

Lessons Learned Using COTS in Real-Time Embedded Systems

**Presented to the Joint Avionics and Weapons Systems Support
Software and Simulation Conference**

June 1998

**Robert Rosenberg
Sabre Systems, Inc.
Pine Hill Technology Park
48015 Pine Hill Run Road, Building C
Lexington Park, MD 20653**

19981029 105

Lessons Learned Using COTS in Real-Time Embedded Systems

Presenter: Robert Rosenberg, Sabre Systems, Inc.

Track: SDT1

It is certainly not news that the use of commercially-available-off-the-shelf (COTS) hardware and software is revolutionizing the acquisition process for new real-time embedded systems. Designs using COTS hardware and software can be the only approach to meeting today's aggressive schedule and cost budgets, while lowering production risks. Using Open Systems is an attractive approach to inexpensively incorporate increased performance as technology improves. It is hard to imagine a program manager who would not want to try to take advantage of these benefits.

DOD is not only encouraging, but also mandating the use of COTS hardware and software in the development of new and replacement (sub)systems to meet current budget constraints.

However, the use of COTS must be carefully managed or the results of these benefits will disappear into a life cycle maintenance nightmare. There are several lessons that have been learned about the use of COTS that should be considered before planning a new program.

1.0 Introduction

Historically there were sound reasons not to use COTS. Today technology has evolved and many of the historic problems have been overcome.

The use of Commercial-Off the Shelf (COTS) hardware and software in real-time systems is not new. I have been involved in developments using COTS in real-time systems since 1983 and I was hardly a pioneer. The use of COTS in real-time systems in the past was limited by technical considerations, operational suitability issues, and life cycle concerns. The lower cost of COTS was always attractive, but for most programs these limitations made its use impractical.

The major performance hurdle to using COTS in Command and Control (C2) real-time applications was always that COTS hardware and its shrink-wrapped operating systems were not designed to handle the quantity of interrupts that C2 systems need to process in a timely fashion. As technology developed, it was possible to use intelligent

controller cards to solve this technical issue. There were only a handful of vendors who started to design their products for the real-time market. Today there is a wider range of hardware and operating system products available from a growing number of vendors whose systems handle frequent interrupts in a fashion that will support demanding real-time requirements.

In the past, COTS was designed to operate in a temperature and humidity controlled environment. That made it unsuitable for many real-time applications, which were required to operate reliably in more extreme environments. Today many hardware manufacturers are designing their equipment to be more tolerant of environmental conditions with improved reliability.

One of the largest management reasons for not using COTS was the lack of control of parts availability. If a program manager (PM) was willing to pay to operate an assembly line, he could guaranty the continued availability of spare parts. When it was necessary to close a line, that PM could purchase a life time supply of spares before closing the line. When using COTS, programs could try to obtain a commitment from the manufacturer to provide notice before making a part unavailable, but I can recall more than one case when such an agreement was in place and the vendor either failed to honor it or forgot about it. Parts obsolescence issues remain one of the largest challenges in the use of COTS.

Another reason that COTS was not used in the past was economic. Programs did not want to incur the expense to re-host and maintain standard software that was already hosted in a MIL-SPEC computer. Sometimes, it was less expensive to purchase an identical MIL-SPEC computer to re-host that function. The C2P hosted on an UYK-43 processor that decoded Link 11 and Link 16 messages was a good example of a function that did not make economic sense to re-host, several years ago. Software reuse is still a sound approach to saving money and is one alternative that should still be considered when designing new systems.

Sometimes it made little sense to re-host software on COTS if a program already had a design tightly coupled to an existing processor. If that processor became obsolete and if its replacement had a compatibility mode, it became very inexpensive to re-host on the replacement MIL-SPEC computer. Many UYK-7 applications were re-hosted on UYK-43s for that reason. Today's faster processors require less tightly coupled designs to meet performance requirements.

Sometimes the use of COTS in the past was not considered for other reasons. For example, Congress mandated the use of the UYS-2 common signal processor on several programs. Other times it was prudent for a program manager to use a "MIL-standard" computer to widen the logistics base for those processors and take advantage of their existing logistics infrastructure. Top level policy has changed and the use of "MIL-

standard" computers is not required or encouraged for new applications. The concept of controlling logistics costs is still valid and can be applied to COTS systems.

Since most of these reasons are no longer valid, the use of COTS is being driven by an effort to substantially reduce development cost and reduce the time required to develop new systems by leveraging off of the extremely large commercial base of COTS hardware and software development tools.

This paper presents lessons derived from my experience related to the use of COTS and how it can be successfully used in real-time systems. It is presented primarily from a government program manager's point of view, but provides information useful to others involved in developing real-time systems. It only addresses acquisition initiatives, policies, development techniques, and practices to the extent that they are uniquely applied to COTS procurement and development. The references are not the source of the information presented; they have been included to provide the reader with additional background on the many topics discussed.

2.0 Methods Used to Reduce the Risk of Initial COTS Developments

The development of new systems using COTS can be successful if the proper approach is taken.

To utilize COTS hardware and software, changes are required to the way that real-time systems have been traditionally developed. These changes affect the entire range of the project and include both the buyer and developer. To achieve an accelerated low cost development, it is necessary to modify the approach to system acquisition from system specification through design, testing, acceptance and maintenance.

Successful COTS developments require a new mind set for both the buyer and the systems integrator¹. In the past the buyer defined requirements and then ensured that the developer properly designed, built and tested a system to meet those requirements. The developer closely held many of the design tradeoffs and cost issues (in fixed price contracts). For COTS developments, roles need to be redefined. For systems that consist entirely of COTS hardware, the role of the developer is closer to that of a "system integrator". For systems that consist of a mixture of COTS and custom hardware, the roles need to be appropriately tailored.

Buyers and system integrators need to work closely to consider the cost and technical tradeoffs. Many system developers have a traditional business base developing custom hardware. Those organizations need to overcome relying on those skills and need to become more adept at doing market surveys and evaluating existing products. A

successful COTS development will define the developer's role in many respects as a system integrator providing services. Product responsibility should be defined as a joint buyer and system integrator responsibility.

