Reusable Automated Platform SIL Testing
A Cost-Effective Risk-Reduced Process for Airworthy Reusable Software

Stephen Simi
TES Program Manager

Steve Koester
TES Software Engineer

Richard Zepeda
TES Software Engineer

Scott Tompkins
US Army SED-Airworthiness

Tucson Embedded Systems, Inc., Tucson Arizona
StephenS@TucsonEmbedded.com

Abstract

Current and projected program requirements are exceeding Department of Defense (DoD) budget and schedule constraints. This applies to the Army's requirements to integrate common avionics equipment onto dissimilar aircraft – both manned and unmanned. As such, innovative approaches and new acquisition business practices are needed to reduce platform integration costs and speed the fielding of important war-fighting capabilities. The Common Software Initiative (CSI) was formed by the US Army's Product Manager of Aviation Mission Equipment (AME) to explore solutions to these problems.

In support of CSI, Tucson Embedded Systems (TES) developed and is applying automated testing capabilities and performing the verification of an AME Alt-Comms reusable software product. TES acts as a third-party platform integrator testing the product prior to it being released to the Platforms. The integrator's environment and automated testing capability supports the development and test phases and promotes the evaluation of embedded control software across a fleet of multiple dissimilar platforms prior to formal release. TES has developed a cost-effective risk-reduced automated test environment to support the development and integration of reusable aviation software for the US Army Aviation Systems.

The planned goal is 100% reuse of automated testing software and testing artifacts, such that one piece of test software and accompanying artifacts may be certified once and reused across multiple platforms as described in the FAA Advisory Circular AC 20-148 [1]. The reuse and automation will reduce costly and time-consuming platform System Integration Laboratories (SILs) testing and will support the formal qualification testing (FQT) efforts of the software. With automated reusable testing, TES and PM-AME estimates a reduction of more than 70% time and more than 50% cost of integration (potentially 57%) when compared to current business practices. This would allow the DoD to field two to six additional capability sets for the same budget as one.

While first applied and used to certify PM-AME's reusable radio control software targeted for PM-Cargo's CH-47F and PM-Kiowa Warrior OH-58, the reusable automated testing capability can be applied to all avionics capabilities including communications, navigational, sensors, actuators, etc.

Introduction

The Army's Assistant Secretary of the Army (Acquisition, Logistics, and Technology) is spearheading efforts [2] for "rapid equipping," "rapid fielding," and transforming the Army's acquisition processes. In response to this call, the Army's Product Manager, Aviation Mission Equipment (PM–AME), is seeking to implement a process by which common software products, to include common avionics integration software, can be identified, acquired, tested, and integrated across the disparate Army Aviation platforms. The PM-AME has identified the need for this process through the Common Software Initiative (CSI).

Implementation of the CSI would position AME into conformance with the acquisition strategy outlined in Chapter 2 of the Defense Acquisition Guidebook [3] and with the directives of AR 70-1 Army Acquisition Policy [4]. These two DoD documents outline prescribed requirements for standardization, commonality, and systematic reusability that will guide Army Aviation practices for improving budget-to-capability performance.

In support of CSI, PM-AME acquired a reusable Alt-Comms software product targeted for the PM-Cargo's CH-47F and

Presented at the American Helicopter Society 66th Annual Forum, Phoenix, AZ, May 11-13, 2010.

PM-Kiowa Warrior OH-58 platforms. This product controls the AN/ARC-201D and AN/ARC-231 radios and will be in service through 2020, until replaced by Joint Tactical Radio System (JTRS).

PM-AME also contracted Tucson Embedded Systems (TES) to act as a third-party platform integrator testing the reusable product prior to it being released to these Platforms. TES has developed a cost-effective risk-reduced automated test environment to support the development and integration of reusable aviation software (see Figure 1).

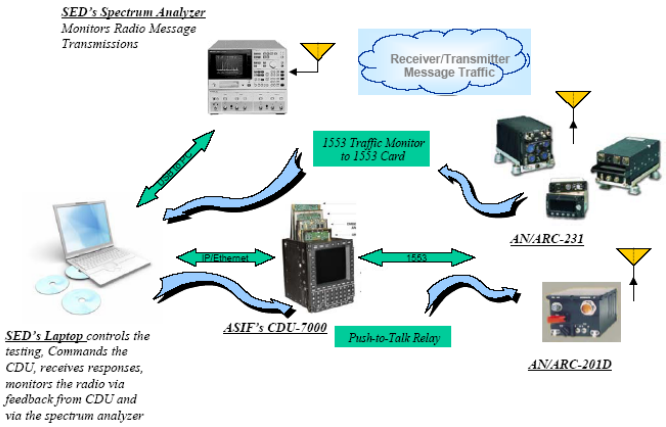
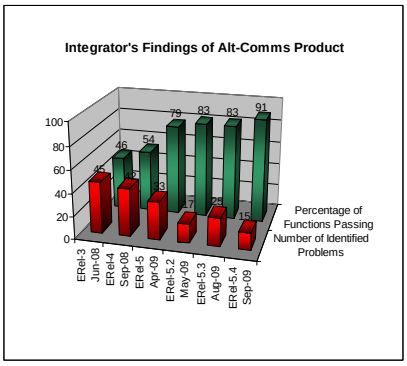


Figure 1. Integrator's Environment

Using a combination of actual target hardware (control display units, and military radios) and automated testing capabilities, an engineer can read in a software release and execute a set of test scripts to verify the control and functional operations of these Alt-Comm radio capabilities. There are 170 functions operating: Ground, HaveQuick, Maritime, SATCOM, SINCGARS, UHF LOS, and VHF ATC, and VHF FM capabilities.

