

# SEAS ISR Architect: Mapping Department of Defense Architecture Views to Military Outcome

**Brian Saulson**

SPARTA

Huntington Beach, CA 92649

*brian.saulson@sparta.com*

The SEAS ISR Architect maps Department of Defense Architecture Framework (DoDAF) views to military outcome and demonstrates a quick turnaround tool for the Initial Capabilities Document (ICD) phase of the Joint Capabilities Integration and Development System (JCIDS) process. The SEAS ISR Architect allows the user to reconfigure intelligence, surveillance, and reconnaissance (ISR) architectures within the DoD Architecture Framework, to parameterize a Systems Effectiveness and Analysis Simulation (SEAS) with a pre-loaded scenario and present a comparison of the warfighting effectiveness of a range of architectural alternatives based on multiple simultaneous theater simulations.

**Keywords:** Agent-based simulation, executable architecture, DoDAF

## 1. Background

Combining platform models into architectures and simulating military outcome have traditionally been ancillary decision support tools for procurement, acting as a complement to subjective decision making. With technology risk the key driver in most pre-transformation DoD acquisitions, procurement officials and contractors had enough of a challenge in simply making devices meet their mission specifications. Therefore, "simulation" consisted mainly of physics-based evaluation tools used to predict individual system performance.

While the COTS/GOTS "system of systems" concept transforms technology risk (ability to build/develop the idea) to schedule risk (ability to get the components), it also reduces the number of engineers and technicians required at each supplier to develop and produce the devices. Thus, it became clear by the end of the 1980s that large scale integration (LSI) should be the principle business model for future defense suppliers.<sup>1</sup>

The 1990s' consolidation<sup>2</sup> of major U.S. defense

1. It could be argued that technology risk is reintroduced when integrators choose to develop software in-house (<http://www.gao.gov/new.items/d04393.pdf>).

2. These consolidations were evaluated by the Government mainly

suppliers [1] resulted in an order of magnitude reduction<sup>3</sup> in the number of contractors in each defense acquisition category.<sup>4</sup> This aggregation of productive resources, combined with today's increasingly "joint" capability expectations, has expanded military procurement's traditional focus on technology risk mitigation to include an additional element of customer satisfaction, or market risk. The consolidation has also been accompanied by an object-oriented software movement that has resulted in general toolsets, like SEAS, that through broad functionality and ability to integrate with domain specific tools have enabled effective warfighting evaluations of entire system domains, such as intelligence, surveillance, and reconnaissance (ISR). Candidate DoD contractors with proven technologies now make offerings based on ensembles, or component groupings, of available technologies. These combinations are valued in terms of architecture "goodness" when proposing a mission capability. These packaged technologies started making inroads into formal operational concept design (see Table 1).

in terms of indirect cost savings and the effect this had on both forward pricing rates and lower contract prices.

3. The reduction effectively changed the Government's procurement decisions from scale, or deep competence in a single area, to scope, or a broad range of capabilities under a single roof. (<http://fic.wharton.upenn.edu/fic/papers/95/9514.pdf>).

4. <http://www.gao.gov/archive/1998/ns98112t.pdf>

**Table 1.** Operational concept definition standards

Name	Description
ANSI/AIAA G-043	<ul style="list-style-type: none"> <li>• Describes               <ul style="list-style-type: none"> <li>- system characteristics</li> <li>- user organization</li> </ul> </li> </ul>
1992 Guide for the Preparation of Operational Concept Documents [2]	<ul style="list-style-type: none"> <li>• Facilitates understanding of the overall system goals</li> <li>• Basis for operations planning</li> </ul>
DI-IPSC-81430	<ul style="list-style-type: none"> <li>• Describes               <ul style="list-style-type: none"> <li>- The user needs that the system will fulfill</li> <li>- The system's relationship to existing systems or procedures</li> <li>- The ways in which the system will be used</li> </ul> </li> </ul>
Operational Concept Description, 12/1994 (MIL-STD-498)	<ul style="list-style-type: none"> <li>• May be "user to developer" or "developer to user"</li> </ul>
IEEE Standard 1362	<ul style="list-style-type: none"> <li>• Describes users' operational needs</li> <li>• Documents system characteristics</li> </ul>
1998 Guide for Information Technology—System Definition—Concept of Operations Document [3]	<ul style="list-style-type: none"> <li>• Results in users               <ul style="list-style-type: none"> <li>- Stating desires, visions, and expectations</li> <li>- Expressing thoughts and concerns on solution strategies</li> </ul> </li> </ul>

In addition to Table 1's operational concept evolution, the focus on component specifications and their associated "stovepipe" systems is losing favor due to the desire to integrate "capabilities" across mission domains. This convergence of schedule, budget, and performance risk(s) into new frameworks is partly due to recent technological advances in enterprise software/systems and their associated data generalization methods.<sup>5</sup>

Prior to publishing the Joint Capabilities Integration and Development System (JCIDS) for DoD procurement, Congress' passage of the Clinger-Cohen Act<sup>6</sup> in 1996 was the largest driver in the recent growth of architecture frameworks for government enterprise system descriptions. This policy level requirement legally requires government contractors to use the Department of Defense Architecture Framework (DoDAF) [4] to document architecture design concepts. Complementing the DoDAF for architecture description are the different areas of JCIDS analysis.

