Architectural Issues, other Lessons Learned in Component-Based Software Development

Will Tracz, Ph.D.
Lockheed Martin Federal Systems

Component-based software development, while reducing initial development time and effort, requires additional infrastructure and process support across the entire lifetime of an application. This paper summarizes lessons learned in dealing with commercial-off-the-shelf (COTS)-based architectures. In particular, it focuses on the technical and managerial issues associated with the acquisition, evaluation, selection, configuration, risk management, and evolution of software components.

"If the process hasn't changed, then the lesson wasn't learned."
—Ben Manning, Lockheed Martin Tactical Defense Systems

By using COTS components, software developers not only face increased opportunities to rapidly create new applications, but also face increased challenges to configure, integrate, and sustain these applications in the future. Arguably the technical key to success lies in the architecture that the system designers select for the components to be integrated into, as well as the middleware or "glue" that holds them together. Unfortunately, as proven by countless COTS-based project failures, project managers must take into consideration certain nontechnical factors, such as the volatility and flexibility of the requirements, the stability of the vendor, and the respective components that they supply. This paper describes some of the misconceptions that software developers and managers often have in planning for COTS-based systems [1]. First a series of myths are exposed in the light of reality of lessons learned from real world experience. This is followed by some rules of thumb for developing COTS-based systems.

This paper addresses COTS software issues, but the lessons learned equally apply to COTS hardware.

Operational Model

The COTS-based system acquisition and development model used by this paper assumes the following roles for the three major stakeholders:

1) The customer—who pays for the application to be built. Note that the customer also selects the developer/contractor/system integrator (e.g., through a competitive bid).
2) The developer—who builds the system out of COTS components. Note the developer may not always be the one to select the COTS components being integrated, as the customer may have stipulated them as part of the system requirements.
3) The COTS vendor—who supplies components to the developer and customer, along with its support and upgrades. It is important to point out that the customer needs to address the sustainment of the COTS-based system, or total ownership cost, because system maintenance can be adversely affected if certain architectural and procurement factors (e.g., licensing fees, security, and technology refresh) are not properly addressed early in the system life cycle upgrades.

The customer must address the sustainment of the COTS-based system or total cost of ownership (TCO) [2], because system maintenance can be adversely affected if certain architectural and procurement factors (e.g. licensing fees, security, and technology refresh) are not properly addressed early in the system life cycle.

Myths

This section contains a collection of COTS component lessons learned and is organized into two parts. The first subsection focuses on architectural issues. The second subsection focuses on management/procurement, or nontechnical issues.

Architectural Issues

The following myths deal with issues that the developer needs to consider when doing the initial trade-off analysis for a COTS-based system..

Myth 1: *It is important to know what COTS components can do for you.*
Reality: *It is important to know what COTS components can do to you.*

A system architect must always evaluate the build, buy, or modify tradeoff when determining how to meet the customer's requirements. COTS components, in general, provide certain functional capabilities at an extremely attractive *initial* cost. However, experience shows that functions come with limitations and implications. As the number of COTS components to be integrated increases, the dependencies and interplay among them becomes more complex, and can lead to intractable problems or difficult negotiations between different suppliers as to whose product is really at fault when things do go wrong. To complicate matters further, certain nonfunctional requirements, such as security, fault tolerance, or error handling, may not be uniformly supported by all components to the degree necessary to guarantee overall system performance. These potential hot spots typically form a list of risks that the architect must trade off in deciding the overall composition of the system under development.

Myth 2: *COTS-based systems can be designed top-down.*
Reality: *COTS-based systems are built bottom-up.*

COTS components facilitate a spiral development model in the sense that functionality can be quickly demonstrated in most

applications. In doing so, the customer benefits by early validation of requirements and the developer reduces risks by learning firsthand about the capabilities and configuration and integration difficulties associated with the components. Most architects understand this point and do not limit their choice of components by making design decisions too early. They recognize the need to remain flexible in trading off functionality across components until the components are fully proven and the integration mechanisms identified.

