Framework for a Common Instructor Operator System, Part 1: Enabling Scenario Development

Michael E. Stiso
Jerry M. Owens
Jennifer E. Fowlkes
Susan Eitelman
Amanda Hafich
CHI Systems, Inc.
12000 Research Parkway
Suite 120
Orlando, FL 32826
407-277-9288
mstiso / jowens / jfowlkes / seitelman / ahafich@chisystems.com

Melissa M. Walwanis Nelson
NAVAIR Orlando TSD
Orlando, FL
Melissa.Walwanis@Navy.mil

David G. Smith
Jardon and Howard Technologies, Inc.
Orlando, FL
David.G.Smith1@Navy.mil

Keywords:
instructor operator system, NASMP, distributed mission training, scenario development, distributed teamwork, virtual worlds, collaboration, interface design, intelligent agents

ABSTRACT: The Navy Aviation Simulation Master Plan makes clear the need for a Common Instructor Operator System (C-IOS) that can support both distributed collaboration and multiple platforms. Numerous challenges face the creation of a C-IOS, however, because it must not only support instructors during all phases of scenario development and execution, it must also structure the interaction and flow of work between and among instructors and trainees in the various platforms. The present work outlines the initial development of a C-IOS that meets the top-level requirements for a distributed, collaborative system. By borrowing features common to the virtual worlds prevalent in the gaming industry’s products, the design enables a hierarchical and iterative, but also distributed, flow of control and communication throughout the scenario development process. The paper also provides an example of the C-IOS user interface and discusses potential roles for intelligent agents in the development process.

1. Introduction

The Navy Aviation Simulation Master Plan (NASMP) [1] is a major new initiative intended to improve the training and technology base for fleet aviation operators and maintainers. The NASMP will define and implement innovative training solutions to enhance force readiness. Central to the planning is the establishment of Distributed Mission Training (DMT) to offset the erosion of critical skills and readiness during down-time. DMT will allow Navy aircrews to interact and train together in a simulated environment that more closely approximates the demands of real-world Navy aviation combat. However, establishing DMT will necessitate the integration of new technology designed for distributed simulation with existing technology designed for stand-alone exercises. Consequently, many current system components may need reinforcing in order to meet DMT requirements. Instructor operating systems (IOSs), in particular, may benefit from enhancements that extend current capabilities into the realm of distributed training. For example, DMT scenarios will be larger and more complex than predecessor scenarios because of both the number of entities (both real and artificial) and the multitude of information transactions among platforms, entities, and agencies necessary to execute missions. At the same time, however, the added DMT capabilities will
introduce both greater challenges and greater opportunities for instructors, and so the IOS must support the instructor, as well as the DMT process, accordingly.

Greater efficiency in conducting DMT should be possible through the introduction of a Common IOS (C-IOS). With a C-IOS, instructors across major platform nodes on the simulation network would have a common frame of reference in relating to activities in the synthetic battlespace, as well as a common set of capabilities for setting up scenarios and executing necessary control actions.

A major goal for C-IOS design is to provide support for instructors to carry out their training responsibilities in a pedagogically sound manner across all phases of the simulation-based training process. An integrated design framework must consider key C-IOS functional capabilities, such as scenario development, exercise control, and performance measurement, as well as support for brief-debrief functions. Moreover, a successful C-IOS design approach will incorporate distributed training requirements, instructor needs, and recent advances in human interface design techniques [2]. The present work represents the first step in the development of a comprehensive C-IOS design: determining how to provide for and structure distributed collaboration in scenario development. Hence, although the design that will be described does take scenario execution and after-action review into account, the focus of the discussion will be on scenario development and associated tasks.

2. Background Review

The purpose of the background review was to assimilate as much information as possible to ensure that DMT instructor needs, fleet training requirements, and appropriate user interface design principles are properly addressed in the C-IOS. Reference materials included NAVAIR’s integrated IOS functional requirements [3], a report on functional analysis and learning methods [4], materials on the Next Generation Threat System [5], the NASMP Interoperability Standard [1], system requirements documents for computer generated forces [6], [7], and several investigations of C-IOS requirements [8], [9], [10], [11], [12].

