile communication schedules that are fully coordinated with the strike management's perceptions. Also, through Independent Research and Development funding, APL has developed Phase I of the Precision Integrated Strike Concept Evaluation Suite (PISCES) to connect various Tomahawk-related simulations resident at the Laboratory into a distributed simulation capability. Although currently internal to APL, PISCES adheres to Distributed Interactive Simulation standards and protocols to allow future examination of strike management in more complex operations. The Laboratory is also preparing the strike management prototype for a more operationally representative assessment of usability and operational utility. Prospective operators will employ the prototype to manage more realistic and complex strike scenarios to confirm useful

support concepts, suggest enhancements, and identify missing capabilities.

Some aspects of the functional prototype may also benefit current Tomahawk employment. The strike summary window at the bottom of Fig. 6 includes a timeline to compare planned missile time of flight with reported progress. The timeline begins at the planned time of launch and ends at the projected time the missile will reach the target. Such a display will help decision makers evaluate coordination of launches and arrivals. This display could be extended to provide a strike planning tool for developing launch sequence plans.

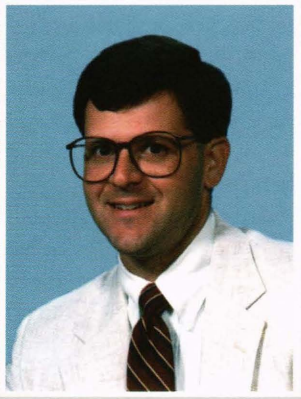
CONCLUSION

The prototype development effort has shown that strike management can be straightforward if operators are provided with the proper tools. Demonstrations to prospective operators have verified the potential benefits of the proposed new Tomahawk strike management functions and helped to identify and prioritize additional capabilities. Several features, such as the hit probability algorithm and the missile progress display, that could be of use in the current Tomahawk system have been identified during these demonstrations. The development of prototypes as part of the BG STC initiative has assisted the PEO(CU) not only in solidifying the strike management concept but also in adapting it for near-term initiatives such as the Tomahawk In-Flight Position Reporting System, a tracking system to be installed in some operational Tomahawk missiles.

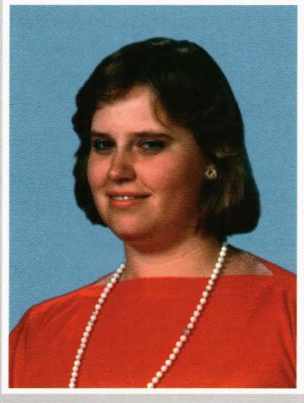
REFERENCES

[1] *Tomahawk Land Attack Missile (TLAM C/D) Employment Manual*, Naval Warfare Publication 3-03.1, Naval Doctrine Command, Norfolk, VA (1994).
[2] Quaranta, T. F., "Fuzzy Systems for Simulating Human-Like Reasoning and Control," *Johns Hopkins APL Tech. Dig.* **16**(1), 43–58 (1995).

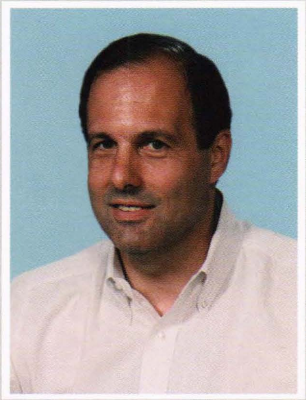
THE AUTHORS



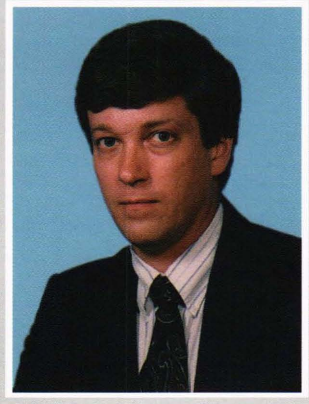
MARK D. LoPRESTO received a B.S. degree in systems engineering from the U.S. Naval Academy in 1982 and an M.S. in electrical engineering from The Johns Hopkins University in 1989. He joined APL in 1987 after 5 years as a Surface Warfare Officer in the Navy and is a Senior Staff engineer in the Surface and Strike Warfare Systems Engineering Group of the Fleet Systems Department. In 1991, Mr. LoPresto assumed his current position as technical lead for the Battle Group Strike Warfare Coordination initiative of the Navy's Tomahawk program. His recent efforts have involved developing functional prototypes, including Tomahawk Strike Management, for use in the Tomahawk Baseline Improvement Program. In addition, he has been supporting an Independent Research and Development project to provide a distributed simulation capability for evaluating advanced precision strike concepts. His e-mail address is Mark.LoPresto@jhuapl.edu.