Myth 3: *An open-system architecture solves the COTS component interoperability problem.*
Reality: *There is no standard definition for "open system," and "plug and play" does not always work.*

Customers and developers clearly recognize the advantages of having plug-compatible components. They not only like having a choice of components, but knowing that if one component supplier goes out of business, there is another source for compatible components. It is debatable as to how successful open system initiatives have been (such as the Defense Information Infrastructure Common Operating Environment) [3]. Most will agree that when it works, it is great, but the number of plug-compatible components has yet to reach critical mass.

Myth 4: *You do not need to test COTS components.*
Reality: *You need to test COTS components more because you do not understand how they were built.*

It would be nice if all COTS components worked as advertised. But oftentimes there is a gap between what is advertised and what is delivered. Being that it is economically and oftentimes physically impossible for a COTS vendor to test all its products in combination with all other products under all operating conditions, subtle feature clashes can occur.

Furthermore, when developers are trying to leverage emerging technology, oftentimes marketing pressures force COTS vendors to deliver products with reduced capabilities along with the promise for increased functionality in future versions. Since the system integrator is usually responsible for the overall performance of the system, the system integrator should evaluate all components before they are selected for inclusion into the system.

Myth 5: *COTS products are selected based on extensive evaluation and analysis.*
Reality: *COTS products often are selected based on slick demos, web searches, or by reading trade journals.*

Because component-based architecture development is a relatively new field, systems integrators and customers still struggle with methods to keep abreast of technology advances and ways to determine which product best suits their needs. Oftentimes, in the rush to make a decision, the choice of COTS products is not made based on a strong business case or the total ownership cost. This problem has been around in various forms for a long time (e.g., GIGO [Garbage In, Garbage Out]) and ways of dealing with it (trade studies, test labs, independent product certification agencies) can be discriminators used by the customer in reducing risks associated with the acquisition of a COTS-based system.

Myth 6: *COTS components come with adequate documentation.*
Reality: *Features sell COTS components, not documentation.*

This myth may be thought of as a continuation of the previous two myths. The lack of documentation is a risk the system architect faces in determining the suitability of COTS components. In some instances the customer, upon being exposed to certain component features demonstrated in a certain (sometimes contrived) context, may place unnecessary or unrealistic constraints on the developer's implementation without adequate justification or flexibility in negotiating for different and possibly better components.

Myth 7: *You can configure a COTS-based system to meet your requirements.*
Reality: *You can configure your process to meet the COTS component capabilities*

The 80/20 rule applies to most COTS-based system efforts. The customer can satisfy 80 percent of its desired business process for 20 percent of the cost of a custom system (in 20 percent of the time). Most difficulties occur when a customer or developer thinks that the additional 20 percent is achievable at traditional software development costs. The cost of modifying COTS, or providing extra functions, is more difficult for the developer because it has little control or insight into how the COTS product was designed, documented, tested, or written/built. This information is usually proprietary and, in the light of upgrades and new versions, maintaining compatibility becomes a challenge. Most successful system integrators *never modify COTS*, and thoroughly understand the requirements (i.e., assuming an *all requirements are negotiable* approach). If the business case justifies modifying COTS components, then the developer should recommend that option.

Managerial Issues

The following myths deal with issues the customer considers when selecting a contractor to develop a COTS-based system.

Myth 8: *The processes COTS products support reflect industry best practices.*
Reality: *The process a COTS product supports often only reflects the market schedule and domain experience of the vendor.*

As much as COTS vendors would like the customers to believe that one size fits all, this is simply not the case (See Rule 12). Market considerations drive product offerings and most COTS component providers have product roll-out plans that include extended features and configuration parameters and hooks that allow tailoring and customization to support a better fit to best practices.

Myth 9: *You buy a COTS product.*
Reality: *You buy the right to use a version of a COTS product.*

COTS components provide immediate solutions at a fixed cost, but most applications have a life cycle that spans several releases of those components, which means that it is unrealistic (except in the case of hardware components) to expect the follow-on costs to be zero. In addition to the acquisition cost of

the components, the customer and developer need to explore the cost and level of support services as well as opportunities for commodity purchases.

