TABLE OF CONTENTS

GENERAL DESCRIPTION ...................................................................................... 4

BENEFITS ........................................................................................................... 5

TYPICAL APPLICATIONS ................................................................................... 7
  OPERATIONAL ANALYSIS ............................................................................. 7
  TRAINING ......................................................................................................... 7
  SIMULATION OF REAL OR SYNTHETIC SYSTEMS ........................................ 7

ARCHITECTURE ................................................................................................. 8
  SCENARIO MANAGER ..................................................................................... 8
  SIMULATION ENGINE ....................................................................................... 9
  STAGE VIEWER ................................................................................................. 10
  STAGE FLIGHTSIM / STAGE HELISIM INTEGRATION ...................................... 10
  VAPS XT INTEGRATION ................................................................................... 10
  AI.IMPLANT INTEGRATION ............................................................................. 11
  DI-GUY™ INTEGRATION .................................................................................. 11
  3-D OPENFLIGHT MODELS ............................................................................ 12
  DEVELOPMENT KIT ......................................................................................... 12

USING STAGE SCENARIO .................................................................................. 14
  GRAPHICAL USER INTERFACE ...................................................................... 14
  COORDINATE SYSTEMS .................................................................................. 15
  TACTICAL DISPLAY ......................................................................................... 15

BUILDING SCENARIOS ....................................................................................... 17
  SCENARIO EDITOR ............................................................................................. 17
  DATABASE EDITOR ............................................................................................ 17

USING THE SCENARIO MANAGER FOR RUNTIME CONTROL ..................... 18
  SIMULATION FLOW CONTROLS ....................................................................... 20
  SNAPSHOT/RESTORE ....................................................................................... 20
  USING MULTIPLE SCENARIO MANAGERS .................................................. 20

BUILDING BLOCKS ............................................................................................. 21
  GAMING AREA .................................................................................................. 21
  ATMOSPHERIC PROFILE .................................................................................. 22
  OCEANOGRAPHIC PROFILE .......................................................................... 22
  PLATFORM PROFILE ........................................................................................ 23
  PLATFORM TYPE AND SUBTYPE .................................................................... 23
  PHYSICAL DATA ............................................................................................... 24
  DYNAMICS AND NAVIGATION ...................................................................... 24
  ATTACHED SENSORS ...................................................................................... 24
  ATTACHED WEAPONS ...................................................................................... 25
  COUNTERMEASURES ...................................................................................... 27
  COMBAT EFFECTIVENESS .............................................................................. 28
STAGE SCENARIO TECHNICAL OVERVIEW

PLATFORM Instances.......................................................... 29
  Initial Conditions.................................................................. 29
  MISSION EDITOR.................................................................... 29
  BEHAVIOR SCRIPTS .................................................................. 30
  FORMATION DATA .................................................................. 31
  BASIC TRAJECTORY .................................................................. 31

WHAT ARE SCRIPTS? ............................................................ 33

EXTENDING STAGE SCENARIO USING THE IDE......................... 35
  EXTENDING THE SCENARIO MANAGER .................................. 35
  RELATIONSHIP TO SIMULATION EXTENSIBILITY ..................... 36
  USING THE IDE ..................................................................... 36

EXTENDING STAGE SCENARIO THROUGH PROGRAMMING........... 37
  EXTENDING THE SIMULATION ENGINE ................................. 37
  EXTENDING THE SCRIPT LANGUAGE .................................. 38
  EXTENDING THE SM .......................................................... 38

LICENSING OPTIONS ............................................................ 39

SUMMARY ............................................................................. 40
GENERAL DESCRIPTION

STAGE Scenario (Scenario Toolkit And Generation Environment) is a software tool used to build and animate, in real-time, synthetic environments containing both moving and stationary entities such as airplanes, ships, land vehicles, missiles and radar sites. These entities interact with one another through pre-determined rule sets or operator intervention during execution of a simulation.

STAGE Scenario provides a graphical user interface—the Scenario Manager—to enter entity parameters into an XML database and assemble them into a dynamic, interactive, complex, and real-time tactical environment. The Scenario Manager includes a set of menus and windows to facilitate the creation of this tactical information, and to control and view the data generated during the runtime simulation.

STAGE Scenario combines the advantages of being both open and extensible, yet easy-to-integrate and requiring minimal programming. It also provides a Mission Editor and a script language that allows entities in the simulation to react to the conditions within their environment.

STAGE Scenario includes a graphical data structure editor and a set of libraries that enable users to extend the capabilities of the core product. Users can add parameters to the entity types provided with the core product, or add new types of entities and define parameters for them. User-written simulation models or legacy code can be integrated into STAGE Scenario models to achieve the required behavior from the simulation. The functionality of the Scenario Manager and mission editor can also be extended.

STAGE Scenario can be used as a totally stand-alone synthetic tactical environment generator, or as a fully integrated simulator for other applications.
BENEFITS

STAGE Scenario is a commercial-off-the-shelf (COTS) product for building custom simulations, in keeping with the recent trend in military procurement towards off-the-shelf products.

The benefits of these COTS products include significant cost savings and superior product support. In fact, STAGE Scenario gives systems integrators a 1 Million plus Source Lines of Code (SLOC) head start in creating complex, custom simulation systems.

STAGE Scenario provides a unique, standards-based solution to developers who are confronted with increasingly aggressive development schedules. It is a proven product that has been used extensively in a large number of simulation projects to shorten development time, simplify scenario creation, and reduce maintenance costs.

Simulation engineers can focus their efforts on customizing the simulation to meet application-specific requirements, rather than expending effort on common features such as scenario generation software, real-time simulation control, shared memory interface, Tactical View and external communications.

STAGE Scenario removes the risk associated with the development of synthetic tactical environments. Users can rapidly prototype a functional simulator with STAGE Scenario entities, then integrate additional high fidelity simulation models. By increasing simulation capability in this way, STAGE Scenario applications can continue to grow with the users’ changing simulation requirements.

STAGE Scenario’s rapid prototyping process allows for an early demonstration of system functionality. The user simply modifies a preset list of parameters and curves using a point-and-click interface to create new systems. No programming is required.