It is critical that some of the traditional boundaries between buyer and developer be reformed into a closer team approach. One example of this type of team currently in use in many projects is the Integrated Product/Process Team (IPT). By working together, the buyer and developer can work the technical tradeoffs to obtain the best affordable technical solution. However, to make IPTs work effectively, both buyer and developer need to modify their methods of system development. Initial training in the planned team organization can be held separately prior to contract award. After source selection, additional training should be held jointly to help form an effective team. Periodically during project execution, each IPT should take some time out from the daily issues to evaluate how well the team is operating and how it might improve.¹¹

There is always resistance to using new methods. Most of us tend to rely on methods that have been successful for us in the past. To overcome this natural tendency, there needs to be incentives for the buyer and developer to cooperate. Proper use of Award Fee and Incentive Fees usually provide a substantial incentive for developers. One aspect of each award fee period should be an evaluation of each IPT's operation as a team. Spot awards are an effective way to incentivize members of the buyer's team to reward teamwork.

Lessons Learned:

- 1. Customize your acquisition approach. Make it clear to all parties that it is a team effort. Make it to everyone's benefit to work as a team.**
- 2. Define ways to reward those team members who find innovative cost effective solutions.**

2.1 Modifying the Approach to Defining Requirements

A flexible approach to defining system requirements is necessary to take full advantage of COTS in a system's development.

Traditional system developments have defined acceptance criteria and then have provided those requirements to one or more vendors to propose the "best" design to meet the requirements. Cost and schedule were usually a key factor in determining the "best" value. However, this approach was flawed since sometimes a small change in a requirement value could cause a large increase in cost.

This is especially true when using COTS. A small change in a requirement can eliminate all available products. A good example of this is operating temperature. Many COTS vendors design and qualify their products to 50 degrees C. Many traditional real-time systems required operating temperatures to 55 degrees C. Since many requirements define a standard of suitability, it is necessary to consider whether that standard is absolute or whether a minor variation would be acceptable.

There are many other examples of requirements that are quantified to establish acceptance criteria, but the specific value is not absolute. Examples of these items are Reliability (MTBF), Fault detection, Fault isolation etc. These values are specified because we want to buy systems that are reliable and maintainable. However, one of those values can be adjusted slightly to achieve significant cost savings.

There are several ways to avoid inadvertently driving cost. One of the most frequently used methods for reducing costs is the use of draft Request for Proposals (RFPs) and systems specifications. This process is very good for avoiding the most obvious problems early in the development cycle, but usually does not identify all of the issues. Other cost drivers may be identified later as the developer's design matures and he is able to identify other factors that reduce the size of the candidate hardware domain.

The use of Alpha Contracting is also a good way of early identification of cost driving requirementsⁱⁱⁱ. Alpha Contracting is a process where the buyer and system integrator work together to write the statement of work and specification to define a product that meets the buyer's needs without inadvertently adding cost drivers.

An innovative way for dealing with cost driving requirements is the use of placeholder requirements. A fair competition can be held since all vendors are required to bid to the placeholder requirements' value. After the contract is awarded, the developer is required to study each placeholder value to determine whether small reductions in any of the values can result in cost savings. The buyer can pool savings to deal with contract changes and the developer will eagerly participate if the identification of these savings results in incentive fees. My experience has been that very few of the requirements actually change. After considering all of the candidates, both buyer and system integrator subsequently agreed that only a few changes to the original placeholder requirements were desirable.

Another important way to keep requirements achievable using COTS is to avoid imposing requirements at too low a level. Why should you impose board level requirements at all? Requirements for performance at the board level should be allocated based on operational need. For example, the operational need is that a system may need to operate anywhere in the world within 30 minutes after the application of power. Let the vendor allocate the specific temperature and humidity requirements to the boards; the

ambient temperature and humidity might be controllable. Low level requirements too often prohibit the system integrator from finding other preferable solutions.

To successfully use top level requirements, the IPT needs to develop a close working relationship when deriving the lower level requirements. This is especially important for User System Interface (USI) functions. Prototyping has always proven to be an effective technique to allow users to see what the system will do and allow them to contribute to the fine-tuning of the detailed requirements. In the past, it was prudent to abandon the software developed for prototypes after the requirements were established. However, now there are COTS tools that allow the reuse of prototype Graphics User Interface (GUI) code that defines the appearance of a screen into a real-time application.

This whole concept of evaluating requirements and schedule in relation to their costs is embraced in another current DOD initiative called Cost-as-an-Independent-Variable (CAIV)^{iv}. Encourage team members to suggest adjustments to requirements that reduce system development, operation or maintenance costs. This encouragement needs to be explicit, such as incentive fee awards for the developer or spot awards for members of the buyer's team. Public recognition of these awards is also a great motivator.

Many COTS boards manufactured can meet most of the traditional requirements, but most vendors will not go to the expense to formally qualify the boards to demonstrate that they do.

To continue with the 50 vs. 55 degree example, one possible impact might be that the environment would need some conditioning (cooling) before operation for a very small percentage of days in the hottest places in the world. The buyer and systems integrator need to decide whether avoiding a short delay for cooling before the system is operational is worth the additional cost and time to develop a custom board. Since many board vendors are very conservative in the testing and rating of their boards, another option is for the team to consider testing the boards at the higher temperature to determine whether it will operate at the required temperature.

Lessons Learned:

- 1. Define requirements at as high a level as possible. Try to keep requirements operationally oriented.**
- 2. Work with the vendor to identify cost driving requirements. Do not be afraid to reconsider requirements that force a custom board and adjust those requirements for an affordable solution.**
- 3. Use placeholder requirements to conduct fair and open competition for those items that may be inadvertently large cost drivers.**

2.2 COTS System Design

The design methodology for COTS is different than for custom developed components.

