

OneSAF: Experiences from the Field in an Open Source Environment

Doug Parsons

U.S. Army, PEO Simulation, Training and Instrumentation

Orlando, FL

doug.parsons@us.army.mil

ABSTRACT

The U.S. Army recently released the One Semi-Automated Forces (OneSAF) simulation system as a single entity-level software solution to serve three modeling and simulation domains. Postured as an open-architecture, open-source application, the OneSAF program will put this software into the hands of a vast number of developers throughout the Department of Defense. Software development of the OneSAF application has been conducted in a robust systems engineering environment that balances agile methods and traditional practices. Now fielded, OneSAF will expand this development environment to include processes and methods to enhance opportunities to integrate code from external developers with varying development capabilities and objectives. These processes will create unprecedented participation, and leverage the talents and resources of developers throughout the modeling and simulation community to include multi-service, international, industry and academia experts. This paper will describe the methods, as well as early experiences and lessons learned while integrating and managing multiple baselines from multiple sources, as well as the subsequent return on investment.

ABOUT THE AUTHOR

Doug Parsons is the Chief Engineer for software development of the One Semi-Automated Forces (OneSAF) program at the U.S. Army Program Executive Office for Simulation, Training, and Instrumentation (PEO STRI). He has over twelve years of experience of virtual and constructive simulation development. Mr. Parsons received a B.S. in Mechanical Engineering from North Dakota State University, a M.S. in Systems Management (Operations Research) from Florida Institute of Technology, and a M.S. in Industrial Engineering (Interactive Simulation and Training) from the University of Central Florida.

OneSAF: Experiences from the Field in an Open Source Environment

Doug Parsons

U.S. Army, PEO Simulation, Training and Instrumentation

Orlando, FL

doug.parsons@us.army.mil

INTRODUCTION

When many software developers are asked about open source they focus upon the impact that the Linux operating system has had upon the software community. Without hesitation we should give Linus Torvalds his due. His decision to freely give Linux source code to a handful of other programmers, some called hackers, created a paradigm shift for software development. What was once a hacker's toy in 1991 has now become an object of affection for hundreds of thousands of users and developers. However, during the early days much of computer software's source code was open to viewing and changes. Proprietary software didn't emerge until 1976 when Bill Gates posted a famous letter on the Homebrew Computing Club demanding that people stop sharing his Altair BASIC [Archibald, 2007]. Mr. Gates must be given his credit as well. Proprietary software has dominated the software development landscape for many years to include Department of Defense (DoD). It wasn't until May of 2003 that the DoD released a policy statement that put open source software (OSS) on equal footing with proprietary software [Stenbit, 2003]. Still, the path to wide acceptance across DoD has been somewhat elusive. As stated in the 2006 "Open Technology Development Roadmap Plan", prepared for Ms. Sue Payton, Deputy Under Secretary of Defense, Advanced Systems and Concepts [Lucas & Scott, 2006]:

"The business model of purchasing physical goods and services has served DoD well in the past; but it falls short when applied to software acquisition. By treating DoD-developed software code as a physical good, DoD is limiting and restricting the ability of the market to compete for the provision of new and innovative solutions and capabilities. By enabling industry to leverage an open code development model, DoD would provide the market incentives to increase the agility and competitiveness of the industrial base."

The report further reads:

"DoD needs to evaluate the impact that locking into one set of proprietary standards or products may have to its ability to react and respond to adversaries and more importantly, to technological change that is accelerating regardless of military conflict. In order to remain competitive in a rapidly shifting technological landscape (including the disruptive technologies leveraged by our adversaries), DoD's software development and business processes must break out of the industrial-era acquisitions mold."

Per this assessment the DoD is clearly primed for another paradigm shift in the software acquisition world.

BACKGROUND ON THE ONESAF PROGRAM

This paper is not intended to discuss the technical capabilities of the OneSAF system; however, it is very important that the reader understands some of the program's history to understand the significance for open source development.