Once the prototype is validated, it can be refined by entering validated data for the systems and the entities and combining them to create scenarios. Aircraft, helicopters, ships, submarines, and armored forces can be created and linked together in a scenario with maneuvers and tactics. The scenario can be executed and controlled in real-time. Various control paradigms can be substituted to provide semi-automated forces.

Furthermore, STAGE Scenario provides the unique flexibility of open, modular simulation architecture. The built-in simulation models of STAGE Scenario can be replaced by other models, with minimal programming, to provide added fidelity. These include weapon systems simulation, sensor propagation algorithms, vehicle dynamics modeling, and other simulation models required to provide a fully cohesive synthetic environment.
STAGE Scenario can link to external simulators, as well as third-party software and hardware. Other simulators can be connected to STAGE Scenario using its own communications technology nCOM, via shared memory, via open standards such as DIS, HLA or CIGI, or via a customer developed protocol integrated into STAGE through the open and extensible architecture.

STAGE Scenario now ships with an out-of-the-box integration with AI.implant. The new integration with AI.implant allows users to choose to leverage artificial intelligence for selected STAGE Scenario platforms. AI.implant makes it easy to accurately model complex crowd & vehicle behavior within simulated environments. With visual behavior authoring and debugging tools, users can create vehicular and human agents for simulation projects that easily scale from a single, intelligent entity to large crowds with autonomous and group behaviors.
TYPICAL APPLICATIONS

Due to its sophistication and flexibility, STAGE Scenario has been used to develop a wide range of simulation applications such as:

- Analysis of tactical scenarios and/or systems in a simulation context, which may include man-in-the-loop interaction
- Training and evaluation of individuals and teams operating in a tactical environment
- Simulation of real or synthetic systems for prototyping purposes
- Monte-Carlo type analysis

OPERATIONAL ANALYSIS

STAGE Scenario is an ideal framework for simulators designed to study new tactics and equipment because it is easily integrated, open, flexible and reliable. STAGE Scenario is being used in tactical simulations for weapons evaluation, doctrine development, and operational analysis, and is ideally suited for use as a backbone for Distributed Interactive Simulation (DIS). STAGE Scenario can also participate in HLA Federations.

TRAINING

STAGE Scenario can be used in any training system where a realistic simulation of an environment is required, including:

- Air weapons control
- Navigation and weapons trainers for aircrew
- Armored vehicle gunnery
- Command & Control
- Naval combat systems
- Naval part-task simulators
- Cockpit and flight procedures

SIMULATION OF REAL OR SYNTHETIC SYSTEMS

STAGE Scenario has been used as a platform for engineering simulators, simulating real and synthetic systems for prototyping in general, and for avionics development, human factors design, and command and control system design in particular.
ARCHITECTURE

STAGE Scenario Architecture

STAGE Scenario consists of four main components:

- Scenario Manager (SM)
- Simulation Engine (SIM)
- STAGE Viewer
- Development Kit (Interface Development Environment (IDE) and Programmer’s Kit)

SCENARIO MANAGER

During the preparation phase, the STAGE Scenario Manager (SM) is used to build and assemble the components of a scenario. This scenario is then run in real-
time by the simulation (SIM) engine to produce an evolving synthetic environment. The components that comprise the scenario are stored in XML format (tactical database) to allow for the easy import, export and management of data.

At runtime, the SM is used to visualize the evolution of the entities within the synthetic environment. By using multicast communications on the network, STAGE Scenario allows multiple SMs to monitor and control the SIM, allowing several operators to interact in the same simulation exercise. The SM can also run without a GUI to control scenario operations and download the different profiles to the SIM engine. During both preparation and runtime phases, the focal point of the SM user interface is the Tactical Display (TD), which provides a global view of the scenario. The TD consists of a map of the gaming area overlaid with symbology representing the scenario and its entities. The TD provides an easy way to create and tune a scenario, using the mouse to drag and drop platforms and add waypoints. STAGE Scenario Manager is available on the Microsoft Windows 2000 and Windows XP platforms only.

SIMULATION ENGINE

The STAGE Scenario Simulation (SIM) consists of a real-time simulation engine and simulation modules that model the various behaviors of the entities within the synthetic environment. The SIM is responsible for “bringing to life” the synthetic environment by running a scenario. It communicates with the SM to obtain the description of the scenario and its entities, to export data about the evolution of the scenario, and to respond to simulation flow control commands. The SIM is composed of five major components:

- Entities
- Model Managers
- Entity Manager
- Scenario Manager
- Real-Time Controller (RTC)

The SIM provides key descriptive and runtime state information about the scenario and its entities to external simulation processes via multiple ways including nCOM, shared memory, HLA, DIS. This allows STAGE Scenario to operate as the central simulation in a set of cooperating simulation processes. The Model Managers can be run in parallel, enabling STAGE Scenario to benefit from machines with multiple CPUs. A thread is created for each available CPU, and synchronization mechanisms ensure that optimal performance is obtained from the CPUs without introducing simulation artifacts. The SIM engine is available on the Windows XP and Linux Red Hat platforms.

The Logger supplied with STAGE Scenario provides the developer with the means to record and playback simulation data or events. The user can, through
the Logger configuration window, select the data and events to be recorded or played back.

STAGE VIEWER

The STAGE Viewer is a high-end 3-D viewer powered by Diamond Visionics technology that includes a unique plug-in that delivers out-of-the-box integration with DI-Guy human models from Boston Dynamics.

Based on Diamond Visionics technology and a common dataset, the STAGE 3-D viewer, terrain services, and tactical display deliver perfect correlation within the simulation system and eliminate the need for time-consuming format conversions.

STAGE FLIGHTSIM / STAGE HELISIM INTEGRATION

In STAGE Scenario, the ability to link to STAGE Flightsim and STAGE Helisim is built in. The flight dynamics of a fixed-wing entity can be externally controlled by STAGE Flightsim, while the flight dynamics of a rotary-wing entity can be externally controlled by STAGE Helisim. From within STAGE Scenario, the user can choose the Flightsim/Helisim settings, including aircraft/helicopter model and initial conditions, and set Flightsim/Helisim to start automatically. The environment within which Flightsim/Helisim operate is the environment as defined within STAGE Scenario.