Myth 10: *Vendors will fix problems in the current release of the product.*
Reality: *Vendors will fix problems in the next version of the product.*

As mentioned in the previous myth, the level of service one receives from the component supplier is negotiable. Unless the contract explicitly states it, the type of problem fixes one receives will be market driven (See Rule 6).

Myth 11: *If you are a large enough customer, you can influence COTS component suppliers.*
Reality: *The market influences COTS component suppliers.*

Again, the size of the current and future customer base drives the COTS component supplier in determining his response to user needs (See Rule 6).

Myth 12: *COTS-based systems are a panacea.*
Reality: *COTS components exacerbate inadequacies in the system development process by compressing the development schedule.*

To some, COTS components may seem like a silver bullet because they can provide faster, cheaper, and better solutions for:

- relatively simple applications.
- use in mature problem domains.
- using a small number of mature, unmodified components.
- proven integration mechanisms.

But not all applications fall into this category. The mere fact that applications are developed so quickly facilitates the possibility that the wrong application will be developed, the wrong COTS components initially selected, and the perceived short-term success will pave the way to long-term disaster.

Additional Myths

The following myths are relatively self-explanatory and reflect some of the points made in the rules of thumb stated in the next section, or previous myths.

Myth 13: *COTS components are free except for the purchase price.*
Reality: *COTS-based system sustainability issues overwhelm acquisition costs.*

Myth 14: *You can ignore vendor upgrades.*
Reality: *You lose support of back systems if you ignore vendor upgrades.*

Myth 15: *You can pay a vendor to modify COTS components to meet your requirements.*
Reality: *You can pay a subcontractor to modify COTS components to meet your requirements.*

Rules of Thumb

Rule 1: "*The cost of COTS is 1 percent of that of developed code.*"

This rule is attributable to Ed Feigenbaum of Stanford University and formerly the Air Force's Chief Scientist. To apply this rule, one would take the cost of a shrink-wrapped component and multiply it by 100 to get a rough approximation of the development cost for comparable function. Clearly there are other factors, such as the size of the customer base, for determining the cost of most COTS products, so one needs to use this rule judiciously.

Rule 2: "*The maximum shelf life of a COTS software component is two years.*"

This rule factors into determining the total ownership of an application in that all COTS components that have been configured and integrated together will probably have to be replaced two years after each was introduced into the marketplace. To complicate matters, each new version of a component might have additional dependencies and possibly introduce new, conflicting functionality. Also, the updates may not be released at the same time or validated with the same versions of other components, further complicating matters.

Rule 3: "*The half-life of COTS product expertise is six months.*"

This rule is attributable to Kurt Wallnau, Software Engineering Institute, who observed that with the fast-paced introduction of new product versions, as well as competing products, there is an unprecedented obsolescence associated with current technology. The inverse of this rule is that every six months you need to plan on evaluating a new version of a COTS product.

Rule 4: "*You need to evaluate COTS in an environment as close to the operational environment as is possible.*"

This lesson learned comes from too many bad experiences with COTS components that have been selected, designed around, and determined to have a pathological dependency that either completely precludes their incorporation, or makes the integration process much more complex and costly.

Rule 5: "*You can never make a 100 percent Diminishing Manufacturing Source-resident COTS-based solution.*"

Any commercial source of technology is outside the direct control of the customer. Consequently, for certain critical applications, system integrators and the customer must work together to take precautionary measures to ensure the sustainability of the application. These measures include paying a third party to store the design documentation and source code of the components or negotiating for the establishment of an open Application Program Interface in hopes of stimulating plug-compatible competition (i.e., a second source).

Additional Rules of Thumb

The following rules of thumb are self-explanatory. It is debatable which of the last two rules is more important, but it is clear that they play an important role in determining the success of any development effort.

