Are You Ready? The Open Technology Development Challenge

Bela Joshi, Ph.D. and Curtiss Murphy
Alion Science and Technology Corporation, BMH Operation
Norfolk, VA
bjoshi@alionscience.com, cmmurphy@alionscience.com

ABSTRACT

The Department of Defense (DoD) released the Open Technology Development (OTD) Roadmap in order to reduce the cost of acquisition and to increase interoperability and re-use across DoD programs. OTD methodologies rely on the application of common standards and interfaces; open source software and design; collaborative, distributed culture and online tools; and technological agility. Adopting an OTD approach implies a paradigm shift in the procurement and management of such programs, as well as in the day-to-day activities of a contractor's development team.

This paper presents case studies of two DoD projects that are early adopters of Open Technology Development: the Common Distributed Mission Training Station (CDMTS) and Delta3D. CDMTS is a Government Off The Shelf (GOTS) cross platform Instructor Operator Station (IOS) framework. Delta3D is an Open Source Software (OSS) Gaming Engine that is used to develop 3D visualizations and game-based trainers.

In this paper, we outline the practical implications of developing and maintaining next generation (training) systems in an open and collaborative manner. We identify the challenges associated with managing multiple customers and funding lines; establishing an effective development team; ensuring common coding standards and practices; and performing configuration and release management in such an environment. Finally, we explore how these two DoD projects address the key challenges and focus areas identified in the OTD Roadmap.

ABOUT THE AUTHORS

Bela Joshi is a Software Engineering Science Advisor at Alion Science and Technology. She is the Project Engineer of the CDMTS development team. Bela's interests include the application of Agile methodologies and best practices in software development. She serves as a member of the Engineering Process Group in the BMH Operation of Alion. Bela has an Electrical Engineering background and received a PhD in Engineering Management from Old Dominion University, Norfolk, VA. She also has an SEI Certificate in CMMI.

Curtiss Murphy is a project engineer at Alion Science and Technology. He manages the game-based training and 3D visualization development efforts for the BMH Operation of Alion and is responsible for the efforts using the Navy's open source game engine, Delta3D (http://www.delta3d.org). He has managed numerous projects that used or contributed to the Delta3D Open Source engine for a variety of Marine, Navy, and Joint DoD customers. He previously managed the Fire Mission Alpha project for the Office of Naval Research – the only non-Army project built on America's Army. He has been developing and managing software projects for 14 years and currently works in Norfolk, VA. Curtiss holds a BS in Computer Science from Virginia Polytechnic University.

Are You Ready? The Open Technology Development Challenge

Bela Joshi, Ph.D. and Curtiss Murphy
Alion Science and Technology Corporation, BMH Operation
Norfolk, VA
bjoshi@alionscience.com, cmmurphy@alionscience.com

INTRODUCTION

Under traditional Department of Defense (DoD) practices, agencies use their individual acquisition pipelines to acquire products that satisfy their individual needs. While such an approach may be practical for some product types, it often results in proprietary solutions, vendor lock-in, lack of inter-operability, and inextensible systems (Scott et al, 2006). This increases the time and cost of bringing critical technology to the warfighter. Furthermore, Scott et al point out that "in order to remain competitive in a rapidly shifting technological landscape… DoD's software development and business processes must break out of the industrial-era acquisitions mold".

Meanwhile, the commercial sector has been employing "open" practices. Using these practices, communities with similar needs develop and share common standards, tools, algorithms, designs, and even source code. They re-use and build upon each other's work; thus enabling enormous gains in productivity and efficiency (Scott et al, 2006). Fortunately, the DoD is beginning to recognize that these techniques can help revitalize DoD programs.

In order to leverage this "open" approach, the Deputy Undersecretary of Defense released the Open Technology Development (OTD) Roadmap Plan (Scott et al, 2006). It describes the urgent need for the DoD to transform the way it does business by adopting open concepts. "The national security implications of open technology development (OTD) are clear: increased technological agility for warfighters, more robust and competitive options for program managers, and higher levels of accountability in the defense industrial base" (Scott et al, 2006).

The Roadmap outlines the following main areas that the DoD needs to address:

- Open Standards and Interfaces
- Open Source Software and Designs
- Collaborative/Distributed Culture and Online Tools
- Technological Agility

The Roadmap acknowledges that there will be many challenges in this transformation. After all, OTD implies a massive transformation in the day-to-day operations of the training, acquisition, and vendor communities. Some of the challenges outlined in the OTD Roadmap include:

- Culture and Process
- Software Project Governance
- Software Policy and Licensing
- DoD Acquistion

In this paper we seek to bridge the gap between the concepts outlined in the OTD and their actual implementation. We describe the processes used by two mature Modeling & Simulation (M&S) training projects that started using open principles years before the publication of the OTD Roadmap. The first project is a Government Off the Shelf (GOTS) product and the other is a pure OSS product. We discuss practical considerations, implementation details, challenges and lessons learned based on the experiences from these successful projects. We also take a look at the differences and similarities of GOTS and OSS projects using OTD.

