Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

SIMULATION TO SUPPORT OPERATIONAL TESTING: A PRACTICAL APPLICATION

Bradford S. Canova
Peter H. Christensen
Michael D. Lee
Bruce R. Tripp
Michael H. Pack
David L. Pack

The MITRE Corporation
234 South Fraley Boulevard
Dumfries, VA 22026, U.S.A.

ABSTRACT

This paper describes a combined effort between The Marine Corps Systems Command (MARCORSYSCOM), the Marine Corps Operational Test and Evaluation Activity (MCOTEA) and the MITRE Corporation to exploit M&S to support Operational Test (OT) of the Predator Short Range Assault Weapon (SRAW). When applied appropriately, the cost benefits of using simulation assisted testing can be substantial. The March 23, 1998 edition of *Aviation Week and Space Technology* contains an article entitled "*Better Modeling Will Alter the Culture of Flight Testing*." The first sentence reads:

> "Over the next decade, budget pressures and a growing dependence on modeling and simulation will alter the philosophy and methods of flight testing military aircraft and weapons."

The Predator simulation system provides an example of how simulation can be applied to the system development process to help reduce cost and ensure a higher quality product. The success of the Predator simulation system shows that with a complete, coherent VV&A process in place, a simulation system can be developed to provide valuable input to the design, development, testing and training phases of the system development process.

1 PREDATOR SRAW OVERVIEW

The Predator SRAW was conceived in 1987 as a low-cost, short-range anti-armor missile with a top down attack warhead. The program began concept exploration in 1989 with selection of five contractors each tasked to refine missile design. In 1990, Demonstration/Validation (DEM/VAL) began with contract award to Loral Aeroneutronic. Risk reduction was initiated in 1992, to refine the missile tactical design. DEM/VAL completed in 1993 with multiple test firings. Engineering and Manufacturing Development (EMD) began in 1994. EMD will produce 125 missile systems to support Technical Evaluation (TECHEVAL) and 103 missiles to support Operational Test (OT).

The Predator SRAW missile weighs approximately 20 pounds and is about 35 inches long. As seen in Figure 1, the airframe holds three modular components: a Target Detection Device (TDD), warhead and Flight Module.

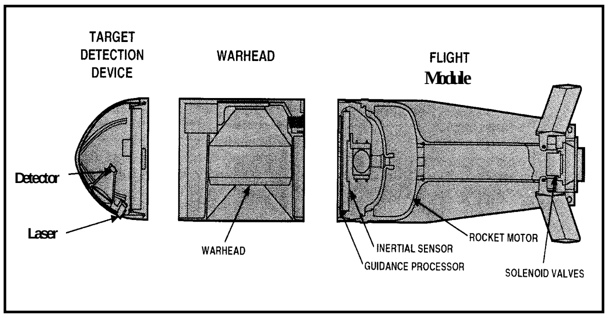

Figure 1: Predator SRAW Missile Components

The TDD is a dual mode device that combines a laser ranger and magnetic detector. The laser ranger is depressed at a 60 degree angle and measures the missile's height above the terrain. The laser ranger locates a target by identifying changes in range associated with leading and trailing edges of a target. The magnetic detector (magnetometer) senses changes in the magnetic field around the missile as it approaches and flies over a target. The TDD commands detonation of the warhead when both laser ranger and magnetic detector have confirmed target overflight. The Explosively Formed Penetrating (EFP) warhead is contained within the missile's airframe support structure. The support structure also houses a Fire-Through Cover, Surveillance/Test Connector, Manifold, Safe and Arm Assembly and Flex Cable. The EFP explodes downward through the Fire-Through Cover and into the target.

