The Integrated Vulnerability Management System

David P. Watson

Real-time estimation of submarine detectability is a complex problem for the modern, forward-deployed attack submarine because of the complexity of the littoral environment and the wide variety of potential detection sensors. This article describes a prototype system, the Integrated Vulnerability Management (IVM) System, for integrating information about the environment, threat systems, and the submarine itself. The IVM System rapidly computes detectability statistics and presents them to a system operator for use in various tactical operations. This capability represents a major advancement over current stealth management procedures and tools, which depend in many cases on manual parameter estimation and coordination. The operational concept for the prototype system is discussed, followed by a detailed description of the core technical elements. (Keywords: Real-time, Signature model, Submarine, Tactical decision aid, User interface.)

INTRODUCTION

The primary role of U.S. Navy attack submarine (SSN) class ships historically has been to hunt, track, and, if necessary, attack enemy submarines. Detectability of ownship by the enemy (counterdetection) is the paramount concern in these operations. The submarine's most effective attribute is stealth, and all tactical guidance relies on the maintenance of this condition. In this article, we refer to the threat of counterdetection as "vulnerability." (In the surface ship community, however, vulnerability refers to a physical damage condition. The term "susceptibility" is used to describe counterdetection in surface ships.) Obviously, the need for stealth is not unique to the submarine community, and the approach presented here could be applied to other platforms such as aircraft or surface ships as well. The submarine domain, however, presents a uniquely complex environment for vulnerability management.

In addition to its primary mission of antisubmarine warfare (ASW), the SSN may be involved in missions in near-shore (littoral) waters in support of surveillance, rescue, or strike operations. In fact, as the ASW threat environment evolves and requirements for a highly capable and covert littoral platform increase, such missions are becoming more frequent and important for the SSN.

This article describes a prototype system, the Integrated Vulnerability Management (IVM) System, developed in APL's Submarine Technology Department for the real-time assessment of counterdetection threats, the quantitative estimation of threat level, and an integrated graphical user interface (GUI) for display and manipulation of this information. The system was developed under sponsorship of the Defense Advanced Research Projects Agency (DARPA) and is currently being transitioned to the Navy for use in next-generation submarine combat systems.

Vulnerability Management

The problem of managing ownship vulnerability can be broken into three fundamental components: threat assessment, vulnerability estimation, and decision support. Threat assessment involves determining the critical attributes of all potential counterdetection threats (note that this might include neutral as well as hostile forces, depending on the mission). In addition to the threat's location, it would be helpful to know sensor types and performance parameters. In general, this information cannot be measured directly, and therefore must be inferred by the system. The second aspect of vulnerability management concerns the detailed estimation of ownship signatures. The final aspect is the utilization of assessment and estimation data to provide recommendations for planning future mission operations and for optimizing current ship configuration.

High-fidelity signature and detector modeling is the core component in the vulnerability management system. APL has a long history of model development in this area for threat assessment and technology development studies. These models are, however, not generally constrained by real-time or embedded system requirements. Therefore, initial efforts in the development of a vulnerability management tool focused on decision support and user interface aspects of the problem.[1]

In 1995, APL began an effort to prototype a tactical decision support tool that would address real-time signature modeling and ship integration problems as well as intelligent decision support and user interface development. The goals of this effort were to leverage emerging technologies in computer graphics, distributed real-time computing, and tactical automation to produce a tool that could be quickly transitioned into operational submarine combat systems to address this critical need. The following sections describe various aspects of this prototype, the IVM System.