OPEN FOR BUSINESS

We start this paper by presenting a brief overview of the value of open technologies for both the commercial sector and the military. Wikipedia defines Open Source Software (OSS) as "computer software whose source code is available under a license that permits users to study, change, and improve the software, and to redistribute it in modified or unmodified form." (Wikipedia.org, 2007) Although "open" is often used in reference to software code, it can apply to almost anything including standards, designs, interfaces, hardware, music, videos, documents, and text. There are many ways to designate something as open, but they are all based on the same fundamental concept: a product is open if the source materials can be used, modified, and redistributed by someone other than the creator. For the purposes of this document we will put all the variations of "open" into a single bucket and refer to them under the same banner.

The Private Sector

The private sector offers numerous examples of successful Open Technology Development. Examples include Java, Apache, Eclipse, GNU, Subversion, GIMP, OpenGL, MySQL, and of course, Linux. Popular open projects are used by millions of people. For instance, the most widely used web server on the Internet is the Apache HTTP Server with 58.86% market share (Netcraft Ltd, May 2007). SourceForge, an open source hosting site has over 130,000 open source projects. In all likelihood, you have used open products in your daily work and you may even be reading this with an open product. Companies both large and small have been vested in open technologies for decades. Open technologies are so pervasive that "OSS technology stacks now form the basis of the bulk of Internet and information sharing technologies" (Scott et al, 2006). David Wheeler's paper contains an excellent survey of OSS including quantitative data on reliability, performance, security and scalability (Wheeler, 2007).

US Navy OTD Approach

The US Navy policy includes a plan for the "Naval Open Architecture". Naval Open Architecture is described as "an enterprise-wide, multifaceted business and technical strategy for acquiring and maintaining National Security Systems as interoperable systems that adopt and exploit open system design principles and architecture" (Guertin, March 2007). It specifically calls for adapting and evolving "commercial" successes by encouraging collaboration; modular and flexible architecture; and acquiring government purpose rights to promote transparency in design. The US Navy Open Architecture plan even advocates the need for open business models for acquisition. (Mullen, Sep 2006).

US Air Force OTD Approach

The US Air Force strategy for adopting Open Source Methodologies includes three stages. (Reichers, 2007). The first stage, described as the "Crawl" stage, requires implementation of Open Standards, Interfaces and Data. The second stage ("Walk") employs open source concepts and agile development. The final stage envisions a common repository for reusable code, components, data, and standards.

US Army OTD Approach

The US Army's vision for OTD is to "establish an open, controlled software foundation, manage access to trusted authoritative C2 and ISR data sources, and open use of commercially available tools" (Justice, 2007). The Army's philosophy is moving from "building" tactical command and control software applications to "composing" these with existing commercial and open source software. In the Army's gaming arena, Pete Marion, of PEOSTRI, expressed the Army's need for an open approach: "What we are looking for is to have a common infrastructure in place, and then contract out for the content." (Weirauch, 2007)

CASE STUDY #1 – CDMTS (GOTS)

Overview of CDMTS (Case #1)

Our first case study describes the development of a GOTS Open Source Software Project. The Common Distributed Mission Training Station (CDMTS) is an Instructor Operator Station (IOS) that is based on Human Factor Engineering principles (Joshi et al, 2006). The theoretical framework of CDMTS specifically addresses cognitive workload, IOS training, and distributed mission training (Walwanis-Nelson et al, 2003). The key capabilities of CDMTS are as follows:

- Integrates virtual and constructive assets in a distributed simulation network
- Serves as a common user interface for multiple constructive simulation applications
- Capable of performing planning, execution, control, and After Action Review (AAR) of an exercise
- Promotes modularity and extensibility for multiple deployments via a plug-in architecture

Figure 1 contains a screen shot of CDMTS being used to monitor an exercise.

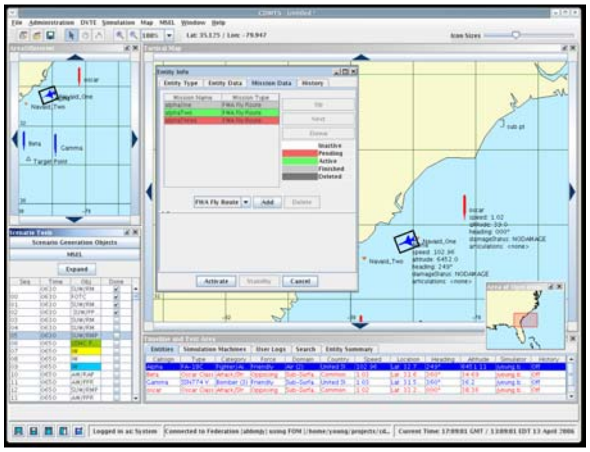

Figure 1. CDMTS with an Exercise in Progress

History (Case #1)

CDMTS was conceived in Fiscal Year 2004 as a single user interface for multiple constructive simulation applications, with initial funds from Joint Forces Command. In the following year NAVAIR Training Systems Division (TSD) assumed program management with the goal of evolving it to a common Instructor Operator Station (IOS).

