Real-Time Hardware-in-the-Loop Simulation of Ares I Launch Vehicle

Patrick Tobbe Ph.D.[1], Alex Matras Ph.D.[2], David Walker[2], Heath Wilson[2], Chris Fulton[2], and Nathan Alday[2]
Dynamic Concepts, Inc., Huntsville, AL, 35806

Kevin Betts[3], Ryan Hughes[4], and Michael Turbe[5]
Science Applications International Corporation, Huntsville, AL, 35802

The Ares Real-Time Environment for Modeling, Integration, and Simulation (ARTEMIS) has been developed for use by the Ares I launch vehicle System Integration Laboratory at the Marshall Space Flight Center. The primary purpose of the Ares System Integration Laboratory is to test the vehicle avionics hardware and software in a hardware-in-the-loop environment to certify that the integrated system is prepared for flight. ARTEMIS has been designed to be the real-time simulation backbone to stimulate all required Ares components for verification testing. ARTEMIS provides high-fidelity dynamics, actuator, and sensor models to simulate an accurate flight trajectory in order to ensure realistic test conditions. ARTEMIS has been designed to take advantage of the advances in underlying computational power now available to support hardware-in-the-loop testing to achieve real-time simulation with unprecedented model fidelity. A modular real-time design relying on a fully distributed computing architecture has been implemented.

I. Introduction

THE United States and NASA have committed to building the Ares I Crew Launch Vehicle as the man-rated launch vehicle to support the Constellation program[1]. The Ares I will carry the Orion Crew Exploration Vehicle (CEV) into orbit for visits to the International Space Station (ISS) as well as future manned missions to the lunar surface. The Ares I vehicle is composed of multiple elements, including the First Stage Reusable Solid Rocket Motor V (RSRMV), the Upper Stage powered by the J-2X engine, and the Interstage used to connect the two primary stages. The Orion CEV also consists of multiple elements (the Command Module and Service Module) as well as the Launch Abort System (LAS). At launch, the integrated vehicle stack is composed of these stages, and throughout the mission, various elements separate from the integrated stack and return through the atmosphere towards the Earth's surface.

The Ares Real-Time Environment for Modeling, Integration, and Simulation (ARTEMIS) is software designed to simulate the Ares I launch vehicle in order to test and verify proper operation and integration of avionics systems across the various stages. This software is developed for use in the Ares I System Integration Lab (SIL) at Marshall Space Flight Center (MSFC). This must be capable of running in a Hardware-in-the-Loop (HWIL) environment for testing of actual avionics components. The software must also include all digital simulations of the avionics components, vehicle subsystems, and flexible-body dynamics.

The Ares avionics architecture consists of components in the Upper Stage and First Stage, including a variety of input/output (I/O) communication protocols, such as MIL-STD-1553B, EIA/TIA-422-B, Gigabit Ethernet (GbE), and analog signals. Within the HWIL test facility, ARTEMIS must be capable of simulating and recording data on each of these bus types. The avionics systems must send data across stages as well as communicate with all boxes within each stage's local avionics ring. In order to test failure modes not easily inserted into the avionics boxes, ARTEMIS must include all digital models of each box. This fault insertion capability requires component

[1] Chief Engineer, 6700 Odyssey Drive, Suite 202, AIAA member
[2] Engineer/Scientist, Simulation, Model, and Test Division, 6700 Odyssey Drive, Suite 202, AIAA member
[3] Engineering Director, Space Systems Development Division, 600 Boulevard South, Suite 304, AIAA Member
[4] Aerospace Engineer, Space Systems Development Division, 600 Boulevard South, Suite 304, AIAA Member
[5] Avionics Engineer, Space Systems Development Division, 600 Boulevard South, Suite 304, AIAA Member

simulation nodes capable of emulating avionics box functionality as well as communicating over the flight I/O data busses.

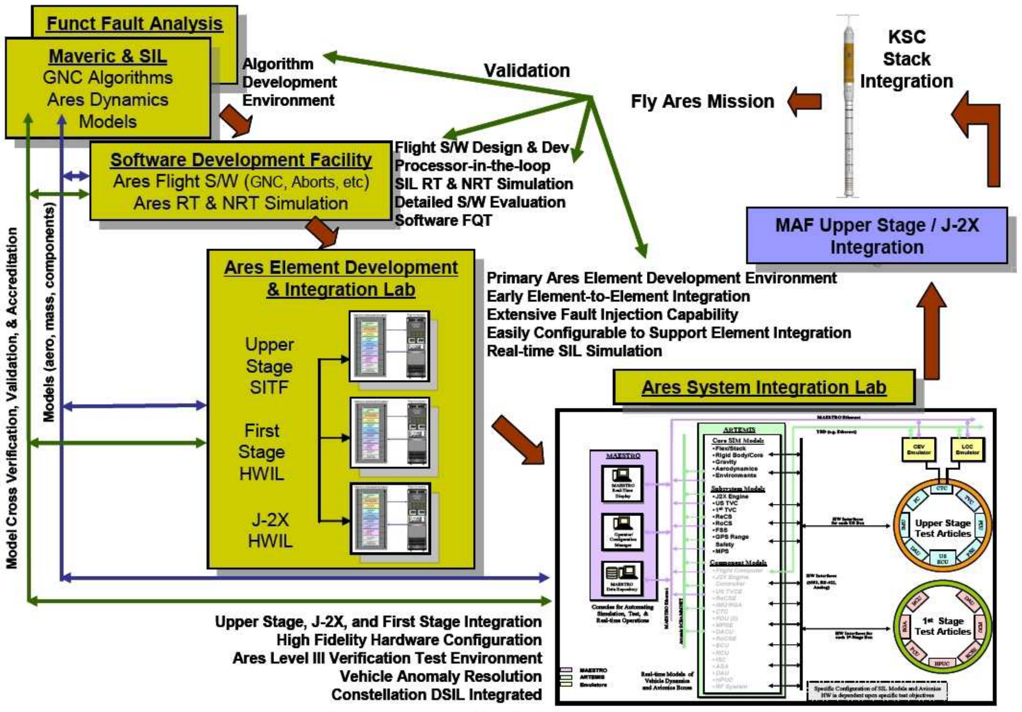
ARTEMIS must be capable of simulating the integrated stack during flight as well as propagating each individual element after separating from the vehicle. In addition, abort sequences can lead to unique configurations of the integrated stack as the timing and sequence of the stage separations are altered. In order to simulate nominal and abort conditions of the vehicle, and provide realistic sensor inputs, ARTEMIS must accurately model the dynamics and subsystems of the Ares I. The dynamics of the Ares I vehicle include significant interactions between vehicle flexible body effects, propellant slosh, and vehicle nozzle inertia effects as well as mass and flexible body properties that vary significantly during flight. Vehicle subsystems that cannot be physically tested in the laboratory must also be modeled with high fidelity inside the HWIL simulation. Examples of subsystem models include propellant flow through the fuel lines and engines, actuator nozzle dynamics, and engine combustion.