OPERATIONAL CONCEPT

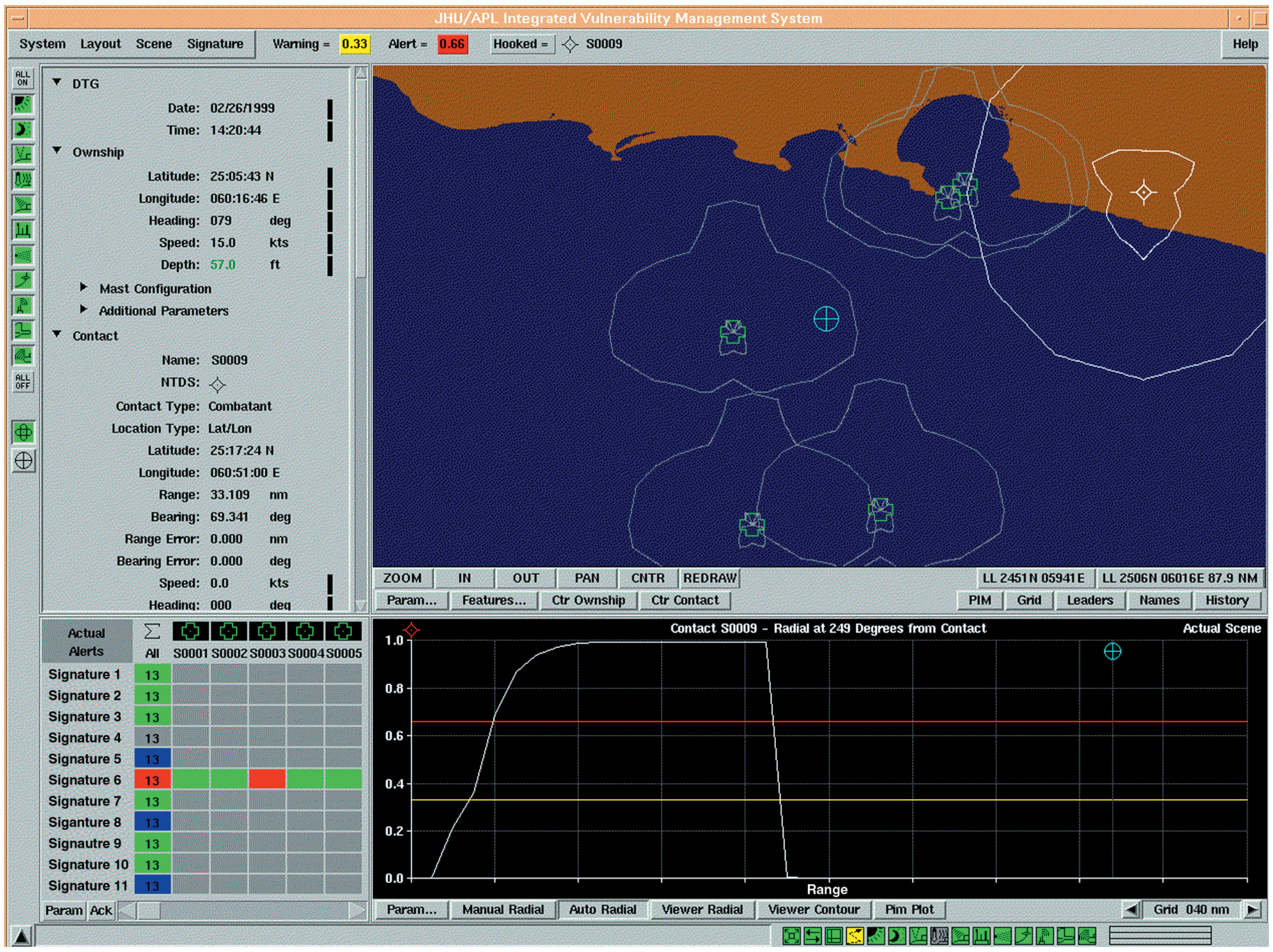
The general concept for IVM is to provide complete functionality within the combat system architecture without the requirement for a dedicated watchstander. Submarines, in particular, are limited in the number of crew (and the amount of space) available for combat control operations, and there is great sensitivity to imposing additional tasks. It was decided early in the IVM concept development stage to design the system for various levels of operator interaction, from unattended alerting to support for one or more operators dedicated to stealth planning.

Figure 1 shows one of the possible user interface configurations for the IVM System, highlighting a number of important features. The interface window is partitioned into frames that display different aspects of the overall vulnerability picture. In the example, the largest frame (top right) displays a geographic scene showing ownship, significant contacts in the tactical scene, and certain environmental features. Overlaid on this scene are contours indicating probability of detection regions for various contacts, allowing the operator to quickly view threat capabilities. A particular contact has been designated for further analysis by the operator, resulting in a highlighted symbol and detection performance contour. The designation also configures other panes of the top-level frame to focus on the contact. The plot pane (bottom right) shows a graph of probability of counterdetection as a function of range in the line of bearing between ownship and the designated contact.

The parameter pane (top left) presents the various measured and inferred values used by the real-time models to derive vulnerability information. These are available for review and for editing by the operator. The parameters consist of the three general categories already noted: ownship state, contact attributes, and environment. Examples of ownship state would be speed, depth, and mast exposure heights. Contact attributes include not only the standard kinematic estimates (position, course, and speed), but inferences the system has made regarding detection capability, i.e., the number and type of sensor systems. Environmental parameters include water and atmospheric parameters, either measured by onboard sensors or broadcast from centralized tactical meteorological or oceanographic sources. In real-time operation, the IVM System automatically updates these parameters through interfaces to the ship's combat system and common tactical databases.

The operator may choose to edit these parameters at any time, either to provide an improved estimate or to experiment with alternative configurations. In this case, once parameter modification has been confirmed, all relevant vulnerability models are recomputed in real time and the tactical picture is updated. This "what if" capability is a central feature in the IVM concept of operations, allowing the operator to plan for future situations.