VAPS XT INTEGRATION

STAGE Scenario automatically communicates with VAPS XT via the nCOM protocol for developing user panels, interfaces, and other Human Machine Interfaces (HMIs) for simulation.
AI.IMPLANT INTEGRATION

The new integration with AI.implant allows users to choose to install artificial intelligence for selected STAGE Scenario platforms. The AI.implant software makes it easy to accurately model complex crowd & vehicle behavior within simulated environments. With visual behavior authoring and debugging tools, users can create vehicular and human agents for simulation projects that easily scale from a single, intelligent entity to large crowds with autonomous and group behaviors.

DI-GUY™ INTEGRATION

STAGE Scenario integrates out-of-the-box with the DI-Guy software from Boston Dynamics. The STAGE Scenario/DI-Guy plug-in provides the user with a way to display lifelike human characters, and their behavior and movement, in the Scenario 3-D Viewer. The DI-Guy software includes a library of human characters.
that the user can use in the scenario. It also includes some non-human simulations such as vehicles. Through the Mission Editor, the user can use the new STAGE Scenario/AI.implant integration to control DI-Guy characters within the Scenario 3D Viewer.

3-D OPENFLIGHT MODELS

STAGE Scenario includes a selection of eleven highly-detailed OpenFlight-format 3-D models created by Aegis Technologies.

Aircraft
- Boeing F-18 Hornet fighter/attack aircraft
- Boeing 757 commercial airliner
- Lockheed Martin C-130 Hercules transport aircraft
- Mi-24 Hind assault helicopter
- Mikoyan-Gurevich MiG-29 Fulcrum fighter/attack aircraft

Tanks
- General Dynamics M1A2 Abrams main battle tank
- Uralvagonzavod T-72A main battle tank

Ships
- CVN-68 NIMITZ-class aircraft carrier
- DDG-51 ARLEIGH BURKE-class destroyer

Vehicles
- M1128 Stryker Mobile Gun System

DEVELOPMENT KIT

The STAGE Scenario Development Kit (DK) allows users to extend the capabilities of STAGE Scenario beyond those built into the SIM, Scenario Manager (SM) and tactical database. The DK consists of the STAGE Interface Development Environment (IDE) and a set of libraries and header files (the Programmer’s Kit).

The IDE is a powerful graphical data structure editor. As simulation modules are developed, the IDE is used to enter the data structures that support these modules. Users can add new fields to the existing SM profiles, create new profiles, and create new relationships between profiles. Based on these additions, the IDE automatically extends the tactical database, and generates modifications to the SM user interface to allow the entry, bounds checking, data preprocessing, display, storage, and retrieval of the additional data. The
IDE also generates a new scenario download module to take into account the user-added data.

The Programmer’s Kit is used to add functionality to the SIM and the SM, to replace built-in modules, to extend the functionality of the script language via user-written code and to extend the logger. User-written modules can also link the SIM to local or remote simulation processes. The data that is monitored and controlled through the SM during the simulation can be modified. STAGE Scenario Manager is based on the Microsoft® Visual Studio® .NET 2005 GUI using MFC (Microsoft Foundation Classes). It is possible to extend its menus, toolbars and dialog boxes. Also, new graphical objects can be added to the Tactical Display using the STAGE Scenario graphical API. STAGE Scenario also provides the ability to use graphics other than those supplied. Users can display their own graphics (for example, a circle representing the range of an entity’s radar).

The Logger is a flexible and extensible tool that allows end-users to record any type of data, whether it originates from the default STAGE Scenario modules or is computed in the user’s own modules. Besides entity runtime data, the Logger can also record other types of data present in the SIM and SM, to include user events.
USING STAGE SCENARIO

The Scenario Manager (SM) provides access to STAGE Scenario functionality such as building scenarios, editing profiles, starting the simulation, and displaying situational data through the Tactical Display (TD).

The SM is used to build a scenario and control the scenario during its execution. During the design phase, the SM is used to build and assemble the components of a scenario. At runtime, the SM is used to control and monitor the selected scenario in real-time.

The STAGE Viewer may also be used to provide enhanced visibility into the scenario during its preparation or as it unfolds.

GRAPHICAL USER INTERFACE

In the SM, STAGE Scenario provides a Windows-based Visual Studio .NET 2005 graphical user interface (GUI), with all of the standard items familiar to Windows users such as a menu bar, toolbars and dialog boxes. These menu and window options are used to select database tables, enter or retrieve data, and perform other database management functions.
Using this SM GUI, data can be entered in one of five ways, depending on the type of information required:

- Numerical and text input using an object inspector window;
- Graphical input using “Curve Entry” windows for information in the form of a two or three dimensional curve such as “Y is a function of X”, or in some curves, “Y is a function of X for a value Z”
- Mission creation through the use of predefined conditions and actions to define a list of tasks that can then be assigned to multiple entities
- Script creation and/or modification into a text editor with color code and contextual help
- Graphical input, using the Tactical Display (TD) for trajectory and Positional data input.
- Graphics Editor for entering and editing shapes in the TD such as polylines, polygons, and circles.

COORDINATE SYSTEMS

Adding to the flexibility and functionality of STAGE Scenario is the ability to select the coordinate system to be used. To locate the position of points in space, STAGE Scenario supports three different coordinate systems:

- Geodetic (latitude/longitude/altitude)
- Universal Transverse Mercator (UTM)
- Military Grid Reference System (MGRS)

TACTICAL DISPLAY

The Tactical Display (TD) provides a “God’s eye view” of the synthetic environment gaming area overlaid with symbols representing the various entities that make up the scenario. The TD is used extensively in STAGE Scenario during both the scenario preparation and runtime phases to input positions and to provide a two dimensional view of the battlespace.

