OneSAF Objective System: Toolkit Supporting User and Developer Lifecycles within a Multi-Domain Modeling and Simulation Environment

Robert L. Wittman Jr., Ph.D.
MITRE Corporation
3045 Technology Parkway,
Orlando, FL 32826
321-235-7601
rwittman@mitre.org

LTC John Surdu*
US Army
3045 Technology Parkway,
Orlando, FL 32826
321-235-7653
John.Surdu@peostri.army.mil

Abstract. OneSAF Objective System (OOS) is the next generation simulation system planned to provide the U.S. Army with an entity-level simulation to serve three modeling and simulation (M&S) domains. The Army's primary need for developing the OOS is centered on reducing duplication of M&S investments by replacing multiple legacy simulations. The number and types of users impacted are significant. In addition, the OOS is postured as an open-architecture, open-source application. As such, the OneSAF program will put source code into to the hands of a vast number of software developers within and supporting the DoD. In developing a toolkit supporting the various use cases, the OneSAF program studied the lifecycle needs of the perspective users and developers to include development, deployment, maintenance, exercise/experiment preparation, scenario execution, and post execution analysis/AAR.

1. INTRODUCTION

The One Semi-Automated Forces (OneSAF) Objective System (OOS) is the United States (U.S.) Army's next generation, composable, entity based simulation system. It is being developed to provide an integral simulation service to the Advanced Concepts and Requirements (ACR), Training, Exercises, and Military Operations (TEMO), and Research, Development, and Acquisition (RDA) domains. With requirements ranging from closed-form analytical support to command level human in the loop training, OneSAF will be a High Level Architecture (HLA)/Distributed Interaction Simulation (DIS) compliant simulation providing a common solution for a broad range of user requirements. In order to realize the OneSAF vision an innovative collaborative development facility has been established to promote a team-based approach to simulation system development. This facility is called the STRICOM Integrated Development Environment (IDE).

The OneSAF concept originated in January 1996 following an extensive study that came to the conclusion that the Army was caught in a wasteful spending cycle, making identical or similar enhancements to legacy simulations across three different user domains.

Furthermore, it was determined that none of the existing legacy simulations were capable of being extended to provide comprehensive support of emerging Army functional requirements and technical standards.

Realizing this, the Army decided the best approach for overcoming the problems associated with the multitude of aging simulations was to create a single general-purpose entity level simulation. This solution relies on using lessons learned from successful simulation projects like the Modular Semi-Automated Forces (ModSAF) simulation, and the Close Combat Tactical Trainer (CCTT) SAF [10].

In May of 1997 the Deputy Commanding General (DCG), Training and Doctrine Command (TRADOC) approved the Mission Needs Statement (MNS) for OneSAF which stated [9]:

"The need for OneSAF capabilities is not a response to a specific warfighting threat against the force; the need is driven by the guidance to reduce duplication of M&S investments, foster interoperability and reuse across M&S domains, and meet the M&S requirements of the future force."

* The work described in this paper represents the combined efforts of dozens of OneSAF developers over several years, including contractors, government engineers, and user representatives. Much of this effort represents an extraction and summarization of information available on the developers' Web portal, written by several OneSAF developers.

Shortly thereafter, a Cross-Domain Integrated Concept Team was established to design a simulation acquisition strategy, begin harmonizing user requirements, and perform early architecture analysis. In FY05 team OneSAF expects to complete the development of the OOS threshold capabilities.

2. THE EVOLUTION OF THE ONESAF LIFECYCLE AND USER ROLES

Well before the PM OneSAF began awarding OOS developmental contracts, the government's technical team was doing its homework. The government's Architecture-Integrated Product Team (A-IPT) led by PEO-STRI (STRICOM at the time) with representatives across the ACR, RDA, and TEMO domains, MITRE, and Alion performed a systematic analysis across the Army modeling and simulation domains to set a firm foundation for the development. As part of their charter to focus the execution of the OneSAF program, the A-IPT began development of the OOS Product Line Architecture Framework (PLAF) in April of 1999.