The bottom left pane of Fig. 1 provides a high-level summary of the vulnerability situation in the form of a counterdetection alert matrix. It is formed by listing all contacts in the tactical scene and then computing counterdetection probability across their individual



Warning = 0.33 Alert = 0.66 Hooked = S0009

▼ DTG

Date: 02/26/1999
Time: 14:20:44

▼ Ownship

Latitude: 25:05:43 N
Longitude: 060:16:46 E
Heading: 079 deg
Speed: 15.0 kts
Depth: 57.0 ft

▶ Mast Configuration
▶ Additional Parameters

▼ Contact

Name: S0009
NTDS:
Contact Type: Combatant
Location Type: Lat/Lon
Latitude: 25:17:24 N
Longitude: 060:51:00 E
Range: 33.109 nm
Bearing: 69.341 deg
Range Error: 0.000 nm
Bearing Error: 0.000 deg
Speed: 0.0 kts
Heading: 000 deg

Actual Alerts	Σ All	S0001	S0002	S0003	S0004	S0005
Signature 1	🟩 13					
Signature 2	🟩 13					
Signature 3	🟩 13					
Signature 4	13					
Signature 5	🟦 13					
Signature 6	🟥 13	🟩	🟩	🟥	🟩	🟩
Signature 7	🟩 13					
Siganture 8	🟦 13					
Signautre 9	🟩 13					
Signature 10	🟩 13					
Signature 11	🟦 13					

Figure 1. Example of the Integrated Vulnerability Management System user interface. (Top left: parameter pane; bottom left: counterdetection alert matrix; top right: ownship, contacts, and environmental features; bottom right: graph of probability of detection.)

sensor configurations. The resulting thresholded color chart provides quick notification of current and anticipated counterdetection threats. The operator has the option to tailor the alerting display by creating summary alert columns based on the logical disjunction of alert conditions across various types of contacts (e.g., all air platforms classified as hostile).

This set of tools forms the basis for an integrated vulnerability analysis capability that might be used in various ways. During mission planning, the system can function as an elaborate signature calculator. The user might configure different hypothetical combinations of environment, ownship state, and expected threats, and then run the relevant signature models in an iterative process to optimize mission timing and ownship configuration. During mission execution, the system is continually updating the vulnerability situation, and might operate unattended until an alert event occurs. When that happens, an operator might be assigned to analyze ship signatures and perhaps recommend an alternative course of action to remove the alert condition based on model predictions. It should be emphasized that definitive operational doctrine for the IVM System has not been produced.

IVM SYSTEM DESCRIPTION

To explore both the computational and operational feasibility of IVM, APL developed the prototype system using commercial off-the-shelf (COTS) workstation technology. The IVM workstation was designed to be integrated directly into the combat system as shown in Fig. 2. Various generic combat system components are shown linked through a generic tactical information network. In future ship combat systems, this network will be implemented on COTS networking components with open interface standards for the sharing of tactical contact, ownship, and environmental information. The figure shows several subsystems that directly relate to the new vulnerability management capability. In most cases, these subsystems currently exist in the submarine combat system. In other cases, the IVM capability may require additional ship instrumentation,