CDMTS is primarily developed by a small team of core developers. This team is responsible for the architecture, technical design and vision. The core team also provides configuration management support and release management support. Auxiliary contributing members are added at the direction of the government to provide specific functionality.

OTD Benefits (Case #1)

This section examines the OTD benefits that the CDMTS project continues to reap.

Re-use across DoD communities: The entire CDMTS user community often benefits from the additions paid for by any one customer. Features requested by one customer are typically built in the main source tree and are then used by the rest of the community. For example, in FY 2007, features built for the Navy's Virtual Technologies and Environments (VIRTE) program were reused by the USMC Deployable Virtual Training Environment (DVTE) program. Similarly, functionality added for the Mission Rehearsal Tactical Team Trainer (MRT3) were reused by the MH-60R Tactical Operational Flight Trainer (TOFT) community.

Reduce Cost (Money and Time): CDMTS actively seeks out mature open source technology before starting to develop new features. For example, CDMTS uses OpenMap (http://openmap.bbn.com) as the basis of its 2D map. The re-use of OpenMap components significantly reduces the cost and time it takes to develop and deploy CDMTS. This allows government resources to be devoted to new functionality and technological challenges instead of re-inventing the wheel.

Extensible and Customizable Architecture: CDMTS utilizes a plugin architecture to facilitate development of features for the unique needs of various platforms. The plugin framework allows the team to generate modular code or plugins, while keeping a clean interface for integration into the core CDMTS. This reduces development cost while facilitating rapid deployment.

Key enablers (Case #1)

In this section, we present some of the key enablers for the CDMTS case study.

Software Process: Software process has a significant impact on the success of any project; perhaps even more so for an OTD project. The software process followed by the team is comprised of best practices culled from Agile Methods, Extreme Programming and SCRUM. (http://agilemanifesto.org, 2007; http://extremeprogramming.org 2007; http://scrumalliance.org, 2007). A description of some of the salient features of the process follows.

The team has **collective ownership** of code. This implies that a developer is empowered to suggest new ideas, make improvements and fix bugs in any area of code. **Unit tests** are required for all new code. Unit tests are also written when bugs are fixed in existing code. The entire suite of unit tests is run by each developer prior to checking in code. Manual tests plans are produced to perform end-to-end functional testing. Test plans are run before each release. The team is encouraged to constantly **refactor** code. The library of unit tests provides a safety net that promotes rapid evolution of the design and code. Another aspect of CDMTS development is the ease of access to **customer proxies**. Subject Matter Experts are co-located with the development team and provide direction during feature development. The team participates in "**daily stand up**" meetings where developers describe their progress on the previous day, work planned for that day and any problem areas and/or risks. The daily stand up meetings provide a powerful forum for developer-to-developer communication.

Tools: **Trac**, an online project management tool is used to create, assign and keep track of development tasks (http://trac.edgewall.org). The wiki based tool has a web site to support needs of a distributed development environment. The web site is used for test plans, Frequently Asked Questions, software releases and downloads. Instant messaging is used by the distributed team to get rapid responses to questions as they arise during the working day. **Continuous integration** is performed via the Cruise Control Server (http://cruisecontrol.sourceforge.net) to integrate code and run unit tests every time a developer commits changes to the code base. If the build or any unit tests fail, the developer who made the offending check in gets notified via email. **Email lists** are integrated with Trac, the Subversion (SVN) source code repository (http://subversion.tigris.org) and CruiseControl to keep the team abreast of changes. (It should be noted that all tools used are free OSS tools.)

Life cycle model: Traditional models such as the waterfall do not lend themselves to projects with multiple customers, feature requests and frequent releases through out the year. Consequently, CDMTS uses an **iterative development** life cycle model which is suited for multiple iterations (http://extremeprogramming,org).

Challenges (Case #1)

As a GOTS project, CDMTS faces challenges that span the cultural, technical and political areas. Some of these challenges are presented here.

Cohesive technical vision: With multiple organizations contributing code at the direction of the government, special attention has to be given to maintaining a cohesive technical design and architecture. To facilitate this, a single organization should be responsible for technical vision and design, establishing coding standards and reviewing code. This cohesive vision is provided by the 'Benevolent Dictator', a concept explained later in this paper.

Visibility: Visibility into tasking of all organizations is important to eliminate duplicate code and plan for re-use of components in development. This can be achieved by sharing statements of work and feature requests by all organizations.