The test results were used in the development phase, quickly identifying operational issues to the software developer earlier in the life-cycle. TES worked closely with the developer improving the capabilities with each release and ensuring the product (software and integrator's documentation) is prepared for Platform integration efforts.



TES also worked with Army representatives from the Aviation & Mission Research, Development, and Engineering Center (AMRDEC) Aviation Engineering Directorate (AED) and the Software Engineering Directorate (SED) to confirm that the results were suitable as supporting artifacts for Airworthiness Qualification Substantiation Records as defined in AR 70-62 [5].

What AME and TES jointly discovered was a potential for a tremendous time and cost savings for the Army, which also, through reuse, represented a tremendous risk reduction to the program.

TES modeled the process described in FAA Advisory Circular AC 20-148 [1] (see Figure 2) as it would be applied to the SED-AED certification process of the Army's rotary fleet, assuming that software tests and testing artifacts could be reused across multiple platforms and support Reusable Software Component (RSC) testing as well as platform integration testing.

The process, aligned with FAA's AC 20-148, implies third-party developers could produce airworthy Reusable Software Components (RSC) and software Reusable Software Verification Components (RVC) which meet DO-178B guidelines [6], build and execute system-level tests at a government-owned Aviation Systems Integration Facility (ASIF), and then with a high-level of confidence rebuild the RSC and RVC on platform-specific SILs and re-run the RVC saving both time and money. On completion, the components then proceed to flight-testing.

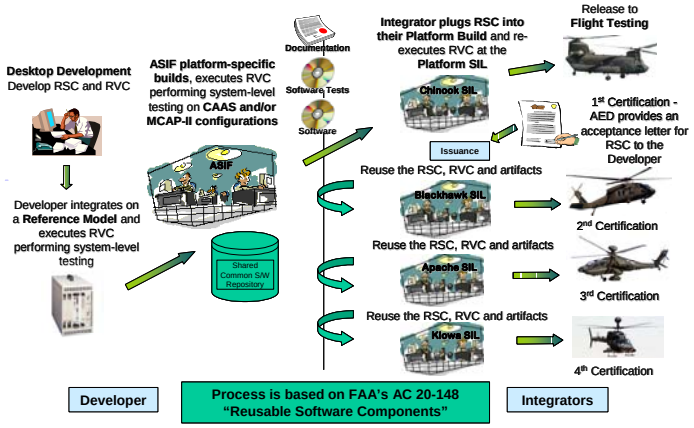


Figure 2. Vision for Reusable Automated SIL Testing

Through the process, an airworthiness qualification is achieved and an acceptance letter of the RSC and its reusable artifacts are presented back to the Developer. The RSC, RVC, and supporting documentation are subsequently reused for integration on other platform SILs, etc.

AED and SED identified that the automation can be used in system-level testing in flight representative Platform System Integration Laboratories (SILs). The key was to have TES engineers work with the Platforms and define the insertion points for this automation, such that it did not "contaminate"

the Platform SIL environment. To instill confidence in the results of the automated testing, operators performed selected functions by hand and those results were compared against the automated results.

Potential Cost Avoidance – 57%

To quantifying the time and cost savings, current business practices were compared to those supplemented with as much automation that would be acceptable to the certifying authority. TES and PM-AME found that time, costs, and risks could be reduced. Interestingly enough, the largest saving came from the ability to auto-generate and reuse life-cycle-testing artifacts.

The estimated cost to test one new capability integrated onto a platform was approximately $225,000. By using automated testing that cost could be reduced to $187,000, a 17% reduction. That savings grew with each reuse of the testing artifacts.

When TES and PM-AME quantified the cost to develop the corresponding certification artifacts, and compared that against the cost of the automation, auto-generating testing artifacts and reusing those artifacts, subtracted out the cost of development, and projected the costs and cost avoidance forward for three other platforms; a potential savings of 51% was found.

The cost of verification and validation for the integration of one Alt-Comms capability set onto four platforms including airworthiness substantiation artifacts is approximated to be $3.1 Million to PM-AME with current business practices. Whereas the cost, if reuse and automated testing were applied would be $1.3 Million, avoiding $1.8 Million, or allowing PM-AME to fund and integrate two additional non-automated up to six additional automated capability sets with the same budget by using automation and reuse. This could become the "CSI new acquisition business practice."

