ADST-II-CDRL-ONESAF-9800101
MAY 8, 1998

ADVANCED DISTRIBUTED SIMULATION TECHNOLOGY II (ADST II)

ONESAF TESTBED BASELINE ASSESSMENT (DO #0069)

CDRL AB02

FINAL REPORT

FOR: NAWCTSD/STRICOM
12350 Research Parkway
Orlando, FL 32826-3224
N61339-96-D-0002
DI-MISC-80711

BY: Lockheed Martin Corporation
Lockheed Martin Information Systems
ADST II
P.O. Box 780217
Orlando, FL 32878-0217

Approved for public release; distribution is unlimited

UNCLASSIFIED

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	8 May 1998	final

4. TITLE AND SUBTITLE
Advanced Distribution Simulation Technology II (ADST-II)
Onesaf Testbed Baseline Assessment Final Report

5. FUNDING NUMBERS
N61339-96-D-0002

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Lockheed Martin Information Systems
ADST-II
P.O. Box 780217
Orlando Fl 32878-0217

8. PERFORMING ORGANIZATION REPORT NUMBER
ADST-II-CDRL-ONESAF-9800101

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

NAWCTSD/STRICOM
12350 Research Parkway
Orlando, FL 32328-3224

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for Public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

During the summer of 1997, the US Army Simulation Training and Instrumentation Command (STRICOM) supported an analysis effort to study the potential architectural implications of the One Semi-Automated Forces (OneSAF) operational requirements. The OneSAF Operational Requirements Document (ORD), developed by the government's OneSAF Integrated Concept Team, defines OneSAF as a "composable, next generation CGF that can represent a full range of operations, systems, and control processes from individual combatant and platform to battalion level, with a variable level of fidelity that supports all models and simulation (M&S) domains." [1, page 1] The analysis effort [2] concluded that an evolution approach using Modular Semi-Automated Forces (ModSAF) or the Close Combat Tactical Trainer Semi-Automated Forces (CCTT SAF) would not satisfy the variety of OneSAF uses nor would it be a maintainable or extensible solution. The analysis further concluded that a new OneSAF architecture should be developed that maximized the reuse of legacy SAF development experience, concepts, design, and code at every opportunity. However, to reduce the technical risks of the architecture development, research and a thorough domain analysis should be conducted prior to this development.

14. SUBJECT TERMS	15. NUMBER OF PAGES
ADST-II, ModSAF, simulation	177
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLAS			

Document Control Information

Revision	Revision History	Date
	Initial Release	04/24/98

List of Figures

List of Tables

EXECUTIVE SUMMARY

Computer Generated Forces (CGFs) are computer systems that emulate the battlefield entities and units whose tactical behaviors and decisions are either made, in part, by human operators (Semi-Automated Forces) or automated decision algorithms (Automated Forces). A number of products have been developed to support Army applications in three modeling and simulation domains: Training, Exercise, Military Operations (TEMO); Advanced Concepts and Requirements (ACR); and Research, Development and Acquisition (RDA). In January 1996, the Deputy Undersecretary of the Army (Operations Research) tasked the Army Materiel Systems Analysis Activity (AMSAA) to conduct an assessment of the various CGF systems for their application to the three domains. The primary object of this study was to assist the Army in developing an optimum investment strategy for evolving its CGF products. The assessment's results recommended that the Army develop a common CGF architecture that would address the linkage and interoperability of models and simulations.

During the summer of 1997, the US Army Simulation Training and Instrumentation Command (STRICOM) supported an analysis effort to study the potential architectural implications of the One Semi-Automated Forces (OneSAF) operational requirements. The OneSAF Operational Requirements Document (ORD), developed by the government's OneSAF Integrated Concept Team, defines OneSAF as a "composable, next generation CGF that can represent a full range of operations, systems, and control processes from individual combatant and platform to battalion level, with a variable level of fidelity that supports all models and simulation (M&S) domains." [1, page 1] The analysis effort [2] concluded that an evolution approach using Modular Semi-Automated Forces (ModSAF) or the Close Combat Tactical Trainer Semi-Automated Forces (CCTT SAF) would not satisfy the variety of OneSAF uses nor would it be a maintainable or extensible solution. The analysis further concluded that a new OneSAF architecture should be developed that maximized the reuse of legacy SAF development experience, concepts, design, and code at every opportunity. However, to reduce the technical risks of the architecture development, research and a thorough domain analysis should be conducted prior to this development.

Due to programmatic constraints the Army accepted the technical recommendation but was not able to execute the recommendation in the near term. The Army Model and Simulation Executive Council (AMSEC) directed a OneSAF development plan that focuses the near term evolution of an existing SAF system, either ModSAF or CCTT SAF. This near term OneSAF Testbed will be used to support research and development for the next generation OneSAF architecture experiments, will be extended toward providing a BBS replacement capability and will provide the training capacity currently implemented by CCTT SAF.

Starting March 1998, the US Army Simulation Training and Instrumentation Command (STRICOM) has supported a five week analysis effort to develop a recommendation of the SAF system to be used as the baseline for integrating ModSAF and CCTT SAF capabilities into a OneSAF Testbed baseline. Given the prioritization to provide the follow capabilities, this analysis recommends an integration effort starting from the ModSAF baseline:

- A similar capacity for "Plug N Play" (i.e., extensibility) as provided by CCTT SAF and ModSAF,
- A capacity to be extended toward providing a battalion level behavior Application Program Interface (API),
- A capacity to support research and development experimentation associated with the OneSAF object architecture using current SAF capabilities,
- A breadth of modeling capability as provided by CCTT SAF and ModSAF,
- A training capability as provided by CCTT SAF and ModSAF.

The recommended approach will build the OneSAF Testbed using the ModSAF architecture and models. The integration approach will include selectively re-engineering CCTT SAF models and architectural characteristics into this C language baseline and extend the ModSAF architecture through software engineering enhancements. The detailed technical characterizations for ModSAF and CCTT SAF and the detailed assessment artifacts that led to this recommendation are provided in this report.

1 INTRODUCTION

Starting March 1998, the US Army Simulation Training and Instrumentation Command (STRICOM) has supported a five week analysis effort to develop a recommendation of the SAF system to be used as the baseline for integrating Modular Semi-Automated Forces (ModSAF) and Close Combat Tactical Trainer Semi-Automated Forces (CCTT SAF) capabilities into a OneSAF Testbed baseline. The OneSAF Testbed will be used for the follow activities:

- To support research and development for the next generation OneSAF architecture experiments,
- To be extended toward providing a BBS replacement capability through a Battalion level behavior API, and
- To provide the training capacity currently implemented by CCTT SAF.

The objectives of this analysis were to provide a recommendation of the SAF system to be used as a testbed baseline from which to evolve toward OneSAF and to identify the technical characteristics of both ModSAF and CCTT SAF.

This document reports the results of this OneSAF Testbed analysis effort. The organization of this report includes a description of the OneSAF problem and the interim solution known as the OneSAF Testbed in Section 2. Section 3 describes the technical characteristics of ModSAF and CCTT SAF and the results of the assessment. The complete list of behaviors and entities implemented within each system is contained in Appendices A and B, respectively. The detailed migration assessments are contained in Appendix C.

2 Problem Statement

The Army has many uses for products that emulate entities and units within a synthetic tactical environment. Several systems have evolved over the past 10-15 years to support these uses. In recent years, the Army and DoD have tried to initiate a plan for encapsulating these multiple efforts into a single investment that supports these many similar requirements. This initiative has been called SAF merge and SAF Integration, among other names, but is now known as OneSAF.

2.1 Industry Background

Computer Generated Forces (CGFs) are computer systems that emulate the battlefield entities and units whose tactical behaviors and decisions are either made, in part, by human operators (Semi-Automated Forces) or automated decision algorithms (Automated Forces). A number of products have been developed to support Army applications in three modeling and simulation domains: Training, Exercise, Military Operations (TEMO); Advanced Concepts and Requirements (ACR); and Research, Development and Acquisition (RDA). In January 1996, the Deputy Undersecretary of the Army (Operations Research) tasked the Army Materiel Systems Analysis Activity (AMSAA) to conduct an assessment of the various CGF

systems for their application to the three domains. The systems considered in this analysis were the Battlefield Environment Weapon System Simulation (BEWSS), the Close Combat Tactical Trainer (CCTT) Semi-Automated Forces (SAF), the Interactive Tactical Environment Management System (ITEMS), the JANUS system, the Joint Conflict Model (JCM), the Joint Tactical Simulation (JTS), and the Modular Semi-Automated Forces (ModSAF).

The primary object of this study was to assist the Army in developing an optimum investment strategy for evolving its CGF products. The assessment's results included the recommendation for the Army to continue supporting multiple CGFs over the following five years to meet all domain requirements. Additionally, it recommended that the Army develop a common CGF architecture that would address the linkage and interoperability of models and simulations, and support Defense Advanced Research Projects Agency (DARPA) and Defense Modeling and Simulation Office (DMSO) initiatives focused on developing a common architecture.

In May 1997, the Deputy Commanding General, Training and Doctrine Command, approved the Mission Need Statement for the OneSAF model. The Integrated Concept Team (ICT), lead by AMSAA, developed the One Semi-Automated Forces (OneSAF) Operational Requirements Document (ORD) during the summer of 1997. [1]

2.2 OneSAF Development Strategy

During the summer of 1997, the US Army Simulation Training and Instrumentation Command (STRICOM) conducted an analysis effort to study the potential architectural implications of the OneSAF operational requirements. The OneSAF ORD, developed by the government's OneSAF Integrated Concept Team, defines OneSAF as a "composable, next generation CGF that can represent a full range of operations, systems, and control processes from individual combatant and platform to battalion level, with a variable level of fidelity that supports all models and simulation (M&S) domains." [1, page 1]

This OneSAF analysis effort analyzed the operational requirements of a OneSAF development that will support the TEMO; ACR; and RDA domains' uses for a Semi-Automated Forces system. The objectives of this analysis were as follows:

- Derive the OneSAF architecture characteristics from the government provided ORD,
- Baseline CCTT SAF and ModSAF architectural components, models, and interfaces for potential evolution into a OneSAF product, and
- Conduct a OneSAF architecture analysis and risk assessment.

The analysis effort [2] concluded that an evolution approach using ModSAF or CCTT SAF would not satisfy the variety of OneSAF uses nor would it be a maintainable or extensible solution. The analysis further concluded that a new OneSAF architecture should be developed that maximized the reuse of legacy SAF development experience, concepts, design, and code at every opportunity. However, to reduce the technical risks of the

architecture development, research and a thorough domain analysis should be conducted prior to this development.

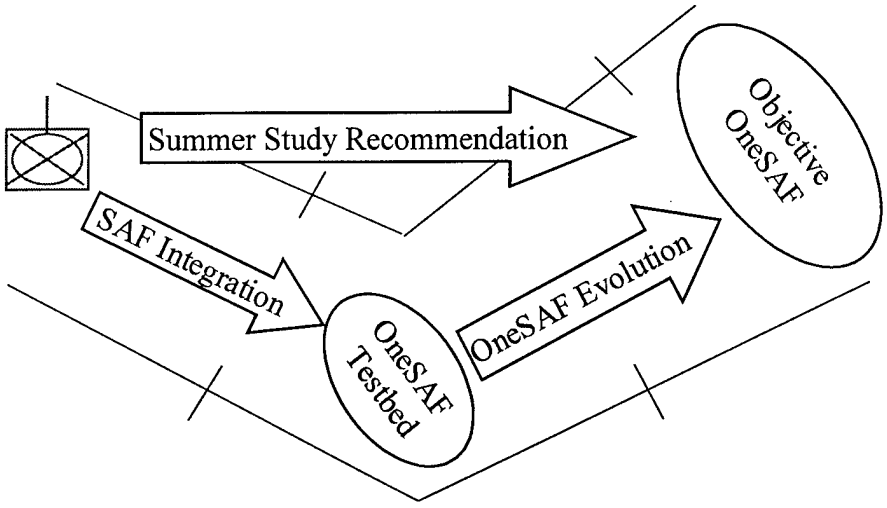

Figure 1: OneSAF Avenues of Approach

Due to programmatic constraints the Army accepted the technical recommendation but was not able to execute the recommendation in the near term. As illustrated in Figure 1, the Army Model and Simulation Executive Council (AMSEC) directed a OneSAF development plan that focuses on a near term evolution of an existing SAF system, either ModSAF or CCTT SAF. This interim OneSAF Testbed will satisfy the following requirements:

- Must replace ModSAF with no decrease in existing functionality and capability,
- Must replace CCTT SAF internally in the CCTT system with no decrease in existing functionality and capability,
- Must be capable of providing battalion level behavior APIs which can be used by the three Domains to further develop battalion and above level behaviors,
- Must be hardware platform independent with the priority on existing CCTT SAF and ModSAF platforms,
- Must provide internal DIS/HLA interface (with respect to SAF Systems only),
- Ease migration to a new, next-generation OneSAF Architecture, and
- Must support the ability to add new equipment, units, behaviors, physical models, editors, and synthetic environment representations.

3 Analysis Activity

The OneSAF Testbed baseline analysis provides a technical characterization of both ModSAF and CCTT SAF capabilities and assesses migration approaches from these SAF baselines into the OneSAF Testbed baseline. This analysis included a recommendation as to which system or combinations of systems would provide the most effective baseline from which to develop the OneSAF Testbed baseline. The analysis considered the migration approach while specifically addressing the criteria in Table 1.

Cost	Benchmark Performance
Program Risk	Adaptability to new Technologies and Architectures
Technical Feasibility	Ease of coordination and integration of multiple developers
Existing Program Impacts	User control (add, modify, tailor)
Infrastructure Issues	Other areas as required
User Interfaces	Development timeline of no more than 24 months
Advantages of each existing baseline	Interoperability with existing simulations
Disadvantages of each existing baseline	

Table 1: Migration Criteria

The assessment considered the ModSAF 4.0 beta and CCTT SAF 5.1.2 baselines. The analysis team assumed that the CCTT system integration effort would be conducted after the 24-month OneSAF Testbed integration. The team also assumed that CCTT SAF and ModSAF extensions would continue during the 24-month period and that any integration of those extended capabilities into the OneSAF Testbed could be conducted after its initial development. The hardware platforms to be supported by the OneSAF Testbed are limited to SGI, SUN, PC (Linux), DEC Alpha, IBM (AIX), and Motorola (AIX). The team concluded early in the assessment that the OneSAF Testbed must be implemented only in C language (compilable in C++) in order to satisfy the portability and accessibility capability already available by ModSAF. The C++ compiler requirement will allow any further extensions or experimentation using the Testbed to be developed using Object-Oriented Programming.

The priority for the OneSAF Testbed characteristics were defined as follows:

1. A similar capacity for "Plug N Play" (i.e., extensibility) as provided by CCTT SAF and ModSAF in their identified baselines,
2. A capacity to be extended toward providing a battalion level behavior API,
3. A capacity to support research and development experimentation associated with the OneSAF object architecture using current SAF capabilities,
4. A breadth of modeling capability as provided by CCTT SAF and ModSAF,
5. A training capability as provided by CCTT SAF and ModSAF.

This section describes the analysis approach for the OneSAF Testbed baseline assessment, describes the technical characterization of ModSAF and CCTT SAF, and summarizes the migration approach assessments.

3.1 Analysis Approach

To achieve the OneSAF Testbed analysis objectives, a set of tasks were implemented by the SAIC – Lockheed Martin team (listed in Table 1). The implemented tasks are described as follows:

- Identify the distinguishing characteristics for both ModSAF and CCTT SAF,
- Assess the capabilities by discussing and recording those CCTT SAF and ModSAF capabilities and implementations that must be maintained in the OneSAF Testbed,
- Conduct a migration path assessment that identifies the alternative migration paths for each critical component of the OneSAF Testbed and evaluate the migration approach with respect to the assessment criteria.

Person	Role
Wesley Braudaway, Ph.D.	SAIC Manager / Project Director / CGF Architect
Jeff Bergenthal	LM Manager / ModSAF Engineer
Anthony Courtemanche	SAIC ModSAF Assessment Lead
Se Hung Kwak, Ph.D.	LM ModSAF Engineer
Michael Longtin	LM ModSAF Engineer
Dan Coffin	LM ModSAF Engineer
Mark Torpey	LM ModSAF Engineer
Wendy Richardson	LM ModSAF Engineer
Bob Burch	SAIC CCTT SAF Assessment Lead
Jon Watkins	SAIC CCTT SAF Engineer
David Blanchard	SAIC CCTT SAF Engineer
Gene McCulley	SAIC CCTT SAF Engineer
Peggy Hughley	SAIC CCTT SAF Engineer

Table 2: OneSAF Testbed Analysis Team

3.1.1 Technical Characteristics Process

As illustrated in Figure 2, the analysis team identified the distinguishing characteristics and capabilities from both systems to be studied. The list of characteristics were normalized between the two systems and served to focus the technical characterization and migration assessment for a complete analysis. The complete list of characteristics were categorized into behavior, system, synthetic natural environment, physical model, and user-computer interface characteristics. The migration assessments were conducted on each of these categories.

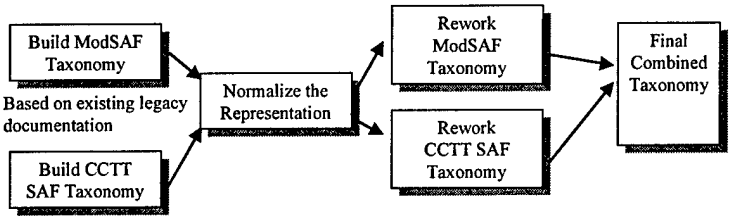

Figure 2: Distinguishing Characteristics Process

3.1.2 Migration Assessment Process

The analysis team assessed the migration options for each category of technical characteristics using the process illustrated in Figure 3. For a particular characteristic

category, each assessment team (ModSAF and CCTT SAF teams) presented the unique characteristics and implementation approaches of their system to the other assessment team. The assessment team leads jointly developed an assessment template to be completed by each assessment team as they evaluated the migration options for integrating the other system's unique characteristics or implementation approach into their system's baseline. These templates defined how the assessment teams would evaluate the migration options from the perspective of the option's technical feasibility, integration program risk, impact to ModSAF users and developers, impact to CCTT SAF users, impact to ModSAF research activities, impact to CCTT extension efforts, adaptability to new technologies and costs relative to the alternative options. Each assessment team then determined how the characteristics and/or implementation could be migrated into their baseline. The assessment teams evaluated these migration options in terms of the criteria identified in their assessment template. The two teams then jointly reviewed the consolidated findings to reach consensus on the resulting quantification of the migration options for the particular category. After all categories were analyzed, the assessment quantification for each criterion across the categories was totaled to provide a combined assessment from which to determine a baseline recommendation.

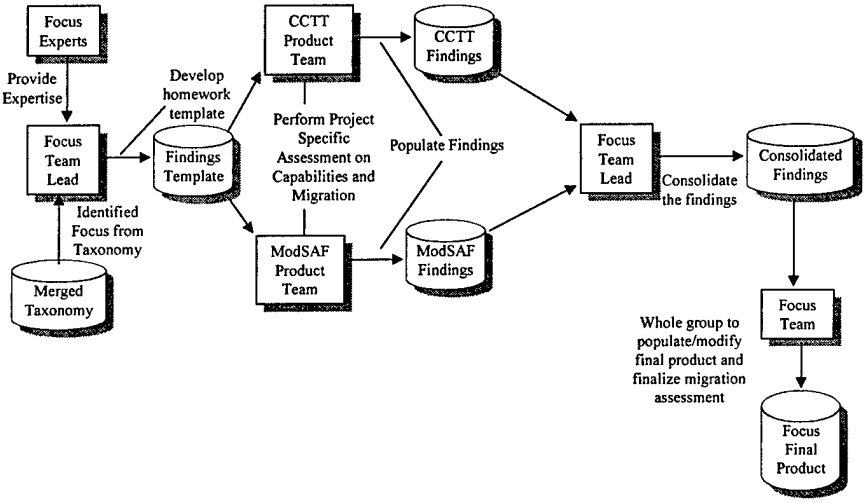

Figure 3: Capability Migration Assessment Process

3.2 Technical Characterization

This section provides a side-by-side technical characterization of the ModSAF and CCTT SAF baselines with respect to their system level, modeling (physical, behavioral, and synthetic natural environment), and user computer interface characteristics. This characterization includes discussion about the system's Software Lines of Code (SLOC), time management techniques, behavioral representation, terrain representation, verification and validation, development language, development environment, hardware platforms, and interoperability and adaptability to other key Army and commercial systems and software. Additionally, this technical characterization describes the maintainability, portability, extensibility, reliability, availability, data collection, traceability, repeatability, flexibility, and affordability of each candidate system.

3.2.1 SAF Development History

ModSAF was designed specifically as a modular, data-driven architecture whose characteristics are based on years of lessons learned through SAF development legacy. The ModSAF architecture was specifically designed to provide all of the general capabilities needed by any entity level virtual modeling system. The ModSAF architecture was also designed to allow the easy development and integration of models that become part of the ModSAF system. The characteristics of the architecture are that it is a single-process architecture and a layered inverted architecture. That is, lower level models are not necessarily invoked though a top-down sequence of procedure calls but rather register themselves to be called based on the occurrence of particular events. To accomplish this characteristic and its data-driven programming style, the ModSAF architecture makes extensive use of particular C-language constructs, most specifically function pointers. Although the ModSAF architecture was developed with general characteristics of SAF models in mind, no explicit set of models were initially required as part the architecture design. A large and diverse development community has added models into the ModSAF architecture over time. These additions range from models developed to support a specific experiment at a US Army Battle Lab, to detailed mine/countermine models in support of the JCOS ACTD, and the environmental and dynamic terrain models in support of the STOW ACTD. This clearly demonstrates the accessibility and extensibility of the ModSAF architecture design.

CCTT SAF was designed specifically to satisfy the modeling and system requirements for the CCTT system. However, CCTT SAF's design was re-engineered using the Ada language based on the ModSAF architecture, since the CCTT SAF design occurred after the initial implementation of ModSAF. Therefore, at the high level of design CCTT SAF and ModSAF are very similar. The CCTT architecture was designed specifically to satisfy CCTT's requirements and no more because of a very short schedule requirement. Adding risk to this short development schedule was a set of challenging requirements that were not satisfied by either ModSAF or any other legacy SAF at that time. These requirements included a very large set of validated and verified combat instruction sets (CISs), a very large and densely populated terrain database, a set of dynamic environment capabilities (e.g., destructible buildings, smoke, weather effects, etc.), and very stringent performance requirements. All of these factors along with CCTT SAF's special roles within the CCTT system caused the CCTT SAF design to diverge from the ModSAF baseline.

CCTT SAF is not as extensible as ModSAF because it did not need to support as diverse a set of developers. CCTT SAF's architecture is designed as a multiple process and multiple thread architecture where the dead reckoning functionality, the network interface, and Plan View Display are separate processes to take advantage of CCTT's multiprocessor platform. CCTT SAF also shares a significant amount of common code within the CCTT software baseline for scenario initialization, terrain interoperability and system component interoperability. The specific areas where ModSAF and CCTT SAF differ are the behaviors architecture since CCTT's design was driven by a specific set of required CISs rather than a requirement to be a general behavior model development environment. Also because CCTT

SAF was designed as a robust production system using Ada, it takes advantage of the language's strong typing and encapsulation practices.

3.2.2 Systems

For the purposes of this side-by-side technical characterization of ModSAF and CCTT SAF, 'systems' will include the features present in each SAF to allow it to operate as part of a larger simulation capability. This includes both the hardware and software architecture as well as the corresponding integration of the SAF software into the surrounding systems. This also includes the development processes, the system performance capabilities, the protocols and standards guiding the system's development and use, and the core simulation services used by the simulation.

3.2.2.1 System Architecture

The system architecture comparison of each SAF includes the system configuration, exercise interface, software architecture, software structure, integration, and accessibility, as described in the following sections.

3.2.1.1.1 System Configuration Overview

Both ModSAF and CCTT SAF were designed to be distributed Computer Generated Forces systems. The system approach for both systems share some similarities due to the fact that CCTT SAF was based on a re-engineering of selected approaches from an early version of ModSAF.

3.2.1.1.1.1 System Diagrams

Figure 4 shows the ModSAF system. ModSAF is composed of two main executables that can be distributed in many ways. The first main executable is the SAF Station. SAF Stations are the human interface to the ModSAF system. The second main executable is the SAF Sims. The SAF Sims provide the modeling capabilities for the ModSAF entities and organizations. SAF Sims and SAF Stations communicate with each other through the use of the Persistent Object (PO) database. This is a distributed database that uses the PO Protocol (POP) to maintain the distributed database. In addition, ModSAF uses the DIS protocol or the SIMNET protocol to communicate synthetic environment physical information. The PO database communicates information between the models that is not represented in DIS or SIMNET. The PO provides command and control information for the simulation of organizations (DIS and SIMNET are oriented to the physical world). Additional executables are noted below.

- DTSim – Dynamic Terrain Simulator: Simulates the dynamic terrain protocol and standards for ModSAF.
- DTScribe – Dynamic Terrain Scribe: Provides persistence, serialization, and distribution of dynamic terrain information.
- Logger – The Logger provides the capability to log DIS or POP PDUs for later analysis or broadcast.

- RTI RD – The RTI Reliable Distributor implements TCP/IP-based reliable communications for the RTI version s.
- Blaster – The Blaster provides the capability to broadcast large amounts of entity state PDUs for network loading testing.
- ModStealth, TAOS, and CFOR represent executables that are not part of ModSAF but the baseline provides interface support for these executables.

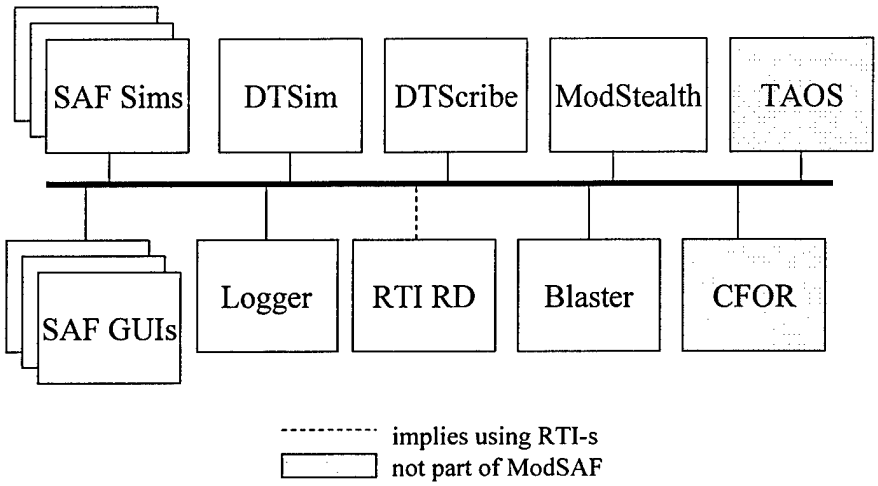

Figure 4: ModSAF System Diagram

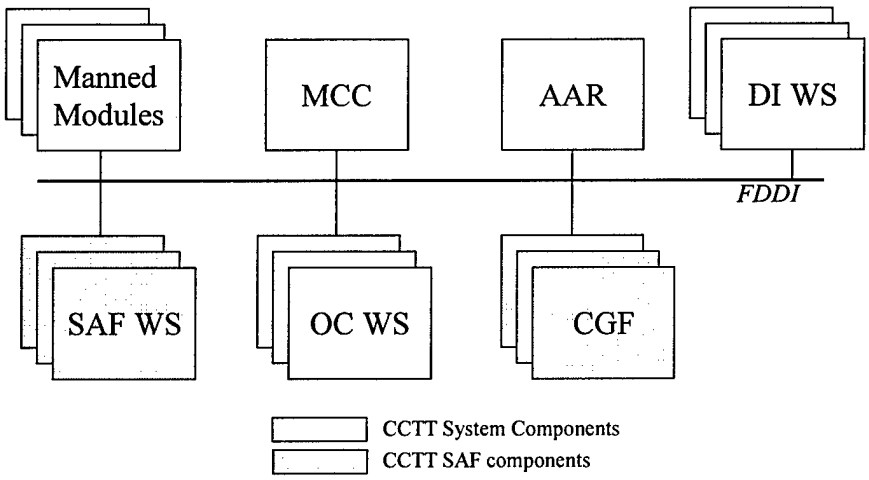

Figure 5: CCTT SAF System Diagram

Figure 5 shows the runtime distribution structure for CCTT. CCTT has similarities to the ModSAF structure. For CCTT, the SAF WS and OC WS are equivalent to the SAF GUI and provide the human interface for the SAF operator and the OC trainee respectively. The CGF corresponds to the ModSAF SAF Sim. The CCTT system consists of other executables such

as the Manned Modules (crew cabin simulations), MCC (master control), AAR (after action review), and the DI WS (dismounted infantry trainee station). CCTT uses DIS to communicate information about the synthetic environment (once again a description of the physical characteristics of that world). CCTT SAF uses the SAF Entity Object Database (SEOD) as a distributed object database for communication between the workstations and the CGF simulators. The SEOD uses the CGF Protocol to communicate data changes and events for command and control information. One point of comparison between the ModSAF diagram and the CCTT diagram is that CCTT SAF exists as a integral sub-component of a larger system.

3.2.1.1.1.2Process Architecture

ModSAF is configured as a single process. All processing is located in a single thread of execution. CCTT SAF is configured as a multi-process system. The following diagrams illustrate the processes in each of the CCTT SAF components.

As shown in Figure 6, the SAF WS component consists of six processes that all communicate through shared memory. The processes are described below.

- SAF WS – This is the SAF WS GUI process. It provides the control over the CGF entities and organizations.
- Comm – This is the communications model process. It provides a simulation of the SINGARS radio for simulated radio traffic.
- PVD – This is the Plan View Display process. This provides the map view of the exercise as well as the graphical overlay capabilities.
- Entity Database – This process handles the processing of remote and local entities for communication via DIS protocols. It provides an abstraction over the entity state PDUs as well as the dead reckoning algorithms.
- Network Manager – This process handles all incoming and outgoing network traffic.
- Routing – The routing process allows routes to be generated for the SAF operator without interfering with other realtime processes.

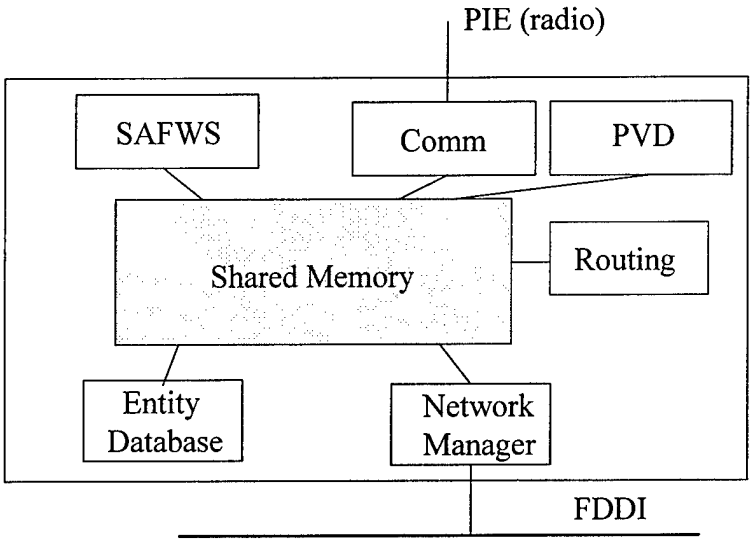

Figure 6: CCTT SAF WS Processes

The CGF component consists of 3 main processes as shown in Figure 7. Two of the processes (Entity Database and Network Manager) are shared processes with the SAF WS and perform the same functions. The additional process is the CGF process. That process provides all the modeling for CCTT SAF entities and organizations.

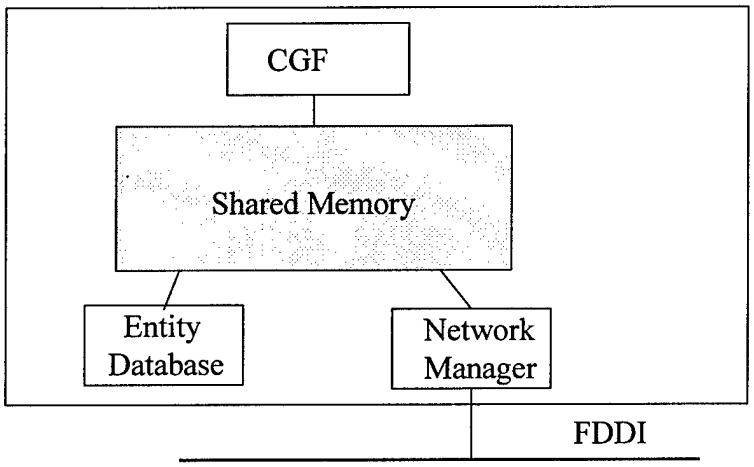

Figure 7: CCTT SAF CGF Processes

3.2.1.1.2 Exercise Initiation/Management

ModSAF exercise initiation and management is localized to the ModSAF executables. It has the ability to create, modify, save or load scenario files, overlay files, and minefield templates. ModSAF accepts simulation management PDUs for start/resume, stop/pause, and

restart. ModSAF uses the concept of a DIS exercise number as well as a partition number for the PO database to allow the partitioning of sets of ModSAF executables between DIS exercises and within exercises.

CCTT SAF is a sub-component of the CCTT training system. For this reason, it has an integrated approach to exercise initiation and management. In CCTT, the MCC (Master Control Console) provides a centralized capability to control all aspects of resource assignments, exercise initiation, and exercise management. CCTT SAF participates in this CCTT exercise protocol. Exercises are centrally stored in an NFS mount that is accessible to all components in the CCTT system. The MCC is responsible for the initialization of components to an exercise and the selection of the exercise to run. This information is communicated to the CCTT system components through a series of simulation management PDUs. CCTT SAF has the ability to create, modify, and delete scenario files and overlay files. The interactions with the CCTT system for starting an exercise are outlined below.

- Exercise assignment – CCTT SAF responds to the create entity and remove entity simulation management PDUs. The create entity PDU assigns the workstations and CGF simulators to a DIS exercise and initiates the system protocols for health and fault status. The remove entity PDU removes an asset from the exercise and places it in a mode to accept a new create entity PDU.
- Initialization – Exercise initialization consists of a series of PDU transmissions that are outlined below.
    - Scenario Assignment – Scenario assignment is performed through the use of DIS action/response PDUs
    - The first is the environmental PDU. It contains the assignment of the synthetic natural environment, weather information, time of day information, environment manager assignments, and checkpoint intervals.
    - The second is the CGF specific PDU. It provides information on the assignment of a CGF to the exercise and a boss assignment. CCTT SAF uses the concept of a boss assignment from the MCC. Since the CGF collection is a peer to peer relationship (rather than a master/slave) one platform is identified as the boss to allow for singular events to be handled (such as creation of relocatables and load balancing).
    - The third is the SAF specific PDU. This contains information on the boss assignment for the workstations, force side information, and the name of the exercise. The boss designation allows for one workstation to load the exercise.
- Exercise control – CCTT SAF responds to the DIS start/resume, stop/pause, and restart PDUs.
- Health status – CCTT SAF responds to periodic health status request messages from the MCC.

3.2.1.1.3 Software Architecture

ModSAF is a single-process architecture. It makes extensive use of function pointers to provide a variety of capabilities as shown below.

- Callbacks – Callbacks provide ModSAF with a registration structure for assigning handlers for a variety of purposes.
- Inverted architecture – ModSAF uses callbacks to invert the architecture. This provides a mechanism for models to register their functions for various purposes such as ticking and event handling.
- Developer defined named events - A developer may create a symbolic name for an event, define when an event is triggered, and support registration of handlers for these events.
- Scheduler – Periodic and one-shot appointments are implemented as function pointers to tick handlers.
- Key technique for extensibility – ModSAF uses the registration capability and the inverted architecture to minimize affected code for new unit or model additions to the system.

ModSAF has callbacks and data driven techniques as the basis for its architecture. This is used in the Graphical User Interface design, entity configuration, network protocols, and behavioral combinations (taskframes).

CCTT is a multi-process architecture. CCTT SAF makes use of multi-process threading to allow for execution of models on the symmetric multi-processing platform used by CCTT. CCTT was written in Ada 83, but makes limited use of function pointers. These are used to implement a callback structure similar to ModSAF. This callback structure is used to allow for registration of event handlers (predefined events) and scheduler appointments (periodic and one-shot). The CCTT SAF architecture is data driven as well but not to the extent of ModSAF.

3.2.1.1.4 Software Structure

ModSAF is a flat hierarchy of about 555 C software libraries. Each library contains (or may contain) a makefile, header files, C source files, data files, GUI resource files, and TeXInfo library documentation. There is a small set (about 10) of source directories containing the main program.

The CCTT SAF software library organization mirrors its design decomposition. It is divided into directories and subsystems (a Rational Apex concept). Subsystems contain the Ada packages (specifications and bodies). CCTT relied on the integrated development environment (IDE) of Apex to build the executables and did not use makefiles. The execution environment contains all the data files and GUI User Interface Language (UIL) files necessary for the execution of CCTT SAF. These are organized along the system architectural types (i.e., the system applications such as the SAF Workstation and the CGF). In CCTT SAF, there is no equivalent documentation to the ModSAF TeXInfo files.

ModSAF SLOC is characterized in Table 3. In this table, ModSAF SLOC is measured in physical lines of code (excluding whitespace and comments), which has been a traditional code-size metric for ModSAF. CCTT SAF SLOC is characterized in Table 4. Traditionally, CCTT SAF has measured SLOC as logical lines of code (which includes only one line of code for every executable statement, no matter how many lines of text that statement spans), however this table presents physical lines of code to provide data that is easier to compare with ModSAF. Depending on the amount of multi-line statements, there will always be more physical SLOC than logical SLOC for a given piece of software.

Component	SLOC
Shared	125,817
Simulation	543,965
Network	67,717
GUI	85,607
Other	99,911
Total	**923,107**

Table 3: ModSAF Physical SLOC

Component	SLOC
CGF	617,549
System Services	100,626
SAF WS	31,068
Total	**749,243**

Table 4: CCTT SAF Physical SLOC

3.2.1.1.5 System Integration

ModSAF is integrated into numerous training, research and development environments as shown below.

- National Guard Training (interoperation through DIS)
- RDA studies (interoperation through DIS)
- BBS linkage/STOW-A
- SOAR
- CFOR
- DARPA STOW '97 (interoperation through DIS)

CCTT SAF software components are integrated throughout the CCTT system (as well as system components integrated into CCTT SAF).

- Network Manager
- Entity Database
- CGF Terrain Algorithms/Data
- TOC workstation support
- DI WS
- PVD

In addition, the SEOD file format is the system exercise format for the CCTT system and the CGF SEOD code is shared by many CCTT components.

3.2.1.1.6 Accessibility

ModSAF is distributed to a wide user base. Processes and organizations (TWSTIAC, DIS Service Center, DMSTIAC, and Nations) have been in place since 1993 that allow the easy distribution and maintenance of the baseline. CCTT SAF is just now being fielded and its distribution is handled by Nations for STRICOM. Presently, the CCTT SAF system is distributed as a sub-component of a larger distribution of the CCTT training system. The ModSAF distribution provides documentation on the build process for SAF and provides readily available build tools that support multiple hardware platforms. The CCTT SAF distribution provides well documented and scripted builds for the CCTT system of which CCTT SAF is just a component. Executables are provided as part of the distribution but to be used, the host system must be configured as a CCTT training site computer. While the automated CCTT build scripts use this configuration assumption, there is nothing inherent in the technical and resource requirements of the CCTT SAF software that limits it to only running in this mode. In fact, several research projects currently use CCTT SAF as a stand-alone system.

ModSAF and CCTT SAF are not specifically designed to provide extensive data collection. ModSAF provides data collection through the logging of DIS PDUs, special verification and validation data PDUs, and built in debug switches. CCTT SAF provides data collection through DIS event or Data Analysis Report (DAR) PDUs, the SEOD Visual Interface, and Aprobe (a real-time debugging tool that was used to support formal VV&A).

ModSAF was developed to allow extension. It relies heavily on a data driven development approach that facilitates modification. In addition, ModSAF provides naming conventions for their library structure, developer courses, and scripts to add new libraries. CCTT SAF was not designed for distributed development. As a production system, it was developed by a co-located Integrated Development Team (IDT). Since this was a large program with a short schedule, the IDT relied heavily on development tool support. CCTT SAF provides some level of data driven design, however, additions to the system rely on code changes.

3.2.1.2 Development Environment

As described in the following sections, the differences in procurement and intended use for each SAF system resulted in very different development environments.

3.2.1.2.1 Development Language and Tools

ModSAF was developed in a research environment and had no restrictions on the language or development processes imposed on CCTT by DoD-STD-2167A. For this reason, ModSAF was originally developed in C. CCTT SAF was developed as a subsystem of the CCTT training system in Ada 83. The ModSAF code is being converted to be compilable by C++ compilers, but it will not use any object oriented language techniques provided by the C++ language.

The development tools for each system were a result of the development language and the intended use for each system. The ModSAF development environment was primarily distributed among many companies and organizations. As stated earlier, ModSAF was not bound to military development standards (2167A) and provided best commercial practices for its processes. ModSAF provides scripts (awk, sed, lex, etc.) and makefiles to support distributed development and experimentation. Also, to keep development costs low ModSAF makes use of the GNU (freeware) tool set. Compilation support is provided either by the operating system native UNIX C compilers or GNU gcc. To promote consistency in a distributed development environment, tools were developed to verify the documentation and programming style of ModSAF code (Interdoc and Interck). ModSAF also provides tools for the generation of task templates to further standardization of behaviors. CCTT SAF does not provide such a tool. CCTT was developed under DoD mil-standard 2167A and has the process and development environment to reflect it. CCTT used Rational Apex and OC Systems PowerAda as the compilation environment. Design and development tools consisted of StateMate, ObjectMaker, RTM, Interleaf, and CMVC. CCTT was developed in a co-located Integrated Development Team with a finite schedule and pre-specified requirements. Runtime support for ModSAF relied on UNIX and GNU through the use of the DBX and gdb debuggers. CCTT relied on Apex, PowerAda adbg, Aprobe, and the developed DIS Visual Interface tool. Analysis support for ModSAF was achieved through the use of V&V Data PDUs while CCTT used the Visual SEOD Interface tool, Map Pages and Aprobe. Aprobe, a standard tool from the PowerAda suite, provided a limited capability for data collection in support of the V&V process. Specifically, software engineers with knowledge of the CCTT SAF architecture and data flows could instrument the system, collect and interpret data.

3.2.1.2.2 Development Process

As stated earlier, ModSAF relied primarily on a distributed development process. In this process, ModSAF code was distributed to many organizations who developed various capabilities as their needs required. The ModSAF baseline was centralized, procedures were established for integration of new capabilities into the baseline, and a review board considered these changes for inclusion in the official baseline. The baseline (under ADST and ADST II) was managed using RCS, CVS, and Clearcase as well as the previously mentioned development environment.

CCTT was developed in an Integrated Development Team consisting of TSM, STRICOM, Nations (IV&V), Lockheed-Martin (prime), SAIC (SAF/CGF), and ECC (manned modules). CCTT was developed under a modified evolutionary development model with 7 builds spread over 4.5 years. The developers of the IDT were co-located in the same facility and shared the development environment and lab. CCTT has been transitioned to Post Deployment Software Support (PDSS) consisting of Nations and SAIC.

3.2.1.2.3 Hardware Environment

ModSAF was developed to execute on multiple platforms. Specifically, ModSAF executes on the Silicon Graphics (SGI), SUN, PC (Linux), and DEC Alpha platforms. The minimum

requirement for a ModSAF capable platform is a PC (Linux) platform costing approximately $5K. CCTT SAF was developed as a subcomponent of a larger production system and was not ported to other platforms. A CCTT SAF platform is an IBM or Motorola AIX computer system and costs approximately $8K.

3.2.1.3 System Performance

The following sections compare ModSAF and CCTT SAF in terms of capacity, scalability, user workload, and reliability.

3.2.1.3.1 Total Capacity

Comparable total capacity and benchmarks are difficult to provide for these two systems. To accurately compare capacity, identical scenarios are required to be built. Currently, this is not possible given the configurations of the two systems. Specifically, ModSAF and CCTT SAF do not execute on a common system. In addition, they do not share common synthetic environments or models. The best that can be done is to characterize existing benchmarks with the realization that the numbers are not directly comparable.

ModSAF expects a processor with a minimum of 128 Mbytes of RAM with 256 Mbytes recommended for performance. ModSAF performance numbers are divided into two groups. The first group is a stand-alone capability with no remote entities or network interactions. The second group is in a network environment with 800 external entities (provided by an external program). This provides information on the interaction across DIS and does not include C2 interactions. For the stand-alone version, a 300 MHz Pentium II supports 200 vehicle simulations while a 195 MHz SGI R10000 supports 112. For the network version, the Pentium supports 132 entities while the SGI supports 80.

CCTT SAF expects a processor with a minimum of 128 Mbytes. CCTT SAF performance numbers are measured as a participant in a DIS exercise owing to the fact that CCTT SAF is a sub-component in a larger system. CCTT SAF is required to provide 25% spare processing capability with 48 vehicles per CGF simulator. This is measured in an 851 entity exercise with 201 manned modules entities and 650 CGF entities. This includes the processing required for the C2 and the interaction of up to 30 SEOD clients.

3.2.1.3.2 Scalability

ModSAF currently partitions the PO Database into groups of 8 maximum participants due to scalability concerns. Each partition is initialized independently and does not share C2 information. CCTT SAF typically supports up to 30 SEOD clients and does not partition within an exercise. CCTT SAF does partition between exercises with each SEOD client belonging to only one exercise.

3.2.1.3.3 User Workload

ModSAF workload was measured during certain Ft. Knox D-site experiments to be on average 40 entities per operator. There was a high variance depending on the side or activity of the entities. During the Synthetic Theater of War (STOW) project, ModSAF averaged up

to 100 entities per operator. CCTT SAF is required to support 120 entities per operator as its baseline case.

3.2.1.3.4 Reliability

Army SAF (based on ModSAF) had average runs of 24 continuous hours during the STOW 97 ACTD exercise. No information presently exists on the ModSAF 4.0 baseline. CCTT SAF requirements are for 1504 hours MTBF.

3.2.1.4 System Protocols and Standards

The following sections describe the network protocols and standards used by each SAF system

3.2.1.4.1 Standards

ModSAF provides support for a variety of standards for synthetic environment communication: DIS 2.0.3, DIS 2.0.4, and SIMNET. In addition, it provides support for the HLA through the use of the RTI version "s" (i.e., the version used by the DARPA STOW program). ModSAF also provides standards for bumper numbers, task organizations, appearance bits for damage, entity enumeration standards, and articulation modeling. CCTT provides support for DIS 2.0.4 and has an extensive interoperability document to support interoperation with the system. In addition, CCTT also provides standards for bumper numbers, task organizations, appearance bits for damage, entity enumeration standards and articulation modeling with CCTT specifics.

3.2.1.4.2 Dynamic Environments

ModSAF provides an extensive simulation of dynamic environment inherited from the STOW development. The ModSAF baseline provides for the DTSim that assesses damage to pre-positioned and dynamic objects due to detonations (not collisions). Dynamic objects (multi-state, repolygonalized, linear, abstract and entities) supported are shown below.

- Tank ditches (abstract and repolygonal)
- Wire (abstract and linear)
- Fighting positions for hull and turret defilade (abstract and repolygonal)
- Minefields (entities) simulated down to individual mines
- Lane markers (entities)
- Craters (abstract and repolygonal)
- Dragons teeth (linear, abstract, and multi-state)
- Rock drops (multi-state)
- Bridges [HAB, AVLB], buildings (multi-state)

ModSAF also provides simulations of a dynamic virtual world (DVW). The capabilities supported are shown below.

- Uniform and gridded weather
- Smoke (obscurants)

- Flares
- Vehicle dust
- Continuous time of day
- Atmospheric haze
- Clouds (cumulus, stratiform, ciriform)
- Uniform ocean surface model
- Total Atmospheric and ocean server interface (TAOS)

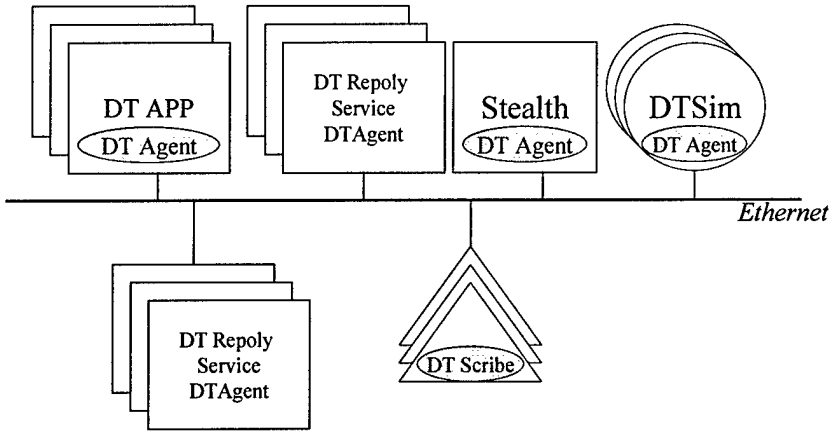

Figure 8: ModSAF Dynamic Terrain System

Figure 8 outlines the processes that are associated with the ModSAF dynamic terrain system. The process for changing the terrain using these services is outlined below.

- Client (a DT APP) sends Engineering Change Notice (ECN) to DTScribe
- DTScribe catches ECNs and is the ECN traffic cop
- DTScribe sends "fully specified ECNs" back out.
- DTSim processes any parameterized ECNs, if any, such as repolygonalization requests. If any, it resends the ECN back to the DTScribe.
- DTAgents embedded in applications catch the fully specified ECNs from the DTScribe and modify the local representations.
- DTSim provides two services: repolygonalization (from parameterized ECNs) and incidental damage assessment (i.e., from detonation PDUs), and a very trivial DT GUI
- Late joiners ask for necessary ECNs from DTScribe.

CCTT provides a simulation of dynamic terrain through the Environment Manager (EM). The MCC provides the EM site/application and IP address to all the applications in an

exercise. Typically, two EMs are added to an exercise with one EM being a hot swap for fault tolerance. The EM assesses damage to pre-positioned and re-locatable objects due to detonation (not collision). Flares are not simulated by the EM but rather simulated by the firing entity. The objects simulated by the EM are listed below. In addition, smoke is modeled only after it detaches from the vehicle.

- Buildings
- Bridges
- Tank ditches
- Wire
- Defilades
- Fighting positions
- Minefields
- Lane Markers
- Log cribs
- Abatis
- Craters
- Ribbon bridges
- AVLB (after placement)

The CCTT EM protocol provides environmental PDUs for point objects, linear objects, and areal objects. Late joiners communicate an action/request PDU to request a rebroadcast of the current objects and their state.

ModSAF provides an extended simulation of minefields. ModSAF supports hand emplaced, mechanical, artillery, customized, and point mines. In ModSAF, minefields are simulated as collections of individual mines. ModSAF uses two DIS PDUs to model minefields: minefield PDU and mines PDU. The mine models determine mine detonations. Damage is assessed by the vehicle. CCTT models the same types of minefields but simulates them as areas and uses probabilistic models rather than modeling individual mines.

3.2.1.4.3 System Protocols

CCTT SAF participates in several system protocols as a sub-component of a training system. These are outlined below.

- Reset
    - 48 checkpoints are saved
    - checkpoint interval defined by MCC
    - Stop/Freeze PDU (stop for reset)
    - Action Request PDU (recall checkpoint data #)
- DRC (Daily Readiness Check) Structure for DRC but presently not used within CCTT SAF
- BIT (Built-In-Test) Action Request PDU, Application returns OK, checked about 1 per minute.
- POST (Power-On Self Test) (AIX stores data in file)
    - MM Code - Return Go for Host computer only
- CCTT Error Detection, Recovery, Logging
    - Event Reporting: Report Exception handling sends error events but no longer used.

3.2.1.4.4 Data Analysis and Review (DAR)

CCTT provides a protocol for system components to report data used for after action review. Data Analysis and Review (DAR) reports are sent for any firing and detonation event providing additional information not found in the standard DIS event PDUs. DAR reports are generated for minefields as a special protocol called the mine detonation event report PDU. In addition, all vehicle simulations send DAR reports for any damage events. DAR reports are also generated for mission request of indirect fire, indirect or Close Air Support (CAS) fire reports, and minefield entry events.

