The Common Image Generator Interface – Is it Really?

Sean Duff, Kenny Dixon
CAE USA, Inc.
Tampa, FL
sean.duff@caemilusa.com, kenny.dixon@caemilusa.com

ABSTRACT

The Common Image Generator Interface (CIGI) is a communication protocol standardized by the Simulation Interoperability Standards Organization (SISO) under SISO-STD-013-2014, Standard for Common Image Generator Interface (CIGI), Version 4.0, dated 22 August 2014. The SISO Product Support Group page for CIGI states, "The purpose of CIGI is to provide interoperability across real-time Image Generator (IG) and Host computational system providers by using a common method of communications. The product will provide a common communications protocol that will enable each disparate visualization tool to quickly interface with other subsystems by providing a set of commonly used subsystem-to-subsystem interactions." This design goal has led some branches of the United States Armed Forces to include the use of CIGI as a requirement when procuring new simulation devices over the past decade and a half. This paper analyzes the success of the CIGI initiative by examining the integration costs and program impacts when upgrading visual systems. Over the past 15 years, the US Navy has issued multiple contracts to build and upgrade MH-60R and MH-60S operational flight trainers for both the US Navy and foreign military customers. Across these various contracts, the flight trainers have been integrated with three different Image Generators (IGs) from three different vendors, all while satisfying the requirement to use the CIGI standard for host to IG communication. This paper will discuss the instances where the CIGI requirement did in fact increase reusability and efficiency across the separate efforts. The paper will also demonstrate where this requirement failed to reduce integration costs. Finally, this paper will discuss opportunities for the CIGI standard, as well as procurement agencies, to improve and truly achieve the goal of reduced integration costs.

ABOUT THE AUTHORS

Sean Duff is the Group Leader of Visual Systems at CAE USA in Tampa, FL. He has over 15 years of experience in military flight simulation, including 7 years in software development and more than 5 years in system architecture, with a focus on design, implementation and deployment of complex training systems. He has experience in leading and managing a small technical team focused on simulation visual systems from image generators and associated interfaces to full-scale visual display systems. Sean has experience in the full Systems Engineering Process including support of the NAVAIR Systems Engineering Technical Review (SETR) processes. He has experience writing proposals of varying levels of complexity for a wide range of training platforms. Sean has been a participant in SISO's CIGI initiatives for years and served on the drafting group for CIG Version 4.0. Sean's current projects focus on Navy Rotary Wing Platforms providing technical leadership and visual systems support for operational training devices. Sean is a graduate of Purdue University with a B.S. in Electrical and Computer Engineering.

Kenny Dixon is a Senior Technical Specialist at CAE USA in Tampa, FL. He has 15 years of design, development and integration experience in image generator interfaces for both fixed and rotary wing military flight simulation platforms. He has experience in the evaluation and acceptance of FAA Level D-equivalent visual display systems. Kenny's current projects on rotary wing platforms for a variety of customers. Kenny is a graduate of the University of Florida with a B.S. in Software Engineering.

The Common Image Generator Interface – Is it Really?

Sean Duff, Kenny Dixon
CAE USA, Inc.
Tampa, FL
sean.duff@caemilusa.com, kenny.dixon@caemilusa.com

INTRODUCTION

The Common Image Generator Interface (CIGI) was originally developed by The Boeing Company and was released to the public as CIGI 1.0 in March 2001. In 2003, the Simulation Interoperability Standards Organization (SISO) formed a CIGI Standing Study Group, which matured the CIGI standard over the years, culminating in the release of SISO-STD-013-2014, Standard for Common Image Generator Interface (CIGI), Version 4.0, dated 22 August 2014. According to its creators, "the purpose of CIGI is to overcome the proprietary nature of existing interfaces by providing it open source, and allow for a plug-and-play solution for any image generator application." (Lechner & Phelps, 2002) This stated purpose has driven some government acquisition agencies to require the use of CIGI as they continue to procure the latest training devices.

This paper explores the successes and challenges experienced implementing CIGI software across several different programs. In some cases, CIGI's stated purpose was realized – increasing reusability of source code. In others, significant effort was required to integrate common software with a new image generator (IG). How can acquisition agencies and industry do better? How can IG vendors do better? How can CIGI do better? What does it mean to be "CIGI compliant"? This paper attempts to answer those questions through the lens of experience.