The Flight Module consists of an Inertial Measuring Unit (IMU), Guidance and Control Unit (GCU), Jet Reaction and Control Assembly (JRCA) and a rocket motor. The IMU provides rate sensing and acceleration data to the GCU. The IMU Rate Sensors translate tracking motion of the operator into a trajectory bias for the missile GCU. The trajectory bias compensates for target movement when the missile is fired. At launch, the GCU commands the missile to climb and corrects for environmental factors in flight. The GCU provides flight path correction commands to the JRCA. The JRCA provides yaw, pitch and roll control for the missile. The JRCA opens and closes eight solenoid valves that port gas laterally about the airframe. The missile intercept solution is designed to overfly the target. Predator SRAW employs a dual phase solid propellant rocket motor that permits employment in an enclosed area, such as a building or bunker. The rocket motor's first phase generates minimal back blast; the rocket motor's flight phase propels the missile to the target.

2 PREDATOR M&S SYSTEM

The Predator SRAW Modeling and Simulation (M&S) System is a duel mode system that executes in both hardware-in-the-loop (HWIL) and software-in-the-loop (SWIL) modes. It supports the evaluation of production representative missile hardware and software in a simulated environment that replicates the missile's operational environment. The simulation system runs on a Silicon Graphics ONYX Reality Engine 2 (SGI ONYX) Workstation. The simulation environment was built using the Force Level Analysis and Mission Effectiveness System (FLAMES). The missile's flight dynamics are simulated by a physics based Six Degree of Freedom (6DOF) model currently used to support Developmental Testing.

3 SYSTEM DESIGN

There are four major software components of the Predator SRAW M&S system. These include the simulation environment, the 6DOF missile flyout model, the GCU, and the TDD. Figure 2 provides a graphical depiction of how the major components interact. The following sections describe the architecture associated with each component, including their states, modes, and software.

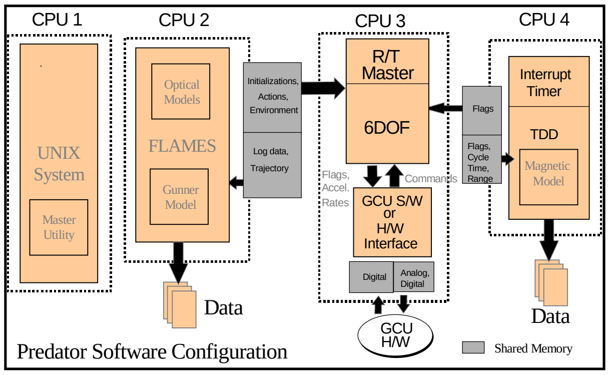

Figure 2: Predator M&S System Configuration

3.1 FLAMES Simulation Environment

FLAMES is a commercially available simulation environment that provides the infrastructure to support the representation of the natural environment, target representations, gunner behavior representations, and data collection functions. The FLAMES component consists of the software required to initialize and control the Predator SRAW real-time simulation, to interface to the simulation users and to record the results of simulation runs for later display, review and analysis.

3.1.1 FLAMES Architecture

FLAMES is an object-oriented simulation environment that provides the mechanisms to represent vehicles, weapons systems, sensor systems, munitions, and terrain. Typically scripts completely control behavior of all objects throughout the duration of the scenario. However, it is possible to build vehicle or weapon system object models that exercise varying degrees of autonomous control.

For the Predator missile, FLAMES does not compute missile behavior internally but rather reads the current missile state vector (position, velocity, mode, status, etc.) from the other components through shared memory interfaces. FLAMES controls the initial state of the Predator SRAW model by reading the parameter tables and the initial state vector from configuration files. FLAMES then relays that information to the other components via the same shared memory interfaces. FLAMES controls the position, movement, and actions of several other models, such as the target models and the gunner model used to sense targets and initiate missile firing.

In addition, FLAMES records the flight data (time, position, attitude, mode, & status) during the engagements for later playback. In playback mode, FLAMES displays the engagement, showing the terrain, the target, the gunner, the Predator SRAW missile and also displays the location of the impact of the EFP warhead.

3.1.2 FLAMES States and Modes

During a simulation run FLAMES executes in three modes: Initialization, Track, and Flight (see Figure 3). In the Initialization mode FLAMES reads the scenarios, the target and gunner models, the parameter files that establish the initial position of the target and the gunner and all of the data files required to initialize the other components from disk storage.