Early in the design process, the research team realized the need to have a model underlying scenario development and execution, collaboration, and communication in a multi-platform training environment that was consistent with key aspects of established Navy aviation training practices. The operating model that best fits is the one in use at the Naval Strike and Air Warfare Center (NSAWC), in which carrier air wing squadrons are brought together to prepare for upcoming deployments. Although instructors at NSAWC are not distributed across remote sites, the manner in which they must come together to accomplish different levels of training is representative of the interactions and involvements relevant to C-IOS design.

To gain a better perspective on DMT, members of the C-IOS research team observed an existing military DMT implementation and several training sessions. The observations indicated that effective collaboration and communication among instructors and operators is crucial throughout scenario generation and exercise control. For example, difficulties in one exercise were the result of the varying experience levels of the multi-platform teams (e.g., Strike, C2). Generally, the inexperience of one team member has the potential to impact significantly the training experience of the rest of the team as a whole. The capability for operators to collaborate quickly and effectively in addressing such difficulties in distributed training is critically important to ensure training quality. If operators have effective collaboration tools during the distributed exercise, then they can more readily modify the scenario and communicate issues across platforms on-the-fly to provide better training for all teams.

Collaboration will also be a crucial component of scenario development in the envisioned C-IOS. During the visit described above, it became apparent that the primary focus of the existing implementation was on the conduct of research in distributed training and simulation. In order to achieve consistency between teams to conduct effective research, the facility made extensive use of canned (pre-programmed) scenarios, which did not require significant modification before the scenario run-time. However, the projected use of the C-IOS described in the following sections is in a system that will require frequent changes to canned scenarios and the development of new scenarios, which will require significantly more coordination among the participating sites. Therefore, collaboration within the scenario development component is a key concept for the C-IOS design.

3. C-IOS Design

The NASMP Interoperability Standard [5] notes 12 usability principles to consider when designing and evaluating training systems. The proposed C-IOS design addresses many of the 12 principles, but the most relevant to the present discussion is collaboration: The system must facilitate and enhance the effectiveness of user collaboration, especially for distributed (not co-located) team tasks. However, if teams are to collaborate in a distributed fashion, they must have a medium for doing so. Resolving the issue of collaboration, then, first required the development of a medium that allowed distributed teamwork.
The following sections describe the design, features, and functions of a proposed C-IOS interface. Currently, the interface is in the conceptual stage, and so it has not yet been prototyped, implemented, nor tested. Hence, the subsequent sections describe the beginnings of a framework for a Common-IOS design, rather than a completed product.

### 3.1 Distribution through virtual worlds

The main obstacle to distributed and collaborative scenario development is to determine who can access and thus modify the scenario at any given time. The research team tackled the issue by incorporating into the design the idea of virtual or online worlds. Online worlds are virtual constructs, similar to interactive web-based street maps, that graphically represent a section of geography. The worlds reside in a central location on one or more computer servers. As with the real world, a coordinate system (x, y, z) describes the virtual versions, and a set of rules governs both the physics of the world and interactions with it.

Via the internet, any authorized individual can access an online world from almost anywhere in the real world. Access involves connecting to the server(s) containing the world and, if necessary, logging in to establish identity. Any number of individuals (within bandwidth limitations) can access the same world at the same time. Once identity and authorization are established, all individuals can view the world and interact with it simultaneously. Furthermore, users can view whatever area of the world map that they choose, even if others are currently viewing the same area.

Users can interact with the world by adding entities (e.g., trees, houses, troops), by modifying or removing objects already within it, and perhaps by modifying the terrain and weather. The server monitors user interactions with the world and updates accordingly the displays of other users accessing the world. As a result, users viewing parts of the virtual world with which other users happen to be interacting can observe those interactions in real time; in other words, they can see objects being added, changed, or removed as those modifications occur. The changes also persist, so that users viewing parts of the world that were modified previously by other users will see those modifications.