The following sections discuss how ARTEMIS has been designed to satisfy the unique requirements of the Ares I SIL. Section II provides an overview of the Ares integration and test facilities that will use ARTEMIS as the simulation backbone. Section III briefly describes the ARTEMIS software components. Section IV discusses the modeling and simulation components in more detail, while Section V presents information on the distributed real-time architecture. Section VI offers some conclusions and a description of the forward work leading to the first Ares I launch.

II. Ares Integration and Test Facilities

ARTEMIS has many applications across the verification of the Ares I vehicle. ARTEMIS will be used in the Upper Stage Software Development Facility (SDF), which supports Ares flight software design, development, test, integration, and verification of the flight computer and Command and Telemetry Computer (CTC). The SDF integrates hardware and software for the flight computer and CTC while ARTEMIS models the remainder of the Ares I vehicle to perform all-digital processor-in-the-loop tests of the flight software and vehicle interfaces.

Three Ares element development and integration labs will then use the software developed in the SDF to perform additional integration and testing with the respective element of the vehicle. The labs, illustrated in Figure 1, are the Upper Stage System Integration and Test Facility (SITF), the First Stage HWIL lab (HIL), and the J-2X HWIL lab. Each element lab is responsible for verifying the interfaces and functionality between all of its avionics components and hardware with the flight computer and other external connections. The SITF will contain all of the Upper Stage avionics hardware and use ARTEMIS to simulate all remaining Ares I avionics components and the vehicle flight.



Funct Fault Analysis
Maveric & SIL
GNC Algorithms
Ares Dynamics
Models
Algorithm
Development
Environment
Validation
KSC
Stack
Integration
Fly Ares Mission
Software Development Facility
Ares Flight S/W (GNC, Aborts, etc)
Ares RT & NRT Simulation
Flight S/W Design & Dev
Processor-in-the-loop
SIL RT & NRT Simulation
Detailed S/W Evaluation
Software FQT
MAF Upper Stage / J-2X
Integration
Ares Element Development
& Integration Lab
Primary Ares Element Development Environment
Early Element-to-Element Integration
Extensive Fault Injection Capability
Easily Configurable to Support Element Integration
Real-time SIL Simulation
Upper
Stage
SITF
First
Stage
HWIL
J-2X
HWIL
Model Cross Verification, Validation, & Accreditation
Models (aero, mass, components)
Ares System Integration Lab
MAESTRO
MAESTRO Real-Time Display
Operator/ Configuration Manager
MAESTRO Data Repository
Consoles for Automating Simulation, Test, & Real-time Operations
ARTEMIS
Core SIM Models
•Flex/Stack
•Rigid Body/Core
•Gravity
•Aerodynamics
•Environments
Subsystem Models
•J2X Engine
•US TVC
•1st TVC
•ReCS
•RoCS
•FSS
•GPS Range Safety
•MPS
Component Models
•Flight Computer
•J2X Engine Controller
•US TVCE
•ReCSE
•IMU/RGA
•CTC
•PDU (2)
•MPSE
•DACU
•RoCSE
•ECU
•RCU
•ISC
•ASA
•DAU
•HPUC
•RF System
Real-time Models of Vehicle Dynamics and Avionics Boxes
MAESTRO Ethernet
TBD (e.g. Ethernet)
CEV Emulator
LOC Emulator
HW Interfaces for each US Bus
HW Interfaces (1553, RS-422, Analog)
HW Interfaces for each 1st Stage Bus
CTC
TVC
PCU
FSE
US ECU
DAU
IMU
FC
Upper Stage Test Articles
1st Stage Test Articles
DAU
PDU
RCSE
HPUC
PCU
RGA
IMU
MAESTRO
ARTEMIS
Emulators
Specific Configuration of SIL Models and Avionics HW is dependent upon specific test objectives
Upper Stage, J-2X, and First Stage Integration
High Fidelity Hardware Configuration
Ares Level III Verification Test Environment
Vehicle Anomaly Resolution
Constellation DSIL Integrated

Figure 1. Ares I test facilities using ARTEMIS

The HIL will be similar to the SITF in that it will have hardware-in-the-loop for all of the First Stage components with ARTEMIS providing simulated components for all other stages during a test. The J-2X HWIL will use the ARTEMIS simulation for the Ares I vehicle; in addition, the J-2X engine hardware will interface with ARTEMIS in this lab.

Testing will then progress to the integrated vehicle level. The Ares I Software Integration Laboratory (SIL) will be used to verify Ares I avionics and system requirements and to validate Ares I system performance through HWIL testing. Figure 2 shows the overall architecture of the SIL. The SIL will contain Flight Equivalent Units for both the Upper Stage and First Stage avionics components as well as the J-2X engine controller. The laboratory will mount avionics systems in flight-like avionics rings and will use flight-representative cables to interconnect the systems. As shown in the green box in the center of Figure 2, ARTEMIS is the simulation backbone used to stimulate the Ares avionics components and flight software. The Managed Automation Environment for Simulation, Test, and Real-time Operations (MAESTRO) software will be used by lab operators and testers to command the SIL, monitor data in real-time through 2D and 3D displays, and control data recording and archiving of simulation results.