The requirements for traditional MIL-SPEC design and its reviews evolved over many years for the review of custom built hardware and software. Today's acquisition initiatives promote a much more flexible approach to systems development. COTS system developments need to take advantage of that flexibility. Systems that use COTS require some modification to the traditional processes. The general approach for Preliminary Design should be "What is the top level architecture and what are its candidate parts" and for Detailed Design: "What parts should we buy?"

In general this makes the design process leading up to a Preliminary Design Review (PDR) an industry survey process which may also identify CAIV issues (as previously discussed). The preliminary design process may revert to a more traditional approach, if it is determined that there are no acceptable COTS products to meet key operational requirements.

This proposed philosophy necessitates a rework to the traditional design time line for systems using COTS items. In general, some of the activities needed to answer those questions were traditionally done later in the process.

Stressing system requirements should be identified. The top-level system architecture should isolate the cause of the stress from the remainder of the system. For example, a device that requires frequent interrupts or very time acknowledgement of interrupts can be serviced by a separate microprocessor to protect the remaining system functions from the demands of a single device.

By forming teams consisting of hardware engineers, systems engineers and software engineers, requirements can be allocated without creating insurmountable hurdles for other activities. The allocation of requirements to the components can be an iterative one. As the evaluation proceeds, reallocation may be prudent if it defines a wider range of components. The entire system should be modeled to validate the top-level architecture. This process may identify cost driving requirements that should be worked within the framework established for that project.

Any placeholder requirements should be defined by the completion of the preliminary design phase, unless there is a very specific reason to continue a tradeoff study during the detailed design.

Lessons Learned:

- 1. Multi-disciplinary teams and modeling are important tools in allocating requirements.**

2.2.1 Software Preliminary Design

The more requirements that COTS software can fulfill, the less custom software will need to be developed. COTS software makes sense if it substantially reduces the amount of custom software that needs to be developed. Without these potential savings, it is better to develop software and maintain control of the source and data rights.^v

More than technical factors need to be considered when selecting COTS software candidates. The financial stability of the vendor, the vendors ability and history of solving user problems and their willingness to establish a long term support arrangement are also key qualifying factors. This long-term support is necessary to correct problems that are discovered later in the system's life cycle. Some developers have made arrangements to obtain the data rights for each product's source code, if a product's developer abandons the product or goes bankrupt.

For COTS software packages, generally there is not enough data available from the vendor in sufficient detail to determine whether the product can meet the basic need. The vendor data can be augmented with data collected from current users; but given the frequency of change of many software products, most user data available will be for older releases and will be of limited value. This requires the system integrator to obtain single copies of each product to be considered. Using the actual software, an evaluation can be conducted to determine how applicable the package is to the requirements. This initial evaluation should also consider gross compatibility issues with other software packages being considered for other requirements and some code generated from the software development environment (at least the compiler) that will be used. If the software development environment is also being selected, that selection process increases the combinations of possibilities to be evaluated.

One scenario for this phase could be a project that wants to try to select a COTS operating system, a COTS data manager, and a COTS Graphics User Interface (GUI). The range of candidate COTS operating systems could be narrowed by the real-time response requirements of the system. Then only those data managers and GUIs compatible with at least one of the remaining candidate operating systems need be considered further. The remaining candidates can be incorporated into strawman systems to evaluate how well these packages meet the requirement and whether they can work together.

Obviously, this phase can be streamlined if the system integrator can identify bundles of COTS software that already have been integrated by other projects. Any previously integrated bundles can be tested against project requirements. Less time can be spent evaluating the compatibility of those bundles. Even if a bundle has been used successfully for many years in another project, the compatibility issue is not totally resolved. Hidden compatibility issues or disqualifying bugs may still exist if the features your project needs are not the same features previously used.

Clearly, the quicker the range of candidates can be reduced; the fewer combinations need to be considered. While this task imposes a large effort during the preliminary design phase, it completes some tasks traditionally performed much later during software component integration and test phase. This "early" software integration helps to reduce development risk by identifying integration problems early.

Award fee during this phase should be partially based on the system integrator's success at identifying candidates that will minimize development cost and risk.

Lessons Learned:

- 1. Select only COTS software candidates that will result in substantial savings in the amount of software that will need to be developed.**
- 2. Balance any potential savings against the risks and loss of control that will result in using COTS software.**
- 3. Consider more than just technical issues when selecting candidates.**

2.2.2 Hardware Preliminary Design

The preliminary design of COTS hardware is more of a paper exercise. The process for laying out the overall size, power, and cooling is fed from data collected from the individual card candidates balanced against the overall physical limitations. Developing the list of candidates requires surveying the market to identify potential candidate items and requesting detailed information on those items. The process of sizing the backplane, memory, storage, and processor speed does not vary much from custom design; however, sometimes the allocation of requirements between components may be affected by what is available on the market.

When conducting market surveys, there will be gaps in the data available to form a candidate list. Many times this initial survey activity will be frustrating since the compliance matrix will identify more questions than definitive answers. Even if a manufacturer claims compliance, the product may not meet the requirement. Marketing claims are often only true from a limited viewpoint. For example, a board may claim

100% fault detection capability, but detection data may reside only in internal registers that may not be accessible by application software. Another possibility is that the 100% claim may only include the board itself and not its interface to the chassis.

Another key point is that Non Development Item (NDI) hardware is not as mature as COTS. These items have been developed by vendors in an attempt to find a market for the item. Several programs have treated NDI similarly to COTS with poor results. NDI products have proven that too often only a single item was able to be hand built by engineers to perform a subset of requirements in an extremely controlled environment. Usually, NDI has no logistics support, has not yet considered manufacturing issues, and needs further development. If you become the only user of an NDI product, you will also need to assume all of its life-cycle support costs. The transition of NDI hardware for use is similar to the effort required to go from a successful demonstration and validation program to a full engineering development. NDI should only be considered as a strawman design approach when custom development is required.