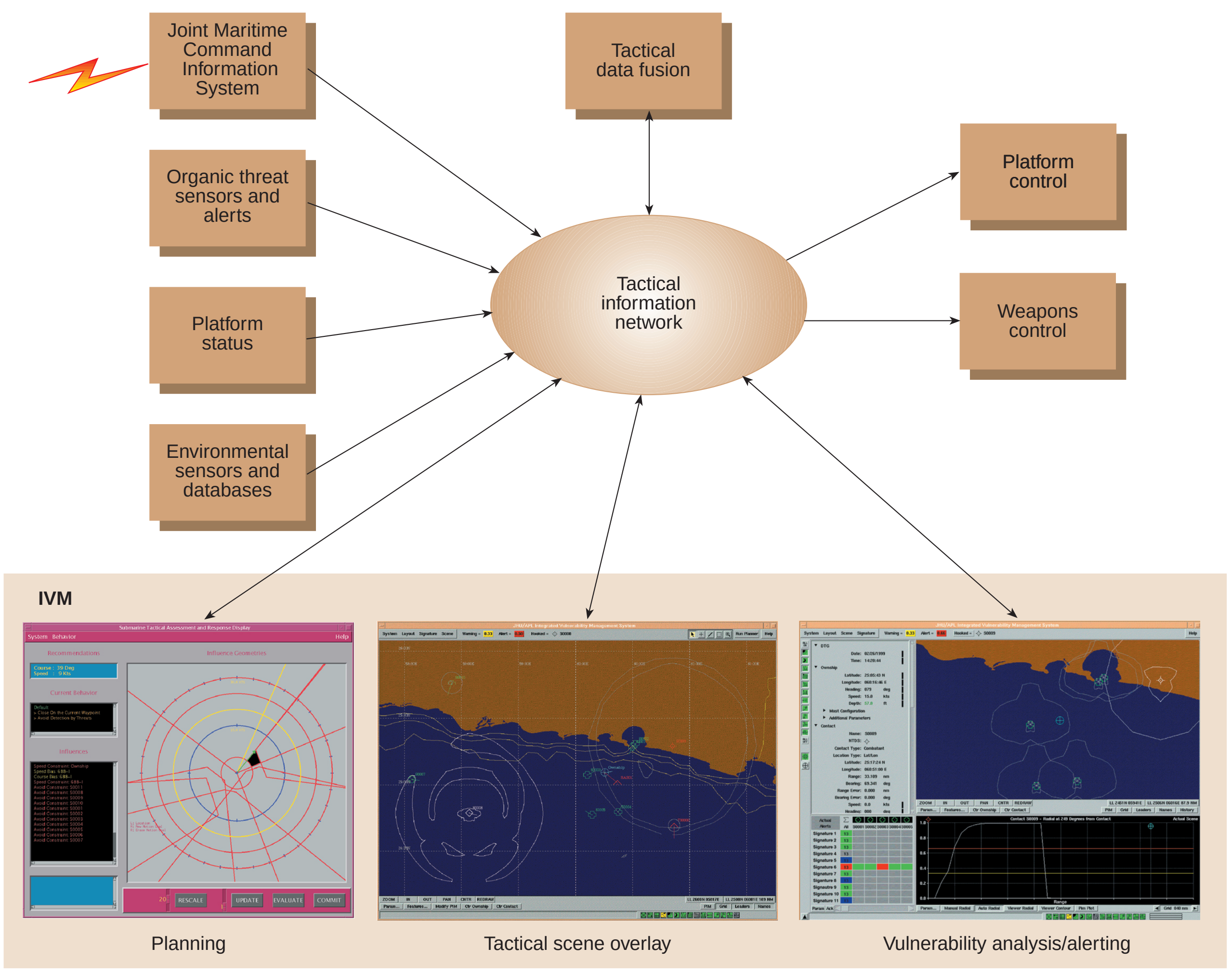

Joint Maritime Command Information System
Tactical data fusion
Organic threat sensors and alerts
Platform control
Tactical information network
Platform status
Weapons control
Environmental sensors and databases
IVM
Planning
Tactical scene overlay
Vulnerability analysis/alerting

Figure 2. Integrated Vulnerability Management System concept.

connectivity, and functionality. In particular, the IVM System is specifically designed to both support and benefit from the higher level of platform connectivity implied in current Navy “network-centric” ASW concepts.

Network Connectivity

As shown in Fig. 2, network connectivity plays a significant role in the acquisition of nonlocal and nonorganic tactical information, including air and surface contact tracks and environmental parameters. This information is used to augment the submarine’s primary onboard (“organic”) sensor systems, which are the primary sources for real-time, local-situation information. The link between onboard and offboard information is made through a data fusion process (external to IVM), which results in a single, unambiguous set of detected and tracked contacts to be processed by the IVM System.

Ownship and Environmental Data

Besides the parameters noted in the section on Operational Concept, ownship state also includes position, course, heading, and depth, which are currently available on the combat system network. In addition, the signature models of the IVM System in many cases can use even more information on ownship state, including details about the ship design or equipment lineup, where relevant. The system is currently configured for direct operator entry of these parameters, but automation of this process through additional ship instrumentation and networking would be beneficial in the future.

The second major information category, environmental parameters (e.g., ocean optical and acoustic characteristics), is important for IVM modeling. Historically, ocean acoustic parameters have been the focus during submarine operations. However, with the introduction of nonacoustic modeling in IVM, nonacoustic ocean and atmospheric parameters have become equally important. In many cases, these parameters are highly variable in both space and time, and yet have dramatic effects on submarine detectability. Without an organic measurement capability for these parameters, the IVM System must rely on historical environmental databases. If IVM capabilities are to be fully exploited, enhancement of the submarine's organic environmental sensing capability and integration with the emerging Navy tactical meteorological and oceanographic information systems will be required.

System Architecture

The primary components of IVM functionality are shown in Fig. 2 as planning, tactical scene overlay, and vulnerability analysis/alerting. The architecture for implementing this functionality has been developed for modularity in both hardware and software to support functional and throughput performance goals. It is also assumed that the IVM System might prove valuable to a broader class of platforms, including stealth surface ships and aircraft. A modular design approach will facilitate system reconfiguration with different models, external interfaces, and displays. This section discusses the hardware and software architecture features that support these goals.