3.2.1.4.5 Battlefield System Protocols

The following battlefield system protocols support the modeling of key tactical capabilities within the CCTT training system. CCTT SAF recognizes these protocols.

3.2.1.4.5.1 Operations Centers

CCTT provides several protocols for integration with the OC trainee stations and their behavioral models. The protocols are a combination of interactions through the CGF protocol and DIS. These are listed below.

- Repair
    - Failed vehicle issue Service Request PDUs to approaching Maintenance Unit continuously
    - Battle Damage Assessment (BDA) - Maintenance sends a Data Query PDU
        - Vehicle sends data defining damage type
    - Repair Complete for particular subsystems (complex set of time outs and authorizations)
- Recovery
    - M88 issues a TOW request (Action Request PDU "hitch")
    - Towed vehicle returns Action Response PDU "complete"
    - Movement by each model (no transfer of control)
        - 5 Meter offset
        - Mass of vehicles affects M88 dynamics
    - M88 issues an Action Request PDU "unhitch"
    - Towed vehicle returns Action Response PDU "complete"
- Resupply (between Manned and CGF)
    - Tether/Untether to MM (Action Request/Response)
    - Service Station Supply from MM (Action Request/Response)
    - Tailgate Resupply
        - Tether-subgroup Action Request/Response
        - Initiate-Tailgate-Resupply Action Request/Response
- AVLB
    - After Detached (using EM protocol), its damage is controlled by EM
    - Reattach (using EM protocol)

For many of these protocols, ModSAF supports similar PDUs (e.g., recovery, resupply) but minor differences exist in the semantic details of the protocols.

3.2.1.4.5.2Dismounted Infantry Workstation (DI WS)

CCTT SAF established standard protocols for the integration of the DI WS, various manned modules, and CCTT SAF. DI WS entities can mount CGF vehicles and manned modules. CGF DI only mount CGF vehicles and never mount manned modules. Three PDUs are used to coordinate the mount/dismount capabilities of CCTT. The first is the action request PDU called the mount intent. This PDU establishes a relationship between the collection of DI WS entities and the vehicle they plan to mount. The second is the mount action/response PDU that initiates the actual mount behavior. The third is the dismount action/response PDU that initiates the dismount behavior. In CCTT SAF, the damage assessment is performed on mounted DI if the vehicle they are mounted on is damaged.

3.2.1.5 Simulation Services

The simulation services present in each SAF system provide core simulation functions. Each SAF system has similar capabilities as described in the following sections.

3.2.1.5.1 Time Management

ModSAF and CCTT SAF time management capabilities are similar. Both systems provide for real-time and simulation time clocks. Both systems rely on Network Time Protocol (NTP) capabilities to ensure that wall clock time is synchronized between distributed components.

3.2.1.5.2 Executive

ModSAF and CCTT SAF executive services are similar. Both systems provide for the allocation and management of computational resources through periodic events and deferred functions (one-shots or time ordered events). Both systems provide for high priority and low priority queues, non-preemptive scheduling, and frame stretching. ModSAF has been altered to provide as fast-as possible processing (available during stand-alone use) while CCTT SAF is multi-threaded to take advantage of Symmetric Multi-Processing (SMP).

3.2.1.5.3 Parameter Data

Both systems provide the organization, loading and defaulting of parametric data. ModSAF provides default data capabilities through architectural features while CCTT SAF provides it through individual application standards. ModSAF provides for more of its capabilities to be parametric owing to the requirements for distributed development and experimentation.

3.2.1.5.4 Information Abstraction

ModSAF provides for DIS PDU abstraction by supporting a data driven description of the information in a DIS PDU. This allows for the redefinition of the semantics of the information in DIS (responding to various standards) through the use of data files rather than code changes. CCTT provides DIS PDU encapsulation to isolate the applications from the

data representations in the PDUs. This minimizes code impacts of changing protocols to specified packages. However, this does not support multiple protocols without coding impact. ModSAF also provides support for bi-endianness.

3.2.1.5.5 C2 Database

Both systems provide a command and control (C2) database capability that is similar. CCTT SAF's SEOD was based on a re-engineering of the ModSAF PO Database. Each of the objects in the database is defined by the clients (typically the behaviors). The definition of the data for ModSAF is associated with the behavioral models. CCTT SAF centralizes this in the SEOD subsystem in multiple package specifications. ModSAF distributes entire object updates and provides for bi-endianness (that is, correct interoperation between computers with differing standards for byte order within a word). CCTT SAF provides for distribution of both objects and events, while ModSAF only distributes objects. CCTT SAF uses attribute updates to reduce bandwidth and increase capacity. In addition, the distribution mechanisms and behavioral architecture of CCTT SAF permits the distribution of a unit organization across multiple platforms. This is currently not possible in ModSAF. The CCTT SAF data model and the application models support a segmented route. This allows for a small route object for typical routes with the ability to chain route objects together for longer routes. This combined with a route abstraction allows for small storage requirements and large route segments.

3.2.2 Models

The following sections describe the modeling capabilities in each system, categorized according to physical, environmental, and behavioral models.

3.2.2.1 Physical Models

Physical models are the simulations of real-world physical devices, such as vehicle hulls or sensor systems. As shown in the following sections, ModSAF and CCTT SAF have very similar approaches to physical modeling.

3.2.2.1.1 General Descriptions

Both CCTT and ModSAF systems include an equivalent level of vehicle classes that are sufficient for supporting typical simulation exercises. For example, both systems have M1 tanks, M2 Armored Personal Carriers, AH64 helicopters, etc.

In order to effectively cover the huge number of classes of vehicles, both systems instantiate a vehicle by assembling components called physical models. In general, hull, turret, sensor, and gun physical model components constitute a vehicle. For example, tank track model (the hull model), daytime optics, night vision, IR sensors (the sensor models), generic turret (the turret model), and main gun and machine gun (the gun models) constitute a M1 tank model in both systems. This composition is specified in a data file. Therefore, each vehicle type is associated with a vehicle physical model composition file.

Behavior models interface with the physical models through well-defined APIs called GMIs (Generic Model Interfaces). In order to make the M1 tank (define above) move, the movement behavior provides a proper command by calling GMIs supported by the hull physical model. Then the hull physical model, which is a generic physical model that interfaces to a specific physical model, calls a proper physical model. In this case, it is a tracked physical model that has M1 tracked hull characteristics. This relationship is drawn in Figure 9. Instead of using the M1 tracked model, the SAF behaviors always go through a software interface whose sole purpose is to provide generic access to physical models. In other words, the behavior models can uniformly invoke functionality regardless of the physical model that executes the invocation. For example, "Set Speed" GMI can be equally used for M1 tank tracked model as well as AH64 helicopter hull model. Thus, the GMI based function invocation implements one very important object oriented programming mechanism, namely polymorphism.

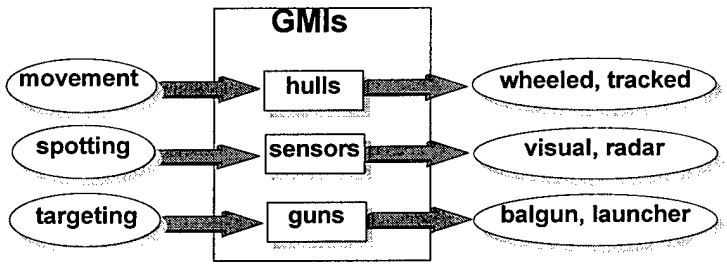

Figure 9: Physical model invocation is performed through GMI

As discussed above, both physical models are essentially equivalent at the conceptual level. Their implementation is also largely equivalent except for small implementation details. The minor differences come from the choices of programming languages -- Ada for CCTT SAF and C for ModSAF.

3.2.2.1.2 GMI Implementation

Translation from GMI invocation to that of a corresponding physical model is the key difference between the two systems. In CCTT SAF, the translation is performed by a dispatch table written in an Ada case (when) statement. For example, when the above "Set Speed" for a M1 tank is invoked through the "Set Speed" GMI, the dispatch table finds a target function from the table and invokes the function related to the M1 tank hull physical model. Therefore, if a new hull physical model is introduced to CCTT SAF, then the dispatch table needs to be updated. Because the dispatch table is textually embedded in the Ada code that represents the hull GMI file, the file must be textually edited, compiled, and linked before the new model is utilized.

ModSAF uses a function pointer registration scheme to resolve the target function that is in the target physical model. Registration of function pointers is performed dynamically in ModSAF. Thus, a new physical model can be introduced in ModSAF without textually editing the ModSAF GMI libraries.

Table 5 lists the current GMIs supported by both systems.

CCTT SAF GMIs	ModSAF GMIs
Hulls, Turrets, Resource Models, Weapons, Sensors	Hulls, Airframe, Turrets, Guns, Deployables, Sensors, Esensors, DsgDetector, MineDetectors, Communication

Table 5: CCTT SAF and ModSAF GMIs

3.2.2.1.3 Model Types

Physical model types are classified into the following categories: hull, turret, weapon, sensor, communication, deployable, damage assessment.

3.2.2.1.3.1 Hulls

Both systems have an almost identical list of hull models. Details are listed in Table 6.

CCTT Hull Models	ModSAF Hull Models
Tracked, wheeled, DI, FWA, RWA, missile, stationary	Tracked, wheeled, DI, FWA, RWA, missile

Table 6: CCTT SAF and ModSAF Hull Models

3.2.2.1.3.2 Turrets

CCTT SAF and ModSAF have only one turret type called generic turret physical model. Weapons and sensors are mounted on components instantiated by the generic turret physical model. A vehicle can have multiple turret components if needed.

3.2.2.1.3.3 Weapons

The weapons models supported by each SAF system are listed in Table 7.

CCTT Weapon Models	ModSAF Weapon Models
ballistic, projectile, missile launcher	ballistic, artillery, missile launcher, rocket launcher, bombrack, laser designation, MICLIC

Table 7: CCTT SAF and ModSAF Weapon Models

3.2.2.1.3.4 Sensors

Sensors are further classified into optical, radar, and generic sensors. The supported sensor types are listed in Table 8.

CCTT Sensor Models	ModSAF Sensor Models
Optical • Direct view • Night vision	Optical • Direct view • Night vision

• Infrared • Thermal Radar • LOS radar	• Infrared • Thermal Radar • Generic radar Generic Sensor

Table 8: CCTT SAF and ModSAF Sensor Models

3.2.2.1.3.4.1 Optical sensors

This sensor is the most extensively used sensor in both systems. This is a line-of-sight class sensor. However, depending on the wavelength used by a sensor, the sensor performance is greatly influenced by environmental effects, such as smoke, fog, clouds, etc.

3.2.2.1.3.4.2 Radar

This sensor type simulates radio frequency line-of-sight sensor.

3.2.2.1.3.4.3 GSM (Generic Sensor Model)

No specific sensing frequency is associated with this model. Instead, it facilitates the implementation of new sensing algorithms. A major use was a ground-based sensor that automatically reports activity of targets within a designated area.

3.2.2.1.3.5 Communication

ModSAF has a generic radio physical model. This model can instantiate any radio type. However, no propagation aspect is modeled. CCTT SAF does not support radio models.

3.2.2.1.3.6 Deployables

Two systems equally support deployables. The details are in Table 9.

CCTT Deployables	ModSAF Deployables
Plow/roller, AVLB, Tent extensions	Plow/roller/rake, Deployable bridge, Mine deployables

Table 9: CCTT SAF and ModSAF Deployables

3.2.2.1.3.7 Damage assessment

Various damage models are supported. Table 10 lists the details.

CCTT Damage Assessment	ModSAF Damage Assessment
Direct fire, Indirect fire, Stochastic, Deterministic	Direct fire, Indirect fire, Stochastic

Table 10: CCTT SAF and ModSAF Damage Assessment

3.2.2.1.4 Configurability

As discussed before, each vehicle type is configured via vehicle configuration data files that allow a user to simply construct vehicle data files using the components listed above. Therefore, both systems have equivalent level of configurability.

3.2.2.1.5 Extensibility

GMI facilitates physical model extension. However, as discussed before, CCTT SAF requires textual editing of the GMI dispatch table while ModSAF does not require modifications. Each physical model is implemented with one clear software unit. CCTT SAF uses one Ada package, while ModSAF employs one C software library. Therefore, both systems essentially support the plug and play paradigm. Physical models are easily plugged in and out.

3.2.2.1.6 V&Ved Models

Physical models of the two systems are implemented based on AMSAA's compendium. This implementation approach is applied to:

- sensor performance
- delivery accuracy (ballistic guns)
- direct fire rate of fire (ballistic guns)
- direct fire damage assessment
- indirect fire damage assessment

Fair fight requirement with CCTT manned modules led to extensive tuning of the CCTT SAF physical model performances.

3.2.2.2 Behavior Models

A computer generated force must have an implementation of the behaviors of the forces that it represents. Both CCTT SAF and ModSAF are dominated (in code size and development effort) by their respective behavioral modeling components. In comparing the two suites of behavioral models, a large number of similarities, and a few key differences become apparent.

3.2.2.2.1 Behavioral Specificity

The two systems take different approaches to populating the virtual world with simulated behaviors. ModSAF takes an approach of attempting to maximize behavioral commonality at the expense of behavioral accuracy. ModSAF behaviors are very generic, and can easily be instantiated on almost any entity with nothing more than a few data file modifications. Many of the behaviors have been implemented to be data-driven, with various configuration parameters to specialize a behavior for a specific purpose. Under this approach, for example, a road march behavior can be used on any ground vehicle or ground unit, however, the

subtleties that distinguish a Tank Platoon roadmarch from a Mech Platoon roadmarch, such as doctrinal stop points, are not represented.

On the other hand, all CCTT behaviors are baselined to a specific Combat Instruction Set (CIS), and explicitly represent the different subtleties between like behaviors documented in the CISs. There are separate code modules representing the Tank Platoon roadmarch and the Mech Platoon roadmarch. In many cases, the components that are common between two or more behaviors, for example basic road movement, have been abstracted into a common service used by those behaviors. Unfortunately, this is not always the case, and duplicate implementation of common behaviors does exist in the software. Despite the coding inefficiency, however, there is a better match to CIS-documented behaviors using this approach, as there are logical places within the software to represent and implement behavioral subtleties that are distinguished by unit.

3.2.2.2.2 Behavior Organization

The two systems take different approaches to the organization of their behaviors.

With respect to software organization, CCTT SAF has a unit-oriented behavior architecture, where behaviors are organized by the types of units that can run them. For example all behaviors corresponding to a BLUFOR tank platoon are organized into one Ada package. As stated earlier, ModSAF has a flat organization of libraries, including the behaviors. ModSAF does distinguish unit-level behaviors from vehicle level behaviors, but the layering of behaviors is not explicit in the source code structure.

In the area of runtime organization, ModSAF uses a task frame approach to organize behaviors. Multiple tasks run in parallel in a given task frame. A module called the task manager performs the scheduling of behaviors within one or more task frames for a unit. The task frame supports parallel cooperative activity. One use of parallel tasks is to support the checking of reaction trigger conditions in parallel with normal behaviors. CCTT takes a more explicit control flow approach, where a specific behavior does an explicit check of trigger conditions prior to the main body of behavior execution. In this sense, CCTT has a more cooperative behavior approach, in that the behavior can explicitly control when to check for certain conditions or when to perform certain functions. ModSAF's independent tasks within a task frame result in a more competitive behavior implementation. Because of this, in some cases it may be more difficult for the ModSAF developer to accomplish his objectives in a ModSAF behavior.

Both systems have a notion of behavioral hierarchy, where, for example, company behaviors may issue platoon behaviors to be executed by the company's platoons. This hierarchical approach present in both systems has a tremendous benefit in simplifying the development of higher echelon behaviors, however it has the unfortunate side effect of requiring that lower level behaviors be built prior to higher level behaviors. This causes some amount of serialization during the behavior development process. Also, it is frequently the case that a higher level behavior requires subordinate behaviors that had not been identified as needing to be implemented.

3.2.2.2.3 Low-Level Behavior Approach

The two systems take different approaches in their low-level behavior approach. Low-level behaviors control physical systems within a vehicle. ModSAF calls these behaviors "vehicle tasks", while CCTT SAF calls these "crew behaviors".

In ModSAF, a competitive behavior approach is taken for low-level vehicle tasks. The ModSAF behavior framework encourages developers to implement multiple tasks that run in parallel. An example of this is the vehicle search behavior, which slews a tank's turret back and forth to scan an area for suspected enemy. A vehicle targeting behavior can run in parallel to this search behavior, to control the turret for tracking a target. These two behaviors compete for access to the turret resource. At any given time, only one of the behaviors should actually command the turret to change its azimuth. A prioritization mechanism is used to arbitrate these competing demands. In ModSAF, a mutual arbitration approach is used, where each task that intends to use a resource must declare its intention to the prioritization module. Resources categories are defined for most physical systems, such as the hull resource, the turret resource, the gun resource, and the sensor resource. A finer level of categorization can be supported, for example to separate driver sensors from turret-mounted sensors.

The process of requesting a resource can take many forms. For example, a behavior may request access to a resource whenever the behavior is in a certain state, or whenever a certain predicate function returns TRUE. The prioritization module accepts all the requests and then selects which task will get exclusive control of a resource. If more than one task requests the same resource, a data-file driven priority list is consulted to pick the highest priority task. The system is called mutual arbitration because each task must be written in such a way that it is aware that it is competing with other tasks and may not get its desires satisfied.

This competitive approach to low-level behaviors can support very complicated control flow between software modules. This power of expressiveness can sometimes be difficult to control by a developer, however, the ModSAF implementation has a very positive benefit of making it very easy to add new vehicle-level behaviors. The prioritization service module is service-based, and its code does not need to be modified when new vehicle-level behaviors are added (however, this module's static data-file may need to be extended to properly address the relative prioritization of a new task with respect to other tasks). This implementation allows a new vehicle behavior to be added into the system without having to explicitly modify software that arbitrates when this vehicle behavior can run and use resources.

In CCTT SAF, a more cooperative behavior approach is taken for crew level behaviors. A number of modules encode the crew level behaviors, such as the driver crew or weapons crew. These modules have explicit control over vehicle resources and were designed to manage these resources to avoid conflicts. This approach has an appeal in explicitly implementing the inherent close-coupling of low-level behaviors, potentially eliminating unnecessary competition and computation. It is likely that this more direct-call interface to vehicle behaviors and vehicle resources can allow implementation of more expressive direct control and coordination.

However, the close coupling approach in CCTT SAF's crew behaviors is not without its cost, and that cost is the embedded complexity of these all-encompassing crew behaviors. By incorporating all of the weapons crew in one module, it may be difficult to allow sub-module replacement or refinement. For example, unlike ModSAF, it is not possible in the existing CCTT SAF crew architecture to develop an entity that uses all of the weapons crew except for a specialized threat-assessment module. The sub-models are not plug-replaceable, as they are in ModSAF's vehicle tasks.

3.2.2.2.4 Specialized Behaviors

Both CCTT SAF and ModSAF represent specialized behaviors that merit attention.

Each system supports a Command from Simulator behavior that allows a manned simulator to lead a platoon of SAF vehicles. Each system has approaches to allow the CGF vehicles and the manned simulator to work cooperatively. Within CCTT, a great deal of attention has been placed on the development of this behavior, and a number of specialized CISs have been developed to allow the SAF operator to support coordination between the manned simulator and the SAF platoon. ModSAF has also developed some of these capabilities, such as a cue-fire function that forces the SAF vehicles to wait for the manned simulator to fire before they will engage the enemy. It is most likely the ModSAF implementation has not seen as much use as is intended within CCTT, however, ModSAF does support this capability for both tanks and rotary wing aircraft. Insufficient time prevented a complete side-by-side analysis of these behaviors and further examination of their relative strengths and weaknesses should be performed as follow-on to this study.

It appears that CCTT SAF has implemented more robust low-level behaviors supporting target search and engagement. A modified scanning algorithm has been implemented that supports lingering on detected but not-yet identified targets, as well as revisiting of known targets during the general scanning process. Once engaged, a distribution of fire algorithm ensures a good allocation of shooters to targets. CCTT SAF can employ multiple weapons on one platform, which ModSAF cannot currently do.

Because of its breadth of use, ModSAF has a larger number of varied specialized behaviors that go beyond ground armored combat. For example, ModSAF supports Theater Missile Defense (TMD), chemical detection, and a number of future systems, such as ASTAMIDS, MMCM, and ORSMC mine-detection and mine clearing vehicles.

3.2.2.2.5 Behavior Representation

Each system has both similar and unique characteristics of the basic architectural representation for behaviors. The following sections describe these characteristics, compares and contrast them for ModSAF and CCTT SAF.

3.2.2.2.5.1FSM Representation

The two systems have both similar and differing behavior representations. Each system uses Finite State Machines (FSMs) to implement behaviors. The FSM approach breaks down a behavior into one or more states. During each state, a behavior may perform some activity.

Transitions between states are performed based on certain state-specific criteria, such as the completion of an activity or the receipt of some input.

Though the FSM approaches in both systems are similar, the implementations of the FSMs are quite different. CCTT SAF encodes FSMs using Ada native language capabilities. An FSMs control structure is encoded as an Ada 'case' statement, where each branch of the case corresponds to a different state. The case statement dispatches to a branch according to the value of the current state, and once within a branch, the current state may be modified to become another state based on some triggering condition. Although the language construct for an Ada FSM is standardized (i.e., the case statement), the organization of the code and its dependent data structures and calling routines can vary according to the preferences of the behavior developer.

In ModSAF, a customized FSM representation was developed. Called the Asynchronous Augmented Finite State Machine (AAFSM) language, this representation is an extension of standardized C syntax. ModSAF scripts convert AAFSM files into pure C code prior to compilation. The AAFSM language standardizes the types of data structures used by each ModSAF behavior, and the code generation performed by the ModSAF scripts takes care of the "house keeping" maintenance of these data structures, including reflecting data changes to the persistent object database and detection of externally-generated input parameter changes.

The AAFSM language extensions allow ModSAF behavior developers to express the textual layout of the state machine and to declare each of the machine's states and transition using explicit syntax. The language supports normal operation of the state machine, as well as the ability for the behavior to specify how to react to changes in the behavior's input during operation (these are called parameter events). This forces the behavior developer to think about how the behavior should respond to input changes in each state. For example, a movement behavior might respond to its input route changing by computing the nearest new waypoint to head toward.

The net effect of the use of the AAFSM format in ModSAF is that all behaviors have a similar software structure that is easily recognized by ModSAF behavior developers.

3.2.2.2.5.2Behavior Communication

CCTT SAF takes a much more real-world approach to communication between behaviors. In CCTT SAF, all behavior cooperation is implemented through a reports and orders paradigm. Behaviors can cause subordinate units to act by issuing orders, and subordinates indicate their behavior progress by issuing reports. This approach results in a looser coupling between behaviors at different hierarchical levels, since a standard order or report format defines the interface. In ModSAF, behaviors directly modify parameter inputs to subordinate behaviors.

One ramification of CCTT SAF's approach is the increased maintenance requirement on the orders and reports data structures. This formal interface makes it somewhat more difficult to extend behaviors to do such things as adding new parameters. A benefit of CCTT SAF's

approach is the reports and order data structures can form a useful API for other external uses.

CCTT SAF implements behavior communications that may result in the modification of a data structure as a request to the owner of that data structure. This forces a receive-order serialization of requests, which obviates the need for arbitration of object changes. An example is the way that a unit's mission is modified. The user interface does not directly manipulate the mission, but instead sends requests to the unit that owns the mission to modify its mission. This decouples specific actions, such as modification, from the data model and data structure, such as lists.

The orders and report paradigm forces the implementation of report processing within CCTT. Every tick of a unit calls one or more functions to handle battle loss reports and spot reports. These functions poll for the existence of an event for the unit (such as receipt of report), and responds appropriately to those reports.

3.2.2.2.5.3Behavior Ownership

ModSAF and CCTT SAF take different approaches to ownership of unit behaviors. In CCTT SAF, an explicit simulation object represents platoons, companies, and other higher-level units. These objects are responsible for when their respective behaviors run. In ModSAF, the vehicle that is currently "in-charge" of a given platoon or company is made responsible for running the platoon or company's behaviors. In a sense, all unit behaviors execute within a vehicle context. This is accomplished as a service of the task manager, which locates all the behaviors that a particular vehicle should execute.

The approaches have relative strengths and weaknesses. In ModSAF, the attrition of a leader will cause some leader context information to be lost for coordinating the behaviors of its subordinate units. This is more closely matched to what happens in the real world. In CCTT SAF, the execution of the platoon behavior is unaffected by vehicle attrition. An advantage of the CCTT SAF approach is that the behavior processing of superior units such as platoons and companies can be reduced compared to the processing required for vehicles, as less frequent updates are required for these higher level behaviors.

3.2.2.2.5.4Order Decomposition

The execution of behaviors by a military unit requires that higher echelon behaviors be decomposed into appropriate behaviors for subordinates. For example, a company attack mission may require that the company's platoons conduct various subordinate behaviors in support of the company's mission. The process of turning a high-level order into one or more low-level orders is called order decomposition.

ModSAF and CCTT SAF take different approaches to order decomposition. Both systems rely on a representation of a Task Organization (TO) to describe the military organization of superiors and subordinates. In CCTT SAF, orders are composed when assigned, primarily through the user interface. When an order is assigned, a special module decomposes that order for the appropriate subordinates. When performed at the user interface, the user has

instant visibility into what the units and subordinates will do. The user may make changes to the decomposition, and may save those changes as part of the scenario exercise.

In ModSAF, order decomposition does not occur until the behavior is activated at runtime. The user does not have up-front visibility into what the higher-level mission will do at the lower level. When the behavior is started, the behavior directs the runtime decomposition to the subordinate units. This allows ModSAF behaviors to perform accurate decomposition based on the runtime environment. For example, if at the time that the company behavior is started, one of the company's platoons has been completely attrited by combat, the company behavior can choose to intelligently decompose the mission to only the surviving platoons. In CCTT SAF, the decomposition process cannot anticipate the battle state that will exist when the order is actually executed.

These two approaches to order decomposition have varying strengths and weaknesses. The ability to edit and save the decomposition is a powerful benefit of the CCTT approach, while the ability to perform decomposition based on runtime information is a powerful benefit of the ModSAF approach. The CCTT approach may have a closer match to real-world mission planning.

3.2.2.2.6 Behavior Control Paradigms

Both ModSAF and CCTT SAF support 3 basic types of behavior control. Pre-planned missions are specified by the SAF operator in the construction of an execution matrix for a unit. The execution matrix specifies the sequencing of behaviors (either CISs or task frames) for a unit, and allows control of the transitions from one behavior phase to the next. Transitions can be specified based on time, crossing a control measure, on command from the operator, or some other simulation event. The types of transition triggers are described in the UCI section.

The SAF operator has the ability to modify the mission during execution, which is the second type of behavior control. ModSAF supports the modification of mission graphics and behavior inputs, as well as the concept of Immediate Interventions that make temporary modifications to the mission. These modifications can be restored (resumed) at a later time. CCTT SAF supports Command Overrides, which present a limited set of modifiable parameters for a mission, such as speed and formation.

The third type of behavior control is the ability to implement automatic reactions in ModSAF, or the Situational Interrupts (SI) in CCTT SAF. CCTT SAF contains codified reaction tables specified by doctrine and CIS. ModSAF has reactions enabled by data files specifying the contents of task frames. When activated, a reaction or SI overrides the current mission with a new behavior. The reaction or SI can be resumed at the direction of the SAF operator.

Behavior control requires the use of some type of User Computer Interface (UCI). ModSAF supports the ability for tasks to specify their corresponding task editor via a data file associated with the particular behavior library. This allows new behaviors to be added to the

ModSAF without requiring new UCI code to be developed. CCTT SAF does not have this capability.

3.2.2.2.7 Traceability & V&V Status

The two SAFs have different requirements on traceability and V&V. ModSAF has had some of its behaviors based on interim CIS specifications, however, many have been implemented based on best engineering judgment and various sources of contractor- and government-based knowledge acquisition. In support of certain exercises such as the Anti-Armor Advanced Concept Demonstration (A2ATD), some ModSAF behaviors have been verified, validated, and accredited for use within a particular exercise. Also, some amount of V&V has been performed under the auspices of the ModSAF Integrated Development Team (IDT) as part of the ModSAF management by STRICOM. However, ModSAF has not undergone exhaustive V&V.

CCTT SAF has had very strict traceability and V&V requirements.

CCTT SAF development has had the advantage of being able to refer back to CIS specifications to resolve questions concerning correct behavior implementation.

3.2.2.2.8 Behavior Breadth & Depth

ModSAF has a total of 106 assignable task frames, composed of 234 separate unit or vehicle tasks. Approximately 26 reactive task frames are implemented, and each of these is embedded in some unit behavior that decides to start the reaction.

CCTT SAF has 276 CGF CISs, as well as 75 OC behaviors. There are approximately 373 FSMs in the CCTT SAF baseline.

3.2.2.3 Synthetic Natural Environment

Synthetic Natural Environment (SNE) represents the passive/non-sentient parts of the simulation, including terrain/features, ocean, and weather. SNE provides services to support behavioral models in reasoning/planning and physical models in correctly responding to ground truth.

Throughout this section, "terrain" means land/terrain skin/terrain features and "environment" means weather, ocean state, and effects (like flares).

3.2.2.3.1 Interface Structure & Architecture

The overall interface structure of CCTT and ModSAF SNE are one of the biggest differences, even though these interfaces provide much the same functionality.

ModSAF's terrain representation is subdivided into independent modules with no externally enforced or visible hierarchy (although there is, of course, a dependency hierarchy within the SNE modules based upon which module calling structure). Also, ModSAF's terrain libraries often export very primitive data (e.g., individual features) through their external interface. For example, libctdb's external interface (i.e., libctdb.h file) contains the routine to get height of terrain and also provides the complete definition of the ModSAF's compact terrain

database (CTDB) structure (over 4,000 text lines). This exposes the casual users to the module's most private (and volatile) data. This approach supports swapping out or adding modules at the expense of interface complexity and a requirement for strong developer discipline. For example, a new module that relies on primitive terrain data can be added without expanding or modifying libctdb, but only developer discipline prevents the movement of a bit in a CTDB feature representation from impacting many modules.

ModSAF's environment representation (DVW) is much more strongly structured, in that libenvironment provides the primary means for extracting environment data (via tags and resolvers). DVW's flexibility is provided by allowing any library(ies) to register their ability to support a tag (or many tags). This leaves the interface strongly abstracted and consistent while allowing the ability to swap libraries / capabilities beneath this interface.

Because ModSAF's terrain and environment capabilities were developed independently, they are not fully integrated across all libraries and capabilities. A given ModSAF consumer (e.g., a sensor) that is impacted by both terrain and environment must separately query those areas, or use another library which does so. The inter- and intra-relationships between SNE libraries and their users must be tracked based upon dependencies (#include relationships) and code inspection. For example, a developer may wonder if a concealment-finding routine considers tactical smoke from libenvironment or just terrain/features from libctdb The developer would have to look through both directories to understand these dependencies.

By contrast, CCTT's SNE interface is strongly centralized and encapsulated at the Environment CSC (and PVD CSCI for 2D viewing data). No portion of a CCTT database file is visible from (or is with'd by) an external package specification. For example, CCTT's model reference terrain database (MrTDB) could be swapped out without impacting an external consumer or external specification. CCTT CGF's Environment software component (CSC) is the sole source for SNE query/reasoning services, and always integrates all underlying data into a single answer for callers (just as the PVD CSCI is the sole source for 2D display information). This strict encapsulation dictates the functionality that needs or operates on primitive SNE data must be pulled into Environment CSC. To emphasize this structure, Environment CSC software is stored hierarchically where only the Environment root directory provides external interfaces and subcomponents of Environment are in subdirectories which consumers may not use. Environment CSC's external specifications are split by type of function (e.g., line of sight, cover and concealment, route planning) to lessen the extraneous material a given customer must wade through. A user of height of terrain functionality need only look at a 300 line specification.

While ModSAF's decentralization and wide-open interfaces require strong developer discipline (and experience), they also provide a great deal of flexibility. Environment CSC's interface supports formal software engineering principles (encapsulation/hierarchy), but it thereby lessens the flexibility and codevelopment possibility ModSAF supports.

3.2.2.3.2 Database Format & Primitives

The terrain databases of ModSAF and CCTT share a common ancestor in SIMNET TDB. CCTT SNE used libCTDB (circa CTDB format 2) as a starting point for reuse and database

format. ModSAF's CTDB format 2 and CCTT's MrTDB were very similar in terms of functionality, but they branched off and simultaneously evolved along separate paths because they were driven by different requirements. A few years later, however, in an effort to produce a common terrain database that was available in both MrTDB and CTDB formats (for a SAF interoperability exercise), features that MrTDB supported that CTDB did not support (i.e., basis sets and model references) were added to CTDB.

3.2.2.3.2.1Header/Global Data

Both CCTT SAF and ModSAF databases contain the zone number, the northing, and the easting of the southwest corner. These numbers describe the database's location on the surface of the earth. Both also contain the database's length and width in meters, which describe the geographic extents of the database. ModSAF databases also contain the geographic extents in latitude/longitude format.

Both databases use models to store repeated/common data, both recognize subdivisions of databases in terms of pages (cacheable units) and patches (feature containers), although CTDB also uses a quadtree storage scheme for its abstract features. MrTDB also has the "obstacle/avenue linkage" to maintain a connection between obstacles and features which override those obstacles.

3.2.2.3.2.2Coordinate Systems

ModSAF supports the following coordinate systems

- geocentric
- topocentric (arbitrarily placed)
- geodetic (latitude/longitude)
- Universal Transverse Mercator projection (UTM)
- milgrid (MGRS)
- global coordinate system (GCS)

CCTT SAF supports the following coordinate systems

- geocentric (for DIS PDUs only)
- topocentric (arbitrarily placed, used internally throughout CCTT software)
- geodetic (latitude/longitude, for database origin only)
- Universal Transverse Mercator (UTM) projection (for PVD only)
- milgrid or MGRS (for PVD only)

Because it supports the global coordinate system, ModSAF databases may have unlimited geographic extents and may span multiple UTM zones. Earth curvature effects on intervisibility and elevation are taken into account for GCS databases. CCTT databases do not support GCS and cannot span multiple UTM zones.

3.2.2.3.3 Terrain Skin Representation

Both ModSAF and CCTT SAF support uniform grids for terrain skin representation. Both

support arbitrary grid post spacing and diagonalization can be specified on a per-grid post basis.

TINs, or triangulated irregular networks, are used to represent terrain skin whose granularity is too great to be represented by the uniform grid. The gridposts are ignored in areas of the terrain skin covered by TINs.

ModSAF has two representations for TINs: microterrain and terrain elements. Microterrain is the more compact way to represent TINs. Microterrain stores the vertices of the terrain skin polygons as well as soil type information. Terrain elements include not only vertices and soil type information, but also information about the TIN topology. This greatly increases the computational efficiency of the intervisibility algorithms. Because of terrain elements, algorithms using fully-TINed databases perform reasonably well whereas these algorithms using the same fully-TINed database stored as microterrain would perform very slowly.

CCTT SAF only supports microterrain as its TIN representation, although it does use a specialized representation for emulating TIN-like data for cut and fill features.

ModSAF supports an advanced attribute database that can be used to store an arbitrary amount of information associated with each soil type code. There can be up to 256 soil types per patch (typically 500m X 500m). CCTT SAF supports up to 256 soil types, although only 32 of these are used.

3.2.2.3.4 Feature Representation

A number of terminology and storage-scheme differences complicate any comparison of features supported by the two SAFs. For example, CCTT does not draw a distinction between "abstract" vs "physical" features (except by how the features are used), whereas CTDB stores and retrieves them separately (even using different storage structure: quadtree vs patches, respectively). ModSAF does not draw a distinction between static and dynamic terrain features (e.g., treelines and deployed fences are both physical linear features), while CCTT stores and retrieves them separately (terrain features vs relocatable objects).

Rather than force the comparison into terminology foreign to both systems, ModSAF's terminology is used for the feature comparisons (with differences noted for CCTT), and dynamic terrain is covered separately.

3.2.2.3.4.1 Physical Features

ModSAF's physical features can be loosely described as those which may impact intervisibility (except, for example, laid linear features), and are oriented toward a description of geometry. They are the descendants of features stored in CTDB format 2 prior to integrating quadtree (with extensions to handle new features such as fences as linears).

Volume Features represent a three-dimensional mass resting on the terrain surface. The vertices of the "roofline" of the feature are stored in the database. It is assumed that the volume rests on the ground; thus, "floating" volumes cannot be represented. Both ModSAF and CCTT SAF can represent the following volume features:

- Buildings (includes power poles, towers, etc.)
- rock drops (known in CCTT as rock outcroppings)

The only significant difference here is that CCTT also supports full model references for geometry (that is, a building instance may also be described as a point and orientation, with the footprint of the building described in a model for dynamic placement). This helps preserve the meaning of a single building (or bridge) across patch boundaries (instead of splitting the building into two), and could also support changing of footprints based upon damage (although only height is used now based upon the CCTT database source).

Linear Features are defined in ModSAF as having a height significant enough to have an effect on intervisibility. In CCTT, "Linear Feature" is analogous to "Laid Linear Feature" described below. Both ModSAF and CCTT SAF can represent the following linear features:

- treelines (although CCTT's version was never tested due to changing requirements)
- wire
- fences (several types)

Additionally, ModSAF can represent dragon's teeth. Note that wire and fences are described in CCTT as "Linear Relocatables".

Treelines are ModSAF's answer to an S1000-specific solution to tree density (paper-thin green wall representing many trees).

Tree Canopies are "circus tent"-like structures with a canopy top represented as a TIN. Treelines surround the outer edge of the canopy. This is a cheap (both computationally and storagewise) method of representing forests. Tree canopies are ModSAF's representation of an S1000-specific visual scheme. CCTT uses individual trees in all cases (never treelines or canopies), but space is provided for both in MrTDB.

Aggregate Features are a collection of features that can be referenced and instanced. They are commonly used to represent forests made up of individual trees (as opposed to tree canopies). Forests are commonly represented in this manner by placing tree templates in a tiled fashion to make up the bulk of the forest, then individual trees are placed along the perimeter of the forest, so that its perimeter does not consist of straight edges. Representing forests in this manner greatly reduces the storage requirements for forests made up of individual trees. This technique was heavily used on CCTT's databases, especially the forested Primary 1.

Both ModSAF and CCTT SAF support aggregate features, although CCTT takes advantage of the representation features unique to the Evans & Sutherland databases for better performance (specifically, plane-slope is stored with instance and implied grid masks are used). The relative performance implications of the more or less generic approaches are unknown.

Laid Linear Features have a specific width and rest on top (conform to) the terrain surface. They have no height that can affect intervisibility. Both CCTT and ModSAF support roads and rivers in this form. CCTT fully expands the outer boundaries of all linears (whether

conformal or not). While ModSAF's centerline representation is more computationally efficient, it leaves gaps in turns (especially sharp turns or wide linears) and also when crossing a patch boundary at an angle (not perpendicular). These gaps can lead to terrain type anomalies, such as a dust cloud suddenly appearing while following an asphalt road.

CCTT calls these features "linear features", and treats railroads the same way. Railroads are an abstract feature in ModSAF, as shown below.

Abstract Features can be loosely described as abstractions that are important to the simulation, but do not affect intervisibility, although some are abstractions of physical features that can impact line of sight (e.g., bridges). These features are the descendents of the old quadtree database (and are stored as a quadtree), which integrated into CTDB to form format 3.

As described above, CCTT does not draw a distinction between abstract vs physical features, except as implied by the functional use of the representation.

ModSAF can represent the following abstract features (italics = not in CCTT):

- *tree canopy* (CCTT: uses forest boundaries)
- soil defrag
- steep slope
- railroad
- *pipeline*
- *political boundary*
- *powerline*
- label (CCTT: part of PVD database)
- *tactical sign*
- *offroad segment*
- bridge (CCTT: obstacle/avenue linkage, road net, and separate feature type)
- crater (CCTT: point relocatable object)
- ditch (CCTT: linear relocatable object)
- fighting position (CCTT: point relocatable object)
- *dragon's teeth*
- *berm*
- *camouflage*
- rubble (CCTT: relocatable object)
- *snow*
- *miclic*
- *contaminant*
- hydro surface (CCTT: areal feature)
- *hydro subsurface*

CCTT SAF can represent the following (italics = not in ModSAF):

- forest boundaries (analogous to tree canopy edge bits)
- soil defrags (typically no-go only)

- steep slopes
- label (part of CCTT PVD database)
- railroad (as a CCTT linear feature)
- bridge (via obstacle/avenue linkage, road net, and separate feature type)
- crater (point relocatable)
- ditch (linear relocatable)
- fighting position (6 types, point relocatable)
- *log crib* (point relocatable; ModSAF could handle a volume physical feature)
- *minefield* (obstacle for path planning; ModSAF supports minefields, they are not part of SNE)
- *obstacle/avenue linkage* (stores "avenues" through obstacles, e.g., bridge over river or road over no-go terrain type).

Multi-Level Terrain refers to multiple valid elevation values at a given 2D location. Both ModSAF and CCTT SAF can represent bridges and overpasses. ModSAF handles tunnels, while CCTT has limited but untested capability with tunnels.

ModSAF takes multi-level terrain one step further and can represent multiple elevation structures. Multiple elevation structures (MESs) can represent buildings with multiple floors and rooms. MESs can represent walls, windows, doors, and stairways. They are also used, in addition to multi-elevation microterrain, to represent bridges, tunnels, causeways, and overpasses.

Model References are used by both SAFs for storage efficiency. A model contains information about the characteristics of a certain terrain feature, for example, a tree. Instead of storing the same information for each feature, the instance of the feature references the model. Thus, many features with similar characteristics can reference the same model. This eliminates duplication of storage of feature characteristics.

Most model capabilities are the same for both SAFs. The differences are:

ModSAF only:

- Linear models for treelines, fences, and dragon's teeth (ModSAF uses the same model for trees/treelines that CCTT uses for trees only).

CCTT only:

- CCTT building (volume) models include footprint information that may selectively be used by model instances (this supports footprint changes from damage and patch crossing buildings).
- CCTT also has bridge span models that may be used to capture the presence of a single bridge which spans multiple patches.

Network Topologies support planning. Both ModSAF and CCTT SAF store precomputed road network topology, as well as relationships between various cross-country obstacles. This information is used by both SAF systems to compute routes.

3.2.2.3.5 Dynamic Terrain

Dynamic terrain refers to features that can be created, modified, or deleted during runtime (or planned pre-ex and instantiated at initialization).

ModSAF's dynamic terrain approach, as developed during STOW, is very different from CCTT's. Both systems' representation reaches beyond SAF into other system components: STOW's Dynamic Terrain Object (DTO) includes DTscribe, DTsim, and ECNs; CCTT uses Environment Manager, MCC, and point/linear/areal PDUs for relocatable objects. These differences are not discussed here.

Independent of differences with the overall architecture/system components, CCTT and ModSAF handle dynamic terrain very differently within their SNE software. CCTT processes dynamic terrain (relocatable objects) separate from terrain features for storage, retrieval, and representation. A given external routine, such as line of sight, separately queries terrain, features, entities, and relocatable objects. There are, of course, some interrelationships between these components, such as features or relocatable objects overriding portions of the terrain skin. In contrast, ModSAF largely integrates dynamic terrain into the in-memory copy of CTDB. CCTT's approach adds some complexity and miscorrelation caused by treating relocatables as an exception to the terrain database, while ModSAF's more generic approach requires locking any modified patches/pages into memory thus limiting scalability.

CCTT's system-wide dynamic terrain approach was heavily influenced by system integration (specifically, image generator costs and correlation) and thus is very CCTT-specific. The network and internal representation of dynamic terrain objects matches the E&S relocatable object representation and geometry closely. For example, tank ditch segments are multiples of 30m long and all relocatable objects are treated as instantiatable models, just like entities. However, CCTT's SNE routines are probably more tightly and completely integrated with the abstract representation of the limited CCTT relocatable object set as a result of this extensive system integration.

ModSAF has virtually unlimited dynamic terrain capabilities in that any feature that can be represented can also be changed dynamically. This includes changes made to the terrain skin. Craters, anti-tank ditches, and fighting positions of arbitrary dimensions and can be "sewn in" to the terrain skin.

CCTT supports the relocatable objects (italics = ModSAF does not support):

- *Log crib* (but could be volume feature)
- *Abatis* (but could be volume feature)
- tank ditch - can be composed of up to four segments, with segment lengths of 30, 60, 90, or 120 meters
- concertina wire fence (same as tank ditch)
- fighting positions (fixed geometry in CCTT)
    - infantry fighting position
    - overhead covered infantry position
    - machine gun prepared position
    - covered machine gun bunker

    - hull defilade vehicle fighting position
- minefield
- armored vehicle launched bridge (AVLB)
- building rubble
- *ribbon bridges*

3.2.2.3.6 Environment

Environment includes weather, ocean state, littoral regions (tides), illumination, and environmental effects (such as flares or tactical smoke). The SAFs are overwhelmingly different in these areas.

3.2.2.3.6.1Natural Effects

ModSAF simulates the following weather phenomena: temperature, dewpoint, relative humidity, barometric pressure, wind velocity, precipitation type, precipitation rate, extinction type, extinction amount, extinction coefficient, ray visibility, visual range, illumination, sky over ground, cloud cover, cloud ceiling, cloud height, cloud type, sim time, sun position, moon position, moon phase, contrast, solar illumination, lunar illumination, and sky illumination. ModSAF's implementation of these phenomena is universally more complex and complete than CCTT's.

CCTT SAF represents the following weather phenomena: rain state, visibility range, haze state, haze visibility range, haze ceiling, fog state, fog visibility range, fog ceiling, cloud state, cloud ceiling, cloud base, cloud visibility range, lunar illumination, temperature, humidity, wind speed, and rain soak. CCTT's implementation of each of these is as an ambient, global parameter with no relationship or dependency between each other except as enforced at the system MCC. Each of these parameters is provided as a value that can be retrieved by consumers. Only rain soak has a side effect on SNE services. That is, rain soak modifies the terrain type values passed back upon a surface type query to reflect "wet" surfaces. CCTT has no ocean/littoral capabilities.

3.2.2.3.6.2Man-Made Effects

ModSAF supports the following man-made environmental effects:

- signal and illumination flares
- tactical smoke (from smoke grenades, burning vehicles, and muzzle smoke)
- vehicle dust (dust kicked up by moving vehicles and muzzle blasts)

CCTT SAF supports the following man-made environmental effects:

- flares (treated as point in air with associated circle of light on ground; numbers limited by CCTT system).
- tactical smoke (from smoke munitions; modeled only as simple opacity and type)

3.2.2.3.7 SNE Services

The previous section on interface structure describes the overall approach of each system to presenting an external interface to customers. Also an extensive comparison of interfaces is

provided in the appendix containing the detailed SNE assessment. This section describes functional differences as visible from each system's SNE external interface.

3.2.2.3.7.1SNE Query / Primitives

Terrain-related queries and primitives were found to be surprisingly similar, although there are numerous differences in details partly caused by the use of different language features, such as macros.

Munition flyout: CCTT has a dedicated routine for this capability with some selectable intersection criteria. This capability could be built out of existing libctdb routines.

Collision Detection: ModSAF supports a simple collision detection algorithm which uses the ground intersection service to detect ground or feature intersections along the ray starting with a vehicle's position during the previous tick and ending with that during the current tick.

CCTT SAF provides a more sophisticated collision detection algorithm that utilizes bounding volumes of both vehicles and terrain features. In addition to the entire entity, parts of entities (turrets, hulls, and guns) may individually be checked for collisions.

3.2.2.3.7.2SNE Reasoning

Again, overall SNE reasoning was similar at a gross level. In this case, there were some notable differences.

Observation Posts: CCTT SAF contains an algorithm which searches for observation posts (high elevation points) in a specified area. ModSAF currently does not have this functionality.

Concealed Routes: ModSAF provides a concealed route planning algorithm which finds routes that are as concealed from the enemy as possible. Any number of enemy descriptors may be provided as input to the service. The three types of enemy descriptors are enemy direction, enemy location, and enemy area. CCTT SAF does not compute concealed routes.

Cross-Country Route Planning: ModSAF's cross-country route planner considers only large terrain obstacles such as forests, rivers, lakes, and steeply sloped areas. It also knows about bridges so it can plan through choke points. It currently does not know about minefields and other dynamic countermobility obstacles with the exception of dynamically placed bridges. ModSAF's representation is orthogonal to a Voronoi diagram (travel through midpoints of segments connecting neighboring obstacles). It is used for platoon-level route planning, and returns corridor width information with its output route. ModSAF treats cross-country and road routing as separate planners.

CCTT SAF's cross-country route planner runs as a separate process. It, too, considers large terrain features such as forests, rivers, lakes, no-go areals and steep slopes. The CCTT cross-country planner does not recognize bridges (the user is required to request road points in the route). CCTT will consider selected dynamic obstacles (known minefields, ditches, and fences). CCTT uses a visibility graph from which corridor width cannot easily be extracted. The planner has a validation feature that verifies manually created or modified routes for

trafficability. CCTT provides a single interface to both cross-country and road route planning, but the two planners cannot consider each other's data. The user must still specify each point as cross-country or road; the planner just stitches them together.

For both SAFs, all routing information is built does not, and loaded into memory at initialization. Both also support destruction of intersections in the road network.

Near-Term Planning: ModSAF's near-term entity-level movement planner is a three-dimensional planner (x, y, and time). It considers all obstacles, including other vehicles, trees, and buildings, as well as everything considered by the cross-country planner. Each vehicle has its own "localmap," which describes the obstacles in the vicinity of that vehicle. The path planner then takes these obstacles as input and plans a smooth path through them using cubic splines. Several paths are considered. The optimal (shortest) path is chosen as the planned path. The concept of an "attractive path" is also supported, where a path through an obstacle may be specified. This feature is often used in breaching countermobility obstacles.

CCTT SAF's near-term movement planner is nearly identical to ModSAF's movement planner in terms of supported functionality. CCTT's planner does not include time explicitly (this is handled by CCTT's driver code); thus, dynamic entities are handled only by request from the caller. Also, CCTT's planner makes use of the obstacle/avenue linkages in MrTDB so it can follow roads through steep slope areals, breaches through obstacles, and over/under-passes. These avenues can be selectively blocked by the caller (e.g., do not consider a particular breach through a minefield). CCTT's planner also handles fighting positions / vehicle scrapes as an obstacle or destination based upon the caller's objectives.

3.2.2.3.8 Terrain Tools

This section describes the offline test tools and compilers available for both SAFs.

3.2.2.3.8.1 Compilers

Both systems' compilers use a "front-end" / "back-end". The biggest difference between ModSAF and CCTT database generation are the sources that can be used and CCTT's division into multiple compilers / databases. ModSAF has a separate database for routing, but everything else is handled by CTDB. However, only CTDB is built by a "compiler", because routing databases are constructed from CTDB by ModSAF's runtime code if one does not already exist. CCTT, on the other hand, has three separate "back-ends": one for the dedicated PVD database, one for the dedicated radio database, and one for all other databases (MrTDB for geometry, MrsTDB for routing, and EmTDB for Environment Manager).

There are numerous terrain database sources available for ModSAF. There are compilers that convert from S1000, EaSiest, and Multigen to CTDB. A SEDRIS-to-CTDB compiler is currently under development. There is also a CTDB-to-CTDB compiler that is used to convert from one version of CTDB to another and to incorporate post-processed features. ModSAF's preprocessor analyzes a CTDB database, and compiles a list of all of the terrain skin polygons that exceed a specified steepness threshold. It then coalesces these polygons and outputs the bounding footprints of these steep areas to an ASCII file that is used by the

CTDB-to-CTDB compiler.

CCTT compilers presently ingest only a CCTT-specific format known as SIF++. The CCTT compiler front-end could be replaced to export data from other sources; this approach is being used for an experimental SEDRIS-to-MrTDB compiler. CCTT SAF handles post-processing via multiple "passes" of the compiler, so there is no distinction between post-processing and the compiler passes. CCTT supports steep slopes, forest boundaries, obstacle/avenue linkages, soil defrag, cut and fill, etc., in this manner.

3.2.2.3.8.2Viewers & Debugging Support

ModSAF has a debugging tool that allows one to view all of the terrain skin polygons and features, perform intervisibility checks as well as soil and elevation queries. It also has the capability to view the terrain in 3D.