In December, 2000 after capturing the initial architecture concepts within the PLAF and the Operational Concept Document (OCD) the A-IPT shifted from architecture development into a contractor selection and education mode. For a description of the products and processes leading up to the OneSAF task order awards please see [1]. Apart from the PLAF the individual products of the OCD include:

- End State Scenarios: seven from the ACR domain, two from the RDA domain, and six from the TEMO domain,
- Four representative Operational Architectures,
- A set of OneSAF use cases – a representative set from each domain,
- A OneSAF exercise lifecycle overview, and
- The set of OneSAF user categories.

These products set the stage for OneSAF's development. Although both the user categories and lifecycle have evolved over the past 4 years the fundamental use and importance of these products has not changed. Both continue to play a key role in formulating and developing the tools and components within the OneSAF PLAF. The next sections discuss each of the fifteen phases of the OneSAF lifecycle and the fourteen user roles.

3. THE VARIOUS SIMULATION PHASES AS SUPPORTED BY OOS

This section introduces the OneSAF Product Line Architecture, its supporting products, and the processes used in the architecture's development. OneSAF developers have identified fifteen lifecycle phases of a simulation. Most simulation products support a subset of those phases, but OOS supports all fifteen with tools and/or processes. These phases, and their relationships, are shown in Figure 1, the "Bubble Chart."

3.1 Knowledge Acquisition/Engineering

The Knowledge Acquisition and Knowledge Engineering (KA/KE) process collects accurate and validated descriptions of the significant elements of the mission space that are to be modeled within OneSAF. This domain description of the mission space is typically described from a real-world point of view. In addition, KA/KE work products include characteristics and performance data used to model weapon systems and military platforms and behavior specifications. The KA/KE process is supported by detailed conceptual modeling (CM) efforts. The CM process helps describe those aspects of the phenomenon being described that are germane to the simulation system to help focus both the KA and modeling efforts. The KE process turns that domain knowledge into useful descriptions of entities, units, and behaviors that can be implemented by modelers.

3.2 Product Line Development

OneSAF product line development is the underlying software development and integration activity that produces useful components and products that can be customized for specific use. This development and integration includes Product Line architecture development, including supporting infrastructure, tools, and examples. Development and integration of models is a key component of this development. The system requirements for the product line are captured in the *Product Line Requirements Specification* (PLRS). The Product Line Architecture is described in the *Product Line Architecture Specification* (PLAS).

The PLAS and PLRS are documents, used as tools, to facilitate and drive development of the software baseline. They contain component specifications sufficiently well defined to support autonomously developed components. The components are integrated and tested according to the processes documented in the *System Integration Plan* (SIP) and the referenced V&V and T&E plans. Development of software employs the OneSAF *Integrated Development Environment* (IDE). Team OneSAF, has in fact, built and deployed tailored IDE's to help collaborative developers become more productive and to integrate them within the base program development processes. Software and data are combined using the *System Composer* product to create system compositions. System composition can be thought of as a collection of OneSAF components that meet a specific simulation use case. The output of Product Line development includes OneSAF software (source code), initial data repositories, and assembled compositions able to be distributed to OneSAF users during the Product Line Deployment phase.

3.3 Product Line Deployment

OneSAF product line deployment and Installation puts OneSAF in the hands of its users. Deployment and installation uses the Software Installer Tool to correctly place the software on the proper hardware resources at a