Software Architecture

Signature computation. The fundamental process in IVM is signature generation. A signature is a point or an array of data representing the probability of ownship detection for a sensor at one or more points in space. Associated with a signature is the set of parameters—ownship state, sensed (or reported) contacts, and environmental—used as inputs to the signature modeling process, known as the "state vector." Parameter values may be measured directly, inferred, or retrieved from historical databases. Once a complete state vector has been formed, it is passed to one or more model processes for use in signature computation. This process is driven periodically, based on real-time updates from the combat system, and aperiodically, based on requests submitted by the operator. Once a signature has been computed by the system, it is made available for display, alert thresholding, or use in mission planning. The formation of state vectors, computation of signatures, and management of computed signatures are central components of the IVM System.

The software architecture for IVM is driven by the fact that signature model processing is computationally intensive, yet can be performed in parallel. IVM configurations typically contain from 3 to 10 independent modeling processes. In other words, the submarine is simultaneously informed about its detectability by optical, radar, infrared, and other means. The signature modeling processing in each of these areas can be performed in parallel and then integrated to form an overall estimate of vulnerability. Figure 3 shows an overview of the multiprocess IVM software architecture that exploits this parallelism. All interprocess communication is performed using a simple TCP/IP (transmission control protocol/internet protocol) socket library.

Kernel process. Central to the software architecture is a kernel process that integrates and coordinates all run-time components of the system including input/output, GUIs, state vector formation, model management, and alert generation. The kernel contains a complete representation of all tactical scene objects and manages updates to and from other system components. An initial attempt to build an asynchronous, multithreaded access layer to the kernel's tactical scene objects for external processes was abandoned during development as unnecessarily complex. The current design uses a single thread of control in the kernel, and all object sharing is through explicit message passing across the links shown in Fig. 4. Thus, the kernel can be thought of as a single loop that sequences among the

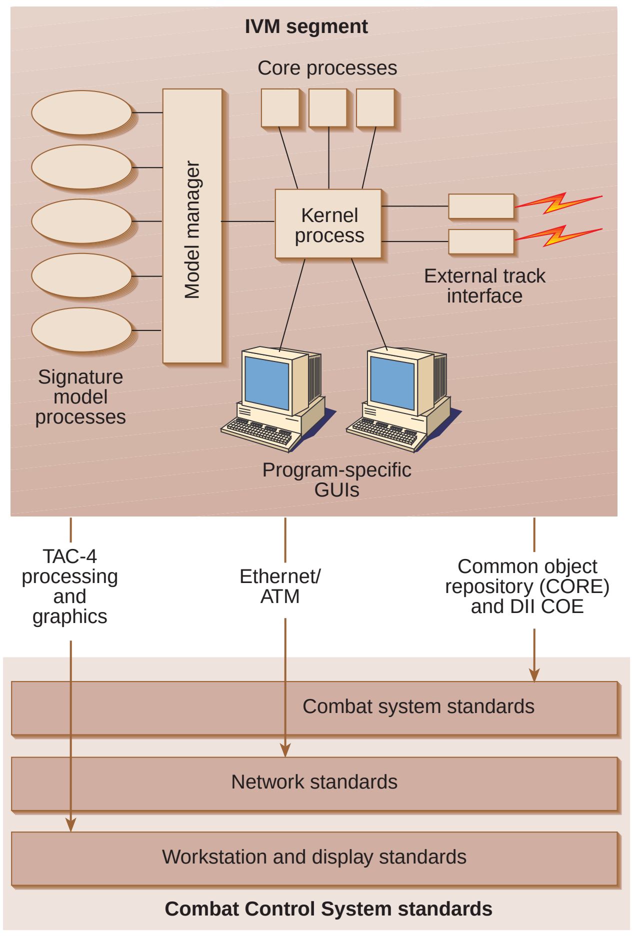

IVM segment
Core processes
Model manager
Kernel process
External track interface
Signature model processes
Program-specific GUIs
TAC-4 processing and graphics
Ethernet/ ATM
Common object repository (CORE) and DII COE
Combat system standards
Network standards
Workstation and display standards
Combat Control System standards

Figure 3. Software architecture summary. (DII COE = defense information infrastructure common operating environment, ATM = asynchronous transfer mode, GUI = graphical user interface.)

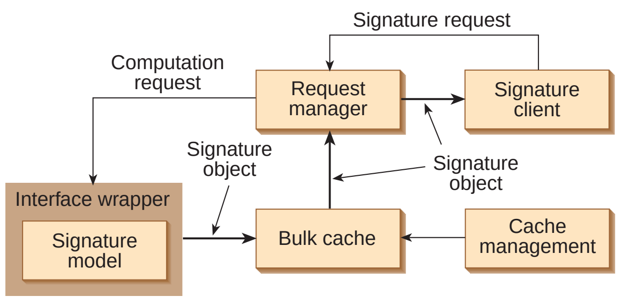


Figure 4. Model management processing.

external interface, GUIs, the model manager, and other system processes.