Note that at the time of writing, the official SISO standard is CIGI Version 4.0. All of the efforts discussed in this paper pertain to contracts that required the use of some iteration of CIGI Version 3. While CIGI v4.0 is not directly backwards compatible with CIGI v3.x, it is a logical progression in the standard and the concepts discussed in this paper are still relevant today.

While contract efforts and image generator vendors are specifically identified in this paper, the data herein is intended to be consumed objectively. In some instances, the data is randomized in order and obfuscated with code names to eliminate any potential bias. Additionally, the raw data has all been normalized to protect against accidental disclosure of company sensitive information. All efforts are made to keep CIGI's experienced commonality as the focus of the paper.

BACKGROUND

At its conception, CIGI was designed to ease the integration of disparate image generators into flight simulators (Lechner & Phelps, 2002). Over time, the simulation industry has found new and wider uses for CIGI. As its scope has grown, so too has the CIGI ICD evolved. Despite this broader perspective, CIGI is, at its core, still aimed to provide a standardized interface and to make simpler the transition from one IG to another. CIGI does not define IG functional requirements, nor does it limit the features an IG can offer. The CIGI ICD provides a standard way for host simulations to communicate with IGs, but implementation of the ICD does not imply that an IG supports all features addressed by the ICD (CIGI PDG, 2014).

To date, no other standardized interfaces exist to compete directly with CIGI. Most Image Generator vendors support CIGI when contractually required. While some IG vendors have moved to make CIGI their only interface, others continue to maintain their legacy proprietary interfaces in parallel as indirect competition to the CIGI standard. However, CIGI's standardized design goals have led defense acquisition agencies in the US and abroad to require the use of CIGI on new contracts as they strive for a more standards-based approach to simulator acquisition.

Over the course of the past 15 years, the United States Navy, through Naval Air Warfare Center Training Systems Division (NAWCTSD), has issued multiple contracts for MH-60R Tactical Operational Flight Trainers (TOFTs). Across those various contracts, the MH-60 Common Baseline software was integrated with three different Image Generators from three different vendors (see Table 1), in all cases meeting a requirement to implement CIGI for host to IG communication.

Table 1. Programs and Associated Image Generators

Year	Program	Cust.	CIGI	IG
2006	MH-60R TOFT #4-6	USN	3.1	Aechelon Technology pC-Nova
2012	RAN MH-60R TOFT	RAN	3.2	CAE, Inc. Medallion
2015	Tech Refresh and Procurement of Simulators	USN	3.3	Collins Aerospace EP-8100

What would eventually become the MH-60 Common Baseline IG interface software was originally developed in 2006 for a contract providing MH-60R TOFT devices to the US Navy. The IG selected for this effort was Aechelon Technology's pC-NOVA™. In 2012, the IG interface software was adapted for a new Foreign Military Sale providing MH-60R TOFT devices to the Royal Australian Navy. For this program, CAE's Medallion was the chosen image generator. Finally, in 2015, the US Navy issued a contract for the Technology Refresh and Procurement of Simulators (TRPS) program, which required an upgrade to the IGs used on the USN's fleet of MH-60R TOFTs and MH-60S Operational Flight Trainers (OFTs). The Collins Aerospace EP-8100 image generator was selected and the IG interface software was again modified.

It is important to note that all three of these efforts called for meeting very similar performance requirements. The air platforms to be simulated were nearly identical in all instances and the target missions of the end users were comparable. The databases had similar control requirements, despite geographical differences, and the moving model sets employed on each program were similar in both size and complexity of controls. The commonality shared among these efforts is important because it allows us to make and accurate apples-to-apples comparison of the integration effort encountered in each instance.

The IG interface software was developed from the start with IG plug-and-play functionality as a major design goal. The project made extensive use of the open source CIGI Class Library (CCL) software to enable easy transition between various versions of CIGI. A layered design approach was employed in an effort to abstract the IG-specific implementation details from the aircraft simulation. Figure 1 shows the high-level architecture of the IG interface software. A full discussion of the architecture is beyond the scope of this paper, but the figure is include here to provide context to the layering discussion and illustrate the attempt to isolate IG-specific implementation from the rest of the software. The intent was to take full advantage of CIGI's stated purpose and limit integration costs when switching to a new image generator. The following sections will discuss where these design goals succeeded and where they fell short.