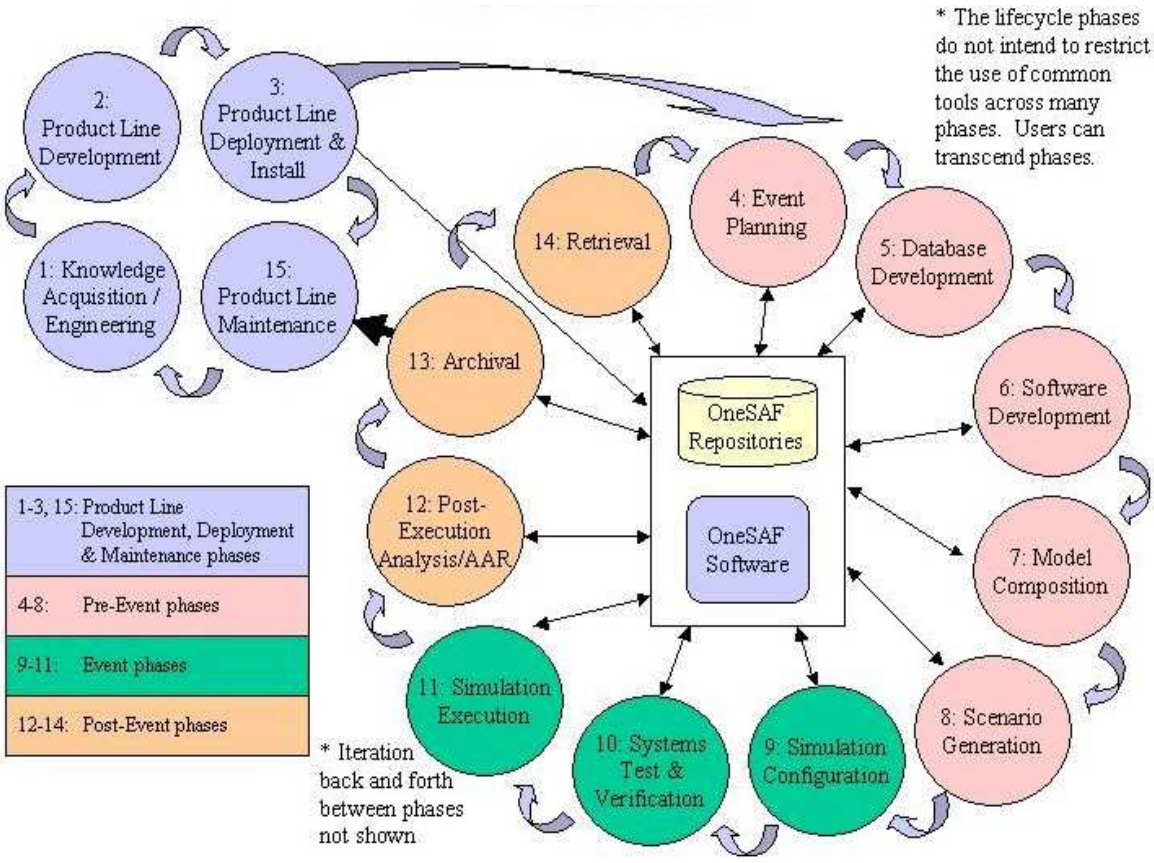

Figure 1: The OOS Bubble Chart

user site – if the site meets basic system requirements. Deployment and installation are not seen as one-time activities. Initial deployment and installation provides 'bootstrap' capabilities via installed Product Compositions sufficient to allow further tailoring of Products and Components as desired.

3.4 Event Planning

OneSAF event planning occurs prior to OneSAF use, and depending on the complexity or uniqueness, may occur weeks to months prior to a simulation event. Typical training use cases will use training objectives as inputs to this phase, while typical analysis use cases will require experiment objectives. A military scenario will be developed via the use of the *Military Scenario Development Environment* (MSDE) tool.

This scenario describes the nature of the synthetic forces to be represented, the required environment conditions, graphical control measures that are part of the plan, geographic locations of the units, and the tasks assigned to those units. A critical portion of the event-planning phase is the determination of modeled interactions required in the event. These may include C4I interactions that must be simulated or stimulated by the synthetic force. Measures of effectiveness (MOEs) and measures of performance (MOPs) are identified in the event planning phase to make sure that the particular OneSAF synthetic force used in the event provides the correct data or stimulates the required interactions needed to achieve training or analysis goals. The operational architecture requirements are captured as *system composition requirements*, and the specific requirements for the synthetic force and environment are recorded, as well as data collection requirements.