External track interface. The external track interface (ETI) process shown in Fig. 3 provides real-time synchronization for the IVM System. During prototyping efforts, a variety of external systems were interfaced to IVM for concept demonstration using a combination of simple socket and high-level CORBA (common object request broker architecture) protocols. In addition to real-time synchronization, the function of the ETI is to convert external ownship, contact, and environmental data formats into a common format for kernel processing. This translation process provides a critical portability capability to the IVM prototype. In order to integrate the system with new tactical or simulation systems, software modifications are isolated to the ETI component rather than distributed across all components. The ETI developed for the prototype system conforms to a layered set of DoD and Navy standards as shown in Fig. 3. At the lowest level, processing is performed on a standard tactical computing platform, TAC-4. All network interfacing is through TCP/IP, implemented on Ethernet or asynchronous transfer mode (ATM) networks, which are the standard for the Navy's new Virginia-class SSN.

DII common operating environment. The defense information infrastructure (DII) common operating environment (COE) comprises software, display, and networking tools and standards applicable to next-generation tactical software development. The IVM prototype is built within the DII/COE and employs the standard geographic display product, the Joint Mapping Toolkit, for top-level geographic scene display. Further interface standardization is provided by a Navy standard for object-oriented tactical information distribution based on the CORBA protocol. This standard specifies standardized object definitions, using the CORBA interface design language for ownship state, contact parameters, sensor parameters, and system events.

Model processing and management. Once data have been input to the IVM System and used for state vector formation, the system must coordinate signature generation through the set of model processes. Conceptually, the process is simple: the state vector is passed to each of the signature model processes, which in turn perform the required computation and return a signature data structure. However, there are a number of issues that must be addressed in the implementation. First, model processing is time-consuming and must not be allowed to interrupt the real-time responsiveness of the system. Processing-time requirements for the current IVM model range from less than 1 s to approximately 20 s per signature. With a worst-case situation requiring 10 different types of signatures to be computed for a new tactical state, it is unacceptable for the system to wait for signature computation to be completed. The problem is solved by implementing model processing as a set of asynchronous parallel processes. The IVM model manager coordinates this processing and provides important filtering and caching functions.

The first step in the management of signature model computation is determining whether changes in state vector values warrant a new model run by the system. Many state vector values, such as ownship state, are updated as frequently as 1 Hz by the combat system, yet small changes in magnitude may have little or no impact on the resultant signature estimation. Thus, some sort of prefilter or threshold must be applied to state vector changes to prevent unnecessary computation. In the IVM System, this process, known as significant event detection, uses a predefined library of thresholds on individual state vector values to trigger new model runs. This library is based on sensitivity analyses performed during system design.

The second technique used in the model management process is signature caching. Once a model run has been completed and the signature data structure results have been made available for display or further processing, the system stores the result and the originating state vector in dynamic memory for fast subsequent retrieval. Figure 4 shows an overview of this process. When requests for new signature computation are generated, either by real-time system updates or asynchronous user requests, the state vector is sent to a request manager. This process performs significant event detection, as described previously, by comparing the new state vector with a set of recently processed state vectors. If a match is found, the resulting signature data structure is passed directly back to the requester.

All of this model management occurs within the main kernel process for efficiency. If no match is found for the new state vector, then an asynchronous model processing request must be made. As shown in Fig. 4, a request queue is maintained for each model to coordinate multiple, sequential requests that might occur at intervals shorter than the time required for model computation. A TCP/IP socket protocol is used for request message passing to the model process itself.

This messaging infrastructure, as well as state vector format conversion to the native model representation, is shown in Fig. 4 as a "wrapper"around the core model processing. This layer is used because IVM models come from a variety of sources outside the IVM development program. In many cases, these are legacy codes with their own variable names and data formats. The IVM model wrapper converts the standard request message format into native model formats and creates a standard signature output data structure for transmission back to the kernel process. Once results are available in the kernel, the request queue is updated, and the signature and associated state vector are recorded in the signature cache and made available to the system for alert generation, display, and further processing.

A cache management function must be used to periodically delete data structures that have not been used for some time to keep the cache from exceeding available memory. This function has not yet been implemented, but must be completed in order for the system to operate for any significant duration.

The main benefit of this somewhat elaborate scheme for data management is to provide very high responsiveness in "what if" queries by the user. Whereas initial computations may take several seconds, once the system has cached the results, feedback is nearly instantaneous. This facilitates the use of these computationally expensive models in a variety of analysis and planning tasks.

Graphical user interface. The final processing component shown in Fig. 3 is the IVM GUI. In fact, the IVM System is currently designed for multiple, semi-independent GUI connections. Many of the particular features of the IVM GUI, including geographic tactical scene display, alert matrix display, and signature/parameter display, were discussed in the Operational Concept section and shown in Fig. 1.