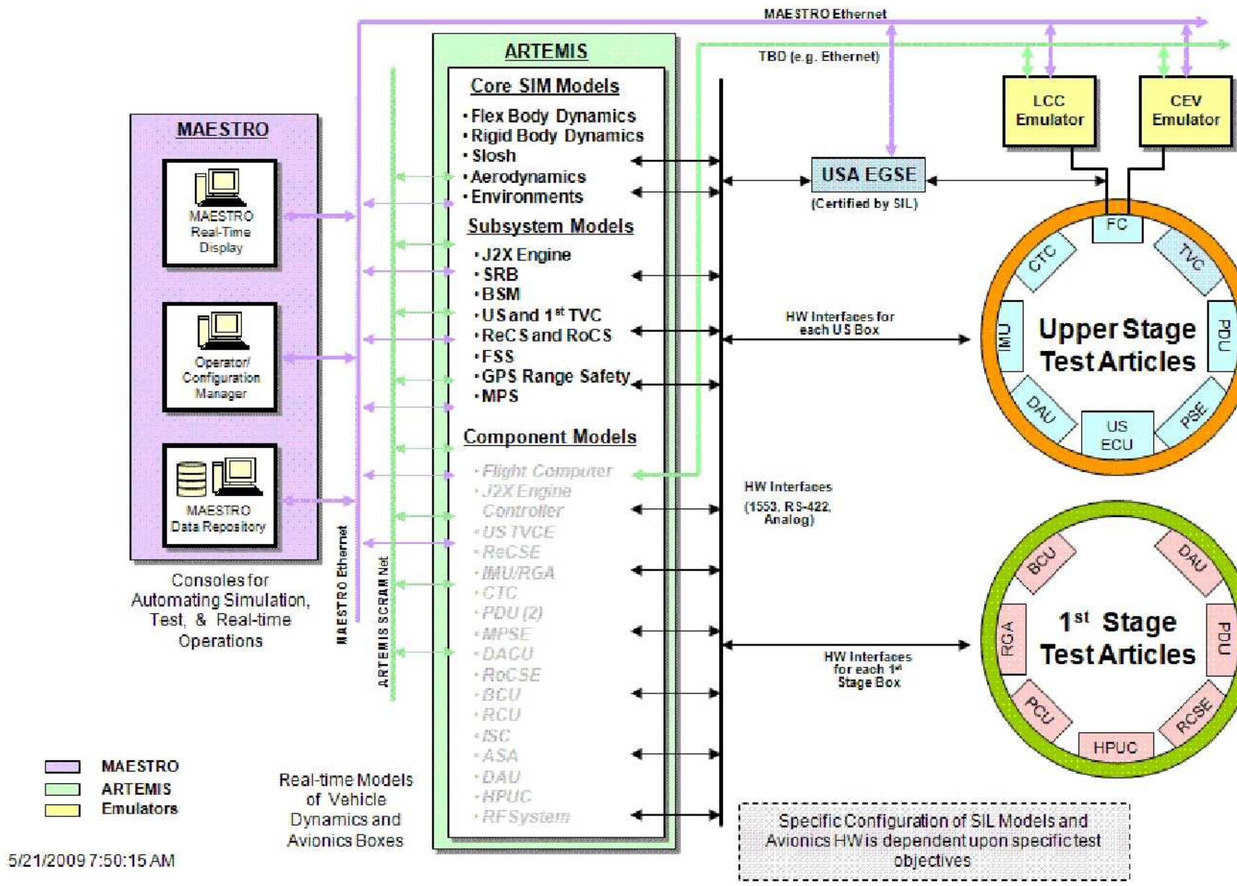


Figure 2. Ares SIL architecture diagram

ARTEMIS will also be used to drive the Ares Emulators. The Ares Emulators are integrated hardware/software systems that will be delivered to other Constellation test facilities. Examples of these facilities include:

- The Orion CEV Avionics Integration Laboratory (CAIL) at Johnson Space Center (JSC) in Houston, TX
- The Ground Operations (GroundOps) Test Facilities at Kennedy Space Center (KSC) in Cape Canaveral, FL

ARTEMIS will simulate all Ares functions and provide data to the Orion and GroundOps through the flight interfaces as well as simulation-to-simulation interfaces as required. In a similar fashion, Orion and GroundOps Emulators will also be delivered to the SIL in order to simulate the functionality of those components during integrated Ares avionics testing.

Additional applications of ARTEMIS include testing during Upper Stage and J-2X hardware integration at the Michoud Assembly Facility (MAF) and during vehicle stacking on the mobile launcher at KSC. After each Ares mission has been completed, flight test data will then be used to validate all of the Ares development labs previously discussed.