3.5 Database Development

The database development phase imports external data and processes it for OneSAF use. These activities include terrain data import – not generation – and the incorporation of characteristics and performance data needed by OneSAF models. *The Environmental Database Generation Environment (EDGE)* produces complete and consistent STF representations of the environment. These are augmented with data file representations of Ultra-High Resolution Buildings (UHRBs). Compilers supported by the various environment representation and visualization developers (such as the Environmental Runtime Component [ERC]) use these STFs and UHRB files to produce runtime terrain databases and 3D visual databases that support runtime access and environment visualization. All of these environment work products are stored in the *Environment Repository* and are made available to OneSAF components that require source STF or derived environmental data products. The compilers and the *UHRB Editor* are tools that will be developed as part of the software baseline.

In addition, characteristics and performance data from the KA/KE repository or from other external sources (e.g., AMSAA) is imported into the *Parametric and Initialization Repository (PAIR)* using repository tools. This data will be used during runtime to support OneSAF models. The selection of which data to import may be governed by the synthetic force requirements of the military scenario and other information captured as KA/KE work products.

3.6 Software Development

The software development phase includes any necessary software modifications required for a specific OneSAF use. This development can include the encoding of new or modified physical models, development of behavior primitives, and modifications or enhancements to the OneSAF infrastructure. Test and re-verification of these modifications will be supported by unit tests and regression tests that help the software developer ensure that the OneSAF software functions correctly even after software modification. Meta-data describing the characteristics of new software models are stored in the *PAIR* for use by later composition tools.

3.7 Model Composition

The model composition phase produces new behaviors, entities, and units for use in a specific OneSAF scenario. Behaviors are composed as combinations of behavior primitives and other behavior compositions. Behavior composition is done with the *Behavior Composer* tool. Entity composition includes the assembly of physical and behavioral capabilities into combat platforms, including individual combatants and non-combatants. Entity composition is done with the *Entity Composer* tool. Unit composition, using the *Unit Composer* tool, includes the association of groups of entities and units into military organizations, as well as the assembly of physical and behavioral capabilities specific to the unit. It also includes the definition of command, support, and communication relationships. Environment composition is also performed in this phase. The *Environment Composer* retrieves existing runtime and plan-view display databases and can compose runtime environment databases from multiple geographic locations. (The Environment Composer will not be fully developed when version 1.0 is fielded.) The importer/exporter tool is used to create jar files of changes and enhancements to the baseline, distribute them to remote locations, and extract those jar files into the correct locations within the PAIR.

In addition, this phase includes the development of a dynamic environment runtime configuration that includes specification for dynamic effects such as weather.

3.8 Scenario Generation

The MSDE tool (mentioned in section 3.4) is used for scenario generation as well as event planning. During the scenario generation phase, the exercise or experiment director sets the initial conditions of the simulation and allows the users create their initial plans. The MSDE tool allows planning to be distributed to multiple users/planners and then reintegrate the separately developed plans into a single exercise scenario. Scenarios are described in a vendor-, platform-, and simulation-independent, XML-based language, called the *Military Scenario Definition Language* (MSDL), which is under consideration as an industry standard. The final step in this process is to import the MSDL formatted file into the *Management and Control Tool (MCT)* to create the OneSAF simulation specific scenario file.

3.9 Simulation Configuration

The simulation configuration phase uses various tools to configure a OneSAF site with the required executables distributed to the appropriate hosts. The heart of this activity is the *System Configuration and Asset Management Tool* (SCAMT) that uses underlying OOS services to correctly configure the various OneSAF applications to execute the scenario. During this phase, resource estimation is performed to estimate required CPU, network, and disk resources required by the scenario. Processor allocation is performed by the SCAMT and applications are assigned to hosts appropriately. In addition, configurations for C4I interfaces (protocols, versions, registries, address books, etc.) are assigned based on what has been specified in the simulation scenario. If federating with external HLA systems, mapping between native OneSAF data and external HLA FOM information is performed during this phase.

Resource estimation is accomplished though the use of performance prediction parameters developed from data from previous OneSAF executions. Network estimates are produced, including the use of the *Network Loader* tool to provide sample loads for the purpose of capacity estimation and verification.