All display processing is performed in separate, asynchronous processes to optimize system responsiveness to user controls. If multiple GUI processes are used, both GUIs share the kernel state. This means that as real-time updates occur, both GUIs are updated concurrently. Also, if a user at one workstation performs a parameter override operation, that override is shared at all other IVM workstations. The loose coupling between display processes and core input/output and model processing provides great flexibility in the use of IVM in an overall combat system architecture. In addition to the GUI developed in the prototype, system interaction might occur through a nongraphic (command line) interface or be abstracted into a "vulnerability server" capability to provide signature and alerting information to other systems through the existing GUI process protocols.

An overall guiding principle in IVM development is cross-platform implementation using proven open computing standards. All IVM software is developed in ANSI C++, using Hewlett-Packard (HP) and Sun Microsystems workstations. GUI processing uses X11/R6 with Motif development libraries. Three-dimensional graphics panes within the overall GUI framework are implemented using standard OpenGL development libraries. As previously discussed, the DII/COE provides an overall run-time and development layer above the operating system. Although this environment is available for the Sun Solaris operating system as well as HP, IVM GUI development has been limited to HP at this time.

Hardware Architecture

Initially, IVM prototype development was not constrained in hardware implementation. The distributed architecture described above was developed in a laboratory environment with several dedicated workstations, including high-performance Silicon Graphics, Inc., Onyx/Reality hardware for three-dimensional signature visualization and GUI prototyping. As the program focus shifted from concept development to technology transition, the constraints of shipboard implementation became a major design driver. Submarines, in particular, are severely constrained in available space for additional processing hardware and displays. The decision to transition the proprietary Silicon Graphics GUI implementation to the DII/COE standard was motivated, at least in part, by the possibility of reconfiguring existing workstations in the submarine control center for IVM user interface as required. Even more significant was the requirement for multiple processors for signature model computation. In the laboratory, each of these processors was a complete workstation, with dedicated keyboard, monitor, and peripheral devices. To meet transition hardware constraints while minimizing software modification, a dedicated signature model multicomputer system was developed based on COTS components as shown in Fig. 5.

The signature model processor design is based on Force Computers, Inc., single-board computer systems. These are identical to a full workstation configuration, with central processing unit, processor memory, disk interface, and network interface hardware. There was no reason to change the processor interconnect topology or technology from that currently operational in the workstation network prototype. Therefore, system configuration simply required physical and power interfaces, and these were provided by a standard 19-in. VME chassis. The resulting implementation requires 7 slots of the chassis (out of 12 available). The signature model processing load is distributed across this array and linked with a single host processor system that runs GUI, kernel, and interface processes.

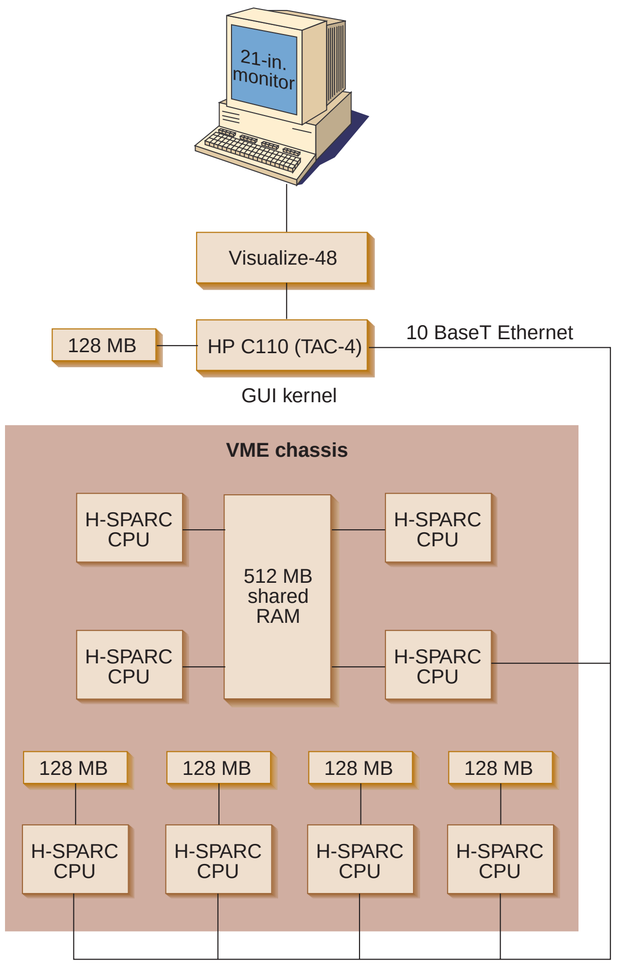


Figure 5. Hardware architecture summary (CPU = central processing unit).

All processor interconnects are implemented with 10 BaseT Ethernet (10 Mbps).