The lack of suitable COTS candidates is not immediately a reason to go to custom development. The rate that COTS hardware is evolving is remarkable and the lack of candidates may be solved by waiting. At the rate that processor performance, storage, memory, and data transfer capabilities are increasing, the lowest risk (and cost) approach to requirement's shortfall may be to wait for the natural evolution of the required capability. This option should only be selected if the vendors are predicting the planned capability within an acceptable time frame and there is an acceptable backup approach if the planned capability does not become reality when needed.

Award fee during this phase should be partially based on the system integrator's success at identifying candidates that will meet requirements and minimize development cost and risk. It should also consider how open the architecture is and how it will affect life cycle maintenance costs.

Lessons Learned:

- 1. Continually follow up on data requests to try to get as much information as you need to assess a product's ability to meet the requirement. Try to get past the marketing department to the design team to get the answers that you need.**
- 1. If a product meets almost all of your needs and has no disqualifying restrictions, you may want to carry it on the candidate list for further consideration.**
- 2. If no products meet your needs, try to find out what planned products are available. Waiting for a planned product to mature may be preferable to embarking on a custom development.**
- 3. NDI is not COTS. If you want to (or need to) include NDI hardware, you need to plan the additional activities required to turn it into a product.**

2.2.3 Detailed Design

In order to complete the selection of hardware components and software packages, perform a detailed requirements evaluation of each candidate product. Configure those items that seem best suited to form a candidate architecture. Next, evaluate the compatibility of all the components in the candidate architecture. When those two steps are successfully completed, validate the real-time response of the resulting architecture. Plan on obtaining loaner hardware parts and evaluation copies of software packages. To complete these steps, plan on developing a mini hardware/software testbed to complete the following activities:

Requirements Evaluation. Data obtained when conducting the market survey is usually not accurate or complete. Plan on verifying key parts of the requirements with the actual hardware/software. One common example is products that claim to meet SCSI standards. While most products can typically handle data within the median range of SCSI standard limits, many products claiming to meet the standard will not operate properly if pushed to the edge of the standards (e.g. products that are close to the SCSI line length limits). Plan on evaluating each product to obtain missing data needed for the product evaluations. Most vendors do provide BIT, but do not have traditional MIL-SPEC fault detection or isolation data. Plan on evaluating and testing each potential vendor's BIT capabilities as part of the parts selection process. Plan on augmenting the BIT to meet the overall requirement.

Compatibility Evaluation. Selection of individual components to meet requirements is not enough. An important part of the design process is to ensure that all of the software and hardware components do not conflict. Plan on putting those parts together and executing COTS software and hardware together to confirm that the design is sound before purchasing components. Include software compiled using the planned development toolset, BIT software, drivers and utilities to confirm that they will all work together.

Real-Time Validation. Most COTS operating systems and utilities are not primarily designed for real-time applications. They may optimize performance for relatively small data sets using cache or memory, but may substantially slow down when operational loads are encountered. Plan on conducting performance tests with realistic operational data loads as part of the design process. Be prepared to alter hardware selection to accommodate COTS software limitations. A good example of this issue is that some versions of UNIX may provide an acceptable real-time operating system until Virtual Memory paging begins. By sizing memory large enough to keep all programs memory resident, the performance of UNIX is greatly enhanced to meet the real-time response requirements. Data collected on this testbed should be used to update system architecture models to project performance of the full system.

Life Cycle Costs. - Balance life cycle cost issues with technical concerns. Unit Production Costs, sparing costs, software licenses costs may offer different solutions to minimize cost for a single program phase (development, production, or over the system's life cycle). The system integrator should be required to show how his proposed approach affects each phase of system costs. It should consider the parts reliability, proposed maintenance plan and include both the spare and labor costs associated with repair. For example, parts that have a higher MTBF may be a better value than those that may fail more frequently, especially if those parts are time consuming to remove and replace.

Award fee for this period should be based on an evaluation of the costs and risks associated with the system integrator's proposed design solutions.

Lessons Learned:

- 1. Confirm the compatibility of selected components.**
- 2. Confirm that the candidate items meet the requirements. Do not rely on vendor claims if it is possible to confirm them.**
- 3. There may not be an exact fit. In some cases programs may want to conduct a Cost as an Independent Variable (CAIV) analysis to consider using the best available COTS product versus the development of a custom board.**
- 4. Most vendors do not have traditional MIL-SPEC reliability data. Plan to customize the reliability program for each vendor's inputs.**

2.3 Modifying what Should be Expected at Design Reviews

The criteria for design reviews must be modified to reflect the changes introduced into the design methodology.

Professionals in the acquisition of systems must evaluate whether the proposed design will meet the requirements, as well as, the technical, cost, and schedule risk associated with the selected approach. In the past, there were two key contractually defined formal design evaluations, PDR and Critical Design Review (CDR). The criteria for these reviews were defined in MIL-STD 1521, which has been canceled as part of the acquisition reform process. While the formal definition of these activities has been eliminated, these reviews are still important milestones in the top-level DOD management review process.

Especially for systems incorporating COTS hardware and software, it is important to plan what needs to be accomplished by each review. These reviews need to be tailored to the specifics of each program. The expectation of what needs to be accomplished at each review needs to be defined early in the development cycle. If these expectations are not clearly defined in detail, the data available may not support the mandated management evaluation or may result in a less than satisfactory review result. This is another item that should be arranged between buyer and system integrator as part of the IPT process. Each IPT should define the entry criteria, presentation topics, and exit criteria for each review. Since some parts of the systems may not be COTS, different topics may be required for the non-COTS items. Incremental reviews are a good way to manage the differences between COTS and non-COTS items. After the completion of all of the increments, a capstone review is an excellent approach to summarize the result of

all of the increments and to show how the increments come together to meet the overall need, and to evaluate risks at the system level.

When establishing criteria for the review of COTS items, it becomes obvious very quickly that many of the traditional MIL-STD 1521 review items are not applicable. For hardware, a key item on the CDR review was a measurement of how many drawings were completed. For software, one of the key items was the completion of detailed design documentation. Reviews of drawings and software detailed design documentation were meaningful when evaluating custom developments, but do not add to an evaluation of a COTS design activity.