3.10 Systems Test & Verification

When a Simulation Configuration has been established and executables have been distributed to the required hosts, Systems Test and Verification can begin. The Systems Test phase will include dry-run execution to produce sample execution results, simulation data collection, and sample system performance data. The outputs of this dry run are stored in the Simulation Output Repository. System Verification is performed based on the results of the dry run execution.

Analysis of the System Test will employ Analysis and Review tools to ensure that the required data has been collected and that the system is performing as expected. Verification of data collection will ensure that enough information has been collected to generate desired AAR and analysis reports. The specification of data to be collected is done using the *Data Collection Specification Tool.* Once verification of data collection has been performed, analysis results will be produced in support of system-wide acceptance. Negative results may lead to revisiting previous phases in order to correct deficiencies.

Positive results allow the next phase to begin in confidence.

3.11 Simulation Execution

The simulation execution phase is the runtime component of the OneSAF lifecycle. System control is employed to start and stop scenarios, provide human operator control inputs, schedule checkpoints and restarts, and to modify data collection requests. Much of the simulation management is done with the SCAMT (described earlier) and the *Federation Management Tool.* System monitoring is performed to include system health and load status, federation management, as well as monitoring the real-time collection of data. The output of simulation execution includes the data and performance logs, as well as final execution results. Results from actual performance logs can be used to adjust parameters for future runtime performance predictions. Checkpoints are stored in the system output repository, and later retrieved to restart the simulation from a given point.

3.12 Post-Execution Analysis/AAR

The post-execution analysis or after action review phase performs the analysis on the data produced in the execution phase. During this phase, data is imported into tools and manipulated to produce AAR or analysis result work products, including actual MOEs and MOPs. The *Data Collection Specification Tool* is used to subscribe to required data, and that data is either fed into an SQL database by the *AAR* tool or is stored in an XML file for later post-analysis by more sophisticated tools. The results of analysis are stored in the Simulation Output Repository, along with the original data. In addition, it should be noted that incremental logging of data is allowed during AAR while the simulation is executing.

3.13 Archival

The Archival phase uses the *Repository Manager* to archive or upload data repository data produced during previous lifecycle phases. This archived data supports later retrieval for rerun. In addition, data can be shared with other OneSAF users or returned to the configuration management and development site for later incorporation into future OneSAF releases.

3.14 Retrieval

The retrieval phase restores data repositories with data from previous archives (either remote or local) to support replay of past exercises or reuse of any work products produced during any of the lifecycle phases.

3.15 Product Line Maintenance

The product line maintenance phase supports the regular upgrade and maintenance of OneSAF. This phase is similar to the product line development phase, with the addition that new software or data may be provided by OneSAF users who have performed field modifications. These new or modified items are reviewed and incorporated into future OneSAF releases according to Configuration Control Board (CCB) procedures described in the (SIP).

4. OOS USERS CATEGORIES

This section describes each of the fourteen user categories and their primary relationship to the lifecycle processes and supporting tools. Related lifecycle phases and tools are italicized within each subsection. Figure 2 shows the fourteen OneSAF user categories.

4.1 Knowledge Acquisition/Knowledge Engineering Subject Matter Expert

The KA/KE SME researches, identifies, and catalogs the real-world sources describing entities (tanks, aircraft, individual combantants, etc.), organizations, and behaviors that are important to Army warfare mission space. This activity occurs within the *Knowledge Acquisition and Knowledge Engineering phase (phase 1)*, of the OOS lifecycle and drives the remainder of the OneSAF modeling processes.

From these real-world descriptions the KA/KE SME synthesizes the descriptions that serve the model implementation process. The synthesis process produces a host of agreed upon knowledge acquisition and knowledge engineering products including: the entity, unit, and equipment lists, the physical knowledge acquisition documents (PKADs), the unit knowledge acquisition document (UKAD), task descriptions (TD), process step descriptions (PSD), behavior process documents (BPD), and behavior data tables. After successful peer review these products are reposed in the *knowledge engineering environment* for consumption by the model developers, software developers, and model validators.