Real-time processing requirements for the IVM System are limited to issues of user interface responsiveness. Therefore, a specialized operating system was not required in the signature model processor. Initial signature model development was performed on Sun workstations using the Solaris operating system, and this was maintained in the final prototype. The overall goal in the model processor hardware specification was to minimize the cost and schedule risk associated with platform transition by making the switch to the new environment "software transparent." This was accomplished, and transition to the new hardware was performed in less than a week.

It should be noted that the hardware architecture is essentially scalable. If new signature models are desired in the future, it is simple to add processing cards to host the new processes. All infrastructures for kernel communication and coordination are in place.

The signature model components of IVM are independent from each other and from the core processing system. The prototype development program did not develop any new model codes, emphasizing instead the essential infrastructure for management and coordination of a diverse set of legacy components. The current IVM System includes a set of demonstration models at different levels of maturity and validation status, but in general, all share a similar overall structure (see the boxed insert).

SUMMARY

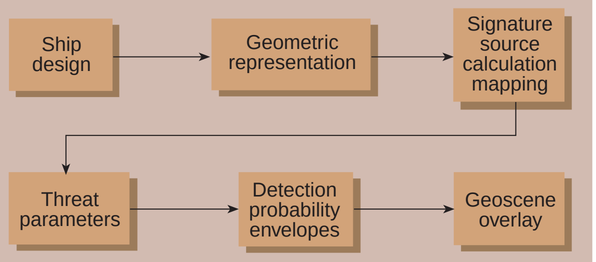
The IVM prototype described here has reached a significant developmental milestone and is ready for evaluation in a structured sea test environment. However, a number of areas require further development. Perhaps the most important of these involves the definition of vulnerability itself. The IVM System currently equates vulnerability with probability of counterdetection. While this metric is convenient for computation and integration across a number of different domains, it does not fully capture the tactical notion of vulnerability. Ideally, such a metric would

IVM SIGNATURE MODEL PROCESSING

The figure shows a generic signature modeling process. The process starts with a physical model of the source of the signature. This might be an active source such as ownship acoustic noise or a passive source such as parts of the ship's structure. In either case, estimation of source parameters is the first step in the formation of the model state vector. The IVM radar model uses a library of ship models that capture all relevant physical features. Once the specific ship configuration has been specified, including operating conditions of the platform, the physical models are used to create a signature map as a function of ship orientation as appropriate.

The next step is to determine a probability of sensor detection. This will be a function of various sensor system parameters, which may be obtained by database lookup based on a hypothesized threat sensor type or measured directly by onboard threat databases. The IVM System typically computes this probability of detection across the full azimuth and out to a fixed maximum range, thus producing an array of probabilities for subsequent processing and display.

All the model processes in IVM follow the same basic structure, starting with analyzing source phenomenology, computing environmental effects, and then modeling the detection process.



Generic IVM signature model process.

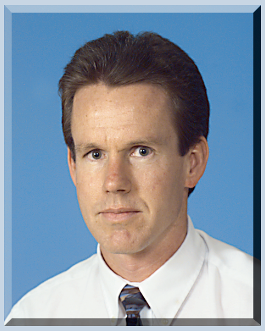
capture notions of classification uncertainty as well as assumptions about the threat's offensive intent and capability. Clearly, more work is needed in this area to link quantitative model results to heuristic rules for tactical assessment and mission objectives.

The IVM concept development effort began at APL with goals of demonstrating a new class of tactical decision support tool for stealth management. This goal has been accomplished with endorsement from both DARPA and the Navy. High-fidelity, real-time signature modeling has been demonstrated as a viable tool for situation awareness, alerting, and planning. An operating concept incorporating advanced graphics capabilities and real-time combat system interface has been developed. Finally, the overall concept of dynamic vulnerability management has been introduced as a potential complement to stealth platform design, with the potential to serve as a critical component in future cost-driven development efforts. As littoral submarine operations become complicated with the proliferation of sophisticated, low-cost sensor technology, real-time vulnerability management will become a primary component of submarine planning and operation.

REFERENCE

[1]Wenstrand, D. C., Dantzler, H. L., Jr., Hall, M. R., Scheerer, D. J., Sinex, C. H., et al., *A Multiple Knowledge Base Approach to Submarine Stealth Monitoring and Planning*, Technical Paper presented at DARPA's Associate Technology Symposium, George Mason University (7 Jun 1991).

THE AUTHOR



DAVID P. WATSON received a B.S. degree in mathematics from Hampden-Sydney College in 1980 and a B.A. in applied mathematics from California State University, Fullerton, in 1987. Before joining APL in 1995, he worked at Lockheed-Martin Laboratories in the application of high-performance computing systems to tactical automation and decision support. Mr. Watson is currently a member of APL's Senior Technical Staff and is Assistant Supervisor of the Information Technologies Group of the Submarine Technology Department. His e-mail address is david.watson@jhuapl.edu.