The focus of the criteria for these reviews for COTS items should be tailored to answer the guidelines for the design previously recommended; Preliminary Design - "What are the candidate parts" and for Detailed Design: "What parts should we buy?"

These reviews should summarize the design and requirements tradeoff issues performed to answer the question relevant to the applicable design phase. The review should present operational performance design issues. It should also present an evaluation of how the candidate design will affect production and life cycle cost issues. Traditional detailed review items should be included to the extent that they support the spirit of the design phase guideline.

Some desired data may not be available from the COTS vendors, as previously discussed. This should not delay the design review. Instead, the system integrator should identify what key data is missing, the approach to collecting that data, and the backup approach if the data, when collected, identifies an issue.

Lessons Learned:

- 1. Tailor design review requirements to the specifics of each program and the guidelines for each design phase.**
- 2. Clearly define the expectations for the design reviews.**
- 3. Don't be afraid to proceed if all the desired data is not available. Define a plan to obtain any missing data and contingency arrangements if the data identifies an issue.**

2.4 After the Critical Design Review

After all of the COTS capabilities and limitations are well understood, the remaining aspects of the system's development can proceed on an accelerated pace with lower risk.

The major benefits for developments using COTS occurs after the completion of detailed design. At this point, a testbed of the key hardware and software items has already been assembled and most compatibility issues have been identified. Custom application development can proceed using the established baseline and the lessons learned from the detailed design testbed. There will be new issues identified as the development proceeds, but the number of integration issues will be substantially smaller than traditional custom hardware and software developments.

Another key benefit resulting from this approach is the ability to start environmental qualification testing much earlier. Moving this testing earlier in the schedule allows more options in dealing with any problems discovered.

Modifying COTS boards should not be considered as one of those options. It will void the part's warranty and will create the requirement for an ongoing production facility. Some system integrators who have traditionally been in the hardware development business are uneasy with the pure unmodified COTS approach and may design modifications (such as a unique interface or firmware) to help ensure their continued existence. Buyers should word specifications accordingly to avoid this problem and should be prepared to challenge any non-standard modifications.

Instead of modifying boards, there are often more attractive options. These include selecting another board, asking the board's vendor for the necessary modifications, or building custom enclosures to address environmental concerns. Proper enclosure design can mitigate many vibration, shock, and temperature issues. The resulting "custom" item does not create a difficult life cycle management problem.

The other post design activity affected by the use of COTS is the approach to software testing. In the past, many projects used a combination of black box and glass box testing. When using glass box, the tester takes advantage of his knowledge of the software coding. Black box testing is designed and conducted without any knowledge of the code. Traditionally, glass box testing was used by the software coder to test each piece. After all the pieces were successfully integrated, major builds were turned over to functional test teams who used black box testing to verify requirements. The lack of insight into the design and coding of any incorporated COTS software prohibits developers from using a glass box approach during software integration^{vi}.

Since the ultimate goal is to provide the end user with needed capabilities at an affordable price, the program manager should keep his team focused on the end result. Use a combination of a large final Award Fee/Incentive Fee increment to provide an added motivation for timely delivery of a quality system.

Lessons Learned:

- 1. Avoid modifying COTS hardware (other than non-intrusive additions such as board stiffeners or added software).**
- 2. Plan to environmentally qualify at least some boards that have not tested to the complete range of the operational requirements. Consider whether testing a single item or whether a complete screening of each part is required.**
- 3. Plan on dealing with the shortfalls of individual cards. Some of the items that may be required include custom enclosures to isolate vibration, additional fans, or heat sinks to help control temperature.**
- 4. Plan on Black Box approach to software integration testing instead of Glass Box testing.**

3.0 Methods to Help Control the Life Cycle Cost Impact

Controlling life cycle costs remains the largest challenge for systems using COTS.

COTS obsolescence is the most significant challenge to the successful deployment and operation of any real-time system. Since no program is large enough to ensure the continued production of a COTS item, each item's obsolescence must be assumed. The COTS vendors are driven by the substantially larger commercial market. As the vendors focus on current market capabilities, spares availability and product support will diminish, resulting in less control of hardware obsolescence. This is already a major issue for the programs that are using COTS.

3.1 Issues that Drive Life Cycle Cost

Minimizing the impact of future changes is a very effective method for reducing life cycle costs.

Both software and hardware issues can become major life cycle cost drivers. To help manage these issues, it is very important to know when changes to the product line

will occur and what impact the changes may have. Develop an ongoing relationship with each COTS vendor to keep abreast of changes in the product line or its support.

One of the most important COTS software products to any real-time system is the operating system. It can be very expensive to upgrade. The impact of any operating system changes must be minimized by using a highly layered architecture. Prohibit direct application interface to the operating system. If the layer interfacing to the operating system is properly designed, it can be modified to update all the interfaces to a new operating system. The use of layered architectures has always been an option, but until recently this option was not viable since it imposed a performance penalty. Today's much faster processors and substantially larger memories now allow this option to be exploited.

When using other COTS software, use the layered design philosophy to insulate that software from direct application calls^{vii}. If possible, try to find a way not to allow the COTS packages to interface directly with the operating systems. If the operating system needs to be replaced, it may also force an update or change to these COTS packages. Changes to the operating system will require a new evaluation of any COTS software previously incorporated into earlier versions of the systems. Changing the operating systems also imposes major cost and performance risks.

An example that illustrates this point comes from one early real-time embedded systems COTS development. UNIX operating on a SPARC 1 was selected as the operating environment. Performance requirements were demanding and the applications were optimized to maximize throughput. This system also utilized a data management package that was tightly coupled with the operating system. As the development proceeded, it became apparent that a faster computer was desired. Fortunately, the state of the art had progressed and faster processors were available and compatible with the rest of the hardware architecture. Unfortunately, the newer faster processors were not compatible with that version of UNIX. Upgrading to a new version of the operating system was expensive since its interface to application programs had changed. To upgrade, all the operating system interfaces needed to be identified and changed, the data management package needed to be replaced, and all the calls to the data management package also needed to be identified and replaced.