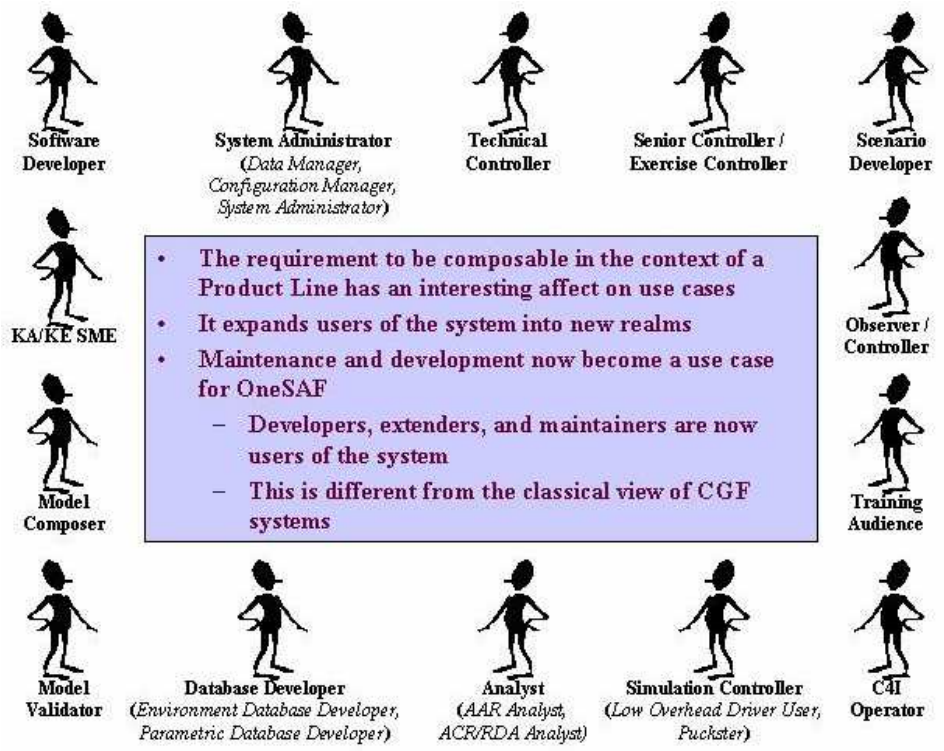

Figure 2: OOS User Categories

4.2 Model Composer

Using the products generated by the KA/KE SME the Model Composer, during the *Model Composition phase (phase 7)*, creates and/or edits entity, unit, and behavioral representations using the *entity composer*, *unit composer*, and *behavior composer* tools. These tools allow the model composer to create completely new representations or to modify existing entities, units, and behaviors without having to modify and recompile the OOS code base.

4.3 System Administrator

The System Administrator is critical to the operation of OOS across *all phases of the lifecycle*. The System Administrator's responsibilities include:

- Data management to ensure the appropriate organization, classification, handling and storage of data.
- Version control to maintain an appropriate baseline across all of the software assets associated with OOS, and
- Configuration management of the entire set of hardware, network, operating system and software components involved in running OOS.

4.4 Database Developer

During the *Database Development phase (phase 5)* The Database Developer uses the military scenario as the primary output of *Event Planning phase (phase 4)* to review and prepare the environment and parametric databases for use. The Database Developer uses the *Data Management Tool* within the *Repository Manager product* to ensure the appropriate environment database in the Objective Terrain Format (OTF) has been created and resides in the *Local Exercise Environment Repository*. The Database Developer also ensures the entire set of parametric and initialization data supporting the intended military scenario resides in the *Parametric and Initialization Repository*, again using the *Data Management tool* within the *Repository Manager product*.

4.5 Analyst

The Analyst participates in a number of OOS lifecycle phases. First, in the *Scenario Generation phase (phase 8)* the Analyst identifies and selects the appropriate data to be collected during execution using the *Data Collection Specification Tool (DCST)* within the *Simulation Generator product*.