Table 2's JCIDS marks a step forward in unifying system-oriented needs and potential solutions into an overall capability evaluation framework. The JCIDS is predicated on functional capability evaluation, or simulation, to prove out the initial design analyses. Therefore, the JCIDS process is hedging that the transformed military's procurement

5. The Extensible Markup Language (XML), for example, is used to package data depending on user needs, thereby facilitating sharing of information between various efforts and their respective tools of choice.

6. The Clinger-Cohen Act of 1996, Public Law 104-106, section 5125, 110 stat. 684 (1996) is also known as the Information Technology Management Reform Act (ITMRA), which requires information technology investments to provide measurable improvements in mission performance.

**Table 2.** JCIDS products and description [5]

JCIDS Analysis Product	Description
Functional Area Analysis (FAA)	Tasks to be accomplished
Functional Needs Analysis (FNA)	Capability gaps
Functional Solutions Analysis (FSA)	Resolution(s) for capability gaps

risk has migrated from technology development to technology integration. Tools like the SEAS ISR Architect are a first step in automating this evolution.

## 2. Introduction

While the DoDAF's main goal is to promote command and control (C2) interoperability, computation of the resulting architecture's utility remains at times a subjective exercise. Reasons for this include 1) system performance and interaction estimates, a.k.a. model inputs, are bounded by the modeler's understanding; 2) scenarios vary widely in terms of red and blue expected behaviors; and 3) architecture space enumeration (e.g., design of experiments) is usually performed via the modeler's "best guess" as to the scenario/architecture composition and the associated sensitivities involved. An example input-output diagram of this system is shown in Figure 1.

While unstructured inputs, via the modeler's representation of the simulation's scenario and entities, will always be the "art" in modeling and simulation, DoDAF architectures are one technique

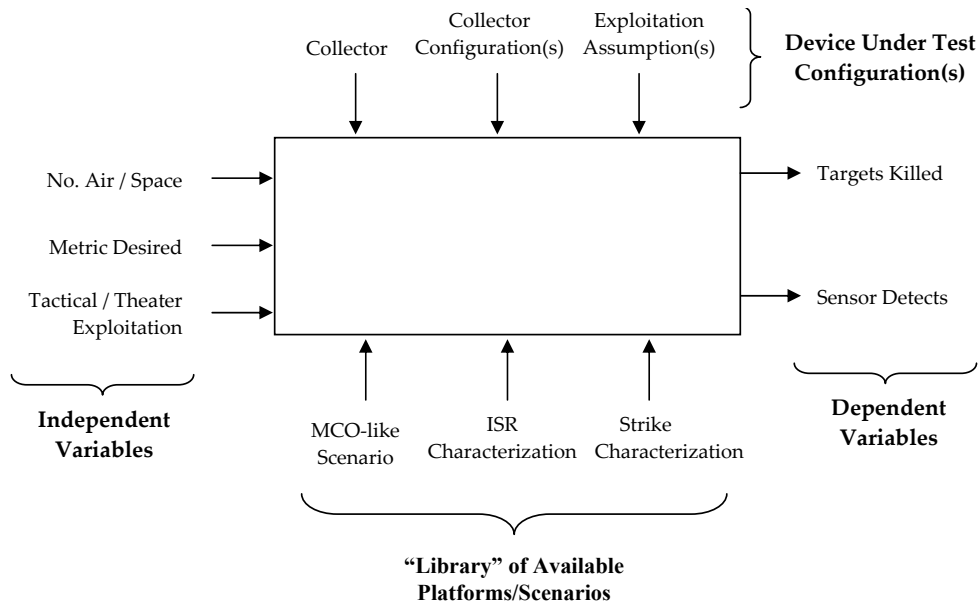


Figure 1. SEAS ISR Architect

for standardizing model initialization data. For example, Zinn [6] found that a DoDAF architecture may be constructed with enough information to populate an agent-based military campaign model.<sup>7</sup> Zinn’s example defines a Popkin SA architecture mapping to an agent-based modeling environment of the Systems Evaluation and Analysis Simulation (SEAS).

Using Popkin SA as the GUI for an architecture evaluation tool results in 1) a finished product on the part of the initial architecture designer, and 2) a data set to initialize SEAS for military utility evaluation based on the scenario and “overhead<sup>8</sup>” assumptions included. Overhead assumptions include the ISR and strike asset infrastructure that is beyond the user’s control, but is assumed into the SEAS scenario that is used for evaluation.