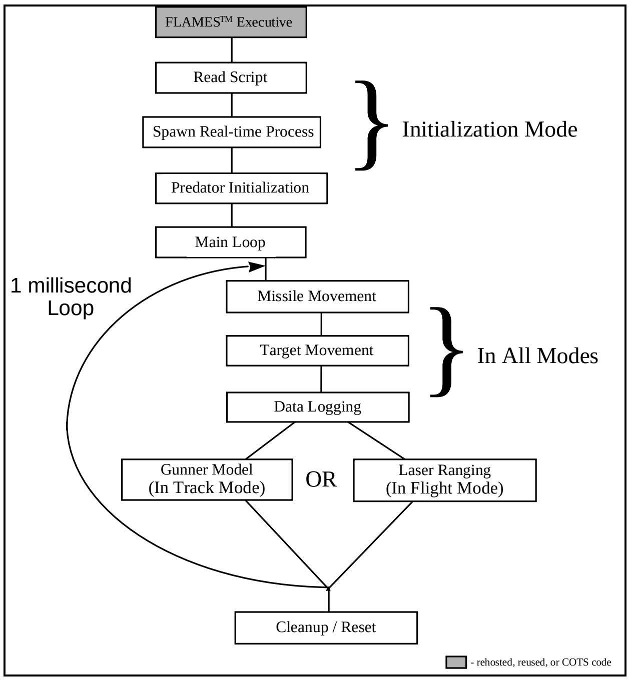

Figure 3: FLAMES Component Modes & Control Flow

The FLAMES component is capable of performing single runs or of performing multiple runs that cover the entire range of environments in which the Predator SRAW is to be tested.

In addition to initializing its own data structures, FLAMES allocates and initializes all of the shared memory structures required for inter-process communication and starts the Master Real-time process and the TDD process on separate CPUs. The CPU layout is shown in Figure 4.

Each process is given its own CPU to ensure real-time performance. CPU 0 contains all the system processing and the operating system housekeeping. Because the Master Graphical User Interface (GUI) Utility is not a Real Time process, it is also put on CPU 0 along with any other non-Real Time utilities.

In order to collect telemetry data from the simulation, a telemetry shared memory segment is created and the 6DOF and TDD populate the shared memory segment during the simulation. The data is stored in a large array until the end of a run when this data is written to an output file.

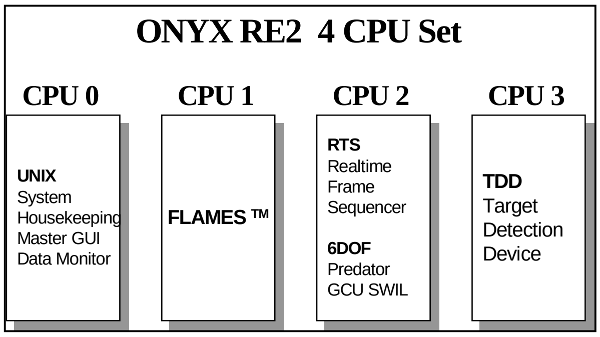

Figure 4: Process Locations

After completing initialization, FLAMES enters the Track mode. In Track mode FLAMES invokes the subroutine that reads the current missile state from the 6DOF component and records that data via data logging. In Track mode FLAMES also invokes the gunner model. The gunner model scans the simulated environment for targets. Once a target is acquired, the gunner model aims the simulated Predator SRAW. The gunner model continues to point the missile at the target for a predefined tracking interval and then initiates firing. After the missile is fired FLAMES enters the Flight mode. In Flight mode the missile movement, the target movement and the data logging continue but the gunner model is terminated. In its place FLAMES invokes a function that computes the distance from the missile to the terrain along the line of sight of the laser ranging sensor housed in the Predator SRAW nose-cone. FLAMES remains in this mode, continuing to read missile position, compute target position and laser ranging until the 6DOF component signals that the run has ended due to either warhead detonation, fuel exhaustion, or time or distance overrun. At the end of the Flight mode FLAMES captures the data from that run and then re-enters the Initialization mode for the next run.