Virtual worlds are not a new idea. The gaming industry has employed the concept in its products for several years now; Origin’s *Ultima Online* (www.uo.com) is a good example. Borrowing the concept for the C-IOS design helps solve the challenges of distributed scenario design by allowing instructors in different locations to contribute either synchronously or asynchronously to the scenario. However, the problem of collaboration still exists. In other words, with any number of instructors from all over the country being able to access and modify the scenario at any given time, the development process could easily devolve into chaos. Instructors would be able to overwrite, purposely or not, each other’s modifications, or to make changes that interfere with the plans of another instructors. And who would have the final say in the design?

### 3.2 Collaboration through hierarchy

The research team resolved the collaboration issue by employing another feature common to virtual worlds: administration and permissions. Virtual worlds often have administrators who monitor the goings-on of the world and who can set permissions for other users. A user’s permission level determines to what extent that user can modify the given world. For example, a low permission level might prevent a user from adding certain Computer Generated Forces (CGFs), or from modifying the parameters of CGFs added by another instructor.

Combining the idea of permissions with the NSAWC training process resulted in the creation of a three-level instructor hierarchy: Lead, Element, and Platform. The Lead Instructor (LI) is responsible for training at the integrated training level (training that combines two or more elements) and is in control of the scenario. The Element Instructors (EIs) report to the LI and focus on element-level training requirements, such as those required to train the air warfare element. The Platform Instructors (PI) report to the EIs and are responsible for individual platforms within each element. Importantly, because one of the design goals was for the C-IOS to be flexible enough for teams of any size to use, the different instructor levels are merely placeholders that can be filled by the same instructor or by a number of additional ones (depending on how much responsibility for scenario development that the LI wants an instructor to shoulder).

An instructor’s level determines his or her permissions, which regulate the ability to modify a scenario. The permissions are customizable, of course, but in general, the EIs selected to participate will have control over many (but not all) of the settings and entities in the scenario, and they will likely be able to add a number of entities; however, they cannot change any of the LI’s settings without the LI’s consent. PI’s will have relatively minimal control over the scenario and will be unable to tamper with changes made by either the EIs or the LI. The LI, in contrast, has total control of the scenario. Hence, by putting one person in charge of the overall scenario, dividing up the work into lower-level chunks, and then “sub-contracting” the responsibility for those chunks to
others, the hierarchy resolves the chaos that could result from a distributed team of instructors simultaneously trying to customize a scenario to their liking.

3.3 Flow of control in scenario development

Although the C-IOS design arranges the instructors into a hierarchy, the flow of control and communication between them is hierarchical, iterative, and simultaneous. It is hierarchical, as described earlier, in that the LI can focus on the overall scenario by dividing it into elements and assigning those to element instructors (EIs). The EIs, in turn, can focus on the higher-level issues associated with their particular elements by assigning the details concerning each of the platforms involved to platform instructors (PIs). Thus, each instructor can focus on his or her primary responsibility in scenario design either by sub-tasking parts of that responsibility to other instructors or by leaving the higher-level duties to others.

However, the iterative nature of the development process in C-IOS ensures that each instructor has the final say regarding scenario modifications made by the instructors reporting to him or her. The process is iterative in the

![Figure 3.1. Description of scenario development phases and collaborative interactions.](image-url)

focus on the overall scenario by dividing it into elements and assigning those to element instructors (EIs). The EIs, in turn, can focus on the higher-level issues associated with their particular elements by assigning the details sense that work starts with the lead instructor (LI), passes to the element instructors (EIs), and then to the platform instructors (PIs), after which it reverses and travels back to the EIs and finally to the LI. (See Figure 3.1.) As the
flow travels from the PIIs back up the hierarchy to the LI, instructors can inspect, approve, or disapprove changes made by the instructors under them.