The Army's Mission to Save Resources

The OneSAF program concept originated in January 1996 following an extensive study which came to the conclusion that the Army was caught in a wasteful spending cycle, making identical or similar enhancements to many legacy simulations across three different user domains. In May of 1997 the Deputy Commanding General, Training and Doctrine Command (TRADOC) approved the Mission Needs Statement (MNS) for OneSAF which stated:

"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 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 and associated simulation event support tools.

Serving the Army and more

The OneSAF system is the U.S. Army's next generation simulation system that can represent a full range of military operations, systems, and control processes. It is an entity-level simulation, meaning that it can simulate the activities of individual or groups of individual combatants or weapon platforms. It will also provide the appropriate representations of the physical environment (e.g., terrain features, weather, and illumination) and its effect on simulated activities and behaviors.

One aspect that makes the OneSAF program unique among Army simulations is that it has been designed for use by three distinct Army Modeling & Simulation (M&S) domains. The Advanced Concepts and Requirements (ACR) domain uses M&S for experimentation and analyses on Army doctrine and force related concepts. The Research, Development, and Acquisition (RDA) domain uses M&S for acquisition analyses focused at equipping and supporting currently fielded and future forces. Finally, the Training, Exercises, and Military Operations (TEMO) domain employs M&S to train and prepare the force. It does so using live simulation (actual equipment on training ranges), virtual simulation (immersing the trainee into a synthetic environment), and constructive simulation (wargames using computer generated forces).

The OneSAF system will not only be used by the Army; the baseline source code will be distributed to multi-service, joint, international, industry and academic organizations by request. Releasing source code to such a vast network of developers will certainly reap benefits for the Department of Defense M&S community as a whole; however, distributing source code alone will not provide the optimal mechanism for a community to work together.

Open Source and OneSAF

The release of source code as part of Department of Defense (DoD) applications raises numerous questions to include security concerns, baseline configuration management, cooperative development, and integration. In particular, security creates a circumstance preventing OneSAF from being truly open source. The OneSAF project was intended to be open source within the constraints of the U.S. Government [Parsons & Wittman, 2005]. While it is

true that the source code is distributed free-of-charge¹, PM OneSAF can not allow the code to be distributed freely. According to the Open Source Initiative (OSI), OSS provides the following rights and obligations [Coar, 2006]:

- Free redistribution.
- Availability of the source code.
- Right to create modifications and derivative works.
- May require modified versions to be distributed as the original version plus patches.
- No discrimination against persons or groups.
- No discrimination against fields of endeavor.
- All rights granted must flow through to/with redistributed versions.
- The license applies to the program as a whole and each of its components.
- The license must not restrict other software, thus permitting the distribution of open source and closed source software together.

While OneSAF is not distributed as a classified system, the baseline does have content (e.g. data, algorithms) deemed sensitive by the U.S. Department of Army. The baseline can not be freely distributed as defined for open source due to security reasons. As such, PM OneSAF must be selective in the distribution of the baseline to DoD/government and organizations that support them. Essentially, there are two basic commitments made when a user signs a OneSAF distribution agreement.

- (1) Authorization to redistribute the baseline outside of the intended project is restricted to PM OneSAF.
- (2) Users who develop new functionality into the OneSAF baseline agree to provide those capabilities back to PM OneSAF for possible reintegration.

These constraints offer advantages across the OneSAF user community. Facilitating distribution through a single focal point allows the Program Manager to have knowledge of who the users are and how they intend to use the baseline. This knowledge will enhance the ability to identify and integrate useful community-developed capabilities into future baseline releases.

LEARNING FROM A LEGACY SIMULATION

The OneSAF program has drawn many lessons from the now retired Modular Semi-Automated Forces

¹ There is a fee for Foreign Military Sales (FMS).