ModSAF also has a program that lists pertinent information about all of the terrain skin polygons and features in a given database in ASCII format. This is useful for debugging feature encoding algorithms and for comparing the results of successive database compiles.

CCTT also supports all of these capabilities with an interactive Motif tool called GIDGET. The 3D viewer is limited to a static VRML image of a selected portion of the database (<1km x <1km). GIDGET supports all major functions at Environment CSC's interface, including collision detection, relocatable object placement, obstacle avoidance, routing, cover and concealment, etc. To support these routines it can simulate placement of entities and relocatable objects, as well as state changes to buildings and bridges. This allows line of sight, for example, to be tested with all possible features, entities, and relocatable objects in a standalone environment. CCTT's text dump is available using the same point-and-click interface, but only dumps out data for selected feature(s) and/or terrain skin by page or patch (not the entire database).

3.2.3 User Control

Both ModSAF and CCTT SAF are controlled with a suite of graphical user interfaces (GUIs) which form a User Computer Interface (UCI). The capabilities of each system's UCI are described below.

3.2.3.1 General Characteristics

ModSAF supports a single display for both the user input and tactical map. CCTT requires a 2-display workstation configuration. ModSAF employs a tiled window presentation where no windows will obscure other windows. CCTT SAF makes use of many pop-up windows that the operator can position and iconify on his workstation.

3.2.3.2 Tactical Map

For each system, the tactical map displays the terrain features, simulated entities, and simulation effects for the DIS exercise. Both tactical maps support arbitrary zoom levels and panning. ModSAF supports runtime generated contour intervals while CCTT SAF uses pre-generated contour lines for redisplay speed. Each system supports the selective display of

feature classes and forces, and contains tools to display terrain elevation and area intervisibility information. In addition, ModSAF supports the display of terrain profile, line intervisibility between points or entities, and query of the environment or soil information at a given location.

CCTT SAF's tactical map is hosted on a separate display from the main user interface. In addition, the tactical map is part of the Plan View Display (PVD) system, which runs as a separate process from the SAF workstation. The SAF workstation and PVD communicate through a shared memory protocol. The PVD software is a common component shared by many other CCTT systems. ModSAF's tactical map is hosted on the same display as the rest of the ModSAF user interface, and runs as part of the same workstation process.

Both systems display entities using either top-down pictures that show vehicle articulations such as turrets and guns, as well as the ability to display doctrinally correct unit and vehicle symbols using FM-101-5 symbology. CCTT SAF makes extensive use of hard-coded binary images for icons, while ModSAF supports completely data-driven icons and vehicle pictures. ModSAF's vehicle pictures support arbitrary scaling, while CCTT SAF's have a set of fixed sizes.

3.2.3.3 Overlay

Each SAF workstation contains a capability to create tactical overlays containing graphical control measures such as phase lines, coordination points, and assembly areas. CCTT SAF supports a fixed set of 6 doctrinal overlays (Maneuver, Fire Support, etc.), while ModSAF has no predefined overlays other than a default unnamed overlay. Both systems allow the selective display of certain overlays. In each system, the user can create, modify, or delete overlay symbols. ModSAF has more of a direct manipulation interface in which the user can click on a symbol in order to edit it. A specialized library to support on-screen sensitivity of objects on the tactical map provides direct visible feedback as to what objects are currently selectable by the mouse, depending on current editor configuration. In CCTT SAF, the user must select the correct mode before he can click on an overlay symbol, and no highlighting of selectable objects is visible. CCTT SAF has a larger palette of specific tactical overlay symbols, while ModSAF supports more generic symbols such as "line" and "area". Each system can support the separate loading and restoring of overlays separate from the scenario (or exercise) file.

3.2.3.4 Execution Matrix

The control of the behaviors of simulated entities is performed using an execution matrix paradigm. The execution matrix divides a mission into phases, where each phase has one or more unique behaviors for each unit and entity in a tactical organization. Trigger conditions cause the mission to progress from one phase to the next. ModSAF's execution matrix is organized with units on the left column and phases running left to right. CCTT SAF's execution matrix is organized with units at the top row, and mission phases running top to bottom. Developers of the execution matrix of both systems claim Subject Matter Expert (SME) input drove the layout of the matrix.

In CCTT SAF, the cells of the execution matrix contain CISs for the specified unit. The user selects which CIS for each cell based on a filtered list of CISs appropriate for that unit's echelon, side, and type. In CCTT SAF, the execution matrix can be sparse, allowing for a varying number of CISs for a particular unit in a particular mission phase. In ModSAF, the cells of the execution matrix contain task frames for the specified unit. The user selects which task frame for each cell based on a filtered list of task frames appropriate for that unit's echelon, side, and type. In ModSAF, and execution matrix can only have one task frame per unit per mission phase.

Each system provides editors to parameterize the contents of an execution matrix cell. In ModSAF, data-driven editors are provided for each unit-level behavior in a given task frame. The system supports forced choices that must be filled out by the SAF operator before the cell is completed. The operator can modify various default parameters for each task as well. In CCTT SAF, a custom-written editor is provided for each CIS. These editors also support forced choices and the modification of default parameters.

Both systems allow the modification of execution matrix cell parameters (either CISs or Tasks in a task frame) during execution to support Fragmentary Orders (FRAGOs). ModSAF behaviors are required to react correctly to modifications of already running behaviors, while CCTT SAF behaviors are not.

Both systems show the modification of the current mission as an outcome of reactions or situational interrupts. In ModSAF, modifications to the current mission show as colored cells whose textual content shows both the original behavior and current reactive behavior. In CCTT SAF, modifications to the current mission show as colored cells that have been inserted before the currently running cell.

Both systems support an alternate execution matrix that allows the operator to construct a new mission while a unit is executing a current mission.

3.2.3.5 Triggers & Control Measures

In the previous section, it was stated that trigger conditions separate mission phases in an execution matrix. The two systems have different capabilities with respect to the control and specification of these triggers.

Both SAF systems support triggers for completion of a previous mission phase, crossing a control measure, or "on order" from the SAF operator. ModSAF also supports triggering from the duration of the previous mission phase, or from a global h-hour clock. CCTT SAF supports triggering from an absolute time since the beginning of the exercise. Also, CCTT SAF supports triggering from other units' activities, including starting an order, completing an order, or crossing a control measure. CCTT SAF supports the logical AND-ing of multiple trigger conditions. ModSAF's behavior architecture supports a suite of logical operations to combine triggers, such as AND, OR, and NOT, but the user interface does not allow the specification of these logical combinations.

3.2.3.6 Command From Simulator

Both systems provide an interface to the command from simulator behavior described earlier. ModSAF's behavior allows a manned simulator to be substituted into an existing platoon. ModSAF vehicles perform station keeping relative to that manned simulator and will exhibit cue fire, firing only when the manned simulator has fired. The ModSAF operator can control the SAF portions of the platoon with immediate intervention commands, such as change formation, change speed, and halt.

In CCTT SAF, the operator role plays the platoon being commanded by the simulator. The simulator is allowed to replace either the Platoon Leader or Platoon Sgt. position. The operator enables the platoon to fire based on doctrinal rules of engagement, such as hold, fire at sector, and free.

3.2.3.7 Task Organizations and Unit Roles

Both systems make extensive use of a task organization structure to organize military units. In CCTT SAF, the operator has the ability to attach or detach units during scenario construction or during runtime. In both systems, a scheme of vehicle markings and bumper numbers is employed for DIS-based tracking and analysis of exercise vehicles. In CCTT SAF, the vehicle marking scheme is fixed, while in ModSAF, a default scheme is provided and it may be overwritten by the operator. The user has the ability to create task organization templates for later use, and he is able to draw on a library of doctrinal and user created organizations during scenario construction.

Within the CCTT SAF software, behavioral "role" information is derived from the task organization data structures present in the SEOD. This role information is used to group vehicles for particular behavior functions, such as determining the particular vehicles that make up a particular section within a platoon. CCTT SAF behaviors use groups of vehicles or units organized by role as their primary data structure for coordination. The behavioral role abstraction includes capabilities to deal with propagation of roles due to attrition.

Although ModSAF has task organizations described in easily modified data files, no capability to modify task organization during scenario creation is available. Runtime modification of a "Functional Organization" allows temporary modification of the task organization. The user can employ specialized "Report-To" and "Accept" task frames to effect runtime modification of the task organization.

In ModSAF, some behaviors, such as bounding overwatch, make use of temporary functional organizations to coordinate groups of vehicles. The concept of behavioral role is not institutionalized within ModSAF software.

3.2.3.8 User Interface Construction

The two SAF systems have different approaches to construction of the user interface. CCTT SAF uses standardized User Interface Language (UIL) data files to describe windows and widgets. A UIL compiler converts this into a UID file that is loaded by the system to create windows, establish symbolic widget names, and establish symbolic callback names. User

interface developers write software to assign the correct Ada functions to be called when the specified widget is activated. New behavior development may require new user interfaces within the exercise editor to specify CIS parameters. These user interfaces will require new UIL files and modification to user interface software.

ModSAF has built its own runtime GUI construction capability based on data files. A suite of pre-defined widgets is available to ModSAF developers. These widgets, and the data structures that they modify are completely specified in data files that are loaded at runtime. UCI developers in ModSAF need only write software that accesses the data structures modified by the UCI. In the case of behavior development, the behavior developer requires no UCI modifications. A behavior editor infrastructure (libtaskedit) allows the proper plugging in of new task editors into the system without the behavior developer doing anything other than declaring the behavior data structures and widgets in a data file.

3.2.3.9 Specific Windows and Editors

The following describes specific windows found on each system.

CCTT SAF contains a unit editor for view and modification of the task organization and unit parameters (marksmanship, fuel, ammo, opening range, damage level, fire status). Multiple unit editors may be open at one time.

CCTT SAF contains an exercise editor which is used to create and select units, overlays, terrain, and CISs. CIS parameters may be specified from this editor. The user can add and remove units with this editor. During runtime, the exercise editor allows FRAGOs to be inserted into the execution matrix.

CCTT SAF contains a UCI controller which supports pre-exercise terrain selection, and selection of PVD symbol types. From this editor, the user can issue command overrides or modify unit parameters. Also, the vehicle control editor can be used from the UCI controller to provide direct control of a vehicle.

ModSAF supports a user preference editor used to configure the layout of the UCI, as well as the default units to represent information such as speed and distance. A PVD editor allows control of icon sizes on the map, DIS events to be animated (such as shooting or collisions), ranges of contour lines, and the PVD refresh rate.

ModSAF supports a terrain tool which provides a rich suite of terrain analysis functions, such as terrain profile and line and area intervisibility with the terrain and between vehicles. An environmental tool allows querying and modification of the simulated environment, such as temperature, weather, and sea state. An illumination editor allows visualization of the relative contributions to light intensity as well as sun and moon position.

ModSAF has two editors for fire support. A message-based fire support control editor sends appropriate DIS radio messages to artillery units, while another editor allows the arbitrary detonation of artillery at the SAF operator's control (also known as the "bomb button").

ModSAF has a number of editors to modify unit information, such as the rules of engagement editor and the damage editor.

3.2.4 High Level Architecture (HLA)

The ModSAF 4.0 Beta baseline supports the STOW version of the High Level Architecture's Runtime Infrastructure (RTI). ModSAF's HLA integration is regarded as a shallow integration where the ModSAF's internal data model is essentially DIS PDUs and its System Object Model (SOM) simply recasts this DIS data model directly into an HLA object model. This implementation supports the Federation Management, Declaration Management, Object Management, and Data Distribution Management services of the RTI. There is a desire to modify the ModSAF baseline to utilize the HLA standard RTI interface version 1.3. Additionally there is a desire to modify the ModSAF internal data model to be more object-based to more fully satisfy the intent and flexibility provided by HLA compliance. Both of these efforts are currently being implemented under DMSO sponsorship.

The CCTT 5.1.2 SAF baseline does not support the HLA and does not utilize the RTI. However, prototype development to support DMSO's Platform Proto-federation experiment was conducted using the CCTT SAF system. This experiment demonstrated the ability to implement a shallow integration of the HLA into the CCTT SAF architecture using a similar approach as used for ModSAF.

3.3 Migration Assessments

Following the technical characterizations for each category of SAF system capabilities (e.g., Behaviors, Physical Models, etc.), assessments were performed on the migration of the characteristics and capabilities of one SAF system to the other. In most cases, only two migration options were considered:

- Migration of CCTT SAF capabilities into the ModSAF architecture, and
- Migration of ModSAF capabilities into the CCTT SAF architecture.

The exceptions to this were the Behaviors Assessment where additional migration options were assessed that involved architectural extensions, and the User Computer Interface Assessment where only one option is feasible. The two focus teams developed the migration options for a particular category in parallel. Each migration option was developed and analyzed with respect to meeting the intended OneSAF Testbed use and providing the desired OneSAF Testbed characteristics described above in Section 0.

A subsequent set of meetings were held in which each focus team presented their migration option(s), and a joint assessment of the migration options were performed. Each migration option was assessed on the criteria described below. A relative rating, on a scale of 1-10, was provided in each criteria, with 1 being low risk/impact/effect and 10 being high risk/impact/effect.

- Technical feasibility: An assessment of the technical risk involved with the migration option. Considerations included the need for architectural extensions/enhancements, need for major new design, and the ability to implement the option within current technology.

- Program risk: An assessment of the schedule risk associated with the migration option.
- Impact to existing ModSAF community: An assessment of the impact that the implementation of the migration option would present to the community of existing ModSAF developers and users. Considerations included the ability to provide current level of ModSAF capabilities, interfaces (internal and external), extensibility, and performance.
- Impact to ModSAF development: An assessment of the impact that the implementation of the migration option would have on current DoD-sponsored ModSAF development efforts, and the ability of those efforts to be integrated into the OneSAF Testbed.
- Impact to existing CCTT Program: An assessment of the impact that the implementation of the migration option would present to the CCTT user community. Considerations included the ability to provide current level of CCTT SAF capabilities, external interfaces, and performance.
- Impact to CCTT development: An assessment of the impact that the implementation of the migration option would have on current DoD-sponsored CCTT SAF development efforts, and the ability of those efforts to be integrated into the OneSAF Testbed.
- Adaptability to new technology: An assessment of the effect that the implementation of the migration option would have on the ability of new technology to be integrated into the OneSAF Testbed.
- Relative cost: An assessment of the effort to implement the migration option, considering the same assumptions, factors, and requirements, with respect to other migration options within the same category. This criteria was rated in terms of staff months of effort required to implement a migration option in isolation (i.e., without regard to the impact of the migration options selected in other major areas).

The following sections provide an overview of the migration options considered in each major area. The detailed assessments are contained in Appendix C.

3.3.1 Behaviors Assessment

The significance of behaviors within each system cannot be overestimated. Behaviors represent the largest component of each simulation system. This is both in terms of lines of code (SLOC), and effort expended. It was discovered that there were substantial differences between how each system represented behaviors, as identified in the system characterizations previous described. A surprising conclusion by both ModSAF and CCTT SAF behavior developers was that the current state-of-the-art for behavior representation and implementation in each system is sub-optimal today. Developers in both communities see behavior implementation as a Research and Development (R&D) area more than a straightforward development. Each behavior expert could come up with a list of improvements that they felt needed to be developed in order to make behavior implementation more correct, easier to develop, and less error prone.

The results of the behaviors assessment are described in the following sections.

3.3.1.1 Behaviors Focus Group Activity

The behaviors focus group consisted of behavior architects and developers from the ModSAF and CCTT SAF development communities. The group analyzed approaches for satisfying the following requirements:

- Support all currently implemented CCTT CISs
- Retain V&V status for CCTT behaviors as much as possible
- Support all currently implemented ModSAF task frames
- Support capability to easily add behaviors into the system without excessive code re-work

In the course of the assessment, six options were generated for achieving the requirements stated above. The options range from migrating behaviors from one system to the other, to making architectural enhancements to ease migration, to supporting dual architectures, to developing a fresh start. The options can be seen in the next section with the following diagram and the accompanying list.

3.3.1.2 Behaviors Options

Figure 10 shows the six options that were considered for the behaviors assessment.

- Option 1: Re-implement missing behaviors from ModSAF into existing CCTT infrastructure (no modification to the infrastructure)
- Option 2: Re-implement missing behaviors from CCTT SAF into existing ModSAF infrastructure (no modification to the infrastructure)
- Option 3: Develop new unified behavior representation and implement all behaviors in it
- Option 4: Extend ModSAF with additional infrastructure to ease re-implementation of CCTT behaviors within ModSAF
- Option 5: Extend CCTT SAF with additional infrastructure to ease re-implementation of ModSAF behaviors within CCTT SAF
- Option 6: Develop capability to run both ModSAF and CCTT SAF behaviors within the unified SAF

On the left vertical axis are the choices of using one of the two architectures to do a straight implementation of the missing behaviors. These are reasonably well understood options of just developing new behaviors in a native environment. The problems with these options are that, although it may be possible to represent a CCTT behavior in a ModSAF architecture, or vice versa, you will likely lose key system requirements.

Implementing ModSAF behaviors in CCTT (option 1) would sacrifice the ability to easily add new behaviors into the system to, for example, support analysis, without major modifications to many subsystems.

Implementing CCTT behaviors in ModSAF (option 2) would sacrifice assignment-time order decomposition, support for periodic check-pointing for restart, and operational distribution of units across simulation resources (i.e., vehicles in a platoon on different CGF simulators).

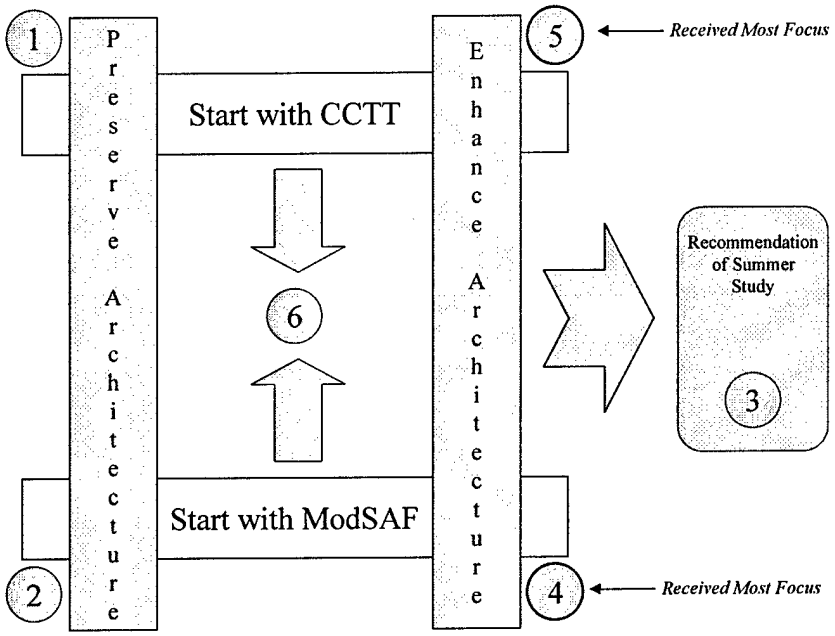

Figure 10: Behaviors Migration Options

If given a choice, both ModSAF and CCTT SAF experts would like to do option 3, which is to start from scratch in some new behavior development technology. This option could be forward-looking to new requirements, such as behavior editing and visibility of behavior algorithms, which are both OneSAF ORD requirements. Because this is a research topic, no candidate approach for option 3 surfaced. The assessment team was not able to judge how long it would take to develop a new behavior architecture, but the team expected that it will take longer than is available for the development of the testbed.

Options 4 and 5 take the approach that if we cannot start new, let us identify what can we do to enhance the exiting architectures to:

1) make the migration cheaper, and

2) make the migration satisfy more requirements

In the case of ModSAF, this includes adding architecture to speed up development, and address some capabilities like order decomposition. In the case of CCTT, this includes making its architecture more extensible.

A final option, option 6, was also considered. This option proposes to somehow support both behavior architectures simultaneously. This would be an attractive option because you could conceptually preserve the V&V characteristics since you would no be re-implementing the CCTT behaviors. The problem with this option was that:

1) The team could not come up with a feasible technical solution to have dual architectures (that does not mean there is not one)
2) The maintenance would be horrific (two sets of behaviors AND two architectures)
3) The option is very unattractive from the point of view of software implementation. Developers would become frustrated trying to extend such a dual system, not having clear guidance as to which architecture to do development in.

With consideration to technical feasibility and the shortness of time to perform the assessments, options 4 and 5 received the most focus.

The detailed assessments for all options are found in Appendix C.

3.3.1.3 Behaviors Assessment Results

The assessment identified a significant risk associated with the sheer size of the behaviors in each system, and the extensive dependencies that behaviors have with respect to the rest of their native software system. Each approach that starts with an existing baseline will experience difficulty in bringing over behaviors from the other baseline. This implies that behavior migration has a substantial per-behavior cost. Each baseline choice has risks of failing to satisfy requirements. In the case of starting with ModSAF, the risks are in the area of validated functionality or usability. In the case of starting with CCTT SAF, the risks are in supporting extensibility, which is a capability that is proven over time.

The following table provides the summary of the assessment results in each criteria area. Note that in all cases, a lower number is better. Except for cost, everything was scored from 1-10. Cost is in person-months.

Options that did not receive primary focus were not fully assessed in all areas such as cost.

	Technical feasibility Risk	Program risk	Impact to existing ModSAF community	Impact to ModSAF development	Impact to existing CCTT program	Impact to current CCTT development	Adaptability to new technology risk	Cost
Option 1	3	2	7	7	1	1	10	260 pm
Option 2	1	7	1	1	7	7	3	425 pm
Option 3	NA	NA	NA	NA	NA	NA	NA	NA
Option 4	2	4	2	2	4	4	3	379 pm
Option 5	4	4	3	3	1	1	4	469 pm
Option 6	9	8	1	8	1	8	7	NA

Table 11: Behavior Migration Assessment

The assessment team focused on options 4 and 5, which were deemed to be the most feasible options that satisfied a full range of requirements. The team focused almost exclusively on cost as the driver, assuming that cost was proportional to schedule, and that schedule was going to be the most difficult item to satisfy. There are, of course, other approaches to supporting combinations of criteria. For example, one could add each criterion together and take the lowest. In this case, you could treat 379 person months as a '3', and 469 person months as a '4'.

Looking at cost, the assessment team settled into choosing option 4, starting with ModSAF. This was not an obvious choice. We cannot say the full range of implications of this choice. Losing the reports and orders paradigm from CCTT may be detrimental to the implementation of CCTT behaviors within ModSAF. The team did not have enough time to evaluate this and other details.

3.3.2 System Assessment

For this system capability assessment, this section describes those priority issues and expectations that were the focus for this particular migration assessment. The system focus area provides the basic framework and architecture for all the application code. In addition it provides all external views and interfaces to the SAF system. The system area considers all over-arching functionality with wide impacts.

3.3.2.1 System Focus Group Activity

The system focus group was responsible for the migration assessment for the system issues for ModSAF and CCTT SAF. The categorization of capabilities to consider was determined

first from the technical characterization taxonomy. From this list, the group developed technical interchange meeting presentations to describe the capabilities of each system. These capabilities were discussed and an agreed list of capabilities was formulated for the system assessment. Two approaches were identified for the migrations are as follows:

- Option 1: Use CCTT SAF System Capabilities as a basis and add the missing system capabilities from ModSAF.
- Option 2: Use ModSAF System Capabilities as a basis and add the missing system capabilities from CCTT SAF.

The detailed assessments for these options are in Appendix C.

3.3.2.2 System Capabilities

The following list outlines the system capabilities considered by this focus group. This list was constructed by the combination of system capabilities through the technical interchange meetings and the technical characterization taxonomy.

- Data Collection
- Architecture
  - Command and Control Protocols (PO vs. SEOD)
  - Allocation and migration
  - Communications/Networking
  - Support for RTI-s
  - Information abstraction
  - Support for bi-endianness
  - Time management
  - Process architecture
- System capabilities
  - CCTT system integration
    - MCC
    - AAR
    - OC Interface
    - Interoperability
  - Concurrent exercises
  - Battalion level workload
  - System configuration
  - User accessibility
  - Developer accessibility
  - Portability
    - Hardware platform
    - Language and programming standards

3.3.2.3 Assumptions

The common goals of the system focus group, independent on which path is taken, are to use C (compilable by C++), not Ada. Also the resulting system must support running on SUN, SGI, PC (Linux), Dec (Alpha), and IBM/Motorola (AIX) systems.

3.3.2.4 Assessment Results

The assessment illustrated a significant risk associated with exercise initialization for CCTT. This presented technical risks in the translation of the SEOD file format and CGF protocol as well as schedule risk necessary to mitigate the technical risk. This risk was shared by both options. The assessment recommended Option 2 (start with ModSAF). However, it was noted that threading, as it is implemented in CCTT SAF, must be migrated to the ModSAF system approach for performance. Further analysis must be performed regarding the migration of SEOD capabilities to the PO as well as the system interfaces inherent in CCTT. The following table provides the summary of the results. Refer to the system assessments in Appendix C for a more detailed description of each option.

	Technical Feasibility Risk	Program Risk	Impact to Existing ModSAF community	Impact to ModSAF Development	Impact to current CCTT program	Impact to current CCTT development	Adaptabilit y to new technology risk	Relative cost
Option 1	3	2	2	2	5	2	5	42.5 pm
Option 2	4	4	2	2	5	2	3	40.5 pm

Table 12: System Migration Assessment

3.3.3 Synthetic Natural Environment Assessment

The SNE assessment examined the feasibility of using the SNE software of either CCTT SAF (Option 1) or ModSAF (Option 2) as a baseline and incorporating unsupported capabilities from the other system. The SNE assessment considered not only runtime capabilities, but also runtime databases, compilers, and correlation with other systems, such as image generators.

The SNE assessment was made easier by the degree of technical exchange between the CCTT and ModSAF communities in SNE over the past few years, including the widely publicized SNE improvements as part of STOW and various ModSAF efforts to absorb CCTT's SNE extensions into the ModSAF baseline.

3.3.3.1 Assumptions

This assessment assumed that the final system is to be implemented in C (compilable by C++) and that the resultant SNE must contain the union of CCTT and ModSAF SNE capabilities. Also, the terrain database sources and feature content of both systems must be supported. Finally, the boundary with behaviors was drawn such that behavioral support utilities (like path planning) were deemed part of SNE. A detailed list of each system's components included with SNE is provided in the appendix.

3.3.3.2 Assessment Highlights

In general, it was discovered that the runtime capabilities and databases were very close on the terrain side to the extent that requirements overlapped; this is not surprising, since the two

systems share a common starting point in database format and CCTT reused a number of libctdb routines. Despite this, the biggest overall factor discovered during the assessment was the difference in breadth of functional capabilities, as illustrated in the technical characterization and detailed assessment. CCTT's SNE representation does not contain a number of capabilities that ModSAF/STOW has, including:

- advanced terrain skin topology
- multiple elevation structures (i.e. building interiors)
- DVW (dynamic virtual worlds; phenomenology effects not found in CCTT)
- DTO (dynamic terrain object capabilities partly unsupported by CCTT)
- GCS (global coordinate system)

These capabilities are extensive enough to discourage use of CCTT as a baseline because of the volume of software that would have to be re-engineered, especially in light of the need to convert CCTT software to C.

The greatest advantage of using the CCTT software as a baseline is the investment in integration with the CCTT system as a whole (including manned modules, visual systems, and user-specific requirements and user-directed performance modifications). However, the combination of an Ada-to-C conversion, plus the integration of significant functionality from ModSAF means that significant effort would still be required with CCTT interoperability and correlation.

Regardless of the approach taken with the runtime functionality and databases, the database generation software would need to be based on ModSAF's compiler, which already handles a wide range of database sources. Although a CCTT database has been built in CTDB format, there is likely to be substantial work in this area to build a repeatable, production-capability CCTT database generation capability from the ModSAF compiler baseline.

3.3.3.3 Results

Three significant risks were identified during this assessment:

- CCTT system impact (interoperability, fair fight, user performance directed modifications)
- Generation of CCTT databases (availability of database compilers)
- Numerous specific technical risks and areas of further study (enumerated in the detailed assessment)

The summary of the options is shown in Table 13, while the detailed assessment can be found in Appendix C. The SNE assessment strongly supports the use of ModSAF as a baseline for OneSAF testbed (option 2), largely because it contains substantially more functional breadth than CCTT and is already implemented in C.

	Technical Feasibility Risk	Program Risk	Impact to Existing ModSAF Community	Impact to ModSAF Development	Impact to Existing CCTT Program	Impact to Current CCTT Development	Adaptability to New Technology Risk	Cost
Option 1	8	7	5	5	5	5	5	132 pm

Option 2	3	3	3	2	8	8	3	103 pm

Table 13: SNE Migration Assessment

3.3.4 Physical Models Assessment

The physical models focus group activity involved three steps. First, the group baselined the physical model capabilities for CCTT SAF and ModSAF. Then they analyzed the approaches for the combination of capabilities. Based on the initial analysis, two options were generated for consideration.

During the initial analysis efforts, the group recognized the following significance related to both physical models. Both systems have a similar classification structure for the physical models. This obviously eased the assessment task. This characteristic automatically leads to low technical risks, low program risk, and minimal program impacts. Detailed descriptions about these risk and program impacts can be found in Appendix C. The group also identified that CCTT models have the most recent V&V status because of its daily use requirement for training, while ModSAF's physical model architecture provides better extensibility. The extensibility of ModSAF comes directly from extensive usage of function pointer registration scheme that is characterized as inverted architecture.

The same two options were considered for physical model assessment as was consider for the other categories. Option 1 starts with the ModSAF baseline and incorporates CCTT models within the ModSAF physical model architecture. Options starts with the CCTT SAF baseline and incorporates ModSAF models within the CCTT SAF physical model architecture.

The focus group's goal, independent of either migration path of the above options, was to construct a model suite that has the best combination of performance, validity, usability, extensibility, and commonality. Thus, the approach envisioned by the group for both options was the following. Whenever possible, the two models would be unified. However, if the unification will not be possible, two models would be implemented in the target OneSAF testbed. Finally, the C programming language was chosen as the implementation language regardless of the options.

Option 1 maximizes the extensibility of ModSAF while implementing the CCTT physical models. Thus, the resultant system will be easier to construct than that of Option 2. However, the V&V process would be more involved because the CCTT models will be re-implemented on top of the ModSAF physical model architecture. There will be a risk that the re-implemented CCTT models might not perform as before in the native CCTT architecture. The level of this risk should be minimal. After examining all of the CCTT models and investigating ModSAF architecture for estimation of the re-implementation task, the assessment group computed 34 relative person months in order to complete the task of unifying physical models of the both systems using option 1.

Option 2 has an advantage that the CCTT requirements are automatically met by the direct use of the CCTT models in their native architecture. The V&V process is minimized for the CCTT physical models. However, in order to re-implement the ModSAF physical models, the CCTT physical model architecture will have to be extended to include the ModSAF's

inverted architecture. Therefore, this option was determined to be more costly than Option 1. Additionally, all of the CCTT SAF Ada code will have to be translated into C language. When all of these cost items are added, the cost associated with unifying the physical models of two systems was estimated as 50 relative person months using option 2. This option carries a slightly higher risk than Option 1 because of the architecture extension for re-implementing the ModSAF models.

The summary of the above two options are shown in Table 14 while the detailed assessment can be found in Appendix C.

	Technical feasibility risk	Program risk	Impact to existing ModSAF community	Impact to ModSAF development	Impact to existing CCTT program	Impact to current CCTT development	Adaptability to new technology risk	Cost
Option1	1	1	1	1	2	2	2	34 pm
Option2	3	3	2	4	1	1	2	50 pm

Table 14: Physical Model Assessment

3.3.5 User Computer Interface Assessment

The significance of the UCI assessment is due to its visibility. The ModSAF and CCTT SAF UCIs are the first interface that any SAF user encounters when using these systems. Therefore, imperfections in the UCI migration will have immediate visibility to the OneSAF testbed development program.

3.3.5.1 UCI Focus Group Activities

The UCI focus group was responsible for the migration assessment for the human interface issues for ModSAF and CCTT SAF. The categorization of capabilities to consider was determined first from the technical characterization taxonomy. From this list, the group developed technical interchange meeting presentations to describe the capabilities of each system. These capabilities were discussed and an agreed list of capabilities was formulated for the UCI assessment.

During the categorization of capabilities, it was recognized that there were driving capabilities that must be incorporated in the final system. It was identified that

- The UCI be C based,
- Window/widget generation be data file driven,
- The data model supports data file descriptions of offsets, and
- The architecture needs to be inverted to support extension.

Each of these requirements for extension and portability directly conflict with the CCTT UCI implementation and architecture. The CCTT assessment team determined that incorporation of ModSAF capabilities would require a complete rework of the CCTT UCI. It was further determined that this rework would be essentially equivalent to new development. This was based on the fact that only the requirements and screen content could be reused from the

CCTT approach. This is true because the UCI architecture and implementation, hard code the structure of the models and their data as well as the functionality of the UCI. The UCI has minimal data driven capabilities limited to the pure unit organizations, behavior assignment lists, and data ranges. Even with this level of data driven capabilities, the contents of the data files are directly driven by the code structure. Any additions to the UCI required both data file changes and code changes. Incorporating this design in the OneSAF Testbed would constitute a step backwards when compared to the data driven capabilities of ModSAF (specifically, all window contents and attributes are data driven). It was determined that the result of a new development from CCTT SAF would be equivalent to an evolved ModSAF UCI. Therefore, this focus group reasoned that the only viable option was an evolution of the ModSAF baseline.

The detailed assessment for this option is in Appendix C.

3.3.5.2 UCI Capabilities

The following capabilities were considered during this focus group.

- Main window
- Exercise Editor
- Unit Editor
- Unit status window
- Report window
- Alert windows
- Vehicle control editor
- PVD capabilities
  - Tactical map
  - Overlay editor

3.3.5.3 Assumptions

The following assumptions were generated for this assessment activity.

- PVD subsystem from CCTT must be preserved
  - Common component used extensively throughout CCTT systems
  - Implies that CCTT PVD, including tactical map and overlay symbols must be present in OneSAF testbed
- Both CCTT's popup oriented and ModSAF's tiled approach must be maintained

It was identified that a new interface and supporting architecture would be added to ModSAF to allow communications between the ModSAF application and the CCTT PVD application. The ModSAF PVD would be adjusted to allow running with or without the PVD (where PVD refers to everything that is displayed on the "map"). When use of the CCTT PVD is desired, the ModSAF PVD would be disabled and ModSAF would switch to using the new interface to communicate with the CCTT PVD using the CCTT PVD API.

As shown in Figure 11, a new interface and architecture would have to allow ModSAF to communicate via the existing CCTT PVD API shared-memory message queue. There will be

issues with an Ada application sharing a shared-memory segment with a C application, the most obvious being "data understanding" (C and Ada do not use the same memory layouts for things like structures and arrays, etc.). It is assumed that these issues can be overcome with reasonable amounts of effort.

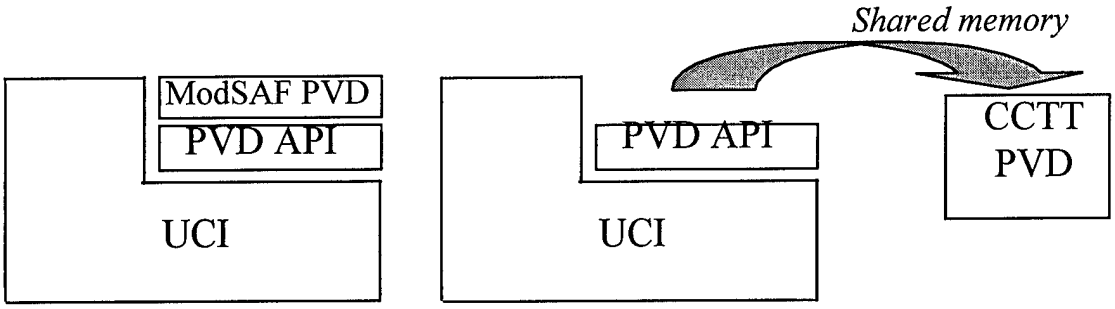

Figure 11: OneSAF Testbed Interface to CCTT PVD

Once the architecture is enhanced, the appropriate function calls and hooks will have to be added to ModSAF (in similar fashion to where all the existing ModSAF PVD functions and hooks). ModSAF Editors that currently use the ModSAF PVD for input (or output), would have to be modified to understand that a ModSAF PVD might not exist, and if appropriate take their input from the CCTT PVD (via the CCTT PVD API).

3.3.5.4 Assessment Results

Table 15 shows the result of the UCI migration assessment. With only one choice available, there is no utility in comparing this assessment with other assessments.

	Technical feasibility Risk	Program risk	Impact to existing ModSAF community	Impact to ModSAF development	Impact to existing CCTT program	Impact to current CCTT development	Adaptabilit y to new technology risk	Cost
Option 1	4	1	1	4	3	2	2	38pm

Table 15: UCI Migration Assessment

It should be noted that though this is the best and only choice within the constraints of this study, this recommendation does not address the potential advantages that could be afforded by performing new development using new UCI technologies. One example would be the development of Java-based UCI capabilities that would allow the exploitation of web-based deployment of the SAF UCI. An approach such as this might increase the usability and accessibility of the SAF for a variety of uses. This approach was not evaluated due to an assumption that new development would be out of the scope of the initial testbed development.

3.3.6 Migration Summary

Table 16 provides a rollup, exclusive of the User Computer Interface assessment, of the various migration assessments. The relative costs shown in the table are not meant to show

the total effort required to implement the OneSAF Testbed; rather, they are meant to provide a relative comparison of the efforts associated with selecting either CCTT SAF or ModSAF as the starting baseline.

Starting Baseline	CCTT SAF	ModSAF
Technical Feasibility Risk	18	10
Program Risk	16	12
Impact to Existing ModSAF Community	12	10
Impact to ModSAF Development	14	10
Impact to Existing CCTT Program	12	16
Impact to CCTT Development	9	16
Adaptability to New Technology	16	13
Relative Cost	694	557

Table 16: Migration Assessment Rollup

Given that data-driven extensibility and a C language base (compilable in C++) are the key characteristics of the OneSAF Testbed, the following is recommended:

- The starting baseline for the OneSAF Testbed will be ModSAF 4.0
- CCTT SAF models will be selectively re-engineered in C from the CCTT SAF code (not from the models' requirements)
- Extensions to the ModSAF architecture and models will be made to capture strengths from CCTT SAF

This recommended approach achieves the highest priority OneSAF Testbed characteristics with a low degree of risk:

- A similar capacity for "Plug N Play" as provided by CCTT SAF and ModSAF
- A capacity to be extended toward providing a Battalion level behavior API
- A capacity to support R&D experimentation associated with the OneSAF Objective Architecture using current SAF capabilities

The recommended approach will also aim to achieve the lower priority OneSAF Testbed characteristics, but with a higher degree of risk associated with achieving these characteristics:

- A breadth of modeling capability as provided by CCTT SAF and ModSAF
- A training capability as provided by CCTT SAF and ModSAF

4 Summary

This document described the analysis effort and its artifacts for recommending the baseline SAF from which the OneSAF Testbed should be developed. These artifacts include a description of the technical characteristics of ModSAF and CCTT SAF. The OneSAF Testbed will be used to support research and development for the next generation OneSAF

architecture experiments, will be extended toward providing a BBS replacement capability through a Battalion level behavior API, and will provide the training capacity currently implemented by CCTT SAF. This analysis recommended the development of the OneSAF Testbed by integrating the desired CCTT SAF characteristics into the ModSAF baseline. However, enough technical detail and assessment artifacts are provided so that should a different prioritization for the OneSAF Testbed characteristics or different evaluation criteria weighting be desired, the reader should be able to reevaluate this recommendation.

5 References

[1] "One Semi-Automated Forces (OneSAF) Requirements Document", OneSAF Integrated Concept Team Requirements Subgroup, August 22, 1997.

[2] "OneSAF Analysis, Final Report", Document Number, SAIC-98/7768&00, Science Application International Corporation, 12479 Research Parkway, Orlando, FL 32826, September 26, 1997.

6 Acronym List

A2ATD	Anti-Armor Advanced Concept Demonstration
AAFSM	Asynchronous Augmented Finite State Machine
AAR	After Action Review
ACR	Advanced Concepts and Requirements
ACTD	Advanced Concept Technical Demonstration
AMSAA	Army Materiel Systems Analysis Activity
AMSEC	Army Model and Simulation Executive Council
API	Applications Programmer Interface
ASCII	American Standard Code for Information Interchange
ASTAMIDS	Airborne Standoff Minefield Detection System
AVLB	Armored Vehicle Launched Bridge
BBS	Brigade and Battalion Simulation
BDA	Battle Damage Assessment
BEWSS	Battlefield Environment Weapon System Simulation
BIT	Built-in Test
BLUFOR	Blue Force
C2	Command and Control
CAS	Close Air Support
CCTT	Close Combat Tactical Trainer
CFOR	Command Forces
CFS	Command From Simulator
CGF	Computer Generated Forces
CIS	Combat Instruction Set
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CTDB	Compact Terrain Database
DAR	Data Analysis Report
DARPA	Defense Advanced Research Projects Agency
DI	Dismounted Infantry
DI WS	Dismounted Infantry Workstation
DIS	Distributed Interactive Simulation
DMSO	Defense Modeling and Simulation Office
DMSTIAC	Defense Modeling, Simulation and Technology Information Analysis Center
DoD	Department of Defense
DRA	Dead Reckoning Algorithm
DRC	Daily Readiness Check
DT	Dynamic Terrain

DTO	Dynamic Terrain Objects
DVW	Dynamic Virtual World
E&S	Evan and Sutherland
ECN	Engineering Change Notice
EM	Environment Model
EmTDB	Environment Model's Terrain Database
FM	Field Manual
FRAGO	Fragmentary Order
FSM	Finite State Machine
FWA	Fixed Wing Air
GCS	Global Coordinate System
GIDGET	Graphical Interactive Database General Evaluation Tool
GMI	Generic Model Interface
GUI	Graphical User Interface
HAB	Heavy Assault Bridge
HLA	High Level Architecture
ICT	Integrated Concept Team
IDE	Integration Development Environment
IDT	Integrated Development Team
IP	Internet Protocol
ITEMS	Interactive Tactical Environment Management System
IV&V	Independent Verification and Validation
JCM	Joint Countermine
JCOS	Joint Countermine Operations
JTS	Joint Tactical Simulation
LOS	Line Of Sight
M&S	Modeling and Simulation
MCC	Master Control Console
MES	Multiple Elevation Structures
MGRS	Milgrid
MHz	megahertz
MICLIC	Mine Clearing Line Charge
MM	Manned Module
MMCM	Magnetic Mine Counter Measure
ModSAF	Modular Semi-Automated Forces
MrsTDB	Multi-level Routing Support Terrain Database
MrTDB	Model Reference Terrain Database

MTBF	Mean Time Before Failure
NFS	Network File System
NTP	Network Time Protocol
OC	Operations Center
OC WS	Operations Center Work Station
OPFOR	Opposing Force
ORD	Operational Requirements Document
ORSMC	Off-Route Smart Mine Clearing
PDSS	Post Deployment Software Support
PDU	Protocol Data Unit
POP	Persistent Object Protocol
POST	Power On Self Test
PVD	Plan View Display
R&D	Research and Development
RAM	Random Access Memory
RDA	Research, Development and Acquisition
RTI	Runtime Infrastructure
RWA	Rotary Wing Air
SAF	Semi-Automated Forces
SAF WS	Semi-Automated Forces Work Station
SEDRIS	Synthetic Environments Data, Representation and Interchange Specification
SEOD	Semi-Automated Forces Entity Object Database
SI	Situational Interrupts
SIF	Standard Interchange Format
SLOC	Statement Lines Of Code
SME	Subject Matter Expert
SMP	Symmetric Multi-Processing
SNE	Synthetic Natural Environment
SOM	Simulation Object Model
STOW	Synthetic Theater Of War
STRICOM	Simulation Training Instrumentation Command
TAOS	Total Atmospheric and Ocean Server
TEMO	Training, Exercise, Military Operations
TIN	Triangulated Irregular Networks
TMD	Theater Missile Defense
TO	Task Organization
TOC	Tactical Operations Center
TWSTIAC	Tactical Warfare Simulation and Technology Analysis Center

UCI	User Computer Interface
UID	User Interface Data
UIL	User Interface Language
UTM	Universal Transverse Mercator
VV&A	Validation, Verification and Accreditation

ADST-II-CDRL-ONESAF-98000101

Appendices

Appendix A – CCTT SAF and ModSAF Behaviors

Appendix B – CCTT SAF and ModSAF Entities

Appendix C – Migration Assessment Detailed Data

Appendix A - SAF Behaviors

This appendix contains the collection of CCTT SAF and ModSAF behaviors that are currently implemented in the CCTT SAF 5.1.2 and ModSAF 4.0 Beta versions.

A.1 CCTT SAF Behaviors

The following table lists the set of orders that are implemented within CCTT SAF. This table identifies whether the order is a BLUFOR or OPFOR order, which echelon and unit type the order controls, and whether the order is assignable by the SAF operator.