Considering the timesaving realized with this new business practice, more capabilities could be funded and fielded to the war-fighter faster thereby addressing the DoD budget and schedule constraints and improving budget-to-capability performance.

The planned goal is 100% reuse of automated testing software and testing artifacts, such that one piece of test software and accompanying artifacts may be certified once and reused across multiple platforms as described in the FAA Advisory Circular AC 20-148 [1]. The reuse and automation will reduce costly and time-consuming Platform SILs testing and support the software formal qualification testing (FQT) efforts. This paper describes those efforts.

Background

The Army has an ongoing need to integrate Aviation Mission Equipment products into aviation platforms. This integration can occur at aircraft delivery or as an aircraft upgrade. The integration cycle includes a significant effort in developing software to interface to new and changing AME products and certifying those products onto the Army's fleet of Aviation Platforms.

Each platform prime contractor is responsible for developing the software to interface with new aviation equipment. Historically, equipment was introduced as mission-specific, and added as non-integrated ("strap-on") equipment into their respective platforms.

Today's aviation mission equipment is highly integrated into the platform and moreover the same equipment is integrated within different platforms.

This arrangement has lead to ad hoc development and stovepipe systems resulting in duplication of effort across the aviation platforms for integrating common aviation equipment. It has also resulted in duplication of efforts within an aviation platform when integrating a new piece of aviation equipment that has similar functional capabilities to already integrated equipment. Duplication of testing across multiple aviation platforms is a significant cost factor of integration and fielding costs.

The result is that current and projected program requirements are exceeding budget and schedule constraints. To address these issues, both technological and process solutions must be developed within the Aviation community. Technological solutions must be based on the integration of functional capabilities across aircraft, and process solutions are needed to accommodate cross-platform integration and certification requirements.

What follows is an introduction and description of how AME's first reusable Alt-Comms application program interface (API) software was put into a process that allowed for the software to be rapidly verified, certified, and reused across the aviation fleet. Verification results were obtained quickly, and when used in conjunction with the development phase of the software, the results identified and assisted in the resolution of software operational issues early in the development phase.

The Platform integrator's environment and how TES developed and used automated testing capabilities for the rapid verification of each software release is described below in Figure 3. A description of the Platform integrator's environment, the process used to parse the developer's feature set, auto-generate tests scripts, and rapidly perform verification on LRU operations follows.

AME's Alt-Comms Integrator's Environment

The integrator's environment consists of a combination of actual military hardware and test environment hardware and software. Collectively, the combination is used to rapidly verify line replaceable unit (LRU) operations using AME's Alt-Comms reusable API software. Illustrated below is the hardware used in the Army's Alt-Comms integrator's environment, which is located in Tucson Arizona (not pictured is the ViaSat DOCCT/S used to simulate SATCOM waveform operations).

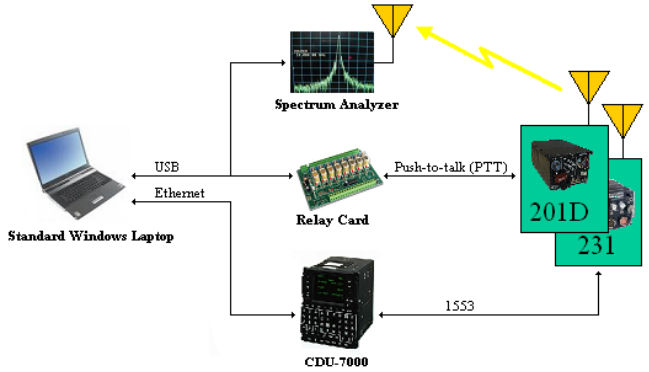


Figure 3. Platform Integrator's Environment - Hardware

AME's reusable Alt-Comms API is targeted for PM-Cargo's CH-47F. As such, the integrator's environment includes the same essential hardware used in the production environment--the AN/ARC-201D and AN/ARC-231 radios, as well as the CDU-7000. In order to provide several enhanced capabilities for test writers, however, the environment has been expanded with additional hardware devices. A relay card that simulates the Push-To-Talk (PPT) switch of the radio microphones, a ViaSat DOCCT/S unit used for SATCOM simulation, and a spectrum analyzer that can verify radio transmissions and frequency hopping all have been added to support the automation process. Additionally, a laptop computer, primarily used to run test scripts over TCP/IP in conjunction with the CDU-7000, has been equipped with a 1553 card that monitors the bus traffic supporting low-level debugging capabilities.

There are three primary components of software in the test environment: 1) a set of Java-based programs for converting provided APIs (C++ header files, for example) to project-specific testing code, 2) a suite[2] of platform-independent C++ libraries and tools for building the testing environment and wrapping devices (both hardware and software) in common code modules, and 3) an application for running user test scripts. Not all of the components are necessary for all projects, but when combined and utilized together, they provide a robust and complete package that implements many of the testing tasks seen in the embedded systems industry.