(ModSAF) program. Funded by the Defense Advanced Research Projects Agency (DARPA) in the early 1990's, ModSAF was developed to facilitate synthetic environment research in support of Distributed Interactive Simulation (DIS) applications. Word of the availability of source code quickly spread through the M&S community and requests for the software steadily grew to the point of its retirement in 2002. If not for the decision to release the source code, ModSAF would likely have been a short-lived, albeit interesting, software project. At retirement over 200 organizations placed requests for the source code. The OneSAF Program Office harvested several lessons learned from the ModSAF program, those worth continuing, as well as those that needed improvement.

There were essentially four lessons learned from ModSAF applied to the OneSAF program. Some were learned through positive experience, but many were the result of wishing to avoid a second sting from the error portion of 'trial and error.'

Lesson 1. Publish standards and establish expectations

Providing source code and inviting developers to submit back their capabilities can be noble and well-intentioned. But, as the saying goes, 'be careful what you ask for.' The OneSAF system was developed under the Software Engineering Institute (SEI) Capability Maturity Model Integration (CMMI) level 5 processes. As such, the code is well structured and documented. Sometimes, looking at code submitted by an external developer the first time was frustrating. If OneSAF was to be a tool used and continually developed by a community, standards, styles and expectations needed to be developed and communicated. However, imposing a CMMI level requirement or expectation could be counterproductive. To twist a phrase from the motion picture *Field of Dreams*, "if we build it, they still may not come." If the handover and reintegration process is too painful for the submitter, they may simply stop providing code. Communication is key. The OneSAF developer's website provides an Electronic Process Guide (EPG) that outlines a structure, work-flow oriented reference for a set of processes and procedures. Most importantly, the website also provides OneSAF standards and policies which will optimize a smooth integration into the core baseline. These standards and policies pertain to design, coding, code tree, build structure, testing standards and others (i.e., error handling, threading, naming convention). Also provided are checklists to include installation, peer review and tools. While developers on the OneSAF

base program must follow these processes, we make them available to external developers as a means to understand our methods and use them at their discretion.

Handover Process

OneSAF has also developed a *handover process* by which the externally developed code can be reviewed. By publishing this process the external developer understands those areas deemed critical to maintaining architectural integrity and software engineering quality. For example, proper documentation is essential. ModSAF had many great capabilities, of which little supporting documentation existed prior to version 3. We would perform maintenance on what was believed to be dead code, and then use hope as a method regarding what other parts of the system secretly relied on that code. The handover process involves the submitter first completing a self assessment in the form of a spreadsheet that helps determine compliance relative to infrastructure, user interface, platform, process, verification and deployment. Completing this form simply provides the OneSAF Architecture & Integration (A&I) team with a road map for conducting a detailed assessment of the code and documentation.

Dependency meetings

The OneSAF A&I team has a formidable task before it. They not only need to plan for the integration and test of capabilities for the ongoing OneSAF base program development, but need to consider unknown numbers of handovers from other organizations. In order to coordinate, control and appropriately resource the process, the user community is invited to participate in a *dependency meeting*. These meetings are scheduled two weeks before the start of the next software build cycle. The purpose is to determine what interaction the A&I team needs to schedule during the build. Understandings and expectations are established regarding code delivery dates, assessments, and possible integration & test support.

Lesson 2. Create a Collaborative Environment

We firmly believe that the OneSAF's composable architecture is a leap ahead in computer generated forces (CGF) technology. However, we also believe that the next leap ahead will come from the M&S community. Fostering interaction among those organizations may be the catalyst to create an epiphany leading to the next great capability that everyone talks about. The ModSAF program benefited greatly through the use of a reflector that allowed developers to communicate ideas and share solutions for problems. The growth of the internet tools and related

technologies provide enablers beyond those available during ModSAF's existence. PM OneSAF is considering the ability for developers to upload/download unofficial modifications, either new capabilities or bug fixes, from the OneSAF.net website. In addition, web forums and community peer reviews may be established for the exchange of information and ideas. However, we are proceeding cautiously to assure that no security protocols are being violated. Most DoD software acquisition websites only need to be security conscious from a user perspective. PM OneSAF needs to consider the sensitivity of providing vastly more information to support a developer community.