3.1.3 FLAMES Component

The FLAMES component is not executed under the real-time simulation executative for two reasons:

1) The FLAMES component must perform limited I/O functions, which would cause frame overruns under the real-time executive.
2) A non real-time processes is needed to initiate and manage the execution of the real-time executive during the execution of the simulation.

However, it is essential that the main FLAMES programming loop complete a cycle in under 1 millisecond in both Track and Flight modes. This is required because the target, gunner, and laser ranging models within the FLAMES component are updating critical data items (i.e.

target position, gunner tracking technique and accuracy, and the laser range) used by the 6DOF, GCU and TDD components. The other components are executing under strict real-time constraints, which requrires that the data being shared with FLAMES must be updated in a timely manner. If it is not, a lag will be introduced in the simulation that would degrade its overall fidelity.

The only way to resolve this discrepancy is to ensure that the FLAMES component completes execution well ahead of the required interval. To achieve this, most normal housekeeping subroutines were removed from the FLAMES "calendar". The FLAMES calendar is an internal list (discrete event list) that determines what events are to be processed in the main loop. Synchronization between FLAMES and the real-time executive is accomplished through the shared memory interface. Any inability of FLAMES to maintain the real-time frame rate of the real-time executive will cause the current simulation run to be terminated.

3.2 Six Degree of Freedom Model

The Six Degree of Freedom (6DOF) component consists of the software required to model the Predator SRAW airframe, aerodynamics, external environment, and all of the missile internal electrical and mechanical components excluding the GCU and the TDD. The 6DOF also uses an input parameter file which initializes 6DOF parameters so the results of each run vary similarly to what would be expected in the real world.

3.2.1 6DOF Architecture

The main module of the 6DOF component is the Master Real-time Sequencer (MRS). The MRS uses the React Pro real-time extensions to the SGI operating system to call all of the other functions in this component. React Pro is a commercial software package written by Silicon Graphics. It adds real-time capability to the standard SGI operating system (IRIX) by providing high-precision time management functions, microsecond-resolution process dispatching, and the capability to override normal IRIX processor and interrupt control. The MRS uses React Pro to ensure that the 6DOF software executes under strict real-time control. In the Predator SRAW HWIL Simulation mode, the 6DOF code is executed 1000 times per second. When in SWIL mode both the GCU code and the 6DOF code must execute 1000 times a second. As a consequence, the software must complete its computations in under 1 millisecond. If the code requires less than 1 millisecond the MRS executes a wait until the next millisecond tick occurs. If the software does not complete in 1 millisecond then a fatal error occurs and the simulation is aborted.

The core functionality of the 6DOF component is contained in the code that provides the six degree of freedom flight characteristics for the simulated Predator missile. The 6DOF contains the models of all the internal functions and components of the Predator SRAW Missile. The 6DOF was originally hosted on a Pentium-based PC written in Ada. Relatively few code changes were required to re-host the 6DOF on the SGI host. Some additional changes to the 6DOF were required to allow it to run in strict real-time. All communication in and out of the 6DOF component is via shared memory.

3.2.2 6DOF States and Modes

The 6DOF component has two operational modes: Initialization and Real-time. In the Initialization mode the 6DOF component reads the initialization parameters supplied by the FLAMES component from shared memory.

The Master Sequencer then calls the 6DOF internal initialization routines, locks the 6DOF component process into memory and enters the real-time mode. Once in real-time, the MRS reads the current state from shared memory and calls the 6DOF model. When the 6DOF completes one pass through its code it returns control to the Master Sequencer. The real-time operating system keeps control until the millisecond tick is issued at which time the next simulation cycle is executed (See Figure 5).