The first component, the CDA Tool plugin, is a set of importers/exporters and a visual interface for viewing and editing the API information. The importers take the APIs in its raw format, and extract the key information into hierarchal columns of data, which includes everything from function names and parameters, to module documentation, and tracing tags. From these lists, which the lab test engineer can navigate, update, and archive to a standard XML format, a set of project-specific source files are created. In general, the exporters are not sufficient to meet all the test code and test script generation requirements of the project, but they can easily be extended through typical programming inheritance to cover a large range of applications. TES modified its product line CDA Tool to import the AME Alt-Comms API.

Secondly, the test engineer utilizes a set of C++ core libraries (CDA Core). The libraries are designed for operating environment portability and work on a multitude of platforms and real-time operating systems (i.e., Integrity, LynxOS, VxWorks, Linux, and Windows). Usually the provided functionality is invoked within the source code generated by the exporters, but the libraries can be also utilized independently for additional processing or within wrappers around existing tools or hardware devices.

The main advantage of using the CDA Core is to provide a straightforward mechanism for getting disparate devices to communicate within the same testing environment. Also in addition, the libraries contain several well-tested communication protocols (i.e., Sockets, TCP, 1553, RS-232) that conform to a standard interface, making the ability to swap between protocols as simple as changing a parameter in one line of code.

The third and final software component utilized in the integrator's test environment is a script-running application called the Programmable Control Test System (PCTS). Often times the device under test has no means to store or process scripts, and, in the case of a software API compiled for several targets, is not inherently bound to any particular hardware. PCTS addresses this problem by providing a safe, platform-independent separation to the devices. Multiple devices, communicating using different protocols, can be accessed concurrently through one of two common scripting languages, REXX and Python. Additionally, PCTS provides several common test harness features—qualified verification, results logging, and report generation.

The next section describes the process of generating the test environment and test scripts for each AME's Alt-Comms API software release.

[2] The TES' CDA, Patent Pending, Core C++ libraries and PCTS testing application were pulled unmodified from TES' product line. The CDA Tool was modified and tailored for this effort.

Process of Auto-Generated Testing

One benefit of the integrator's environment is the ability to use the CDA tools to auto-generate integrated code and test scripts for the test environment, which are then used to verify the functional operations of the system very efficiently.

The TES' developed PCTS and CDA tools are product line products. Using these tools, an engineer can read in an Alt-Comms software release and auto-generate test scripts and the underlining symbol tables and software required to auto-test the reusable software component on the target hardware.

Results have been impressive. Within a few hours of receiving a new software release, an engineer can begin the verification of the AME Alt-Comms API software operating on actual target hardware (CDU-7000) and controlling actual military radios (AN/ARC-231 and AN/ARC-201D). This process, if performed without automation, typically takes several weeks or months to accomplish. Using the integrator's environment and the auto-generated automated testing suite has greatly reduced this time and allowed further enhancement of test capabilities.

The testing process has four distinct actions as is illustrated in Figure 4 and as described below.

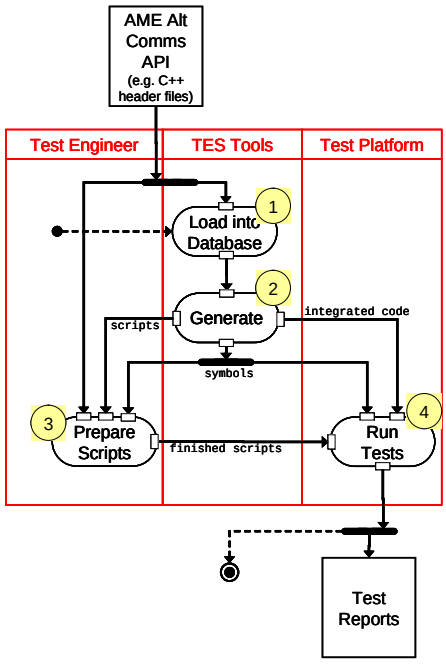


Figure 4. The 4-Step Process of Auto-Generation

1) Import the Reusable API into Database. TES tools are used to import the AME Alt-Comms API from its C++ header files. The CDA Importer parses the API headers files and populates a database.

2) Auto-Generate the Test Environment. Using the populated database, TES tools then generate integrated code, symbol tables, and test scripts. The integrated code produced is platform independent and will run on any target platform supported by the reusable software component. The integrated code when combined and used in conjunction with CDA core libraries creates a test environment prepared to test the API on target platforms. The symbol tables provide a mapping to invoke the reusable API by the PCTS script engine. PCTS is capable of communicating with the target through various protocols, in this case through TCP/IP.

3) Prepare/Finish Test Scripts. Typically preparing the test scripts is the most time consuming process outside of the actual testing. The testing of 170 Alt-Comms tests typically required several weeks to more than a month to complete. TES was able to automate this process by separating the scripting logic from the data. This allows the logic to be reused on subsequent releases and the data portion to be generated from the tool and the database. Currently generated scripts are being *diffed* and compared with scripts that were generated from the previous API revision to quickly identify revision modifications.