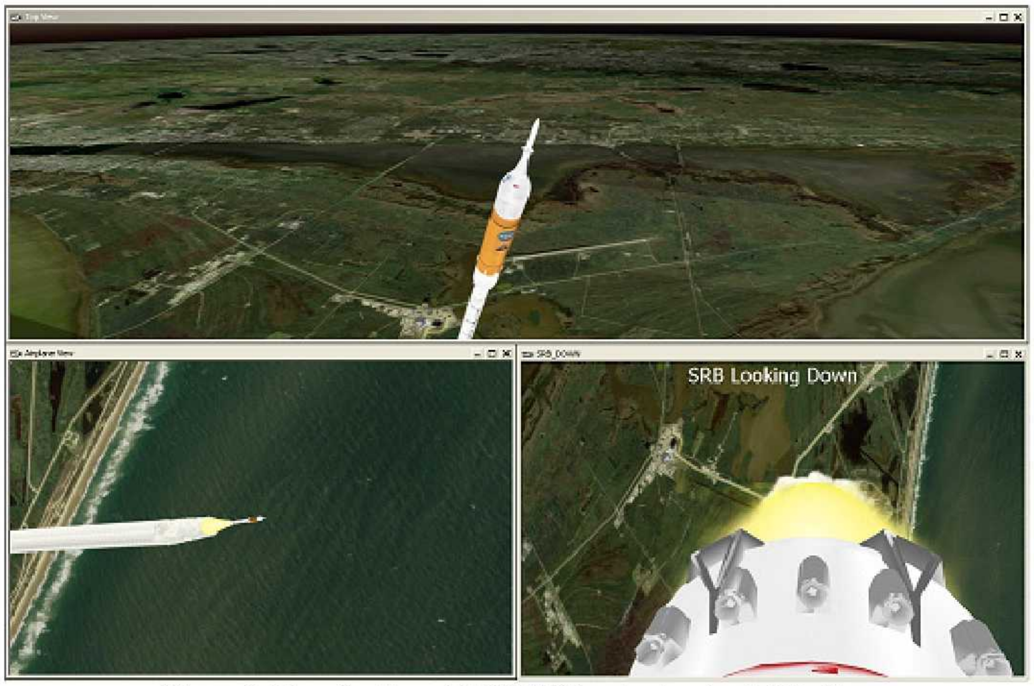


Figure 3. Visualization of ARTEMIS outputs using bdStudio

III. ARTEMIS Overview

The majority of ARTEMIS is written in C and is designed to allow distribution across multiple computing nodes. ARTEMIS can be run in the open source Linux environment for development and non-real-time simulation purposes; it is run under Concurrent's RedHawk Real-Time Operating System when hard real-time performance is required. There are five primary functional software components within ARTEMIS: Models, Simulation Infrastructure, Timing, Data Input/Output (I/O), and Data Recording. ARTEMIS also contains interfaces with the MAESTRO lab command and control software as well as bdStudio[2], the 2D and 3D visualization program used to render trajectory results during real-time tests in the SIL. Figure 3 shows a bdStudio screen shot of ARTEMIS output from a test run.

The following section discusses each of the major software components and some of the driving requirements and design features that differentiate ARTEMIS from other HWIL simulations developed in the past by NASA.

Models

The Models software component contains all of the code necessary to simulate the Ares vehicle and associated avionics components and subsystems. Due to the number and complexity of vehicle system models, ARTEMIS must take advantage of today's relatively inexpensive multi-processor computing hardware to run an extremely high-fidelity simulation in real-time for avionics testing.

The Models are divided into three major categories: (1) Core Simulation models, (2) Subsystem Models, and (3) Component Models. The core simulation models consist of rigid body and flexible body dynamics equations of motion as well as the environment models, such as atmosphere, aerodynamics, and gravity, necessary to simulate the external forces, moments, and trajectory of the vehicle. A subsystem model is a digital physics-based model representing the vehicle's physical subsystems that are not typically tested in the laboratory. Examples of subsystem models include the thrust produced by the firing of a reaction control system (RCS) jet, the accelerations, and angular rates sensed by the accelerometers and gyroscopes of the Inertial Navigation System (INS), and the flow of fluids through the Main Propulsion System (MPS) and the engines. A component model is a digital model that represents the functionality of an actual Ares avionics box. The component models will be used during laboratory build-up prior to development hardware being available, as well as for simulating select faults that cannot be easily exercised in a real avionics box. This must also be capable of commanding ARTEMIS to start at select pre-defined simulation checkpoints, including pre-tanking, post-tanking, and immediately before launch.

Simulation Infrastructure

This software component encompasses the overall simulation infrastructure necessary to run a hard real-time physics-based simulation. It is responsible for commanding the simulation phase (initialization, run, and shutdown) as well as other functions such as input data processing, error handling, fault insertion, Monte Carlo dispersion control, and interfacing with the MAESTRO test configuration and control software. This also contains the libraries of code necessary for numerical integration and other mathematical operations needed in a physics-based simulation.

One of the key drivers is a requirement to have a single tree of source code that could be used to run in all required ARTEMIS modes of operation, including:

- Non-real-time: All digital (desktop environment for developers)
- Real-time, multi-processor: All digital (no hardware boxes in the loop)
- Real-time, multi-processor: Partial HWIL, partial digital (mix of hardware boxes and component models)
- Real-time, multi-processor: Full HWIL (all avionics components are in the loop)

Timing

The Timing software component is responsible for maintaining real-time operation and synchronization for the simulation. The Timing component must be capable of calling all ARTEMIS models at the required frequency and maintaining overall control of the simulation loop. It must be capable of running in a real-time mode or a non-real-time mode as specified by the ARTEMIS user. The Timing module is responsible for triggering fault insertions during simulation runs, either through overwriting data input/output variables or through triggering embedded flags in the models.

Data Input/Output (I/0)

The Data I/O software component must be capable of simulating all data being transferred as part of the Ares I flight architecture. The Ares I will use a wide variety of both digital and analog transmission methods across various components and subsystems. The number of different transmission protocols being used, as well as the density of data being used, makes the I/O requirements for ARTEMIS one of the key design drivers. Examples of digital busses used on the vehicle include MIL-STD-1553B, GbE, and various serial interfaces (e.g. EIA/TIA-422-B). In addition, the Data I/O Component is responsible for the transfer of model-to-model simulation data; through shared memory (single box configuration) or reflective memory (distributed configuration).

Data Recording

The Data Recording software component must be capable of recording all flight avionics bus traffic as well as all simulation model-to-model communications. The Data Recorder must also be capable of providing local data recording to capture internal model variables as needed for initial simulation integration and debugging.

IV. Models and Simulation

A. Core Simulation Models

The Core Simulation models include vehicle and fuel slosh dynamics, mass property calculations, atmosphere and winds, vehicle aerodynamics, gravity, and the trajectory.

1. Vehicle Dynamics and Mass Properties

The mass properties model calculates composite vehicle mass properties at each integration step of the simulation. The model includes the contribution of the structural mass of each stage as well as time-varying components based on propellant levels and mass in each tank. The mass properties model uses NASA Stress Analysis (NASTRAN) structural output files directly as the input files for the simulation. These files include structural properties for each component of Ares as well as flexible body properties for the integrated stack vehicle.

The flexible body dynamics uses the assumed modes method to calculate the bending effects of the vehicle3. A 500+ degree of freedom (DOF) model was created for ARTEMIS by reducing a 500,000 DOF model of the integrated stack. Modal frequencies and mode shapes were compared between the two models to ensure the reduction captured the necessary fidelity for simulation purposes.

The vehicle flexible body characteristics are calculated using vehicle mode shapes for the integrated stack. The time-varying nature of the mass matrix (and hence the modal mass matrix) of the system allows the modal frequencies to change during the simulation run to match the natural frequencies of the vehicle with a particular level of propellant loading. Figure 4 shows a screen shot from bdStudio of the vehicle flexure; this display is available for real-time rendering while the simulation is being run.