Distributed Team: A geographically dispersed team comprised of multiple organizations imposes constraints on communication. In addition to daily stand up meetings, in-person Technical Interchange Meetings are held at least annually. This ensures that technical design decisions are taken in a collaborative team environment.

Configuration Management: This critical function gains more importance in the OTD environment, where multiple organizations are simultaneously developing features on different branches of the source code repository. In order to ensure the integrity of the repository, the code from the branches needs to be carefully merged into the main tree, while avoiding conflicts and ensuring that no functionality is lost. For Case Study 1, there is a single organization that hosts the repository and is responsible for merging code from branches into the main trunk.

Maintenance and Quality Assurance: With different organizations contributing code, special attention must be paid to testing new features as well as regression testing. When bugs are found, these need to be communicated to the originating organization for fixing. Program management has to establish clear authority for this function. This important activity is often overlooked in DoD acquisition cycles. Assigning a single organization as the custodian is an effective solution.

CASE STUDY #2 – DELTA3D (OSS)

Overview of Delta3D (Case #2)

Delta3D is an open source game and 3D visualization engine that provides the essential capabilities and code needed to develop game based applications. It is particularly geared to the M&S community and includes typical M&S features such as Large Terrains, AAR, Task Tracking, High Level Architecture (HLA) and Distributed Interactive Simulation (DIS) protocol interfaces, Learning Management System (LMS) support, and Cross Platform support. (Murphy, et al 2006). Delta3D applications are built using core libraries for actor creation, message passing, and game management. In contrast to the 1st case study which was GOTS, Delta3D is a completely open source project that is easily available via the internet.

The key features of Delta3D are as follows:

1. Cross-platform C++ Application Programming Interface (API)
2. High level libraries for baseline application framework.
3. Libraries for 3D rendering, user input, audio, physics, weather, character animation, user interface, networking, level creation.
4. Graphics support for 30+ formats, OpenGL Shading Language, and Particle Effects.
5. Active community website with 40+ tutorials, 30+ example applications, and 30+ articles.

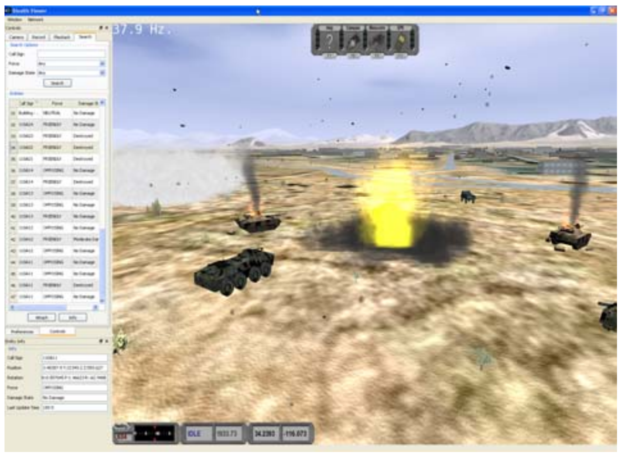

Figure 2. USMC Delta3D Stealth Viewer

Figure 2 shows a screen shot of an HLA Stealth Viewer application built with Delta3D for USMC.

History of Delta3D (Case #2)

Delta3D was originally developed by the Naval Postgraduate School (NPS). The first incarnation was called the P51 graphics library and dates back to 2002. This first version was a simple wrapper for a collection of popular open source graphical libraries. In 2004, it was officially renamed to Delta3D. In 2005, the current website was launched (www.delta3d.org).

By most standards, Delta3D would be considered a successful open Source project. It is used by dozens of companies and organizations around the world, has 1400+ registered users, and over 10,000 forum posts on its website. The project continues to be maintained by NPS, but has become so successful that the majority of code is now contributed by developers from the Delta3D community.

OTD Benefits (Case #2)

In order to demonstrate the fundamental approach of OTD, it is important to understand how an open project is funded and evolves. Delta3D is widely used, especially within the M&S community, making it difficult to list every customer and contributor. Below are a few of the most significant contributions from projects in the last couple years.

Table 1. Delta3D Contributions

Organization	Project/Feature
USMC – PMTRASYS	Game Manager, Messaging Infrastructure, HLA, Sim Viewer Core, Stealth Viewer
USN – NETC	STAGE Map Editor, Dynamic Actor Layer, LMS Integration, Website
US ARMY – PEOSTRI	Common Sensor Model for Night Vision
NPS	Creator and Official Maintainer
USA – Alion	Stealth Viewer User Interface, Character Animation, Road Dead Reckoning
EU – Virthualis	Client/Server Networking
JNTC	AAR Record/Playback, Task Tracking

If you look at the table, it appears that each organization is funding features in isolation based on their own needs. Although that is partly true, it is only half of a really interesting story. For instance, to figure out how the European Virthualis conglomerate was able to develop client-server networking components, you'd have to first look at the underlying Game Manager architecture developed for USMC's DVTE. But the Game Manager wouldn't exist without the Dynamic Actor Layer contributed by the Naval Education and Training Command (NETC). Each new technology is built on top of the others - a series of building blocks that contributes to the whole open source project. As another example, consider the Stealth Viewer developed by Alion. This technology could not exist without the USMC Sim Viewer Core architecture, which in turn relies on the AAR record/playback capability developed for Joint National Training Capability (JNTC), which in turn exists only because of the core infrastructure developed by NPS.