4) Perform Automated Testing - Running Batch Test Scripts. The scripts are then batched together, run, and test results are recorded for software verification and validation purposes. Because TES has automated the development of the test environment, regression tests can be performed the very day a new revision of the AME Alt Comms software is made available. A typical testing cycle utilizing reusable test automation still requires results analysis after conclusion of the text executing. Including this step, the total test cycle for the PM-AME reusable Alt-Coms API can be completed in a week.

Overall, with this process set of revised testing methodology, the time to verify a software release has been reduced from several weeks or more than a month down to a week, *i.e.*, 5+ weeks down to 1-week.

Future Process Enhancements to Test Robustness. The FAA's DO-178B software verification processes [6] specifies objectives for robustness testing. The amount of robustness testing varies in degrees depending on the certification level of the product. For software robustness testing, parameter inputs are to be tested outside the normal range to demonstrate that the software does not result in unpredictable outputs or failures.

To support the automation of this verification process objective, TES is currently designing and developing ways to specify more descriptive header files so that parameter boundary information can be incorporated into the database. This will in-turn support the auto-generation of more extensive test scripts to include boundary condition testing

(i.e., five tests scripts per parameter). The goal is to extract sufficient information from the header files and auto-generate tests scripts that would support the verification of DO-178B's robustness testing objective.

The rapid verification process was designed to work in conjunction with the Developer's design and development phase so that coding issues can be identified early in the process. If the API is to be reused as intended, most of the test suite is reusable simply by injecting a different CDA core library.

The combination of auto-generation, auto-testing, and reusing testing artifacts is producing tremendous savings in integration time and costs, and with reuse and early identification of issues, it is also reducing program risk.

Verification Activities

Verification activities are the development, documentation, and execution of specific methods to specific portions of the product or product development work products. The methods may be by review, testing, analysis, simulation, mock-up, black box, white box, formal methods, or other reasonable technique or methodology deemed suitable.

Additionally, the verification methods chosen are reviewed. Every verification result is reviewed to discover any errors that need to be fixed, and those problems discovered are identified and dealt with in a documented process. All information (method, procedure, results, and problems) is documented and traceable back to the requirement and specification documents.

Software verification process objectives are satisfied through a combination of reviews, analyses, and the development and execution of test cases and procedures. Reviews and analyses provide an assessment of the accuracy, completeness, and verifiability of the software requirements, software architecture, and source code. The development of test cases and procedures may provide further assessment of the internal consistency and completeness of the requirements. The execution of the test procedures provides a demonstration of compliance with the requirements.

Verifications Support for Army Airworthiness Certifications

The Aviation and Missile Research, Development and Engineering Center (AMRDEC) Aviation Engineering Directorate (AED) assesses and develops the flight airworthiness qualification requirements for each AME product. The AMRDEC Software Engineering Directorate (SED) evaluates software life-cycle artifacts and assesses platform software airworthiness on behalf of AMRDEC AED.

A program's airworthiness requirements are documented in an Airworthiness Qualification Plan (AQP), which is generated by the AED with input from the SED. All system life-cycle phases are addressed during the airworthiness evaluation process; however, the verification phase will be addressed here in more detail.

A verification plan must be developed early in the project to affect thorough product verification. AED and SED must be consulted during the development and review of the plan. The primary goal of a verification plan in this context is to plan for compliance with the verification requirements contained in the AQP. The plan should include the specific test activities, process, artifacts, and program milestones that will be completed with regard to verification. AED and SED will review the verification work products and serve as an independent test witnesses to ensure the processes were followed and the results are acceptable.

It is essential to the success of an automated testing approach for the automated testing to be incorporated into the verification plan. By planning for test automation early in the product lifecycle supporting processes can be put in place. Planning for automated testing can enable the adoption of coding standards and design standards that support automated methods. For example, the use of descriptive header files which specify boundary values can be required to support the automated test script generation process for robustness testing. The verification plan encompasses both software and hardware implementations. Verifying activities depend on:

- Required level of process rigor
- Type of product, product constituents
- Development environment
- Concept of Operations / Concept of Employment
- Target platform requirements for integration

The primary way in which reusable automated testing would fit into the verification of AME products is by generating a test environment which utilizes platform independent test scripts. These test scripts would be traceable to the system requirements and provide complete (100%) coverage of the system specification. SED has evaluated the approach presented by TES and shown that given a reasonable set of assumptions, it is possible and indeed profitable to meet the airworthiness verification goals utilizing reusable automated testing methodologies.

Verification is unique to each product, and evaluating the completeness of a verification plan requires understanding of all these factors. Analysis must be performed on each product or product integration to verify that the assumptions used during the generation of the reusable verification components are not violated. The Verification Checklist [7]

provides general areas of verification activities. A combination of government and developer resources must consult the verification plan to determine which activities are appropriate for a specific requirement. Executing the verification plan will require a significant allocation of resources. This reinforces the need to develop the plan early in the project to manage schedule and resource needs. Test automation and software reuse was combined on this effort as a means to reduce verification time and cost.