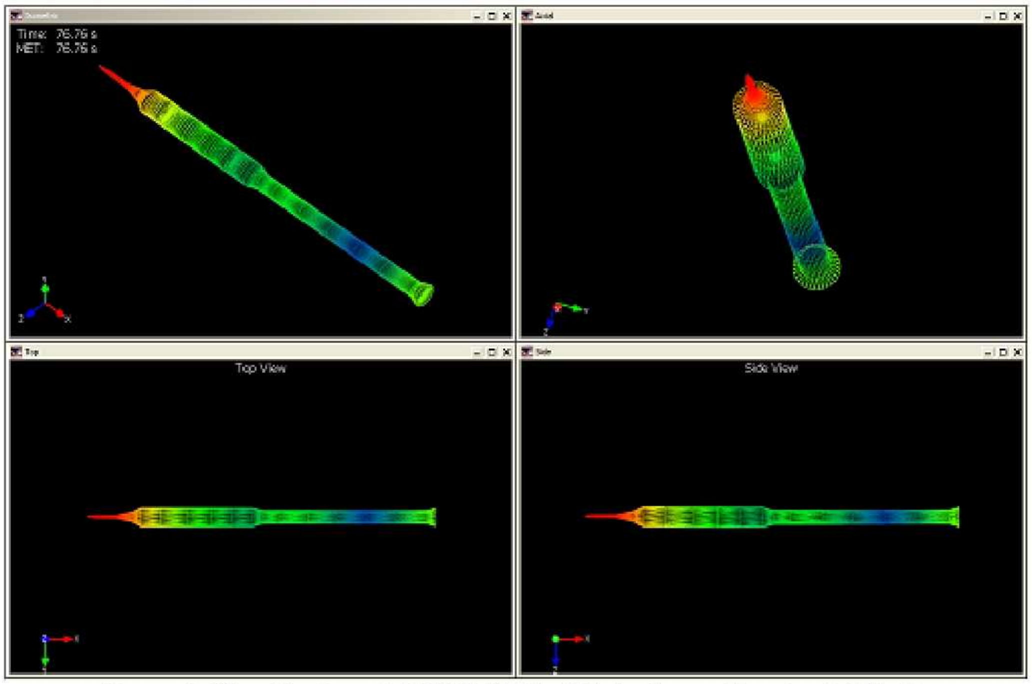


Figure 4. Visualization of ARTEMIS flexible body results using bdStudio

The propellant slosh model is part of the integrated flexible dynamics model. The mass, stiffness, and damping elements that represent the major propellant tanks in the vehicle flexible body properties are calculated using a special NASTRAN plug-in (HYDRO) that is designed to model hydrodynamic flow.

2. *Atmosphere & Winds*

Two atmosphere models are implemented in ARTEMIS. The first model is the 1976 U.S. Standard Atmosphere[4] model (US76), which uses a table lookup to provide the temperature and pressure, and then calculates the density, speed of sound, and dynamic viscosity. The second atmosphere model is the 2007 version of the Global Reference Atmospheric Model (GRAM2007), which was developed at the Marshall Space Flight Center (MSFC). This model is written in FORTRAN 77 and utilizes a C wrapper to interface with the simulation.

A separate winds model was added for the reference winds at the Kennedy Space Center to be used in conjunction with the US76 model. GRAM2007 also contains a winds model that provides wind velocity, direction angle, and magnitude in addition to the parameters of the GRAM2007 atmosphere model. A ground winds model has also been implemented for a more accurate representation of the wind environment while the vehicle is sitting on the launch pad.

3. *Aerodynamics*

Three aerodynamics models are currently implemented in the ARTEMIS simulation. The first model, the lumped aerodynamics model, utilizes tables of aerodynamics coefficients indexed by various variables, such as Mach number, altitude, angle of attack, and sideslip. The second aerodynamics model, the distributed aerodynamics model, calculates the aerodynamic forces and moments at each specified node on the vehicle. The distributed aerodynamics model is only used for the stack during the first stage of flight, and all other vehicles use either the lumped aerodynamics model or no aerodynamics model at all. The third aerodynamics model calculates the aerodynamic forces on the vehicle while it is sitting on the launch pad in a similar fashion to the distributed aerodynamics model.

4. Gravity

The gravity model provides the gravity force and gravity-gradient torque using the vehicle's inertial position and mass properties. The available models are a simple, Keplerian gravity model, a fourth-order gravity model, and the Gravity Recovery and Climate Experiment (GRACE) model[5], which is valid up to degree and order 200.

5. Trajectory Calculation Utility

This code uses the vehicle state vector and the initial launch parameters to calculate various trajectory parameters such as altitude, relative velocity, dynamic pressure, and angle of attack. These intermediate calculations are used as index variables for look-up tables for properties (such as aerodynamics) as well as being useful for real-time analysis of simulation results.

B. Subsystem Models

The subsystem models represent physical subsystems such as thrusters, engines, actuators, and sensors that will not be physically present in the SIL; these will always be simulated by ARTEMIS using physics-based models. The following sections provide some details on the critical subsystem models that are required for closed-loop flight control for the vehicle.

1. Redundant Inertial Navigation Unit (RINU) and Rate Gyro Assembly (RGA)

The RINU and RGA models are responsible for simulating the output of Ares I inertial flight sensors. The RINU model is responsible for calculating the full six-degree of freedom translation and attitude solution of the Ares vehicle for use by the Guidance, Navigation, and Control (GN&C) software algorithms. The RINU model includes realistic error models to simulate the imperfect sensing of rotation and acceleration by the RINU's gyroscopes and accelerometers. The RGAs are located at various points on the vehicle and are used only to measure angular rotation. The outputs of these gyroscopes are blended together to help compensate for the effects of vehicle flexure to provide a composite angular rate signal for use by the vehicle control system.

2. Main Propulsion System (MPS)

The MPS model simulates the fluid flow of propellants through the tanks, pipes, and valves used by the Ares vehicle. The current MPS model is an intermediate tanking model intended for aiding in testing the pre-launch tanking procedures within ARTEMIS. It simulates filling the hydrogen and oxygen tanks by reading the liquid flow rate from the umbilical interfaces and integrating that flow rate as the amount of fluid contained in the Ares second stage tanks changes. Figure 5 shows a bdStudio schematic of the MPS system that can be driven in real-time by MPS subsystem model data. The current MPS model used in ARTEMIS will be replaced by a high-fidelity model provided by the MPS design team at MSFC.