The SEAS tool is broadly designed for evaluating airspace integration over a range of scenario and component assumptions. In addition, the user is free to choose the type of units or metrics that best illustrate CONOPS/platform/sensor impact on the given scenario in SEAS.

The sea-based power projection scenario used here provides the user with control over the available number of space-based radar (SBR) collection nodes and tasking the RATTLSR ship for offshore striking of SBR generated targets. The user does this primarily

through manipulating the OV-1 concept of operations for the blue force. Additional architecture viewing is possible via the OV-5 operational activity node tree, which provides functional force hierarchies and a description of operational and tactical behaviors. This architecture to military utility demonstration provides a JCIDS-compliant Initial Capabilities Development (ICD)-level tool for facilitating the initial phase of materiel or non-materiel procurement.

### 3. SEAS ISR Architect

The SEAS ISR Architect represents a visual basic macro project customization of Telelogic’s Popkin System Architect (SA), which allows the user to reconfigure intelligence, surveillance, and reconnaissance (ISR) architectures within the DoD Architecture Framework. The architecture of the SEAS ISR Architect tool in relation to SEAS, Excel, and System Architect is shown below in Figure 2. System Architect provides a graphically depicted architecture database that may be populated or configured by the user. The purpose of the SEAS ISR Architect is to help guide the user in making these changes and to translate the changes into text file inputs of SEAS Tactical Programming Language (TPL) code. At this point, the interface allows the user to execute the altered simulation and select utility measures to compare against previous runs. The Excel-based SEAS Post Processor is called from the interface and provides the utility measurement feedback which the user may incorporate into analysis and use to guide new architectural configurations.

7. Zinn’s work consisted of populating a SEAS model via the Air Operations Center (AOC) Architecture version 2.0, developed for USAF by MITRE (Hampton, VA).

8. Overhead might be standard communications delays, time to exploit collected information, or simply break times factored in for the operators.