The key point is that each of these individual technologies evolved from previous developments by other organizations. Due to the extensive amount of re-use, each organization bore only a small portion of the risk and cost of the overall technology stack. Case #2 is a great example of what the OTD Roadmap hopes to achieve. "The 'Open Source Model' is a very practical way of evolving technology in a rapidly changing environment. The supporting collaborative tools that have enabled open source development harnesses the collective wisdom, experiences, and requirements of its most demanding users to ensure that needs are rapidly met." (Scott et al, 2006)

Key enablers (Case #2)

In the following sections, we present some of the key enablers for the Delta3D case study.

Software Process: Similar to the first case study, the Delta3D team credits their Team Process as an extremely important part of their success. The Delta3D team uses Agile development techniques taken from SCRUM, the Agile Manifesto, and Crystal. The team engages in **Daily Stand Ups** and follows a **Weekly Iteration** cycle. The team uses build servers to ensure **Constant Integration** of code. The Delta3D team strives for **Co-location** whenever possible. Having the team members located in the same area maximizes knowledge transfer through **Osmotic Communication** (Cockburn, 2005). **Unit Tests** are a critical part of minimizing defects across multiple organizations and are especially useful when code is refactored. As with Case 1, **Collective Ownership** of code is critical to the success of Delta3D. Unit tests provide a layer of safety that allows many developers to work on any aspect of the system. After the overall requirements are broken down, individual developers are then responsible for selecting, estimating, tracking, and completing their tasks. This process is known as **Task Ownership** and is accomplished through use of Trac, task cards, and other information radiators (Cockburn, 2005).

Tools: Although developer tasks and bugs are managed in Trac, the most important tool is **Subversion**. The Delta3D team uses the **Sourceforge** (www.sourceforge.net) subversion repository to house Delta3D code. The Delta3D website is built with the open tool, Geeklog (www.geeklog.net). Geeklog provides powerful forums which are a critical part of the community. Some developers use the open development environment, Eclipse (www.eclipse.org).

Reuse: Delta3D is a fully open Source project, and it routinely leverages other open projects. At any given time, Delta3D leverages as many as 12 other open source projects. Some of them are key parts of the engine, such as Open Scene Graph (OSG) and some are staples of the gaming industry, such as Open Audio (OpenAL). The use of other open projects significantly reduces the time and cost needed to develop and maintain Delta3D.

Challenges (Case #2)

Delta3D reflects many of the desired benefits of the OTD Roadmap, but it is also a good example of some of the challenges open projects face.

Slow Start: Successful open projects are often slow to catch on. “Successful projects are characterized by long incubation periods with delayed rewards” (Babcock, 2007). Although Delta3D is now widely used, it has been in existence since 2002. As an open project, Delta3D grew organically to meet the needs of the early adopters. The project has been successful because it provides a much needed capability for an M&S oriented gaming engine.

Adoption and Awareness: One of the early hurdles to Delta3D was generating general awareness of the engine. There are so many places people go to find information about software that it requires a conscious effort to target them. Since the primary user base of Delta3D is the M&S industry, the general strategy was to do presentations and papers at popular industry events such as the annual Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC). Over time, this strategy has paid off.

Maintenance: The Delta3D project has had great success with organizations willing to use and contribute to the engine. Unfortunately, the DoD community is typically oriented towards either the R&D or acquisition cycle. This means that few organizations are interested in directly funding ongoing maintenance of an open project. This has been and will likely continue to be a serious long-term impediment to the project. The OTD Roadmap highlights this as one of the primary challenges to this transition. (Scott, et al 2006).

Distributed Team: Delta3D is managed by two primary groups: NPS and Alion. These two groups are the Benevolent Dictators (see below) for this project. As with all distributed teams, being on opposite sides of the continent (NPS in Monterey, CA and Alion in Norfolk, VA) introduces the possibility of significant issues such as loss of focus, conflicting designs, and separate release schedules. These issues are primarily handed through frequent and open engineering discussions. Discussion is brutally honest and occurs at least once a week.

Configuration Management: In the past, multiple repositories were used by the two primary overseers of the project. This resulted in painful merges as well as a significant disconnect in between the merge points. Now, a single central repository is used and periodic branches are used to support disparate release schedules. The team has identified some drawbacks to a single repository, with the most notable being frequent build issues between platforms. However, on the whole, the single repository has resulted in more stable code, more open discussion, and quicker roll-outs of new technology to the larger Delta3D community. Having a single repository also encourages collective ownership and drastically decreases the time it takes to find and fix bugs.