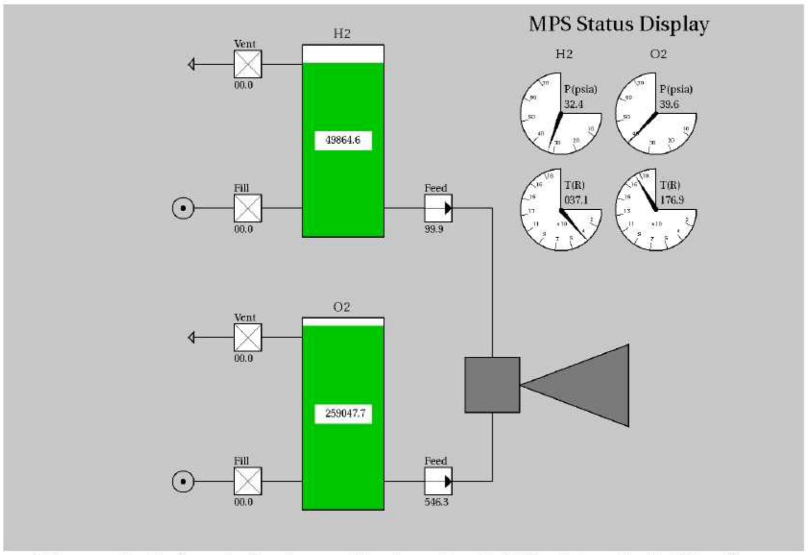


Figure 5. Animated schematic showing MPS status in bdStudio

3. Thrust Vector Control (TVC)

The TVC subsystem model is responsible for simulating the actuators and hydraulic control system used to steer the engine nozzle of the vehicle in order to direct the rocket thrust along the desired vector. The TVC model supports four levels of fidelity: an ideal model, a second-order model, a high-fidelity simplex model, and a tail-wags-dog model. In the ideal model, the nozzle angular positions are set equal to the commanded angles. In the second-order model, a discrete filter mathematically introduces dynamics into the nozzle's response to the commands while limiting rates and positions. The high-fidelity simplex model calculates an applied actuator load force using the electrical current commands from the Booster Control and Power Distribution Unit (BCPDU). This load is then output to a nozzle dynamics function, which uses a separate integration job to calculate the nozzle angular rates and position. Additionally, several outputs from the nozzle dynamics model feed back into the TVC model. The tail-wags-dog model fully couples the dynamic motion of the nozzle and the vehicle through constraint and TVC actuator forces. Future work for the TVC model consists of implementing the model provided by the vendor responsible for the TVC system.

4. Engines & Booster Separation Motors (BSMs)

The engine and BSM models simulate the thrust and mass flow rate for the FS RSRMV and US J-2X engines as well as the three types of BSMs: FS Booster Deceleration Motors (BDMs), FS forward frustum Booster Tumble Motors (BTMs), and US Ullage Separation Motors (USMs). The engine models support both solid and liquid (mono or bi-propellant) configurations. Each engine and BSM model takes in fire commands; in the case of liquid engines, cut-off commands are also received. The engine models also take in nozzle gimbal angles. These models output a force vector at the engine or BSM node location. For certain configurations, a torque vector and propellant flow rates are provided. Vacuum thrust is determined by a table lookup based on burn time or throttle setting; this value is adjusted to account for atmospheric back pressure. Startup and shutdown tables can be used as well, and a maximum of two tanks can be specified for each engine. For the RSRMV, the roll torque due to propellant swirl is calculated using a separate lookup table. The BSM models calculate thrust in a similar fashion to the engine models. Multiple BSM models use the same lookup table and fire command, and one tank may be specified per BSM. The engine and BSM models will be replaced by high-fidelity models provided by the manufacturers.

5. Reaction Control System (RCS)

The RCS models simulate the thrust and mass flow rate for the First Stage Roll Control System (RoCS) and the Upper Stage Reaction Control System (ReCS) to control the attitude of the vehicle. The RCS models take in the individual thruster valve commands and return a force vector at the various RCS node locations. The valve dynamics are modeled to use the incoming valve command along with pre-defined timing parameters to throttle the thruster. The timing parameters vary depending on if the thruster bed is cold, around the first 100 ms of operation, or hot. This model also provides propellant flow rates, and the thrusters can be configured to share propellant tanks or have individual tanks. These models will be replaced by higher fidelity subsystem models provided by the selected RCS vendors.

C. Component Models

The component models represent the avionics boxes. These models are designed to be a digital model of flight hardware and are interchangeable with the flight hardware for testing. There are numerous avionics components due to the complexity of the vehicle and the manned flight redundancy requirements. The following sections provide details on some of the critical avionics components that are required for closed-loop flight control of the vehicle.

1. Flight Computer

The flight computer is responsible for issuing all commands to vehicle subsystems through the components beginning with pre-launch operations. In order to simulate a sample mission, a basic model of the flight computer is needed to allow for a controlled ascent. The ARTEMIS flight computer contains a development version of Ares flight software algorithms that is segmented into the appropriate functional partitions. The partitions include functions such as the mission manager, GN&C, and bus communications. The flight computer bus communications partition contains prototype interfaces for MIL-STD-1553B communication to the BCPDU and GbE communication to the Orion emulator.

2. Command and Telemetry Computer (CTC)

The CTC model provides the interface between the flight computers and the ground, camera controllers, and data recording systems. ARTEMIS will transform the data into the appropriate message format before passing it to the rest of the avionics system.

3. RINU Electronics

The RINU electronics component model simulates the RINU firmware and software that is responsible for calculating vehicle position and attitude from the gyroscope and accelerometer measurements. While the RINU subsystem model is responsible for adding realistic errors to account for sensor imperfections, the RINU electronics component model simulates all functions provided by the RINU hardware, software, and firmware including communications on the flight avionics busses.

4. Combined Control System Electronics (CCSE)

The CCSE model is responsible for commanding the Upper Stage MPS and ReCS subsystems. The CCSE receives commands from the flight computer and sets voltages in an output structure, simulating the output lines of the flight CCSE. These voltages and commands drive the solenoids and pumps of the MPS. The CCSE model also contains the functions that provide the interface between the flight computer and the US ReCS by sending commands to the thruster valves. The CCSE model is also responsible for relaying sensor data from the MPS and ReCS to the flight computer.

5. *Upper Stage Engine Control Unit (USECU)*

The USECU is used to control the J-2X engine to produce the desired vehicle thrust based on the commands from the flight computer.