The OneSAF program also supports regular user group conferences. Initially, the briefings were completely conducted by PM OneSAF, TRADOC Project Officer (TPO) OneSAF, and associated contractor personnel. The intent is that external developers will take an increasingly greater role in the presentations and discussions at these conferences.

Lesson 3. Establish processes that support agility

OSS development and integration emphasizes a spiral development approach. The OneSAF spiral development process was based in part on the following principles from the Agile Manifesto [Highsmith & et al., 2001]:

- The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

A OneSAF build is approximately 10 weeks in length with full version releases occurring annually. Interim releases are also released over the course of the year. The purpose of these releases is two-fold. First, users receive the latest capabilities and bug fixes. Secondly, more frequent releases keep the baselines between PM OneSAF and an external developer from becoming too widely disparate and make for a smoother integration events [Parsons & Surdu, 2006].

Lesson 4. Apply the appropriate method for baseline governance

OSS programs employ a variety of structures for controlling the content of the software baseline. Steven

Weber [2004] (*The Success of Open Source*) describes governance through either Benevolent Dictator(s) or Exclusive Group. Linux is an example of the benevolent dictatorship as Linus Torvalds makes the final decision with respect to the direction of the software code. The Apache webserver group is an example of an exclusive group. A group of about a dozen people take suggestions, but are the only ones who make and release changes in code. The OneSAF model is actually a hybrid between the two, but more closely related to the benevolent dictatorship. Configuration control involves both PM OneSAF and TPO OneSAF. If there is disagreement between the two offices, then representatives from the three Army M&S domains break the tie. Together this group forms the OneSAF Configuration Control Board (CCB). These organizations have technical representatives that work with the external developers and make recommendations to the CCB. They are the Engineering Configuration Control Board (ECCB).

One of the purposes of the CCB is to provide a formal structure to assure that versions to be integrated are compatible. In keeping within the spirit of providing a single "One" SAF solution, the OneSAF program manager does not want the baseline to *fork*. The Army has seen baselines become divergent in an attempt to satisfy both the analytical and training communities. When this happens, the mission to invest resources a single time becomes doubled. When an organization's capabilities aren't integrated or aren't provided back to PM OneSAF for integration, there exists the chance for the baseline to fork. This can occur when developers try to create alternative means for their code to play a more significant role than achieved in the base product. Linux provides a positive example of fork control [Kenwood, 2001]. Ninety-nine percent of Linux distributed code is the same. This success can be attributed to its accepted leadership structure, open membership and long-term contribution potential.

CHALLENGES

There are no doubts regarding whether or not the OneSAF system is technically challenging – it definitely is. To satisfy such a wide range of users with an even more extensive list of use cases requires a complex architecture. However, the challenges regarding acceptance of the baseline as an open source solution is not technical. The resistance to transition away from a legacy application or method is partially programmatic and very much cultural. Self preservation is very strong in the DoD software

acquisition world. PM OneSAF has endured criticisms that found their way to the desks of some of the Army's highest ranking officials. To date PM OneSAF has successfully defended the program's business model. The primary concerns have been in regards to security and configuration management.

Security

In 2003 Terry Bollinger, Mitre, prepared a report for the Defense Information Systems Agency (DISA) regarding the use of Free and Open-Source Software (FOSS) in the DoD. The main conclusion of the analysis was the FOSS software played a more critical role in the DoD than has generally been recognized. One unexpected result was the degree to which DoD security depends upon FOSS. While the DoD has become more accepting of the use of open source solutions as applied to security, information assurance is critical. With the 2003 policy memorandum provided by John Stenbit, DoD Chief Information Officer, which essentially leveled the playing field between OSS and proprietary systems, came the same obligations to include security considerations. Some people have expressed concerns that releasing OneSAF source code can represent a risk of something sensitive, yet unclassified, falling into the wrong hands. There are additional concerns regarding malicious code finding its way into the baseline. These both are valid concerns and should be considered for all DoD software acquisition programs.