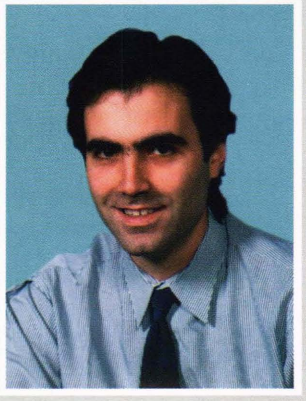
ANN F. POLLACK received a B.E.S. in electrical engineering and an M.S.E. in computer science from The Johns Hopkins University in 1988. She joined APL in 1988 and is a Senior Staff engineer in the Surface and Strike Warfare Systems Engineering Group. Ms. Pollack has worked primarily in the Tomahawk program, performing a variety of systems engineering tasks. Most recently she has served as the lead engineer for Tomahawk deconfliction (eliminating conflict between Tomahawk missiles in flight). She has continued her education, earning an M.S. in electrical engineering from The Johns Hopkins University in 1991, and is currently pursuing a Ph.D. in computer science from the University of Maryland, Baltimore County. Her e-mail address is Ann.Pollack@jhuapl.edu.



JOHN FLORENCE received a B.S. in mathematics from West Chester State University of Pennsylvania in 1966, an M.S. in operations research from George Washington University in 1972, and an M.S. in computer science from The Johns Hopkins University in 1987. Before coming to the Laboratory, Mr. Florence developed simulation models for the U.S. Army and numerous private businesses. He joined APL in 1985 and is a Senior Staff mathematician in the Surface and Strike Warfare Systems Engineering Group who has specialized in software engineering, simulation, modeling, and analysis. His current efforts include developing object-oriented simulations to support prototype development and distributed simulation capabilities at APL. His e-mail address is John.Florence@jhuapl.edu.



ROBERT C. FERGUSON received a B.S. in electrical engineering from Cornell University in 1984 and an M.S. in electrical engineering from Purdue University in 1985. He joined APL in 1986 as a member of the Strategic Systems Department, where he contributed to Trident II missile accuracy analysis. In 1989, he became a member of the Fleet Systems Department and contributed to the Aegis and Tomahawk programs by developing correlator/tracker algorithms for the Aegis Anti-Air Warfare Correlator/Tracker and performing system analysis for the Tomahawk Baseline Improvement Program. Currently, Mr. Ferguson is participating in Tomahawk deconfliction analysis and data fusion efforts for the E-2C program. His e-mail address is Robert.Ferguson@jhuapl.edu.



IAN E. FELDBERG received a B.S. in electrical engineering and computer science and an M.S. in computer science from The Johns Hopkins University in 1984 and 1987, respectively. Mr. Feldberg joined APL in 1984 and is a Senior Staff engineer in the Surface and Strike Warfare Systems Engineering Group. He has contributed to a variety of software projects, including computer animation for simulation, machine vision and machine learning research, and biomedical research. He became a member of the Fleet Systems Department in 1993, primarily in support of the Tomahawk program, and has served as the principal software engineer for the Tomahawk Terminal Fratricide Visualization. Mr. Feldberg is the principal software engineer for Tomahawk strike management prototyping efforts as well as technical consultant for the Synthetic Environment Workstation, an Independent Research and Development project coordinated through the Research Center. Mr. Feldberg is pursuing a doctorate in computer science with specialization in machine learning and synthetic environments. His e-mail address is Ian.Feldberg@jhuapl.edu.