6. *TVC Electronics (TVCE)*

The TVCE component model represents the electronics used to command the TVC actuators. The TVCE model reflects both hardware and software used to produce the actual TVC actuator commands based on what is commanded by the flight computer.

7. *Booster Control and Power Distribution Unit (BCPDU)*

The BCPDU model performs both remote terminal and bus controller functions. The model receives data from the flight computers and relays it to the appropriate avionics components on the First Stage. The BCPDU model contains the prototype MIL-STD-1553B interface to the flight computer, which sends the data message containing commanded current for the TVC command. The message is then decoded into engineering units in the BCPDU for use by the simulated TVC system. The BCPDU is also responsible for distributing the correct amount of power to the First Stage avionics boxes.

In addition to the flight control critical models discussed above, the vehicle also contains a wide variety of other component models located on the First Stage, Interstage, and Upper Stage. These include: Camera Controllers (CCs), Ares GPS Tracking Unit (AGTU), Altitude Sensor Assembly (ASA), Cryogenic Level Sensor System (CLSS), Flight Safety System (FSS), Ignition & Staging Controller (ISC), Power Distribution & Control Unit (PDCU), Recovery Control Unit (RCU), Radio Frequency (RF) System, and various Data Acquisition Units.

D. Emulator Models

ARTEMIS contains low-fidelity emulators of both the Launch Control Center (LCC) GroundOps and Orion CEV systems that will both be replaced by emulators from KSC and JSC, respectively. The GroundOps emulator contains an initial tanking model for the LH_2 and O_2 tanks. It also has built-in checkpoint start capability that allows the user to start the simulation from four different starting times: Tanks Empty, Tanks Full, Terminal Count, and Launch. An initial model of all of the umbilical interfaces has also been created and tested.

The current Orion emulator contains a model of the Orion flight software, which allows the Ares simulation to respond to abort commands and nominal flight staging events. A prototype GbE interface has been developed in the Orion emulator to communicate with the Ares flight computer over a flight-like bus.

V. Distributed Real-Time Simulation

A. Timing

The core element for the timing of ARTEMIS is the synchronization (Sync) functionality. The Sync program synchronizes the execution of the simulation executables based on the user input frame time to an external clock source -- such as Inter Range Instrumentation Group designation 'B' (IRIG-B) or Concurrent's Real-Time Clock & Interrupt Module (RCIM). It also coordinates the data transfer to and from the shared memory region (or reflective memory when run as a distributed system) to ensure data coherency and interfaces with the MAESTRO software during a SIL test to pass commands from the test engineer to ARTEMIS. In addition, Sync is responsible for inserting faults by writing over data in the shared/reflective memory region when prompted by either a user command or a pre-defined condition. Lastly, Sync sets all of the real-time parameters including locking memory, real-time priority, and running specific processes on different processors and different computers.

Fault insertion is triggered by a user or pre-defined condition and carried out by Sync in one of two ways. The first way is to write over a specific variable in the shared/reflective memory region, which limits the variables that can be faulted. A wider range of faults can be injected using the second method, which requires additional functionality so that a flag specifies a fault or set of faults embedded in the model.

All of the simulation data that is needed for simulation-to-simulation communication is transferred through shared or reflective memory. Reflective memory is used in a distributed simulation; however, development tests can run on a single machine using local shared memory. At the beginning of each software frame, the Sync process copies in all of the data from the shared/reflective memory region to each executable's local memory. Once all processes have completed the computational cycle, Sync allows each process to copy output data back to a specific region of shared or reflective memory that is reserved specifically for that model. The simulation data needed to communicate between the various emulators in the SIL will also use reflective memory to transfer data that is not on one of the flight busses.

To achieve real-time performance, the ARTEMIS executables must be distributed across many processors within multiple computers (Simnodes), which allows for higher-fidelity, computationally intensive models to be used. The models communicate via hardware and software interfaces as described above. This allows each model to have a separate executable, which is valuable for HWIL scenarios. Each model starts independently; however, ARTEMIS utilizes a multi-phased initialization because some models depend on information from others. After each phase of the initialization, all of the interface data is copied to shared memory.

In addition to moving executables from one node to another, a single executable may be threaded across multiple processors on a single Simnode to reduce the impact of computationally robust models. For a robust model, threading one executable may be better than separating the computation into multiple executables. Threading allows all independent sections of a model to be completed in parallel using data from a single frame, while splitting the model into multiple executables would create frame lags. Additionally, more data would likely need to be passed through the shared/reflective memory region in the multiple executable cases because threads share local memory. Currently, the flexible body dynamics model is the most computationally intensive model and requires the highest level of threading. Real-time performance cannot be achieved without threading the individual jobs of the flexible body calculations. Although the threaded jobs must be independent from one another, it is very important to understand the dependencies on the previous and following jobs. Understanding these dependencies is the key to correctly threading jobs in a model.

The grouping of executables on the Simnodes allows the Simnodes to be included or excluded for various HWIL test configurations. The model executables are physically distributed in close proximity to the flight hardware units in the SIL so that a model may be substituted for the avionics hardware during any test. For example, Simnode 1 could contain the simulated flight software, but when actual flight computers are being tested, Simnode 1 would not be used in the test to simulate flight software. In addition, ARTEMIS contains a data recording executable that is required to record all avionics bus traffic, simulation-to-simulation data, and local model data. Local model data recording can also be turned off during a run if necessary.

The test configuration, provided by the MAESTRO user, dictates the Simnodes that are used and the hardware or software being tested. Currently, five Simnodes are necessary to achieve real-time performance. One Simnode is designated as the master node and the other four are the slave nodes. The relationship between the nodes is shown in Figure 6. The current hardware architecture for ARTEMIS includes 64-bit multi-core computing nodes, with the number of cores ranging from 8-16 depending on the required functionality of the box. As discussed previously, the real-time configuration uses Concurrent's RedHawk operating system on all computing nodes.