Following the framework of NSAWC training, the research team divided the process of scenario development into five phases: exercise preplanning, scenario development, assessment of trainee planning and briefing performance, instructor discussion of trainee performance, and final scenario modifications. (See Figure 3.1.) Each of the five phases consists of a number of tasks specific to that phase, such as editing mission parameters (e.g., training objectives) and editing the scenario (e.g., populating it). The flow of work on a given task starts with the LI, proceeds to the EIIs and PIIIs, and then iterates back up the hierarchy, and the five development phases consist of three major iterations of that sort: the first for developing and refining the scenario’s fundamental goals and Training Objectives (TOs), the second for populating and refining the scenario, and the third for developing and accommodating trainee mission plans. The exercise preplanning and scenario development phases form the first two iterations, respectively. In essence, training needs in the first iteration drive how the initial scenario (e.g., threats, targets) is “laid down” at the integrated level; then, in the second iteration, elements and units provide scenario population inputs in a structured fashion. The third iteration consists of the final three phases. First, trainees examine the completed scenario to plan their mission; afterward, the instructors assess trainees performance and modify the scenario to address trainee weaknesses.

Finally, the development process is simultaneous in that any authorized instructor can log in at any time both to see what changes have been made and to make additional changes. (See dotted arrows in Figure 3.1. The exception is when the LI locks the scenario to additional changes in between the three cycles.) Even during scenario population, different instructors at different levels and in different locations can log into C-IOS at the same time, viewing each other’s modifications to the map in real time. Intelligent agents could facilitate the process by highlighting and suggesting alternatives to potential conflicts between changes made either by the same or by different instructors.

### 3.4 User interface look and feel

The design philosophy behind the C-IOS was to have the interface adapt to a specific user’s needs by recognizing the role of that user in the scenario development process. The interface also helps users to maintain a high-level view of a scenario while accessing lower-level details. It does so by making all information and capabilities easily accessible through one screen. Specifically, C-IOS divides the UI into customizable “chunks” or modules that provide the most relevant information and tools for the given phase of development, but which also provide clear “hooks” to other information and tools that may be needed. (See Figure 3.2.) Each module is a window containing functions that serve a related purpose, and those windows have tab-like features that let the user access the different functions. For example, depending on the development phase, the Scenario Tools window can contain either training objectives and mission documents, or a list of entities with which to populate the scenario. Similarly, because the C-IOS must allow for both synchronous and asynchronous communication, the Comms & Chat window can contain either asynchronous tools such as email or synchronous ones such as chat tools.

The main point is that the windows are all contained on the same screen, at the center of which is the Map window. Depending on the stage of scenario development (see Figure 3.1), a map may not always be present in the window; however, the window will always be present,
and as such it serves as an anchor for the interface. Essentially, the user can then access all relevant information and C-IOS capabilities needed for a given task without losing track either of other information or of the map window. Depending on the user’s needs, he or she can customize the size of the different windows to display more details or to create more room for text. For example, the window on the left of the user interface (UI) can expand to show such information as TO lists, and the timeline window at the bottom of the screen can expand to show timeline details as well as timelines for individual platforms and elements. However, the Map window will continue to be present both to serve as an anchor and so that the user can reference it for other tasks, such as timeline inspections.

The Timeline and Area of Operation windows are also intended to facilitate situation awareness without getting the user lost in the interface, during both the development and the running of the scenario. In the Timeline window, the user can zoom out to view the entire timeline or zoom in to view a section of it. Viewing the entire timeline provides an overview of the scenario and also enables the instructor to see clusters of planned events, providing an overall grasp of the scenario; zooming in provides details regarding the events in a given cluster. Furthermore, as described earlier, the instructor can expand the window to separate the main timeline into individual timelines for each element or platform. Enabling the user to view timeline details without having to switch to a different screen serves to maintain continuity of actions and thought, reduces information gathering requirements, and facilitates at-a-glance processing of information. Seeing the map window and the timeline together also promotes a better understanding of the situation and reduces demand on memory, because users are not required to hold information in memory from one screen while viewing another.