Operating system upgrades may have other unplanned side effects, including forcing compiler changes or driver changes for other system boards. Any upgrade to the operating system will also require a major regression test effort. Since the operating system or compiler changes are integral to the performance of a real-time system, an extended regression testing effort will be required to re-verify the system's functionality.^{viii} Safety and security critical items will need to be reexamined.

The other major life cycle cost issue is the availability of replacement boards. As boards need to be replaced, the original parts may no longer be available. Sometimes it is even difficult to determine whether the boards have changed, because many commercial vendors will change their boards without changing part numbers if the board's functionality did not change. With an open systems design, it was envisioned that the new boards would simply replace the older ones; however, this cannot be guaranteed and replacement boards needed to be re-qualified (environmental testing) and operationally tested in the target environment.

Some replacement boards come with new software drivers that are not compatible with older versions of the operating system. These drivers may or may not be compatible with the operating system version already in use. Unless a compatible driver is available, replacing a board could force an operating system upgrade. Even if functional equivalent replacement boards are available, it may be very expensive to environmentally re-qualify boards and operationally test systems as part of a normal maintenance cycle.

Some program managers have envisioned that the commercial board vendors would make provisions for the long-term support of their systems. Unfortunately, this has not happened. Instead, COTS vendors typically offer each product for a relatively short period of time (1-2 years). Traditionally, replacement parts for each product are only available until the supply is exhausted^{ix}.

- Lessons Learned:**
- 1. Try to select operating systems vendors with a history of backward compatibility. If the vendor continues that trend, the operating system upgrade should be less expensive.**
 - 2. Design systems to minimize the changes required due to hardware or operating system changes.**
 - 3. COTS compilers, software tools, and any COTS embedded software may or may not be compatible with systems upgrades. Utilize these items in a fashion that will minimize the impact of future changes.**
 - 4. When buying parts, require that every item is identical. Have QA verify that all parts meet that requirement.**
 - 5. Establish very close relationships with your COTS suppliers.**

3.2 Approaches to Manage Life Cycle Costs

Part of any COTS system development must be defining and planning an approach to life cycle maintenance.

Replacing each part of a system when spares are no longer available is usually not a sound approach to life cycle maintenance. There are two other approaches to better manage this problem.

The first approach is to buy enough spares for the planned life of the system. This approach can require a large initial investment in parts, but could be the lowest risk and lowest overall life cycle cost to the program. For programs that will field multiple copies of a system, this option ensures a single system configuration and reduces configuration management issues. There are a number of scenarios that make this stockpiling an unattractive option:

1. If program requirements later change and a basic system upgrade is required, then the initially purchased spares will probably be scrapped.
2. If the spares requirements are underestimated or if the planned system life is extended, there may not be enough spares to sustain operations.
3. If the number of units is large, available near term budgets may not support a large initial buy.
4. There may be shelf life issues.

The second approach consists of regularly planned systems upgrades (Planned Product Improvements (P3I)) throughout the systems life cycle. Only enough spares are stockpiled to sustain the system until the planned upgrade is complete, thus minimizing the up front cost while maintaining an acceptable risk. Spares for the out years can be budgeted to coincide with the P3I updates. This approach is recommended for most programs.

Evolving systems requirements can also be introduced at the planned upgrade. Addressing obsolescence issues simultaneously with new requirements enables the design to confirm sufficient resources to support the new requirements while renewing the availability of spare parts. This approach allows the often substantial cost of system testing to be shared between systems evolution and maintenance budgets.

Configuration management for the transition of fielded systems can be handled like any traditional obsolescence solution.

However, the second approach also has its drawbacks. Each P3I cycle can be very expensive. It may be necessary to replace all components – even if some of the

components remain functional, they may not be compatible with the newer components. For example, many of us remember buying 486 computers with VESA bus display controllers. When it was desirable to upgrade to Pentium processors, the VESA adapter (and other cards) also needed to be replaced.

Either of the options can be modified to further reduce initial cost by requiring suppliers to provide notification before they discontinue a product. The program manager can then decide whether to stockpile spares, develop an additional source for the item or initiate a P3I cycle. Experience has shown that this notification will not always be provided.

Some programs have even encountered obsolescence problems in the midst of their extended development cycles. If a program's development spans more than a couple of years, many parts may be obsolete before the completion of the operational test. Since most programs cannot purchase production parts before the successful completion of operational test, it may be illegal to purchase parts for production during EDM. The first system upgrade may be the transition from EDM to production!

Lessons Learned:

- 1. Select a life cycle approach as part of the system's development.**
- 2. Plan for Planned Product Improvement (P3I) programs at regular intervals to solve any compatibility problems created by forced operating system (or other) upgrades.**

4.0 Leveraging Off of Other Programs

The cost savings for COTS systems will be even greater when the real-time community takes a few more steps to take advantage of the common needs between systems.

Since the major reason for using COTS is to reduce cost, a few more methods for helping to control the cost of COTS developments are recommended. These cannot be accomplished by individual projects; rather they are concepts that will work when the entire community participates.

4.1 Sharing Data

Since the key to COTS system design is the evaluation of products, sharing data learned about the candidates with other programs will streamline design efforts.

The first of these cost savings concepts is the collection and sharing of detailed requirements data collected during the design phase. In that phase, the relevant characteristics of hardware components are evaluated using the actual hardware. COTS software is evaluated for requirements compliance and compatibility. The real cost savings are achieved when the data collected by these processes is captured and shared. As the systems development progresses, additional lessons are learned about the COTS and that information should also be captured and shared.

Many companies already capture a portion of this data. Most commonly captured are some of the "non-technical" evaluation factors about the vendors. These supplier databases are usually collected by the quality organizations and are used to identify "preferred" suppliers.