Verification Audits and RVC Qualification

A process that is integral to a software verification plan is the Verification Audit. Audits typically cover a minimum of 15% of the test cases and procedures. Of the total, 5% should be end-to-end audits starting with requirements tracing and continue through the process to the results. All activity around the audit items should be checked, which includes problem reporting and problem resolution. The intent is to assure the documented processes are correct, appropriate, and are followed.

Verification Audits also assist the certifying authority in providing assurance in the correctness and completeness of the automated testing performed by the RVC. To instill confidence in the results of the automated testing, operators performed selected functions by hand and those results were compared against the automated results.

A RVC itself would ideally be developed and qualified in accordance with DO-178B as a verification tool. Qualification is required if a tool automates a software verification process activity. The benefit of tool qualification is that the tool has the pedigree to fully or partially automate the testing of aviation software without requiring additional verification processes.

Automated Unit and System-Level SIL Testing

As identified, testing is categorized into Unit-White Box, CSC-Black Box, Integration SIL, and Flight Test.

SED, after review of the automated test scripts, identified that it would accept the results for Unit/Functional-White Box testing, with a one-to-one mapping of test script to requirement. That is, the Alt-Comms (AN/ARC-231 and AN/ARC-201D) Common Avionics Architecture System (CAAS) partitioned requirements are being met using the automated test scripts.

AME Engineering Life Cycle Management

PM-AME manages its product lifecycle in accordance with *Systems Engineering Process and Procedures for Life-Cycle Management Command Acquisition* [7], which includes Requirements Management, Change Control Board, and storage of all life-cycle and verification artifacts within an engineering repository.

AME Engineering Repository

Verification artifacts including plans, procedures, test cases, scripts, and reports are configuration controlled and managed. They provide a clear and comprehensive history of verification for certification analysis and future development and maintenance activities.

The AME Engineering Repository [8] is a controlled library of AME product and project documents, references, standards, procedures, manuals, etc. The Repository provides control and access to the information about products, projects, activities, and external information used in AME business.

The AME Engineering Repository is intended for products approved for reuse by the certification authority. *AME Common Software Initiative Reuse of Common Software* [8] identifies what artifacts are controlled, how they are managed, and applications of the AC-20-148 [1] as applied to PM-AME reusable products.

Test Readiness and Formal Qualification Tests

Typically two formal events are used to qualify the readiness of a system, TRR and FQT. The intent of the Test Readiness Review (TRR) is to determine that the product or system's requirements, design implementation, test cases and procedures are complete and have reached a state of maturity to perform Formal Qualification Test (FQT). Additionally, TRR ensures that the FQT environment is ready for a successful FQT. One test readiness review is conducted for each major configuration item [typically identified as a Computer Software Configuration Item (CSCI) or Hardware Configuration Item (HWCI)]. This review looks at test plans and procedures, Computer Software Unit (CSU) and Computer Software Component (CSC) test results, and informal CSCI testing to verify that the completed CSCI is ready for formal testing and approval. The TRR occurs after all change proposals are addressed for the product's baseline.

Typically the product developer supports two dry-run test events and one formal FQT. TES and AME are interested in comparing the cost and time of current business practices against the cost and time of automated testing. This testing assumption formed the basis for investment in the "CSI new acquisition business practice" utilizing RVC methodology.

Cost Estimations and Potential Savings with Automated Platform SIL Testing and Documentation Reuse

In order to show the business case for automation and reuse, TES and AME qualified and quantified the activities of TRRs and FQTs, and compared the results.

The projected cost avoidance would allow PM-AME to fund and integrate two to eight additional capabilities using automation business practices with the same budget. Considering the timesaving with this new business practice, PM-AME could fund and field more capabilities to the war-fighter faster thereby addressing the DoD budget and schedule constraints and improving budget-to-capability performance.

The assumptions of the cost estimates are illustrated in the following tables.

Current Business Practice	Reusable Automated Scripting	Table of Assumptions
2	2	Num. Dry Run(s) conducted
1	1	Platform SIL Software FQT
12	8	FQT labor hours per day
2	0.5	FQT labor weeks low-end
3	1	FQT labor weeks high-end
2	1	Government Resources supporting FQT
3	2	Full-Time Engineer (FTE) supporting FQT
$ 80	$ 80	assumed salary Documentation (hourly rate)
$ 100	$ 100	assumed salary Engineer (hourly rate)

The Platform FQT time estimates and potential savings using automation are quantified in the following table.