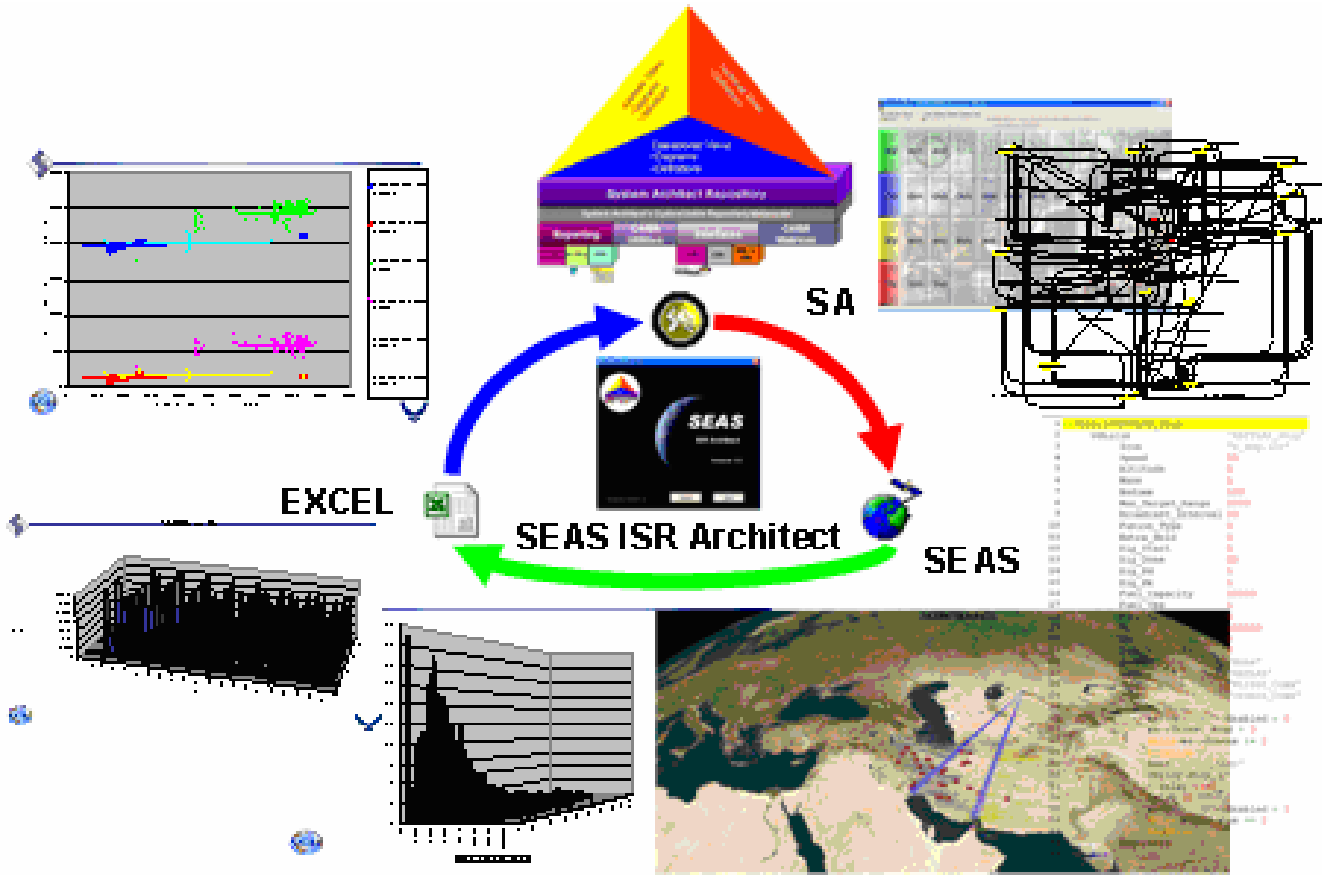


Figure 2. Architecture of the SEAS ISR Architect tool and iterative process

Table 3. Scenario nodes and descriptions

Agent Type	Number	Description
Blue RATTLRS* Ship	1	Strike platform
Blue SAR Satellites	{3, 6, 9, 12}	Only source of targeting data
Blue Comm. Nodes/Channels	2	One for orders and the other one for target sightings
Red TBMs	120	Primary RATTLRS targets
Red Confusers	50	Trucks, buses used to confuse the RATTLRS targeting system
<b>Total</b>	<b>185</b>	

\* Revolutionary Approach To Time-Critical Long Range Strike (RATTLRS) project represents a project that seeks “to demonstrate and increase high-speed flight capabilities and performance for expendable supersonic vehicles.” Additional information is available via GlobalSecurity.org (<http://www.globalsecurity.org/military/systems/munitions/rattlrs.htm>).

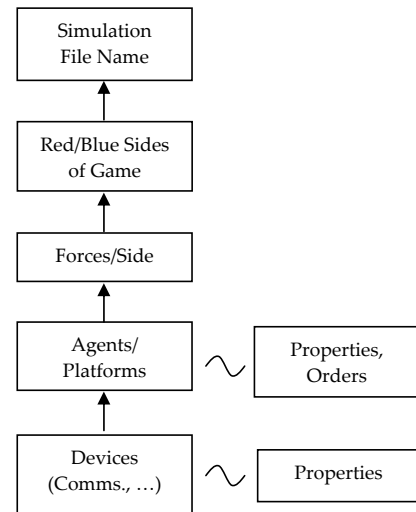


Figure 3. SEAS simulation file structure\*\*

\*\* Popkin SA can also be used to view the existing model structure, which would bring out the hierarchy described here.

The SEAS ISR Architect’s scenario in this case is a 2015-2020, 24-hour response, hyper-velocity missile deterrence of a rogue nation’s mobile tactical ballistic missile (TBM) capability. Exploring the use of an existing asset for limited conflict strike is done with the assets described in Table 3.

Along with Table 3’s entities, communication nodes are included as “devices.” These “devices,” the main technique for coupling the respective entities in a scenario, are at the lowest level of the SEAS abstraction hierarchy (see Figure 3).

As shown in Figure 3, an SEAS simulation is built constructively from its lowest level communications device, through its agents, up to the overall scenario and coupled red/blue model instances. Popkin SA provides a technique for visually inputting any of the elements in Figure 3’s hierarchy. The DoD Architecture Framework and the SEAS Tactical Programming Language both provide relatively general and free environments to communicate system properties and interconnections. In order to

facilitate automated mapping from one language into the other, additional discipline on restricting system representations was required.

### 3.1 Popkin SA

The tool, as presently configured, uses the OV-5 to document the overall scenario. Red and blue force decompositions become obvious with the OV-5, which decomposes the overall warfile (SEAS input file with .war extension) down to the devices used to communicate between the agents.

While the OV-5 is used simply for documentation, the OV-1 is used as the GUI with a range of architecture instance and connectivity capabilities. A Visual Basic macro within Popkin SA is used to create control forms to change the SEAS ISR Architecture or reroute the communications. The OV-1 updates a text file that automatically replaces the Tactical Programming Language (TPL) code of the warfile in initializing the simulation.