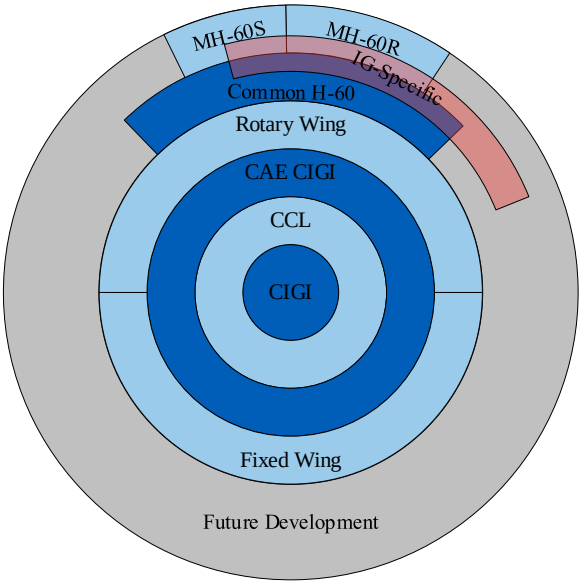

Figure 1. High-Level Software Architecture

INTEGRATION EFFORTS

This section analyzes the three disparate integration efforts. The source code is easily broken into three categories. The first is common source code that is applicable to all of the training devices – shown in blue in Figure 1. Then there is source code that is specific to each image generator implementation – shown in red. Finally, the CCL is open source software maintained by the CIGI community to aid in integration efforts. As it is non-developmental and used as-is for all three instances, the CCL is not considered in this integration analysis.

Common Software

The design approach has allowed IG specific code to be completely encapsulated inside its own layer. This allows the developer to copy the current code base and then add a new layer for the specific image generator that has been chosen for the program. This also allows the developer to identify functionality that may be common between one or more IG specific implementations. In the case of such commonality, efforts have been made to generalize the implementation in one of the parent layers shared by all IG specific layers. Over the course of the three programs analyzed in this paper, the common set of source code has been revised following this exact method. Each successive effort has built upon those before it to refine and expand the common layers. The evolution of the common software code base is illustrated in Figure 2.