Controlling access

The OneSAF program must use a *trusted environment* to reasonably distribute source code within current resources. By trusted environment we mean that we have full faith that the operational security and information assurance departments at each receiving organization abide by the conditions in the OneSAF distribution agreement. Specifically that U.S. Government (USG) and USG contractors will handle and maintain OneSAF software in accordance with AR 380-5 (Department of Army Information Security Program) requirements as applied to For Official Use Only (FOUO) and export controlled software.

Controlling malicious intent

Proponents of OSS will argue that open source development provides better security cover since it puts many more eyes on the code than a proprietary software program. Likely one method is not inherently better or worse than the other. Both methods need to

put safeguards in place to minimize security risks. The OneSAF program uses an integrated development environment (IDE) to develop the internal baseline capabilities. All developers either have security clearances to the secret level or have those clearances pending. The IDE is protected from external electronic attacks by dual firewalls. The IDE also includes protected logins to control access to only those with a need for that level of access. Code peer reviews are conducted primarily from a software quality perspective, but also guard against a threat from a malicious developer. The configuration management process tracks history of developers' integration activities. Finally, the OneSAF system is required to be compliant with DoDI 5200.40 (DoD Information Assurance Technology Security Certification and Accreditation Process (DITSCAP)), dated 30 December 1997. In accordance with these processes the OneSAF program has received an Authority to Operate (ATO).

Configuration Management

Controlling and coordinating integration activities and configuration management in an OSS environment is certainly challenging. However, the challenge is not primarily technical. Our experience is that it is more a question of appropriate resources. Figure 1 shows the various co-development activities impacting the OneSAF baseline. Without going into the various program acronyms, this figure illustrates how development from the Objective Force/Future Combat Systems (OF/FCS), international FMS cases, external developers, as well as ongoing OneSAF base program integrates into baseline version releases. The figure rightfully looks complex, but the process is technically controllable. Keeping in mind that OneSAF version 1.0 was only released in October 2006, the number of programs that may provide code back to PM OneSAF could become overwhelming. The programs in the figure are only those that PM OneSAF currently has direct programmatic relations. To date there have been more than 200 signed distribution agreements requesting the OneSAF software. The number of those organizations that are actively developing new capabilities and planning to submit code back to PM OneSAF is yet undetermined as of the date of this paper. However, the possible numbers could be programmatically overpowering and force the OneSAF CCB to make some hard decisions based on available resources; specifically which capabilities can be integrated and which must wait. As stated earlier, there is a heavy desire to prevent the baseline from forking, but we all must live within our resources.