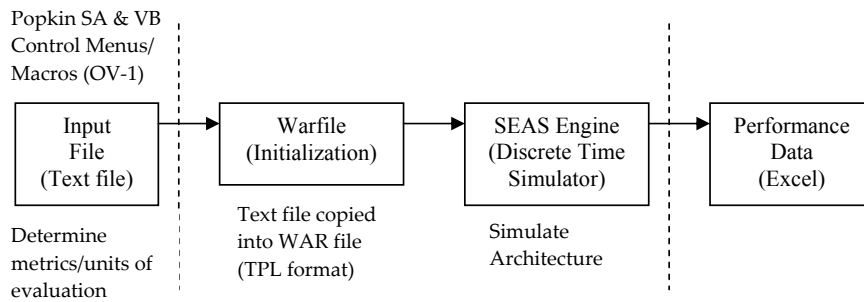


Figure 4. SEAS model construction and execution

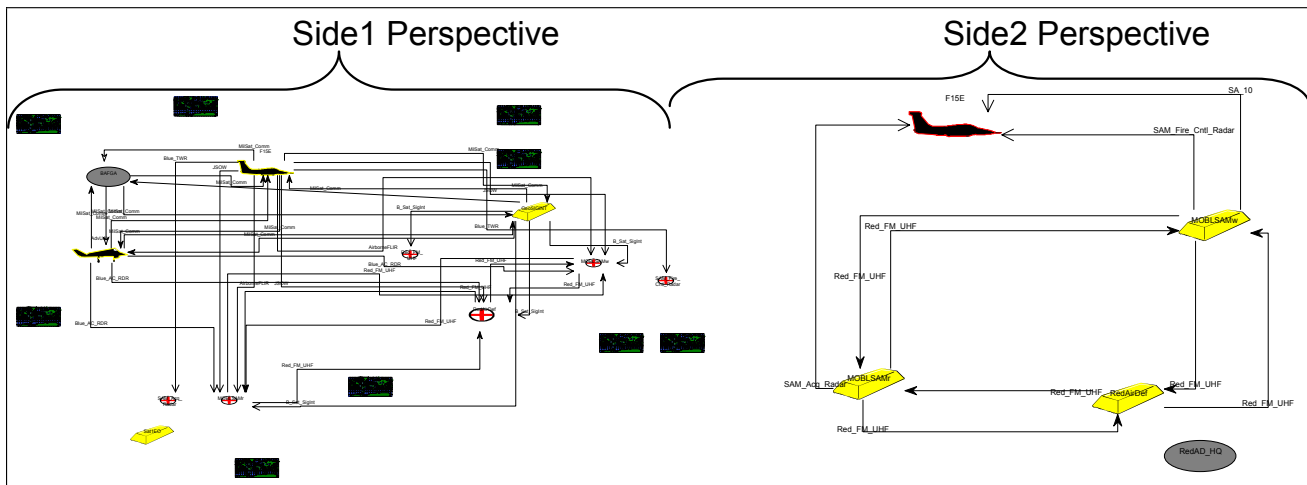


Figure 5. Sample OV-1/concepts of operation perspectives by side\*

\* Figure 5 is intended to demonstrate the necessary separation between the different concepts of operation for each side involved in a conflict, and the names of individual nodes and connections from this specific case are not necessarily intended to be legible.

### 3.1.1 OV-1

The OV-1 and related visual basic forms serve as the graphical user interface for the SEAS ISR Architect. ISR components, connections, the strike component of the scenario, and all of the red assets are included in the OV-1. The OV-1 is translated into a warfile via a text file input, driven by visual basic macros, that overwrites a section of the SEAS warfile prior to simulating the chosen architecture (see Figure 4).

The nature of the OV-1 format within the System Architect DoDAF tends to limit the concept of operations to be defined for one side's perspective at a time as shown with sample OV-1s below.

A description of the System Architect nodes and links used in the above OV-1s are shown in Figure 6.

### 3.1.2 OV-5

OV-5 breaks down a simulation "warfile" into operational activities consisting of both the blue and red "players" in the scenario. It is described hierarchically, with a dependence on the graphical component of the scenario to be shown. A sample simulation OV-5 operational activity node tree is shown in Figure 7.

### 3.2 SEAS

SEAS<sup>9</sup> is a constructive agent-based simulation tool generally applied toward force or system effectiveness and mission-level utility and certified as a key component of the U.S. Air Force's Space Command Modeling and Simulation toolkit. The agent-based methodology applied with SEAS provides the analyst with the tools to quantify strategic effects based on the methodology of observe-orient-decide-act (OODA) loops [7]. This environment allows the analyst to bring out the nonlinear effects of an individual agent's awareness and actions on the overall strategic outcome [8]. SEAS offers the capability to explore system performance trade-offs between various system architectures and concepts of operations. The object-oriented nature of the Tactical Programming Language (TPL) code of SEAS allows systematic and rapid changes to system models within the larger scenario context. The simulated example scenario (regarding the operational nodes described in Table 3) was limited to a single simulated day to provide a less-than-one-minute response on mission utility plotting.