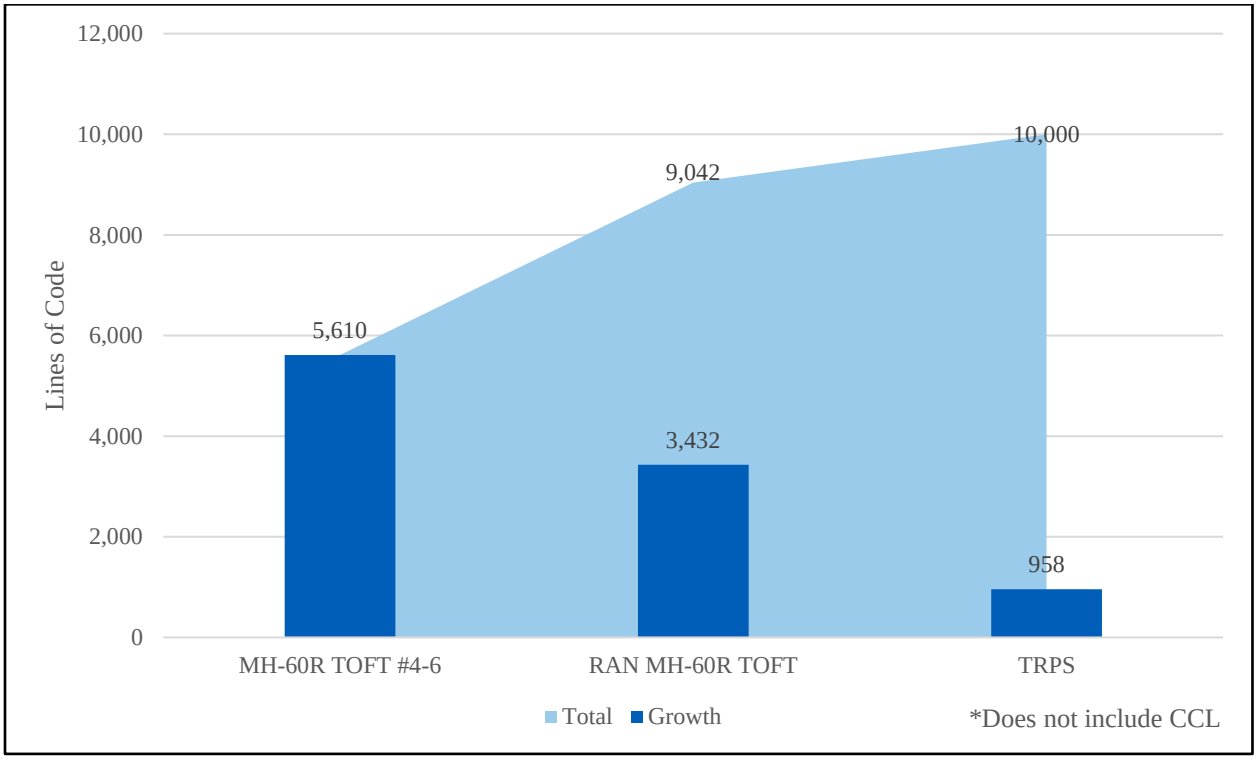

Figure 2. Lines of Code in Non-IG Specific Layers

The overall growth of the common layers is perceived as positive progress for the code base, as anything that can be moved into this layer will possibly reduce the development time for the next IG specific layer. At the same time, the downward trend of the growth in the common software code base through each program is also positive. Analysis of this metric indicates that the design goal aimed at minimizing future integration costs is potentially being realized. It is expected that future integrations will continue to see shrinking modifications to the common code base.

IG Specific Software

The previous observations focused on only the software common to all implementations. Analysis of the IG specific layers tells a different story. Figure 3 shows the lines of code that are unique to each IG as well as the labor hours required to write those lines of code. The IG-specific labor hours were estimated by calculating the fraction of new lines of code attributed to the IG and then taking an equal proportion of the programs' total labor hours. Because the focus of this discussion is CIGI, and not any vendor's specific implementation of the standard, the data for IG specific lines of code has been randomized and the program names have been coded. As discussed in the introduction, this is intended to eliminate any bias for or against any specific image generator.