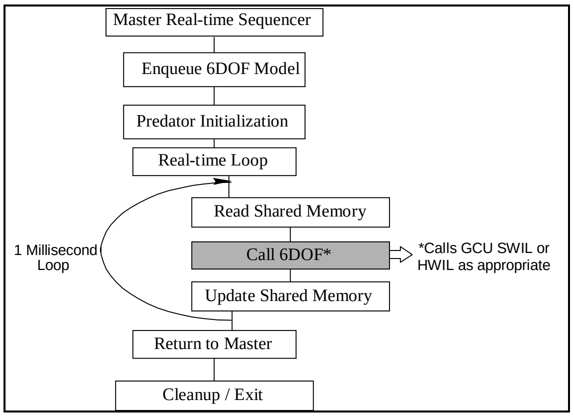

Figure 5: 6DOF Component Modes & Control Flow

The real-time loop continues until the 6DOF model sets a flag indicating that the firing scenario has completed. After completion of the scenario, the 6DOF component exits.

3.2.3 6DOF Component

The 6DOF component is comprised of Master Real-time Sequencer (MRS) and a 6DOF model that models the flight dynamics and characteristics of the Predator missile. The MRS is responsible for controlling the constrained, real-time execution of the simulation system. The steps required to establish a set of real-time processors with deterministic response to interrupts are outlined below:

1. Direct interrupts not related to real-time processes away from real-time processors. Direct real-time interrupts to the real-time processors.
2. Restrict each real-time processor so that all processes not explicitly assigned to it will be run on a non-real-time processor.
3. Lock real-time processes onto real-time processors.
4. Allocate and lock physical memory to all virtual addresses used by real-time processes.
5. Isolate the real-time processors from interprocessor interrupts used in a multiprocessor system.
6. Exempt real-time processors from system clock interrupts and UNIX timesharing scheduler activity.

Once these steps are performed the system is ready to begin scheduling and executing events. The frame scheduler is a kernel module that cyclically schedules processes at intervals defined by a regularly occurring interrupt. When enabled on a processor, the frame scheduler replaces all other IRIX scheduling policies on that processor, and configures the processor for real-time operation. Frame schedulers can be enabled on all but one processor in a multiprocessor system; i.e. all but the system processor. Each frame scheduler manages execution of processes only on its own processor, but multiple frame schedulers can be synchronized to enable frames on separate processors in a system to be synchronized.

The 6DOF model consists of an initialization phase and a state integration phase. The initialization is accomplished by calling the modules that have states requiring initialization. The state integration routine employs a Runge Kutta integrator with a selectable number of cycles. The model is commonly run using a 2-cycle integration. During the execution of each cycle, calls are made to the main modules of the 6DOF model. At the end of each time step (1 millisecond), the state integration routine increments time if the flight has not ended and calls the other modules again.

The 6DOF can be executed in a Monte Carlo mode that provides random variations to specified variables on successive runs. The 6DOF has missile linear and angular accelerations, rates and position, event times, and line of sight rates and accelerations as input data. In addition, there is a large input data file including aerodynamic coefficients, misalignments, wind data, inertial sensor compensation coefficients, inertial sensor errors, jet reaction control errors and delay times, propulsion data and target data.

3.3 Guidance and Control Unit

The GCU component provides the software infrastructure that allows the GCU software to be executed in both the HWIL and SWIL modes. In the HWIL mode the 6DOF component provides input and receives output from an actual GCU hardware component through a custom hardware interface. While in the SWIL mode the GCU software is executed on the SGI ONYX under the control of the real-time executive.

3.3.1 GCU Architecture

The core functionality of the GCU component is contained in the GCU code itself. For the SWIL mode this code was re-hosted to the SGI Onyx from the Predator SRAW missile onboard computer with minimal modifications. In the SWIL mode the GCU is essentially a function call to the main function in the GCU code that executes flight control software used to control the flight dynamics of the missile. The execution of the GCU code is controlled by the 6DOF component, which executes once every millisecond. When the Predator SRAW simulation is integrated with the real-world Predator SRAW GCU hardware, the SWIL code is not executed. Instead an interface routine is called that communicates with the Predator SRAW hardware, through a custom hardware interface, and relays all the signals to the rest of the simulation.