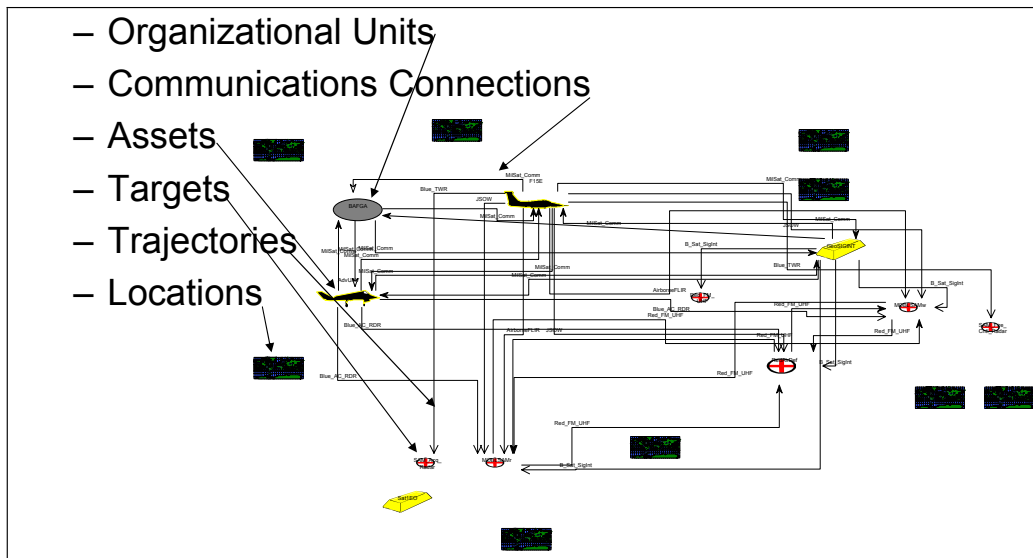


Figure 6. Sample System Architect OV-1 nodes and links\*

9. Additional materials on SEAS, including the "SEAS Overview and Demonstration" briefing presented at the 74th Military Operations Research Society Symposium, are available via <http://www.teamseas.com>.

\* Figure 6 is intended to provide overview perspectives, and the specific names of individual nodes and connections from this specific case are not necessarily intended to be legible.

### 3.3 SEAS ISR Architect: Software System Architecture

SEAS ISR Architect is a software integration between Popkin SA and SEAS. The available architecture views, along with the control panel for choosing architecture alternatives, are stored in Popkin SA (as represented in Figure 8).

Architecture evaluations, the product of SEAS simulations, are stored in the SEAS post-processor and displayed graphically.

### 4. Example Results

The scope of architecture evaluations spans from three to twelve space-based radars (SBRs) with an option of either onboard (RATTLRS Ship) or off-board (in theater distributed common ground system (DCGS)) processing of the downlinked imagery. Figure 9 describes the relative performance of 3-ball SBR architectures with onboard versus off-board image processing using the “killer victim scoreboard” to evaluate the number of red TBM launchers destroyed over a 24-hour period.

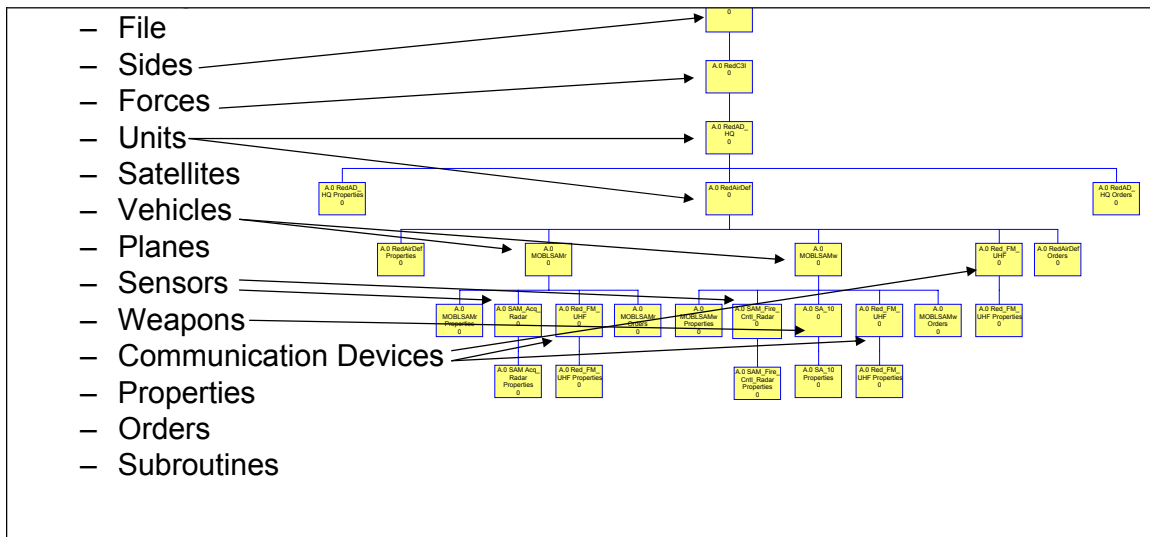


Figure 7. Sample from OV-5 operational activity node tree (operational activities represented within a hierarchy encompassing various agent objects in the simulation)\*