CROSSING THE OTD DIVIDE

These two case studies present some of the challenges and successes faced by open projects from both the Government Off the Shelf (GOTS) and Open Source Software (OSS) perspective. One of the intents of the authors was to examine the similarities and differences between a GOTS and an OSS project. Upon reflection, we believe these two projects are more similar than different. To cross the OTD divide, they faced many of the same challenges and achieved success through many of the same techniques. In the final examination, we identified these three minor differences:

Table 2. Minor Differences - GOTS vs OSS

Government Owned	Open Source Software
The Government owns the software	The community owns the software
Features added at the direction of the government	Features added at the direction of early adopters and significant contributors
Distribution controlled by the Government	Distribution controlled by members of the community

The rest of this paper uses the lessons of these two case studies to reflect upon the major topics of the OTD Roadmap. This discussion includes the four areas identified in the OTD Roadmap: Open Standards and Interfaces, Open Source Software and Design, Collaborative Culture & Online Tools, and Technological Agility. It also includes three of the four challenges: Culture and Process, Software Project Governance, and Software Policy and Licensing. DoD Acquisition is outside the scope of this paper.

CHALLENGE - CULTURE AND PROCESS

The OTD roadmap highlights Culture as the single biggest challenge to the success of "open" in the Department of Defense. The nature of bureaucracy has led to a rewards system that focuses on maintaining the status quo as opposed to increasing value to the government. (Scott et al, 2006). Below are some of the lessons learned about Culture from these two case studies.

Business Model

Contractors can still make money in an OTD environment. The primary difference is a paradigm shift from vendor lock-in to integration and open architecture. Vendors can either compete on the strength of their services or by offering proprietary products that integrate well into open infrastructures.

Sensitivity

Because of the nature of DoD projects, a common assumption is that a project is unable to participate in OTD because it involves sensitive, export controlled, or classified information. Our case studies showed two fallacies with this argument. First, a project can certainly participate in OTD by leveraging and building upon other open technologies. The second way these projects can contribute is by separating sensitive material from the core software. Often, it is possible to isolate the sensitive data, models, and algorithms from the core software, so that they can still leverage and contribute to the underlying open project. As an example, Delta3D supports the concept of Actor libraries that encapsulate sensitive behaviors. In the DVTE deployment, the behavior for driving and firing the HMMWV has been built into an Actor library, separate from the underlying engine. This separation enabled significant contributions to the core engine while still protecting sensitive material.

Intellectual Property

Within an OTD framework, it is still possible to maintain Intellectual Property and proprietary technologies. Similar to sensitive material, the key is to be able to integrate the proprietary technologies into an open platform. As an example, some organizations have integrated proprietary technologies such as SpeedTree™ and DI-Guy™ directly into Delta3D applications.

Ongoing Maintenance – Funding

At some point, an open project reaches a level of success where maintenance becomes an issue (Babcock, 2007). Existing users expect bugs to be fixed, new users want tutorials and detailed documentation, and the entire community wants a frequent schedule of releases. Both case studies show that many organizations are willing to participate and contribute to the OTD process, but few are willing to provide a mechanism for ongoing support. The OTD Roadmap recommends the following: "Program managers and leaders need to take ownership of the life cycle costs of their solutions. … Recognition and rewards need to be established for program managers that deliver open solutions" (Scott et al, 2006).

CHALLENGE – SOFTWARE GOVERNANCE

Another major challenge identified by the OTD Roadmap is the issue of Software Project Governance. The issue is who will guide the project and maintain a clear technical vision. For our two case studies, this is managed by the Benevolent Dictator.

Benevolent Dictator

A Benevolent Dictator refers to a person or small group that guides and directs the project. Typically they are the evangelists of the project and are the only ones that can directly modify the official baseline. Technically speaking, there is a distinction between a Benevolent Dictator and an Exclusive Group. However, since they are very similar when put into practice, we will refer to them both as the role of Benevolent Dictator.

The importance of a Benevolent Dictator cannot be over emphasized (Babcock, 2007 & Scott et al, 2006). Open projects are typically agile (see below) and evolve quickly based on the needs of their community. However, without a cohesive vision, the project can easily become just one more of the 130,000 open projects on Sourceforge. CDMTS and Delta3D each have a clearly defined group that guides the project, approves code, and holds the responsibility for the direction of the product. Both projects credit this as key to their success. For Delta3D, this group is the combination of Alion and the MOVES Institute at NPS. For CDMTS, this group is comprised of NAVAIR Program Management and Alion.

Although critical to the success of both projects, this role does have a potential drawback. If acquiring funding for the Benevolent Dictator role is difficult, this group should be prepared to shoulder associated expenses. Any OTD project that chooses this model should plan to account for the management of the project, release preparation, standards, testing, and quality, in addition to pushing the project ahead through development of new features and functionality.