This concept should be expanded to include all technical evaluation information. If one program has gone to the time and expense of evaluating (or testing) any characteristic of a component, that information should be shared with others considering using that component. This is even more important for information learned about product incompatibilities and successful workarounds.

Many companies have not formally captured this type of information in the past because there was an informal technical network that effectively shared this kind of data. However, today with more and more projects using COTS and with the large number of mergers and reorganizations, these informal networks are losing their effectiveness.

This information, if collected, could be even more effective, if it could be shared outside of the organization collecting the data. DOD could facilitate this sharing. If each contract requires this information as a deliverable, it could be inexpensively posted to a public database accessible from the web. Users could benefit from a service that would automatically provide additional information when it becomes available for selected items.

Recently, I came across an example of how shared information could help after the design phase. A friend of mine told me about problems that he discovered when mating 5 row connectors to a VME 64 backplane. When I asked another colleague who was also using VME 64, I discovered that he had also invested time identifying and solving the same problem. This duplicate effort could have been avoided if a centralized information facility existed.

4.2 Centers of Expertise

Funding common technology evaluation centers to monitor the evolution of key technologies will provide information more quickly and less expensively than having each program study those items individually.

There is another incentive that can result in large DOD wide savings. I recommend that DOD fund centers of expertise to perform ongoing technical evaluations for key evolving COTS technologies of interest to many programs. These proposed activities could be successful if they work with interested programs to evaluate a particular technology's potential capability to meet specific program requirements.

A good example of this kind of activity occurred recently. The Naval Air Warfare Center performed a study of COTS 19" flat panel displays^x. Flat panel displays are attractive to many aircraft programs because those displays weigh a lot less, require less power and cooling, and require less space than current displays. This study evaluated the functional suitability of the available displays to perform several different applications. While this study was not centrally funded, and had a smaller scope, it did achieve many of the benefits of the proposed activities.

Similar studies would be helpful to many programs especially if the scope is expanded to include environmental and compatibility evaluations. These studies would be most helpful if they are funded as ongoing activities, so that the evaluation data remains current. New and upgraded products should be evaluated when available^{xi}. The core group performing these studies would be a very cost-effective avenue to evaluate a range of products for any unique requirements.

These types of studies should not be blindly used to mandate a single common item that all programs must use. Each program must consider the logistics as part of its component selection criteria. Using items already in the supply system does decrease cost and should be considered as a CAIV tradeoff issue.

4.3 A New Approach to Using Commonality to Control Life Cycle Costs

DOD has always recognized the savings of common subsystems, but many common programs have not been overwhelming successes. To overcome these past problems, more control of common systems must be retained by the applications.

Another major way that programs can leverage off of other programs is by developing common basic architectures. The more commonality that two programs have,

the more potential savings from common spares, shared development costs, and the potential for common obsolescence solutions.

There are well known reasons why common systems are not attractive to individual program managers:

1. Technical/Requirements - Common systems were force fit into applications where they did not provide a sound engineering approach and /or did not meet key operational requirements.
2. Schedule - The schedule for the development of common systems considered the user base, but frequently provided additional constraints for the users to incorporate into their program plans.
3. Funding - The budgets for the activities contributing to the development of the common system were subject to change. When the budget for one participant changed, it substantially changed the cost effectiveness for the other user(s).

All of these issues can be summed up to the primary aversion that program managers have towards common systems - Lack of Control. Any new approach to commonality must solve this control issue. It must incentivise the participants to find a way to make it work in a cost-effective manner.

One approach is to "encourage" each program manager to negotiate commonality with other programs. This encouragement could be as simple as only partially funding each program's requirements. Each program would then need to find creative ways to accomplish their needs. After arranging a potential partnership, a Memorandum of Agreement should be established between all parties defining what each party will provide to their joint effort.

Any partnering arrangement should be contingent on a successful design review approved by all parties. This design review needs to address a sound engineering approach showing how the operational and support needs of all parties are addressed. This design review should demonstrate also how the planned development would be integrated into the schedule of all applications.

To be successful, the commonality approach must include technical support from the system integrators of each application. To eliminate conflicts and keep competing organizations focused, use an independent government agent to direct the design activity and coordinate the tasking of each application's system integrator.

DOD should protect innovative program managers who are able to design cost effective common systems. Allocating independent funding for each common system will protect the participating program from any changes in budget allocations for the other

partners in the common system. Each program identifies the amount of funding that it is committed to provide and then that money is reallocated to a DOD level budget line item not available for reallocation without the consent of all parties. This type of protection is necessary to provide a reasonable measure of cost control. In the past, programs have not been able to reliably project the cost of common systems, since frequently their partners have either pulled out or delayed their participation. With today's lean budgets, it is especially important to mitigate this cost risk or PMs will refuse to participate in the development of any common systems.

5.0 Ongoing Trends

It is important for the real-time community to monitor ongoing trends and promote those that will facilitate the use of COTS.

Both hardware and software trends will affect real-time systems using COTS. The trends having the largest potential impact affect software portability. Since it is unlikely that any COTS hardware will be used for the entire life of many real-time systems, the ease or difficulty of re-hosting software will contribute significantly to life cycle costs.

Without evolving into a debate over the Ada requirement, the elimination of that requirement does leave each system with the possibility that the selected high order language (HOL) may face obsolescence. This is a major issue that could drastically affect the cost to re-host custom applications. The software support activity (either government or contractor) should develop a strong relationship with the compiler vendors actively planned support for the compiler and its supporting software environment.

The C4I community is trying to establish a rating system to evaluate the "openness" of real-time software architectures. This is a good concept and should be expanded into the definition of an "open systems standard" for software. I believe that this kind of standard could be very helpful in defining an approach to allow software packages to become as interchangeable as "open system" hardware boards.