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Battalion	BLUFOR_ARMORED_CAVALRY_SQUADRON	Assignable	SUBORDINATE_LEVEL_TASKING
BLUFOR	Battalion	BLUFOR_ARMORED_CAVALRY_SQUADRON	Not_Assignable	CONDUCT_MOVEMENT_TO_CONTACT
BLUFOR	Battalion	BLUFOR_ARMORED_CAVALRY_SQUADRON	Not_Assignable	CONTINUE_MISSION
BLUFOR	Battalion	BLUFOR_ARMORED_CAVALRY_SQUADRON	Not_Assignable	EXECUT_TACTICAL_ROAD_MARCH
BLUFOR	Battalion	BLUFOR_ARMORED_CAVALRY_SQUADRON	Not_Assignable	EXECUTE_DELAY
BLUFOR	Battalion	BLUFOR_ARMORED_CAVALRY_SQUADRON	Not_Assignable	EXECUTE_HALT
BLUFOR	Battalion	BLUFOR_ARMORED_CAVALRY_SQUADRON	Not_Assignable	IDLE
BLUFOR	Battalion	BLUFOR_ARMORED_CAVALRY_SQUADRON	Not_Assignable	OCCUPY_ASSEMBLY_AREA
BLUFOR	Battalion	BLUFOR_ARMORED_CAVALRY_SQU	Not_Assignable	RESUME

Side	Echelon	Unit	User Assignable	Order
		ADRON		
BLUFOR	Battalion	BLUFOR_ARMORED_CAVALRY_SQUADRON	Not_Assignable	TRM_HALT
BLUFOR	Battalion	BLUFOR_BATTALION	Assignable	SUBORDINATE_LEVEL_TASKING
BLUFOR	Battalion	BLUFOR_BATTALION	Not_Assignable	BREACH_DEFENDED_OBSTACLES
BLUFOR	Battalion	BLUFOR_BATTALION	Not_Assignable	CONTINUE_MISSION
BLUFOR	Battalion	BLUFOR_BATTALION	Not_Assignable	EXECUTE_HALT
BLUFOR	Battalion	BLUFOR_BATTALION	Not_Assignable	IDLE
BLUFOR	Battalion	BLUFOR_BATTALION	Not_Assignable	MOVE_TACTICALLY
BLUFOR	Battalion	BLUFOR_BATTALION	Not_Assignable	OCCUPY_ASSEMBLY_AREA
BLUFOR	Battalion	BLUFOR_BATTALION	Not_Assignable	PERFORM_TACTICAL_ROAD_MARCH
BLUFOR	Battalion	BLUFOR_BATTALION	Not_Assignable	RESUME
BLUFOR	Battalion	BLUFOR_BATTALION	Not_Assignable	TRM_HALT
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Cancel_Check_Fire_Operations
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Cancel_Fire
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Check_Fire
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Dispatch
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Extend_Tent
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Fire
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Halt
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Idle

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	React_To_Air_Attack
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	React_To_Ground_Attack
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	React_To_Indirect_Fire
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	React_To_Minefield
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	React_To_Obstacle
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Perform_Resupply_Operations
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Perform_Resupply
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Terminate_Mission
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Request_Repair
BLUFOR	Battalion	BLUFOR_Mortar	Assignable	Stow_Tent
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Assignable	BOUNDING_OVERWATCH
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Assignable	EXECUTE_DELAY
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Assignable	EXECUTE_HALT
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Assignable	EXECUTE_TRAVELING
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Assignable	OCCUPY_ASSEMBLY_AREA
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Assignable	PERFORM_PASSAGE_OF_LINES
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Assignable	PERFORM_RESUPPLY_OPERATIONS

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Assignable	SCREEN_MOVEMENT
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Assignable	SUBORDINATE_LEVEL_TASKING
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Assignable	TACTICAL_ROAD_MARCH
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Assignable	TRAVELING_OVERWATCH
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	ACTIONS_AT_DEFILE
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	CONDUCT_MOVING_SCREEN
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	CONDUCT_STATIONARY_SCREEN
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	CONTINUE_MISSION
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	EXECUTE_ACTIONS_ON_CONTACT
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	HASTY_OBSTACLE_BREACHING
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	IDLE
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	PERFORM_ZONE_RECON
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	REACT_TO_INDIRECT_FIRES

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	RESUME
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
BLUFOR	Company	BLUFOR_ARMORED_CAVALRY_TROOP	Not_Assignable	TRM_HALT
BLUFOR	Company	BLUFOR_Artillery	Assignable	Cancel_Check_Fire_Operations
BLUFOR	Company	BLUFOR_Artillery	Assignable	Cancel_Fire
BLUFOR	Company	BLUFOR_Artillery	Assignable	Change_Ammunition_Load
BLUFOR	Company	BLUFOR_Artillery	Assignable	Check_Fire
BLUFOR	Company	BLUFOR_Artillery	Assignable	Dispatch
BLUFOR	Company	BLUFOR_Artillery	Assignable	Fire
BLUFOR	Company	BLUFOR_Artillery	Assignable	Halt
BLUFOR	Company	BLUFOR_Artillery	Assignable	Idle
BLUFOR	Company	BLUFOR_Artillery	Assignable	Move
BLUFOR	Company	BLUFOR_Artillery	Assignable	Move_To_Alternate_Position
BLUFOR	Company	BLUFOR_Artillery	Assignable	React_To_Air_Attach
BLUFOR	Company	BLUFOR_Artillery	Assignable	React_To_Ground_Attack
BLUFOR	Company	BLUFOR_Artillery	Assignable	React_To_Indirect_Fire
BLUFOR	Company	BLUFOR_Artillery	Assignable	React_To_Minefield
BLUFOR	Company	BLUFOR_Artillery	Assignable	React_To_Obstacle
BLUFOR	Company	BLUFOR_Artillery	Assignable	Refuel

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Company	BLUFOR_Artillery	Assignable	Resume
BLUFOR	Company	BLUFOR_Artillery	Assignable	Resupply
BLUFOR	Company	BLUFOR_Artillery	Assignable	Terminate_Mission
BLUFOR	Company	BLUFOR_ASSAULT_HELO_COMPANY	Assignable	CONDUCT_AIR_ASSAULT_OPERATIONS
BLUFOR	Company	BLUFOR_ASSAULT_HELO_COMPANY	Assignable	EXECUTE_TRAVELING
BLUFOR	Company	BLUFOR_ASSAULT_HELO_COMPANY	Not_Assignable	ACTIONS_ON_CONTACT
BLUFOR	Company	BLUFOR_ASSAULT_HELO_COMPANY	Not_Assignable	CONTINUE_MISSION
BLUFOR	Company	BLUFOR_ASSAULT_HELO_COMPANY	Not_Assignable	HALT
BLUFOR	Company	BLUFOR_ASSAULT_HELO_COMPANY	Not_Assignable	IDLE
BLUFOR	Company	BLUFOR_ASSAULT_HELO_COMPANY	Not_Assignable	RESUME
BLUFOR	Company	BLUFOR_ASSAULT_HELO_COMPANY	Not_Assignable	SUBORDINATE_LEVEL_TASKING
BLUFOR	Company	BLUFOR_ATTACK_HELO_COMPANY	Assignable	ENGAGE_TARGETS
BLUFOR	Company	BLUFOR_ATTACK_HELO_COMPANY	Assignable	ENGAGE_TARGETS_RUNNING
BLUFOR	Company	BLUFOR_ATTACK_HELO_COMPANY	Assignable	EXECUTE_TRAVELING
BLUFOR	Company	BLUFOR_ATTACK_HELO_COMPANY	Not_Assignable	ACTIONS_ON_CONTACT
BLUFOR	Company	BLUFOR_ATTACK_HELO_COMPANY	Not_Assignable	CONTINUE_MISSION

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Company	BLUFOR_ATTACK_HELO_COMPANY	Not_Assignable	COUNTERMEASURES
BLUFOR	Company	BLUFOR_ATTACK_HELO_COMPANY	Not_Assignable	HALT
BLUFOR	Company	BLUFOR_ATTACK_HELO_COMPANY	Not_Assignable	IDLE
BLUFOR	Company	BLUFOR_ATTACK_HELO_COMPANY	Not_Assignable	RESUME
BLUFOR	Company	BLUFOR_ATTACK_HELO_COMPANY	Not_Assignable	SUBORDINATE_LEVEL_TASKING
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	ASSAULT_AN_ENEMY_POSITION
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	ATTACK_POSITION_ACTIVITIES
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	BREACH_AN_OBSTACLE
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	CONSOLIDATE_ON_OBJECTIVE
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	DEFEND
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	EXECUTE_HALT
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	OCCUPY_ASSEMBLY_AREA
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	PERFORM_ASSAULT_POSITION_ACTIVITIES
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	PERFORM_TACTICAL_MOVEMENT
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	REORGANIZE_ON_OBJECTIVE
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	SERVICE_STATION_RESUPPLY
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	SUBORDINATE_LEVEL_TASKING
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	SUPPORT_BY_FIRE
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Assignable	TACTICAL_ROAD_MARCH
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	ACTIONS_ON_CONTACT
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	ATTACK_POSITION_HALT

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	BOUNDING_OW
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	BOUNDING_OW_1
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	BOUNDING_OW_2
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	CONTINUE_MISSION
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	DEFEND_AGAINST_AIR_ATTACK
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	EXECUTE_ACTION_DRILL
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	HASTY_RIVER_GAP_CROSSING
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	IDLE
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	REACT_TO_INDIRECT_FIRES
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	REACT_TO_REINFORCED_OBSTACLE
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	TRAVELING
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	TRAVELING_OW
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	TRAVELING_OW_1
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	TRAVELING_OW_2
BLUFOR	Company	BLUFOR_COMPANY_TEAM	Not_Assignable	TRM_HALT
BLUFOR	Company	BLUFOR_Engineering	Assignable	React_To_Obstacle
BLUFOR	Company	BLUFOR_Engineering	Assignable	React_To_Minefield
BLUFOR	Company	BLUFOR_Engineering	Assignable	React_To_Ground_Attack
BLUFOR	Company	BLUFOR_Engineering	Assignable	React_To_Indirect_Fire
BLUFOR	Company	BLUFOR_Engineering	Assignable	React_To_Air_Attack
BLUFOR	Company	BLUFOR_Engineering	Assignable	Request_Repair

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Company	BLUFOR_Engineering	Assignable	Move
BLUFOR	Company	BLUFOR_Engineering	Assignable	Halt
BLUFOR	Company	BLUFOR_Engineering	Assignable	Hitch
BLUFOR	Company	BLUFOR_Engineering	Assignable	Unhitch
BLUFOR	Company	BLUFOR_Engineering	Assignable	Dispatch
BLUFOR	Company	BLUFOR_Engineering	Assignable	Terminate_Mission
BLUFOR	Company	BLUFOR_Engineering	Assignable	Idle
BLUFOR	Company	BLUFOR_Engineering	Assignable	Perform_Resupply_Operations
BLUFOR	Company	BLUFOR_Engineering	Assignable	Attach_To_Unit
BLUFOR	Company	BLUFOR_Engineering	Assignable	Detach_From_Unit
BLUFOR	Company	BLUFOR_Engineering	Assignable	Construct_Relocatable
BLUFOR	Company	BLUFOR_Engineering	Assignable	Breach_Or_Clear_Obstacle
BLUFOR	Company	BLUFOR_Engineering	Assignable	Construct_Minefield
BLUFOR	Company	BLUFOR_Engineering	Assignable	Breach_Or_Clear_Minefield
BLUFOR	Company	BLUFOR_Engineering	Assignable	Construct_Fighting_Positions
BLUFOR	Company	BLUFOR_Engineering	Assignable	Deploy_Avlb
BLUFOR	Company	BLUFOR_Engineering	Assignable	Retrieve_Avlb
BLUFOR	Company	BLUFOR_Engineering	Assignable	Transfer_Mclic
BLUFOR	Company	BLUFOR_Engineering	Assignable	Mark_Minefield
BLUFOR	Company	BLUFOR_Engineering	Assignable	Construct_Ribbon_Bridge
BLUFOR	Company	BLUFOR_Engineering	Assignable	Destroy_Bridge

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Company	BLUFOR_Support	Assignable	Move
BLUFOR	Company	BLUFOR_Support	Assignable	Halt
BLUFOR	Company	BLUFOR_Support	Assignable	Dispatch
BLUFOR	Company	BLUFOR_Support	Assignable	Terminate_Mission
BLUFOR	Company	BLUFOR_Support	Assignable	Idle
BLUFOR	Company	BLUFOR_Support	Assignable	React_To_Ground_Attack
BLUFOR	Company	BLUFOR_Support	Assignable	React_To_Indirect_Fire
BLUFOR	Company	BLUFOR_Support	Assignable	React_To_Air_Attack
BLUFOR	Company	BLUFOR_Support	Assignable	React_To_Minefield
BLUFOR	Company	BLUFOR_Support	Assignable	React_To_Obstacle
BLUFOR	Company	BLUFOR_Support	Assignable	Request_Repair
BLUFOR	Company	BLUFOR_Support	Assignable	Move_Prestock
BLUFOR	Company	BLUFOR_Support	Assignable	Unload_Prestock
BLUFOR	Company	BLUFOR_Support	Assignable	Scheduled_Resupply
BLUFOR	Company	BLUFOR_Support	Assignable	Emergency_Resupply
BLUFOR	Company	BLUFOR_Support	Assignable	Authorize_Refuel
BLUFOR	Company	BLUFOR_Support	Assignable	Transfer_Supply
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	Authorize_Refuel
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	Dispatch
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	Extend_Tent
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	Halt

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	Idle
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	Move
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	Perform_Resupply_Operations
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	React_To_Air_Attack
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	React_To_Ground_Attack
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	React_To_Indirect_Fire
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	React_To_Minefield
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	React_To_Obstacle
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	Request_Repair
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	Stow_Tent
BLUFOR	Hq	BLUFOR_Command_Post	Assignable	Terminate_Mission
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Assignable	DISMOUNT_VEHICLE
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Assignable	EXECUTE_HALT
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Assignable	EXECUTE_TRAVELING
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Assignable	MOUNT_VEHICLE
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Assignable	OCCUPY_FIRING_POSITION
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Not_Assignable	CONTINUE_MISSION
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Not_Assignable	IDLE
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Not_Assignable	OCCUPY_ASSEMBLY_AREA
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Not_Assignable	OCCUPY_NIGHT_DEFENSIVE_POSITION
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Not_Assignable	RESUME

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Not_Assignable	STINGER_DEFENSE_CONVOY
BLUFOR	Platoon	BLUFOR_ADA_PLATOON	Not_Assignable	STINGER_DEFENSE_MOVING
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	BOUNDING_OVERWATCH
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	CFS_DISMOUNT
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	CFS_HALT
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	CFS_MOUNT
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	CFS_OCCUPY_POSITION
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	CFS_TRAVEL
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	DEFEND_BP
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	DISENGAGE_MOUNTED
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	DISMOUNT_VEHICLE
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	ESTABLISH_HASTY_BP
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	EXECUTE_HALT
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	EXECUTE_TRAVELING

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	MOUNT_VEHICLE
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	OCCUPY_ASSEMBLY_AREA
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	PASSAGE_OF_LINES
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	RESUPPLY
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	SCREEN_MOVEMENT
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	SUPPRESSIVE_FIRE
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Assignable	TRAVELING_OVERWATCH
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Not_Assignable	ACTIONS_AT_OBSTACLE
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Not_Assignable	ACTIONS_ON_CONTACT
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Not_Assignable	CFS_CONDUCT_TACTICAL_ROAD_MARCH
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Not_Assignable	CONDUCT_TACTICAL_ROAD_MARCH
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Not_Assignable	CONTINUE_MISSION
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Not_Assignable	EXECUTE_ACTION_DRILL

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Not_Assignable	IDLE
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Not_Assignable	REACT_TO_AIR_ATTACK
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Not_Assignable	REACT_TO_IFIRE
BLUFOR	Platoon	BLUFOR_ARMORED_CAVALRY_PLATOON	Not_Assignable	RESUME
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	React_To_Ground_Attack
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	React_To_Indirect_Fire
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	React_To_Minefield
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	React_To_Obstacle
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	React_To_Air_Attack
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Request_Repair
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Move
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Repair
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Recover
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Hitch
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Halt
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Unhitch
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Dispatch
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Authorize_Repair

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Transfer_Maintenance_Crew
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Terminate_Mission
BLUFOR	Platoon	BLUFOR_Maintenance	Assignable	Idle
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	ALTERNATE_POSITIONS
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	ASSAULT_MOUNTED
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	BREACH_OBSTACLE
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CFS_DISMOUNT
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CFS_HALT
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CFS_MOUNT
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CFS_OCCUPY_POSITION
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CFS_TRAVEL
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CLEAR_WOODLINE
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CONDUCT_FIRE_AND_MOVEMENT
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CONDUCT_SCREEN_GUARD
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CONDUCT_TACTICAL_ROAD_MARCH
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CONSOLIDATE_AND_REORG
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CROSS_DANGER_AREA_DISMOUNTED
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	CROSS_DEFILE
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	DEFEND_BP
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	DISENGAGE_DISMOUNTED
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	DISENGAGE_MOUNTED

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	ESTABLISH_HASTY_BP
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	EXECUTE_BOUNDING_OVERWATCH
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	EXECUTE_HALT
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	EXECUTE_TRAVELING
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	EXECUTE_TRAVELING_OVERWATCH
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	HASTY_DISMOUNT
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	MOUNT_VEHICLE
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	MOVE_DISMOUNTED_OVERWATCH
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	MOVE_DISMOUNTED_TRAVELING
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	OCCUPY_ASSEMBLY_AREA
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	SUPPORT_BY_FIRE
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	SUPPRESSIVE_FIRE
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Assignable	SUSTAIN
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	ACTIONS_AT_OBSTACLE
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	ASSAULT_POSITION_ACTIVITIES
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	BAO_MP_ASSAULT_FORCE_ACTIVITIES
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	CONTINUE_MISSION
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	EMPLACE_HASTY_MINEFIELD
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	EXECUTE_ACTION_DRILL
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	IDLE
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	REACT_TO_AIR_ATTACK

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	REACT_TO_CONTACT
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	REACT_TO_CONTACT_DISMOUNTED
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	REACT_TO_DIRECT_FIRE
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	REACT_TO_INDIRECT_FIRE
BLUFOR	Platoon	BLUFOR_MECH_PLATOON	Not_Assignable	RESUME
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Assignable	CONDUCT_SCREEN
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Assignable	EXECUTE_BOUNDING_OVERWATCH_3_TEAM
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Assignable	EXECUTE_HALT
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Assignable	OCCUPY_ASSEMBLY_AREA
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Assignable	PERFORM_PASSAGE_OF_LINES
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Assignable	PERFORM_RESUPPLY_OPERATIONS
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Assignable	PERFORM_TACTICAL_ROAD_MARCH
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Assignable	PERFORM_ZONE_RECON
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Assignable	SUBORDINATE_LEVEL_TASKING
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Not_Assignable	CONTINUE_MISSION
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Not_Assignable	EXECUTE_ACTIONS_ON_CONTACT
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Not_Assignable	IDLE
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Not_Assignable	REACT_TO_AIR_ATTACK
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Not_Assignable	REACT_TO_IFIRE
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Not_Assignable	RESUME
BLUFOR	Platoon	BLUFOR_SCOUT_PLATOON	Not_Assignable	TRM_HALT

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	CONDUCT_SCREEN
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	CONTINUE_MISSION
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	EXECUTE_ACTIONS_ON_CONTACT
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	EXECUTE_BOUNDING_OVERWATCH
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	EXECUTE_HALT
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	EXECUTE_TRAVELING
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	IDLE
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	OCCUPY_ASSEMBLY_AREA
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	PERFORM_PASSAGE_OF_LINES
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	PERFORM_RESUPPLY_OPERATIONS
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	PERFORM_ZONE_RECON
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	REACT_TO_AIR_ATTACK
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	REACT_TO_IFIRE
BLUFOR	Platoon	BLUFOR_SCOUT_TEAM	Not_Assignable	RESUME
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	ALTERNATE_POSITIONS
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	ASSAULT_AN_ENEMY_POSITION
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	BREACH_OBSTACLE
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	CFS_HALT
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	CFS_OCCUPY_POSITION
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	CFS_TRAVEL
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	CONDUCT_TACTICAL_ROAD_MARCH

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	CONSOLIDATE_AND_REORG
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	DISPLACE_TO_A_SUBSEQUENT_BATTLE_POSITION
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	EXECUTE_BOUNDING_OVERWATCH
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	EXECUTE_HALT
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	EXECUTE_PLATOON_DEFENSIVE_MISSION
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	EXECUTE_TRAVELING
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	EXECUTE_TRAVELING_OVERWATCH
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	HASTY_OCCUPATION_OF_BP
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	OCCUPY_A_PLATOON_BATTLE_POSITION
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	PERFORM_ASSEMBLY_AREA_ACTIVITIES
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	PERFORM_ATTACK_BY_FIRE
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	PERFORM_PASSAGE_OF_LINES
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	PERFORM_PLATOON_FIRE_AND_MOVEMENT
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	PERFORM_RESUPPLY_OPERATIONS
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Assignable	SUPPRESSIVE_FIRE
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	ASSAULT_POSITION_ACTIVITIES
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	AT_ENEMY_AT_OBSTACLE
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	BAO_TP_BREACH_FORCE_ACTIVITIES
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	BAO_TP_SUPPORT_FORCE_ACTIVITIES
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	CONTINUE_MISSION
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	EXECUTE_ACTION_DRILL

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	EXECUTE_ACTIONS_ON_CONTACT
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	EXECUTE_CONTACT_DRILL
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	IDLE
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	REACT_TO_AIR_ATTACK
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	REACT_TO_DISMOUNTED_ATTACK
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	REACT_TO_IFIRE
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	RESUME
BLUFOR	Platoon	BLUFOR_TANK_PLATOON	Not_Assignable	TAKE_ACTIONS_AT_OBSTACLE
BLUFOR	Squad	BLUFOR_FIRETEAM	Assignable	EXECUTE_HALT
BLUFOR	Squad	BLUFOR_FIRETEAM	Assignable	EXECUTE_TRAVELING
BLUFOR	Squad	BLUFOR_FIRETEAM	Assignable	HASTY_DISMOUNT
BLUFOR	Squad	BLUFOR_FIRETEAM	Assignable	MOUNT_VEHICLE
BLUFOR	Squad	BLUFOR_FIRETEAM	Assignable	OCCUPY_POSITION
BLUFOR	Squad	BLUFOR_FIRETEAM	Not_Assignable	BREACH_OBSTACLE
BLUFOR	Squad	BLUFOR_FIRETEAM	Not_Assignable	CHANGE_STANCE
BLUFOR	Squad	BLUFOR_FIRETEAM	Not_Assignable	CLEAR_MINEFILED
BLUFOR	Squad	BLUFOR_FIRETEAM	Not_Assignable	CLEAR_OBSTACLE
BLUFOR	Squad	BLUFOR_FIRETEAM	Not_Assignable	ENGAGE_ENEMY
BLUFOR	Squad	BLUFOR_FIRETEAM	Not_Assignable	IDLE
BLUFOR	Squad	BLUFOR_FIRETEAM	Not_Assignable	RECOVER_MINES
BLUFOR	Squad	BLUFOR_TACP_CAS_Mission	Assignable	Cancel_Mission

Side	Echelon	Unit	User Assignable	Order
BLUFOR	Squad	BLUFOR_TACP_CAS_Mission	Assignable	Idle
BLUFOR	Squad	BLUFOR_TACP_CAS_Mission	Assignable	Perform_Mission
BLUFOR	Troop	BLUFOR_AIR_CAVALRY_TROOP	Assignable	CONDUCT_AREA_RECONNAISSANCE
BLUFOR	Troop	BLUFOR_AIR_CAVALRY_TROOP	Assignable	ENGAGE_TARGETS
BLUFOR	Troop	BLUFOR_AIR_CAVALRY_TROOP	Assignable	ENGAGE_TARGETS_RUNNING
BLUFOR	Troop	BLUFOR_AIR_CAVALRY_TROOP	Assignable	EXECUTE_TRAVELING
BLUFOR	Troop	BLUFOR_AIR_CAVALRY_TROOP	Assignable	HALT
BLUFOR	Troop	BLUFOR_AIR_CAVALRY_TROOP	Assignable	SUBORDINATE_LEVEL_TASKING
BLUFOR	Troop	BLUFOR_AIR_CAVALRY_TROOP	Not_Assignable	ACTIONS_ON_CONTACT
BLUFOR	Troop	BLUFOR_AIR_CAVALRY_TROOP	Not_Assignable	CONTINUE_MISSION
BLUFOR	Troop	BLUFOR_AIR_CAVALRY_TROOP	Not_Assignable	IDLE
BLUFOR	Troop	BLUFOR_AIR_CAVALRY_TROOP	Not_Assignable	PERFORM_ACTIONS_ON_CONTACT
BLUFOR	Troop	BLUFOR_AIR_CAVALRY_TROOP	Not_Assignable	RESUME
OPFOR	Battalion	OPFOR_MR_BATTALION	Assignable	ATTACK_FROM_THE_MARCH
OPFOR	Battalion	OPFOR_MR_BATTALION	Assignable	EXECUTE_HALT
OPFOR	Battalion	OPFOR_MR_BATTALION	Assignable	OCCUPY_AN_ASSEMBLY_AREA
OPFOR	Battalion	OPFOR_MR_BATTALION	Assignable	SUBORDINATE_LEVEL_TASKING
OPFOR	Battalion	OPFOR_MR_BATTALION	Assignable	TRAVELING
OPFOR	Battalion	OPFOR_MR_BATTALION	Not_Assignable	CONTINUE_MISSION
OPFOR	Battalion	OPFOR_MR_BATTALION	Not_Assignable	IDLE
OPFOR	Battalion	OPFOR_SP_ARTILLERY_BATTALION	Assignable	SUBORDINATE_LEVEL_TASKING

Side	Echelon	Unit	User Assignable	Order
OPFOR	Battalion	OPFOR_SP_ARTILLERY_BATTALION	Not_Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Battalion	OPFOR_SP_ARTILLERY_BATTALION	Not_Assignable	CONTINUE_MISSION
OPFOR	Battalion	OPFOR_SP_ARTILLERY_BATTALION	Not_Assignable	DEPLOY_COMBAT_FORMATION
OPFOR	Battalion	OPFOR_SP_ARTILLERY_BATTALION	Not_Assignable	EXECUTE_HALT
OPFOR	Battalion	OPFOR_SP_ARTILLERY_BATTALION	Not_Assignable	IDLE
OPFOR	Battalion	OPFOR_SP_ARTILLERY_BATTALION	Not_Assignable	TRM_HALT
OPFOR	Battalion	OPFOR_TANK_BATTALION	Assignable	EXECUTE_HALT
OPFOR	Battalion	OPFOR_TANK_BATTALION	Assignable	SUBORDINATE_LEVEL_TASKING
OPFOR	Battalion	OPFOR_TANK_BATTALION	Not_Assignable	ATTACK_FROM_THE_MARCH
OPFOR	Battalion	OPFOR_TANK_BATTALION	Not_Assignable	CONTINUE_MISSION
OPFOR	Battalion	OPFOR_TANK_BATTALION	Not_Assignable	IDLE
OPFOR	Battalion	OPFOR_TANK_BATTALION	Not_Assignable	OCCUPY_AN_ASSEMBLY_AREA
OPFOR	Battalion	OPFOR_TANK_BATTALION	Not_Assignable	TRAVELING
OPFOR	Battalion	OPFOR_TOWED_ARTILLERY_BATTALION	Assignable	EXECUTE_HALT
OPFOR	Battalion	OPFOR_TOWED_ARTILLERY_BATTALION	Assignable	SUBORDINATE_LEVEL_TASKING
OPFOR	Battalion	OPFOR_TOWED_ARTILLERY_BATTALION	Not_Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Battalion	OPFOR_TOWED_ARTILLERY_BATTALION	Not_Assignable	CONTINUE_MISSION
OPFOR	Battalion	OPFOR_TOWED_ARTILLERY_BATTALION	Not_Assignable	DEPLOY_COMBAT_FORMATION

Side	Echelon	Unit	User Assignable	Order
OPFOR	Battalion	OPFOR_TOWED_ARTILLERY_BATTALION	Not_Assignable	IDLE
OPFOR	Battalion	OPFOR_TOWED_ARTILLERY_BATTALION	Not_Assignable	TRM_HALT
OPFOR	Company	OPFOR_AD_2S6_BATTERY	Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Company	OPFOR_AD_2S6_BATTERY	Assignable	ENGAGE_AERIAL_TARGET
OPFOR	Company	OPFOR_AD_2S6_BATTERY	Assignable	EXECUTE_HALT
OPFOR	Company	OPFOR_AD_2S6_BATTERY	Assignable	EXECUTE_MOVEMENT
OPFOR	Company	OPFOR_AD_2S6_BATTERY	Assignable	OCCUPY_AIR_DEFENSE_FIRING_POSITION
OPFOR	Company	OPFOR_AD_2S6_BATTERY	Assignable	PROVIDE_AD_COVERAGE
OPFOR	Company	OPFOR_AD_2S6_BATTERY	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Company	OPFOR_AD_2S6_BATTERY	Not_Assignable	CONTINUE_MISSION
OPFOR	Company	OPFOR_AD_2S6_BATTERY	Not_Assignable	IDLE
OPFOR	Company	OPFOR_AD_2S6_BATTERY	Not_Assignable	REACT_TO_ENEMY_GROUND_ATTACK
OPFOR	Company	OPFOR_AD_2S6_BATTERY	Not_Assignable	RESUME
OPFOR	Company	OPFOR_AD_SA13_BATTERY	Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Company	OPFOR_AD_SA13_BATTERY	Assignable	ENGAGE_AERIAL_TARGET
OPFOR	Company	OPFOR_AD_SA13_BATTERY	Assignable	EXECUTE_HALT
OPFOR	Company	OPFOR_AD_SA13_BATTERY	Assignable	EXECUTE_MOVEMENT
OPFOR	Company	OPFOR_AD_SA13_BATTERY	Assignable	OCCUPY_AIR_DEFENSE_FIRING_POSITION
OPFOR	Company	OPFOR_AD_SA13_BATTERY	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Company	OPFOR_AD_SA13_BATTERY	Not_Assignable	CONTINUE_MISSION

Side	Echelon	Unit	User Assignable	Order
OPFOR	Company	OPFOR_AD_SA13_BATTERY	Not_Assignable	IDLE
OPFOR	Company	OPFOR_AD_SA13_BATTERY	Not_Assignable	REACT_TO_ENEMY_GROUND_ATTACK
OPFOR	Company	OPFOR_AD_SA13_BATTERY	Not_Assignable	RESUME
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Assignable	DISMOUNT_VEHICLES
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Assignable	ENGAGE_AERIAL_TARGET
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Assignable	EXECUTE_HALT
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Assignable	EXECUTE_MOVEMENT
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Assignable	OCCUPY_AIR_DEFENSE_FIRING_POSITION
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Assignable	REMOUNT_VEHICLES
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Not_Assignable	CONTINUE_MISSION
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Not_Assignable	IDLE
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Not_Assignable	REACT_TO_ENEMY_GROUND_ATTACK
OPFOR	Company	OPFOR_AD_SA15_BATTERY	Not_Assignable	RESUME
OPFOR	Company	OPFOR_MORTAR_BATTERY	Assignable	CONDUCT_UNOBSERVED_FIRE_MISSION
OPFOR	Company	OPFOR_MORTAR_BATTERY	Assignable	EXECUTE_HALT
OPFOR	Company	OPFOR_MORTAR_BATTERY	Assignable	EXECUTE_MOVEMENT
OPFOR	Company	OPFOR_MORTAR_BATTERY	Assignable	OCCUPY_A_FIRING_POSITION
OPFOR	Company	OPFOR_MORTAR_BATTERY	Assignable	TACTICAL_ROAD_MARCH
OPFOR	Company	OPFOR_MORTAR_BATTERY	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE

Side	Echelon	Unit	User Assignable	Order
OPFOR	Company	OPFOR_MORTAR_BATTERY	Not_Assignable	CONTINUE_MISSION
OPFOR	Company	OPFOR_MORTAR_BATTERY	Not_Assignable	IDLE
OPFOR	Company	OPFOR_MORTAR_BATTERY	Not_Assignable	RESUME
OPFOR	Company	OPFOR_MORTAR_BATTERY	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	ASSAULT_AN_ENEMY_POSITION
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	ATTACK_FROM_THE_MARCH
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	EXECUTE_A_FIRE_ENGAGEMENT
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	EXECUTE_HALT
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	EXECUTE_TRAVELING
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	OCCUPY_A_DEFENSIVE_STRONG_POINT
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	OCCUPY_A_TEMPORARY_DEFENSIVE_POSITION
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	OCCUPY_AN_ASSEMBLY_AREA
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	OCCUPY_STRONG_POINT_SUPPLEMENTARY_POSITIONS
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	PROVIDE_SECURITY_ON_THE_MOVE
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	SUBORDINATE_LEVEL_TASKING
OPFOR	Company	OPFOR_MR_COMPANY	Assignable	WITHDRAW_DISENGAGE
OPFOR	Company	OPFOR_MR_COMPANY	Not_Assignable	ACTIONS_ON_INCOMING_ARTILLERY
OPFOR	Company	OPFOR_MR_COMPANY	Not_Assignable	CONDUCT_A_MEETING_ENGAGEMENT
OPFOR	Company	OPFOR_MR_COMPANY	Not_Assignable	CONDUCT_COMBAT_IN_AN_ENCIRCLEMENT

Side	Echelon	Unit	User Assignable	Order
OPFOR	Company	OPFOR_MR_COMPANY	Not_Assignable	CONTINUE_MISSION
OPFOR	Company	OPFOR_MR_COMPANY	Not_Assignable	EXECUTE_ACTIONS_ON_CONTACT
OPFOR	Company	OPFOR_MR_COMPANY	Not_Assignable	IDLE
OPFOR	Company	OPFOR_MR_COMPANY	Not_Assignable	TAKE_ACTIONS_AT_AN_OBSTACLE
OPFOR	Company	OPFOR_MR_COMPANY	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Company	OPFOR_MR_COMPANY	Not_Assignable	TRM_HALT
OPFOR	Company	OPFOR_SP_ARTILLERY_BATTERY	Assignable	CONDUCT_UNOBSERVED_FIRE_MISSION
OPFOR	Company	OPFOR_SP_ARTILLERY_BATTERY	Assignable	EXECUTE_HALT
OPFOR	Company	OPFOR_SP_ARTILLERY_BATTERY	Assignable	EXECUTE_MOVEMENT
OPFOR	Company	OPFOR_SP_ARTILLERY_BATTERY	Assignable	OCCUPY_A_FIRING_POSITION
OPFOR	Company	OPFOR_SP_ARTILLERY_BATTERY	Assignable	TACTICAL_ROAD_MARCH
OPFOR	Company	OPFOR_SP_ARTILLERY_BATTERY	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Company	OPFOR_SP_ARTILLERY_BATTERY	Not_Assignable	CONDUCT_DIRECT_FIRE_MISSION
OPFOR	Company	OPFOR_SP_ARTILLERY_BATTERY	Not_Assignable	CONTINUE_MISSION
OPFOR	Company	OPFOR_SP_ARTILLERY_BATTERY	Not_Assignable	IDLE
OPFOR	Company	OPFOR_SP_ARTILLERY_BATTERY	Not_Assignable	RESUME
OPFOR	Company	OPFOR_SP_ARTILLERY_BATTERY	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	ASSAULT_AN_ENEMY_POSITION
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	ATTACK_FROM_THE_MARCH
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	EXECUTE_A_FIRE_ENGAGEMENT

Side	Echelon	Unit	User Assignable	Order
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	EXECUTE_HALT
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	EXECUTE_TRAVELING
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	OCCUPY_A_DEFENSIVE_STRONG_POINT
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	OCCUPY_A_TEMPORARY_DEFENSIVE_POSITION
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	OCCUPY_AN_ASSEMBLY_AREA
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	OCCUPY_STRONG_POINT_SUPPLEMENTARY_POSITIONS
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	PROVIDE_SECURITY_ON_THE_MOVE
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	SUBORDINATE_LEVEL_TASKING
OPFOR	Company	OPFOR_TANK_COMPANY	Assignable	WITHDRAW_DISENGAGE
OPFOR	Company	OPFOR_TANK_COMPANY	Not_Assignable	ACTIONS_ON_INCOMING_ARTILLERY
OPFOR	Company	OPFOR_TANK_COMPANY	Not_Assignable	CONDUCT_A_MEETING_ENGAGEMENT
OPFOR	Company	OPFOR_TANK_COMPANY	Not_Assignable	CONTINUE_MISSION
OPFOR	Company	OPFOR_TANK_COMPANY	Not_Assignable	EXECUTE_ACTIONS_ON_CONTACT
OPFOR	Company	OPFOR_TANK_COMPANY	Not_Assignable	IDLE
OPFOR	Company	OPFOR_TANK_COMPANY	Not_Assignable	TAKE_ACTIONS_AT_AN_OBSTACLE
OPFOR	Company	OPFOR_TANK_COMPANY	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Company	OPFOR_TANK_COMPANY	Not_Assignable	TRM_HALT
OPFOR	Company	OPFOR_TOWED_ARTILLERY_BATTERY	Assignable	CONDUCT_UNOBSERVED_FIRE_MISSION
OPFOR	Company	OPFOR_TOWED_ARTILLERY_BATTERY	Assignable	EXECUTE_HALT

Side	Echelon	Unit	User Assignable	Order
OPFOR	Company	OPFOR_TOWED_ARTILLERY_BATTERY	Assignable	EXECUTE_MOVEMENT
OPFOR	Company	OPFOR_TOWED_ARTILLERY_BATTERY	Assignable	OCCUPY_A_FIRING_POSITION
OPFOR	Company	OPFOR_TOWED_ARTILLERY_BATTERY	Assignable	TACTICAL_ROAD_MARCH
OPFOR	Company	OPFOR_TOWED_ARTILLERY_BATTERY	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Company	OPFOR_TOWED_ARTILLERY_BATTERY	Not_Assignable	CONDUCT_DIRECT_FIRE_MISSION
OPFOR	Company	OPFOR_TOWED_ARTILLERY_BATTERY	Not_Assignable	EXECUTE_CTRM_HALT
OPFOR	Company	OPFOR_TOWED_ARTILLERY_BATTERY	Not_Assignable	IDLE
OPFOR	Company	OPFOR_TOWED_ARTILLERY_BATTERY	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Assignable	ASSAULT_ENEMY_POSITION
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Assignable	DISMOUNT_VEHICLES
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Assignable	ENGAGE_AERIAL_TARGET
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Assignable	OCCUPY_AIR_DEFENSE_FIRING_POSITION
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Assignable	REMOUNT_VEHICLES

Side	Echelon	Unit	User Assignable	Order
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Not_Assignable	EXECUTE_A_FIRE_ENGAGEMENT
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_AD_SA16_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_AGL_PLATOON	Assignable	DISMOUNT_VEHICLES
OPFOR	Platoon	OPFOR_AGL_PLATOON	Assignable	EXECUTE_A_FIRE_ENGAGEMENT
OPFOR	Platoon	OPFOR_AGL_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_AGL_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_AGL_PLATOON	Assignable	OCCUPY_A_DEFENSIVE_POSITION
OPFOR	Platoon	OPFOR_AGL_PLATOON	Assignable	REMOUNT_VEHICLES
OPFOR	Platoon	OPFOR_AGL_PLATOON	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Platoon	OPFOR_AGL_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_AGL_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_AGL_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_AGL_PLATOON	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Platoon	OPFOR_ASSAULT_HELICOPTER_FLIGHT	Assignable	CONDUCT_AIR_LIFT_OPERATIONS
OPFOR	Platoon	OPFOR_ASSAULT_HELICOPTER_FLIGHT	Assignable	EXECUTE_TRAVELING
OPFOR	Platoon	OPFOR_ASSAULT_HELICOPTER_FLIGHT	Not_Assignable	ACTIONS_ON_CONTACT

Side	Echelon	Unit	User Assignable	Order
OPFOR	Platoon	OPFOR_ASSAULT_HELICOPTER_FLIGHT	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_ASSAULT_HELICOPTER_FLIGHT	Not_Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_ASSAULT_HELICOPTER_FLIGHT	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_ASSAULT_HELICOPTER_FLIGHT	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_ATG_FIRING_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_ATG_FIRING_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_ATG_FIRING_PLATOON	Assignable	OCCUPY_A_DEFENSIVE_POSITION
OPFOR	Platoon	OPFOR_ATG_FIRING_PLATOON	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Platoon	OPFOR_ATG_FIRING_PLATOON	Not_Assignable	CONDUCT_A_DEFENSE
OPFOR	Platoon	OPFOR_ATG_FIRING_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_ATG_FIRING_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_ATG_FIRING_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_ATG_FIRING_PLATOON	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Assignable	DISMOUNT_VEHICLES
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Assignable	OCCUPY_A_DEFENSIVE_POSITION
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Assignable	OCCUPY_A_TEMP_DEF_POSITION

Side	Echelon	Unit	User Assignable	Order
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Assignable	OCCUPY_DEFENSIVE_SUPPLEMENTARY_POSITIONS
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Assignable	REMOUNT_VEHICLES
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Not_Assignable	ACTIONS_ON_INCOMING_ARTILLERY
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Not_Assignable	CONDUCT_A_DEFENSE
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Not_Assignable	EXECUTE_A_FIRE_ENGAGEMENT
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Not_Assignable	EXECUTE_ACTIONS_ON_CONTACT
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Not_Assignable	TAKE_ACTIONS_AT_AN_OBSTACLE
OPFOR	Platoon	OPFOR_ATGM_PLATOON	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Platoon	OPFOR_ATTACK_HELICOPTER_FLIGHT	Assignable	CONDUCT_ATTACK_ON_GROUND_TARGET
OPFOR	Platoon	OPFOR_ATTACK_HELICOPTER_FLIGHT	Assignable	CONDUCT_ATTACK_ON_GROUND_TARGET_RUNNING
OPFOR	Platoon	OPFOR_ATTACK_HELICOPTER_FLIGHT	Assignable	EXECUTE_TRAVELING
OPFOR	Platoon	OPFOR_ATTACK_HELICOPTER_FLIGHT	Not_Assignable	ACTIONS_ON_CONTACT
OPFOR	Platoon	OPFOR_ATTACK_HELICOPTER_FLIGHT	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_ATTACK_HELICOPTER_FLIGHT	Not_Assignable	EXECUTE_HALT

Side	Echelon	Unit	User Assignable	Order
OPFOR	Platoon	OPFOR_ATTACK_HELICOPTER_FLIGHT	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_ATTACK_HELICOPTER_FLIGHT	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_BRIDGE_PLATOON	Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Platoon	OPFOR_BRIDGE_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_BRIDGE_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_BRIDGE_PLATOON	Not_Assignable	ACTIONS_ON_INCOMING_ARTILLERY
OPFOR	Platoon	OPFOR_BRIDGE_PLATOON	Not_Assignable	CONDUCT_BRIDGING_OPERATIONS
OPFOR	Platoon	OPFOR_BRIDGE_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_BRIDGE_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_BRIDGE_PLATOON	Not_Assignable	OCCUPY_COVERED_CONCEALED_POSITIONS
OPFOR	Platoon	OPFOR_BRIDGE_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_BRIDGE_PLATOON	Not_Assignable	RETRACT_BRIDGE
OPFOR	Platoon	OPFOR_BRIDGE_PLATOON	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Platoon	OPFOR_FW_AIRCRAFT_FLIGHT	Assignable	CONDUCT_ATTACK_ON_GROUND_TARGET
OPFOR	Platoon	OPFOR_FW_AIRCRAFT_FLIGHT	Not_Assignable	ACTIONS_ON_CONTACT
OPFOR	Platoon	OPFOR_FW_AIRCRAFT_FLIGHT	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_FW_AIRCRAFT_FLIGHT	Not_Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_FW_AIRCRAFT_FLIGHT	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_FW_AIRCRAFT_FLIGHT	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_MINE_WARFARE_PLATOON	Assignable	CONDUCT_TACTICAL_ROAD_MARCH

Side	Echelon	Unit	User Assignable	Order
OPFOR	Platoon	OPFOR_MINE_WARFARE_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_MINE_WARFARE_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_MINE_WARFARE_PLATOON	Not_Assignable	ACTIONS_ON_INCOMING_ARTILLERY
OPFOR	Platoon	OPFOR_MINE_WARFARE_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_MINE_WARFARE_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_MINE_WARFARE_PLATOON	Not_Assignable	LAY_A_MINE_FIELD
OPFOR	Platoon	OPFOR_MINE_WARFARE_PLATOON	Not_Assignable	OCCUPY_COVERED_CONCEALED_POSITIONS
OPFOR	Platoon	OPFOR_MINE_WARFARE_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_MINE_WARFARE_PLATOON	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	ASSAULT_ENEMY_POSITION
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	DISMOUNT_VEHICLES
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	EXECUTE_A_FIRE_ENGAGEMENT
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	OCCUPY_A_DEFENSE_STRONG_POINT
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	OCCUPY_A_TEMP_DEF_POSITION
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	OCCUPY_AN_ASSEMBLY_AREA
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	OCCUPY_STRONG_POINT_SUPPLEMENTARY_POSITIONS
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	PROVIDE_FORCE_SECURITY
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	REMOUNT_VEHICLES

Side	Echelon	Unit	User Assignable	Order
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	TRAVEL_BYPASS_ROUTE
OPFOR	Platoon	OPFOR_MR_PLATOON	Assignable	WITHDRAW
OPFOR	Platoon	OPFOR_MR_PLATOON	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Platoon	OPFOR_MR_PLATOON	Not_Assignable	CONDUCT_A_DEFENSE
OPFOR	Platoon	OPFOR_MR_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_MR_PLATOON	Not_Assignable	EXECUTE_ACTIONS_ON_CONTACT
OPFOR	Platoon	OPFOR_MR_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_MR_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_MR_PLATOON	Not_Assignable	TAKE_ACTIONS_AT_AN_OBSTACLE
OPFOR	Platoon	OPFOR_MR_PLATOON	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Platoon	OPFOR_MR_PLATOON	Not_Assignable	TAKE_EVASIVE_ACTIONS
OPFOR	Platoon	OPFOR_RECON_PLATOON	Assignable	CONDUCT_AN_AREA_RECON_MISSION
OPFOR	Platoon	OPFOR_RECON_PLATOON	Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Platoon	OPFOR_RECON_PLATOON	Assignable	DISMOUNT_VEHICLES
OPFOR	Platoon	OPFOR_RECON_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_RECON_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_RECON_PLATOON	Assignable	REMOUNT_VEHICLES
OPFOR	Platoon	OPFOR_RECON_PLATOON	Assignable	WITHDRAW
OPFOR	Platoon	OPFOR_RECON_PLATOON	Not_Assignable	ACTIONS_ON_INCOMING_ARTILLERY
OPFOR	Platoon	OPFOR_RECON_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_RECON_PLATOON	Not_Assignable	EXECUTE_A_FIRE_ENGAGEMENT

Side	Echelon	Unit	User Assignable	Order
OPFOR	Platoon	OPFOR_RECON_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_RECON_PLATOON	Not_Assignable	OCCUPY_COVERED_CONCEALED_POSITIONS
OPFOR	Platoon	OPFOR_RECON_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_RECON_PLATOON	Not_Assignable	TAKE_ACTIONS_AT_AN_OBSTACLE
OPFOR	Platoon	OPFOR_RECON_PLATOON	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Platoon	OPFOR_RECON_PLATOON	Not_Assignable	TRAVEL_BYPASS_ROUTE
OPFOR	Platoon	OPFOR_RECOVERY_PLATOON	Assignable	CONDUCT_RECOVERY_OPS
OPFOR	Platoon	OPFOR_RECOVERY_PLATOON	Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Platoon	OPFOR_RECOVERY_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_RECOVERY_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_RECOVERY_PLATOON	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Platoon	OPFOR_RECOVERY_PLATOON	Not_Assignable	ACTIONS_ON_CONTACT
OPFOR	Platoon	OPFOR_RECOVERY_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_RECOVERY_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_RECOVERY_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_RECOVERY_PLATOON	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Platoon	OPFOR_SUPPLY_PLATOON	Assignable	CONDUCT_REARM_REFUEL_OPS
OPFOR	Platoon	OPFOR_SUPPLY_PLATOON	Assignable	CONDUCT_TACTICAL_ROAD_MARCH
OPFOR	Platoon	OPFOR_SUPPLY_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_SUPPLY_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_SUPPLY_PLATOON	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE

Side	Echelon	Unit	User Assignable	Order
OPFOR	Platoon	OPFOR_SUPPLY_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_SUPPLY_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_SUPPLY_PLATOON	Not_Assignable	OCCUPY_COVERED_CONCEALED_POSITIONS
OPFOR	Platoon	OPFOR_SUPPLY_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	ASSAULT_ENEMY_POSITION
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	EXECUTE_A_FIRE_ENGAGEMENT
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	OCCUPY_A_DEFENSE_STRONG_POINT
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	OCCUPY_A_TEMP_DEF_POSITION
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	OCCUPY_AN_ASSEMBLY_AREA
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	OCCUPY_STRONG_POINT_SUPPLEMENTARY_POSITIONS
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	PROVIDE_FORCE_SECURITY
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	TACTICAL_ROAD_MARCH
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	TRAVEL_BYPASS_ROUTE
OPFOR	Platoon	OPFOR_TANK_PLATOON	Assignable	WITHDRAW
OPFOR	Platoon	OPFOR_TANK_PLATOON	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Platoon	OPFOR_TANK_PLATOON	Not_Assignable	CONDUCT_A_DEFENSE
OPFOR	Platoon	OPFOR_TANK_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_TANK_PLATOON	Not_Assignable	EXECUTE_ACTIONS_ON_CONTACT

Side	Echelon	Unit	User Assignable	Order
OPFOR	Platoon	OPFOR_TANK_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_TANK_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_TANK_PLATOON	Not_Assignable	TAKE_ACTIONS_AT_AN_OBSTACLE
OPFOR	Platoon	OPFOR_TANK_PLATOON	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Platoon	OPFOR_TANK_PLATOON	Not_Assignable	TAKE_EVASIVE_ACTIONS
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Assignable	CLEAR_ENGINEER_OBSTACLE
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Assignable	CONDUCT_ROUTE_CLEARING_OPERATION
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Assignable	DISMOUNT_VEHICLES
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Assignable	EXECUTE_HALT
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Assignable	EXECUTE_MOVEMENT
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Assignable	REMOUNT_VEHICLES
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Not_Assignable	ACTIONS_ON_INCOMING_ARTILLERY
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Not_Assignable	CONSTRUCT_ENGINEER_OBSTACLE
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Not_Assignable	CONTINUE_MISSION
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Not_Assignable	IDLE
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Not_Assignable	OCCUPY_COVERED_CONCEALED_POSITIONS
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Not_Assignable	RESUME
OPFOR	Platoon	OPFOR_TECH_CONST_PLATOON	Not_Assignable	TAKE_AIR_DEFENSE_MEASURES
OPFOR	Squad	OPFOR_ATGM_SQUAD	Assignable	DISMOUNT_VEHICLES
OPFOR	Squad	OPFOR_ATGM_SQUAD	Assignable	EXECUTE_HALT
OPFOR	Squad	OPFOR_ATGM_SQUAD	Assignable	EXECUTE_MOVEMENT

Side	Echelon	Unit	User Assignable	Order
OPFOR	Squad	OPFOR_ATGM_SQUAD	Assignable	OCCUPY_A_DEFENSIVE_POSITION
OPFOR	Squad	OPFOR_ATGM_SQUAD	Assignable	OCCUPY_DEFENSIVE_SUPPLEMENTARY_POSITIONS
OPFOR	Squad	OPFOR_ATGM_SQUAD	Assignable	REMOUNT_VEHICLES
OPFOR	Squad	OPFOR_ATGM_SQUAD	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Squad	OPFOR_ATGM_SQUAD	Not_Assignable	AIR_DEFENSE_MEASURES
OPFOR	Squad	OPFOR_ATGM_SQUAD	Not_Assignable	CONDUCT_A_DEFENSE
OPFOR	Squad	OPFOR_ATGM_SQUAD	Not_Assignable	CONTINUE_MISSION
OPFOR	Squad	OPFOR_ATGM_SQUAD	Not_Assignable	IDLE
OPFOR	Squad	OPFOR_ATGM_SQUAD	Not_Assignable	RESUME
OPFOR	Squad	OPFOR_DI_SQUAD	Not_Assignable	ACTIONS_ON_ARTILLERY_FIRE
OPFOR	Squad	OPFOR_DI_SQUAD	Not_Assignable	AIR_DEFENSE_MEASURES
OPFOR	Squad	OPFOR_DI_SQUAD	Not_Assignable	CONDUCT_A_DEFENSE
OPFOR	Squad	OPFOR_DI_SQUAD	Not_Assignable	CONTINUE_MISSION
OPFOR	Squad	OPFOR_DI_SQUAD	Not_Assignable	DISMOUNT_VEHICLES
OPFOR	Squad	OPFOR_DI_SQUAD	Not_Assignable	EXECUTE_HALT
OPFOR	Squad	OPFOR_DI_SQUAD	Not_Assignable	EXECUTE_MOVEMENT
OPFOR	Squad	OPFOR_DI_SQUAD	Not_Assignable	IDLE
OPFOR	Squad	OPFOR_DI_SQUAD	Not_Assignable	OCCUPY_POSITION
OPFOR	Squad	OPFOR_DI_SQUAD	Not_Assignable	REMOUNT_VEHICLES
OPFOR	Squad	OPFOR_DI_SQUAD	Not_Assignable	RESUME

A.2 ModSAF Behaviors

The following table list the set of orders (task frame) implemented within ModSAF. For each task frame the table identifies the list of vehicles and units that can be assigned that particular order and which vehicle and units cannot be assigned the particular order.

Task Frame Name	Allowed Units/Vehicles	Disallowed Units/Vehicles
Accept Unit	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES
ADA Hasty Occupy Position	(AIR_DEFENSE_VEHICLES AIR_DEFENSE_UNITS)	GROUND_SUPPLY_VEHICLES TMD_PLATOON_UNITS TMD_VEHICLES TMD_BN_CONTROL_UNITS PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS
Assault	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES ARTY_VEHICLES ARTY_UNITS CRUSADER_UNITS GROUND_SUPPLY_VEHICLES ORSMC_VEHICLE TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS UNARMED_VEHICLES
Assemble	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES ORSMC_VEHICLE TMD_VEHICLES
Assign FAC	(FAC_VEHICLES)	
Assisted Recover	(GROUND_VEHICLES)	DI TOWED_VEHICLES TMD_VEHICLES
ASTAMIDS Mine Detection	(ASTAMIDS_VEHICLES)	
ATD/Crater Breach	(GRIZZLY_VEHICLE)	
Attach Part	(MINE_BRCH_VEHICLES OBSTACLE_BRCH_VEHICLES MMCM_VEHICLES)	GRIZZLY_VEHICLE ACE_VEHICLE

Task Frame Name	Allowed Units/Vehicles	Disallowed Units/Vehicles
Attach Vehicle	(GROUND_VEHICLES)	DI TOWED_VEHICLES TMD_VEHICLES
Attack By Fire	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES ARTY_VEHICLES ARTY_UNITS CRUSADER_UNITS GROUND_SUPPLY_VEHICLES ORSMC_VEHICLE TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
Bn Halt	(GROUND_BATTALIONS)	
Bn March	(GROUND_BATTALIONS)	
Brdg Deploy	(OBSTACLE_BRCH_VEHICLES)	
Brdg Retrieve	(OBSTACLE_BRCH_VEHICLES)	
Change Formation	(GROUND_UNITS)	ORSMC_UNIT
Chemical Survey	(CHEM_RECON_VEHICLES)	
CO Accept Unit	(GROUND_COMPANIES)	TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
CO Assembly Area	(GROUND_COMPANIES)	
CO Attack	(GROUND_COMPANIES)	TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
CO Bounding OW	(GROUND_COMPANIES)	TOWED_VEHICLES ARTY_VEHICLES ARTY_UNITS CRUSADER_UNITS GROUND_SUPPLY_VEHICLES ORSMC_UNIT
CO Halt	(GROUND_COMPANIES)	

Task Frame Name	Allowed Units/Vehicles	Disallowed Units/Vehicles
CO Hasty Occupy Position	(GROUND_COMPANIES)	ARTY_UNITS CRUSADER_UNITS TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
CO March	(GROUND_COMPANIES)	
CO Pre-Battle March	(RED_GROUND_COMPANIES)	
CO Report To	(GROUND_COMPANIES)	
CO Road March	(GROUND_COMPANIES)	
CO Service Station	(GROUND_COMPANIES)	
CO Traveling OW	(GROUND_COMPANIES)	
CO Unassign Unit	(GROUND_COMPANIES)	
CO Withdraw	(GROUND_COMPANIES)	TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
Concealment	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES ORSMC_VEHICLE TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
Crater Breach	(CEV_VEHICLE)	
Cross-leveling	(GROUND_UNITS)	MIXED_IFV_DI_UNITS DI ORSMC_UNIT TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS

Task Frame Name	Allowed Units/Vehicles	Disallowed Units/Vehicles
Crusader POC	(CRUSADER_UNITS)	CRUSADER_UNITS_NO_PLATOON
Delay	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES ARTY_VEHICLES ARTY_UNITS CRUSADER_UNITS GROUND_SUPPLY_VEHICLES ORSMC_VEHICLE TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
Detach Part	(MINE_BRCH_VEHICLES OBSTACLE_BRCH_VEHICLES MMCM_VEHICLES)	GRIZZLY_VEHICLE ACE_VEHICLE
Detach Vehicle	(GROUND_VEHICLES)	DI TOWED_VEHICLES TMD_VEHICLES
Dig in Vehicle	(FIGHTING_POS_VEHICLES)	
Dismount	(MIXED_IFV_DI_UNITS)	
ESMB Operations	(ESMB_VEHICLES)	
Follow a Vehicle	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES ORSMC_VEHICLE PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
Follow Simulator	(GROUND_UNITS)	MIXED_IFV_DI_UNITS TOWED_VEHICLES
FWA Attack Ground Target	(FWA_VEHICLES FWA_UNITS)	FWA_SOAR_VEHICLES FWA_RECON_VEHICLES
FWA CAP	(FWA_VEHICLES)	FWA_SOAR_VEHICLES FWA_RECON_VEHICLES
FWA CAS Mission	(FWA_VEHICLES FWA_UNITS)	FWA_SOAR_VEHICLES FWA_RECON_VEHICLES
FWA Egress	(FWA_VEHICLES FWA_UNITS)	FWA_SOAR_VEHICLES FWA_RECON_VEHICLES
FWA Ingress	(FWA_VEHICLES FWA_UNITS)	FWA_SOAR_VEHICLES FWA_RECON_VEHICLES
FWA Interdiction Mission	(FWA_VEHICLES FWA_UNITS)	FWA_SOAR_VEHICLES FWA_RECON_VEHICLES

Task Frame Name	Allowed Units/Vehicles	Disallowed Units/Vehicles
FWA Return To Base	(FWA_VEHICLES)	FWA_SOAR_VEHICLES FWA_RECON_VEHICLES
FWA Sweep	(FWA_VEHICLES)	FWA_SOAR_VEHICLES FWA_RECON_VEHICLES
FWA UAV CAP	(FWA_RECON_VEHICLES)	
FWA UAV Return To Base	(FWA_RECON_VEHICLES)	
FWA UAV Sweep	(FWA_RECON_VEHICLES)	
Gun Breach	(GUN_BRCH_VEHICLES)	
Halt	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES
Hasty Occupy Position	(GROUND_VEHICLES GROUND_UNITS)	AIR_DEFENSE_VEHICLES AIR_DEFENSE_UNITS TOWED_VEHICLES ARTY_VEHICLES ARTY_UNITS CRUSADER_UNITS GROUND_SUPPLY_VEHICLES ORSMC_VEHICLE TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
Load Item	(GROUND_VEHICLES)	DI TOWED_VEHICLES TMD_VEHICLES
MICLIC Operations	(AVLM_VEHICLES)	
MICLIC Operations	(MICLIC_VEHICLES)	
Mine Breach	(MINE_BRCH_VEHICLES)	
Mine Edge Detection	(MINE_BRCH_VEHICLES)	
Mine Sweep	(MINE_SWEEP_UNITS)	
MLRS	(MLRS_VEHICLES)	
MMCM Breach	(MMCM_VEHICLES)	
Mount	(MIXED_IFV_DI_UNITS)	

Task Frame Name	Allowed Units/Vehicles	Disallowed Units/Vehicles
Move	(GROUND_VEHICLES GROUND_UNITS LINE_PAIR_BOARD WATER_VEHICLES)	TOWED_VEHICLES ORSMC_VEHICLE
Move Patients	(MEDICAL_VEHICLES)	
Move Repairmen	(GROUND_VEHICLES)	TMD_VEHICLES
ORSMC operation	(ORSMC_UNIT)	
Overwatch Movement	(GROUND_UNITS)	TOWED_VEHICLES ARTY_VEHICLES ARTY_UNITS CRUSADER_UNITS GROUND_SUPPLY_VEHICLES ORSMC_UNIT TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
Patriot Control	(PATRIOT_CONTROL_UNITS)	
Position Part	(POSPART_VEHICLES)	
Pursue	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES ARTY_VEHICLES ARTY_UNITS CRUSADER_UNITS ORSMC_VEHICLE ORSMC_UNIT
Recover	(GROUND_VEHICLES)	DI TOWED_VEHICLES TMD_VEHICLES
Rendezvous	(GROUND_VEHICLES GROUND_UNITS)	MIXED_IFV_DI_UNITS DI ORSMC_VEHICLE ORSMC_UNIT TOWED_VEHICLES TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
Repair	(GROUND_VEHICLES GROUND_UNITS)	MIXED_IFV_DI_UNITS DI TOWED_VEHICLES ORSMC_VEHICLE TMD_VEHICLES
Report To	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES
Road March	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES ORSMC_VEHICLE

Task Frame Name	Allowed Units/Vehicles	Disallowed Units/Vehicles
RWA Accept Unit	(RWA_VEHICLES RWA_UNITS)	
RWA Assemble	(RWA_VEHICLES RWA_UNITS RWA_ATTACK_UNITS)	
RWA Attack	(RWA_ATTACK_VEHICLES RWA_ATTACK_UNITS)	
RWA Battle Position Search	(RWA_VEHICLES RWA_UNITS RWA_ATTACK_UNITS)	
RWA Bounding Overwatch	(RWA_UNITS RWA_ATTACK_UNITS)	
RWA Fly Route	(RWA_UNITS RWA_ATTACK_UNITS RWA_VEHICLES)	
RWA Follow Simulator	(RWA_UNITS RWA_ATTACK_UNITS)	
RWA GND Laser Designation	(RWA_GND_DESIGNATOR_PAIRS)	
RWA Hover	(RWA_VEHICLES RWA_UNITS RWA_ATTACK_UNITS)	
RWA Land	(RWA_VEHICLES RWA_UNITS RWA_ATTACK_UNITS)	
RWA Laser Designation	(RWA_DESIGNATOR_PAIRS RWA_ATTACK_UNITS)	
RWA Orbit	(RWA_VEHICLES RWA_UNITS RWA_ATTACK_UNITS)	
RWA Report To	(RWA_VEHICLES RWA_UNITS)	
RWA Unassign Unit	(RWA_VEHICLES RWA_UNITS)	
SCUD	(SCUD_TEL_VEHICLES)	
Service Station	(GROUND_VEHICLES GROUND_UNITS)	MIXED_IFV_DI_UNITS DI TOWED_VEHICLES ORSMC_VEHICLE TMD_VEHICLES

Task Frame Name	Allowed Units/Vehicles	Disallowed Units/Vehicles
Set up FAC	(FAC_VEHICLES)	
SupplyRWA	(RWA_FARRP_VEHICLES RWA_FARRP_UNITS)	
Tailgate Resupply	(GROUND_UNITS GROUND_SUPPLY_VEHICLES)	TOWED_VEHICLES
Thaad Control	(THAAD_CONTROL_UNITS)	
TMD Bn Control	(TMD_BN_CONTROL_UNITS)	
Traveling Overwatch	(GROUND_UNITS)	MIXED_IFV_DI_UNITS DI TOWED_VEHICLES ARTY_VEHICLES ARTY_UNITS CRUSADER_UNITS GROUND_SUPPLY_VEHICLES ORSMC_UNIT TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS
Unassign Unit	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES
Unload Items	(GROUND_VEHICLES)	DI TOWED_VEHICLES TMD_VEHICLES
VMMD Mine Detection	(VMMD_VEHICLES)	
Wire Breach	(WIRE_BRCH_VEHICLES)	
Withdraw	(GROUND_VEHICLES GROUND_UNITS)	TOWED_VEHICLES ORSMC_VEHICLE TMD_PLATOON_UNITS TMD_VEHICLES PATRIOT_CONTROL_UNITS THAAD_CONTROL_UNITS TMD_BN_CONTROL_UNITS

Appendix B - SAF Entities

This Appendix lists the complete set of entities that are implemented in CCTT SAF 5.1.2 and ModSAF 4.0 Beta version.