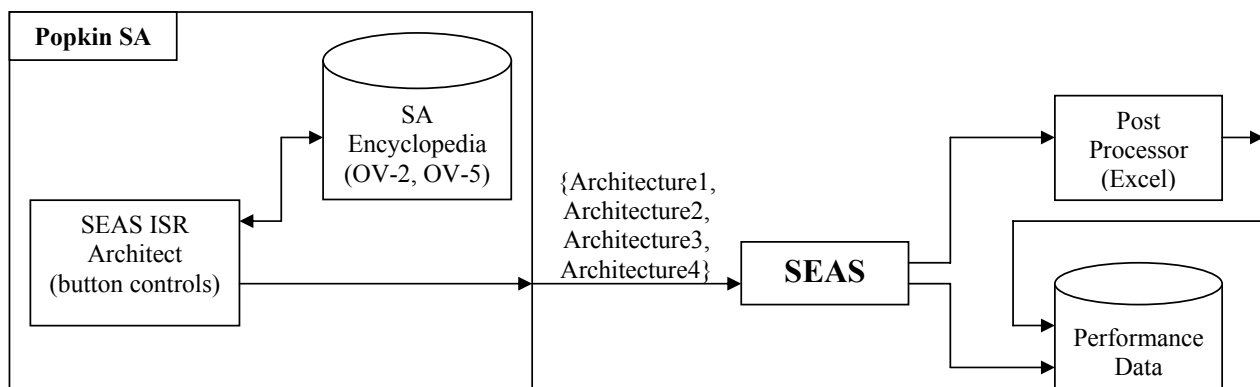


Figure 8. SEAS ISR Architect—Software system architecture

\* Figure 7 is intended to provide overview perspectives, and the specific names of individual nodes and connections from this specific case are not necessarily intended to be legible.

In addition to Figure 9's configuration evaluation, we can also look at the net KVS gain due to additional sensors (see Figure 10).

Figure 10 describes the net increase in the number of kills due to additional SBRs. It is notable that the orbitology results in the greatest number of collects for the 9-ball system—and in this case the greatest number of kills. The 12-ball architecture has saturated the tactical processing station and is just catching up to the 9-ball system at the end of the 24-hour evaluation period. It should be noted that the above results represent only a single run of what would typically require dozens or more runs for a more statistically significant comparison.

### 5. Conclusion

Mapping DoDAF views to military outcome through executable architectures and software integration is not only feasible but it may become necessary. To

some extent, automation of this mapping requires that the instantiation of the models within the DoD Architecture Framework and simulation tools such as SEAS be restricted to formats that limit the possibilities for miscommunication. The SEAS ISR Architect is a step in the right direction as far as proving the concept and providing insight into a possible approach. The architecture, simulation, and utility output tools are individually validated and verified, but testing of the integrated software suite requires validation and verification as a whole. Scoping the SBR architecture to a manageable MCO-like scenario resulted in the SEAS ISR Architect providing immediate insight into the utility of offshore negation of a red TBM threat. Of course, blue/red CONOPS along with system performance assumptions are key elements in any scenario evaluation. Each of these is documented in the SEAS ISR Architect through either the Popkin SA or in the SEAS simulation itself. In addition to increasing customer interaction, integrating related