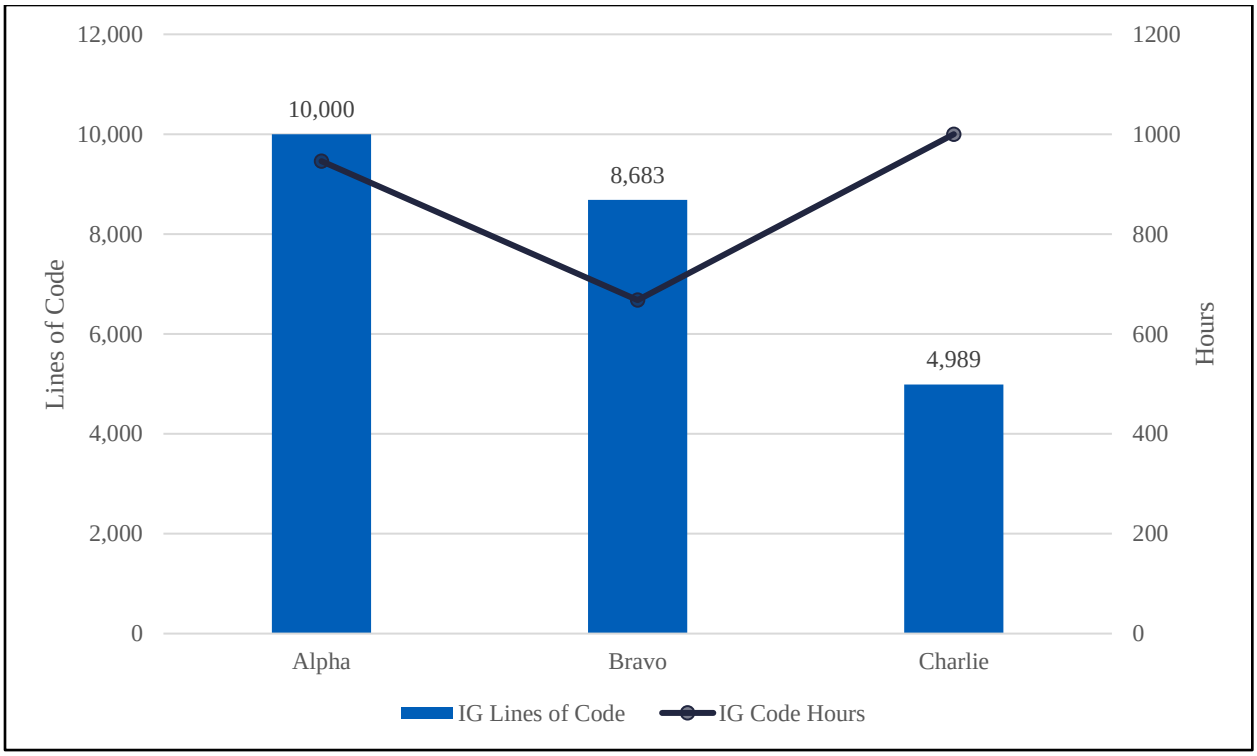

Figure 3. IG Specific Integration Efforts

The raw IG specific source code data, as presented in Figure 3, suggests a downward trend. However, the randomization of the program order means this assumption cannot be made. With that in mind, the data becomes random and no trend can be inferred. Especially notable is the lack of correlation between labor hour and lines of code. With only this information, the cost of integrating any specific IG, in terms of software to be written, appears to be unpredictable.

FINDINGS

Initial analysis of the integration artifacts was enlightening, but still left questions unanswered. Further investigation was performed both objectively and subjectively. The implementation of the CIGI standard by each image generator was explored. Additionally, anecdotal evidence about the disparate integration efforts was collected from the engineers who implemented each interface. These findings are presented in the following sections.

CIGI Compliance

After analysis of lines of code and hours spent across all three integration efforts yielded no clear pattern indicating a potential increase in efficiency over time, further research was required. The next obvious target was to analyze each image generator's compliance with the CIGI standard. However, it turns out that there is no industry consensus opinion on exactly what it means to be compliant with CIGI. To date, the best effort to create such a consensus has been conducted by the North Atlantic Treaty Organization (NATO) Science and Technology Organization (STO).

The NATO Modeling and Simulation Group (NMSG) Research Task Group MSG-118 was created to address the issue of CIGI compliance directly. According to their report, compliance ensures implementation of a standard, which in turn ensures greater reuse and interoperability. The group created a testing framework and began implementation of a suite of testing tools. They ended with the tools in a functional state, but limitations still exist in terms of implemented tests and use cases. Further work is required to cover more use cases and be of better general use to the CIGI community at large (MSG-118, STO/NATO, 2018).

Because this paper is exploring past efforts, applying the testing tools created by STO/NATO MSG-118 was not an option. In lieu of this more objective testing, the Interface Control Document (ICD) used to integrate with each IG was analyzed. The creators of CIGI knew they could not predict every possible use case for an IG interface and they included generic provisions for implementers to cover non-standardized functionality. The two provisions are known as Component Controls and User Defined Packets. The Component Control is a standard CIGI packet that provides a generic mechanism to control various components within the simulation. The User Defined Packet is a framework included in CIGI allowing an implementer to design new packets that conform to the overall CIGI packet structure but contain custom data. Combined, these two generic provisions account for much of the IG-specific development for a given CIGI implementation. For each image generator, the total number of User Defined Packets and unique Component Controls were counted. The results of this analysis are shown in Table 2.

Table 2. Non-Standard CIGI Controls

Image Generator	User Defined Opcodes	Unique Component Controls	Total
Alpha	8	59	67
Bravo	43	0	43
Charlie	2	34	36

The details revealed through this analysis were interesting. All three IGs made extensive use of CIGI's provisions allowing non-standardized messages to be defined by the interface implementers. The Alpha and Charlie image generators heavily favored CIGI's component control messages, while the Bravo IG made exclusive use of CIGI's user-defined message framework. Graphically, the data was even more interesting – as illustrated in Figure 4.