CHALLENGE - SOFTWARE LICENSING

A major hurdle for open technologies is licensing. It is easy to overlook the complexity of licensing because the general assumption is that “open” equates to “free for all”. For better or worse, this is not true. Just as with proprietary technologies, open projects must have some sort of license or release agreement. Unfortunately, there are hundreds of similar, but distinct licenses to choose from. Examples are GNU GPL, Library GPL, BSD, Apache, and Opensource.org, to name a few. Each license has complex subtleties that distinguish it from the others. Some licenses are very forgiving and some are so strict they are almost viral in nature. The Delta3D case study uses a fairly liberal license called Lesser-General-Public-License (LGPL). The CDMTS case study uses a specific government-administered release agreement.

The legal issues surrounding licensing are many. Licensing has a bearing on warranty and reliability of the software. Licensing also complicates who can use the technology and how. This becomes especially important once the project is large enough to receive contributions from external sources; thus introducing the risk of Intellectual Property infringement. Unfortunately, many of these legal issues and full ramifications of these types of licenses are yet to be officially resolved in the software industry. Currently, the American Bar Association advises lawyers to be extremely cautious about recommending open source for inclusion in commercial software products (Amercian Bar Association, 2007). Due to all these reasons, the authors recommend that each project carefully consider their license options before selecting one.

OTD AREA - COLLABORATIVE CULTURE

Most people already associate open projects with the concept of a collaborative culture. This is commonly known as the ‘community’. A project’s ‘community’ is the entire collection of users, contributors, interested parties, and stakeholders, including the Benevolent Dictator. The activity of the community is critical to the health of an open project and is often used as a key measure of success (Babcock, 2007).

Although a project cannot directly control the growth of its community, it can take steps to encourage it. For instance, all communities can benefit from online collaborative tools. Projects can ensure the community has visibility into the API and system design. Our two case studies leverage tools such as forums, email lists, a knowledge base, tutorials, Wiki’s, API documentation, news, and web pages. They both encourage feedback, contributions, submissions, and interaction. At a minimum, a project should have some sort of email list or online forum where the community can ask questions, explore problems, and voice their concerns.

OTD AREA - TECHNOLOGICAL AGILITY

Another key idea for open projects is technological agility. Simply stated, it refers to the ability to change direction quickly without being encumbered by bulky processes. It is a powerful concept that is both an advantage and a disadvantage. It’s an advantage because an OTD project can drastically shift direction in order to meet the needs of major adopters or market forces. Agility helps OTD projects rapidly develop new technologies for the warfighter. It’s a disadvantage because, if not managed well, it can have the potential to be difficult to predict and control.

Agile Methodologies

In software, the term Agile typically refers to a set of techniques developed to improve software quality while also reducing development time. Agile methodologies are typically in sharp contrast to the traditional ‘waterfall’ lifecycle model. The traditional waterfall model is known for a strong delineation of phases such as requirements, design, development, verification, and validation. One phase cascades down, like water, to the next, with an emphasis on preventing changes in later phases. Agile methodologies assume this approach is both slow and inherently flawed. Instead Agile recognizes that change is inevitable and that the phases of a project often overlap.

Although open projects do not need to use Agile methodologies, our two case studies successfully leveraged them to achieve technological agility. These techniques were particularly useful in supporting contributions from multiple sources with sometimes competing requirements. Agile techniques such as daily stand ups, unit tests, rapid iterations, continuous integration, and collective ownership were incorporated into the software process of both case studies. Additional techniques such as constantly working software, task ownership, osmotic communication, and customer proxies were also beneficial and used by at least one of the case studies.

OTD AREA - OPEN SOFTWARE AND DESIGN

OTD involves a self-organizing community that is able to contribute and participate in the process of software development. In so doing, OTD introduces an agreement that binds the community: each contributor accepts some loss of control over source code in order to benefit from the upstream and downstream propagation of ideas and technologies. Depending on the perspective, this could be seen as a loss of control or as a huge opportunity.

Once a contribution is made to the project, it belongs to the community and may get used in ways the contributor never envisioned. Sometimes the material gets changed in ways the original contributor does not want. (On the Delta3D project, this happened with modifications to some documents and to the STAGE level editor). However, more often than not, an individual contribution is debugged, made faster, improved, or even massively enhanced. Naturally, future users gain advantage from the improvements downstream.

In addition to the downstream effect, sometimes improvements go upstream as well. In other words, when a technology is improved, it ends up benefiting the original contributors. Consider the following Delta3D example. When NETC contributed the Dynamic Actor Layer two years ago, they could not have known that USMC would come along and build the Game Manager on top of it or that JNTC would then add a task tracking capability. However, these two features were critical during NETC's follow-on Learning Management System project. So, USMC and JNTC benefit downstream, but NETC benefited upstream.