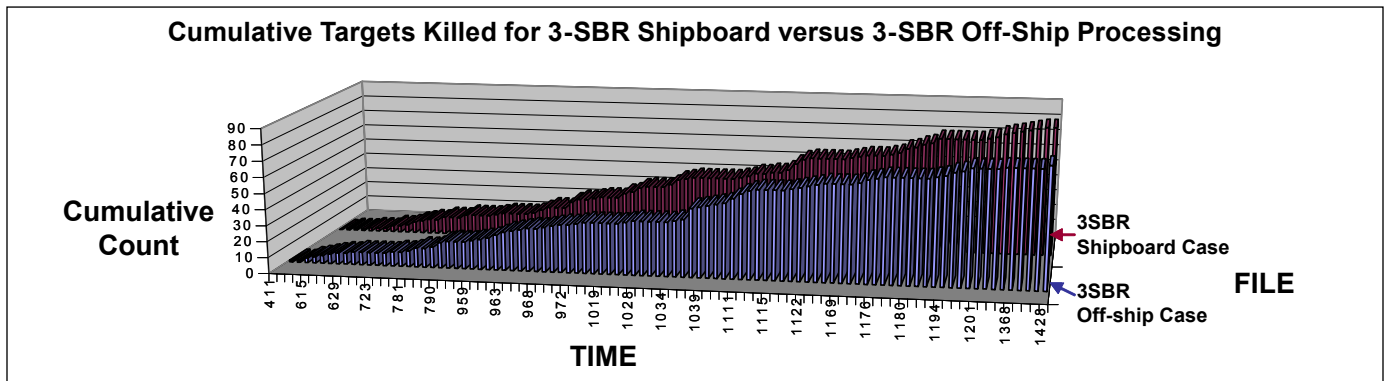


Figure 9. Time versus KVS over 24-hour evaluation (in minutes)\*

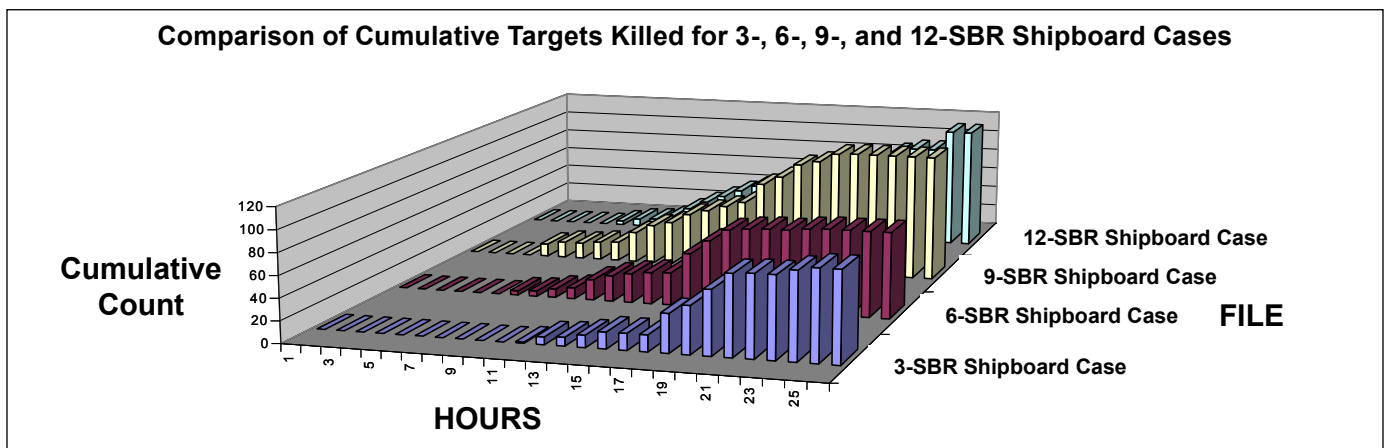


Figure 10. Time versus KVS over 24-hour evaluation (in minutes)\*

\* Figures 9 and 10 are intended to demonstrate types of output evaluation available, and the results from this specific comparison are not necessarily intended to be legible.

systems with respect to scenarios is an approach that should synergistically help unite efforts in multiple project areas. While the SEAS ISR Architect relies on the Telelogic (Popkin) System Architect tool, the eventual development of a more general architecture-simulation interface is a goal to be investigated.

Other modifications to the tool might be through modular growth. For example, users might be interested in specific “slices” through their design, such as life cycle cost. Another application may be to use the tool as is and simply build it out for each evaluation. Whichever path the tool may take, ultimately it brings the promise of additional and/or quicker spirals of development and analysis by more efficiently bridging the gap between the architecture and analysis in a documented and repeatable approach.

## 6. References

- [1] GAO. *Defense Contractor Restructuring: Benefits to DoD and Contractors*. Washington, D.C., 1998.
- [2] ANSI. *Guide for the Preparation of Operational Concept Documents (ANSI/AIAA)*. Washington, D.C.: American National Standards Institute, 1992.
- [3] IEEE. *IEEE Guide for Information Technology – System Definition – Concept of Operations Document*. New York: IEEE, 1998.
- [4] DoD Architecture Framework Working Group. *DoD Architecture Framework Version 1.0*. Washington, D.C.: Department of Defense, 2-1, 2004.
- [5] JCS. *CJCSI 3170.01E*. Washington, D.C.: Joint Chiefs of Staff, 2005.
- [6] Zinn, A. W. “The Use of Integrated Architectures to Support Agent Based Simulation: An Initial Investigation.” Master’s thesis, Aeronautics and Astronautics, Graduate School of Engineering and Management, Wright-Patterson Air Force Base, OH, Air Force Institute of Technology, 158, 2004.
- [7] Bullock, Richard K., et al. “Using Agent Based Modeling to Capture Airpower Strategic Effects.” In *Proceedings of the 2000 Winter Simulation Conference*, 2000.
- [8] Tighe, Thomas R. “Strategic Effects of Airpower and Complex Adaptive Agents: An Initial Investigation.” Master’s thesis, Air Force Institute of Technology, Graduate School of Engineering and Management, 1999.

## Author Biography

**Brian Saulson** graduated with an M.S. in systems engineering from George Mason University and is an Engineer with Sparta, Inc.’s Space & Missile Defense Operation, within the Space Systems Division. The focus of his recent work includes post processing and analysis of simulation outputs, agent-based modeling and simulation, and development of executable DoDAF architectures.