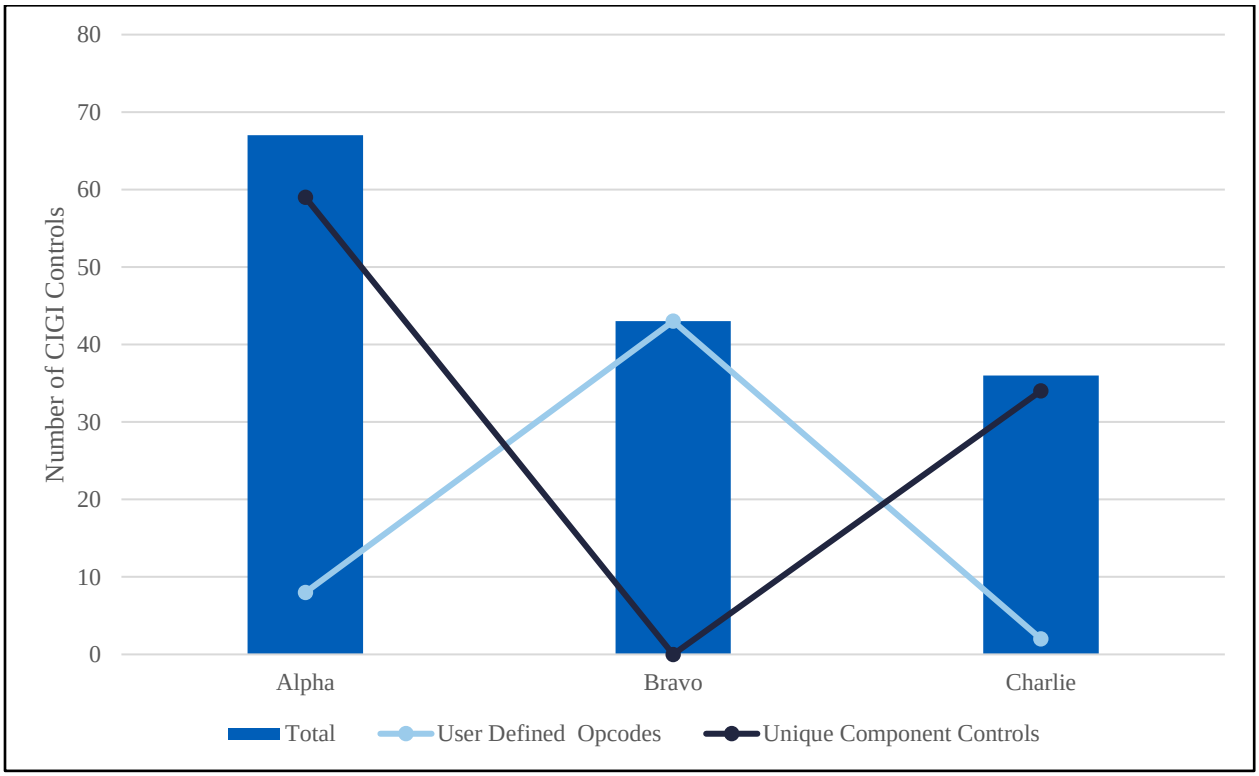

Figure 4. Non-Standard CIGI Controls

It is notable that Figure 4 shares some obvious similarities with Figure 3. There is a clear correlation between the number of non-standard CIGI controls included in a given IG's ICD and the number of lines of code that were written to implement the interface to that IG. For instance, the Charlie IG had the fewest IG-specific lines of code and also used the fewest non-standard CIGI controls. Logically, this makes sense as the common layers of the software baseline should handle the standard CIGI controls with little modification and IG-specific layers are focused exclusively on a particular image generator's unique controls. Surprisingly, there was a strong correlation between the number of unique component controls and the hours required to implement a specific IG's interface.

The Bravo IG used no unique component controls and required fewer hours than either the Alpha or Charlie image generators. The reason for this correlation was not immediately obvious, but discussion with the developers provided extra insight. The Component Control packet's generic nature allows the implementer extensive creativity when it comes to filling the available space with data. The developers found that, at times, the IG vendors had encapsulated portions of their legacy proprietary ICD into a series of component controls. This resulted in a complicated mixture of bits and bytes that must be packed into whole words. The resulting software was not appreciably larger than a similar number of user-defined packets, but the time spent writing and debugging those lines of code was demonstrably greater.

Subjective Analysis

The anecdotal recollections of the software engineers who implemented the disparate IG interfaces support the more concrete findings of this analysis. Basic, "first flight" capabilities were functional very early in the RAN MH-60R TOFT and TRPS efforts. Simple weather and time of day controls were also rapidly in place and usable. Even basic moving model placement and movement was available with little effort. However, the more complex, and IG-dependent, moving model and mission-oriented controls took significant time. This includes things like:

- Complex weather (e.g. 3D clouds)
- Special effects (e.g. weapon launch/impact, smoke, tracers)
- Entities (each IG implements articulated parts and state switches differently, even if using standard CIGI packets)
- Airports and database (same as entities)

This subjective analysis aligned well with the objective data already collected. It also made logical sense, as the basic simulation functionality is well-defined, while the more specific special effects and moving model functionalities rely heavily on any particular IG's traditional, and often proprietary, data formats.