3.3.2 GCU States and Modes

As shown in Figure 6, the GCU component has two modes: SWIL and HWIL. In the SWIL mode the 6DOF calls the GCU embedded code directly.

In the HWIL mode the 6DOF component calls the hardware interface routine instead of the GCU embedded code. That interface routine sends the current state to the Predator SRAW GCU hardware through the analog and digital VME boards, then reads the current commands and status flags back from the Predator SRAW GCU hardware.

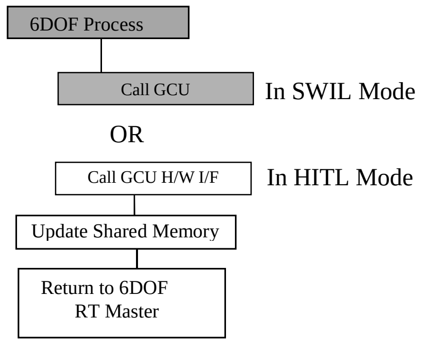

Figure 6: GCU Component Modes & Control Flow

3.3.3 GCU Component

The heart of the GCU compoment is the re-hosted Predator SRAW GCU. For faithful replication of Predator SRAW performance, the code was modified as little as possible. Only those modifications absolutely necessary to re-host it to the SGI were implemented. In the Predator SRAW CPU the GC_INTERRUPT_ SERVICE is an interrupt handler, i.e. it is never called explicitly but rather control is transferred to it via a hardware action - in this case, the firing of a 1 millisecond timed interrupt. Since the 6DOF is controlling the call to the GCU software, the timing is managed by the 6DOF Real-time Sequencer. This essentially causes the 6DOF Real-time Sequencer to act as the interrupt handler when in the SWIL mode.

3.4 Target Detection Device

The TDD component consists of the software required to execute the Predator SRAW onboard Target Detection software in the SGI in a real-time environment. In addition, the target magnetic signature model is integrated into the TDD component.

3.4.1 TDD Architecture

Because the TDD software updates at variable rates, the use of a Slave Real-time Sequencer was not sufficient to control the target detection algorithms. Instead the TDD component uses software interrupt functions controlled by the 6DOF's MRS to control the TDD code, and to insure that it executes in real time.

The core functionality of the TDD component is contained in the TDD code itself, which is written in Ada. This code was re-hosted from the Predator SRAW missile onboard computer with minimal modifications. For the Predator SRAW Simulations (HWIL & SWIL), the TDD code is executed at variable rates up to once every 200 microseconds.

In addition to the re-hosted TDD, the TDD component also contains a model of the magnetic field associated with the simulation targets. The magnetic signature model was written in C and is called from the TDD via an Ada C-language binding.

3.4.2 TDD States and Modes

The TDD component has three modes: Initialization, Pre-Flight, and Flight. In the Initialization mode the TDD component reads the initialization parameters supplied by the 6DOF component from shared memory and then enters the real-time loop (See Figure 7). In the Pre-Flight mode, the software waits for the TRIGGER_PULL message from the 6DOF code via shared memory indicating that the trigger was pulled.

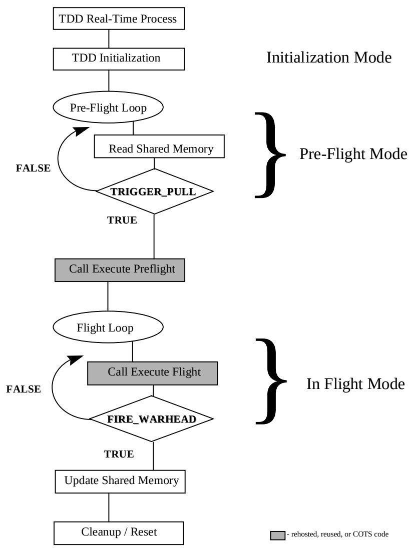

Figure 7: TDD Modes & Control Flow

Once this occurs the software interrupts are enabled and the TDD component enters the Flight mode.