The TD shows the position of all entities in the scenario with, as its underlay, a map generated from Digital Terrain Elevation Data (DTED) or another format such as DBDB5 or OpenFlight. The TD also makes use of contour lines to show elevations above sea level. Features such as roads, rivers and bridges, imported from DFAD (Digital Feature Analysis Data) or DFD (Digital Feature Data) files, can be shown as an additional underlay.

NavAids can also be shown as complementary symbology and visual information by importing ARINC 424 format files.
The user can attach the following types of map files to a scenario:

- Elevation maps
- Topographical maps
- Surface Feature maps
- Navigational Aid (Navaid) maps

Terrain formats supported:

- 3D Terrain — OpenFlight, DTED, DBDB5
- Raster maps — GeoTIff, CADRG, .bmp, .png, .jpeg, etc.
- Vector data — DFAD, ESRI shapefiles, .dfd

In addition to maps, the user can now use charts in the STAGE Scenario Tactical Display. Support has been added for chart files in the following formats:

- Geo-referenced Tag Image File Format (GeoTIFF)
- Compressed ARC Digitized Raster Graphics (CADRG)
- All data types supported by Global Mapper

During the preparation phase, the TD provides a graphical overview of the simulation that gives users a better understanding of the potential behavior of a scenario. The TD can be displayed as multiple instances showing different gaming areas with different zoom.

The TD provides a variety of controls that enable users to manipulate the display to best suit their current position including zooming and de-cluttering functions, mechanisms to position and move entities graphically, editing capabilities (cut, copy, paste) to facilitate creating large numbers of entities, and vertical view for waypoint placement. A mouse can be used to obtain terrain or position information from the TD. Users can work in different modes: Zoom, Cut & Paste operations, or Range-Bearing and Lat/Lon/Alt position information.
BUILDING SCENARIOS

In the STAGE Scenario Manager it is possible to design, edit or add “Profiles”, which specify the generic characteristics of something in the scenario that will be instantiated multiple times. This allows the user to define the components used in the building of a scenario from building blocks.

SCENARIO EDITOR

When a scenario is opened in the Scenario Editor, a Tactical Display (TD) appears. Users can open multiple TD windows to work with multiple views of a scenario. The initial TD is located in the main viewing area of the SM window. Each additional TD appears in its own window and can be moved, resized and manipulated like any other window. Users can easily create and access their profiles via the Library pane, which contains a list of Platforms and Special Zones.

DATABASE EDITOR

The Database Editor contains the Database Tree and an Object Inspector (shown below) to allow users to specify the configuration of each platform (e.g., its type and subtype, dimensions, and cross-section factors). Users can create sensors, weapons, countermeasures etc. Also, a Curve Editor (bottom right) is used to define the combat effectiveness (i.e., defense against missiles and/or torpedoes) of the platform. Once configured, these platform profiles are the building blocks for creating scenarios. They can be used repeatedly as a template or prototype for the entities used in the scenario.
USING THE SCENARIO MANAGER FOR RUNTIME CONTROL

The Runtime Environment mode of the Scenario Manager (SM) allows users to perform scenario control and monitoring. A scenario may be loaded into the simulation without running it, or it may be set up to load and run immediately. The Runtime environment includes Run/Stop and Freeze/Resume commands, as well as an attrition table that shows the current status of the entities in the scenario.

The Tactical Display (TD) offers multiple features for scenario control at runtime. The Runtime Control window allows users to start the simulation at any point during the scenario. This feature enables the user to jump ahead of the default start time (00:00:00) and begin the simulation at a specified point in the scenario.

During a scenario, users can double-click an entity to display the Platform Hook window. This window displays the name of the platform or weapon and allows users to view and modify certain parameters of the selected entity, such as its
heading and altitude. If the selected entity is a platform with a basic trajectory, the TD will display the trajectory when the entity is double-clicked.
SIMULATION FLOW CONTROLS

The STAGE Scenario simulation flow control mechanism allows the simulation to perform the following tasks:

- Load a scenario
- Start forward in time
- Run the simulation for a named scenario in scaled real-time (includes loading the scenario if not already loaded)
- Stop the simulation
- Freeze the simulation
- Resume the simulation

The simulation scheduler can use either a fixed iteration rate or synchronize itself with the system clock. An external simulation process can also be used to invoke the simulation flow control mechanism via the gateway mechanism.

SNAPSHOT/RESTORE

The Snapshot/Restore feature of STAGE Scenario allows users to save the state of the runtime simulation, fully restore it at a later time, and begin the simulation from the saved state. Features include:

- Full and incremental snapshots
- Automatic snapshots
- User-triggered snapshots
- Ability to Start exercise in an "in combat" situation
- Repeatability of the "initial" conditions

The Snapshot/Restore feature is also fully extendible, and can be adapted to specific user requirements through the creation of additional user modules that make the necessary function calls.

USING MULTIPLE SCENARIO MANAGERS

STAGE Scenario supports multiple SMs in runtime control mode. This enables multiple users to concurrently monitor and control the evolution of a single scenario using multiple SMs on different machines. Running multiple SMs requires a Master SM and one or more Slave SMs. Both Master and Slave SMs can monitor and control the specified scenario, but only the Master SM can start and stop the simulation.
BUILDING BLOCKS

Users build scenarios out of instances of platforms (called entities) and environmental components that are assembled hierarchically. Each of these “building blocks” is constructed separately and can be reused in multiple scenarios.

The STAGE Scenario basic building blocks consist of:

- Gaming Area
- Atmospheric Profile
- Oceanographic Profile
- Platform Profiles including Physical Data, Dynamics and Navigation Data, Acoustic Profiles, Attached Sensors, Attached Weapons, and Combat Effectiveness Data
- Platform Instances including Initial Conditions, mission specifications, Formation Data, and Basic Trajectories

These are just the basic building blocks. Since STAGE Scenario is extensible, other building blocks can be easily added. Every entity and environmental component is defined through a list of parameters. The simulated behavior is then generated in accordance with the specified parameters. The sections that follow describe the STAGE Scenario building blocks.

GAMING AREA

The gaming area defines the geographic location and extent of the area in which the scenario will operate. STAGE Scenario uses the Genesis RT technology from Diamond Visionics to define its gaming area within its map representation, the terrain simulation and the 3-D viewer.