Savings

From a financial perspective, the hope of everyone involved with the CIGI standard is that it results in reduced costs of integration and deployment. This analysis shows that, for the base CIGI functionality, integration costs do decrease over time. The RAN MH-60R TOFT effort enjoyed an approximate 50% decrease in labor hours spent to integrate and deploy the common CIGI portions of the software relative to the first effort, MH-60R TOFT #4-6. And the TRPS effort experienced a further 25% drop relative to the RAN MH-60R TOFT program.

For IG specific efforts, the labor hours required to integrate and deploy the software correlate to the method selected by the IG vendor to implement their non-standard interface details. The Bravo development effort, which eschewed CIGI's component controls in favor of user-defined packets, required just 66% of the labor hours relative to the Alpha and Bravo programs, which relied almost exclusively on unique component controls.

CONCLUSION AND RECOMMENDATIONS

At the start of this analysis, several questions were asked. How can acquisition agencies and industry do better? How can IG vendors do better? How can CIGI do better? What does it mean to be "CIGI compliant"? The data in this paper points to answers for these questions, but also asks more. It is clear that the main purpose of CIGI is being achieved, at least in part. But it is also clear that there is great room for improvement and only cooperation between all stakeholders will drive the CIGI initiative closer to its full purpose.

Acquisition agencies have been very slow to adopt and require the use of the CIGI Version 4.0 – published in 2014 and the only version backed by a standards organization – and the answer cannot be to simply declare a requirement to use the CIGI standard. This paper shows that the standard alone is not enough. Acquisition agencies must engage industry and address the issue of CIGI compliance directly. CIGI Version 4.0 strives to define testable requirements for all functionality and STO/NATO MSG-118 made strides to implement a testing framework.

These efforts must be picked up and driven to completion. The CIGI standard provides an interface framework and does not levy a requirement on the image generator to support any particular functionality. Can or should acquisition agencies take the standard one step further and require a contract-specific set of CIGI messages to be supported? The STO/NATO MSG-118 testing tools (once completed) could then be used to validate an IG's "compliance" with the required set of CIGI functionality.

Confounding these efforts are the continued use of IG-specific database and moving model formats, as well as CIGI's provisions for unique component controls and user defined messages. As long as disparate IGs are implementing moving model controls according to their own propriety formats, moving from one IG to another will always carry significant cost. Acquisition agencies can mitigate this cost by moving towards requirements for truly standardized database formats.

The SISO CIGI PSG has made strides toward mitigating the cost of integrating IG-specific packets by providing an avenue to register extension packets. The goal is for IG vendors to identify functionality not present in the standard and, when the community sees value in the functionality, drive a conversation to find a way include a standardized interface for the functionality. The potential benefits to this paradigm are made clear in this paper's analysis – when specific functionality is standardized by CIGI, the recurring cost to integrate and deploy that functionality decreases dramatically over time. Unfortunately, uptake of the CIGI v4.0 standard has been low and this avenue has not been heavily used.

Image generator vendors are the lynchpin of this entire effort. They must take an active role in the further development of the CIGI standard. If there are barriers to implementing the standard interface in its current form, the IG vendors must identify and help remove those barriers. If critical functionality is missing, they must point it out and help to define a standard interface. In all cases, the IG vendors should strive to avoid the use of non-standard packets and component controls, because their use only contributes to integration cost growth as shown in this paper.

At the time of this writing, the SISO CIGI PSG is actively working on the next iteration of CIGI. All stakeholders – acquisition agencies, IG vendors, and industry – must participate in this conversation to ensure the future version of the CIGI standard is widely adopted and steps closer to the most basic goals of the standard. The title of this paper asks whether the Common Image Generator Interface really is common. The answer is "yes", but only to a point. There is much that is still non-standard about a typical CIGI implementation, but it does not have to remain that way. Common effort and collaboration between all stakeholders will put CIGI on the path to true commonality.

REFERENCES

CIGI PDG. (2014). *SISO-STD-013-2014, Standard for Common Image Generator Interface, Version 4.0.* Simulation Interoperability Standards Organization, Inc.

Lechner, R., & Phelps, W. (2002). *CIGI, A Common Image Generator Interface.* Interservice/Industry Training, Simulation and Education Conference (I/ITSEC).

STO/NATO. (2018). *Development of Common Image Generator Interface (CIGI) V4.0 Compliancy Testing Tools Final Report.* NATO Science and Technology Organization.