In the Flight mode, the software interrupts control code execution. A single interrupt initiates one pass through the TDD code. When the TDD completes one pass, it sets up the next software interrupt time and waits. This real-time loop continues until the detection algorithms initiate a FIRE_WARHEAD message or the 6DOF component sets a Boolean flag that indicates that the simulation run is complete.

3.4.3 TDD Component

The heart of the TDD component is the re-hosted Predator SRAW on-board target detection code. The original code uses hardware interrupts to call the three main routines in the Predator SRAW TDD. The remainder of the Predator SRAW TDD code is designed to perform system services in the Predator SRAW TDD CPU. Because either UNIX or the MRS provides those services in the simulation, that code is not required. To faithfully replicate Predator SRAW performance, the code was modified as little as possible.

The other major piece of the TDD component is the target Magnetic Model. In order to feed sensor data to the Predator SRAW TDD software, the simulator has to contain a model of the appearance or signature of the target(s) as they would appear to the TDD sensors. The model of the target magnetic signature is based on several reports provided by Naval Surface Warfare Center (NSWC), Dahlgren, VA. These studies indicated that a second order model would provide the required accuracy.

4 VERIFICATION AND VALIDATION

The Verification and Validation (V&V) process applied rigor to the development of the Predator M&S System. Verification and Validation were conducted concurrently with M&S system development, which reduced overall risk to the program. In addition, periodic scrutiny of key components of the M&S system ensured that the developer was building exactly what the customer needed and thus held cost in check. M&S System verification was accomplished by disciplined review of the products produced during each phase of the development process. M&S System validation was accomplished by comparison of M&S System output with telemetry data from real world developmental test firings.

4.1 V&V Process

The "DMSO VV&A Recommended Practices Guide" includes finely crafted definitions of verification and validation as it refers to modeling and simulation. Simply stated, the verification process answers the question "did you build what you said you would?" The validation process answers the question "does the model work right?" The Predator M&S System was developed using a waterfall development process. Each stage of the development process required specific deliverables. Deliverables produced during each development phase were used to support model verification.

The IV&V Team reviewed the Preliminary Design Document, Critical Hardware Design Document and Critical Software Design Documents in support of verification. The previously mentioned documents largely supported verification, however some additional documents were required. For example, one of the unique aspects of the Predator M&S System was the requirement to verify that the M&S System was using production representative GCU Software and TDD Software. In addition, the rehosted software must function exactly as it would in the actual missile. Identical software function was a concern, because the Predator GCU and TDD Software were targeted for a Motorola 68332 Processor, while the Predator M&S System runs on an SGI Onyx Quad R10000 Processor. In order to verify that the rehosted software was functioning properly static and dynamic analyses were conducted using McCabe Tools. The McCabe Tools were used to statically inventory and dynamically trace run time execution of the rehosted software. The static and dynamic analysis verified that the rehosted software was functioning as advertised.

4.2 Analysis Methodology

For the purpose of supporting the VV&A process for the Predator simulation system, the following analysis methodology was employed:

1. An analysis was performed to determine the appropriate sample size needed to provide the desired level of confidence that the simulation is accurately representing the behavior of the real system.
2. Telemetry data is captured from fifteen identical live fire tests. Identical implies that the firing conditions (e.g. target range, target aspect, missile temperature, …) for the live fire tests were matched as closely as possible.
3. An equal number of simulated Monte Carlo runs are executed using the same initial conditions present at the live fire tests.
4. Data from the simulated runs is compared to data from the live fire tests. This comparison determines if the simulation is replicating the flight characteristics of the real missile within some tolerance specified by the USMC and the VV&A team.

4.3 Data Comparison

The validity of data provided by the Predator SRAW M&S System may be determined by comparing live telemetry data with the output of the simulation. To conduct this comparison, a common set of variables was selected.

The Predator SRAW GCU and TDD function independently and two aspects of missile vs. simulation performance must be contrasted. The first aspect is actual missile position as compared to simulated missile position throughout the flight of the missile. The second aspect of missile performance that must be considered is actual TDD target detection during real firings as compared to simulated TDD target detection.