Along similar lines, the Area of Operation window promotes an understanding of the overall situation by providing information concerning the relative location of all CGFs in the scenario, not just those present in the map window. In conjunction with the timeline, the Area of Operation window enables instructors to see at a glance the current locations of all units, whether and when those units will interact, what is in store for those units down the road, and where those units have already been. In sum, the C-IOS interface is designed to help users maintain a high-level grasp of the scenario’s geographical...
and temporal layout while still accessing the smaller details that comprise it.

Figure 3.3 shows a mockup of the scenario population stage of the proposed interface. The Lead Instructor has already selected the area of operation that will form the foundation of the scenario. The entire area of operation, including the relative positions of CGFs, the range of surface-to-air missile and other sites, and the section currently being viewed, can be seen in the Area of Operation window to the lower right of the screen. The LI has selected the timeline view in the bottom window, which will display the events planned during the scenario. If he chooses, he could expand the window upward to view individual timelines for each element and platform.

In the window on the left of the screen, the LI has clicked on the menu containing the CGFs and other scenario objects. To populate the scenario, he clicks on one of those objects and drags it onto the desired section of the map; in the example, he has placed an aircraft. When he releases the mouse button, a window pops up asking the LI to set certain parameters for the aircraft—for example, whether the unit is blue force or red force, the type of aircraft, how many aircraft he wants in that location, and the formation in which they should be arranged. The window will already have default or common choices selected so the LI can just click OK if he chooses; also, if he would rather populate the map first and set the parameters later, he can just click the cancel button.

Double clicking on a CGF object will bring up the entity configuration window (ECW; see Figure 3.4), which is a tabbed dialog that allows the user to set other CGF parameters—e.g., level of automation, armament, and Rules of Engagement. The ECW is relatively dynamic and easily customizable in terms of information content, and so it provides hooks for incorporating the capabilities.
of CGFs from different platforms within a Common-IOS. In a nutshell, users can drag and drop CGF placeholders onto the scenario map. The placeholders point to databases containing platform-specific values for a set of parameters. Clicking on the placeholder pulls the relevant subset of parameters and associated values from the database and populates a dialog with them. Users can thus set both platform-specific attributes and attributes common to a variety of platforms. Figure 3.4 shows potential parameters (e.g., weapons, fuel, automation) for a hypothetical platform. The Weapons tab in the figure, for example, displays an aircraft with weapons slots that the user can fill with arms available to the hypothetical prototype.

The scenario population features of C-IOS build off of the Joint Mission Planning System Software (JMPS) [13] and the Portable Flight Planning Software (PFPS) [14]. For example, borrowing the drag-and-drop capability of JMPS both augments the usability of C-IOS and, with the pop-up entity configuration windows, allows the user to add and edit virtual, live, and constructive entities to the scenario quickly and easily. Borrowing from PFPS led to the inclusion of, among other elements, airfield, waypoint, and aimpoint overlay symbols. In addition, several interface design components, as well as the structure and organization of the C-IOS, were developed out of instructor utilization handbooks for various aircraft platforms (e.g., [15]).

3.5 Intelligent agents

C-IOS provides several opportunities for agents to facilitate the scenario development process. Most of the opportunities involve monitoring user actions in one form or another. For example, when instructors choose training objectives, edit mission documents, and populate scenarios, they do so as part of a team of other instructors who are also modifying those materials. As a result, one instructor’s changes may overlap with or impact those of another instructor, potentially creating an unworkable or illogical scenario. Agents in C-IOS would monitor the modifications made by each instructor to determine whether those changes have any impact on each other, on the timeline, or on the lead instructor’s overall scenario. If the agent determines that a given modification is potentially troublesome, it would highlight the problem for the instructor who requested the change. It would also provide an explanation of the problem and offer suggestions to remedy it. Furthermore, instructors can mark any of their modifications as high or low priority, to indicate how important that change is to the given instructor’s training goals. Thus, if given authority by the lead instructor, the agent could settle any disputes itself by determining both which instructor (e.g., element vs. platform instructor) and which of the conflicting changes has the higher priority.