Current Business Practice	Reusable Automated Scripting	Time Investment and Potential Savings per Platform FQT
1800	180	FQT hours time investment low-end
2700	360	FQT hours time investment high-end
		Additional Time to Develop Reusable Automated Test Scripts
	1280	hours time investment low-end - 16 weeks - 2 FTE
	1920	hours time investment high-end - 24 weeks - 2 FTE
		Total FQT Time Investments and Potential Savings
1800	1460	FQT hours time investment low-end
2700	2280	FQT hours time investment high-end
		Cost Investments and Potential Savings
$ 180,000	$ 146,000	cost investment low-end
$ 270,000	$ 228,000	cost investment high-end
		Automation Time-Labor Return on Investments
	$ 34,000	First Platfom Cost Avoidance - low-end
	$ 42,000	First Platform Cost Avoidance - high-end
	$ 162,000	Per Platform Reuse Cost Avoidance - low-end
	$ 234,000	Per Platform Reuse Cost Avoidance - high-end
	19%	low end cost avoidance (Single Platform)
	16%	high end cost avoidance (Single Platform)
		Projected Savings for Multiple Platforms
3	3	4 FQTs, assuming reuse of automation on 3 Platforms
7200	2000	total hours time investment low-end
10800	3360	total hours time investment high-end
$ 720,000	$ 200,000	total cost investment low-end
$ 1,080,000	$ 336,000	total cost investment high-end
$ 900,000	$ 268,000	Average Cost - Cost Potential with automation
	72%	low end savings with automated testing and reuse (3 Platforms)
	69%	high end savings with automated testing and reuse (3 Platforms)
	70.6%	**Average Cost Avoidance**

The airworthiness documentation required to support FQT and the potential reuse savings are quantified in the following table.

Current Business Practice	Reusable Documentation	Documentation Required for Airworthiness	Reusable
document labor - units: weeks			
3	3	Software Requirements Specification	Yes
4	4	Software Design Description	Yes
1	1	Software Version Description	Maybe
3	3	Software Product Description	Maybe
3	3	Software Test Plan	Yes
6	6	Software Test Description	Yes
3	3	Software Test Report	No
2	2	Requirements Verification Matrix (Tracability)	Yes
2	2	Software Problems / Change Reports	No
2	2	Safety Assement Report (software)	No
2	2	Statement Coverage Analysis Testing (Path Testing)	Yes
4	4	Software Verification Cases and Procedures (SVCP)	Yes
7000	7000	hours for documentation per platform	
	4800	can reuse 24 of 35 weeks per platform (69%)	
$ 560,000	$ 560,000	cost for documentation per platform	
4	reuse	**multiple platforms**	
$ 2,240,000	$ 1,088,000	total documentation costs	
	51.4%	reuse documentation cost avoidance	

The combined cost avoidance of multiple Platform FQT using automations and documentation reuse are quantified in the following table.

Current Business Practice	Reusable Automated Scripting	Combined Cost Avoidance with Automation and Documentation Reuse
		Projected Total Cost Avoidance
	$ 900,000	Total Cost Current Method for 4 SW FQTs
	$ 268,000	Total Cost Automated Method for 4 SW FQTs
	$ 632,000	**total cost avoidance**
	$ 3,140,000	Total Cost Current Method for 4 SW FQTs + Documentation
	$ 1,356,000	Total Cost Automated Method for 4 SW FQTs + Documentation reuse
	$ 1,784,000	**total cost avoidance including documentation**
	57%	**Projected Cost Avoidance**
	2.0	additional BAU capability sets for the same budget as one set
	6.7	additional automated capability sets for the same budget as one set
		* not assuming auto-generation of documentation

Summary

Two reuse concepts have been combined and are being used by PM-AME to improve budget-to-capability performance. The first is the use of common software, and the second is the use of automated reusable testing. Combined they have the ability to reduce program risk and field more capabilities to the war-fighter faster.

In the Government, the challenges for the adoption of usable common software are two-fold. First is the acceptance of new acquisition practices. The new acquisition practices must be crafted to allow the product developer intellectual property protection, while allowing the PM-AME the management and control of all life-cycle work products. Developers will have to provide either their product under either Government Purpose Rights or Unlimited Rights. Contract deliverables and sustainment contracts will have to be carefully crafted to protect the interest of the developer and the investment of the government.

Second is the resistance to breaking down the system of "stovepipes." This will reduce direct integration funding to Platforms and platform integrators to fund common cross-platform solutions, and there will be resistance to adoption of reuse and automation until confidence in the "plug-and-play" software and the reproducibility of results is established and schedule reductions are realized within these Programs.

In an arena where war-fighting capabilities are a big business advantage, moving toward more open systems and sharing software and testing artifacts among Programs versus stovepipes will be a challenge for both the Platforms and their industry partners. With any change is resistance, especially when funding lines are shared across programs.

In order for the U.S. Military to maintain its readiness and war fighting advantage, it will require top-level support before Government and industry will embrace the implementation and conformance with the acquisition strategy outlined in Chapter 2 of the Defense Acquisition Guidebook [3] and with the directives of AR 70-1 Army Acquisition Policy [4]. These DoD documents outline requirements for standardization, commonality, and systematic reusability that will guide Army Aviation practices for improving budget-to-capability performance.