B.1 CCTT SAF Entities

This following table lists the complete set of entities that are implemented for CCTT SAF version 5.1.2. For each entity, the table defines the side (BLUFOR or OPFOR) and the type of entity.

Side	Type	Entity
BLUFOR	Tank	M1_Abrams, -- M1 Abrams (105mm)
BLUFOR	Tank	M1A1_Abrams, -- M1A1 Abrams
BLUFOR	Tank	M1A2, -- M1A2
BLUFOR	Tank	M1A1_Mine_Rollers, -- M1A1 w/ Mine Rollers
BLUFOR	Tank	M1A1_Mine_Plows, -- M1A1 w/ Mine Plows
BLUFOR	Mech	M2A2, -- M2A2 Bradley Infantry Fighting Vehicle (IFV)
BLUFOR	Mech	M3A2, -- M3A2 Bradley Cavalry Fighting Vehicle (CFV)
BLUFOR	Mech	M113A3, -- FMC M113 Armored Personnel Carrier (APC), M113A3
BLUFOR	Mech	M981, -- M981 FIST-V
BLUFOR	Mech	M1064, -- M1064 120-mm Mortar Carrier
BLUFOR	Mech	M93, -- N93 NBC Recon. Vehicle
BLUFOR	Artillery	M270, -- M270 Rapid Deployment Multiple Launch Rocket System
BLUFOR	Artillery	M109A5, -- M109 155-mm SP Howitzer, M109A5
BLUFOR	Artillery	M109A6, -- M109 155-mm SP Howitzer, M109A6
BLUFOR	Engineering	M88A2, -- M88 Medium Recovery Vehicle, M88A2
BLUFOR	Engineering	M60A1, -- M60A1 AVLB
BLUFOR	Engineering	M728, -- M728 Combat Engineer Vehicle (CEV)
BLUFOR	Engineering	M9, -- M9 Armored Combat Earthmover (ACE)
BLUFOR	Engineering	M577A2, -- M577/M577A1/M577A2 Command Post
BLUFOR	Engineering	M113, -- FMC (M113) APC Ambulance
BLUFOR	Engineering	M60_Bridge, -- M60 Bridge, AVLB Launched
BLUFOR	Engineering	M60_Chassis_Without_Bridge,--

Side	Type	Entity
BLUFOR	Engineering	Mtu20_Chassis_Without_Bridge, --
BLUFOR	Engineering	Mtu20_Bridge
BLUFOR	Infantry	Observation_Post, -- USA Dismounted Infantry (non-visible), Observation Post
BLUFOR	Infantry	Usa_Soldier, -- USA Dismounted Infantry (DI) Soldier (Generic/No Weapon)
BLUFOR	Infantry	Usa_Soldier_M16A2, -- USA DI Soldier with M16A2 Assault Rifle 5.56-mm Colt
BLUFOR	Infantry	Usa_Soldier_M60, -- USA DI Soldier with General Purpose M60 7.62-mm
BLUFOR	Infantry	Usa_Soldier_M203, -- USA DI Soldier with M203 Grenade Launcher
BLUFOR	Infantry	Usa_Soldier_Dragon, -- USA DI Soldier with Dragon missile, M47
BLUFOR	Infantry	Usa_Soldier_Javelin, -- USA DI Soldier with Javelin AAWS-M
BLUFOR	Infantry	Usa_Soldier_At4, -- USA DI Soldier with AT4
BLUFOR	Infantry	Usa_Soldier_M249, -- USA DI Soldier with M249 Squad Automatic Weapon (SAW)
BLUFOR	Infantry	Usa_Soldier_Stinger, -- USA DI Soldier with Stinger, FIM-92
BLUFOR	Infantry	Usa_Soldier_Claymore,-- USA DI Soldier w/ Claymore Mine
BLUFOR	FWA	F16, -- General Dynamics F-16 Fighting Falcon, Generic
BLUFOR	FWA	A10A, -- Fairchild Republic A-10 Thunderbolt II, A-10A
BLUFOR	RWA	Ah64, -- McDonnel-Douglas AH-64 Apache
BLUFOR	RWA	Ah1S, -- Bell Model 209 Hueycobra, Seacobra, Supercobra, AH-1S
BLUFOR	RWA	Uh60A, -- Sikorsky S-70A, UH-60A Blackhawk
BLUFOR	RWA	Ch47D, -- Boeing Models 114/414, CH-47D
BLUFOR	RWA	Oh58D, -- Bell Model 406 AHIP, OH-58D Kiowa/Kiowa Warrior
BLUFOR	Logistics	M998, -- LTV HMMWV, M998 Cargo/Troop Carrier w/o winch
BLUFOR	Logistics	M966, -- LTV HMMWV, M966 TOW Missle Carrier, Basic Armor, w/ winch
BLUFOR	Logistics	M1025, -- LTV HMMWV, M1025 Armament Carrier, Basic Armor, w/o winch
BLUFOR	Logistics	M1043, -- HMMWV, M1043 Armament Carrier, Supplemental Armor, w/o winch
BLUFOR	Logistics	M1044, -- HMMWV, M1044 Armament Carrier, Supplemental Armor, w/ winch

Side	Type	Entity
BLUFOR	Logistics	Teledyne_4X4, -- Teledyne 4x4 725-kg light forces vehicle
BLUFOR	Logistics	M151, -- M151 4x4 362-kg light vehicle and variants
BLUFOR	Logistics	M58A3_Without_Rocket, -- M58/M59 Mine-clearing Charge (MICLIC) (towed), M58A3
BLUFOR	Logistics	M58A3_With_Rocket, --
BLUFOR	Logistics	M977, -- Oshkosh Heavy Expanded Mobility Tactical Truck (HEMTT), M977 Cargo
BLUFOR	Logistics	M977_Mine_Rollers, -- Oshkosh Heavy Expanded Mobility Tactical Truck (HEMTT), M977 Cargo
BLUFOR	Logistics	M977_Mine_Plows, -- Oshkosh Heavy Expanded Mobility Tactical Truck (HEMTT), M977 Cargo
BLUFOR	Logistics	M978, -- Oshkosh Heavy Expanded Mobility Tactical Truck (HEMTT), M978 Fuel-Servicing
BLUFOR	Logistics	M984E1, -- Oshkosh Heavy Expanded Mobility Tactical Truck (HEMTT), M984A1 Wrecker
BLUFOR	Logistics	M985, -- Oshkosh Heavy Expanded Mobility Tactical Truck (HEMTT), M985 Cargo
BLUFOR	Logistics	M985_Mine_Rollers, -- Oshkosh Heavy Expanded Mobility Tactical Truck (HEMTT), M985 Cargo
BLUFOR	Logistics	M985_Mine_Plows, -- Oshkosh Heavy Expanded Mobility Tactical Truck (HEMTT), M985 Cargo
BLUFOR	Logistics	M1078, -- M1078 Truck, Cargo; LMTV w/ Equipment
BLUFOR	Logistics	M1079, -- M1079 Truck , Van; LMTV w/ Equipment
BLUFOR	Logistics	M1083, -- M1083 Truck Cargo; MTV w/ Equipment
BLUFOR	Logistics	M1083_Volcano, -- M1083 Truck, Cargo; MTV w/ Equipment; w/ Vocano
BLUFOR	Logistics	M1089_Winch, -- M1089 Truck, Wrecker; MTV, w/ Equipment, w/ Winch
BLUFOR	Logistics	M1091_Winch, -- M1091 Truck, Tanker; MTV w/ Equipment, w/ Winch
BLUFOR	Logistics	M992, -- M992 Field Artillery Ammunition Support Vehicle (FAASV)
BLUFOR	Logistics	Prestock_Ammo, -- Prestock Entity (Ammo)
BLUFOR	Logistics	Prestock_Fuel, -- Prestock Entity (Fuel)
OPFOR	Tank	T80, -- T-80 MBT, T-80
OPFOR	Tank	T80_With_Mine_Plow_Roller,--
OPFOR	Tank	T80Uv, -- T-80 MBT, T-80UV

Side	Type	Entity
OPFOR	Tank	T80Uv_With_Mine_Plow_Roller, --
OPFOR	Tank	T72, -- T-72 MBT
OPFOR	Tank	T72_With_Mine_Plow_Roller, --
OPFOR	Tank	T72Bv, --T-72BV
OPFOR	Tank	T72Bv_With_Mine_Plow_Roller, --
OPFOR	Tank	T64Bv, -- T-64 MBT, T-64BV
OPFOR	Tank	T64Bv_With_Mine_Plow_Roller,--
OPFOR	Tank	T62, -- T-62 MBT, T-62
OPFOR	Tank	T62_With_Mine_Plow_Roller,--
OPFOR	Mech	Bmp1P, -- BMP-1F w/ AT-4 ATGW
OPFOR	Mech	Bmp1Kshm, -- BMP-1KShM Unarmed Command
OPFOR	Mech	Bmp2, -- BMP-2, BMP-2
OPFOR	Mech	Bmp3, -- BMP-3
OPFOR	Mech	Brdm2, -- BRDM-2 Reconnaissance Vehicle
OPFOR	Mech	Brdm2_At5_Atgm, -- BRDM-2 Reconnaissance Vehicle,BRDM-2 w/ AT-5 ATGM
OPFOR	Mech	Mtlb_1V12, -- MT-LB Tracked Vehicle, MT-LB 1V12
OPFOR	Mech	Btr60P, -- BTR-60, BTR-60P
OPFOR	Mech	Btr80, -- BTR-80, BTR-80
OPFOR	Artillery	Spa_2S3, -- M-1973 152-mm gun (2S3)(SO-152)
OPFOR	Artillery	Spa_2S1, -- M-1974 122-mm Howitzer (2S1)(SO-122)
OPFOR	Artillery	Zsu23_4, -- ZSU-23/4 Quad 23-mm AAA
OPFOR	Artillery	Sa13Sam, -- SA-13 SAM
OPFOR	Artillery	Spa_2S6, -- 2S6 Quad 30-mm/SA-19 AD System
OPFOR	Artillery	Sa15Sam, -- SA-15 SAM
OPFOR	Artillery	Spa_2S19, -- 152-mm 2S19 (aka MSTA-S)
OPFOR	Artillery	Spa_2S23, -- 120-mm Howitzer/mortar (2S23)
OPFOR	Artillery	Spa_2S31, -- BMP Chassis w/120-mm Combination Gun (2S31)
OPFOR	Artillery	D30, -- D-30 122-mm Gun-Howitzer
OPFOR	Artillery	Mt12, -- T-12/MT-12 100-mm Antitank Gun
OPFOR	Infantry	Cis_Soldier, -- CIS Dismounted Infantry (DI) Soldier (Generic/No Weapon)

Side	Type	Entity
OPFOR	Infantry	Cis_Soldier_Ak74, -- CIS DI Soldier w/ Assault Rifle AK-74 and AKS-74, 5.45-mm
OPFOR	Infantry	Cis_Soldier_Rpg7V, -- CIS DI Soldier w/ VAT Rocket Launcher RPG-7
OPFOR	Infantry	Cis_Soldier_Rpk74, -- CIS DI Soldier w/ Light RPK-74 5.45mm
OPFOR	Infantry	Cis_Soldier_Ags17, -- CIS DI Soldier w/ Plamya Launcher, 30-mm AGS-17
OPFOR	Infantry	Cis_Soldier_Sa16, -- CIS DI Soldier w/ Gimlet SA-16
OPFOR	Infantry	Cis_Soldier_At7, -- CIS DI Soldier w/ Saxhorn AT-7
OPFOR	Infantry	Cis_Soldier_At4, -- CIS DI Soldier w/ AT-4
OPFOR	Infantry	Cis_Soldier_Sa18, -- CIS DI Soldier w/ SA-18
OPFOR	FWA	Mig27, -- MiG-27 Flogger (Generic)
OPFOR	FWA	Su17, -- SU-17 Fitter (Generic)
OPFOR	FWA	Su24, -- SU-24 Fencer (Generic)
OPFOR	FWA	Su25, -- SU-25 Frogfoot (Generic)
OPFOR	RWA	Mi28, -- Mi-28 Havoc, Mi-28
OPFOR	RWA	Mi24P, -- Mi-24/25/35 Hind, Mi-24P Hind F
OPFOR	RWA	Ka50A, -- Ka-50 Hokum A Close Air Support
OPFOR	RWA	Mi8Tbk, -- Mi-8/9/17/171 Hip, Mi-8TBK (Mi-17)("E" Uprated)
OPFOR	Logistics	Bat2, -- BAT-2 Combat Engineer Vehicle
OPFOR	Logistics	Brem1, -- BREM-1 Recovery and Repair Vehicle
OPFOR	Logistics	Kmt5M, -- KMT-5M Roller/Plow
OPFOR	Logistics	Uaz469B, -- UAZ-469B 4x4 600-kg Light Vehicle
OPFOR	Logistics	Gaz66, -- GAZ-66 4x4 2000-kg Truck
OPFOR	Logistics	Kraz255B, -- KrAZ-255B 6x6 7500-kg Truck (Generic)
OPFOR	Logistics	Kraz255B_Fuel, -- KrAZ-255B 6x6 7500-kg Truck, Fuel Service
OPFOR	Logistics	Gmz_Tml, -- GMZ Tracked Mine Layer
OTHER	Tank	Chieftain, -- Chieftain MBT
OTHER	Tank	Challenger, -- Challenger MBT
OTHER	Tank	Amx30, -- AMX-30
OTHER	Tank	Amx40, -- AMX-40 MBT
OTHER	Tank	Leo1A4, -- Leopard 1 A5 MBT, A4

Side	Type	Entity
OTHER	Tank	Leo2, -- Leopard 2 MBT
OTHER	Mech	Warrior, -- FV 510 Warrior
OTHER	Mech	Amx10Rc, -- AMX-10RC Armored Car
OTHER	Mech	Amx10, -- AMX-10 IFV
OTHER	Mech	Marder2, -- Marder 2

B.2 ModSAF Entities

This following table lists the complete set of entities that are implemented for ModSAF version 4.0 Beta. For each entity, the table defines the country and the domain (ground or air).

Vehicle	Country	Domain
vehicle_US_IMF_Gateway	US	GROUND
vehicle_US_IMF_AOS	US	GROUND
vehicle_US_IMF_WAM	US	GROUND
vehicle_US_LAH_SADARM	US	GROUND
vehicle_US_PGM_Mortar	US	GROUND
vehicle_US_VIRT_REMOTE_SENTRY	US	GROUND
vehicle_US_HunterVPS	US	GROUND
vehicle_US_RC2T	US	GROUND
vehicle_US_MPL_Launcher	US	GROUND
vehicle_US_TOW_Launcher	US	GROUND
vehicle_US_AGS	US	GROUND
lifeForm_US_DI_FOFAC	US	GROUND
vehicle_US_Outrider	US	GROUND
vehicle_US_NLOS	US	GROUND
vehicle_US_M1_CPS	US	GROUND
vehicle_US_GRIZZLY	US	GROUND
vehicle_US_VMMD	US	GROUND
vehicle_US_LOSAT	US	GROUND
vehicle_US_Stingray	US	GROUND
vehicle_US_Crusader_SPH	US	GROUND
vehicle_US_Crusader_RSV	US	GROUND
vehicle_US_Crusader_M577A1	US	GROUND
vehicle_US_Crusader_M977	US	GROUND
vehicle_US_XM8	US	GROUND
vehicle_US_ASTAMIDS	US	AIR

Vehicle	Country	Domain
vehicle_Germany_JAGUAR1	FRG	GROUND
vehicle_Germany_Jaguar_Obs	FRG	GROUND
vehicle_Germany_LEO1A5	FRG	GROUND
vehicle_Germany_LEO2	FRG	GROUND
vehicle_Germany_LUCHS	FRG	GROUND
vehicle_Germany_MARDER	FRG	GROUND
vehicle_Germany_M113_Skorpion	FRG	GROUND
vehicle_Germany_M113_ambulance	FRG	GROUND
vehicle_Germany_M113_observer	FRG	GROUND
vehicle_UK_GR7_AIRSAF	UK	AIR
vehicle_USMC_AAV	US	GROUND
vehicle_USMC_AAV_C7	US	GROUND
vehicle_USMC_AAV_REC	US	GROUND
vehicle_USMC_AAV_MINE	US	GROUND
vehicle_US_Avenger	US	GROUND
vehicle_US_HMMWV	US	GROUND
vehicle_US_HMMWV_FDC	US	GROUND
vehicle_US_HMMWV_CCSIL_FO	US	GROUND
vehicle_US_HMMWV_AMBUL	US	GROUND
vehicle_US_HET_Trailer	US	GROUND
vehicle_US_M127	US	GROUND
vehicle_US_M353_MICLIC	US	GROUND
vehicle_US_M1073_ESMB	US	GROUND
vehicle_US_GBSFAAD	US	GROUND
vehicle_US_M1	US	GROUND
vehicle_US_M1A1	US	GROUND
vehicle_US_M1A2	US	GROUND
vehicle_US_AVLB	US	GROUND
vehicle_US_HAB	US	GROUND
vehicle_US_AVLM	US	GROUND
vehicle_US_M60	US	GROUND

Vehicle	Country	Domain
vehicle_US_CEV	US	GROUND
vehicle_US_ACE	US	GROUND
vehicle_US_M2	US	GROUND
vehicle_US_M3	US	GROUND
vehicle_US_M3A3	US	GROUND
vehicle_US_M93_Fox	US	GROUND
vehicle_US_M2_Stinger	US	GROUND
vehicle_US_M102	US	GROUND
vehicle_US_M105	US	GROUND
vehicle_US_M106A1	US	GROUND
vehicle_US_M252	US	GROUND
vehicle_US_M120	US	GROUND
vehicle_US_M1064	US	GROUND
vehicle_US_M109	US	GROUND
vehicle_US_M109A1	US	GROUND
vehicle_US_M109A3	US	GROUND
vehicle_US_M109A5	US	GROUND
vehicle_US_M109A6	US	GROUND
vehicle_US_M113_ambulance	US	GROUND
vehicle_US_M113_engineer	US	GROUND
vehicle_US_M113A2	US	GROUND
vehicle_US_M198	US	GROUND
vehicle_US_M270	US	GROUND
vehicle_US_M270_GAT2	US	GROUND
vehicle_US_M270_M26	US	GROUND
vehicle_US_M270_M77	US	GROUND
vehicle_US_M270_ATACMS	US	GROUND
vehicle_US_M577A1	US	GROUND
vehicle_US_M577A1_FDC	US	GROUND
vehicle_US_M577A1_CCSIL_FDC	US	GROUND
vehicle_US_M577A1_FSO	US	GROUND

Vehicle	Country	Domain
vehicle_US_M88A1	US	GROUND
vehicle_US_M911	US	GROUND
vehicle_US_M923	US	GROUND
vehicle_US_M984A1	US	GROUND
vehicle_US_M977	US	GROUND
vehicle_US_M978	US	GROUND
vehicle_US_M979	US	GROUND
vehicle_US_M981	US	GROUND
vehicle_US_M992	US	GROUND
unit_US_M978_M979_FARRP	US	GROUND
vehicle_US_M35A2_FDC	US	GROUND
vehicle_US_FAC_WHEELED	US	GROUND
vehicle_US_FWA_RECON	US	AIR
vehicle_US_A10	US	AIR
vehicle_US_F14D	US	AIR
vehicle_US_F14B_D_AIRSAF	US	AIR
vehicle_US_F16C	US	AIR
vehicle_US_F16C_AIRSAF	US	AIR
vehicle_US_A10A_AIRSAF	US	AIR
vehicle_US_F15C_AIRSAF	US	AIR
vehicle_US_F15E_AIRSAF	US	AIR
vehicle_US_F117_AIRSAF	US	AIR
vehicle_US_U2R_AIRSAF	US	AIR
vehicle_US_KC10_AIRSAF	US	AIR
vehicle_US_KC135_AIRSAF	US	AIR
vehicle_US_AC130U_H_AIRSAF	US	AIR
vehicle_US_MC130E_H_AIRSAF	US	AIR
vehicle_US_MC130N_P_AIRSAF	US	AIR
vehicle_US_E3A_AIRSAF	US	AIR
vehicle_US_E8A_AIRSAF	US	AIR
vehicle_US_RC135_AIRSAF	US	AIR

Vehicle	Country	Domain
vehicle_US_EC130E_AIRSAF	US	AIR
vehicle_US_Predator_AIRSAF	US	AIR
vehicle_US_EA6B_AIRSAF	US	AIR
vehicle_US_S3B_AIRSAF	US	AIR
vehicle_US_ES3A_AIRSAF	US	AIR
vehicle_US_E2C_AIRSAF	US	AIR
vehicle_US_P3C_AIRSAF	US	AIR
vehicle_US_AV8B_AIRSAF	US	AIR
vehicle_US_KC130_AIRSAF	US	AIR
vehicle_US_F16D	US	AIR
vehicle_US_FA18C_AIRSAF	US	AIR
vehicle_US_FA18D_AIRSAF	US	AIR
vehicle_US_RAH66	US	AIR
vehicle_US_AH64	US	AIR
vehicle_US_AH64J	US	AIR
vehicle_US_AH64D	US	AIR
vehicle_US_CH47D	US	AIR
vehicle_US_RAH66_LONGBOW	US	AIR
vehicle_US_OH58D	US	AIR
vehicle_US_UH60	US	AIR
vehicle_USSR_BM21	USSR	GROUND
vehicle_USSR_BMP1	USSR	GROUND
vehicle_USSR_BMP2	USSR	GROUND
vehicle_USSR_BRDM2	USSR	GROUND
vehicle_USSR_BTR60PU	USSR	GROUND
vehicle_USSR_BTR80	USSR	GROUND
vehicle_USSR_SA_9	USSR	GROUND
vehicle_USSR_SA_15	USSR	GROUND
vehicle_USSR_T62	USSR	GROUND
vehicle_USSR_T72M	USSR	GROUND
vehicle_USSR_T80	USSR	GROUND

Vehicle	Country	Domain
vehicle_USSR_URAL375C	USSR	GROUND
vehicle_USSR_URAL375F	USSR	GROUND
vehicle_USSR_ZIL131_FDC	USSR	GROUND
vehicle_USSR_ZSU23_4M	USSR	GROUND
vehicle_USSR_1V13	USSR	GROUND
vehicle_USSR_1V14	USSR	GROUND
vehicle_USSR_1V15	USSR	GROUND
vehicle_USSR_1V16	USSR	GROUND
vehicle_USSR_2B11	USSR	GROUND
vehicle_USSR_2S12	USSR	GROUND
vehicle_USSR_2S1	USSR	GROUND
vehicle_USSR_2S6	USSR	GROUND
vehicle_USSR_2S19	USSR	GROUND
vehicle_USSR_XM375S	USSR	GROUND
vehicle_USSR_XMG1S	USSR	GROUND
vehicle_USSR_SA_6_FCR	USSR	GROUND
vehicle_USSR_SA_6_TEL	USSR	GROUND
vehicle_USSR_XMLTS	USSR	GROUND
vehicle_USSR_XMTSS	USSR	GROUND
vehicle_SWA_MAZ543	USSR	GROUND
vehicle_USSR_FWA_RECON	USSR	AIR
vehicle_USSR_MIG23_AIRSAF	USSR	AIR
vehicle_USSR_MIG27	USSR	AIR
vehicle_USSR_MIG27D	USSR	AIR
vehicle_USSR_MIG29	USSR	AIR
vehicle_USSR_MIG29_AIRSAF	USSR	AIR
vehicle_USSR_Su17_AIRSAF	USSR	AIR
vehicle_USSR_Su25	USSR	AIR
vehicle_USSR_Su25_AIRSAF	USSR	AIR
vehicle_USSR_Su27_AIRSAF	USSR	AIR
vehicle_USSR_F1_AIRSAF	USSR	AIR

Vehicle	Country	Domain
vehicle_USSR_Mi8	USSR	AIR
vehicle_USSR_Mi24	USSR	AIR
vehicle_USSR_Mi28	USSR	AIR
vehicle_USSR_Ka50	USSR	AIR
structure_AMMO_PALLET_GERMAN_AT2	FRG	GROUND
structure_AMMO_PALLET_GERMAN_120HEAT	FRG	GROUND
structure_AMMO_PALLET_GERMAN_120SABOT	FRG	GROUND
structure_AMMO_PALLET_GERMAN_DM63	FRG	GROUND
structure_AMMO_PALLET_GERMAN_DM81	FRG	GROUND
structure_AMMO_PALLET_GERMAN_HOT	FRG	GROUND
structure_AMMO_PALLET_GERMAN_MILAN	FRG	GROUND
structure_AMMO_PALLET_GERMAN_PANZERFAUST	FRG	GROUND
structure_AMMO_PALLET_US_155_SADARM	US	GROUND
structure_AMMO_PALLET_US_DRAGON	US	GROUND
structure_AMMO_PALLET_US_HELLFIRE	US	GROUND
structure_AMMO_PALLET_US_HELLFIRE_RF	US	GROUND
structure_AMMO_PALLET_US_HYDRA70M151	US	GROUND
structure_AMMO_PALLET_US_HYDRA70M261	US	GROUND
structure_AMMO_PALLET_US_JAVELIN	US	GROUND
structure_AMMO_PALLET_US_L8A1	US	GROUND
structure_AMMO_PALLET_US_LOSAT_Missile	US	GROUND
structure_AMMO_PALLET_US_M1	US	GROUND
structure_AMMO_PALLET_US_M107	US	GROUND
structure_AMMO_PALLET_US_M110E2	US	GROUND
structure_AMMO_PALLET_US_M116A1	US	GROUND
structure_AMMO_PALLET_US_M121A1	US	GROUND
structure_AMMO_PALLET_US_M135	US	GROUND
structure_AMMO_PALLET_US_M240	US	GROUND
structure_AMMO_PALLET_US_M329A1	US	GROUND
structure_AMMO_PALLET_US_M329A2	US	GROUND
structure_AMMO_PALLET_US_M33	US	GROUND

Vehicle	Country	Domain
structure_AMMO_PALLET_US_M392A2	US	GROUND
structure_AMMO_PALLET_US_M449A1	US	GROUND
structure_AMMO_PALLET_US_M456A1	US	GROUND
structure_AMMO_PALLET_US_M483A1	US	GROUND
structure_AMMO_PALLET_US_M485A2	US	GROUND
structure_AMMO_PALLET_US_M548	US	GROUND
structure_AMMO_PALLET_US_M549A1	US	GROUND
structure_AMMO_PALLET_US_M58_chargelet	US	GROUND
structure_AMMO_PALLET_US_M59	US	GROUND
structure_AMMO_PALLET_US_M687	US	GROUND
structure_AMMO_PALLET_US_M692	US	GROUND
structure_AMMO_PALLET_US_M712	US	GROUND
structure_AMMO_PALLET_US_M718	US	GROUND
structure_AMMO_PALLET_US_M731	US	GROUND
structure_AMMO_PALLET_US_M741	US	GROUND
structure_AMMO_PALLET_US_M76	US	GROUND
structure_AMMO_PALLET_US_M77_M26	US	GROUND
structure_AMMO_PALLET_US_M789	US	GROUND
structure_AMMO_PALLET_US_M792	US	GROUND
structure_AMMO_PALLET_US_M795	US	GROUND
structure_AMMO_PALLET_US_M804	US	GROUND
structure_AMMO_PALLET_US_M819A1	US	GROUND
structure_AMMO_PALLET_US_M821A1	US	GROUND
structure_AMMO_PALLET_US_M825	US	GROUND
structure_AMMO_PALLET_US_M829A2	US	GROUND
structure_AMMO_PALLET_US_M830A1	US	GROUND
structure_AMMO_PALLET_US_M853A1	US	GROUND
structure_AMMO_PALLET_US_M855	US	GROUND
structure_AMMO_PALLET_US_M864	US	GROUND
structure_AMMO_PALLET_US_M889A1	US	GROUND
structure_AMMO_PALLET_US_M919	US	GROUND

Vehicle	Country	Domain
structure_AMMO_PALLET_US_M933_934	US	GROUND
structure_AMMO_PALLET_US_M940	US	GROUND
structure_AMMO_PALLET_US_MK22	US	GROUND
structure_AMMO_PALLET_US_MLRS_MSTAR	US	GROUND
structure_AMMO_PALLET_US_MLRS_SAD	US	GROUND
structure_AMMO_PALLET_US_MLRS_SAD_ER	US	GROUND
structure_AMMO_PALLET_US_MX943	US	GROUND
structure_AMMO_PALLET_US_NLOS	US	GROUND
structure_AMMO_PALLET_US_PGMM	US	GROUND
structure_AMMO_PALLET_US_STINGER	US	GROUND
structure_AMMO_PALLET_US_TOW	US	GROUND
structure_AMMO_PALLET_US_XM867	US	GROUND
structure_AMMO_PALLET_US_XM898	US	GROUND
structure_AMMO_PALLET_US_XM951	US	GROUND
structure_AMMO_PALLET_US_M971	US	GROUND
structure_AMMO_PALLET_US_XM982	US	GROUND
structure_AMMO_PALLET_US_XR_M77	US	GROUND
structure_MINE_PALLET_munition_US_BLU91B	US	GROUND
structure_MINE_PALLET_munition_US_BLU92B	US	GROUND
structure_MINE_PALLET_munition_US_HORNET	US	GROUND
structure_MINE_PALLET_munition_US_M14	US	GROUND
structure_MINE_PALLET_munition_US_M15	US	GROUND
structure_MINE_PALLET_munition_US_M16A2	US	GROUND
structure_MINE_PALLET_munition_US_M18A1	US	GROUND
structure_MINE_PALLET_munition_US_M19	US	GROUND
structure_MINE_PALLET_munition_US_M21	US	GROUND
structure_MINE_PALLET_munition_US_M67	US	GROUND
structure_MINE_PALLET_munition_US_M70	US	GROUND
structure_MINE_PALLET_munition_US_M72	US	GROUND
structure_MINE_PALLET_munition_US_M73_mine	US	GROUND
structure_MINE_PALLET_munition_US_M74	US	GROUND

Vehicle	Country	Domain
structure_MINE_PALLET_munition_US_M75	US	GROUND
structure_MINE_PALLET_munition_US_M76_mine	US	GROUND
structure_MINE_PALLET_munition_US_M77_mine	US	GROUND
structure_MINE_PALLET_munition_US_SLAM	US	GROUND
structure_MINE_PALLET_munition_US_VOLCANO_AP	US	GROUND
structure_MINE_PALLET_munition_US_VOLCANO_AT	US	GROUND
structure_MINE_PALLET_MECH_Volcano	US	GROUND
structure_MINE_PALLET_MOPMS	US	GROUND
structure_AMMO_PALLET_USSR_120MM_MCS_E	USSR	GROUND
structure_AMMO_PALLET_USSR_125HE_Frag	USSR	GROUND
structure_AMMO_PALLET_USSR_125HEAT	USSR	GROUND
structure_AMMO_PALLET_USSR_125SABOT	USSR	GROUND
structure_AMMO_PALLET_USSR_127AA	USSR	GROUND
structure_AMMO_PALLET_USSR_127MG	USSR	GROUND
structure_AMMO_PALLET_USSR_145MG	USSR	GROUND
structure_AMMO_PALLET_USSR_23AP	USSR	GROUND
structure_AMMO_PALLET_USSR_3023_DPICM	USSR	GROUND
structure_AMMO_PALLET_USSR_30HE	USSR	GROUND
structure_AMMO_PALLET_USSR_30HE_2S6	USSR	GROUND
structure_AMMO_PALLET_USSR_30HE_BMP2	USSR	GROUND
structure_AMMO_PALLET_USSR_30HE_Ka50	USSR	GROUND
structure_AMMO_PALLET_USSR_30SABOT_BMP2	USSR	GROUND
structure_AMMO_PALLET_USSR_30SABOT_Ka50	USSR	GROUND
structure_AMMO_PALLET_USSR_73HEAT	USSR	GROUND
structure_AMMO_PALLET_USSR_9K25	USSR	GROUND
structure_AMMO_PALLET_USSR_ATP_H2	USSR	GROUND
structure_AMMO_PALLET_USSR_BK_M_HEAT_FS	USSR	GROUND
structure_AMMO_PALLET_USSR_D	USSR	GROUND
structure_AMMO_PALLET_USSR_D_462	USSR	GROUND
structure_AMMO_PALLET_USSR_D_540	USSR	GROUND
structure_AMMO_PALLET_USSR_ER540	USSR	GROUND

Vehicle	Country	Domain
structure_AMMO_PALLET_USSR_GASKIN	USSR	GROUND
structure_AMMO_PALLET_USSR_GAUNTLET	USSR	GROUND
structure_AMMO_PALLET_USSR_HR843A	USSR	GROUND
structure_AMMO_PALLET_USSR_MASD_DPICM	USSR	GROUND
structure_AMMO_PALLET_USSR_MCS_SMART	USSR	GROUND
structure_AMMO_PALLET_USSR_MMB_120_SMART	USSR	GROUND
structure_AMMO_PALLET_USSR_M_21_OF	USSR	GROUND
structure_AMMO_PALLET_USSR_OF843B	USSR	GROUND
structure_AMMO_PALLET_USSR_OF_45	USSR	GROUND
structure_AMMO_PALLET_USSR_OF_462	USSR	GROUND
structure_AMMO_PALLET_USSR_OF_540	USSR	GROUND
structure_AMMO_PALLET_USSR_OF_61	USSR	GROUND
structure_AMMO_PALLET_USSR_PS	USSR	GROUND
structure_AMMO_PALLET_USSR_ROCKET_64	USSR	GROUND
structure_AMMO_PALLET_USSR_SAGGER	USSR	GROUND
structure_AMMO_PALLET_USSR_SA_16	USSR	GROUND
structure_AMMO_PALLET_USSR_SA_19	USSR	GROUND
structure_AMMO_PALLET_USSR_SONGSTER	USSR	GROUND
structure_AMMO_PALLET_USSR_SPANDREL	USSR	GROUND
structure_AMMO_PALLET_USSR_SPIGOT	USSR	GROUND
structure_AMMO_PALLET_USSR_SPIRAL	USSR	GROUND
structure_AMMO_PALLET_USSR_UV32_57	USSR	GROUND
structure_AMMO_PALLET_USSR_VOG_17M	USSR	GROUND
structure_MINE_PALLET_munition_USSR_MON100	USSR	GROUND
structure_MINE_PALLET_munition_USSR_MON200	USSR	GROUND
structure_MINE_PALLET_munition_USSR_MON50	USSR	GROUND
structure_MINE_PALLET_munition_USSR_OZM3	USSR	GROUND
structure_MINE_PALLET_munition_USSR_PGMDM	USSR	GROUND
structure_MINE_PALLET_munition_USSR_PMK40	USSR	GROUND
structure_MINE_PALLET_munition_USSR_PMN	USSR	GROUND
structure_MINE_PALLET_munition_USSR_POMZ2	USSR	GROUND

Vehicle	Country	Domain
structure_MINE_PALLET_munition_USSR_PTMIK	USSR	GROUND
structure_MINE_PALLET_munition_USSR_TM46	USSR	GROUND
structure_MINE_PALLET_munition_USSR_TM57	USSR	GROUND
structure_MINE_PALLET_munition_USSR_TM62	USSR	GROUND
structure_MINE_PALLET_munition_USSR_TMD_B	USSR	GROUND
structure_MINE_PALLET_munition_USSR_TMK2	USSR	GROUND
structure_MINE_PALLET_munition_USSR_TMN46	USSR	GROUND
structure_MINE_PALLET_munition_USSR_TMRP6	USSR	GROUND
structure_MINE_PALLET_munition_USSR_YAM5	USSR	GROUND
structure_AMMO_PALLET		GROUND
structure_MINE_PALLET_munition_YUGO_IU		GROUND
structure_MINE_PALLET_munition_YUGO_TMRP6		GROUND
structure_MINE_PALLET_munition_EGYPT_M80		GROUND
structure_MINE_PALLET_munition_ITALY_SB81		GROUND
structure_MINE_PALLET_munition_ITALY_SBMV		GROUND
structure_MINE_PALLET_munition_ITALY_SH55		GROUND
structure_MINE_PALLET_munition_ITALY_TC24		GROUND
structure_MINE_PALLET_munition_ITALY_TC6		GROUND
structure_MINE_PALLET_munition_ITALY_TCE6		GROUND
structure_MINE_PALLET_munition_ITALY_VS16		GROUND
structure_MINE_PALLET_munition_ITALY_VSHCT		GROUND
structure_MINE_PALLET_munition_OTHER_MIACAH		GROUND
structure_MINE_PALLET_munition_OTHER_P2MK3		GROUND
structure_MINE_PALLET_munition_OTHER_PTMIBAIII		GROUND
structure_MINE_PALLET_munition_UK_L14A1		GROUND
structure_MINE_PALLET_munition_UK_MK7		GROUND
structure_RPARTS_PALLET		GROUND
structure_CONST_MATL_PALLET		GROUND

Appendix C - Migration Assessments

This appendix contains the detailed migration approach assessments for integrating from the CCTT SAF baseline or the ModSAF baseline into a OneSAF Testbed baseline. This appendix provides the behaviors, system, synthetic natural environment, physical models and user computer interface assessments.

C.1 Behaviors Assessment

The behavior assessment considered six migration options listed as follows:

- Option 1: Re-implement missing behaviors from ModSAF into existing CCTT infrastructure (no modification to the infrastructure)
- Option 2: Re-implement missing behaviors from CCTT SAF into existing ModSAF infrastructure (no modification to the infrastructure)
- Option 3: Develop new unified behavior representation and implement all behaviors in it
- Option 4: Extend ModSAF with additional infrastructure to ease re-implementation of CCTT behaviors within ModSAF
- Option 5: Extend CCTT SAF with additional infrastructure to ease re-implementation of ModSAF behaviors within CCTT SAF
- Option 6: Develop capability to run both ModSAF and CCTT SAF behaviors within the OneSAF testbed

Behavior Assessment: Option 1(Re-Implement ModSAF Behaviors in CCTT SAF)

Technical Description

Start with CCTT SAF as the baseline and migrate approximately 106 ModSAF behaviors, 240 ModSAF units, and ModSAF platforms into CCTT SAF. CCTT SAF will only be modified as necessary to support the incorporation of the additional ModSAF behaviors, units, and platforms. It will not be modified to incorporate any other ModSAF functionality or features. The following list provides a more detailed look at the approach for this option.

- Maintain the existing CCTT SAF Behavioral Architecture
- Maintain the existing set of 276 CCTT SAF behaviors
- Add each additional ModSAF unit as a distinct unit class (where a unit class is currently represented as an individual Ada package)
- Add each additional ModSAF behavior as a common FSM (where common FSMs are currently collected in separate Ada packages that can be accessed by all units)
    - Where possible, generalize existing CCTT SAF behaviors to provide ModSAF functionality

- Extend the existing CCTT SAF GUI to support the additional ModSAF units and behaviors
- Extend the SEOD to support the orders and reports needed by the additional ModSAF units and behaviors
- Extend the CCTT SAF GMI to support the additional ModSAF platforms and platform capabilities
- Extend the CCTT SAF crew level behaviors to support both additional ModSAF unit-level behaviors and additional ModSAF platforms and platform capabilities
- Extend the Order Decomposition capability to support the additional ModSAF units and behaviors
- Re-engineer CCTT SAF behaviors and behaviors architecture into C

Advantages of using this Approach

The principal advantages of this approach are low cost and low technical risk. Although ModSAF has far more units and platforms than CCTT SAF, the cost of implementing either a unit or platform is substantially less than the cost of implementing a user-assignable behavior (in both systems). Since CCTT SAF has far more user-assignable behaviors than ModSAF, the cost of migrating behaviors, units, and platforms from ModSAF to CCTT SAF is actually less than the cost of migrating behaviors, units, and platforms from CCTT SAF to ModSAF. Also, although ModSAF behaviors are implemented in a subsumptive architecture, there is no reason to believe that those same behaviors can't be implemented in a cooperative architecture. This assertion is supported by the fact that there is overlap between the behaviors that have been implemented in CCTT SAF and ModSAF (e.g., both systems have platoon, company, and battalion behaviors, both systems have traveling, bounding overwatch, and traveling overwatch movement behaviors, etc.)

Disadvantages of using this Approach

The principal disadvantage to this approach is that it does not meet ModSAF's extensibility requirements. The resulting baseline will not have:

- Architectural integrity – in general, FSMs will be collected in libraries rather than being represented as distinct libraries
- Dynamic debugging – debugging information will only be available when the FSM developer builds in the appropriate debugging statements
- Run-time modifications to FSMs – only certain pre-defined run-time modifications to unit behaviors will be allowed (e.g., speed, spacing)
- User-configurable reactions – each unit's reactions will be implemented in a dispatch table
- Tools for generating FSMs – developers will continue to build FSMs by hand
- Data-driven GUI – modifications to the GUI will have to be implemented in code
- Status monitor – there will be no status monitor to provide details of executing behaviors

Technical Feasibility

This approach has very little technical risk and is therefore technically feasible. The strategy for migrating ModSAF units, behaviors, and platforms to CCTT SAF will have to address the change in behaviors architecture (cooperative vs. subsumptive), and the change in data structures (public and private task data vs. SEOD data). However, once a conversion technique has been developed, that technique can be applied iteratively to the ModSAF units, behaviors, and platforms.

Requirements satisfaction

This approach satisfies all of CCTT SAF's requirements since it is CCTT SAF with an extended set of behaviors. This approach does not satisfy ModSAF's extensibility requirements, and the specific limitations are listed in under Disadvantages.

Performance

The performance requirements are met by this approach since it represents the CCTT baseline and CCTT currently has the most stringent requirements.

Schedule

This approach has very little schedule risk because the only long-lead item is the task to establish the technique for converting ModSAF units, behaviors, and platforms into CCTT SAF units, behaviors, and platforms. Once this task has been accomplished, developers simply apply this technique iteratively.

Reliability

This approach should result in a baseline that is at least as reliable as CCTT SAF.

Scalability

This approach should result in a baseline that is at least as scalable as CCTT SAF.

Behavior Requirements Satisfaction

Since this approach fully implements both CCTT SAF and ModSAF behaviors, the behavior requirements of both systems are fully satisfied.

Behavior V&V

Since this approach retains the CCTT SAF behaviors without modification, the only V&V effort should be associated with the conversion of CCTT SAF behaviors from Ada to C. This effort should be minimal when compared to the other approaches. Although the V&V agents may have to re-validate each behavior, there should be fewer defects resulting from a programming language conversion than from an architecture conversion.

Extensibility

This approach does not address extensibility. The specific limitations are listed under Disadvantages.

Baseline Maintenance

This approach does not adversely impact baseline maintenance beyond the existing problems associated with either ModSAF or CCTT SAF.

Adaptability to new Technology and Architectures

This approach will result in a OneSAF testbed that has the same capability as CCTT SAF to adapt to new technology and architectures.

Ease of coordination and integration of multiple developers

This approach will result in a OneSAF testbed that has the same capability as CCTT SAF to support the coordination and integration of multiple CCTT SAF developers. This approach will not support the coordination and integration of any ModSAF developers. ModSAF development and OneSAF testbed development would proceed along separate development paths.

Enumeration of capabilities to migrate

The Technical Description provides a list of capabilities that must be migrated and re-worked to support this approach.

Effort to migrate related to capabilities

ModSAF Behaviors Incorporation	
New Behaviors	106
Overlap	25%
Per CIS effort (weeks)	7
Weeks effort	578
Months effort	144

ModSAF Units Incorporation	
New Units	240
Per CIS effort (weeks)	0.5
Weeks effort	120
Months effort	30

CCTT Behaviors Rework

Behaviors Rework	86
Months effort	86

Total Months Effort	260

Assumptions on migration environment

The following list outlines the assumptions for this approach:

- The existing CCTT SAF behaviors will not have to be modified in any way (e.g., no impacts from GMI, SEOD, or vehicle behavior extensions)
- The existing CCTT SAF crew behaviors can be extended to support the additional ModSAF platforms and platform capabilities
- The CCTT SAF GUI will not need any new widgets to support ModSAF behaviors
- There are no ModSAF behaviors which rely so thoroughly on a subsumptive architecture that they can't be implemented in a cooperative architecture (where "can't" means "takes too much resources")
- The ModSAF behaviors can function within the information context of reports and orders
- ModSAF background tasks can be implemented as built in behaviors of the Unit class or specific doctrinal unit classes
- ModSAF behaviors do not have to be validated

Assessment

The following table provides an assessment of this option for the specified criteria.

Criteria	Low (1)	Medium (5)	High (10)	Score
Technical Feasibility risk	The technical feasibility is rated very good.	The technical feasibility is good but the migration involves some very difficult technical issues.	The technical feasibility is poor. The migration technical is very difficult or the approach has never been tried before.	3
Program Risk	There is little risk of the migration approach	There is a moderate risk of the migration approach	The migration approach is very risky and has never been attempted before	2
Impact to existing ModSAF user community	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically	7

		come with moderate effort	challenging	
Impact to existing ModSAF development community (e.g., STOW)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	7
Impact to existing CCTT program	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Impact to current CCTT development (e.g., FBCB2)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Adaptability to new technology risk	The resulting infrastructure or environment can be easily extended	The resulting infrastructure does not support extension and will require additional enhancements to support future modification	The resulting infrastructure is not unified and should be completely redesigned to support extensibility	10
Relative Cost				260 mm

Criteria Assessment Justification

The following sections provide the rationale for the assigned scores.

Technical Feasibility

The score of 3 was assigned to this criterion since the migration of behaviors from ModSAF to CCTT SAF should be straightforward. However, there may be some issues related to representation of subsumptive approaches in a cooperative system.

Program Risk

The score of 2 was assigned to this criterion since there is very little program risk. Once the initial technical issues associated with converting ModSAF units, behaviors, and platforms into CCTT SAF units, behaviors, and platforms have been resolved, the conversion process should be easy to measure and track.

Existing Program Impact

The score of '7' was assigned to the ModSAF impacts since this will have a substantial impact on all ModSAF based programs. In short, ModSAF developers would have to learn how to develop behaviors for CCTT SAF. The impacts to CCTT SAF current and future efforts are scored as '1' since the base system is CCTT SAF.

Adaptability to new technology

The score of 10 was assigned to this criteria since the resulting infrastructure will not support ModSAF's extensibility requirement.

Cost

Refer to the effort to migrate section.

Behavior Assessment: Option 2 (Re-Implement CCTT Behaviors in ModSAF)

Technical Description

ModSAF will be used as the baseline. Take the existing CCTT CISs and re-implement any missing functionality in them as ModSAF behaviors. Not all CISs would require new ModSAF behaviors because there is expected overlap (these are common behaviors). No architectural changes will be made to support the new CCTT SAF behaviors.

Models and/or vehicles in CCTT SAF that are not in ModSAF will have to be implemented as well.