The Genesis RT technology uses a world-wide database to provide terrain generation anywhere in the world. This terrain representation can then be extended with more precise information.

Defining the Gaming Area
STAGE Scenario supports the use of NavAids. In the NavAid editor, it is possible to create NavAid maps. By reading the NavAid data from ARINC 424 data files, these maps can be linked to a scenario. This allows the user to see the position of the NavAids on the TD and examine their parameters. NavAids are particularly useful for defining aircraft maneuvers (e.g., SIDs and STARs).

ATMOSPHERIC PROFILE

The atmospheric profile specifies the characteristics of the environment above the terrain surface. The atmospheric profile consists of locations, which are points in the gaming area where a column of break points defines the characteristics of each layer of the atmosphere.

Each location is specified by a latitude/longitude position. At each location, users can specify the sea level temperature and air pressure. The variation of these parameters as a function of altitude is then modeled in accordance with the standard ICAO model. At each location, users can define a column of 12 break points. Each break point is specified by an altitude and defines the north/south, east/west and up/down wind at that altitude.

The atmospheric profile’s effects on sensors can be specified in two ways:

- Globally, allowing users to specify an overall atmospheric attenuation factor for a given scenario. The attenuation factor acts upon all simulated sensors by linearly degrading their detection capability.
- Locally, allowing users to specify an attenuation factor at each break point. The effect of these local attenuations is obtained by interpolating from three nearby locations surrounding the sensor under consideration.

OCEANOGRAPHIC PROFILE

The oceanographic profile specifies the characteristics of the environment below sea level. The oceanographic profile consists of locations, which are points in the gaming area where a column of break points defines the characteristics of each layer of the ocean, in the same way as for the atmospheric profile.

The environmental conditions modeled in accordance with the specified atmospheric and oceanographic profiles affect the motion and dynamics of the scenario entities. For each entity, the local atmospheric and oceanographic conditions are calculated by interpolating from three surrounding, nearby locations.
PLATFORM PROFILE

A platform profile defines the characteristics and capabilities of a type of platform. It may be helpful to think of a profile as the catalogue description corresponding to a real object that could be found in a tactical environment.

Individual platforms of a given type can then be instantiated within a scenario. Each platform instance is named and associated with a platform profile and initial conditions. For example, a platform profile could be used to describe the characteristics of the F-15 fighter, and a weapon profile used to describe a Sidewinder missile.

The components of a platform profile include:

- Platform type and subtype
- Bitmap representation
- Physical data (Dimensions and Cross-Section)
- Dynamic data
- Acoustic profile
- Attached sensors
- Attached weapons
- Attached radios
- Combat effectiveness
- Articulated part
- Countermeasures
- Damage impact

PLATFORM TYPE AND SUBTYPE

STAGE Scenario platforms are internally identified by a type and subtype.

- Types are used by the simulation models to control certain aspects of the simulated behavior of the platform. For example, specifying a platform type allows for the fact that ships and planes bank in different manners when they turn.
- Subtypes are not used by the simulation models. They are present only for more finely detailed platform identification.

STAGE Scenario supports the following platform types:

- Fixed Wing
- Helicopter
- Surface Ship
- Submarine
PHYSICAL DATA

The platform physical data (or silhouette) determines how a platform is “seen” by other entities for sensor detection, collision detection and weapon scoring. The silhouette consists of two main parts:

- The physical dimensions of the platform. Platforms are modeled physically as volumes, which are defined by a height, length and width.
- The platform dimensions in the supported sensor spectrums, i.e., visual, radar, sonar and infrared. Here the platform dimensions are established relative to other scenario entities. Each platform can be assigned a “cross-section factor” for each sensor spectrum.

Thus, a platform with a larger cross-section factor than another for a given sensor spectrum is considered to be proportionately larger than the other in that spectrum. The assigned cross-section factors are not aspect dependent.

DYNAMICS AND NAVIGATION

Data for the dynamic maximums and responses can be set for a platform. In the standard dynamics model, the dynamic responses of the platform are simulated via a first-order filter to smooth out any sharp moves. The transient coefficients for the response filter are computed at SIM initialization based on the execution frame rate. An alternative, simplified, dynamics model is available that ignores the specified dynamic response profile and responds instantaneously to requested changes in speed and heading. This model is particularly useful when a scenario with a very large number of platforms is needed.

ATTACHED SENSORS

The sensor and detection mechanisms model the combined effect of systems that can detect the content of the synthetic environment in accordance with predefined filtering rules and specified performance envelopes.

The output of the sensor and detection mechanisms is made available to the scripts for detection and engagement processing. It is also available to user modules for user-supplied additional processing. STAGE Scenario supports both active and passive sensors of various types. Active sensor types are radar and sonar. Passive sensor types are visual, infrared, ESM, and passive sonar. For both active and passive sensor types, the emphasis is on simulating the aggregate detection capabilities of the sensors rather than their detailed model capabilities or emitter characteristics.
The modeling approach used to achieve this goal is as follows:

- Divide the types of sensors into spectrums: visual, radar, sonar and infrared.
- Assign “sizes” (cross-section factors) in each spectrum to each entity that can potentially be detected.
- For passive sensors that detect active sensors on-board other entities (i.e., ESM and passive sonar), assign an “emitted power” to the complementary sensor type (e.g., radar for the ESM) and use this as a “size”.
- Model the detection capabilities of the sensor as a “probability of detection vs. range” for a given “size” of entity.
- For a given entity size and range, compute the probability of detection by interpolating among the specified detection functions.
- The global effect of atmospheric or oceanographic conditions can be simulated via the atmospheric and oceanographic attenuation factors, specified at the scenario level, that linearly degrade the performance of all sensors. Local effects of atmospheric or oceanographic conditions can be simulated via the attenuation factors, which can be attached to individual break points. The effect of these local attenuations is obtained by interpolating from three nearby locations surrounding the sensor.
- The minimum and maximum elevations and azimuths of the sensor can be specified. Users can also specify the scan rate of the sensor and the number of detection hits (or misses) needed before a track is considered detected (or dropped).