PM-AME and SED invested in efforts of software reuse and reusable automated testing, and they have shown that the U.S. Military can achieve these Defense Acquisition objectives, maintain its readiness and war fighting advantage by embracing the combination of reuse and automation.

For additional information on Tucson Embedded Systems' common software products and automated testing capabilities for airworthiness certifications (commercial and military), visit www.TucsonEmbedded.com.

Lessons Learned

Three areas for future growth and research were identified during this effort.

- First is the need to formalize more descriptive header files used in reusable API control code. These descriptions must be sufficient to specify parameter boundary values to support the automated generation of test scripts that can be used for FAA's DO-178B's robustness testing [6].

- Second is the need to instill confidence in the results of the automated testing. The reusable verification component (RVC) itself would ideally be developed and qualified in accordance with DO-178B as a verification tool. Qualification is required if a tool automates a software verification process activity. The benefit of tool qualification is that the tool has the pedigree to fully or partially automate the testing of aviation software without requiring additional verification processes.

- Third is the need to accommodate additional communication protocols, e.g., SNMP etc, so that future waveforms and radios can be incorporated into the integrator's environment, ensuring that the environment remains viable well into the future.

To reach the goal of 100% reuse of automated testing software and testing artifacts, such that one piece of test software and accompanying artifacts may be certified once and reused across multiple platforms as described in the FAA Advisory Circular AC 20-148 [1], these three areas of growth should be further investigated.

PM-AME is a proponent of software reuse within Army Aviation, and will continue to support its implementation into the Army Acquisition process. Software reuse and automation will reduce costly and time-consuming Platform SILs testing and support the software formal qualification testing efforts.

Conclusions

The PM-AME funded TES, SED, and AED efforts to create a Platform integrator's environment and develop automated testing capabilities to perform rapid verification of PM-AME's Alt-Comms common software on Army LRUs and artifacts. This has proved to be a cost-effective risk reduction to the program.

While architectures exist that can claim software reuse, few, if any, can claim software reuse for safety critical airworthy applications and also include reusable automation to support the Airworthiness Qualification efforts.

TES took PM-AME's Alt-Comms common software product and is verifying the control of two tactical radios integrated on two disparate Aviation platforms, Chinook and Kiowa Warrior, using one automated test suite.

We have found that when combining test automation and reuse, PM-AME and TES estimates a reduction of more than 70% time and more than 50% cost of integration (potentially 57%) when compared to current business practices. This would allow the DoD to field two to six additional capability sets for the same budget as one.

The knowledge and experience gained from this effort has advanced the methods of common software development, airworthiness qualifications, and clarified a vision that will further the implementation of the Army's Common Software Initiative.

The long-term vision for AME should include an outline of AME Best Business Practices [7, 8, 9, 10, 11, 12] for not just communications, but for all of the AME Functional Areas (Communications, Mission Planning, Interoperability, and Navigation) using the CSI and reuse concepts as they evolve.

For additional information about software reuse and automated testing capabilities, contact PM-AME or Tucson Embedded Systems, Inc. Mr. Stephen Simi, TES–Army Program Manager at 520.575-7283x154.

References

[1] "Advisory Circular AC 20-148 – Reusable Software Components," US Department of Transportation, Federal Aviation Administration, December 2004.

[2] Claude M. Bolton Jr. Assistant Secretary of the Army (Acquisition, Logistics, and Technology), "Talks to Defense AT&L," Defense AT&L, November-December 2004.

[3] "Defense Acquisition Guidebook, Chapter 2–Defense Acquisition Program Goals and Strategy," 20 December 2004.

[4] "Army Acquisition Policy, AR 70-1," 16 January 2006.

[5] "Airworthiness Qualification of U.S. Army Aircraft Systems, AR 70-62," Research, Development, Acquisition HQ Department of Army, 21 May 2007

[6] RTCA DO-178B, "Software Considerations in Airborne Systems and Equipment Certification," RTCA, Inc., 1140 Connecticut Avenue, Northwest, Suite 1020, Washington, D.C., 1 December 1992

[7] "PM Aviation Systems - Aviation Mission Equipment, Systems Engineering Process and Procedures for Life-Cycle Management Command Acquisition," Tucson Embedded Systems, Inc., February 2007.

[8] "AME Common Software Initiative Reuse of Common Software for US Army Aviation Missile Command," PM-AME, Tucson Embedded Systems, Inc 30 August 2006.

[9] "Software Product Lines – Practices and Patterns, CMMI, and CMMI-AM," Carnegie Mellon Software Engineering Institute, March 2004.

[10] "Commonality Working Group Common Software Demonstration, Lessons Learned," Tucson Embedded Systems, Inc., 20 July 2006

[11] "Supporting the Common Software Initiative, Capability Driven Architecture – Radio Control, Reusable Software Component, Integrator's User Guide," Tucson Embedded Systems, Inc., 7 August 2006.

[12] "The ASIF Standard Reference Model (ASRM)–The Development Environment that will Enable Common Software Development for Army Aviation Aircraft," Tucson Embedded Systems, Inc. 9 February 2007