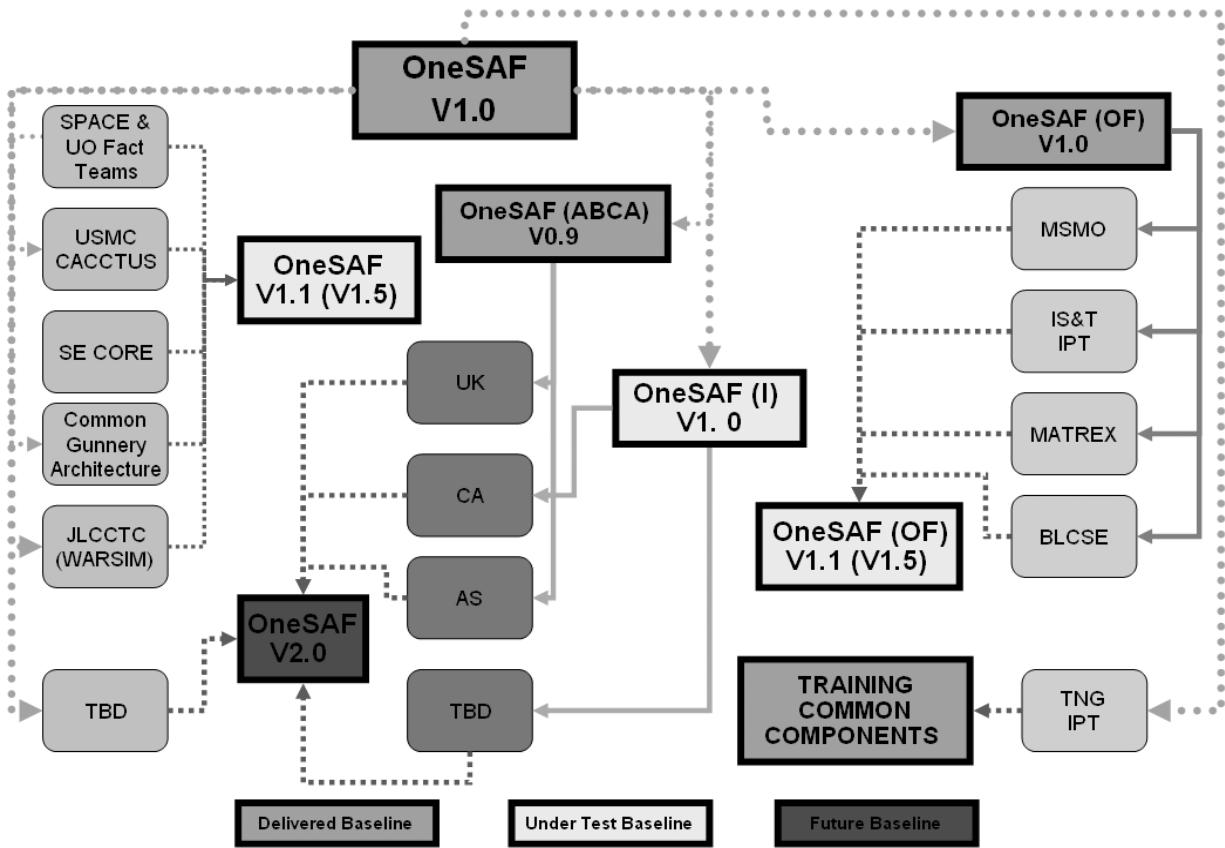


Figure 1. Current OneSAF Co-Developer Baseline Relationships

RETURN ON INVESTMENT (ROI)

The open source paradigm is an enabler for reuse. Organizations that foster this type of environment can potentially realize more rapid innovation and progress on their programs. Plenty of opportunities exist to leverage the tremendous talents and resources both within and among organizations to receive a favorable return on integration and test investment toward a return of capabilities for an entire community's use.

Practice what you preach

While PM OneSAF has embraced the reuse philosophy for many years, the Program Executive Office for Simulation, Training and Instrumentation (PEO STRI) as an organization has become an enthusiastic supporter of reuse among our programs. Dr. James Blake, the PEO, signed a policy memorandum on the use of common standards, products, architectures and/or repositories [2006]. In the memo Dr. Blake states, "the current Army and Joint operational environment requires that we be more adaptable in the training and test systems we provide to the Warfighter." He further states, "this era of interoperability, product lines,

common interfaces, enterprise architectures, etc. provides us an opportunity to leverage capabilities across domains and to facilitate cross domain collaboration and integration." Within PEO STRI, the OneSAF system will be reused in a variety of ways. It will become the integrated SAF within virtual simulations, such as the Close Combat Tactical Trainer (CCTT) and the Aviation Combined Arms Tactical Trainer (AVCATT). OneSAF components, i.e. Environmental Runtime Component (ERC) and C4I adapter, are being reused and further developed in other programs. The Common Gunnery Architecture (CGA) program created unique capabilities with the OneSAF source code. Much of this code has been reintegrated, and PM OneSAF is already extending some of those capabilities. It should be noted that OneSAF is not the sole benefactor of this policy, as reuse crosses programmatic boundaries and is an organizational commitment.