Rule 6: *"The smaller the customer base, the higher the COTS cost and the better the service."*

Rule 7: *"The largest problem with COTS is its short life span."*

Rule 8: *"Stay away from the cutting-edge COTS products, unless it is the only way you can get the performance you need."*

Rule 9: *"By using COTS components you decrease development time and increase integration time."*

Rule 10: *"The selection of COTS components is a risk-mitigation or risk-creation situation."*

Rule 11: *"A COTS-based system may not be the cheapest solution."*

Rule 12: *"A COTS-based system will never completely or exactly satisfy a customer's need."*

Conclusion

Who is at fault for most COTS-based system failures? Is the customer to blame for expecting COTS to be a panacea? Are the developers to blame for not using good engineering judgement in identifying risks and opportunities to mitigate them? The customer must be flexible and must understand the short- and long-term tradeoffs with respect to certain COTS options. Current customer acquisition processes often force asking the wrong questions at the wrong time. In the case of government acquisitions in general, the customer neither has enough COTS-smart people nor has strong policy guidance.

But the customer should not take all the blame. Developers have been naive in trusting vendor vaporware claims and in underestimating the challenges of component configuration and interoperability. Fortunately, they are becoming savvier in their testing and integration capabilities. The ultimate solution, used by many of the leading system integrators, relies on setting up Integrated Product Teams consisting of the customer, the user, the developer, and the COTS suppliers. Such a forum often provides a venue where all stakeholders can better understand the requirements, their priorities, and the total ownership cost tradeoffs that are available with full insight on their short-term and long-term impact.

There are many COTS-based system development lessons learned. Unfortunately, the near-term trend seems to indicate that these lessons will be re-learned by many customers unless proper education, policy definition, and sharing of experience occur. Finally, it should be clear that a COTS-based system might not always be the best solution available. When all the factors are considered, a business case laid out, and a TCO study done, a custom implementation may be more cost-effective over the life of the project. ◆

About the Author

Dr. William Tracz is a senior software engineer for Lockheed Martin Federal Systems. Currently, he is lead architect for several COTS and Reuse Repository projects as well as the principle investigator on an internal independent research and development project focused on nonintrusive software integration mechanisms. Dr. Tracz also is an ad hoc member of the Air Force Scientific Advisory Board COTS panel, chairman of the Lockheed Martin Software Subcouncil Working Group on COTS Software and Reuse, as well as editor of ACM SIGSOFT Software Engineering Notes.

Lockheed Martin Federal Systems
Mail Drop 0210, 1801 State Route 17C
Owego, N.Y. 13827
Voice: 607-751-2169
Fax: 607-751-2169
E-mail: Will.Tracz@lmco.com
Internet: http://www.owego.com/~tracz

References

1. SEI, *COTS-Based Systems (CBS) Initiative*, available at http://www.sei.cmu.edu/cbs/index.html
2. Gartner Group, *Total Cost of Ownership: A New Tool for Controlling the Cost of IT*, http://www.info.edge.com/5509toc.htm
3. IPESO/DISA, *DII COE Defense Information Infrastructure Common Operating Environment*, http://dii-sw.ncr.disa.mil/coe/

New Guidebook Released on Systems Engineering Fundamentals

The Defense Systems Management College (DSMC) has released a new systems engineering guide, *Systems Engineering Fundamentals*, that it calls a more basic tutorial on systems engineering than DSMC's guidebooks released in 1982, 1986 and 1990. *Systems Engineering Fundamentals* was developed as a supplemental text to DMSC's systems engineering courses. DSMC uses the text in all its courses, and promotes it as a companion to the "How-to" Handbook published by the International Council of Systems Engineering (INCOSE).

The new guidebook is intended to be the systems engineering foundation for these courses; whereas, the INCOSE book is the application.

Additional description, as well as information on downloading a free pdf version or ordering hard copies, can be found at
http://www.dsmc.dsm.mil/pubs/gdbks/sys_eng_fund.htm

John Leonard of the Washington Military Area is the primary author.