Advantages of using this approach

- ModSAF behaviors are already done.
- Use ModSAF behaviors as base for CIS migrated behaviors.
- Upgrade of ModSAF building blocks to support CCTT behaviors may improve the quality of some ModSAF behaviors.
- Using a ModSAF base will ease re-baselining with new ModSAF technologies.
  - Efficiency improvements
  - Behavior improvements (RWA, new Chem/Bio for STOW, Countermine (JCOS))
- No Ada (easier, cheaper, and faster development environment without Ada)
- Takes advantage of the extensibility of ModSAF and it remains extensible.
- Maintains composable characteristics of ModSAF.

Disadvantages of using this approach

- Total number of CCTT SAF behaviors x time to implement = large cost
- Re-implementation of existing VV&Aed CCTT SAF behaviors.
- Makes it difficult to keep up with CCTT enhancements.
- Must V&V all new and common behaviors again.
- Modification of building blocks increases the risk of breaking existing ModSAF capabilities.
- Do not have full compliance with all CCTT requirements.
  - Order decomp vs. ModSAF dynamic decomposition
  - Check point (no approach identified)

Technical Feasibility

Done lots of behaviors, so technically not challenging, but there will be lots of unsatisfied capabilities.

There will be risk in getting everything done in time (by "brute force").

Program Risk

Requirements satisfaction

ModSAF requirements satisfied (by default)
CCTT deficient according to list, below

FSM Language As-Is
Order Decomp not done
Distributed Units not done
Checkpointing not done
Orders (in general) not done
Logging of orders not done
Inefficiencies
No timing mechanisms
Difficult to handle subordinate management
Difficult to handle leader dying
Difficult to handle taskorg changes.

Performance

It is unknown whether the ModSAF architecture satisfies the performance requirements needed by CCTT. There is no way to currently evaluate this.

Schedule

Estimate 425 mm. Plus, add V&V.

Reliability

As with all approaches outlined, there are echelon scalability limitations to using an FSM approach for behaviors. This approach does not make the problem worse.

Scalability

If interpreted as echelon, all current bottoms-up architecture approaches have echelon-scaling limitations.

Existing Program Impact

Behavior Requirements Satisfaction

See above.

Behavior V&V

All re-implemented CISs (and their building blocks) would have to be V&Ved. We would recommend doing the V&V as was done originally for CCTT. Regression testing would have limited utility because the tests would be against a different baseline. There would need to be some interpretation of the results, and you could not take advantage of previous testing results.

Extensibility

No ModSAF extensibility lost. CCTT behaviors now extensible in the same style that ModSAF is.

Baseline Maintenance

There will be low maintenance because there will generally be only one implementation of each behavior. In some hopefully rare cases, there may be a CCTT version and a ModSAF version of the same behavior. Hopefully that is near zero.

Adaptability to new Technology and Architectures

New technology implemented in ModSAF derivatives such as Joint SAF will have a straightforward migration path to this unified architecture. New technology implemented in CCTT SAF (such as FBCB2) will require re-implementation in OneSAF.

Development	Adaptability Assessment
Ongoing ModSAF development	Good
Ongoing CCTT SAF development	Poor (re-implementation)
Existing and ongoing STOW enhancements	Good
Joint SAF	Good
CFOR	Good
CCSIL	Good
Command Talk	Good
OneSAF R&D	Unknown

Ease of coordination and integration of multiple developers

The OneSAF testbed can leverage off of the existing ModSAF and Joint SAF approaches for parallel distributed development.

Enumeration of capabilities to migrate

- No new architecture.
- Implementation of CCTT-Training mode vs. unconstrained mode.

- Re-implementation of all CCTT CISs that are not common.
- Upgrade to building blocks. Includes:
    - Movement
    - Shooting

Effort to migrate related to capabilities

This is a development activity.

Building Block Enhancements	Estimate (Man-Months)
Misc.	12
Movement	18
Shooting	6
Total	**36**

CISs	CIS-CIS Overlap	Per-CIS Effort Weeks	CIS Man-Weeks	CIS Man-Months
346	75%	6.0	1557.5	**389.25**

CIS Development Man-Months	Architecture Enhancements Man-Months	Bulding Block Additions Man-Months	Total Man-Months
324.38	0	36	**425.25**

Assumptions on migration environment

- Expect to be able to reuse some existing ModSAF behaviors with CCTT SAF behavior re-implementation.
- Reuse as many physical models as possible.

Assessment

The following table provides the scores for the specified criteria.

Criteria	Low (1)	Medium (5)	High (10)	Score
Technical Feasibility Risk	The technical feasibility is rated very good.	The technical feasibility is good but the migration involves some very difficult technical issues.	The technical feasibility is poor. The migration technical is very difficult or the approach has never been tried before.	1

Program Risk	There is little risk of the migration approach	There is a moderate risk of the migration approach	The migration approach is very risky and has never been attempted before	7
Impact to existing ModSAF user community	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Impact to existing ModSAF development community (e.g., STOW)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Impact to existing CCTT program	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	7
Impact to current CCTT development (e.g., FBCB2)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	7
Adaptability to new technology risk	The resulting infrastructure or environment can be easily extended	The resulting infrastructure does not support extension and will require additional	The resulting infrastructure is not unified and should be completely redesigned to support extensibility	3

		enhancements to support future modification		
Cost				425 mm

Behavior Assessment: Option 3 (New Architecture)

Technical Description

This effort results in a new behavioral design that is not an extension to either architecture. This architecture would result in a new set of primitives, controlling capabilities, and behavioral representations. Using this approach, all behaviors would be re-represented.

Advantages of using this Approach

The architecture is designed to satisfy all the requirements.

Disadvantages of using the approaches

- Cost would be expected to be higher
- There would be schedule risk due to the increased new development requirements
- V&V effort would be increased since the behavior suite is newly developed

Technical Feasibility

The technical feasibility of this approach is very good. This approach assumes that the resulting design is specifically tailored to the existing set of requirements and encompasses known future usages. The resulting system should be an optimal solution set for this focus area.

Requirements satisfaction

It is expected that this approach would satisfy all requirements placed on OneSAF testbed since the design would be tailored to this combined set.

Performance

Performance requirements for CCTT SAF would be fully met by this architecture since this would be a driving requirement for new development.

Schedule

There would be considerable schedule risk since this is a new design effort. All SLOC estimates would have to be applied to a full up development phase

Assessment

This option represents the suggestion of the summer study that was the development of a new architectural approach. This is not feasible considering the risks of a new development and the incorporation of 350+ behaviors. For this reason, this option was discarded and no estimation was performed on the necessary tasks.

Behavior Assessment: Option 4 (Extend ModSAF with additional infrastructure to ease re-implementation of CCTT behaviors within ModSAF)

Technical Description

This approach starts with ModSAF as the initial baseline. We will improve the behavioral architecture to facilitate the re-implementation of the CCTT SAF behaviors based upon existing CIS documents. There will be minimal or no changes to the current ModSAF behaviors, yet all of the CCTT SAF behaviors will be re-implemented within the ModSAF behavior architecture (with the appropriate extensions). The benefit of this option over 2 (implementing all CCTT CISs within the unmodified ModSAF architecture) is to make the re-implementation of the CCTT behaviors easier and cheaper.

FSM Enhancements

Additional FSM Abstraction Support

There are a number of capabilities that can be improved in the FSM language through the use of additional abstraction. These include management of subordinates (attrition, leader changes, TO changes), timing management, defaulting of events, storage management for created objects, and testing support for FSMs.

Designs for these modifications have been circulating between different members of the ModSAF development community for some time.

Event Based Task Transitions

The largest problem with supporting distributed units in ModSAF is the race condition where a task was monitoring the state of another behavior on a different machine. The problem is that if two state changes occur very quickly, and you miss the first state change due to dropping a PO packet, you will never get notified about the original change (which may be required in the FSM logic).

If we made the ending of the task an event (it already is an object-changed event) and we registered for that event, we would no longer need to poll for events. This will improve efficiency of tasks. Since we are almost always checking for the end of a task (except in the case of movement with utraveling and vmove), the state will not be changing after it goes to the end state, there is no problem of missing an intermediate change. Since we have already agreed that we need to re-visit movement in the upgrade building blocks section, this utraveling/vmove coupling will be on the list of things to address.

Potential to add additional CCTT-specific FSM abstractions

The best abstraction would allow us to adapt CCTT's implementation of behaviors or CISs directly into the abstracted approach.

- IST did a paper on converting CISs to ModSAF code, but they used a brute-force approach. This will provide little of use in supporting an improved FSM abstraction.
- It is unclear that you can reduce behavior development arbitrarily. Perhaps no new abstractions would speed up the 5.0-week effort per CIS behavior that we assume with FSM language cleanup.
- The CCTT FSMs rely on the CCTT behavior architecture, which is substantially different from the ModSAF behavior architecture. It is difficult to imagine an abstraction that could smooth all the differences between these architectures.
- We may be limited to relying on behavior developers discovering opportunities for more abstractions during initial CIS development.

The ModSAF behavior assessment team believes that there is no feasibility in developing a new FSM language to support CCTT-specific abstraction.

Checkpointing

To support CCTT checkpointing requirements, ModSAF will develop a capability to periodically save scenarios, and to restore scenarios with the same DIS and PO IDs. This approach is comparable to CCTT's existing checkpointing implementation.

Order Decomposition

ModSAF will develop an order decomposition technique similar to that supported by CCTT. The capability to support runtime-decomposed orders will be retained in a backward-compatible manner so as to not disrupt the existing ModSAF behaviors.

Providing an order decomposition approach in ModSAF will be important in order to:

- Support CCTT users' existing insight into the decomposition process
- Support CCTT's existing upfront user editability of decomposed orders
- Support CCTT's capability to save modified decomposed orders.

Advantages of using this approach

This approach has the same advantages as option 2:

- ModSAF behaviors are already done.
- The approach uses ModSAF behaviors as base for CIS migrated behaviors.
- The upgrade of ModSAF building blocks to support CCTT behaviors may improve the quality of some ModSAF behaviors.
- Using a ModSAF base will ease re-baselining with new ModSAF technologies.
  - Efficiency improvements (STOW)
  - Behavior improvements (RWA, new Chem/Bio for STOW, Countermine (JCOS))
- No Ada
- Takes advantage of the extensibility of ModSAF and it remains extensible.
- Maintains composable characteristics of ModSAF.

In addition, new benefits of this approach includes:

- It satisfies the CCTT SAF and ModSAF requirements.
- It cleans up, extends, and unifies the current ModSAF behavior architecture.
- It provides a more open architecture and is more extensible.
- It is cost effective; It will be easier and cheaper to implement CIS behaviors within the extended ModSAF architecture.
- It implements potential runtime efficiency improvements.

Disadvantages of using this approach

The disadvantages of this approach are similar to option 2:

- Total number of CCTT SAF behaviors x time to implement = large cost
- Re-implementation of existing VV&A'd CCTT SAF behaviors.
- Makes it difficult to keep up with CCTT enhancements.
- Must V&V all new and common behaviors again.
- Modification of building blocks increases the risk of breaking existing ModSAF capabilities.

In addition:

- The extension to the existing ModSAF behavioral architecture needs to be done up front before any behaviors can be re-implemented. This presents an up-front scheduling problem that is mitigated by reducing the total schedule.

Technical Feasibility

The architectural enhancements have been outlined and there is little risk in the implementation of these enhancements. Since we will be using the same AAFSM language with extensions and we have already developed many behaviors in this architecture, the technical risk of this approach is minimal. The only enhancement to which we do not yet have a design is the approach for order decomposition, however this is low risk since order decomposition is only an implementation that supports a requirement, not a requirement itself.

Program Risk

Requirements satisfaction

The ModSAF and CCTT SAF requirements will be satisfied.

Performance

As in option 2, it is unknown whether the ModSAF architecture satisfies the performance requirements needed by CCTT. There is no way to currently evaluate this. However, some of the architectural enhancements will support more event-driven behaviors and hence should allow for better total performance.

Schedule

This approach has minimized the schedule risk from option 2. The architectural changes have reduced the estimate down by approximately 46 staff months. However, the total effort is substantial (379 staff months).

Reliability

As in option 2, there is no loss in reliability vs. ModSAF baseline reliability. It is unknown how to compare reliability between ModSAF baseline and CCTT SAF. STOW ACTD had average 10 hours uptime for Joint SAF during the 48-hour exercise, and Army SAF was approximately 24 hours. The only difference in reliability might be that new features have been added to the architecture. However these features are considered low-risk, and should not impact reliability once fully tested.

Scalability

As with all approaches outlined, there are echelon scalability limitations to using an FSM approach for behaviors. This approach does not make the problem worse.

Existing Program Impact

Behavior V&V

As in option 2, all re-implemented CISs (and their building blocks) would have to be V&Ved. We would recommend doing the V&V as was done originally for CCTT. Regression testing would have limited utility because the tests would be against a different baseline. There would need to be some interpretation of the results, and you could not take advantage of previous testing results. However, the behavior design will make use of the original CCTT SAF behavior states wherever possible in order to maximize ease of V&V.

There is very little in these architecture enhancements that would help with V&V. We will implement an FSM exerciser that will help verify control flow. Note that this will not significantly help the V&V process, but will be a useful tool for developmental testing.

Extensibility

No ModSAF extensibility lost. CCTT behaviors will now extensible in the same style that ModSAF is. The architectural enhancements in this approach support better extensibility for behaviors. Nothing done here is in support or in detriment to of higher level behaviors (battalions). This is true in both CCTT and ModSAF.

Baseline Maintenance

As in option 2, there will be low maintenance because there will generally be only one implementation of each behavior. In some hopefully rare cases, there may be a CCTT version and a ModSAF version of the same behavior. Hopefully that is near zero.

The enhancements are dominated by extensions to the FSM language. Once fully tested, there will be no new maintenance issues associated with this approach. So, the maintenance issues are the same as option 1.2 (which is low).

Adaptability to new Technology and Architectures

This approach has the same adaptability to new technology as approach 2. New technology implemented in ModSAF derivatives such as Joint SAF will have a straightforward migration path to this unified architecture. New technology implemented in CCTT SAF (such as FBCB2) will require re-implementation in OneSAF.

Development	Adaptability Assessment
Ongoing ModSAF development	Good
Ongoing CCTT SAF development	Poor (re-implementation)
Existing and ongoing STOW enhancements	Good
Joint SAF	Good
CFOR	Good
CCSIL	Good
Command Talk	Good
OneSAF R&D	Unknown

Ease of coordination and integration of multiple developers

This approach has the same ease of coordination as approach 2. The OneSAF testbed can leverage off of the existing ModSAF and Joint SAF approaches for parallel distributed development. The ModSAF architecture has already proven itself to be extensible within the CGF community, and the enhancements described here only improve that extensibility.

Cost

Enumeration of capabilities to migrate

There are three efforts for this approach.

(1) The upgrade of the building blocks
(2) Implementation of architecture extensions
(3) Implementation of the CISs not already implemented in the new architecture

There are several building blocks that need to be upgraded to support CCTT requirements. This includes movement issues (utraveling, vmove, movemap), and shooting issues (vassess, vtargeter), etc. These upgrades are also required for option 1.2. These enhancements are estimated in a table, below.

(1) There are architectural pieces that need to be added to the FSM language for improving it. These estimates are included in a table below, and include the following items:

- Abstract the subordinate management, which includes attrition, change of leader, and changes to task organization.
- Abstract a better timing mechanism (potentially based off of HHours) into the FSM language or task manager. This would allow an event to be fired when a period of time has elapsed.
- Support better defaulting of params events. There should be a default params events for each task.
- Have the subordinate fire an event when its behavior completes so that we don't need to poll for it.
- Abstract into the behavior architecture the storage management for creation of PO objects. This includes the storage as well as garbage collection. There is duplicated code in several places doing this right now.
- Implement an FSM exerciser that would allow you to test the FSM without forcing the author to run the entire system. This may have a key role in helping the VV&A process.

(1) Implement all of the CCTT CISs not already implemented in this new architecture.

Effort to migrate related to capabilities

The time estimates are 379 staff months without including V&V. This is approximately 46 staff months less than option 2. This is a development activity.

Building Block Enhancements	Estimate (Man-Months)
Misc.	12
Movement	18
Shooting	6
Total	**36**

Architecture Enhancements	Estimate (Man-Months)
FSM Architecture	6
Checkpointing	1
Order Decomposition	12
Total	**19**

CISs	CIS-CIS Overlap	Per-CIS Effort Weeks	CIS Man-Weeks	CIS Man-Months
346	75%	5.0	1297.5	**324.38**

CIS Development Man-Months	Architecture Enhancements Man-Months	Bulding Block Additions Man-Months	Total Man-Months
324.38	19	36	**379.375**

Assumptions on migration environment

- The assumptions are the same as option 2, namely:
  - We expect to be able to reuse some existing ModSAF behaviors with CCTT SAF behavior re-implementation.
  - We expect to reuse as many physical models as possible.
- We assume that the architecture changes will gain us approximately 1 week per task therefore justifying the architecture changes.

Assessment

For each assessment criteria (each row), fill in the score columns and make an initial assessment of the weight for each category. Describe your rationale for each weight value. The Low/Medium/High columns qualitatively describe the meaning of each value for a particular criterion.

Criteria	Low (1)	Medium (5)	High (10)	Score
Technical Feasibility Risk	The technical feasibility is rated very good.	The technical feasibility is good but the migration involves some very difficult technical issues.	The technical feasibility is poor. The migration technical is very difficult or the approach has never been tried before.	2
Program Risk	There is little risk of the migration approach	There is a moderate risk of the migration approach	The migration approach is very risky and has never been attempted before	4
Impact to existing ModSAF user community	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	2
Impact to existing	The approach has virtually no	The approach has some	The approach has many impacts, the	2

ModSAF development community (e.g., STOW)	impacts to the existing programs or community	impacts to the existing programs that could be over come with moderate effort	effort involved in overcoming the impacts are very large, or technically challenging	
Impact to existing CCTT program	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	4
Impact to current CCTT development (e.g., FBCB2)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	4
Adaptability to new technology risk	The resulting infrastructure or environment can be easily extended	The resulting infrastructure does not support extension and will require additional enhancements to support future modification	The resulting infrastructure is not unified and should be completely redesigned to support extensibility	3
Cost				379mm

Criteria Assessment Justification

In this section, describe your justification/rationale for the scores provided. Also describe your rationale for the initial weights assigned to each criteria.

Technical Feasibility Risk

All items to be modified are designed and understood, except for Order Decomposition. The system changes resulting from the designs are anticipated to be minor. If it were not for the

current lack of a design for Order Decomposition, the total feasibility would be assessed as a "1".

Program Risk

Program risk was assessed as a "4" because although all program requirements are met, there will be great effort involved, and there is risk to completing the task within schedule.

Existing Program Impact

Existing program impact for ModSAF was assessed as a '2' due to the changes being applied to ModSAF and the potential rewrite of some behavioral support and models. Existing program impact for CCTT was assessed as a "4" because of the large V&V effort required to re-validate all CCTT-specific behaviors. This design will make use of the original CCTT SAF behavior states wherever possible in order to maximize ease of V&V.

Adaptability to new technology risk

This approach preserves the ModSAF architecture, and will not preclude the incorporate of future ModSAF-derived enhancements, such as developed under the STOW program. CCTT SAF enhancements will be more difficult to incorporate since this approach is disjoint from the CCTT SAF architecture. It is assumed that there will be more ModSAF-derived technology development for SAF than CCTT SAF, based on a distributed funding base that includes resources outside the Army.

Cost

Refer to the effort to migrate section for a discussion on cost.

Behavior Assessment: Option 5 (Extend CCTT SAF)

Technical Description

Start with CCTT SAF as the initial baseline and incorporate architectural changes to satisfy extensibility. The CCTT SAF behaviors will remain largely unchanged in the sequencing of actions but the supporting software and control structure will be re-designed. This re-design includes a inversion of the control strategy where units and order handlers (FSM's or behaviors) are registered rather than hard-coded in the structure of the code. In addition, the concept of common behaviors will be unified between the BLUFOR and OPFOR systems to promote describable layering of the software. The crew system will remain largely unchanged except that extensions required for new ModSAF vehicle characteristics will be incorporated. In addition, the Weapon Crew model will be reworked into an FSM approach in a manner similar to the driver re-implementation during CCTT development (driver provides an existing model for the definition of a cooperative collection of FSM's). The UK CATT development effort supports this re-architecture in that they are performing some level of behavioral architecture re-design with similar approaches and objectives. The following list provides a more detailed look at the approach for this option.

- Maintain the overall CCTT SAF Behavioral Architecture
  - Concept of units that receive control individually
  - Communication by events
  - Keep organizational structure and capability for re-organization
  - Keep and modify the CCTT SAF crew design
    - Keep the driver and low level path planning (move map) approach
    - Modify the weapon crew to incorporate FSM's as re-engineered from both ModSAF and CCTT SAF
    - Keep the other crew approaches since they are designed to satisfy specific CCTT requirements
    - Keep the concept of a decoupled crew behavioral model which receives inputs only through messages and events
- Alter the architecture to reuse good solutions of ModSAF
  - Unify orders from specific unit based data models to more generic orders
  - Invert the architecture to remove the explicit structure in the code to support a registration service for order handling and unit simulation
    - Units are described by data files
    - Orders handled by a unit are described by data files
    - Handlers for an order are described by data files
    - Central default situational context handlers with standard information which can be replaced (battle loss, obstacle, spot lists, organization)
  - Partitioned data model
  - Multiple resumes from overrides implemented as stackable slots

    - External interface control accomplished by using the existing object interfaces in CCTT combined with the slot data model for symbolic lookup and introspection
    - Configurable reactions
        - Background situational assessment tasks identified by the order
        - Default set of tasks supplied with the system which perform the current CCTT situational assessments (ground enemy, obstacle, air threat, indirect fire)
        - The order supplies the behavioral responses to situational assessment task messages (it provides the reactionary behavior to execute if the situational assessment task triggers).
        - Create a message type of an assessment task trigger with appropriate reasons and data.
    - Scenario save which supports pausing a scenario and restarting the scenario at a later time (while not relying on the previous distribution topology).
- Change the data model structure to a slot based approach (similar to an HLA data model)
- Alter the FSM paradigm
    - Formalize the FSM representation
        - Utility services and encapsulated representations of the state machine structure
        - Promote the FSM to an architectural type with a method of separation from the supporting code and structures
    - Standard entry points to mimic standard functions for an FSM (prepare, execute, pause, resume, interrupt, stop)
    - Formalize specific events for state transitions
    - Support for fully event driven FSM's rather than just periodic approaches (mission space events and next tick events)
    - Tools for the generation of new FSM's (structure, files, data models, incorporation into the system)
- Expand the supporting architecture
    - Domain analysis of CCTT and ModSAF code to identify reuse
        - Stress utility collections that support primitive services for behaviors (lead identification, route manipulation, location manipulation, location, organization, C&C, location calculations, enemy context, control measure proximity, control measure construction)
        - Identify events which cause transitions and provide primitive support for identification and notification (enemy sited
        - Encapsulate all data models for events into service functions (crew orders, reports, all orders)
    - Common FSM's formalized, standard interfaces/control, and documented for developer query and use
    - Re-factorization of Behaviors
        - CCTT behaviors reengineering into C and supporting architecture
        - ModSAF behaviors redesigned into support architecture
    - Support for parallel, cooperative FSM's at the crew level incorporates ModSAF subsumptive approaches

Advantages of using this Approach

The advantage of this option is that it retains the CCTT SAF behavioral approach thus minimizing any re-design or re-implementation of behaviors in a different representation. This minimizes risk by reducing the rework of the suite of CCTT complex behaviors to a re-factorization effort. In addition, the re-architecture embraces the major requirements of the unification. The ModSAF behaviors will be reverse-engineered and incorporated into this baseline.

- Encapsulation
  - Minimizing the ability to set data from other object promotes extensibility
  - Aligned to the real world model of information transmission
  - Decouples information management
  - Minimizes interactions between distributions
  - Each unit receives control individually
  - Promotes load shedding (degradation remains proportional)
  - Provides support for multi-threading at the scheduler
- Extensibility is improved
  - Inverting the architecture allows for minimal software design overhead when adding new behaviors or units (registration for both units and orders that units handle)
  - Less concentration on unit formalism minimizes software design overhead for adding new units
  - Generic order structure minimizes maintenance by increasing reuse of data model elements
  - Lessons learned from ModSAF inversion of the architecture minimizes risk
  - Systemic approach to layering and primitive utilities centralizes capabilities
    - Results in clear API's by formalizing interfaces
    - Clear selections for event generation
    - Clear use of contextual information
    - Minimizes maintenance by isolation of capability implementations in one place
- FSM formalization promotes standardization
  - Active FSM structure definitions promote addition of FSM capabilities system-wide with minimal impact
  - Reduces process overhead for development of new behaviors (tool sets)
  - Data model standards promote multi-threading
  - FSM's trigger on events rather than periodic processing reduces behavioral processing requirements
  - Standard entry points support system-wide standard control of FSM resources and execution
- Data model re-representation
  - Supports extensibility by removing meta-data from the code structure and allowing for re-definition without code changes
  - Distribution subsystem complexity is greatly reduced
  - Scenario file generation is simplified by allowing for ASCII representation
    - Promotes the reuse of exercises between revisions of the data model

    - Promotes reuse and distribution of the exercise parameters to other technologies and representations (i.e. charts generated by a drawing package, distributed scenario generation, web-based browsing and examination of the scenario by other technologies/languages (Java)).
  - GUI support reduced to simple engine without an understanding of the meta-data (GUI screens for a behavior are data driven and can be specified by the behavioral engineer)
  - Close alignment to an HLA style of data representation allows for easy incorporation of HLA with the object model expressed in the models (rather than a translator at the bottom of the layers).
  - Data model can be defined by the model directly with support from tool sets
- CCTT SAF behavioral structure is not altered minimizing errors in re-representation of behavioral models affecting verification and validation of doctrinally correct behaviors.
  - Concept of a cooperative behavioral model for the unit is preserved allowing for language translation rather than representation translation
  - Primitive utilities and layering of the behaviors architecture represent an improvement on an existing behavior rather than a re-design
  - The CCTT SAF behavioral model represents a more encapsulated and decoupled approach to behavioral modeling compared to explicit manipulation of subordinate data models (unit CPU usage on CCTT represented 3% to 8% of the processor utilization)
  - The Command From Simulator (CFS) model is preserved as a proven approach
  - Changes from PDSS on completion of requirements satisfaction would be easier to incorporate
- ModSAF generic unit behaviors can be represented using this extended CCTT SAF paradigm
- Extensibility would be addressed by incorporating lessons learned from internal architectural re-design analyses, UK CATT re-archirecture and primitive re-structuring, and ModSAF modular, distributed support for behaviors.
- Legacy systems already exhibit portions of the requirements
  - CCTT SAF brings a set of doctrinal behaviors which satisfy CATT training requirements
  - ModSAF brings concepts for modular behavioral infrastructure to support extensibility
  - UK CATT is already in the design phase of a behavioral infrastructure re-architecture to promote the easy addition of units and behaviors
- Adaptability to new technologies and architectures is supported
  - Layering promotes the addition of new concepts and technologies as long as the information flow is maintained at the interfacing level
    - A new technology or approach needs to use the underlying information flows correctly
    - The depth of the perturbation to the information flow determines how deep into a layer the new approach reaches

    - Standard layering approaches and the encapsulated nature of the CCTT SAF information flows promotes the incorporation by well defined flow of control and authority to alter information
  - Inversion of the architecture allows for models to replace capability implementations at various levels as has been demonstrated on ModSAF
- Re-factorization of CCTT SAF behaviors minimizes the cost of the translation from Ada to C
  - All behaviors must be examined and converted to C no matter what the approach
  - Pre-defined methods of handling information flows through the use of layering and architectural inversion are incorporated during the translation to C
  - The structure of the behaviors, decisions in the behaviors, and information flows are maintained from the original design maximizing the reuse and minimizing the cost associated with the original validated behavior (i.e. it is not being re-represented in a different paradigm).
  - All the CIS implementations must be visited for both the re-factorization and the translation to C so both could be done in parallel.
- V&V would take less effort since the underlying models have not changed

Disadvantages of using this Approach

The main disadvantage of this approach is that some development must be accomplished up-front which is not required for the options that use the existing baselines. In addition, this involves a translation of CCTT SAF Ada source code to C. However, the behaviors layer in CCTT does not use any Ada language specific constructs, which would reduce risk.

- Schedule/Cost risk
  - The re-design involves some effort that would not be required in a straight CCTT SAF or ModSAF solution
    - There would need to be a domain analysis of the information flows and supporting capability implementations for both CCTT SAF and ModSAF to determine the list of capabilities, the partitioning of these capabilities into layers, and re-factorization of the information flows
    - This would introduce additional code for testing from an un-modified CCTT SAF or ModSAF baseline
  - The CCTT SAF behaviors would need to be re-represented in C (this is actually a disadvantage when compared to an unmodified CCTT SAF solution only, the rendering of CCTT SAF behaviors in C is a given for most other choices)
- The re-design of the primitives and behavioral layers would delay the immediate incorporation or re-factorization of behaviors which, while a technically correct approach, might be counter to industry/user expectations
- The representation of behavior models in line with the current CCTT SAF approach changes what the ModSAF development community is currently using. This will result in changes to existing non-baseline ModSAF systems and some level of impacts to the community.

Technical Feasibility

The CCTT SAF behaviors are proven to satisfy the CCTT SAF requirements. The only questions associated with a straight CCTT approach are the Ada language and the extensibility of the behavior infrastructure. On the language issue, this option assumes a level of effort in the translation of the CCTT behaviors and infrastructure to C. This is technically feasible since the behaviors and the supporting infrastructure do not use the Ada language in any unique ways, and all language constructs used can be directly translated to C. In fact, the representation of the infrastructure in C facilitates the re-factorization issue of an inverted architecture (use of function pointers). The issue of extensibility of the CCTT SAF behaviors is rooted in the wish to provide extensibility for a user group that is new to CCTT. The re-factorization of CCTT SAF behavioral infrastructure does not represent changes in information flows or data models. It does represent how they are managed and flow through the system. In this manner, the re-factorization is applied to the simulation space only and not the mission space models. This is based on lessons learned from ModSAF infrastructural support for distributed development and extensibility.

Program Risk

Requirements Satisfaction

This fully supports the CCTT requirements since the underlying baseline is a CCTT baseline and the proposed approach does not alter the CCTT behavioral models or information flows. The ModSAF requirements will be mostly met in that all the ModSAF behaviors can be rendered in this architecture and extensibility is supported. What will not be met is any derived requirement to support the existing AAFSM language for behavioral definition.

There is some requirements satisfaction risk for the ModSAF development community. If the behavioral representation, control strategy, and implementation is changed, they will incur some impact in their existing efforts. However, the resultant changes should be a unifying change

Performance

This change does not adversely affect the performance characteristics.

Schedule

There is some level of schedule risk (development and testing) since the infrastructure is being re-designed, the behaviors are being re-factorized, and the data model representation is being re-worked (slot based). However, the addition of new units and behaviors should be easier.

Reliability

This represents a unifying modification and should retain or enhance reliability.

Scalability

This does not affect the scalability limitations inherent in either of the systems.

Existing Program Impact

Behavior Requirements Satisfaction

Since the CCTT SAF behaviors are retained, this fully satisfies the specific CCTT requirements. There would be some issue of CCTT SAF's order creation time decomposition when compared to ModSAF's superior order decomposition to subordinates.

Behavior V&V

This approach should result in low PTR counts for the V&V effort since the structure of the behavior is determined from existing CCTT SAF behavior FSM's which have already been V&Ved.

Extensibility

Extensibility is the key component of this effort and therefore is the basis of this extention. This approach would benefit from lessons learned in designing ModSAF for extensibility.

Baseline Maintenance

This does not adversely impact baseline maintenance beyond the existing problems associated with either ModSAF or CCTT SAF.

Adaptability to new Technology and Architectures

There is nothing inherent in this approach that prevents adaptability to new technologies or architectures beyond the existing systems. The only affect is that current ModSAF developers will be exposed to a different method of behavioral design, unit and crew control, and primitive sets. This has impact to existing projects that rely on the AAFSM language or the ModSAF behavioral implementation. However, the decoupling of the crew and unit behaviors from the simulation space and a closer alignment with the mission space should provide more modular connections to the resulting system.

Ease of coordination and integration of multiple developers

This approach should facilitate coordination and integration of multiple developers in a manner similar to the current ModSAF baseline. Developers, however, must be able to adapt to a different method of rendering behaviors and controlling subordinate units.

Cost

Effort to migrate related to capabilities

New Functionality (LSLOC)	
Multiple Resumes	500
FSM Abstraction	

	3,000
External IF	500
Configurable Reactions	600
Total LSLOC	4,600
SLOC/Hour effort	3
Hours effort	1,533
Months effort	10

Infrastructure Rearchitecture	
CCTT Infrastructure PSLOC	250,000
SLOC/Hour effort	15
Hours Effort	16,667
Months Effort	104

ModSAF Behaviors Incorporation	
New Behaviors	106
Unique Behaviors	75%
Per CIS effort (weeks)	7
Weeks effort	557
Months effort	139

Existing Behaviors Rework	
CCTT CIS's	346
Per CIS effort (weeks)	3
Weeks Effort	865
Months Effort	216

Total Months Effort	469

ModSAF Behaviors Incorporation

These numbers are based on the number of generic behaviors currently in ModSAF (106). The ModSAF analysis team determined that there was a 25% overlap between the behaviors for CCTT and ModSAF. It is estimated that 7 weeks per CIS will be expended to re-represent the ModSAF behaviors.

Infrastructure Re-architecture

The infrastructure re-architecture is based on the amount of CCTT SLOC that needs to be revisited. The number in the table represents physical SLOC for the behavior common subsystem, the SEOD, and common utility packages in each of the behavior specific subsystems (such as BLUFOR, OPFOR small unit tactics). The SLOC/hour effort represents the lines of code that can be converted to a layered C implementation per hour.

Existing Behaviors Rework

These numbers are based on the number of selectable CIS's implemented in CCTT. This number does not include the reactive behaviors or the non-selectable common behaviors since the original estimate of 291 hours subsumed those developments. The estimate is that a CIS can be converted to C and re-factored in 4 weeks.

Assumptions on migration environment

The following list outlines the assumptions for this approach:

- This approach expects to reuse design approaches from the current UK CATT behavioral re-architecture.
- This approach expects to result in an architecture that simplifies the incorporation of new behaviors and units.
- This approach assumes that the GMI interface approach to physical model control remains largely unchanged (for effort estimation purposes only)
- This approach assumes that the existing crew control model can be extended to support any unique vehicle capabilities in the ModSAF collection that is not in the CCTT system.
- This approach assumes that the AAFSM language and structure does not have to be retained to be compliant with the ModSAF replacement requirement.
- This approach assumes that the current model for the distribution of objects and events is not counter to the behavioral model approaches and requirements in ModSAF.
- This approach assumes that the conversion to C for behaviors occurs in parallel with the re-factorization of the behaviors and that there will not be a artificial programmatic requirement to fully convert CCTT SAF to a workable C version without alterations first.

Assessment

The following table provides quantification of the criteria (note the scale in the title bars, the numeric scale was never defined).

Criteria	Low (1)	Medium (5)	High (10)	Score
Technical Feasibility risk	The technical feasibility is rated very good.	The technical feasibility is good but the migration involves some very difficult technical issues.	The technical feasibility is poor. The migration technical is very difficult or the approach has never been tried before.	4

Program Risk	There is little risk of the migration approach	There is a moderate risk of the migration approach	The migration approach is very risky and has never been attempted before	4
Impact to existing ModSAF user community	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	3
Impact to existing ModSAF development community (e.g., STOW)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	3
Impact to existing CCTT program	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Impact to current CCTT development (e.g., FBCB2)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Adaptability to new technology risk	The resulting infrastructure or environment can be easily extended	The resulting infrastructure does not support extension and will require additional	The resulting infrastructure is not unified and should be completely redesigned to support extensibility	4

		enhancements to support future modification		
Cost				469mm

Criteria Assessment Justification

In this section, describe your justification/rationale for the scores provided. Also describe your rationale for the initial weights assigned to each criteria.

Technical Feasibility

The score of 4 assigned to this criterion is based on the fact that the approach is tailored towards extensibility and feasibility. However, there is some rework of the representation of the data model that has not been implemented in any system to date. The starting architecture is currently in use and the prototypical re-architecture is being implemented on the Firm Fixed Price UK CATT program (which cannot suffer a technical infeasible approach).

Program Risk

Program risk is rated as a 4 because of the additional effort in the domain analysis and re-design of the behavioral infrastructure. This should result in a unified approach promoting easier extensibility allowing the ModSAF unique capabilities to be incorporated.

Existing Program Impact

Existing program impact for ModSAF is rated as a 3 because of impact to current ModSAF users. This represents a change in the behavioral representation that these consumers are accustomed to. There would be some learning involved and potential re-tooling for these customers. CCTT impact is assessed as a '1' since its impact is minimized by this approach by retaining much of the behavioral representation and control strategy.

Adaptability to new technology

Adaptability to new technology is rated as a 4 due to the impact to new development currently being performed on STOW and other ModSAF derivatives as well as the lack of support for truly subsumptive behavioral representation currently on ModSAF. CCTT SAF work for PDSS, P3I, and UK CATT enhancements are supported.

Cost

Refer to the effort to migrate section for a discussion on cost.

Behavior Assessment: Option 6 (Develop capability to run both ModSAF and CCTT SAF behaviors within the OneSAF testbed)

Technical Description

Start with ModSAF as the initial baseline. Modify the ModSAF infrastructure to support both ModSAF and CCTT SAF behaviors as is (with little or no changes to the two sets of behaviors). This can be done in one of two ways:

1. Link the Ada FSMs into ModSAF using a tool such as GNAT, such that wrapper functions are used and provide additional architectural support such that the Ada FSMs can be "run" from within ModSAF. There would need to be a SEOD to PO translation as well.
2. Use an Ada to C translator to convert the Ada FSMs to C. Either maintain the Ada or C version in the repository as the "master" version.

Advantages of using the approaches

- no reimplementation of existing ModSAF or CCTT SAF behaviors
- VV&A effort is minimal
- perhaps the cheapest short term solution in terms of cost and time
- very high level of code reuse

Disadvantages of using the approaches

- perhaps the most expensive long term solution in terms of cost and time due to maintenance of the "two headed system"
- duplication or overlap of the behavioral capabilities is not reduced
- overall code size increases drastically
- Ada code or "automatically translated" Ada code is undesirable
- expected CCTT execution will be less efficient due to additional layers of function invocations or poorly "translated" Ada to C code.

Technical Feasibility

We have no idea what problems we might run into. We have not extensively used an Ada to C translator or GNAT and therefore are unaware of potential problems. We are trying to merge very different architectures and we will redefine how much we are going to have to pull over from CCTT. There are huge issues of whether the machine/compiler/linker will be able to handle to large size of this effort. Heavy risk associated with many unknowns.

Program Risk

Requirements satisfaction

Both systems' behavior requirements should be satisfied.

Performance

There are likely to be fairly significant performance issues due to the size and the wrapper approach. This approach will likely not satisfy the performance requirements of either system.

Schedule

Unknown at this time and therefore high schedule risk.

Reliability

Depends on the reliability of the tools and the translation process that is compounded with the reliability of both systems.

Scalability

This not a scalable solution is any way shape or form. When we are talking about behavior echelon scalability there should be only minor differences from the other approaches. However, if we are going to scale up, we will now have to effectively implement behaviors in 2 different trees.

Existing Program Impact

Behavior V&V

Since only considering the Behavior V&V of CCTT, this should be a fairly low risk item. There should be less V&V effort since the behaviors remain the same.

Extensibility

This approach does not satisfy ModSAF's extensibility requirement.

Baseline Maintenance

By far this is the worst maintenance nightmare possible. In fact, it is worse than having 2 independent systems.

Adaptability to new Technology and Architectures

Half of the behavior architecture would be extensible to new technology and architectures. Which half would depend on where the new technology was developed. For example, bringing in STOW software would only work with the ModSAF behaviors (it would need to be implemented completely differently in the other). An advantage is that this would potentially be able to share common with the rest of the CCTT system.

Ease of coordination and integration of multiple developers

If a behavior was to be added to both the ModSAF side and CCTT side, the developers will have to know both systems.

Cost

Enumeration of capabilities to migrate

Add CCTT SAF architecture and behaviors (including SEOD) on top of the low level ModSAF architecture.

Effort to migrate related to capabilities

This is an unknown. This has the potential of being the lowest cost alternative, but it is also the largest cost risk.

Assumptions on migration environment

A huge assumption is that the CCTT behaviors will work with the physical models in ModSAF. If not, then other subsystems will need to be brought over. This may lead to more subsystems being brought over, which leads to the domino effect.

Assessment

For each assessment criteria (each row), fill in the score columns and make an initial assessment of the weight for each category. Describe you rationale for each weight value. The Low/Medium/High columns qualitatively describe the meaning of each value for a particular criterion.

Criteria	Low (1)	Medium (5)	High (10)	Score
Technical Difficulty	The technical feasibility is rated very good.	The technical feasibility is good but the migration involves some very difficult technical issues.	The technical feasibility is poor. The migration technical is very difficult or the approach has never been tried before.	9
Program Risk	There is little risk of the migration approach	There is a moderate risk of the migration approach	The migration approach is very risky and has never been attempted before	8
Impact to existing ModSAF user community	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically	1

		come with moderate effort	challenging	
Impact to existing ModSAF development community (e.g., STOW)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	8
Impact to existing CCTT program	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Impact to current CCTT development (e.g., FBCB2)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	8
Adaptability to new technology	The resulting infrastructure or environment can be easily extended	The resulting infrastructure does not support extension and will require additional enhancements to support future modification	The resulting infrastructure is not unified and should be completely redesigned to support extensibility	7
Cost				

Criteria Assessment Justification

The assessment teams determined that the baseline impacts of dual competing architectures was not feasible and did not perform any costing assessment.

Behavior Assessment Summary

This is the summary document we plan to fill out at the end of the assessments.

The migration approaches are as follows.

1. Re-implement missing behaviors from ModSAF into existing CCTT infrastructure.
2. Re-implement missing behaviors from CCTT SAF into existing ModSAF infrastructure.
3. Develop new unified behavior representation and implement all behaviors in it.
4. Extend ModSAF with additional infrastructure to ease re-implementation of CCTT behaviors within ModSAF.
5. Extend CCTT SAF with additional infrastructure to ease re-implementation of ModSAF behaviors within CCTT SAF.
6. Develop capability to run both ModSAF and CCTT SAF behaviors within the OneSAF testbed.

Criteria	Re-implement ModSAF Behaviors in CCTT SAF	Re-implement CCTT SAF Behaviors in ModSAF	Develop New Behavior Architecture	Extend ModSAF and Re-implement CCTT SAF Behaviors	Extend CCTT SAF and Re-implement ModSAF Behaviors	Develop Dual Behavior Architecture Capability
Technical Feasibility Risk	3	1	N/A	2	4	9
Program Risk	2	7	N/A	4	4	8
Existing impact to ModSAF user community	7	1	N/A	2	3	1
Impact to existing ModSAF development community (e.g., STOW)	7	1		2	3	8
Impact to existing CCTT program	1	7		4	1	1
Impact to current CCTT development (e.g., FBCB2)	1	7		4	1	8

Adaptability to new technology	10	3	N/A	3	4	7
Cost	260 mm	425 mm	N/A	379mm	469 mm	N/A

C.2 System Assessment

System capability Assessment: Option 1

Use CCTT System Capabilities as a basis and re-implement the missing system capabilities from ModSAF.

Technical Description

Take CCTT SAF as the baseline.

Add the following ModSAF system capabilities to CCTT SAF:

Develop a Pocket SAF
Develop or Support DT executables
Develop or Support the ModStealth executable
Develop or Support the Logger executable
Develop or Support the Blaster
Develop a strategy for partitioning the CGF Protocol
Develop the approach for supporting the ModSAF protocols and protocol logics for SIMAN, Environment, Minefields, Weather, and Exercise Management
Develop the External APIs to the current users such as CFOR, Command Talk, Soar, BBS, and other PO users.

Pocket SAF Capability
Provide an executable version of the CCTT baseline that combines the capabilities of the SAF WS and the CGF.

DT executables
Provide executables for these as part of the CCTT Baseline. Impact is loss of common code just as we're losing common code within CCTT.

ModStealth executable
Provide executables for these as part of the CCTT Baseline. Impact is loss of common code just as we're losing common code within CCTT.

Logger executable
Provide executables for these as part of the CCTT Baseline. Impact is loss of common code just as we're losing common code within CCTT.

Blaster
Provide executables for these as part of the CCTT Baseline. Impact is loss of common code just as we're losing common code within CCTT.

CGF Protocol Partitioning
Expand the SEOD object attributes to include a logical partition attribute and modify the network controller to filter on this attribute. Each object in a SEOD for a given node will be identified as belonging to a logical partition. Each partition will be a contained SAF WS and

CGF suite and will not interchange information (except for any necessary simulation management objects).

ModSAF protocols
Standardize API around protocol logic execution and provide pluggable modules: One for CCTT Standards and one for ModSAF standards
Alternative: merge capabilities into the same logical stream and provide flags for which standard is to be used.

External APIs
Modularize and standardize external hooks that have been established for ModSAF and provide the like capabilities within the CCTT SAF architecture.

Distribution Mechanism
Merge the data models from both CCTT SAF and ModSAF in support of the unified clients. Incorporate the features from ModSAF that support GCS, endianess support, scenario save, and the ModSAF baseline RTI support.

Framework Rework
Rework the current CCTT framework to develop a C language approach.

Advantages of using this approach

- All CCTT System Capabilities are satisfied with minimal effort.
- ModSAF scalability issues related to the distribution mechanism are mitigated.

Disadvantages of using this approach

- Maintainability will be a lot of work for each duplicate set of functionality (ie. Supporting both minefield protocols, etc.).
- Extensibility will be compromised due to the duplicate sets of functionality in the system.
- Maintainability will be more difficult with translators and interpreters used to interface to existing un-modifiable CCTT components.

Technical Feasibility

Overall, this approach is technically feasible but there are some challenges. These challenges are based on the conflicting desire to maintain backward compatibility with both system interfaces and the desire to merge disparate systems. The use of translators and interpreters are minimized due to their inherent problems with maintenance and the tendency to reinforce deprecated interfaces and data models. In addition, care must be taken to limit the single point translation of information that can promote bottlenecks in a distributed environment. Other system requirements which must be met are feasible considering the fact that the infrastructure (or simulation services) of both systems have similarities. For most of the external executables that support a protocol or service, the existing infrastructure can be reused promoting some level of ease of maintenance.

Program Risk

Requirements satisfaction

All ModSAF requirements can be met. All CCTT SAF requirements will also be met. CCTT specific data requirements for the SEOD file format and the CGF protocol will be met by providing an implementation of the SEOD which translates the data model to the OneSAF testbed data model. In this manner, CCTT will suffer minimal impact caused by an addition to its code baseline that represents a new implementation to the SEOD API.

Performance

It is anticipated that the performance goals required by each capability will be met and not be degraded when compared with present systems.

Schedule

There is minor schedule risk associated with this option based on the additional requirements that are being re-represented in this baseline.

Reliability

There is no anticipated detrimental change to the reliability of each of these systems when compared to the current systems.

Scalability

This does not affect the scalability limitations inherent in either of the systems. There is an anticipated increase in the distribution system's inherent scalability based on the increased scalability of the SEOD when compared with PO scalability.

Existing Program Impact

System capability Requirements Satisfaction

It is anticipated that all system capability requirements will be maintained and met.

System capability V&V

This change has no impact on V&V and provides no adverse impact on regression testing.

Extensibility

This change would incorporate system and architectural lessons learned from the ModSAF development that promoted extensibility. The impact to extensibility is anticipated to be minimal.

Baseline Maintenance

There is some issue of a loss of connection to the shared baseline of both the ModSAF systems and the CCTT system. The ModSAF capabilities which provide the DT executables,

ModStealth executables, Logger executable, and Blaster executable rely on common code. It is anticipated that the resulting infrastructure will be usable to these systems but that their closeness to the resulting system will be less. In addition, the present shared infrastructure throughout the CCTT system will be removed and this baseline will incorporate new infrastructure components that will not be shared with CCTT.

Adaptability to new Technology and Architectures

There is nothing inherent in this approach that prevents adaptability to new technologies or architectures beyond the existing systems. The only affect is that current ModSAF developers will be exposed to architectural approach and system approach. This might provide some impact to developers that rely on specific implementations of system components in ModSAF.

Ease of coordination and integration of multiple developers

This approach should facilitate coordination and integration of multiple developers in a manner similar to the current ModSAF baseline.

Cost

Enumeration of capabilities to migrate

Pocket SAF – The approach for this capability is to combine the capabilities for the workstation and CGF code into one executable. This was the standard configuration for Build 2 for CCTT and is the present configuration for the DIMM.

DT executables, ModStealth executable, Logger executable, Blaster - The approach for these capabilities are to combine the current CCTT infrastructure components with each of the models and protocols that currently exist on ModSAF for these systems. This is the approach used for several services and systems in CCTT and is similar to the approach used on ModSAF.

CGF Protocol partition – The approach for this capability is to provide an attribute for each SEOD object in the current object data model. This attribute would be a byte which represents a logical partition. The SEOD code would then be modified to accept the assignment to a partition and filter incoming information based on this partition.

ModSAF protocols – There are two strategies for this capability that are outlined in earlier. The first approach would be to provide configurable modules that provide each set of protocols and can be replaced when needed. The second approach would be to provide both sets in one model set and parameterize which is used at any given time.

External APIs – The approach would be to identify the external APIs to be developed, identify what information or control is provided to the system capabilities, and incorporate modifications to the system for similar information and control to the CCTT baseline.

Distribution Mechanism – The approach would be to merge the data model requirements from ModSAF with the protocol and distribution mechanisms of CCTT. This approach would preserve the CCTT distribution scalability and capacity.

Framework Rework – The approach would be to convert the framework components to a C language representation. There would be minimal risk for this effort since the framework components are non-complex and their similarity to ModSAF infrastructure components facilitate selective reuse.

Effort to migrate related capabilities

Capability	Months
Pocket SAF	2.5
Dynamic Terrain	7.0
ModStealth	3.0
Logger	2.0
Blaster	0.5
Protocol partitioning	2.0
ModSAF Protocols	4.0
External API's	6.5
Distribution Mechanisms	10.0
Framework Rework	5.0
Total Months	42.5

Assumptions on migration environment

- The OC and MCC components of CCTT SAF cannot be modified unless it is funded by this OneSAF project.
- The infrastructure components of the modified CCTT SAF is similar to the existing services and components in ModSAF
- The major reuse of the disparate executables is preserved in the support for the protocols and services in the client models and the infrastructure. This allows existing ModSAF models to be incorporated into the new infrastructure with the new information flows (i.e. the major models of the DTS-Sim can be re-hosted in this new infrastructure).

Assessment

For each assessment criteria (each row), fill in the score columns and make an initial assessment of the weight for each category. Describe you rationale for each weight value. The Low/Medium/High columns qualitatively describe the meaning of each value for a particular criterion.

Criteria	Low (1)	Medium (5)	High (10)	Score
Technical Feasibility risk	The technical feasibility is rated very good.	The technical feasibility is good but the migration involves some very difficult technical issues.	The technical feasibility is poor. The migration technical is very difficult or the approach has never been tried before.	3
Program Risk	There is little risk of the migration approach	There is a moderate risk of the migration approach	The migration approach is very risky and has never been attempted before	2
Impact to existing ModSAF user community	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	2
Impact to existing ModSAF development community (e.g., STOW)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	2
Impact to existing CCTT program	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	5
Impact to current CCTT development (e.g., FBCB2)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over	The approach has many impacts, the effort involved in overcoming the impacts are very	2

		come with moderate effort	large, or technically challenging	
Adaptability to new technology risk	The resulting infrastructure or environment can be easily extended	The resulting infrastructure does not support extension and will require additional enhancements to support future modification	The resulting infrastructure is not unified and should be completely redesigned to support extensibility	5
Cost				42.5mm

Criteria Assessment Justification

This section, describes the justification/rationale for the scores provided

Technical Feasibility Risk

Technical feasibility risk is a '3' due to the required re-implementation of the different ModSAF executables and the requirement to translate the SEOD data model (including support for GCS and endianness).

Program Risk

This criterion is rated as a '2' based on the re-implementation of the ModSAF executables and the re-forging of the data model.