Consider missile position first. During an actual flight, missile position and state at any given point may be fixed and measured in three dimensions X, Y and Z. The X-axis refers to the location of the missile as it moves down range towards the target. The Y-axis refers to missile position as it translates right or left. The Z-axis refers to missile elevation. Parameters of interest in each dimension include position, velocity and acceleration.

Next consider TDD performance. TDD parameters are important in determining where the simulated missile detected and detonated over the target. Several TDD flags are used to assess simulation performance and make some comparison with telemetry runs. Because the TDD is a dual mode device, flags associated with the Laser Ranger, Magnetic Detector and Warhead detonation are of interest.

These flags are related to optical and magnetic signature detection and detonation commands.

Subsequent analysis is based upon a comparison of missile position and TDD performance as observed in the missile and as set in the simulation during flyout. Data generated by the Monte Carlo runs of the simulation, for both the GCU and TDD, is compared to the telemetry data from the test flights.

5 CONCLUSION

The Predator simulation system is an excellent example of how simulation can be applied to the system development process to help reduce cost and ensure a higher quality product. The success of the Predator simulation system shows that with a complete, coherent VV&A process in place, a simulation system can be developed to provide valuable support across the entire system development life cycle.

One of the significant concerns raised when developing a hard real-time system, with component executing at frame rates in excess of 1000Hz, is the ability to model physical factors, environmental factors and sensor inputs that precisely represent the dynamics and inputs to the system. The Predator simulation system has shown that this can be done. This is not to say that one should not carefully choose the applications to be simulated. There are many examples of failed simulation efforts, but judicious analysis and design can greatly reduce the risk of failure.

With companies such as Boeing and Ford relying more and more on simulation to support their design, development, and manufacturing processes, many new areas of simulation research and application are opening up.

AUTHOR BIOGRAPHIES

BRADFORD S. CANOVA is a Principal Software Systems Engineer at the MITRE Corporation. He provides technical guidance to numerous programs at MITRE in the areas of simulation, real-time systems, and distributed systems. In addition, he acts as the program area manager for Naval Programs within the Information Systems and Technology Division at MITRE. He holds a B.S. in Computer Science from the Pennsylvania State University.

PETER H. CHRISTENSEN has been employed by the MITRE Corporation as a Lead Member Technical Staff since January 1995. Prior to that he served with the United States Navy as a Naval Flight Officer. At MITRE Pete was assigned to the Software & Information Architecture Technology Area, to support both Modeling and Simulation and C4I Programs. In the Navy he accumulated over 2200 hours in the EA-6B Aircraft and has extensive experience in Command and Control, Electronic Warfare and Computer Science. He was the first recipient of the Prowler Systems Excellence Award. Flew 35 Combat Missions during Desert Storm. He served as Suppression of Enemy Air Defense (SEAD) Lead for several key Air Wing Five night strikes into Iraq throughout Desert Storm. He retired as a Commander in 1995 from his final billet as a Deputy Program Manager at the Naval Air Systems Command.

BRUCE R. TRIPP is a Senior Software Systems Engineer at the MITRE Corporation. He is the Technical Lead for the Predator/SRAW Modeling and Simulation System. He holds a B.S. in Physics from the University of South Florida and an M.S. in Studies of the Future from the University of Houston at Clear Lake. In addition, he has 20+ years experience in real-time hardware-in-the-loop simulation.

MICHAEL D. LEE is a Senior Software Systems Engineer at the MITRE Corporation. Mr. Lee provides modeling, simulation and visualization support for MITRE. He received his B.S. in Physics from Mary Washington College in 1994.

MICHAEL PACK is a Software Systems Engineer for the MITRE Corporation, where he was hired in January 1998. He graduated from West Virginia University Institute of Technology in December 1997 with a B.S. in Computer Science.

DAVID L. PACK is a Software Systems Engineer at The MITRE Corporation. Mr. Pack provides modeling and simulation support for MITRE. He received his B.S. in Computer Science from West Virginia University Institute of Technology in 1997.