The sensor simulation models take into account horizon effects that limit the maximum detection range. Also, distinct modes, which differ in the types of entities that can be detected, are supported. The modes are all, air-only, and ground-only (which has the additional parameter of all and moving-only).

ATTACHED WEAPONS

Weapon types are user-defined via weapon profiles. STAGE Scenario supports two types of weapons: vector and point. A vector weapon is simulated as an entity within the synthetic environment, whereas a point weapon is simulated statistically. Weapons can be launched via commands in behavior scripts, user modules or by external simulation. The result of a weapon at tack is calculated via a scoring mechanism. Weapons are attached to platforms at scenario creation time as clusters or batteries of a given weapon profile. Each cluster is given a name, weapon profile, and quantity of weapons in the cluster on the platform to which it is attached.
The following types of vector weapons are supported:

- **Tracking Missile.** The target must be another entity. The missile heads for the target using pure pursuit and a direct trajectory.

- **Cruise Missile.** The target must be another entity, or a lat/long in the case of area weapons. The missile heads for the target using pure pursuit. The missile flight is in three phases:
  - Initial direct trajectory towards the cruising altitude and speed.
  - Maintain cruising altitude and speed.
  - Head for the target using a direct trajectory.

- **Torpedo.** The target must be another entity. The torpedo heads for the target using pure pursuit. The torpedo trajectory is in three phases:
  - Initial direct trajectory towards the cruising depth and speed.
  - Maintain cruising depth and speed.
  - Head for the target using a direct trajectory.

Each vector weapon is characterized by physical data containing width, length, and height, as well as cross-section factors for the four sensor spectrums, i.e., radar, sonar, IR and visual.

Tracking missiles, cruise missiles and torpedoes are also characterized by a maximum range, emitted power and a dynamic profile. Cruise missiles and torpedoes add a cruising speed and cruising altitude/depth to their dynamic characteristics. The dynamics model used is the same as for platforms.

Vector weapons are managed as entities in the synthetic environment and can have scripts or sensors attached to them. The lethality of a vector weapon is expressed as a *probability of kill vs. launch range to the target*.

The vector weapon is flown in accordance with its dynamic profile and its guidance mechanism. If the weapon reaches its maximum range without hitting a target, it self-destructs.

When the weapon reaches its closest point of approach to the target, impact is determined by computing whether the volumes of the weapon and target have intersected.

- If it is determined that no physical impact has taken place, the weapon is marked as having missed its target and continues its flight until it reaches its maximum range and self-destructs.
- If it is determined that there was an impact, the target’s probability of defense function is applied.
- If the defense is successful, the weapon is destroyed with no effect on the target.
• If the defense is not successful, the weapon is destroyed and the
  weapon lethality function is applied.
• If the result is positive, the target suffers cumulative damages up to the
  point of complete destruction; otherwise it suffers no effect.

The point weapons supported in STAGE Scenario are the gun and artillery. A
gun is characterized by a firing rate. The gun’s target must be another entity.
The gun’s lethality is expressed as a probability of kill vs. firing range to the

  target. The lethality function of the gun is applied to each projectile fired by it
during the specified firing duration. Artillery is similar but covers a greater
range than the gun weapon.

Based on the specified firing duration and the gun’s firing rate, the number of
fired projectiles is computed. For each projectile, the lethality function of the
gun is applied. If the result is positive for any projectile, the target suffers
cumulative damages.

COUNTERMEASURES

STAGE Scenario implements two types of countermeasures: physical and
electronic.

Physical countermeasures are modeled as follows:

• Physical entities
  • Launched via the corresponding Platform Hook window, scripts, or
    through the countermeasures model API
  • Detected by sensors (yes/no)
  • Effectiveness probability vs. range
  • Types include acoustic decoys, chaff, flares and smoke.

Electronic countermeasures are modeled as follows:

• Effectiveness envelope
  • Activated/de-activated via the corresponding Platform Hook window,
    scripts, or through the countermeasures model API
  • Effectiveness probability vs. range
  • Hooks available to define jammer effect on sensor, namely:
    • Deception
    • False targets generation
  • Types include IR jammers and radar jammers.
COMBAT EFFECTIVENESS

The combat effectiveness of a platform within STAGE Scenario is defined in terms of its ability to defend itself against incoming vector weapons. These capabilities are specified as a probability of defense vs. launch range of the incoming weapon function. Two functions of this type can be defined: one for missiles, one for torpedoes. As a result of a weapon impact, entities sustain damage, which reduces their combat effectiveness according to the following levels:

1. No damage
2. Loss of sensors
3. Loss of weapons
4. Loss of mobility
5. Wreck
6. Destruction

As a further countermeasure, platforms have an overall cross-section factor (which applies to all sensor spectrums) that linearly affects their detectability by sensors. This can act as camouflage to modulate the platform’s combat effectiveness. The platform’s overall cross-section factor can be set via scripts extensions or user modules.
PLATFORM INSTANCES

When building a scenario, users require instantiated individual platforms of a given type. Each platform instance is named and associated with a platform profile and initial conditions. These initial conditions can include a behavior script, formation data and basic trajectory for navigation.

INITIAL CONDITIONS

The platform initial conditions consist of the following:

- Color identification: valid colors are blue, red and white (neutral).
- Initial position, speed, heading and altitude.

Requests to change the platform current speed, heading or altitude can be made via the scripts. These inputs will not immediately affect the platform state but will instead be inputs to the dynamics model that will “move” the platform state.

MISSION EDITOR

The Mission Editor lets the user build, execute, monitor and control missions. A mission is a pre-defined series of tasks that are executed by an entity. During the course of the mission, some tasks may or may not be executed, depending on whether a triggering event or condition occurs.