This type of upstream benefit is especially true for significant contributions. In effect, users that contribute large sections of technology that are adopted by the community get the biggest benefits. The USMC DVTE project is a perfect example of this. USMC contributed a very large module called the Game Manager that is now the basis of all Delta3D applications. Since the Game Manager was first contributed, the Delta3D community has made thousands of updates, additions, bug fixes, performance improvements, tutorials, and publications. As USMC is still using the Game Manager, they continually reap dividends from their original investment.

OTD AREA - OPEN STANDARDS AND INTERFACES

For an open project to succeed it is imperative that the project's standards and interfaces are visible to the entire user community. The project should have a transparent and well documented API for other components to integrate with. Both case studies allow users to add their own behaviors and content without having to modify the base layer of software. The CDMTS project accomplishes this through a plugin architecture, while the Delta 3D project has library extensions. In addition, inter-operability and ease of integration are key considerations for both case studies. For example, both CDMTS and Delta 3D use HLA as their primary communication protocol, since this protocol is widely used in the M&S industry.

CONCLUSIONS

In this paper we presented two case studies of successful open projects within the DoD M&S training industry. For each project, we explored OTD benefits, challenges, and key enablers of success. On going maintenance funding, configuration management, ensuring a cohesive technical vision, and distributed teaming were common challenges. Key enablers included team software process, collaborative tools and re-use of other open source components. The need for Agility as well as a Benevolent Dictator role was highlighted.

The OTD Roadmap lays out a clear argument for an open approach. OTD implies a major shift for both the government and vendor communities and demands a major transformation to the way DoD does business. Successes in the commercial industry show us the light at the end of the tunnel; and exemplar projects within the DoD provide the template for success. To cross the OTD divide, we merely need to open our minds and put one foot in front of the other.

ACKNOWLEDGEMENTS

The views expressed herein are those of the authors and do not necessarily reflect the official position of the organizations with which they are affiliated. The authors would like to thank the program managers who have supported this work.

REFERENCES

American Bar Association (2007). An Overview of 'Open Source' Licenses. http://www.abanet.org/intelprop/opensource.html

Babcock, Charles (2007). What Makes The Cut? Information Week. Feb 5, 2007. http://informationweek.com

Cockburn, A., Highsmith, J. (2005). Crystal Clear - A Human-Powered Methodology for Small Teams. Addison-Wesley.

Guertin, N. (2007). Naval Open Architecture: Open Architecture and Open Source in DOD. Presented at the AFEI Conference on Open Technology Development, March 14-15, 2007. Arlington, VA. (http://www.afei.org/brochure/7a03/documents/Nick Guertin_03.14.07.pdf)

Joshi, B., King, R., Teer, B. (2006). CDMTS: A Common User Interface for Multiple Training Environments, Proceedings of the 2006 Interservice/Industry Training, Simulation, and Education Conference.

Justice, B. G. March 14, (2007). Open Source Software Challenge: Delivering Warfighter Value. Presented at the AFEI Conference on Open Technology Development, March 14-15, 2007. Arlington, VA.

Murphy, C., Johnson, E. (2006). Building Game Based Trainers with the Delta3D Game Manager. Presented at the 2006 Interservice/Industry Training, Simulation, and Education Conference.

Netcraft Ltd. (2007). April 2007 Web Server Survey http://news.netcraft.com/archives/2007/04/02/april_2007_web_server_survey.html

Reichers, C. (2007). The Role of Open Technology in Improving USAF Sofware Acqusition. Presented at the AFEI Conference on Open Technology Development, March 14-15, 2007. Arlington, VA. http://www.afei.org/brochure/7a03/documents/Riechers_AFEI_final.pdf

Scott, J., Lucas, M. and Herz,, J. C. (2006). Open Technology Development Roadmap Plan. Prepared for the Deputy Under Secretary of Defense, Advanced Systems and Concepts. http://oss-institute.org/NCOSPR/OTDRoadmap_v3_Final.pdf

Sheffield, Brandon (2005). Breaking the Waves, Doug Whatley and Breakaway Games get serious. Game Developer Magazine.

Walwanis-Nelson, M., Owens, J., Smith, D., Bergondy-Wilhelm, M. (2003). A Common Instructor Operator Station Framework: Enhanced Usability and Instructional Capabilities. Proceedings of the 2003 Interservice/Industry Training, Simulation and Education Conference.

Weirauch, Chuck. (2007). Institutionalizing a Grass Roots Movement. Modeling, Simulation, and Training Magazine.

Wikipedia (2007). Open-source software. http://en.wikipedia.org/wiki/Open_Source_Software

Wheeler, D., (2007). Why Open Source Software / Free Software (OSS/FS, FLOSS or FOSS)? Look at the Numbers! http://www.dwheeler.com/oss_fs_why.html