Existing Program Impact

This criterion is rated as a '5' for the current CCTT program based on the changes required to provide a SEOD translation layering to support the CCTT requirement for the SEOD file format and the CGF protocol (OC workstations). This would tend to invalidate the current exercise files and require translation to the new format. The other impacts are assessed as a '2'. Future CCTT development will use the new baseline and not be impacted by a change. ModSAF development will be minimally impacted except for exercise invalidation.

Adaptability to new technology

This criterion is rated as a '5' based on the need for an improved system capability to support future OneSAF architectural composition requirements. OneSAF will require additional enhancements to this underlying architecture to support OneSAF requirements.

System capability Assessment: Option 2

Use ModSAF System Capabilities as a basis and add the missing system capabilities from CCTT SAF.

Technical Description

Take ModSAF as the baseline. Add the following CCTT SAF specific system capabilities into ModSAF: Comm, CCTT Protocol Standards, EM Protocol, DAR Messages, BIT (Built In Test), Repair/Resupply Protocol, DI Mount/Dismount Protocol, MCC (exercise initialization setup), OC, and DIWS.

Comm (Real Radio Communications)
The Comm software is almost all in C, with some Ada wrappers. Comm cannot be modified. Use the existing Comm software from CCTT SAF and add a shared memory interface and manager into the ModSAF architecture that is equivalent to Comm's shared memory interface. Comm is kept as a separate process since CCTT SAF systems are designed for use on existing dual processor machines. This will help meet the performance requirements for CCTT SAF. The shared memory is allocated in big chunks and is managed by a shared memory manager. The shared memory will contain information about a unit's location and Signal PDUs. Comm will have the ability to be run from any ModSAF station, with the understanding that Comm depends upon other hardware that is not a part of OneSAF testbed software to run. ModSAF will need an editor for the user to specify the speaking unit or intercom to MCC or AAR channels. This editor will be available from ModSAF only when the Comm is also running.

Estimate: 4MM
Work: Comm Editor, Shared Memory interface on ModSAF and Comm.
Assumptions: Assumes ModSAF GUI choice for OneSAF testbed.

CCTT Protocol Standards
1. Add an entry in ModSAF's disconst.rdr file for CCTT SAF's DIS 2.0.4 enumerations that will be used when running in CCTT SAF mode. Use CCTT SAF's appearance bit methodology, enumeration standards, and articulation modeling when in CCTT SAF mode.

2. CCTT SAF conflicting requirement is to have in flight weapons cease to exist when pausing and resuming the exercise. When in CCTT SAF mode, follow CCTT's requirement, and when in ModSAF's mode, follow ModSAF's requirement to keep in flight weapons when resuming.

Estimate: 1MM for part 1, 1MM for part 2
Work: disconst additions, minor coding
Assumptions: ModSAF GUI chosen

EM Protocol (Environment Manager)
Use ModSAF's DT capabilities and add the missing capabilities from CCTT SAF into ModSAF (abatis and log cribs). Support both the ECN and EM protocols. The EM protocol will be supported by a translator to/from MSO and parameterized ECNs and EM PDUs.

Estimate: 6MM

Work: Adjust sizes of relocatable objects before they are created in order to adjust to CCTT SAF's representation limits, and can breach them properly. Develop a converter to/from ECN and EM.

Assumptions: ModSAF GUI chosen
Risky – Assumes that these can be mapped!

DAR Messages (Data Analysis Reporting)
DAR messages are DIS Event PDUs. Re-implement DAR messages in ModSAF, based on the CCTT SAF code.

Estimate: .5MM
Work: We only need to write DAR PDUs (no receipt), less than 10 types.
Assumptions: Protocol changes only

BIT (Built-In-Test)
CCTT SAF's BIT is implemented with the Action Request PDU. Applications return OK status which is checked about 1 per minute. This is needed in ModSAF if we base our system capabilities on ModSAF because the MCC uses it.

Estimate: .5 MM
Work: Make ModSAF capable of receiving Action Request PDU and sending out appropriate acknowledgement.
Assumptions:

Repair/Resupplies Protocol
Use the OC's Repair/Resupply/Recovery protocols in ModSAF.

Estimate: .25MM
Work: minimal, only PDUAPI changes
Assumptions: Just the protocol setup is trivial, assume the rest is addressed in the behaviors.

DI Mount/Dismount Protocol
Implement CCTT SAF's mount/dismount protocol in ModSAF.

Estimate: .25MM
Work: minimal, only PDUAPI changes
Assumptions: Assume DIS only (no SEOD). Just the protocol setup is trivial, assume the rest is addressed in the behaviors.

MCC (Exercise File Init, Pre-Exercise setup)
The MCC remains unchanged. The MCC depends upon SEOD and writes exercise/overlay files from SEOD. A setup time one-way translation must occur between MCC's SEOD representation and the new C2 representation of scenarios and overlays (not during runtime). This can be implemented in one of three ways, two methods are described here, and the other option overlaps with the OC, so the last option is described in the following OC section:

Create a separate external data model file conversion program that converts between SEOD/new-C2 scenarios and overlays. This is analogous to CTDB's file conversion program.

Modify the MCC to translate SEOD into new C2 format only when writing to a file. The OneSAF testbed's new C2 protocol would read these scenario/overlay files like any other.

These two options only meet the needs of the MCC since they only fix the startup scenario and overlay conversion. Both options are roughly the same cost. The first option is easier to do since it does not require any modifications to the MCC.

Issues are with translation (PO task state data, PDUs represented at different fidelities, etc...) or interpreting SEOD and using something like taskutil/unitutil to assign tasks and units, or only translating a subset of the PDUs.

The MCC also uses SIMAN PDUs for exercise initialization. These PDUs will have to work with ModSAF (start/stop, pause/resume, restart, delete entity).

Estimate:6MM
Work: Develop a translator between C2 and SEOD for exercise init time only, and SIMAN PDU handling.
Assumptions: Assumes both data models are mapable.

OC (Operations Center)
There are two potential options with regard to the OC, only the first option also satisfies the MCC requirement: 1. Modify the OC to use new runtime C2 (data model, events, and protocol) instead of SEOD, if we use ModSAF as a starting point, and 2. Incorporate the OC capabilities into OneSAF testbed. The OC may be replaced in a few years. If the OC is going to be replaced before we really need to support it, then don't bother supporting the interface to the OC.

Estimate: 6MM for option 1, 4MMfor option 2.
Work: Option 2: Translate OC to C. Make more data driven, modular, and extensible. Replace SEOD with new C2 protocol.

Assumptions: Assume another task handles merging of PO and SEOD.

DIWS
The DIWS has the same *initialization* issues as OC.

Using ModSAF's PO as a basis and add CCTT SAF's SEOD Capabilities (Data Distribution)

We have the luxury in this case to have semi-equivalent capabilities since SEOD was based on PO. There are still many differences between the two, but at a higher level, they appear similar. There are three main components to PO and SEOD: the features, the protocols, and the data models. Both models support scenario/overlay saving and loading.

- Features
  - PO features that are different from CGF (SEOD)
  - Partitioning of PO (large scale simulation support)
  - Big/little endian mixing
  - HLA/RTI
  - GCS

CGF (SEOD) features that are different from PO

- Events (reliability of events)
- Strict Object Ownership (others request changes)
- Supports about 30 machines using CGF Protocol together
- Partial object updates (data reduction on network)
- Distribution of units across machines
- Better load balancing (boss concept)
- Segmented routes

Estimate: 6MM
Work: Merge SEOD's features into OneSAF testbed

Protocols

PO Protocol PDUs

- Simulator Present
- Describe Object
- Objects Present
- Object Request
- Delete Objects
- Set World State
- Nomination

CGF Protocol PDUs

- Describe Application
- Describe Object
- Announce Object
- Request Object
- Delete Object
- Describe Event

- Request Event

Estimate: 3MM
Work: Merge SEOD's protocols into OneSAF testbed

Data Models

The data models between the two systems are organized differently.
All of the special features in PO and CGF (SEOD) need to be combined into the new C2 protocol for OneSAF testbed. The protocols need to be merged, and the data models need to be selectively merged. The merging of the data models will be the most difficult part of this task. The merging of SEOD data models should be done on an as needed basis. For example, add all data model components needed to support events, segmented routes, and boss concept, and merge in needed differences between comparable data models.

Estimate: 3MM
Work: Merge SEOD's data models into OneSAF testbed

Advantages of using the approaches

All ModSAF System Capabilities will already be done.

Scalability still exists on the ModSAF side.
Both PO and SEOD have their own special features that will benefit in OneSAF testbed. A merge of the features of PO and SEOD will be support much more than any one system can.

All ModSAF PO capabilities will already be done.
Scalability still exists on the both sides (CCTT SAF and ModSAF).
Extensibility is greater than in any one system. The use of CCTT SAF's Boss CGF and SAF WS concepts in ModSAF should help ModSAF to do better distribution of units (load balancing).

Disadvantages of using the approaches

Maintainability will be a lot of work for each duplicate set of protocols (ie. supporting both minefield protocols, etc.). Extensibility will be compromised due to the duplicate sets of protocols in the system.

Maintainability will be more difficult with supporting duplicate protocols.

The time to merge PO and SEOD will take longer than just settling with the capabilities currently in one system.

Technical Feasibility

These tasks can be accomplished on a technical level. Converters/translators/interpreters are not ideal. They require constant maintenance, limit the scalability of the system as a whole, and make it difficult to extend the functionality.

There are assumptions made about PO and SEOD having corresponding data models, which add to the risk of this work.

On a technical level, this merge of PO and SEOD is very feasible. This is due to the fact that SEOD was based on an older version of PO. Their high-level similarities make this merge very feasible.

There will be some risk with respect to the selective merging of the two data models.

Program Risk

Requirements satisfaction

All ModSAF requirements will be easily met. All CCTT SAF requirements will also be met, but will contain some form of translation/conversion for protocols, which will not meet OneSAF's extensibility requirement on the CCTT SAF side.

All ModSAF PO and CCTT SAF SEOD functionality requirements will be met.

Performance

Run-time translators will lead to decreased performance in the system.
The performance of the resulting OneSAF testbed C2 protocol will be better than what is currently in either system.

Schedule

The mapping of data models in PO and SEOD increases schedule risk since it is not known at this time if the assumptions we made are correct.

There is a slight risk in schedule due to the large difference in data models between PO and SEOD.

Reliability

Because these approaches require significant changes to many parts of the system, especially near the protocol layer, there is some risk in reliability. The system needs to be thoroughly tested after these changes are made.

The reliability of the resulting OneSAF testbed C2 protocol will be better than what is currently in either system.

Scalability

Scalability will be met on both sides.

The scalability of the resulting OneSAF testbed C2 protocol will be better than what is currently in either system.

Existing Program Impact

System capability Requirements Satisfaction

The current system capability requirements for ModSAF and CCTT SAF will be met.

Extensibility

Extensibility with regard to protocols will be compromised because there will be duplicate sets of protocols that must be maintained in order to meet existing requirements from both systems.

Because the best features of both PO and SEOD will be merged, the system will be extensible toward modification of existing and new system capabilities.

Baseline Maintenance

Baseline maintenance will be more complicated because duplicate sets of protocols need to be supported and tested.

Regarding the merging of PO and SEOD, there is still a significant amount of work to be done to merge existing capabilities into the new system, but it will be easiest with this approach since the merge will contain the features of both systems.

Adaptability to new Technology and Architectures

The system capabilities in the OneSAF testbed will be as adaptable to new technologies as the current systems are, but are not as easily adaptable to both modes (CCTT and ModSAF) in OneSAF testbed at the same time since twice the work would need to be done to support duplicate protocols.

Merging the features of PO and SEOD makes the OneSAF testbed very adaptable to new technology and architectures, more so than either one of the SAFs could be alone.

Ease of coordination and integration of multiple developers

The ease of coordination and integration of multiple developers will be equivalent to ModSAF's current capabilities since this system capability work will be based on ModSAF, but will require more work to test and debug the system since it will be larger and support different protocols.

Since this merge will be based on ModSAF's PO capabilities, the coordination and integration of multiple developers will be the same (or a little better) as what ModSAF's ability currently is for this.

Cost

Enumeration of Capabilities to Migrate

- Comm (Real Radio Communications)
- CCTT Protocol Standards
- EM Protocol (Environment Manager)
- DAR Messages (Data Analysis Reporting)
- BIT (Built-In-Test)

- Repair/Resupplies Protocol
- DI Mount/Dismount Protocol
- MCC (Exercise File Init, Pre-Exercise setup)
- OC (Operations Center)
- DIWS
- Using ModSAF's PO as a basis and add CCTT SAF's SEOD Capabilities

Effort to migrate related capabilities

Capability	Months
Comm (Real Radio Communications)	4
CCTT Protocol Standards	2
EM Protocol (Environment Manager)	6
DAR Messages (Data Analysis Reporting)	.5
BIT (Built-In-Test)	.5
Repair/Resupplies Protocol	.25
DI Mount/Dismount Protocol	.25
MCC (Exercise File Init, Pre-Exercise setup)	6
OC (Operations Center)	6 – option 1 4 – option 2
DIWS	3 – option 1 3 – option 2
Using ModSAF's PO as a basis and add CCTT SAF's SEOD Capabilities	12
Total	40.5

Assumptions on migration environment

The OC and MCC components of CCTT SAF cannot be modified unless it is funded by this OneSAF project.

Assessment

For each assessment criteria (each row), fill in the score columns and make an initial assessment of the weight for each category. Describe you rationale for each weight value. The Low/Medium/High columns qualitatively describe the meaning of each value for a particular criterion.

Criteria	Low	Medium	High	Score
Technical Feasibility Risk	The technical feasibility is rated very good.	The technical feasibility is good but the migration involves some very difficult technical issues.	The technical feasibility is poor. The migration technical is very difficult or the approach has never been tried before.	4
Program Risk	There is little risk of the migration approach	There is a moderate risk of the migration approach	The migration approach is very risky and has never been attempted before	4
Impact to existing ModSAF user community	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	2
Impact to existing ModSAF development community (e.g., STOW)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	2
Impact to existing CCTT program	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	5
Impact to current CCTT development (e.g., FBCB2)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	NA
Adaptability to new technology	The resulting infrastructure or	The resulting infrastructure	The resulting infrastructure is not	3

risk	environment can be easily extended	does not support extension and will require additional enhancements to support future modification	unified and should be completely redesigned to support extensibility	
Cost				40.5MM

Criteria Assessment Justification

Technical Feasibility

The assumptions about mapping PO and SEOD data models and the duplicate protocol support leads to a medium level assessment of its technical feasibility.

The technical feasibility of this approach is very good because it combined the best features of both (PO based) systems into OneSAF testbed.

Program Risk

The risk involves managing duplicate protocols and in mapping PO and SEOD data models.

Impact to existing ModSAF user community

There is very little impact to the existing ModSAF user community because this work is based on ModSAF. The score is '2' and not '1' because as with any change, there is the potential to have some kind of impact of the existing ModSAF user community.

There will be a small amount of existing program impact due to feature, protocol and data model changes. The impact will be small because both system capabilities are already similar.

There will be some impact to the existing ModSAF community unless ModSAF remains unchanged. The impact will be much less with this option when compared to starting with CCTT SAF system capabilities since this work is based on ModSAF.

The PO and SEOD merger is going to be very useful to the existing ModSAF development community.

There is a medium level impact to the existing CCTT program because there is a significant amount of work required to meet the requirements of CCTT SAF.

Adaptability to new technology risk

This is not as easily adaptable as it could be because duplicate protocols need to be supported which requires twice the maintenance.

Taking the best features of both systems makes OneSAF testbed very adaptable to new technology.

Cost

The cost for this approach is more than the cost of just settling with one of the existing ModSAF PO or CCTT SAF SEOD protocols. It is probably the least cost when considering merging the two protocols together.

C.3 Synthetic Natural Environment Assessment

Synthetic Natural Environment (SNE) Analysis

Introduction

This analysis covers the OneSAF testbed Synthetic Natural Environment. Two possible approaches for forming a union of CCTT Environment and ModSAF Environment capabilities are described, as well as a number of issues common to both. A quantitative analysis is provided, as well as supporting documentation on functionality differences, assumed SNE scope, and CCTT interoperability.

The analysis conclusion is that ModSAF environment software should be the starting point for OneSAF testbed SNE.

Criteria Assessment

The following table summarizes the criteria and the definition of low, medium, and high rankings. Rankings are provided for each criteria. The number of criteria at each rank is provided in the "Risk Ranking" row.

A quantitative evaluation is provided by assigning a score to each criterion. Higher ranking values are better (less risk)

This analysis shows significantly better score (lower risk) for the ModSAF baseline approach. In fact, with the given rankings, the CCTT Baseline approach would rank higher only if "Existing Program Impact" (namely, CCTT Interoperability) were deemed more important than all other criteria combined. A detailed justification for each ranking follows.

Criteria	Low (1)	Medium (5)	High (10)	Migrate within CCTT Baseline	Migrate within ModSAF Baseline
Technical Feasibility Risk	The technical feasibility is rated very good.	The technical feasibility is good but the migration involves some very difficult technical issues.	The technical feasibility is poor. The migration is technically very difficult or the approach has never been tried before.	8	3
Program Risk	There is little risk of the migration	There is a moderate risk of the migration	The migration approach is very risky and has never been	7	3

	approach	approach	attempted before.		
Impact to existing ModSAF user community	The approach has virtually no impacts to the existing programs	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts and or the effort involve in over coming the impacts are very large or technically challenging.	5	3
Impact to existing ModSAF development community (e.g., STOW)	The approach has virtually no impacts to the existing programs	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts and or the effort involve in over coming the impacts are very large or technically challenging.	5	2
Impact to existing CCTT program	The approach has virtually no impacts to the existing programs	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts and or the effort involve in over coming the impacts are very large or technically challenging.	5	8
Impact to current CCTT development (e.g., FBCB2)	The approach has virtually no impacts to the existing programs	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts and or the effort involve in over coming the impacts are very large or technically challenging.	5	8
Adaptability to new technology Risk	The resulting infrastructure or environment can be	The resulting infrastructure does not support extension and will require	The resulting infrastructure is not unified and should be completely redisigned to	5	3

	easily extended	modifications to support modification.	support extensibility.		
Cost				132mm	103mm

Technical Feasibility Risk

The CCTT baseline option is rated as '8' for the technical feasibility primarily due to the need to convert CCTT Ada software into C, which would certainly introduce bugs into the primitives of the implementation. Also, the wide breadth of ModSAF capabilities that would have to be assimilated into the CCTT Environment structure means that the majority of the capabilities of both systems would be rewritten or at least modified.

The ModSAF baseline option is rated as '3' for technical feasibility since the existing STOW capabilities cover a significant portion of the CCTT Environment capabilities and the ModSAF system is already implemented and tested in C. The risk of integrating a ModSAF-derived OneSAF testbed SNE into the CCTT system is covered separately.

Program Risk

The CCTT baseline option is rated as '7' because of the expected conversion to C, as well as the need to move a tremendous breadth of ModSAF/STOW capabilities into the CCTT structure.

The ModSAF baseline option is rated as '3' because the majority of CCTT capabilities are already available or are low cost to add. This leaves plenty of time and resources to directly address the only high-risk item identified for the ModSAF option, namely CCTT interoperability.

Existing Program Impact

The CCTT baseline option is rated as '5' for its impact to ModSAF. Most of ModSAF's users are fairly flexible in terms of requirement specifications (if any). It is likely that the greatest impact would be to the industry-wide knowledge base on ModSAF's structure which would be sharply altered if using the CCTT Environment interface / architecture. Although the CCTT baseline approach may suffer instability due to the volume of code that would have to be re-engineered, this risk is covered under Technical Risk.

The ModSAF baseline is scored as a '3' for its impact to the existing community based on the incorporation of a new environment format. However, the overall impact should be minimal. The impact to the future development of ModSAF is scored as a '2' due to its correlation to the existing development. The impact to the CCTT community is scored as an '8' since the underlying language and representation will be changed. Because the analysis did not isolate a specific area that appeared unsolvable, this ranking could have been medium. However, a number of lesser issues, ranging from database generation from an Evans and Sutherland source to correlation with a different Environment representation within the CCTT system provide the possibility of problem areas. The largest risk here is not

with the capability or flexibility of the ModSAF Environment representation, but rather with the stringent demands and requirements of CCTT (including extensive integration testing, and many person years of low-level, minor coding "tweaks" to support integration with other system components or to match a user request). ModSAF has never been involved in such extensive testing where an overriding consideration was meeting the needs of a specific program.

Adaptability to new technology

The CCTT baseline option is rated as '5' in adapting to new technology. Although the CCTT Environment structure is tightly aimed at a CCTT system solution, it does provide one of the essential elements for a strongly layered, composable system. Specifically, CCTT has a strong interface that fully hides implementation details from view (i.e. it would be relatively easy to replace the implementation behind the interface). This strong interface structure will serve CCTT well if it is decided to use the OneSAF testbed SNE software on all CCTT nodes, because the Ada interface can remain the same while replacing the database primitives underneath. The CCTT baseline does suffer from two negatives relative to ModSAF: its flexibility is unproven and it would require the addition of a second layer of interface structure behind the external interface to achieve the granularity of functional division that ModSAF's libraries exhibit.

The ModSAF baseline option is rated as '5' in adapting to new technology. ModSAF's strengths are a proven track record in uses ranging from SIMNET-D sites to extensive and varied research efforts. The ModSAF baseline suffers a number of potential negatives for support across many user domains, especially "production" oriented uses; these negatives are derived from ModSAF's history as an open, research-oriented system. Little software engineering process has been applied to ModSAF (although much of it is well documented via "info" files); this has allowed developers to do what "makes sense" to them with little controlling influence from a validation or end-user perspective. ModSAF has received little system-level testing with an end objective beyond the demonstration of new technology; this means that problems that have proven annoying in the past may become major issues when faced with rigorous demands for performance and functionality. The range of contributors to ModSAF leads to a hodge-podge of libraries with no easily discernible hierarchy or relationship (e.g. libaeenviron vs libenvironment); this may lead to problems with composability and maintainability. Perhaps most important, the need to keep the architecture as open as possible has lead to interfaces which are often tightly bound to the underlying implementation. This is true even in cases where a part of the interface is abstracted away, other routines in the same header file can bind a caller to the library's implementation.

Cost

The CCTT baseline option is rated as having high cost risk because of the volume of development and re-engineering required. Since the CCTT baseline was designed to match the needs of a specific system, it is risky to assume that its structure can be extended to include the full breadth of ModSAF/STOW capabilities. Because the required up-front development effort, there will be little calendar time available to deal with unforeseen issues.

The ModSAF baseline option is rated as having low cost risk because the majority of CCTT capabilities are already available or are low cost to add. This leaves plenty of time and resources to address unforeseen issues. The ModSAF baseline cost risk would have been Medium if existing program impact were not a separate criteria.

Recommendation

The "ModSAF as Baseline" approach is selected for OneSAF testbed Synthetic Natural Environment.

Because the only high risk item with the selected approach was identified as CCTT Interoperability, a more detailed discussion of this issue follows. The remaining sections provide supporting information on differences between CCTT and ModSAF, plus the portions of each system considered part of OneSAF testbed SNE.

CCTT Interoperability

Common Software

For interoperability purposes, it would be ideal if the OneSAF testbed SNE software could replace the existing Environment CSC software (by leaving the current Environment CSC interface in place and replacing the implementation behind it). If this is not practical for programmatic / cost reasons, then the run-time differences between the OneSAF testbed database representation and the Environment CSC would have to be analyzed for impact on correlation, fair-fight, and consistent functional results. Since some portion of the CCTT Environment software is used on every CCTT run-time component, the effort to conduct correlation and interoperability experiments may exceed the cost of integrating the OneSAF testbed SNE implementation.

If the OneSAF testbed SNE representation were to be used, an Ada-to-C wrapper layer would have to be added, which would include the ability to convert from OneSAF testbed types/interfaces to CCTT types/interfaces. Also, an additional analysis would have to be conducted to cover all interfaces / capabilities required by other CCTT components but not OneSAF testbed.

Production System Hardening and Integration "Tweaking"

The final OneSAF testbed system must be fully integrated into the CCTT system, which will likely mean extensive testing and discovery of problems or issues that would not have been uncovered given the typical uses of ModSAF.

Besides extensive low-level and system testing and hardening, an extensive effort will be requiring in fine-tuning the OneSAF testbed capabilities to the user needs of CCTT. It is estimated that the CCTT Environment team invested over 5 person years of effort after development completion on modifications and extensions *besides* bug fixes. The majority of this was in short-term movement control and reaction to source data errors or omissions for database generation. But the rest ranged far and wide: altering terrain geometry from source

to avoid flipping modules on shoulders leading into wadis; new functionality supporting specific CGF units/behaviors; specialized collision capabilities for manned modules; altering mobility mappings based upon system needs such as what is visible on the PVD and/or visuals; fine-tuning dimensions of fighting positions, AVLBs, etc. to match the performance/maneuvering/collision of the modules as well as the visibility of the human trainees; database-specific filtering (e.g. of bushes in P2); database-specific optimizations (e.g. aggregate models); tweaking dynamic terrain placement routines such that the corresponding visual representation is easily seen; specialized support routines for primitive planning around relocatables (fighting position occupancy, automated exiting from vehicle defilades); supporting case-by-case user input on most major routines (e.g. many specific cases on what does or does not block a minefield breach lane based upon nearby entities, obstacles, approach angle, proximity of other breaches, etc.).

Regardless of the baseline selected, a new Environment (and SAF) implementation will expose the developers to the aforementioned integration issues, even those which have never been discovered or addressed before.

Database Generation

Regardless of the baseline approach selected, OneSAF testbed Environment should investigate solutions for database generation that do not use or rely on the "SIF++" files generated by Evans and Sutherland for CCTT. Instead, a more direct data extraction from E&S tools or, preferably, SEDRIS should be considered as replacements. Regardless of how this problem is approached, database generation is likely to be the second most costly part of integrating OneSAF testbed SNE into CCTT (second only to stability and integration effort).

Dynamic Terrain

The dynamic terrain network protocol in CCTT was aimed specifically at solving CCTT needs and is tightly coupled to the capabilities of CCTT systems (especially the visuals). As a result, OneSAF testbed will not use the CCTT Environment protocol as its native/preferred representation. CCTT Interoperability will thus require some form of gateway or conversion between CCTT's Environment Manager and the OneSAF testbed SNE capabilities. This could take place between servers (i.e. DTSim <-> Environment Manager), at the network object level (i.e. translation of CCTT PDUs to OneSAF testbed SOM), as a part of OneSAF testbed itself (i.e. a built-in capability to receive and interpret CCTT Environment PDUs), or by replacing the CCTT Environment Manager capability with OneSAF testbed's server (which would be capable of generating CCTT and OneSAF testbed network representations). Any mechanism selected must accommodate CCTT's use of unique identifiers and the visual-system derived geometry and state limitations (e.g. linear segments in multiple of 30, with breachlines possible a predefined locations).

PVD

The CCTT PVD had a specialized database to meet requirements for redraw time and for map-like displays. This included explicit storage of different levels of detail, thinning, altering geometry where required for map display (e.g. showing rivers at a standard width),

adding labels, off-line calculation of contour lines, etc. Any differences between the OneSAF testbed and CCTT GUIs will introduce additional training overhead and could result in a different "look" to the 2D display of the terrain. Depending upon the approach selected for OneSAF testbed GUIs as well as CCTT integration requirements, the OneSAF testbed SNE could be required to match the functional/display characteristics of the CCTT PVD, generate a CCTT PVD format database, or generate a dedicated OneSAF testbed PVD database to directly replace some or all uses of the CCTT PVD.

Other Databases / Unexpected Issues

The radio database may need to be generated via the OneSAF database process or replaced with equivalent capabilities, if all other terrain generation capability of CCTT is replaced.

Because the Environment CSC and PVD CSCI are used across all components of CCTT in differing ways, it is very likely that a number of unexpected uses or tight dependencies have been built between these components.

Cost Details and Comparison

The following table provides the cost estimate for each option. Note that the estimates are sometimes the same for both options (e.g. database generation). This is because the resultant C implementation would be handled much the same way regardless of the baseline chosen for the remainder of the OneSAF testbed SNE development.

Descriptions for each of the line items follows the tables....

	CCTT Baseline	**ModSAF Baseline**
Convert to C	**50k lines * 80/day = 4 pm**	**N/A**
CCTT Capabilities->ModSAF ♦ Detailed differences search ♦ Interface Level ♦ Primitives ♦ Weather / Effects	**N/A**	**11 pm** 1 pm 3 pm 4 pm 3 pm
ModSAF Capabilities -> CCTT ♦ library restructure plan ♦ libCTDB / ICTDB / DTO ♦ DVW ♦ Reasoning ♦ Other	**29 pm** 3 pm 12 pm 5 pm 5 pm 4 pm	**N/A**
OneSAF testbed Development	**3 pm**	**3 pm**

DB Generation Code/Test ♦ Detailed differences search ♦ Obstacle boundary reconst. ♦ Cut and fill (source dependent) ♦ CCTT source spec. cases ♦ SEDRIS source use	**16 pm** .5 pm .5 pm 6 pm 3 pm 4 pm dev / 2 pm test	**16 pm** .5 pm .5 pm 6 pm 3 pm 4 pm dev / 2 pm test
DB Generation/Integration for P1 and P2	**6 pm**	**6 pm**
ModSAF baseline issue investigation (execution after investigation min-max) ♦ Routing DB generation ♦ O/A Linkage use) ♦ Structure I/F investigation ♦ Near term planning tests ♦ Better DT memory mgmt	**N/A**	**8 pm (+ 5-21 pm execution)** 2 pm (+2-8) 1 pm (+0-5) 1 pm (+0-2) 3 pm (+2-4) 1 pm (+1-2)
CCTT baseline issue investigation	**12 pm (+8 - 31 pm execution)** Difficult to estimate before library restructure plan; assume 1.5 * ModSAF case.	
CCTT Interop Development ♦ No Common Reuse -or- ♦ Common Reuse ♦ DT translator ♦ Specialized DTSim module ♦ Weather/Effects ♦ CCTT DB performance and special cases ♦ PVD	**14 pm - 16 pm** 3 pm dev / 3 test -- or -- 1 pm dev / 3 pm test 3 pm 2 pm 3 pm 2 pm	**17 pm - 20 pm** 5pm dev / 4 test – or-- 2 pm dev / 4 pm test 3 pm 2 pm 4 pm 2 pm
CCTT Integration	**15 pm**	**18 pm**

Total	**107 - 132 pm**	**84 – 103 pm**

Convert To C represents the initial conversion of CCTT Environment Ada software into C. This task is not required for the ModSAF option.

CCTT Capabilities -> ModSAF covers the implementation of CCTT capabilities not covered by ModSAF for the ModSAF as baseline option.

ModSAF Capabilities -> CCTT covers the implementation of ModSAF/STOW capabilities not covered by CCTT. The subcategories represent major ModSAF areas at least partly unsupported by CCTT. The cost for the ground capabilities is so high because line-by-line integration of capabilities would be required with lots of potential for conflicting implementation decisions and test problems, especially given the fact that the CCTT implementation would have been freshly converted to C. DVW and DTO would largely replace existing CCTT capabilities and therefore would be more of a bulk conversion to fit under the CCTT architecture.

OneSAF testbed Development covers effort required to step OneSAF testbed SNE up to interface or architectural changes that result from the integration of components outside of SNE (e.g. merging of physdb and ECDI). It is assumed that other development areas cover their cost to adapt to SNE interface / structure changes.

DB Generation Code/Test covers all database generation system development and stand-alone testing. The cost is the same for both baselines since the ModSAF database generation software would serve as the starting point with either option. Cut and fill is costed high in case the source data does not have directly usable TINs (either reconstruction is required or raw polygons have errors). The use of SEDRIS as a data source is estimated under the assumption that the existing SEDRIS-to-CTDB implementation is available for our use, plus a usable SEDRIS version of Primary 1 and Primary 2 is available *prior* to the SEDRIS development effort.

DB Generation/Integration covers direct effort to build, test, integrate, and repair CCTT Primary 1 and Primary 2 databases. Estimate includes some effort for specialization of database test tools, manual and automated self consistency check, as well as manual correlation tests against source visuals. Testing of these databases from the perspective of the ModSAF user is covered as well. However, problems caused directly by database density (including cut&fill density) are *not* covered. These problems may range from short-term movement control to performance (some of this is covered under CCTT Interop).

ModSAF Baseline Issue Investigation covers detailed investigation of a number of issues uncovered during our analysis. Each of these issues was too involved to reach a quick conclusion during this analysis effort, but was recognized as having a large potential for impacting the development. The estimate is shown as: A+(B-C), where A is the effort required to conduct a detailed study of the problem and decide on a development approach, while B-C represents a rough estimate of the min/max effort based upon the study results.

- Routing database generation has the greatest possible impact depending upon what source data is available and how many problematic cases from the CCTT databases are already handled by the ModSAF database generation process.
- Obstacle Avenue linkages represent a concept CCTT implemented which avoided a number of problems encountered by ModSAF. If deemed useful, it should be fully integrating into the OneSAF testbed compilers, low-level representation, and short-term planners.
- Structured I/F investigation includes a study of ModSAF's current interface structure to see if elements of the CCTT approach would provide better encapsulation and support for future development.
- Near-term planning tests covers a parallel implementation and comparison of the ModSAF vs CCTT path planning capabilities. Entity-level movement is so critical to SAF success, that the analysis phase is given plenty of time.
- Better DT memory mgmt is intended to consider mechanisms to avoid locking an entire patch in memory when modified as a result of dynamic terrain changes. This approach may prove problematic in large CCTT exercises with hundreds of relocatables.

CCTT Baseline Issue Investigation represents the CCTT baseline equivalent of the preceding task. Because of the apparent shotcomings of the CCTT approach, this cost was not fully investigated. For purposes of this comparison, we assume the cost is roughly 1.5 times the ModSAF baseline investigation task.

CCTT Interop Development represents effort focused on connecting OneSAF testbed to the CCTT system. Some of this software would be more part of CCTT production than part of OneSAF testbed itself. See the section on CCTT Interop Issues for more details.

- No Common Reuse / Common Reuse: if the OneSAF testbed SNE implementation must correlate with both visuals and Environment CSC, additional cost is levied due to extra correlation tests and database generation software modification. If OneSAF testbed SNE replaces Environment CSC, then cost covers Ada-to-C bindings and integration testing specific to different CCTT components. Because this is an "OR" option, the total for this section is shown as a range.
- PVD is not costed pending better resolution of what is needed. See "Issues/Variables", below

CCTT Integration covers lab test time, test team support, PTR repairs, minor tweaks for customer issues, test tools, new development to support other integrators, bug tracing to SNE boundaries, and may even include modifications to CCTT software (e.g. if Environment CSC is still part of the system). Because the CCTT baseline option starts with a conversion to C, the difference between the two approaches is not very large.

Issues/Variables:

- Will CCTT common code be replaced with OneSAF testbed SNE? Both ways costed for SNE *only*?
- What are the requirements for interop with CCTT PVD? Must OneSAF testbed look the same? Will OneSAF testbed PVD replace CCTT's? Will we need to maintain correlation between CCTT and OneSAF testbed database generation systems for the

PVD? Should we directly build a CCTT PVD database from the OneSAF testbed database generation system?

- What format will P1 and P2 be available in, if not SEDRIS? What is the cost of having a third party create and maintain SEDRIS databases, including repairs as needed?

Assumptions:

- Costs cover effort required to coordinate SNE internal design, interfaces, implementation, test, etc. Estimates do not cover status reporting, metrics, process execution, documentation (beyond the level of ".info" files), execution of formal process, etc. It is assumed these costs will be added based upon an LOE multiplier or based upon delivered SLOC across the system.
- Effort does not include extensive coordination across or travel to multiple development sites. It is assumed tools / cost for this is covered separately across the system.
- Cost for OneSAF testbed "standalone/system" integration is covered via delivered SLOC or some other global estimation, not included in SNE estimates.
- Other development teams include their own cost to adapt to OneSAF testbed interface changes.
- Besides stated assumptions, no sequencing or calendar-time considerations were included.
- SEDRIS-to-CTDB compiler is available from SEDRIS research projects and SEDRIS source for P1 and P2 is available.
- Supplied SEDRIS P1 and P2 will be repaired by producer when source data problems are discovered.
- If OneSAF testbed SNE doesn't replace Environment CSC throughout CCTT system, the cost provided assumes that some problems will be accepted and/or are unsolvable. Cost covers "good" correlation, not perfect.
- Database generation costs cover generation of the equivalent capabilities of the following CCTT databases: MrTDB, MrsTDB, EMTDB with the exception that full MrsTDB capabilities supported is determined under "ModSAF baseline issue investigation: routing DB generation). The CCTT PVD and radio databases are ignored. Also, estimates assume that there is no dedicated PVD database, except for thinned CTDB format.
- SNE plans for 3 pm of investigation/comparison between ModSAF and CCTT near-term movement control approaches. It is expected that behaviors will have a similar effort which would tie-in.

Supporting Material: Functionality Differences

The following information was generated during the OneSAF testbed Environment analysis effort to search for issues, problem areas, and differences between the CCTT and ModSAF Environment capabilities. While this is not intended to be an exhaustive, fully annotated list of all implementation differences, it is an excellent starting point for such a list.

This section includes the following:

- a summary of major issues that are known to require additional investigation. Most of these issues impact both proposed approaches.

- a list of issues with CCTT Interoperability which bear further investigation
- a list of weather / effects differences
- a detailed breakdown of the CCTT Environment interfaces (not including PVD) and their counterparts in ModSAF. This information lead to the conclusion that ModSAF supports the majority of CCTT capabilities at an interface level.
- finally, a laundry list of possible differences provided largely without supporting information. The analysis effort did not allow for further investigation of the differences, but they are provided as useful information for the next phase of OneSAF testbed

General Issues

Following are some issues that will deserve front-end additional analysis as part of the OneSAF testbed project. Most of them will be influenced by the decisions of other OneSAF testbed development areas.

OneSAF testbed Development Impacts

Regardless of the baseline chosen, the SNE interface will change a least a little to accommodate the union of CCTT/ModSAF capabilities, so other OneSAF testbed developers will have to evolve with this changes. Likewise, all users of OneSAF testbed will be evolving their own areas, possibly requiring changes to existing interfaces and even introduction of new capabilities / interfaces to accurately reflect the new OneSAF testbed structure and capabilities. Obviously, these changes are difficult to cost in advance except as a project-wide estimate of interface upgrading.

The interface structure and style of Environment CSC vs the related components in ModSAF is often very different. ModSAF generally provides more low-level access with greater granularity, while Environment CSC provides a more abstract interface covering all inputs / outputs through a single set of Ada specifications. OneSAF testbed development should include an upfront analysis as to which style to adopt (independent of the chosen baseline), with an eye toward supporting future OneSAF testbed. It is the author's opinion that a stricter, explicitly hierarchical interface (i.e. enforced via subdirectories) will better serve OneSAF testbed objectives.

Environment CSC / ModSAF Differences

- Routing database generation and run-time use
- Different approach to entity-level path planning
- Different approaches to database components (CCTT compilers have flow control and call common read API; ModSAF front-ends have flow control and call a write API).

Weather / Effects Differences

This sections describes the CCTT / ModSAF Weather and Effects differences from the perspective of supporting CCTT capabilities within ModSAF's DVW architecture.

Tactical Smoke:

ModSAF supports tactical smoke based on the Army Research Lab's COMBIC model. COMBIC is a physics based obscurants model that includes diffusion, buoyant rise, and boundary layer models. ModSAF supports over 31 different smoke sources. Each evolves with time and is effected by weather conditions. Smoke opacity varies with location in the smoke plume and with time. CCTT appears to represent smoke by switching through 5 discrete representations (size and opacity) over time, under the control of the Environment manager.

Illumination

CCTT support discrete illumination states. For example, lunar illumination may be: None, Starlight, Halfmoon, Full Moon). ModSAF supports continuous, geospecific illumination based on the Army Research Lab's ILUMA model. ModSAF provides both direct and diffuse components of illumination.

Weather and Clouds

ModSAF supports 3 spatial weather models (uniform, observed, and gridded). Variable rates of snow and rainfall are supported. CCTT supports only the equivalent of ModSAF "uniform" weather. Limited parameters support haze, fog, clouds, and rain and rain-soak state.

Flares

CCTT aggregates flares to keep the IG load within system limits.

Interface Differences

Area Intervisibility
CCTT: Area_Intervisibility and Initialize_Area_Intervisibility_State
ModSAF: ctdb_point_to_point() and libtdbtool
DESC: Computes intervisibility from a point to an area.
NOTES: Libtdbtool calls ctdb_point_to_point() multiple times over an area to provide the user with an area intervisibility tool.

Awareness State
CCTT: Clear, Object_State, Remove_Object, Update_Object_State
ModSAF: Near-term movement planning (movemap) is updated by vterrain. All dynamic obstacles are updated immediately after the obstacles change state. The cross-country route planner, routemap, recognizes dynamic bridges, albeit not on a per-vehicle basis; all vehicles are instantaneously aware of dynamic bridges, regardless of side or proximity. Also, the cross-country route planner does not currently recognize minefields (although movemap does).
DESC: Used to keep track of dynamic terrain features that are relevant to route planning, such as dynamic bridges (AVLBs)

and minefields.

Collision Detection
CCTT: Detect_Collisions
ModSAF: ctdb_find_ground_intersection()
DESC: Used to find intersections of terrain features, vehicles, and/or the terrain skin with an arbitrary ray.
NOTES: CCTT's version returns lots of information about the intersected object. ModSAF's version only returns the type of intersected object and its location. If the user desires more information, s/he can access it via libCTDB's feature extraction API.

Cover and Concealment
CCTT: Initialize_Search_State
Location_Is_Hidden
Search_For_Concealment
Search_For_Cover
Search_For_Forested_Locations
Search_For_Hide
Search_For_Hide_Fire_Positions
Search_For_Outside_Forest_Locations
ModSAF: tr_search_for_cover()
tr_search_for_concealment()
tr_search_for_hide_pts()
DESC: These routines search for covered, concealed, or hidden locations.
NOTES: Search_For_Forested_Locations can be done in ModSAF by first performing a search for concealed locations and then using ctdb_point_within_abstract() to determine whether the point is "forrested."

Dynamic Environment
CCTT: Update_Prepositioned and Update_Relocatable
ModSAF: ctdb_create_linear_feature()
ctdb_modify_linear_feature()
ctdb_delete_linear_feature()
ctdb_create_volume_feature()
ctdb_modify_volume_feature()
ctdb_delete_volume_feature()
ctdb_create_abstract_feature()
ctdb_modify_abstract_feature()
ctdb_delete_abstract_feature()
ctdb_create_terrain_skin()
ctdb_modify_terrain_skin()
ctdb_delete_terrain_skin()
DESC: These routines modify dynamic terrain features.
NOTES: The ModSAF functions listed form a superset of the functionality offered by the corresponding CCTT functions.

Tactical Smoke State
CCTT: Update_Tactical_Smoke
ModSAF: libsmokesim - provides the handler for the ENV_SMOKE effect
DESC: These routines create or update tactical smoke.
EFFORT: Rewrite every invocation of these CCTT functions to use the ModSAF interface.

Database Extents
CCTT: environment: Database_Extents
ModSAF: wld_get_playbox_extent()
DESC: These routines get the geographical extents of the terrain database.

Database Origin Info
CCTT: environment: Get_Utm_Info
ModSAF: wld_get_playbox_utm_data()
DESC: Obtains information that DIS needs to perform transformations on coordinates.

Initialization
CCTT: environment: Initialize and Initialized
ModSAF: env_init()
DESC: Initializes the environmental simulation software.

Shutdown / Caching
CCTT: environment: Lookahead_Regions, Reset, Shutdown
ModSAF: no corresponding functions
DESC: These are administrative functions not required by the ModSAF environmental software.

Weather / Ambient Conditions
CCTT: environmental_effects: Get_Natural_Conditions,
Get_Natural_Conditions_Numeric, Set_Natural_Conditions
ModSAF: libenvconstant
DESC: These functions are used to set the values of the following environmental parameters:
rain state, visibility range, haze state, haze visibility range,
haze ceiling, fog state, fog visibility range, fog ceiling,
cloud state, cloud ceiling, cloud base, cloud visibility range,
lunar illumination, temperature, humidity, wind speed, rain
soak
NOTES: ModSAF's libenvconstant provides a set of values for every environmental phenomenon simulated by ModSAF:
temperature, dewpoint, relative humidity, barometric pressure,
wind velocity, precipitation type, precipitation rate, extinction
type, extinction amount, extinction coefficient, ray visibility,
visual range, illumination, sky over ground, cloud cover, cloud
ceiling, cloud height, cloud type, sim time, sun position, moon

position, moon phase, contrast, solar illumination, lunar illumination, sky illumination

Line of Sight
CCTT: line_of_sight: Get_Entity_Visibility
line_of_sight: Get_Obstacle_Visibility
line_of_sight: Get_Point_To_Area_Visibility
ModSAF: ctdb_point_to_point()
DESC: These functions provide intervisibility information between a point and an area, obstacles, or entities.
NOTES: CCTT's "Get_Point_To_Area_Visibility" function maps directly to ModSAF's "ctdb_point_to_point()" function. The other two CCTT functions currently do not have ModSAF equivalents, although it would not require much effort to write them. They would simply call existing libctdb functions which have the necessary functionality. The new functions would essentially use existing libctdb functionality and mimic the interface of the CCTT functions.

Munition Flyout
CCTT: munition_flyout: Munition_Flyout
ModSAF: ctdb_find_ground_intersection()
DESC: Returns information about where the specified line segment intersects the ground, terrain features, and vehicles.
EFFORT: Rewrite every invocation of these functions to use the ModSAF interface.

Near-term planning support
CCTT: obstacle_avoidance_map: Create
obstacle_avoidance_map: Clip_Segment
obstacle_avoidance_map: Get_Extents
obstacle_avoidance_map: Inside
obstacle_avoidance_map: Reset
ModSAF: libvterrain
DESC: This is the code that creates the per-vehicle obstacle avoidance maps. This is handled by libvterrain in ModSAF and it has no public interface.
EFFORT: none, provided we use ModSAF's method of keeping track of the local obstacle space (namely vterrain)

RWA Landing
CCTT: obstacle_avoidance_map: Find_Clear_Landing_Point
ModSAF: tr_search_for_landing_positions()
DESC: Finds the location(s) of clear landing spots within a specified search area. The landing site(s) must be free of any terrain objects.
NOTES: The CCTT version has an input parameter for a suggested location, while the ModSAF version does not. This should not be very difficult to add to ModSAF.

EFFORT: Rewrite tr_search_for_landing_positions() so that it takes a suggested location.

OA Map Retrieval
CCTT: obstacle_avoidance_map: Retrieve
ModSAF: localmap_get_movemap()
DESC: Retrieves the obstacle map for a given vehicle.
EFFORT: none, provided we use ModSAF's method of keeping track of the local obstacle space (namely vterrain)

Breach Trafficability
CCTT: obstacle_avoidance_map: Set_Funnel_Trafficability
ModSAF: no corresponding function
DESC: Tags a given obstacle as being immobile. This is used for vehicles, so that behaviors know if they are killed or disabled (and therefore essentially obstacles), or are just temporarily stopped.

NOTES: In ModSAF, an obstacle's velocity is attached to the obstacle's information, and the planner uses this. Theoretically, this functionality is already included in the ModSAF near-term movement planner.

Direct Path Creation
CCTT: obstacle_avoidance_path: Direct_Path
ModSAF: no corresponding function
DESC: Used by CCTT test programs to generate a path with obstacle avoidance turned off.
NOTES: This is not a required function.

Flight Paths
CCTT: obstacle_avoidance_path: Plan_Flight_Path
obstacle_avoidance_path: Get_Agl_Points
ModSAF: tr_contour_direction()
tr_contour_segment()
tr_contour_desired_velocity()
DESC: Functions for the calculation of flight paths that follow the contour of the terrain without crashing into the terrain skin.
NOTES: "Plan_Flight_Path" is a coarser version of "Get_Agl_Points." The corresponding ModSAF functions get the job done with one level of fidelity.

Near-term Planning
CCTT: obstacle_avoidance_path: Plan_Path
ModSAF: movemap_preplan_route()
DESC: Plan a path through the terrain, avoiding obstacles.
NOTES: ModSAF's version does not return a list of blocking obstacles if the plan failed. If CCTT behaviors really use this, it will have to be implemented in ModSAF.

Path validation
CCTT: obstacle_avoidance_path: Validate_Path

ModSAF: movemap_update()
DESC: Called periodically to ensure the path is still valid (i.e. to make sure dynamically-placed obstacles don't make the path invalid).

Retrieve Bridges
CCTT: prepositioned_object: Get_Bridges_In_Area
ModSAF: ctdb_next_abstract()
ctdb_bridge_info()
ctdb_next_volume()
DESC: Obtains a list of bridges in a specified area. For ModSAF, bridges can be represented as abstract features with microterrain for the bridge decks, or as volume features (for destroyable bridges).
NOTES: In order to make the interface nice and uniform, we may wish to write a function that will return all bridges, destroyable or not.

Entity Placement
CCTT: regional_information: Find_Clear_Placement_Point
ModSAF: no corresponding function
DESC: Places a vehicle at the specified location on the terrain. If there is an obstacle at that location, then a search for a suitable location is made along a specified vector.
NOTES: In ModSAF, intelligent vehicle placement for units is done by libformationdb, but there are no such checks done for individual vehicles. To write such a function should not be difficult.

Highest Post in Area
CCTT: regional_information: Highest_Post_In_Area
ModSAF: no corresponding function
DESC: Finds the (x, y) location of the highest elevation post in a given area. Microterrain is not considered.
NOTES: There is currently no function in ModSAF to do this, but writing one would be trivial.

Point Off Road
CCTT: regional_information: Point_Off_Road
ModSAF: no corresponding function
DESC: Computes a point normal to the current road segment at a specified distance away.
NOTES: There is currently no function in ModSAF to do this, but writing one would be trivial.

Region Classification
CCTT: regional_information: Regional_Classification
ModSAF: no corresponding function
DESC: Provides a classification of the type of area around the supplied location. Possible return values are Region_Open, Region_Woods, and Region_Urban.

NOTES: There is currently no function in ModSAF to do this. Writing one would not be very difficult.

Defilade Exit Point
CCTT: relocatable_object: Find_Defilade_Exit_Point
ModSAF: ctdb_get_next_abstract()
DESC: Returns the exit point for a fighting position object.
NOTES: This information is available in the abstract data for a fighting position feature in ModSAF.

Retrieve Breaches
CCTT: relocatable_object: Get_Breach_Objects
ModSAF: no corresponding function
DESC: Returns the relocatable objects breached by or breaching a given object.
NOTES: This can be achieved in ModSAF via a pbtab search (to find the breachers) or via ctdb_next_abstract (to find the breachees).

Relocatable Object Retrieval
CCTT: relocatable_object: Get_Relocatable_Object
ModSAF: ctdb_get_next_abstract()
DESC: Provides data on dynamic terrain objects; differences derive from storage and representation differences

Fighting Position Dimensions
CCTT: relocatable_placement: Get_Dimensions
relocatable_placement: Get_Interior_Dimensions
ModSAF: ent_get_physdb()
DESC: Gets the corresponding entity type of a relocatable object so its physical dimensions can be retrieved.

Movable Bridge Placement
CCTT: relocatable_placement: Place_Bridge
ModSAF: libubrdgbrch
libvposdeploy
libvattach
DESC: Places a dynamic bridge across an obstacle on the terrain.
NOTES: The code that does the checking to see if a given location is suitable for placing a bridge is contained within libubrdgbrch and is not public.

Not Investigated…
CCTT: relocatable_object: Get_Smoke_Object
relocatable_object: Get_State
relocatable_object: Give_Cc_Point_In_Fighting_Position
relocatable_object: Give_Points_In_Infantry_Fighting_Position
relocatable_object: In_Fighting_Position

relocatable_object: In_Infantry_Fighting_Position
relocatable_object: In_Vehicle_Fighting_Position
relocatable_object: Parallelpiped_Around_Ro
relocatable_object: Relocatable_Inside_Polygon
relocatable_object: Relocatables_In_Area

ModSAF: Mostly specialized variations of ctdb_get_next_abstract, some may require additional information be supported in CTDB or from a model library.

List of Functional / Representational Differences to Investigate

This list captures some underlying capabilities within the Environment CSC that may be different from ModSAF's implementation. Each of these will need to be investigated as part of the OneSAF testbed integration.