Our Co-Developers

Despite the fact that the first version of OneSAF has been on the streets only since October 2006, there is already a wealth of external organizations collaborating with PM OneSAF. David Tuma [2005] stated in an

article on OneSAF open source software opportunities and challenges, “The open source paradigm is based on the idea that software reuse need not stop at organizational boundaries.” One of the strengths of OSS is that support will last as long as there is demand. Use of the OneSAF baseline by other programs is still at initial stages. However, the Army could elect to cease funding the OneSAF base program and the OneSAF baseline could continue indefinitely, based upon its usefulness to those other programs. It is likely that the value to the M&S community could diminish, and the baseline could splinter into many other baselines, without a central program to guide configuration control and conduct integration. But OneSAF could survive through the co-developers. The following is a brief discussion of our current co-development activities from organizations external to PEO STRI.

U.S. Marine Corps, PM Training Systems (PM TRASYS)

The Marines have the distinction of fielding the OneSAF system for active training applications before the Army. PM TRASYS incorporated OneSAF into their Combined Arms Command & Control Trainer Upgrade System (CACCTUS). PM TRASYS initiated a programmatic relationship with PM OneSAF a full year before version 1.0 was released. They assumed a risk which consequently paid off. A few of the capabilities being handed back to PM OneSAF for re-integration include:

Entity Compositions -

- CH-53E Sea Stallion Cargo Helicopter
- EA-6B Prowler Fixed Wing Aircraft
- USMC Scout/Sniper team

Models -

- 5” / 54 Caliber MK 45 Naval Artillery Gun
- Clamped Pitch Turret

Behaviors -

- Hasty Defense (Automated Halt)
- Maintain Contact Reconnaissance

Behavior compositions –

- Fire Direction Center (FDC)
 - Maximum Ordnance Information Sent to the Forward Observer
 - Support for Variable Rate of Fire
 - Extending Fire Missions
 - Suppression of Indirect Fire by Indirect Fire
- Nine-Line Close Air Support
 - Cleared Hot Response

- Abort Mission
- Multiple Bomb Drop
- Bomb Delivery Profiles

- Target Unit Ownership

Space and Missile Defense Command (SMDC)

The SMDC also engaged with PM OneSAF prior to version 1.0 release. They established an integrated development environment which closely resembled that of our program. They also worked closely with our engineers to shape their code handover packages to assure smooth re-integration into the OneSAF baseline. Some of the capabilities provided to PM OneSAF include:

Unit -

- BLUFOR Space Unit (Ground station and satellite)

Behaviors -

- Conduct Satellite Surveillance
- ISR Space Fly Route

Physical models -

- Forecast Satellite Image Collection Times
- Low Earth Orbit (LEO) Satellite Downlink Delay
- Satellite Mobility
- LEO Satellite Mobility Waypoint Aim point Calculation

Naval Representations

The effort to create Navy representations in the OneSAF baseline is somewhat unique in that the work was actually a customer-funded effort to the OneSAF program office. PM OneSAF developed, tested, and integrated the following capabilities:

- Land-Sea Fire Support (Call For Fire)
- Vessel IED emplacement
- Naval Anti-Air Engagement
- Naval Direct Fire Sea Engagement
- Weapon Systems; e.g. MK-45, 5” Gun, Tomahawk, MK-15 Phalanx

Future Combat System (FCS)

The U.S. Army’s FCS program is a bold move forward in regards to the use of simulation for analysis and embedded training. This program selected OneSAF as its onboard SAF training solution. The following candidates are a partial list of capabilities developed through FCS resources:

- Crowd behaviors
- Sniper behaviors
- Urban insurgent modeling

- FCS platform and behavior models
- C2 architecture
- Communication Effects Server (CES) integration
- MATREX 2.0 FOM support
- 1516 RTI support
- Virtual simulation support
 - Ownship framework
 - IC mobility/posture control
 - UAV control

Return On Investment

Integrating capabilities from a disperse modeling and simulation community provides the ability for more robust exercises and experiments. This becomes especially true as more emphasis becomes placed on joint, multi-service and international involvement. Yet, we must return to the original mission to understand the value. The Army initiated OneSAF in order to save resources by allowing a variety of legacy simulations to be retired. The return on investment with an open source approach adds to those savings by trading an integration, test and maintenance investment to gain the investment made by multiple other organization. To date PM OneSAF has invested roughly \$350k to integrate and test capabilities from the CACCTUS, SE Core, CGA, SMDC and Navy programs². In return these programs have invested approximately \$5 million to develop capabilities, which are now available to the modeling and simulation community free-of-charge.

CONCLUSIONS

From this paper the reader may believe that we are preaching open source as the sole solution. To the contrary, we are simply relating what has worked well for the OneSAF program. All program managers should weigh the benefits and risks to find a solution best tailored to the success of their projects. Software development is like a smorgasbord dinner; use what processes and tools look palatable for your program and leave the rest on the table. In fact not all of the software tools used in the OneSAF development environment are free and open source. Our philosophy toward open source was and remains to first look for free and open source software solutions. However, we have no concerns with using proprietary solutions, if they best support successful developmental processes. We do encourage organizations to establish a culture

² Integration of the FCS capabilities was not complete at the writing of this conference paper.

where open source development and acquisition can be an accepted alternative for possible solutions. We believe that such an environment can enhance technical agility, encourage innovation, foster software re-use and ultimately save resources. There will be challenges, which will require commitment to overcome. However, no one ever promised that developing open source software would be easy, just that it would be worthwhile.

REFERENCES

- Archibald, James, "Ten things you didn't know about open source". Retrieved on 16 May 2007 from <http://www.tectonic.co.za/view.php?id=1465>.
- Stenbit, John, "Open Source Software (OSS) in the Department of Defense (DoD)," Chief Information Officer, 28 May 2003.
- Herz, J.C., Lucas, Mark & Scott, John, "Open Technology Development Road Map Plan," April 2006. Retrieved on 17 May 2007 from <http://www.acq.osd.mil/jctd/articles/OTDRoadmapFinal.pdf>.
- Parsons, Doug & Wittman, Rob, "Open Source Opens Opportunities for Army's Simulation System," CrossTalk, volume 18, January 2005.
- Training and Doctrine Command, "OneSAF Mission Need Statement," 2004.
- Coar, Ken, "The Open Source Definition:," 7 July 2007. Retrieved from <http://opensource.org/osi3.0/node/4> on 17 May 2007.
- Agile Alliance, "Principles Behind the Agile Manifesto." Retrieved on 19 May 2007 from <http://agilemanifesto.org/principles.html>
- Parsons, Doug & Surdu, John, "Army Program Balances Agile and Traditional Methods with Success," CrossTalk, April 2006.
- Weber, Steven, "The Success of Open Source," Harvard University Press, 2004.
- Kenwood, Carolyn A., "A Business Case Study of Open Source Software," Mitre Product MP01B0000048, July 2001.
- Bollinger, Terry, "Use of Free and Open Source Software (FOSS) in the U.S. Department of Defense," v1.2.04, MITRE report number MP 02 W0000101, 2 January 2003.
- Blake, James T., "Authority to Operate (ATO) for the One Semi-Automated Forces (OneSAF) Objective System Version 1.0 (OOS V1.0)," SFAE-STRI-CIO, 28 August 2006.
- Tuma, David, "Open Source Software: Opportunities and Challenges," CrossTalk, Vol. 18, January 2005.