Before the Mission Editor, STAGE Scenario provided two ways to create a mission:

- Scripts
- User Modules

With the introduction of the Mission Editor, the user can now use pre-defined rules, doctrines, and event analysis to create elaborate scenarios quickly, often without scripts or user modules, and with little or no programming. Also, the Mission Editor has been designed so that the user can easily change mission parameters and then execute the mission, allowing the user to test the effect of the change on the progress and outcome of the scenario.
The Mission Editor includes:

- A Mission Editor window, where the user can build missions and sub-missions. Sub-missions are missions that occur within the larger mission.
- A Mission Control window that lets the user control the execution of a mission at scenario runtime.
- A Mission History window that lets the user view a list of all generated events and messages sent during the execution of the mission. The list is saved to a log file that the user can view and analyze later.

BEHAVIOR SCRIPTS

Behavior scripts can be attached to individual platforms in a scenario. Each script is expressed in the STAGE Scenario scripting language and consists of a procedural description of the actions to be carried out by the platform based on the status of the tactical environment.

Examples of what you can do with scripts include:
- Change the platform speed, heading and altitude.
- Turn off/on any of the platform systems, i.e., sensors and countermeasures.
- Scan a list of detected entities or incoming weapons and examining selected information about any of the detected platforms or weapons.
- Identify an entity as an opponent, obtaining additional information about it, and if it comes within a specified range, engaging it by firing any of the platform weapons at it.

Scripts are ASCII files that users can edit using any text editor. A script can be analyzed within the SM to determine its correctness. Behavior scripts are created separately and then associated with particular platforms. Scripts can be activated and deactivated dynamically during scenario execution.
can activate or deactivate the script on another platform, as well as deactivate itself.

A powerful feature of the script language is the ability to allow many scripts to share common data via a data-link. This feature is often used in the master/slave configuration.

FORMATION DATA

Users can build formations of platforms when creating scenarios. A formation consists of a master platform and a collection of slave platforms. The slaves positions are specified by their relative range, bearing and altitude with respect to the master. A rank is also assigned to each slave position to ensure the conformity between unit members and slave positions. The slave's relative positions are sorted by their rank within the formation. A formation can be used to coordinate the movements of entities during the evolution of an exercise. Entities moving in formation will move together according to a pattern, e.g., four aircraft in a diamond formation or a truck convoy following a road at specified distances between them.

Formations are useful when dealing with groups of entities such as units; they help to quickly layout entities according to predefined patterns. When the formation is used to represent a unit, this unit will assume the position of the master to which it is assigned. The slaves will follow the master according to the formation pattern. However, the motion of slaves is still governed by their own dynamics model. Consequently, the formation pattern might be altered at runtime by the dynamic parameters or physical dimensions of the slaves; if the formation pattern places two entities closer than their physical dimensions allow, the entities may collide and be destroyed.

A mission model has been developed to allow units to control their member’s behavior by assigning mission scripts to members within the unit. These scripts can be used to assign tasks to members of a tactical unit and thus coordinate their actions. A mission consists of a set of instructions defined using the STAGE Scenario behavior script language. Mission scripts can be assigned to unit members through the parent unit behavior script or mission (if the parent unit is a member of a higher echelon unit), or using the runtime graphical user interface.

BASIC TRAJECTORY

A platform can have an associated trajectory consisting of a set of waypoints to attain. Each waypoint is a set of coordinates in the gaming area to which an Estimated Time of Arrival (ETA) has been attached. The set of waypoints defines the navigation course that the platform will follow at runtime. The navigation model continuously evaluates the platform's current position
relative to the trajectory waypoints and produces requested speed, heading and altitude changes to keep the platform on course.

A set of actions is available for a platform when it reaches the last waypoint of its trajectory. At the last waypoint, a platform may do one or more of the following:

- Stop
- De-activate itself
- Become invisible
- Destroy itself
- Continue on its course
- Loop back to a previous waypoint
- Repeat the pattern defined by a tail subset of its trajectory

The mechanism that STAGE Scenario uses to go from one waypoint to another is known as the Waypoint Maneuver. By default, it is “direct”, meaning that the entity will go straight to the waypoint. STAGE Scenario includes some typical ATC aircraft maneuvers (SID, STAR, Take off, final approach). It is also possible for developers to use the Developers Kit to add more maneuvers.
WHAT ARE SCRIPTS?

In the STAGE Scenario tactical environment, platform behavior can be “scripted”, allowing the platform to react to environmental and tactical conditions by performing navigation, engagement or other actions. These conditions and actions are specified within the platform behavior script file.

A script file is a text file consisting of a list of statements. These statements can be grouped into three broad categories: conditions, actions and controls.

- Condition statements allow the user to specify characteristics of the tactical environment.
- Action statements allow the user to specify desired changes in the platform behavior.
- Control statements bind conditions and actions together by allowing the user to:
  - Trigger actions based on conditions;
  - Select among alternative actions based on conditions;
  - Repeat statements a fixed or variable number of times;
  - Provide temporary storage for the results of conditions and calculations.

The behavior script language is built on a set of predefined constants, functions and objects. Script objects parallel objects within the scenario. Sample predefined objects are entity, opponent and system weapon.

Attached to each object is a set of members and member functions. Members are attributes of the object, which are accessible by name, e.g., a platform speed or the activity status of a radar. Member functions perform some action on the object to which they are attached, e.g., changing a platforms speed or firing a missile. As with the other components of STAGE Scenario, a Development Kit is provided for the Script Language.

It allows users to add new objects and functions to the language. STAGE Scenario uses two categories of scripts:

- Behavior
- Maneuver

The Behavior script controls the navigation, engagement, or other actions of a platform in reaction to environmental and tactical conditions. It is possible to add or replace an entity behavior script while in runtime.
Maneuver scripts are procedures to follow when executing a specific maneuver.