External Interface

Selectable tree trunk intersection for munition flyout
Awareness state (for minefields)
Maneuver_intent indicates whether a fighting position is an obstacle or an objective.
Routing corridor widths (for bridges)
Selectable road/cross/direct during throughout proposed route
Use of handle for multi-level history
Unit normal returned for height of terrain (MM only)
Env CSC received entity information at interface (no use of DIS code)
Env CSC provided geometry primitives plus general utilities for use by other CSCs

Functional support

Riverbed geometry for height of terrain (centerline linear problem)
Trunk collisions
Collisions with bridge/overpass sides (selectable via environment variable)
Collisions with shoulders (steep terrain) (selectable via environment variable)
Collisions account for "bumper height" (modules, by config file)
Collision detection using full volumes
Munition flyout hits span (and names it), span sides
OA recognizes over/under for overpasses and bridges
Generic map utility (used for placement, OA, etc.)
Tree trunks, foliage radius, foliage height and distance above ground used throughout
All around "tweaking" for integration (visuals, modules, operators, behaviors, etc.)

Terrain (Storage / retrieval)

Validation header (byte-wise storage at beginning to verify file type, purpose, and endianness)
Centerline linear reconstruction problems (roads for mobility/dust, wide rivers, riverbed geometries)

Aggregate models with virtual gridmasks and plane equations for performance
Patch-crossing bridges/overpasses (1=many and side representation)
What happens when an abstract feature (like a fighting position) crosses a storage boundary?
Or are abstract locations "global" with quadtree only as a referencing tool?

Routing

Verify storage includes a segment for each point to nearest obstacle?
Overlapping obstacles (forest gap with spanning steep slope)
Obstacle boundary reconstruction or use of primitive areals from source in compiler
Verify simplification doesn't create errors (S-curves)

Example issue: obstacle part way up between two forests that overlaps both.

- Can't draw from each forest to obstacle
- Can't "not draw" because last segment on each side would imply trafficability between

Dynamic Modifications

Use of PO and RO ID's as unique identifiers issue with patch crossers?

Need to verify use / reaction of every relocatable object:

- visual appearance
- functional effect behind Environment interface
- data exported via Environment interface
- protocol encoding limitations (e.g. linear segment lengths)
- CCTT breaches don't break one thing into two things
- path planning (through and around, including fighting positions as obstacles and complex destinations)

External callers given geometric and abstract view of data (e.g. outer boundary and C&C positions within).

Compilers

Ramps leading to spans…connection checks?
We did all tree and building placement (Z value)
Had to use "special" (non-visual) areals for terrain types
Had to remove linears "underneath" bridges and overpasses
Linear boundary reconstruction (esp. wide rivers) difficult
Fighting position interior geometries "built" by compiler
Obstacle/avenue linkages created (C&F through steep slopes, multi-level terrain, bridge over water, road over no-go)
Region classes calculated based upon density parameters from config files.
Forest boundaries reconstructed from aggregate instances.
Basis set roads NOT available via source (not used)

Spans are n-sided beasts, must search for entry and exit edges.
Config file based mapping of terrain type number to "meaning" (road, water, deep, impassable)
Can use E&S "microterrain" from source? Otherwise, reconstruct ground for abstract C&F.
Can get forest boundaries as areals?
Labels reconstructed from different source (not SIF) for PVD
Intersection models overlap with surrounding cut and fill
Configurable damage assessment per building model and span

Utilities / Test

GIDGET provides access to all external routines (include OA map, etc.), plus dumps raw feature data. Simple 2D display for debugging, plus VRML viewer for selected regions. Need to ensure that the comparable ModSAF tools support similar stand-alone test capabilities.

Supporting Material: Scope

General Description

The OneSAF testbed Synthetic Natural Environment (SNE) Assessment covers the following areas:

- Terrain/Environment/Weather databases and their compilers
- Software that operates directly on (or encapsulates) these databases
- Software that provides an abstract answer based mostly on primitive data from Terrain/Environment Databases
- PVD display software only to the extent that it directly operates on and draws Environment data. Includes contour lines, tactical smoke, etc. Does not include overlays, overall GUI design, non-environment entities, etc.
- All environment "servers" which manage the state of terrain, environment, weather

CCTT components

For CCTT, the provided definition of SNE includes all or part of the following CCTT CSCIs/CSCs. This list includes all things that *could* be part of OneSAF testbed SNE, even though some may be shared with other areas (e.g. behaviors, GUI) and/or may not be included in OneSAF testbed because they are not required by CCTT SAF.

- Database API CSC COMMON DATABASE UTILITY
  Provides a common API to retrieve database source information. Hides details of source data from compilers and provides selected database features based upon filtering regions. Includes capabilities common to multiple database compilers.
- SAF Terrain Compiler CSC DATABASE COMPILER
  Target database compiler that generates MrTDB and MrsTDB (for use by the Environment CSC), and EMTDB (for use by the Environment Manager CSCI).
- PVD Compiler CSC DATABASE COMPILER

Target database compiler that generates the PVD database for use by the PVD CSCI.

- Radio DB Compiler CSC DATABASE COMPILER
  Target database compiler that generates the radio database for use in communications degradation.
  NOTE: Radio DB will probably not be part of OneSAF testbed, since it is not used / required by CCTT SAF.
- Environment CSC COMMON SERVICE
  Encapsulates representation of terrain/features. Provides wide range of operations (line of sight, collision detection, route planning, short-term path planning, cover and concealment, etc.). Used by almost all CCTT components.
- PVD CSCI COMMON SERVICE
  Provides graphical display of terrain in map-like form.
- Environment Manager CSCI CCTT APPLICATION
  Acts as "owner" of the CCTT Environment, by determine prepositioned object states and authenticating creation/modification of relocatable objects.
  NOTE: Environment Manager may not be part of OneSAF testbed.

ModSAF libraries

For ModSAF the provided definition of SNE includes all or part of the following libraries. This list includes all things that *could* be part of OneSAF testbed SNE, even though some may be shared with other areas (e.g. behaviors, GUI).

Terrain Database

- CTDB SHARED DATABASE SERVICE
  Represents underlying terrain, and provides services to determine visibility, collisions, surface orientation and soil type, elevations, etc., and support graphical displays of the geography.
- World: SHARED DATABASE SERVICE
  Responsible for management of collections of terrain databases and data structures associated with them.

Dynamic Terrain

- DTOConst: NETWORK DATABASE UTILITY
  Provides a database of DTO features and translations for the network.
- DTACTDB: SHARED ARCHITECTURE SERVICE
  Provides ability to recieve incoming ECNs and make the appropriate modifications to the CTDB.
- DTAgent: NETWORK MANAGEMENT SERVICE
  Process of validating incoming ECNs and firing of appropriate callbacks.
- DTSim: SIM MANAGEMENT SERVICE
  Provides Dynamic terrain simulation services
- ECN: NETWORK ARCHITECTURE SERVICE
  Provides the infrastructure for sending and recieving ECNs.

- Repoly: SIM MANAGEMENT SERVICE
  Librepoly provides DTO_POLY_DATA strutures and functions used to repolygonalize terrain.
- Obstacle: GUI ARCHITECTURE C2OBJ
  Provides ability to create obstacles from a graphical user interface.
- Obstutil: SIM ARCHITECTURE C2OBJ
  Utility library that handles generic creation of obstacles.
- DitchUtil: SIM COLLECTIVE UTILITY
  Provides common functionality for ditch entity processing.
- Bridge: SIM PHYSICAL UTILITY
  Bridge simulation utility functions.
- MSO: SIM PHYSICAL UTILITY
  Multi-state objects simulation utility functions.

Dynamic Virtual Worlds

- EnvAtm: SIM ENVIRONMENT SERVICE
  Environmental model for atmospheric transmissivity.
- EnvCloud: SIM ENVIRONMENT SERVICE
  Environmental model for transmissivity through natural sky water clouds.
- EnvConstant: SIM ENVIRONMENT SERVICE
  Constant environment model.
- EnvContrast: SIM ENVIRONMENT SERVICE
  Contrast environment model.
- EnvDB: SIM ENVIRONMENT SERVICE
  Gridded Weather database model.
- EnvDust: SIM ENVIRONMENT SERVICE
  Vehicle dust model.
- EnvDustIV: SIM PHYSICAL UTILITY
  Vehicle dust intervisibility effect model.
- EnvEnt: NETWORK ARCHITECTURE SAFOBJ
  Environmental Entity management.
- EnvIllum: SIM ENVIRONMENT SERVICE
  Illumination environment model.
- EnvInit: SHARED ENVIRONMENT UTILITY
  A library for initializing other environmental libs
- Environment: SIM ARCHITECTURE SERVICE
  A library for dispatching requests for environmental state to the appropriate handler.
- EnvObsWx: SIM ENVIRONMENT SERVICE
  Observed Weather environmental model.
- EnvSea: SIM ENVIRONMENT SERVICE
  Collection of simple sea models.
- EnvSimple: SIM ENVIRONMENT SERVICE
  Collection of simple environment models.
- SmkInt: SIM PHYSICAL UTILITY

Determines intervisibility effects of obscurant clouds.

- SmokeRdr: SIM DATABASE UTILITY
  Accesses data related to obscurant clouds stored in reader format.
- SmokeSim: SHARED ENVIRONMENT SERVICE
  The main ModSAF interface to the COMBIC smoke model (works in conjunction with SmokeRdr and SmkInt.)
- Weather: SIM ENVIRONMENT SERVICE
  A simple networked weather distribution library. Uses EnvEnt, Environment, and SimReq.
- FlareSim: SHARED ENVIRONMENT SERVICE
  The main ModSAF interface to the signal flare model.

Modified to Use DVW / Possible SNE

These libraries were modified or developed via the DVW project. Some of these libraries may largely "belong" to other areas, such as behaviors or vehicle modelers, but impacted by DVW.

- Waveresp - hull response and sinking
- Visual
- Tracked
- Vtargeter - "Target DR" to estimate target position when obscured by own smoke
- Seaedit - Ocean editor
- Envgedit - Gridded Weather Editor (aka Hand of God)
- Simreq - Utility library for load leveling smoke and flares over PO
- Ssmk

Reasoning Libraries

- EnvAssess: NETWORK ARCHITECTURE SAFOBJ
  An environmental assessment functionality library.
- EnvReason: SIM ARCHITECTURE SAFOBJ
  Provides reasoning functions to take environmental effects into accounts
- TeReason: SHARED ARCHITECTURE UTILITY
  Terrain reasoning related utility functions.

Route Planing and Movement

- MoveMap: SIM ANALYSIS SERVICE
  Generates courses through space-time which achieve a specified goal without violating physical constraints or colliding with obstacles.
- LocalMap: SIM ARCHITECTURE SAFOBJ
  Creates and destroys one libMoveMap movement map per vehicle.
- Route: SHARED MANAGEMENT UTILITY
  Generic route manipulation routines.
- RouteMap: SIM ANALYSIS SERVICE

Unit level planning service.

- Collision: SIM PHYSICAL SAFOBJ
  Detects collisions with terrain features and other network entities.
- CollPred: SIM PHYSICAL SAFOBJ
  Predicts potential collisions and calculates simple avoidance maneuvers.
- MESRoute: SIM ARCHITECTURE SAFOBJ
  Performs unit routing through Multiple Elevation Surface objects.
- VMove: SIM INDIVIDUAL SAFOBJ
  Individual-level task for vehicle movement.
- VTerrain: SIM INDIVIDUAL SAFOBJ
  Individual-level local terrain awareness task.
- UTraveling: SIM COLLECTIVE SAFOBJ
  Unit-level traveling task.

Map Drawing

- TactMap: GUI COMMAND SERVICE
  Tactical map display which includes terrain features and an arbitrary number of dynamically created lines, text, pixmaps, pictures (defined using a simple language), BGRDB icons, and application-drawn objects.
- PVD: GUI ANALYSIS SAFOBJ&SERVICE
  Provides a set of controls for tactical map display, defines map interaction modes (zoom, pan, info), stealth/preview controls, and updates positions of network entities on tactical map.

Environment Assessment: CCTT as baseline

Technical Description

The initial effort associated with using CCTT Environment as the OneSAF testbed SNE baseline would be conversion of the software from Ada to C. C is extensively used throughout the simulation community, and C++ is now the language of choice in many simulation production programs (e.g. the JSIMS family). While Ada would surely meet the functional needs of many users, the research community which has traditionally relied on ModSAF would certainly prefer C over Ada.

Because the CCTT Environment CSC was oriented towards meeting CCTT requirements while ModSAF Environment capabilities were expanded throughout STOW, the main focus of OneSAF testbed SNE effort based upon CCTT code would be to move the extensive ModSAF capabilities over into the CCTT Environment architecture.

Thus, the Ada-to-C conversion would be followed by a transfer of a significant amount of capability from ModSAF into the CCTT structure.

Advantages of using the approaches

This approach would provide three benefits:

1) The extensive effort of refining the CCTT Environment software to meet the specific demands of CCTT would largely be retained. However, some of the finer capabilities may inadvertently be lost in the translation to C.
2) The CCTT Environment CSC has a strong, localized interface which completely isolates SNE software from the remainder of CCTT. CCTT Environment doesn't even rely on network services. This strong structure would automatically be retained using this approach.
3) CCTT interoperability would be improved, but it would not be guaranteed because of the conversion to C and the inclusion of a wide range of new ModSAF/STOW capabilities.

Disadvantages of using the approach

The disadvantages of this approach are numerous. An obvious negative is the need to translate the existing Ada software into C. Perhaps 25% of the original CCTT Environment implementation started from a C-to-Ada conversion from an earlier ModSAF version. As a result, the final software would sometimes bear a strong resemblance to existing ModSAF code, but it will just be "freshly developed" and thus require extensive testing.

A second negative is that the CCTT Environment structure was designed to meet CCTT requirements and thus is not necessarily able to support the full breadth of Environment capabilities implemented in ModSAF for STOW.

The biggest disadvantage of this capability is the simple fact the ModSAF largely supports the capabilities that CCTT Environment does but the opposite is not true. Using CCTT as a baseline would mean that almost all software in the OneSAF testbed implementation will

have been converted from Ada or will have been restructured to fit under the CCTT Environment architecture. In contrast, using ModSAF as a baseline would provide a large, stable base of capabilities from the start, with point modifications as necessary to match CCTT capabilities and support CCTT interoperability.

Because the cost, effort, and stability disadvantages of this approach are so great compared to the ModSAF approach, the remainder of the description of the CCTT baseline approach is brief. Most analysis effort is focused on the ModSAF baseline option, CCTT Interoperability, and enumeration of differences between CCTT and ModSAF Environment software.

Technical Feasibility

The CCTT baseline option would be risky from a technical standpoint due to the need to convert CCTT Ada software into C, which would certainly introduce bugs into the primitives of the implementation. Also, the wide breadth of ModSAF capabilities that would have to be assimilated into the CCTT Environment structure means that the majority of the capabilities of both systems would be rewritten or at least modified.

Program Risk

Requirements satisfaction

The OneSAF testbed SNE implementation that would result from a CCTT baseline option should meet most or all requirements levied separately on CCTT and ModSAF. However, full CCTT interoperability would not necessarily be guaranteed by this approach because of the conversion to C and incorporation of many new capabilities.

Performance

The initial Ada-to-C conversion of CCTT software would probably not perform as well as the existing ModSAF Environment capabilities unless a separate optimization step had been performed on the converted software (because a straight conversion does not necessarily mean comparable efficiency between languages).

Schedule

The CCTT baseline approach would require a significant amount of extra development. Schedule risk would be much higher for this option compared to the ModSAF baseline option.

Reliability

The CCTT baseline approach would be much less reliable than the ModSAF baseline option because of the sheer volume of software to be modified.

Scalability

Both Environment options would have comparable scalability. As discussed elsewhere, the CCTT Environment interface may support a stronger boundary between SNE and consumers, while ModSAF's greater granularity of libraries and low-level access is more flexible.

Existing Program Impact

Fair Fight

See the separate section on CCTT Interoperability for an overall discussion of this topic. While the CCTT baseline option would better support CCTT fair fight and correlation, these items would still be at risk due to the planned conversion to C and incorporation of new capabilities.

Correlation

While the CCTT baseline option would better support CCTT fair fight and correlation, these items would still be at risk due to the planned conversion to C and incorporation of new capabilities.

Baseline Maintenance

Because of the planned conversion to C, this option provides no particular benefit for CCTT baseline maintenance.

Database Generation

Because the ModSAF database generation process has been proven with more data sources (and supports capabilities like endianness and TINs), the CCTT database generation process would only be retained to the extent required to support CCTT software that is not replaced by new OneSAF testbed capabilities and/or CCTT compiler capabilities that are not supported by ModSAF's compilers.

Adaptability to new Technology and Architectures

The CCTT baseline option is rated as having medium risk in adapting to new technology. Although the CCTT Environment structure is tightly aimed at a CCTT system solution, it does provide one of the essential elements for a strongly layered, composable system, namely a strong interface that fully hides implementation details from view (i.e. it would be relatively easy to replace the implementation behind the interface). This strong interface structure will serve CCTT well if it is decided to use the OneSAF testbed SNE software on all CCTT nodes, because the Ada interface can remain the same while replacing the database primitives underneath. The CCTT baseline does suffer from two negatives relative to ModSAF: its flexibility is unproven and it would require the addition of a second layer of interface structure behind the external interface to achieve the granularity of functional division that ModSAF's libraries exhibit

Ease of coordination and integration of multiple developers

No significant advantages / disadvantages for the CCTT baseline relative to the ModSAF baseline approach.

Effort to migrate related to capabilities

Detailed cost is not captured for this option because it is so apparent from the analysis that use of the CCTT baseline would be much more costly than the ModSAF baseline.

A rough description of the cost is as follows:

1) Cost to convert CCTT Environment-related capabilities from Ada to C (~50,000 lines of Ada)
2) Cost to move STOW capabilities from ModSAF architecture to the converted CCTT Environment architecture (unknown number of lines)
3) Cost to integrate libCTDB internal differences (endianness, TINs).
4) Cost to support ModSAF-based programs (e.g. STOW's TAOS)

Assumptions on migration environment

None relevant to selection of approach.

Environment Assessment: ModSAF as Baseline

Technical Description

This approach involves using ModSAF as the baseline and implement any differences between CCTT Environment and ModSAF Environment in the ModSAF baseline. The following capabilities will be maintained:

1) Representation of all of ModSAF's and CCTT's terrain features, including:
- microterrain
- buildings
- trees
- treelines
- tree canopies
- roads
- rivers
- craters
- ditches
- berms
- dragons teeth
- rock drops or point blocks
- wire road blocks
- concertina wire
- minefield fences
- single fences
- fighting positions
- bridges
- camouflage canopies
- rubble
- snow
- log crib
- abatis

2) 256 soil types
3) Both diagonalizations in a single grid
4) TIN topology
5) Earth curvature support
6) UTM support
7) unlimited maximum geographic extent (databases can span multiple UTM zones)
8) dynamic terrain, including dynamic creation, modification, and deletion of terrain features as well as modification of the terrain skin
9) forests represented as basis sets
10) simulation of the following environmental phenomena:
- temperature
- dewpoint

- relative humidity
- barometric pressure
- wind velocity
- precipitation type
- precipitation rate
- extinction
- type
- extinction amount
- extinction coefficient
- ray visibility
- visual range
- illumination
- sky over ground
- cloud cover
- cloud
- ceiling
- cloud height
- cloud type
- sim time
- sun position
- moon
- position
- moon phase
- contrast
- solar illumination
- lunar
- illumination
- sky illumination

11) near-term movement planning
12) cross-country movement planning

Advantages of using the approaches

Feature support by CTDB is a superset of those supported by CCTT SAF. CTDB format 1 and MrTDB were very similar in terms of functionality, but then they branched off and simultaneously evolved along separate paths because they were driven by different requirements. A few years later, however, in an effort to produce a common terrain database that was available in both MrTDB and CTDB formats (for a SAF interoperability exercise), features that MrTDB supported that CTDB did not support (i.e. basis sets and model references) were added to CTDB. At that point, CTDB's functionality became a superset of MrTDB's functionality.

After the CCTT Environment baseline approach was determined (by late 1994), CCTT engineers were unable to incorporate any ModSAF ideas. In contrast, ModSAF often

purposefully or coincidentally incorporated changes that CCTT had made from the common baseline. A quick list includes:

Purposeful integration of CCTT/MrTDB changes into ModSAF:

- variable diagonals by post, not by database
- use of models (trees, buildings) including opacity by model (not global) and trunk radius
- aggregate models for forests
- support for CCTT mobility codes, including wet mapping
- support for building CCTT databases
- development of an E&S to CTDB compiler

Coincidental (developed separately for CCTT and ModSAF):

- Feature retrieval engine
- Use of terrain type palettes
- Better approach to cut and fill than microterrain (i.e. TINS)
- Reintroduction of min/max Z by patch
- Full overpass/bridge capability (correct dynamic entity avoidance, LOS, etc.)
- More generic approach to dynamic terrain (fighting positions, tank ditches, etc.)

Thus, ModSAF has largely "kept up" with CCTT/MrTDB changes, but the opposite has not been true since around the time STOW (ICTDB) started.

ModSAF, primarily via ICTDB and other STOW work, has added a lot of functionality never in the CCTT baseline. The effort to move all of these capabilities to CCTT is much greater than moving CCTT into ModSAF. This is overwhelmingly true in the weather/ocean/effects area, but even the libCTDB side has some major additions including endianness, TINs, GCS, and an advanced soil attributes storage representation which allows the representation of an arbitrary amount of information associated with the terrain skin soil types, including FACC codes. ModSAF/STOW also added the representation of building interiors via MESs. This includes the representation of buildings with multiple floors, internal walls, windows, doors, and stairways.

Because CCTT Environment has a strong, localized interface and few dependencies on other parts of CCTT (this was required because it was used everywhere in CCTT), it will be much easier to clearly identify what it takes to replace the Environment CSC interface and capabilities. It would be harder to clearly delineate ModSAF's environment from the rest of the system (different interfaces for libenvironemnt vs libCTDB, libCTDB is used directly at a "primitive" level, dependency on DTSim for repoly, etc.).

Finally, ModSAF is in C, and thus is closer to the "right" language to use, namely C++.

Disadvantages of using the approach

There are several components currently used with CCTT SAF that will still be used "as is" in the OneSAF testbed system (AAR, for example). These components will need to use the MrTDB representation of the terrain database. If the OneSAF testbed used CTDB, then in order for everything to interoperate properly, there must be perfect correlation between the MrTDB representation and the CTDB representation of a given database. This perfect correlation will be difficult to achieve because of the storage representation differences between MrTDB and CTDB. As an example, MRTDB stores cut and fill features as parameterized abstractions (i.e. bank slope, river depth, etc.). CTDB stores them as TINs. Therefore, cut and fills are almost guaranteed not to correlate. This issue is discussed more in the Cost and CCTT Interoperability sections.

Technical Feasibility

There is very little technical risk involved in implementing this approach.

A small number of functions will have to be written in order to fill the gap between MrTDB's functionality and CTDB's functionality; the functions which exist in MrTDB's public interface will have to be imported or reimplemented in CTDB. However, these are few and far between.

Program Risk

Requirements satisfaction

This approach satisfies all of both SAFs' requirements. Obviously, it satisfies all of ModSAF's requirements because it uses the ModSAF code. It also satisfies all of CCTT SAF's requirements because ModSAF's representation and simulation of the synthetic natural environment is almost a superset of CCTT SAF's representation, and the differences will either be reimplemented in OneSAF testbed, or migrated from CCTT SAF into OneSAF testbed.

Performance

Performance of ModSAF's SNE code is likely higher than that of CCTT SAF, since ModSAF has undergone more concentrated performance improvements in libCTDB. There is, however, a potential that the ModSAF architecture may not satisfy the performance requirements of CCTT SAF. Currently, there is no way to evaluate this.

Schedule

Since this approach doesn't involve any new radical implementation methods that have never been attempted before, there is little schedule risk.

Using ModSAF as the baseline is much less risky schedule-wise than using CCTT SAF as the baseline because less functionality has to be moved.

Reliability

The synthetic natural environment code will be as reliable as ModSAF's synthetic natural environment code, which has been proven to be very reliable. The STOW ACTD had an average of 10 hours uptime for Joint SAF during the 48-hour exercise, and ModSAF 4.0 is expected to be even more reliable.

Using ModSAF as the baseline will most likely be more reliable than using CCTT SAF as the baseline because it will involve far fewer changes, and the ModSAF baseline is stable and proven. Also, using CCTT SAF as the baseline would involve converting the ADA code to C code, which will likely introduce numerous errors.

Scalability

This approach has the same scalability characteristics as the current ModSAF; the size of database that can be handled depends on the percentage of the database that exists within physical RAM. The size of the terrain cache can be configured on the command line at startup. If the terrain cache is too small, excessive thrashing can occur while running cache-unfriendly scenarios. This thrashing can adversely affect system performance.

Existing Program Impact

Correlation

Since this approach uses only the ModSAF SNE architecture and not a hybrid of both SAFs, there are no correlation issues to address in the OneSAF testbed product. There are, however, correlation issues with respect to interoperating with existing CCTT components which will not be affected by the unification.

Baseline Maintenance

No differences from CCTT as baseline option.

Database Generation

Because the ModSAF database generation process has been proven with more data sources (and supports capabilities like endianness and TINs), the CCTT database generation process would only be retained to the extent required to support CCTT software that is not replaced by new OneSAF testbed capabilities and/or CCTT compiler capabilities that are not supported by ModSAF's compilers.

Adaptability to new Technology and Architectures

Because ModSAF is intrinsically a research and development tool, it was designed to be extensible and to accept new technologies. Its architecture is flexible and proven. It is for this reason that basing the synthetic natural environments on ModSAF will result in the OneSAF testbed code being more extensible, since CCTT SAF was designed in order to meet specific requirements, it is not as extensible as ModSAF.

Ease of coordination and integration of multiple developers

The ModSAF baseline has been proven in heavy parallel development on many platforms. An integration facility has been developed to maintain a constantly changing baseline.

C.4 Physical Models Assessment

Physical Model Assessment: Option 1

Start with the ModSAF Baseline and incorporate CCTT models.

Technical Description

Start with the ModSAF physical model infrastructure, and incorporate the CCTT physical models on a case-by-case basis. Models will be converted from Ada to C. Wherever possible, a CCTT model and its corresponding ModSAF model will be unified. If the unification of two models is not possible, then they will exist separately in OneSAF testbed.

In transferring the CCTT models to the ModSAF infrastructure, the code will be modified to work with the ModSAF architecture while maintaining the CCTT specifications as described in the CCTT CGF Physical Model Design Document.

Advantages of using this approach

- Maintain the extensibility of the ModSAF architecture
- Takes advantage of ModSAF's "Plug and Play" capability
- Maintains the capabilities of both systems' models
- The transferring of the implementation of the CCTT models facilitates the V&V process. The V&V process would be more involved if starting from the CCTT model specification.
- The requirements on the ModSAF side are automatically satisfied.

Disadvantages of using this approach

- V&V would have to be redone for the transferred CCTT models.
- Where the unification of models is not feasible, multiple versions of a model need to be maintained.
- Incorporating updated capabilities to a unified model tends to be costly.
- The transferring of models from one infrastructure to another introduces risk:

    1. In order to make a CCTT model "fit into" the ModSAF infrastructure, the hooks to the infrastructure may have to be changed.
    2. Due to the differences in the runtime characteristics of the infrastructures, the models may have to be modified in order to make them more adaptable to ModSAF's runtime characteristics.

Technical Feasibility

Due to the extensible nature of the ModSAF architecture, this approach carries very little technical risk.

Program Risk

This section outlines the program risk for this option.

Incorporation of CCTT models into the ModSAF architecture is, for the most part, achievable.

Requirements Satisfaction

Because of the strong similarities in the physical models architecture between CCTT and ModSAF (namely, common use of the "Generic Model Interface"), the CCTT software should be very amenable to straight translation with few difficulties. The primary difficulty will lie with breaking some of the larger CCTT capabilities/packages into smaller, separate components matching the library decomposition of ModSAF. For example, the tracked hull package in CCTT is primarily designed for hull dynamics, but it also contains the tent extension model, because all tent simulation required in CCTT was associated with tracked vehicles. When moving this code over, it will have to be broken into independent pieces, such as hull dynamics, tent simulation, articulated parts, etc. A secondary concern is matching the physical model simulation with appropriate controllers, although this will need to be addressed cooperatively between physical models and behaviors; in general this will impact implementation decisions more than requirements satisfaction.

CCTT's functional requirements should be easily matched by the conversion to C and incorporation into the ModSAF architecture. The primary requirements risk will be in the areas of validation and integration test (i.e., verification of fair fight).

Performance

Because the CCTT physical models code is well suited to an Ada-to-C translation, it is expected that there will be little or no variation in performance. While there may be a slight loss of specialized performance tweaking because of the language translation, it is likely that this performance variation will be matched by the Ada vs. C performance difference. Thus performance risk is minimal, with the recognition that the translated CCTT models may need some fine-tuning once tested in C to meet specific CCTT loading requirements.

Schedule

This effort will be very safe from a schedule standpoint. Because ModSAF has its own variation of most CCTT physical model capabilities, the translation of CCTT models can be done in whatever order desired and either sequentially or in parallel as deemed most appropriate for supporting other teams' efforts. Because the translation will be relatively straightforward, there will be plenty of calendar time to resolve issues.

However, if there are any specific model capabilities that are required by specific CCTT behaviors, the development of those behavioral models must follow the completion of the porting of the CCTT physical models. Therefore, the schedule of the behavioral model development may be negatively impacted.

Because of the likelihood of unforseeable problems arising during the testing of the CCTT models with their new behaviors, there is a reasonable risk of schedule impact during the later phases of the program.

Reliability

There will be minimal reliability risk with the physical models implementation resulting from this approach. The greatest issue will be the need to conduct thorough integration testing as part of the CCTT system to verify that the Ada-to-C translation is as stable as the original Ada implementation. Besides direct translation problems, the interaction of the controller systems and near-term path planning must be verified together with the physical models since these components depend so heavily on each other.

Once verified and tested, the reliability of the final system should be no worse or no better than that of either ModSAF of CCTT SAF.

Scalability

With respect to physical models, the scalability of the final system should be no worse or no better than that of ModSAF.

Existing Program Impact

V&V

As part of the OneSAF testbed and CCTT system integration effort, the physical models should receive much the same kinds of V&V they received as part of CCTT, except that less detailed analysis will be required since the algorithms themselves have already been validated. Thus, the effort can focus primarily on ensuring that the implementation is free of Ada-to-C translation errors rather than conceptual/algorithmic flaws. Also, system integration should include "fair fight" experiments oriented toward the manned modules, although the existing CCTT SAF implementation could also be used for some comparisons if necessary.

No major risks are anticipated with V&V, although the LOE may be high.

The transferring of the implementation of the CCTT models facilitates the V&V process. The V&V process would be more involved if starting from the CCTT model specification.

Extensibility

The extensibility of the final system will be the same as that of ModSAF because the ModSAF infrastructure is being used as the baseline.

It is possible that there will be notable improvements in the extensibility of the final system over the current CCTT implementation. This is due the fact that functionality currently

wrapped into large packages in CCTT will likely be broken down into smaller components which will be better-suited to localizing problems and swapping atomic components for improving or adding functionality.

Baseline Maintenance

In general, ease of maintenance of the final system will be equivalent to that of ModSAF. However, multiple models resulting from non-mergeability will have a slightly negative impact on ease of maintenance.

Adaptability to new Technology and Architectures

There is nothing inherent in this change that prevents adaptability to new technologies or architectures when compared with the existing systems.

Ease of coordination and integration of multiple developers

This approach should facilitate coordination and integration of multiple developers in a manner similar to the current ModSAF baseline.

Cost

Enumeration of capabilities to migrate

In the following list, use the tables in the rest of the document as an enumeration of the models to migrate.

1. Update ModSAF GMI to handle all CCTT model capabilities.
2. Migrate each GMI-based CCTT physical model into ModSAF.
3. Migrate each non-GMI-based CCTT physical model into ModSAF (damage, etc.)
4. Unify similar CCTT/ModSAF models into one model (may be done during the migration if assessed as technically feasible).
5. Perform validation testing.

Effort to migrate related to capabilities

The following table shows the relationships between existing CCTT and ModSAF GMI-based models. Non-GMI-based models (such as damage) are not included in this analysis.

CCTT Package	CCTT P-Sloc	ModSAF P-SLOC	ModSAF Equivalent	Ratio
Architecture				
		628	libcomponents	
Dynamics				
Hull	7,474	168	libhulls	

Tracked	1,993	4879	libtracked	2.45
Wheeled	1,635	2932	libwheeled	1.79
RWA	2,913	2416	librwa	0.83
FWA	2,647	2359	libfwa	0.89
Missile	642	3008	libmissile	4.69
Stationary	864			
DI	1,588	1560	libdi	0.98
Turret				
Turret	1,244	678	libgenturret	0.55
Resources				
Resource Models	2,884	3197	libsubcomp	1.11
Weapons				
Weapon	7,195	11	libguns	
Ballistic	1,875			
Projectile	5,017	5151	libbalgun	1.03
Mlauncher	1,937	2222	libmlauncher	1.15
		1809	libcmbtmodels	
		1019	libflyouteq	
Sensors				
Sensor	2,265	19	libsensors	
Radar	1,726	1545	libgenradar	0.90
Optical	6,998	5420	libvisual	0.77
		444	libsensorutil	

		731	libpdetection	
Total	**50897**	**40196**		**0.79**

The conclusions that can be drawn from this table are as follows:

1. In general there is a good correlation between CCTT packages and ModSAF functionality for physical models. In some cases, certain models are not represented in ModSAF (such as the stationary hull).
2. In many model-model comparisons, the ratio of ModSAF physical SLOC to CCTT physical SLOC is near-unity. In cases where this ratio does not hold, there may be good explanations. For example, ModSAF's missile hull contains specific Phoenix missile modeling, which may account for it's large size as compared with CCTT's model.
3. ModSAF has more "architectural" components separated as individual libraries. However the total comparable SLOC has ratio is very close to 1 (0.79). This would indicate that the CCTT model suite is comparable to ModSAF's model suite in terms of code size. From this fact, it may be possible to draw the inference that the model complexity of the two suites of models is equivalent.

If the last point is true, than the cost to migrate CCTT models into ModSAF should be no-worse than the development of a new ModSAF physical model. Since it is a migration and not a new development, the cost could actually be much less. Using this assumption, the following table provides a costing estimate of model development effort, based on a constant per-model cost.

Bottom's Up Model Development Estimate		
Number of GMIs	5	
Per-GMI Infrastructure Update	0.5	
Subtotal		***2.5***
GMI-Based Models	16	
Per-Model Reimplementation	1.5	
Subtotal		***24***
Non-GMI-Based Models:	5	
Minefields		
Damage (Direct, Indirect)		
Smoke,Flare Special Effects		
Per-Model Reimplementation	1.5	
Subtotal		***7.5***
Total Man-Months		**34**

Assumptions on migration environment

- Expect to be able to convert the CCTT SAF implementations over to the ModSAF infrastructure.
- Existing CCTT SAF physical models follow the CCTT CGF Physical Model Design Document. This ensures that the required functionality of the CCTT models is not lost in the conversion process.
- There are no drastic differences between CCTT SAF's physical model interface and ModSAF's GMI. Any discrepancies between the two interfaces can be resolved without significant effort by ModSAF's extensible GMI infrastructure.

Assessment

Criteria	Low (1)	Medium (5)	High (10)	Score
Technical Feasibility Risk	The technical feasibility is rated very good.	The technical feasibility is good but the migration involves some very difficult technical issues.	The technical feasibility is poor. The migration technical is very difficult or the approach has never been tried before.	1
Program Risk	There is little risk of the migration approach	There is a moderate risk of the migration approach	The migration approach is very risky and has never been attempted before	1
Impact to existing ModSAF user community	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Impact to existing ModSAF development community (e.g., STOW)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Impact to existing CCTT program	The approach has virtually no impacts to the existing programs or	The approach has some impacts to the existing programs that could be over	The approach has many impacts, the effort involved in overcoming the impacts are very	2

	community	come with moderate effort	large, or technically challenging	
Impact to current CCTT development (e.g., FBCB2)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	2
Adaptability to new technology risk	The resulting infrastructure or environment can be easily extended	The resulting infrastructure does not support extension and will require additional enhancements to support future modification	The resulting infrastructure is not unified and should be completely redesigned to support extensibility	2
Cost				34 Man-months

Criteria Assessment Justification

Technical Feasibility Risk

This approach is completely feasible and is assessed as a "1". ModSAF's physical model architecture was designed to support multiple versions of models.

Program Risk

The program risk is assessed as "1" because all requirements should be able to be met. There already exist very strong similarities in both ModSAF and CCTT SAF's physical model architectures and implementations. Maintenance of fair fight will be addressed as part of validation testing.

Existing Program Impact

Current programs should see no impact. CCTT will have to support revalidation of the models, but this is true of all approaches. Future programs will have only positive impacts as they will have a larger suite of physical models to choose from for reuse. Any model maintenance under CCTT PDSS or any new model development for CCTT will have a more difficult migration path into this ModSAF based OneSAF testbed architecture. However, since the GMI approaches are similar and since the models will be translated to have the same algorithms, this impact should be relatively low.

Adaptability to new technology

Newly developed physical models within future programs will have a straightforward migration path into OneSAF testbed. However, new technology coming from OneSAF research, such as an O/O implementation of physical models that can exploit specialization through inheritance is not addressed by this approach.

Physical Model Assessment: Option 2

Start with the CCTT Baseline and incorporate ModSAF models.

Technical Description

Use CCTT physical model architecture as a basis and re-implement the suite of physical model capabilities from ModSAF. Where possible, unify common models. In addition, rework the architectural approach to invert the structure (remove the dispatch table approach for a registration approach).

Advantages of using this approach

- The primary advantage of this approach is that the CCTT requirements are met by the direct use of the CCTT models in their native architecture.

Disadvantages of using this approach

- The CCTT architecture was not designed to be extensible.
- The GMI would have to be expanded to incorporate the new model types and capabilities.
- The resulting rework of the model architecture would result in a similar approach to ModSAF.

Technical Feasibility

The technical feasibility of this approach is high. There are no technical risk items in this rework and incorporation of new models.

Program Risk

This section outlines the program risk for this option.

Requirements Satisfaction

The CCTT requirements would be met directly since the CCTT models are being directly reused. It is anticipated that the incorporation of the ModSAF models and the inversion of the architecture would meet the ModSAF requirements.

Performance

Performance requirements would not be adversely impacted by this option.

Schedule

There would be a moderate schedule risk since the architecture would be inverted to incorporate ModSAF extensibility.

Reliability

This option does not adversely impact reliability.

Scalability

This option does not adversely impact scalability. Scalability should be the same as the existing systems.

Existing Program Impact

Requirements Satisfaction

It is anticipated that existing requirements will be met and provide little impact to existing programs. There would be some level of analysis that would need to be performed on the common models. Specifically, a divergence between the model approaches would need to be examined and the expectations of each user examined to determine if both models would need to be supported (i.e. both tracked models are required based on the usage of the models).

V&V

This change would provide minimal impact to V&V since the CCTT models are being reused.

Extensibility

This change would increase the extensibility of the baseline architecture of CCTT by the incorporation of ModSAF concepts. The resulting extensibility would be equivalent to the current ModSAF extensibility.

Baseline Maintenance

There is no adverse impact to baseline maintenance for this approach. However, if it is determined that two implementations of a specific model are required, both set would need to be maintained in parallel for selected changes.

Adaptability to new Technology and Architectures

There is nothing inherent in this change that prevents adaptability to new technologies or architectures when compared with the existing systems.

Ease of coordination and integration of multiple developers

This approach should facilitate coordination and integration of multiple developers in a manner similar to the current ModSAF baseline.

Cost

Enumeration of capabilities to migrate

- ModSAF Models missing from CCTT baseline.
- Inversion of the architecture to match the ModSAF extensibility.

- Incorporation of the CCTT models into the new inverted architecture.

Effort to migrate related to capabilities

Capability	Months
GMI	1
ModSAF Models	34
CCTT Models	13
Inversion	2
Total	50 mm

Assumptions on migration environment

- The majority ModSAF entity component models are represented in CCTT.
- The entities that need to be added do not provide significant conflicts with existing CCTT component models or approaches.
- The data associated with the ModSAF models is expressed in a similar manner to CCTT data.

Assessment

Criteria	Low (1)	Medium (5)	High (10)	Score
Technical Feasibility Risk	The technical feasibility is rated very good.	The technical feasibility is good but the migration involves some very difficult technical issues.	The technical feasibility is poor. The migration technical is very difficult or the approach has never been tried before.	3
Program Risk	There is little risk of the migration approach	There is a moderate risk of the migration approach	The migration approach is very risky and has never been attempted before	3
Impact to existing ModSAF user community	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	2
Impact to existing ModSAF development	The approach has virtually no impacts to the	The approach has some impacts to the existing	The approach has many impacts, the effort involved in	4

community (e.g., STOW)	existing programs or community	programs that could be over come with moderate effort	overcoming the impacts are very large, or technically challenging	
Impact to existing CCTT program	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Impact to current CCTT development (e.g., FBCB2)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Adaptability to new technology risk	The resulting infrastructure or environment can be easily extended	The resulting infrastructure does not support extension and will require additional enhancements to support future modification	The resulting infrastructure is not unified and should be completely redesigned to support extensibility	2
Cost				50 mm

Criteria Assessment Justification

Technical Feasibility Risk

This is scored as a '3' based on the need to rework the existing architecture for the models.

Program Risk

This is scored as a '3' based on the effort required to rework the architecture.

Existing Program Impact

Impact to the current ModSAF development community is listed as a '2' due to the re-architecture effort will re-implement the existing models and provide a slightly different model. Impact to ModSAF developers is listed as a '4' since the architecture and models are different and may require a rework of their systems. Impacts to CCTT are scored as a '1' since the underlying models and approaches are the same as are currently on CCTT SAF.

Adaptability to new technology

This is scored as a '2' since the resulting architecture will be extensible.

C.5 User Computer Interface Assessment

UCI Assessment

Use ModSAF UCI Capabilities as a basis and add the missing UCI capabilities from CCTT SAF.

Technical Description

Common goals, independent on which path taken, are to use C (not Ada) in OneSAF testbed, support running OneSAF testbed on SUN, SGI, PC/Linux, Dec Alpha, and AIX.

It has been directed from PM-CATT that the only UCI related interfaces that must be preserved in CCTT are the components that are shared across the system. This implies that the CCTT PVD application is untouchable. Though the look-and-feel of the CCTT SAF workstation may be modified, certain key-features, such as the ability to have multiple-windows open to edit and control multiple units, must be maintained.

The migration approaches for UCI capabilities are as follows.

1. Use CCTT SAF UCI Capabilities as a basis and add the missing UCI capabilities from ModSAF.
2. Use ModSAF UCI Capabilities as a basis and add the missing UCI capabilities from CCTT SAF.

The first approach has been considered but is not seen as a reasonable migration approach due to the lack of data-driven extensibility in the basic CCTT SAF UCI. It is understood that the final product, an extensible, C-based, data-driven GUI should look like the ModSAF architecture instantiated with both ModSAF GUIs and CCTT SAF GUIs. Starting with a CCTT SAF base, recoding for extensibility, and adding the ModSAF GUIs is seen as an obviously more costly approach than starting with a ModSAF base and adding the CCTT SAF GUIs.

The only reasonable approach then is to take ModSAF as the baseline and to add the following CCTT SAF specific UCI capabilities into ModSAF.

In addition, the following enhancements will need to be made to the new baseline:

PVD API

Add a new interface to ModSAF to communicate to the CCTT PVD using the existing CCTT PVD API.

The CCTT PVD uses the CCTT PVD API to communicate with the other components via a shared memory message queue. An interface to this shared memory must be implemented to allow ModSAF to use that PVD. That interface includes tactical map operations, and the overlay editor (including graphics).

ModSAF PVD

Modify the existing ModSAF UCI architecture to allow the ability to turn off (at startup time) the ModSAF PVD and use the new interface instead. There are several functionalities that currently depend upon having the ModSAF PVD available and these will all have to be adjusted:

- When running without the ModSAF PVD, do not tick the PVD vehicle subclass. Also, libpvd should not register for various events (such as detonations), and should not install the terrain redraw bitmaps handlers.
- Editors that depend upon the ModSAF PVD for input need an alternate method to obtain input. The editables that these editors use will need to be enhanced. One option is to add a button to the editable(s) that will popup a window that has a list of the available objects that make sense. For example, the Location3D editable would be modified to have this button. When pressed, it would popup a window that listed all the existing locations (i.e. Point Objects) and would allow the user to select one.
- All libraries that depend upon the existence of the ModSAF PVD would have to make sure that it exists before trying to perform operations on it. Alternately, add passive error detection to libpvd to simply ignore requests when running without a PVD.
- Pass an additional argument to the UCI related library init routines to indicate that there may be no PVD. For example, libpvd would be told not to create a PVD, libtactmap would be told not to install a tick routine to update the PVD, etc.

Multi-Instance Popup Editor Support

CCTT SAF supports multiple pop-up editors open to view the task-organizations and missions of different units. This supports concurrent control of multiple units by moving from one window to the next. This approach is fundamentally contradictory to ModSAF's tiled windows. The GUI infrastructure must be enhanced to support multiple instances of the same editor running using different data. This may have impact to architectural libraries such as libeditor and libSAFGUI.

Advantages of using the approaches

All ModSAF UCI Capabilities will already be done, preserving the inverted, data driven ModSAF UCI architecture. The architecture would remain extensible and maintainable. In addition, the existing ModSAF UCI does implement much of the CCTT SAF UCI functionalities.

Disadvantages of using the approaches

Due to requirements, two different PVDs will exist and could present limitations in terms of maintainability and extensibility. The look and feel of the CCTT UCI mostly likely will not be preserved.

Technical Feasibility

This approach is technically feasible. Except for the new shared memory API to communicate to the existing CCTT PVD, nothing in this approach is technically challenging. There will be a need for a thorough feature-by-feature comparison during the design and test phases of the effort to guarantee that no user-visible capabilities are lost from CCTT.

Program Risk

Requirements satisfaction

All ModSAF requirements will be easily met. All CCTT SAF requirements will also be met, but the look and feel of the CCTT UCI will not be preserved.

Performance

Performance will be comparable to the existing ModSAF UCI performance, and the CCTT PVD will be unchanged.

Schedule

There is a low to moderate risk in implementation of some of the more complex GUIs and interfaces. For example, the interface to the CCTT PVD may be difficult to implement in ModSAF, and it's integration with the rest of the system may be difficult to achieve. Also, the Execution Editor may be difficult to implement, as was demonstrated by the effort to implement it in CCTT.

Reliability

The ModSAF UCI architecture will be minimally changed and therefore should work as reliably as it does now.

Scalability

Scalability will be comparable to the existing ModSAF UCI scalability. Large CCTT exercises will continue to use the CCTT PVD, which can meet current CCTT performance requirements. Since the CCTT behaviors will be implemented, similar user workload should result in this system.

Existing Program Impact

System capability Requirements Satisfaction

The current system capability requirements for ModSAF and CCTT SAF will be met.

Extensibility

Extensibility will be maintained. The ModSAF UCI architecture was designed with extensibility in mind.

Baseline Maintenance

There will be an impact to baseline maintenance since there will need to be support for both the ModSAF and CCTT PVDs. Also, the API to the CCTT PVD will not be under OneSAF testbed control, so external modifications to that API will require OneSAF testbed modifications and new releases.

Adaptability to new Technology and Architectures

The UCI capabilities in the OneSAF testbed will be as adaptable to new technologies as ModSAF currently is. If the CCTT PVD API changes, the new interface on the ModSAF side would have to change as well.

Ease of coordination and integration of multiple developers

The ease of coordination and integration of multiple developers will be equivalent to ModSAF's current capabilities since this UCI capability work will be based on ModSAF.

Cost

Enumeration of capabilities to migrate

See spreadsheet below.

Editor	Capabilities			Man-Months
Main Window				2
	Load/Save Exercises (scenarios)			
	High-level edit of exercises			
Exercise Editor				12
	Execution Matrix			
		Scenario-time use		
		Run-time use		
		Order Inserts		
		Triggers		
		Lists of applicable CISs		
		Order Parameter Displays		
		Order Decomposition		
		Auto-generation of order parameters		
Unit Editor				
	Task Organization			3
		Attach/Dettach/Cross-Attach		
		Unit Parameters (weapons loads)		
		Palette of Units		
			Predefined (heterogenous, homogenous)	
			User Defined	
		Doctrinal Unit Markings		
	Command Overrides			1
		(like Immediate Interventions)		
Unit Status Window				1
	Weapons Status			
	Speed Status			
	Behavior Status			
	Damage Status, etc.			
Report Window				1
	Filters			
	Select			
	Delete			
Alert Windows				1
	Select			
	Delete			
Operator Notification				1
	Pre-defined User Inputs			
Vehicle Control Editor				2
	Crew-behaviors			
		Fire		
		Tether		
		Goto Point		
Spot Report Processings				2
	(like Commander's View)			
PVD Interface				12
	Map Interface			
	Overlay & Graphics Interface			
Total Man-Months				**38**

Assumptions on migration environment

The CCTT PVD component of CCTT SAF cannot be modified, and a new interface would have to be added to ModSAF to hook up to the CCTT PVD API. It is assumed that any memory layout requirements to communicate with the shared memory can be satisfied via a new ModSAF C library interface.

There are roughly 40 libraries that comprise the ModSAF GUI. However, a small subset forms the core (libeditor, libtactmap, libpvd, libbgr, etc.).

There are roughly 70 functions in the CCTT PVD API. For each of these functions, it is expected that they will be created in the OneSAF testbed UCI API.

There are roughly 50 enumerated messages in the CCTT PVD API. Each of these will have to be created in the OneSAF testbed UCI API.

Assessment

For each assessment criteria (each row), fill in the score columns and make an initial assessment of the weight for each category. Describe you rationale for each weight value. The Low/Medium/High columns qualitatively describe the meaning of each value for a particular criterion.

Criteria	Low	Medium	High	Score
Technical Feasibility	The technical feasibility is rated very good.	The technical feasibility is good but the migration involves some very difficult technical issues.	The technical feasibility is poor. The migration technical is very difficult or the approach has never been tried before.	4
Impact to existing ModSAF user community	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Impact to existing ModSAF development community (e.g., STOW)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	1
Impact to existing CCTT program	The approach has virtually no impacts to the	The approach has some impacts to the existing	The approach has many impacts, the effort involved in	4

	existing programs or community	programs that could be over come with moderate effort	overcoming the impacts are very large, or technically challenging	
Impact to current CCTT development (e.g., FBCB2)	The approach has virtually no impacts to the existing programs or community	The approach has some impacts to the existing programs that could be over come with moderate effort	The approach has many impacts, the effort involved in overcoming the impacts are very large, or technically challenging	3
Program Risk	There is little risk of the migration approach	There is a moderate risk of the migration approach	The migration approach is very risky and has never been attempted before	2
Adaptability to new technology	The resulting infrastructure or environment can be easily extended	The resulting infrastructure does not support extension and will require additional enhancements to support future modification	The resulting infrastructure is not unified and should be completely redesigned to support extensibility	2
Cost				38mm

Criteria Assessment Justification

Technical Feasibility Risk

The feasibility is rated as a "4" because of the potential integration difficulties with the CCTT PVD, and the complexity inherent in the Execution Editor.

Existing Program Impact

There will be little to no impact to existing ModSAF users and developers as ModSAF's GUI architecture is only being enhanced, not replaced. There will be a moderate (4) impact to existing CCTT users and documentation, as the GUIs will change. There will be a slightly less (3) impact to future CCTT development, as it is anticipated that such future development will have minimal GUI modifications.

Program Risk

The only program risk is that the new GUIs may have some modified or missing functionality with respect to the current CCTT SAF system. This must be mitigated during design, implementation, and test, through the repeated use of side-by-side comparisons of features.

Adaptability to new technology

The adaptability risk is low (2), as ModSAF's architecture has basic extension capabilities. However, this approach does not take advantage of any new developments in GUI technology (e.g., Java, Swing, pluggable look-and-feel, HTML, VRML, low-cost 3D, etc.) which would likely be part of an ultimate OneSAF development.

Cost

See spreadsheet.