Another role for agents is available during the initial scenario population phase of scenario development when the lead instructor begins to populate the area of operation with CGFs and other elements. Particularly in situations in which a mission plan had already been developed for the trainees, an agent would help the lead instructor to populate the scenario fully. For one instructor to populate an entire scenario alone, he or she would need expertise in all aspects of every element and platform involved. Combined with the high cognitive demand caused by trying to specify all the details of an entire scenario, such knowledge requirements make the task a daunting one. An agent would help here either by automatically populating or by offering suggestions concerning certain aspects of the scenario, based on the TOs selected by the LI. The agent also would provide guidance concerning arrangement, planning, and timing of the elements and platforms involved, again based on the TOs and mission documents.

4. Summary

The present work sets forth new conceptualizations for underpinning a comprehensive and integrated C-IOS development environment. The described formulations are not intended as a final prescription, especially because user feedback could not be obtained during the brief time available for the initial design and development activities. However, the next spiral of C-IOS development will incorporate user inputs to achieve the most desirable features for scenario development and CGF integration while also expanding the capabilities of the C-IOS, such as through weather servers, weapon servers, and scenario execution features.

The present C-IOS design framework fosters and anticipates the incorporation of essential training system components such as performance measurement, diagnosis and feedback in a collective task-intensive environment. The data and information collected and analyzed at every step of the way will support effective debriefing and After Action Review functions. Results will feed into learning management system capabilities that can track and archive performance results for future task list construction and scenario development purposes. Continuity of efforts toward such ends will greatly facilitate the C-IOS that is finally implemented to serve critical Navy aviation training needs.

5. References

Author Biographies

MICHAEL E. STISO is a Research Psychologist at CHI Systems, Inc. He earned a Ph.D. in cognitive psychology from Texas A&M University, and he has worked on a range of human-factors and usability issues for both the military and industry. His research interests include UI design and usability, performance measurement, and game-based training.

JERRY M. OWENS has extensive experience in the analysis of Navy aviation training system requirements, with particular focus on distributed training. He served as an aviation experimental psychologist in the U.S. Navy and worked on human engineering and training development teams at the Johnson and Kennedy Space Centers.

JENNIFER E. FOWLKES is a Research Psychologist at CHI Systems, Inc. She holds a Ph.D. in Experimental Psychology from the University of Georgia. Dr. Fowlkes has over fifteen years of experience in areas of human factors and training, which includes team training and performance, training effectiveness evaluations, simulator sickness research, and performance test battery development. Most recently, her research experience has focused on assessment of team performance in distributed training environments.

SUSAN EITELMAN Susan Eitelman is currently a Cognitive Engineer for CHI Systems, Inc. in Orlando. She conducts cognitive task analyses, contributes to the design of training and performance support tools, and develops intelligent agents. Her current interests are in cognitive modeling, human learning and problem solving, and human machine interaction. Ms. Eitelman received her B.S. in Mechanical Engineering and Psychology from Carnegie Mellon University and is pursuing a Master's degree in Industrial Engineering at the University of Central Florida.

AMANDA HAFICH is a Human-System Engineer at CHI Systems, Inc. Her experience includes user interface and systems design, conducting cognitive task analyses, and generation of functional requirements. She is currently pursuing her Master's degree in Modeling and Simulation at the University of Central Florida, with a concentration in Human Systems.

MELISSA M. WALWANIS NELSON is a Research Psychologist at the NAVAIR Orlando Training Systems Division. She has a M.S. degree in
Industrial/Organizational Psychology from the University of Central Florida. Her research interests include distributed mission training, leadership development, and simulator instructional tools.

**DAVID G. SMITH** is a Research Scientist at JHT, Inc. He has a M.S. degree in Industrial/Organizational Psychology from the University of Central Florida. His research interests include distributed mission training, team training, and performance measurement.