STAGE Scenario supports two types of maneuver scripts:

- SID
- STAR

These scripts specify the conditions and restrictions applied to the SID and the STAR for a fixed-wing or a rotary-wing aircraft. A SID Script describes the procedure for a “Standard Instrument Departure”, while a STAR Script describes a “Standard Terminal Arrival Route”. These aircraft maneuvers are commonplace in civil aviation, yet the SID and STAR scripts are also quite useful in a tactical environment. For example, with a SID script, the user can specify the altitude, speed and performance of an aircraft during the transition from takeoff to arrival at the first waypoint. Alternatively, through a STAR script, one can program the location, altitude range and speed range of a sequence of waypoints.
EXTENDING STAGE SCENARIO USING THE IDE

EXTENDING THE SCENARIO MANAGER

The SM graphical user interface consists of menus and windows that provide options to create, edit and assemble Atmospheric, Oceanographic, Sensor, Weapon, Platform and other profiles to create scenarios. The SM utilities allow users to open multiple Tactical Displays, control the runtime simulation, view runtime statistics, de-clutter the display, and generally monitor the evolution of the synthetic environment.

The available SM data options are consistent with the simulation models provided with STAGE Scenario. Because user applications may require different data to drive different simulation models, the Integrated Development Environment (IDE) was created.

STAGE Scenario IDE

The IDE provides the needed extensibility mechanisms that enable users to adapt the SM to the data that their applications require.
RELATIONSHIP TO SIMULATION EXTENSIBILITY

The STAGE Scenario Development Kit allows users to create new simulation models and incorporate them into the STAGE Scenario Simulation Engine. As is true for STAGE Scenario core models, these new models need to be driven by data entered by the scenario designer. By using the IDE to enter the data structures needed to support their models, users are actually building a custom STAGE Scenario Manager that supports their own simulation models in addition to those of the core STAGE Scenario.

USING THE IDE

The STAGE Scenario Integrated Development Environment (IDE) provides a graphical user interface with which to add the data structures required to support the simulation models incorporated into STAGE Scenario. Using the IDE, users can add new fields to existing Scenario Manager (SM) profiles, add new profiles and add new relationships between profiles.

The IDE automatically generates code, thereby creating a custom version of the SM and Simulation Engine (SIM). All the data fields and data structures added through the IDE are then supported by those custom processes. This includes scenario design (user interface and database storage) and scenario download from the SM to the SIM.

The IDE allows users to display the data structures for each SM profile and all the data-blocks therein. It displays all the necessary information about the SM database schema, and helps users to visualize this schema when adding new elements to the data structures.

When users save their work, the IDE generates new code to be compiled and linked to produce new dynamic libraries. These new dynamic libraries are then loaded by the SM and the SIM, which will then use the new features. Thus it saves users from having to code the user interface or database access themselves.

The IDE also generates the code for sending the user-added data to the simulation and incorporating it in the simulation global data structure, making it accessible from users’ simulation models.
EXTENDING STAGE SCENARIO THROUGH PROGRAMMING

In addition to the IDE, the STAGE Scenario Development Kit (DK) includes libraries, tools and other utilities that allow users to extend the capabilities of STAGE Scenario through programming. The major components of STAGE Scenario—the Scenario Manager, the SIM, the Logger and the Script Language can be customized using these tools, as described below.

EXTENDING THE SIMULATION ENGINE

The STAGE Scenario Simulation Engine (SIM) is used to “bring to life” a synthetic environment created by the SM. The SIM can be extended in a number of ways:

- User modules can be hooked into the SIM.
- User functions can be hooked into the Entity models.
- External processes can access Entity shared memory and send and receive signals from the SIM process.
- Model parameters can be configured to fit the specific needs of an application.

Using the Development Kit (DK) users can link their own simulation code into the SIM. These user modules allow users to customize the behavior of any Entity in a scenario, or add other capabilities that are not supported by the SIM. Customizing the behavior of an Entity can be accomplished by adding simulation models not supported by STAGE Scenario or by replacing the built-in simulation models with user-written models.

It is possible to replace the model processing function of a Model with a user function. The Models include Physical Environment Managers (Atmosphere, Oceanographic, Collision), Entity Simulation Managers (Navigation, Dynamic, Position Keeping, Sensor, Acoustic Script, Weapon), and System Managers (Relative Position Keeping, Scoring, Platform, Export). This capability allows the user to select the Entities that will have their model processing customized.

Entity state, position and velocity, and other DIS-related data are stored in shared memory and so may be accessed directly by external processes.
EXTENDING THE SCRIPT LANGUAGE

Entities in STAGE Scenario can be made to react to conditions in the synthetic environment. This behavior is specified through the STAGE Scenario Script Language. The Script Language can be extended with user written code to add new constants, functions or objects.

EXTENDING THE SM

STAGE Scenario provides runtime monitoring and control of the simulation from the STAGE Scenario RuntimeEnv window. When an entity, platform or weapon is hooked, a hook window is displayed. This window allows the user to monitor and control the hooked entity.

The default hook window comes with a set of predefined STAGE Scenario parameters. However, the STAGE Scenario DK includes a hook window specification mechanism that allows the user to control the layout and content of the Platform Hook window. Users can also add widgets for monitoring and controlling new data that they have added with the STAGE Scenario IDE. This is done by using the tools provided to extend the hooked entity data buffer. The simulation sends an updated copy of this buffer to the SM once every second while an entity is hooked.

In addition, users can extend the content of the Tactical Display (TD). STAGE Scenario includes a comprehensive graphics library, which supports geodetic coordinates for graphic objects. The STAGE Scenario SM provides the necessary hooks for registering user functions for specific events. Users can then use these functions to add graphics in the scene and control the display of the TD.
LICENSING OPTIONS

STAGE Scenario has three licensing levels: Developer, Preparation, and Runtime. The license the user has will determine not only what the user can install, but also the degree to which the user may customize STAGE Scenario and the simulations:

- **Developer**: Build, run, modify, save, and extend.
- **Preparation**: Build, run, modify, and save.
- **Runtime**: Build, run, and modify.
SUMMARY

STAGE Scenario is a proven COTS product that is used successfully as the simulation framework for a wide variety of military and aerospace applications. STAGE Scenario shortens the time required to build complex simulation and training systems by providing the tools and the framework to generate customized applications. Through its openness and flexibility, STAGE Scenario allows users to quickly prototype a simulation environment and then integrate their own high fidelity models. STAGE Scenario is thus an invaluable tool for researchers as well as for systems integrators and manufacturers of tactical simulation systems.