Continuing to the *System Test & Verification phase (phase 10)*, the Analyst ensures the scenario proceeds as expected using the *Management and Control Tool (MCT)*. At the completion of the test the Analyst reviews and verifies that the necessary data has been collected to support the analytical, review, and other reporting requirements. If possible the Analyst makes any corrections to the collected data set prior to entering the *Simulation Execution phase (phase 11)*; although if necessary, additional data can be selected during the *Simulation Execution phase*. Also, during this phase the Analyst uses the *Data Management Tool and the*

Simulation Configuration and Asset Management tool to ensure data is collected and that there is enough disk space to hold the data.

Next, within the *Post Execution Analysis/AAR phase (phase 12)* the analyst uses the *After Action Review component* within the *Analysis & Review product* to analyze, summarize, and produce specific after action review products. Products may include Powerpoint presentations, spreadsheet-based analytical charts, or other text-based reporting products.

Finally, the Analyst participates in the *Archival and Retrieval phases (phases 13 & 14)* by identifying the appropriate products and data sets for archival and if necessary may retrieve these data sets at a later date for replay or further analysis. The Analyst uses the *Data Management Tool* to archive and retrieve the specific data sets.

4.6 Software Developer

As with other roles the Software Developer actively participates during several phases of the OOS lifecycle. During the *Product Line Development, Product Line Deployment and Installation, and the Product Line Maintenance phases (phases 2, 3, & 15)* the Software Developer creates the necessary software analysis, design, implementation, and test artifacts consistent with the Product Line Architecture Specification to support user requirements and/or address maintenance issues. To do this the Software Developer's use *the Software Engineering Environment, Configuration Management Tool, the Defect Tool, and the Software Verification Tool within the Maintenance Environment product*. Additionally, during these *Product Line phases* the Software Developer uses the *System Composer* to create the various *OneSAF System Compositions* and *Products* required by the user community.

The same software engineering tools are used by the Software Developer during the *Software Development phase (phase 6)*. During this phase the developer may be creating, enhancing, or maintaining primitive physical or behavioral models or other software components to support a specific scenario or exercise.

4.7 Scenario Developer

Within the *Event Planning phase (phase 4)* the Scenario Developer uses the *Military Scenario Development Environment* to create military scenarios identifying the geographic location where the simulated battle will occur, the type and placement of the friendly, enemy, and neutral forces, an initial situation confronting the forces, and any planned tasking that may be assigned to the forces.

In the *Scenario Generation phase (phase 8)*, the Scenario Developer uses the *MCT* to import and refine the initial conditions within the military scenario. This may include adding specific entity, unit, or behavioral data that was unknown during the earlier *Event Planning phase (phase 4)*.

4.8 Technical Controller

During the *Simulation Configuration phase, the System Test & Verification phase, and the Simulation Execution phase (phases 9, 10, & 11)* the Technical Controller configures, monitors, and controls the hardware and software components involved in the simulation-based event using the *Simulation Configuration Management Tool (SCAMT)*. The *SCAMT* provides a GUI showing hardware, software, and network status. Additionally, the SCAMT can be used to remotely configure OOS components on the hardware platforms executing the OOS infrastructure.

4.9 Simulation Controller

The Simulation Controller, sometimes called the Puckster, controls the simulated entities and units during the *System Test & Verification phase* and the *Simulation Execution phase (phases 10 & 11)*. Entity and unit control are provided through *the Management and Control Tool (MCT)* within the *Simulation Controller product***.** Several important components within the MCT provide the Simulation Controller a wide range of capabilities. Three of the fundamental capabilities offered by the MCT include:

The *Plan View Display (PVD)* allows the Simulation Controller to visualize the simulated environment and the entities operating within that environment. The PVD can be configured to show ground truth or entity-based perception information. Perception-based views show information that an entity or organization has collected using its own sensors or from reports sent by other entities.