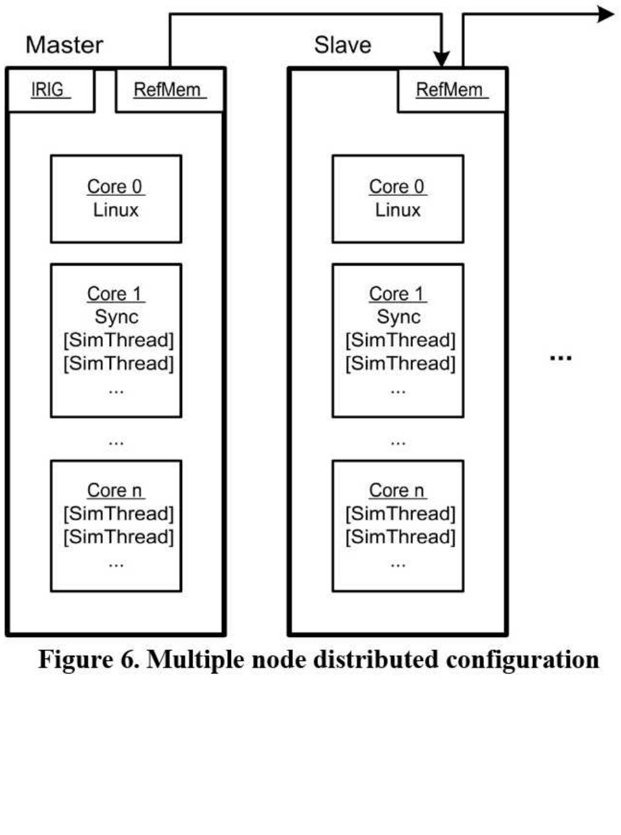


Figure 6. Multiple node distributed configuration

B. Data Recording

Since ARTEMIS will be used for hardware-in-the-loop testing, it must support all of the Ares hardware interfaces for communications (MIL-STD-1553B, GbE, and EIA/TIA-422-B), power, analogs, and discretes, as well as simulation data interfaces between ARTEMIS and the emulators from other NASA centers. ARTEMIS must also simulate any of the hardware interfaces when digital models are used in place of flight hardware for a test.

The majority of the interfaces that are needed in ARTEMIS are for Ares I communications. A custom piece of software has been developed in conjunction with ARTEMIS to handle all of the I/O for these interfaces. This software, the I/O Layer, provides a common set of functions to call from within a component or subsystem model that will initialize the communication device, read or write data in the appropriate format, and close the device. Based on an XML input file, the I/O Layer configures each device and the node where it is installed. The I/O Layer input file also contains a field that tells the simulation if it is running with a simulated device (e.g. no flight-like hardware for that particular interface) so the I/O Layer will then write the data to either shared or reflective memory depending on the test configuration. ARTEMIS is not currently set up to handle the power, analog, or discrete interfaces, but those will become a major focus in the future as each interface is further defined.

C. Data I/O

As previously discussed, the data recorder must be capable of capturing all flight avionics I/O communications as well as simulation model-to-model data transfer. The data recorder must also be capable of capturing local model internal variables. Since complete tests of the Ares avionics system can run from pre-tanking through post Orion separation, the simulation must be capable of capturing high-rate data for over 80 hours of testing. A variety of potential hardware options (including solid state drives) are being evaluated than can hold the required amounts of data and transfer it in real-time to support these long duration tests.

VI. Conclusion

ARTEMIS has been developed to support Ares I HWIL test requirements and has been designed to be as modular as possible to support all required test configurations including the SIL, SITF, SDF, and Ares Emulators. ARTEMIS takes advantage of modern computing power to provide an extremely realistic dynamics and subsystem simulation capability that is a significant improvement in fidelity compared to previous launch vehicle test facilities. ARTEMIS also provides hard real-time determinism and supports the high-density I/O operations required to support Ares I avionics test.

ARTEMIS will be used in the coming years to integrate and test the Ares avionics systems. High-fidelity component models will be provided by the avionics box vendors and will be integrated into ARTEMIS over the next 6 months. Engineering Development Units for the avionics hardware will arrive in 2010 and will be integrated into the laboratories to provide initial test and check-out capabilities. As part of the overall SIL certification effort prior to the commencement of formal testing, ARTEMIS must undergo a significant verification and validation effort. ARTEMIS subsystem and component models will be compared against test results from the actual systems as well as the subject matter expert's critical math model representations for those systems. The flight dynamics and environment models will be compared against independent dynamics simulations used on other parts of the Ares project. The integrated simulation will be validated through comparisons against flight test data for unmanned flight tests. Until flight test data is available, ARTEMIS will compare results against other integrated simulations used on the Ares program as well as test data from other major ground-based test activities such as the Integrated System Test Assembly ground testing and Ares Ground Vibration Testing. ARTEMIS has been designed to be modular and easily upgraded as the computational power of simulation nodes continues to increase. ARTEMIS can be easily adapted to simulate other launch vehicles and extended to support future missions such as Ares V integration and test to support lunar mission development.

Acknowledgments

The ARTEMIS development team would like to thank all of the ES50 Flight & Ground Software Division at MSFC, notably ARTEMIS Lead for NASA Bobby Powell, SIL Team Lead Drew Hall, and SIL Chief Engineer Mark Bartholomae, for their continued leadership and persistence during the development phases of ARTEMIS and the Ares I SIL. The ARTEMIS development team would also like to thank Mark Coffman, Tony Mayhall, Dan O'Reilly, Mike Lee, Rich Cole, and Dyana Beabout for their contributions to ARTEMIS, particularly on the real-time implementation activities.

References

[1] NASA's Exploration Systems Architecture Study; NASA-TM-2005-214062.

[2]Uchitel, V., Turbe, M., Hughes, R., and Betts, K., "bdStudio: Accurate And Easy-To-Use Visualization Tool For Complex Real Time Aerospace Simulations," *AIAA Modeling and Simulation Conference*; Hilton Head, SC, 2007.

[3]Craig, R. R., *Structural Dynamics: An Introduction to Computer Methods*, John Wiley & Sons, New York, 1981.

[4]U.S. Standard Atmosphere, 1976," NASA-TM-X-74335, 1976.

[5]Tapley, B., J. Ries, S. Bettadpur, D. Chambers, M. Cheng, F. Condi, B. Gunter, Z. Kang, P.Nagel, R. Pastor, T. Pekker, S.Poole, F. Wang, "GGM02 - An improved Earth gravity field model from GRACE", Journal of Geodesy (2005), DOI 10.1007/s00190-005-0480-z.