Evolving hardware standards are also a potential life cycle cost driver. Many supposed Industry standards do not stay standard for long. Many real-time embedded systems have life cycles that will exceed the life of existing standards. One key example of this issue is the SCSI standard that has evolved through many names and revisions (SCSI, SCSI 2, fast SCSI, narrow SCSI, wide SCSI, and ultra wide SCSI). In open systems, the VME standard is evolving. VME 64 is already in use and is likely to replace the older slower standard very quickly. Industry needs to consider the impact of proposed standards changes as part of its change process.

6.0 Summary

Most of the historical reasons for avoiding COTS have been overcome by advances in technology. The remaining COTS concerns can be successfully addressed by modifications to the methods by which systems are developed and managed throughout their life cycle.

The systems development and its life cycle management are not independent. Part of the development must be the selection of a life cycle management approach. Each system's design needs to be evaluated for development costs and risk and also needs to tradeoff those concerns with life cycle cost and risk issues.

Although COTS can be effectively used today, more can be done to increase its benefits. Since DOD is committed to using COTS in real-time systems wherever possible, DOD should provide some basic common support services to facilitate the use of COTS. The real-time community also should take an active role in establishing methods that will promote this policy.

About the Author:

Mr. Rosenberg is currently an Operations Director at Sabre Systems, Inc. He manages Sabre's contributions to several Advanced Technology programs. Mr. Rosenberg has over 20 years experience developing real-time computer systems and its supporting software for a wide range of DOD programs providing expertise in software engineering, Software Independent Verification and Validation, systems engineering, acquisition management and life cycle management and planning. Mr. Rosenberg can be reached at buzzr@erols.com

-
- ⁱ The Commandments of COTS: Still in Search of the Promised Land, David J. Carney and Patricia A. Oberndorf, Crosstalk, The Journal of Defense Software Engineering, May 1997
 - ⁱⁱ Chapter 4.4 Using IPTs in Performance-Based Contracting, Writing Performance Based RFPs, Center of Acquisition Research Technology and Education (CARTE) Inc., December 1996
 - ⁱⁱⁱ Alpha Acquisition Presentation, "Reinventing Negotiations" Lessons Learned from the Lamps Block II EMD Contracting Effort, Unpublished Handout

-
- iv Cost As An Independent Variable (CAIV) Policy Guidance, Memorandum for Distribution, John H. Dalton, Office of the Secretary, Department of the Navy, 16 April 1998
 - v Chapter 13 Contracting for Success, Guidelines for Successful Acquisitions and Management of Software Intensive Systems: Weapons Systems, Command and Control Systems, Management Information Systems, June 1996, Volume 1, Department of the Air Force, Software Technology Support Center
 - vi A Software Development Process for COTS-Based Information System Infrastructure, Part II Lessons Learned, Greg Fox, Steven Marcom, and Karen Lantner, Crosstalk, The Journal of Defense Software Engineering, April 1998
 - vii An Architectural Approach to Building Systems from COTS Software Components, Dr. Mark Vigder and John Dean, Canadian National Research Council Report Number 40221
 - viii The Problems Imposed by Commercial-Off-the Shelf Tools in a Multi-Baseline Software Engineering Environment -- A Laboratory Manager's Perspective, Douglas Reichl and Linda Evans, Presented to the Joint Avionics and Weapons Systems Support Software and Simulation Conference, June 1997
 - ix Engineering Off The Shelf Solutions: A Life Cycle View, Doug Belaire and Mike Eagan, Naval Engineers Journal, May 1996
 - x The Liquid Crystal Display (LCD) Evaluation (A Performance Assessment by Military Users), NAWCADPAX-98-79-TM, 13 March 1998 (draft)
 - xi The Opportunities and Complexities of Applying Commercial-Off-the-Shelf Components, Lisa Brownsword, David Carney and Trica Oberndorf, Crosstalk, The Journal of Defense Software Engineering, April 1998

PLEASE CHECK THE APPROPRIATE BLOCK BELOW:

-AO# _____

_____ copies are being forwarded. Indicate whether Statement A, B, C, D, E, F, or X applies.

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

DISTRIBUTION STATEMENT B:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES ONLY; (Indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

DISTRIBUTION STATEMENT C:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND THEIR CONTRACTORS; (Indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

DISTRIBUTION STATEMENT D:
DISTRIBUTION AUTHORIZED TO DoD AND U.S. DoD CONTRACTORS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

DISTRIBUTION STATEMENT E:
DISTRIBUTION AUTHORIZED TO DoD COMPONENTS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

DISTRIBUTION STATEMENT F:
FURTHER DISSEMINATION ONLY AS DIRECTED BY (Indicate Controlling DoD Office and Date) or HIGHER DoD AUTHORITY.

DISTRIBUTION STATEMENT X:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND PRIVATE INDIVIDUALS OR ENTERPRISES ELIGIBLE TO OBTAIN EXPORT-CONTROLLED TECHNICAL DATA IN ACCORDANCE WITH DoD DIRECTIVE 5230.25. WITHHOLDING OF UNCLASSIFIED TECHNICAL DATA FROM PUBLIC DISCLOSURE. 6 Nov 1984 (Indicate date of determination). CONTROLLING DoD OFFICE IS (Indicate Controlling DoD Office).

This document was previously forwarded to DTIC on _____ (date) and the AD number is _____.

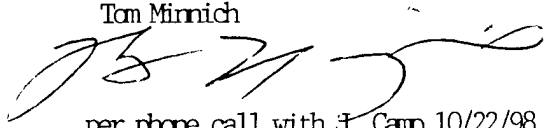
In accordance with provisions of DoD instructions, the document requested is not supplied because:

It will be published at a later date. (Enter approximate date, if known).

Other. (Give Reason)

DoD Directive 5230.24, "Distribution Statements on Technical Documents," 18 Mar 87, contains seven distribution statements, as described briefly above. Technical Documents must be assigned distribution statements.

Tom Mirnich



per phone call with J. Camp 10/22/98

Authorized Signature/Date

AFRL/FSD
2241 Avionic Circle
Wright -Patterson AFB, OH 45433-7518

John Camp
Print or Type Name

437 255-2164 3562
Telephone Number