The *Mission Editor* allows the Simulation Controller to plan and task entities on the battlespace. This is done by creating mission and tasks and assigning them to specific entities or units within the battlespace. Execution of the tasks is controlled using automated trigger events or on operator command.

The *Task Organization tool* provides a hierarchical representation of the task organization structure and allows the Simulation Controller to task organize the units under his or her control.

4.10 Model Validator

The Model Validator verifies the model representations, data, and algorithms within the implemented models accurately reflect the information specified within the KA/KE products. Typically the Model Validator uses the *Data Specification Tool* during the *Scenario Generation phase (phase 8)* to identify the data elements to collect during simulation execution. As the simulation executes within the *Systems Test and Verification phase (phase 10)* or the *Simulation Execution phase (phase 11)* the Model Validator uses the *MCT* to perform a face validation of the simulation by reviewing the events as they occur on the MCT.

During the *Post-Execution Analysis/AAR phase (phase 12)* the Model Validator uses the *Model Verification tool* to analyze model inputs and outputs to ensure they faithfully represent the information in the KA/KE products. The Model Validator reports problems to the software developers using the *Defect Tool*.

4.11 Observer/Controller

During the *System Test & Verification* and the *Simulation Execution phases (phases 10 & 11)* the Observer/Controller ensures the simulations event's objectives are met by reviewing the progress of the simulation as seen through the *MCT*. Additionally, the Observer/Controller records important user and simulation interactions that are not captured by OOS. Then, during the *Post Execution Analysis/AAR phase (phase 12)* the Observer/Controller integrates the findings into the after action review products and process.

4.12 Training Audience

The Training Audience interacts with the simulation during the *Simulation Execution phase (phase 11)* as they exercise their command and control and staff behaviors in response to the activity within the simulation. The Training Audience activities are input into the simulation through normal staff actions via their real-world C4I systems or through communications with C4I Operators and Simulation Controllers.

The Training Audience also receives the after action review products during the *Post Execution Analysis/AAR phase (phase 12)*.

4.13 C4I Operator

The C4I Operator operates the real-world command and control system during *the Simulation Execution phase (phase 11)* and may be considered part of the Training Audience if the operator is exercising his normal duties as part of the command and control staff. There are times when the C4I Operator is not part of the Training Audience and has a primary role of facilitating the training offered to the training audience.

The C4I Operator also participates in *the Simulation Configuration phase (phase 9)* to define and configure the connectivity between the simulation, the *C4I Adapter product*, and the live C4I devices. After configuration the C4I Operator also participates in the *System Test and Verification phase (phase 10)* to ensure the C4I and simulation interactions work as expected.

4.14 Senior Controller/Exercise Controller

The Senior Controller/Exercise Controller manages and controls the simulation event to ensure exercise objectives are met. To this end, the Senior Controller/Exercise Controller is involved in every step of the OneSAF Lifecycle except for the KA/KE and Product Line Development phases. The Senior Controller relies on a staff and the various OneSAF tools to identify issues and maintain the tempo of the exercise.

5. SUMMARY

This paper starts with a general introduction of the OneSAF Objective System (OOS). Section 2 describes the OneSAF Lifecycle phases in terms of process and important inputs and outputs. Section 3 explains the user roles and relates them to the lifecycle phases and to the tools offered by the OOS.

REFERENCES

1. Wittman, Robert; Harrison, Cynthia. (2001) "OneSAF: A Product Line Approach to Simulation Development", 01E-SIW-061, 2001.

2. OneSAF User Definitions, https://www.onesaf.net/Architecture_and_Integration/SystemsEngineering/TechManuals/WorkinProgress/OneSAF_User_Definitions_r1.doc.

3. OneSAF Lifecycle Phases to Objectives, https://www.onesaf.net/Architecture_and_Integration/SystemsEngineering/TechManuals/WorkinProgress/OneSAF_Phase_to_Tools_r5.xls

4. OneSAF User Tasks to Tools, https://www.onesaf.net/Architecture_and_Integration/SystemsEngineering/TechManuals/WorkinProgress/OneSAF_Tasks_to_Tools_r3.xls.