Security in a COTS-Based Software System

Arlene F. Minkiewicz
PRICE Systems

Planning and budgeting for the development of a software system composed primarily of commercial off-the-shelf (COTS) components presents unique challenges to those with project and process responsibilities. These challenges are further intensified when security issues are present such as security requirements in the system being developed, security constraints applied to the development team, or both. Because these security issues may impact the functional size of the software being developed, the productivity of the development team, and the ability of the team to communicate, they are important considerations when estimating cost and effort for any software project. They have additional effects when the system is COTS-based. In August's CrossTalk, *work by this author presented a methodology for approaching COTS-based software projects [1]. This article extends that methodology to consider the impacts that security requirements or constraints may impose on each of the activities in the process. It presents an overview of the causes of security vulnerabilities in software and an understanding of how to assess what impact security constraints will have on your COTS-based software projects.*

The developers of software systems are relying on incorporating commercial off-the-shelf (COTS) software components to decrease cost and time to market for delivery of new software systems. Increasingly, the procurers of these software systems are driving this decision to consider COTS solutions for significant functionality. This emergence of COTS solutions as viable and desirable has occurred over time and has not been without trials on both sides. It has taken time for software developers and their customers to begin to understand where COTS solutions will save money and where they will not. Great strides have been made in this area, but there is still much to learn.

As this COTS revolution unfolds, another issue emerges that impacts the entire software industry and has particular ramifications to those developing COTS-based systems. This issue is the increased need for security in software systems driven by rapidly growing networking capabilities introduced by leaps in networking technology and the corresponding leaps in dependence on this technology. According to the National Strategy to Secure Cyberspace:

> Identified computer security vulnerabilities – faults in software and hardware that could permit unauthorized network access or allow an attacker to cause network damage – increased significantly from 2000 to 2002, with the number of vulnerabilities going from 1,090 to 4,129. [2]

The Computer Emergency Response Team Coordination Center at Carnegie Mellon University, which tracks incidents of malicious software intrusions, reports a 2,099 percent increase in incidents from 1998 to 2002. These incidents range from security breaches impacting a single site to those impacting hundreds of sites [3]. These security breaches cost U.S. industry and government billions of dollars.

At the same time that purchasers of software systems are driving solution providers toward possible cost savings with a well-implemented COTS-based solution, they are putting additional demands on the software developers that these solutions meet specific software security criteria. These criteria are generally based on Evaluation Assurance Levels (EALs) as specified in the Common Criteria (CC) for Information Technology Security Evaluation [4] or some analog to these criteria. The CC sets a standard for assigning levels of security compliance of software. This additional requirement for secure software could clearly throw new and unexpected complications into the slowly emerging sense of what drives the costs of COTS-based software projects.

This article addresses the software security issue in general and then in the context of developing COTS-based systems. It begins with an outline of the author's methodology for this research, then describes and bounds the problem being addressed. Next, the issues surrounding software security are addressed. The author then outlines the six steps to a successful COTS implementation and addresses how these steps are impacted by security requirements.

Solution Methodology

The first step in any operations research project is to identify the problem being solved. The problem we are attempting to solve is that of identifying how security constraints impact the activities involved in delivering a COTS-based software system and how these impacts affect project costs. This was accomplished using literature reviews, expert knowledge, and interviews with practitioners to supplement the cost data available.

Once the problem is identified, the next step in constructing a parametric cost-estimating solution is to study and understand the subject process, and from this construct a mathematical model. Research led to the development of a mathematical model based on assumptions of productivity standards for each activity in the process, and adjustments of these standard values to accommodate for the additional rigor and processes associated with elevated security requirements.

This mathematical model was then exercised by software developers and project mangers to determine how it fared when applied to real-life situations. Once satisfied that the model was useful for practitioners, data from various datasets containing both commercial and aerospace data was applied to determine where it worked well and where further work was required.

Bounding the Problem

COTS-based software solutions can be a cost effective method for successfully delivering software systems if these projects are planned with a proper understanding of the activities associated with the implementation and on-going sustenance of COTS-based software systems. These activities and their cost drivers have been well defined in [5]. Adding security constraints to a COTS-based software system will affect the execution of some of these activities and, correspondingly, change how these activity's costs should be evaluated.

In bounding the problem, it is important to understand what definition of COTS is being applied and how security constraints are measured. To have a meaningful discussion on the costs of COTS software, it is important to start with a clear understanding of what is and is not included when we discuss COTS software. For this article, we started with the definition from the University of Southern California's Center for Software Excellence study that led to the Constructive COTS (CoCOTS) model [6]. The definition of a COTS software product follows:

- Commercially available software product – sold, leased, or licensed.
- Source code unavailable but documentation provided.
- Periodic releases with new features, upgrades for technology, etc.

We found this definition too limiting for several reasons. It is not consistent with many of our observations of actual COTS software integration efforts. Customization appears to be quite common, particularly with embedded COTS software. In creating a general-purpose solution, it seemed important that we not overlook the issue of customization. For this reason the definition was altered to include off-the-shelf software with source code available.

We have used the CC as the basis for evaluating security requirements. It was developed to provide a standard for security criteria and evaluation processes of that criteria. The CC contains seven hierarchical sets of assurance requirements called EAL's and named EAL1 through EAL7, with EAL1 representing the least amount of security and EAL7 representing the most security. Increasing degrees of documentation, design rigor, formal processes, etc. are required as you move from EAL1 to EAL7.

Software security issues can be grouped into three categories [7]:

- **Development for Security.** The effect of security constraints on the development of software.
- **Operational Security.** The effect of security policies and processes on the development environment.
- **Physical Security.** The effect of developing software in a secure environment.

The research in this article only covers the first two of these categories. Physical security is left as a future research challenge.

Software Security

In its report issued March 2004, the Software Process Subgroup of the Task Force on Security across the software development life cycle defines the goals of software security:

> The primary goals of software security are the preservation of the confidentiality, integrity, and availability of the information assets and resources that the software creates, stores, processes, or transmits, including the executing programs themselves. [3]

In other words, users of secure software have a reasonable expectation that their data is protected from unauthorized access or modification and that their data and applications remain available and stable. Clearly some applications have a need for a much higher degree of assurance than others. Software designed for stand-alone desktop use is much less likely to be subject to threats than multi-user software intended for use across a network. Software that is shipping confidential client information or government secrets requires more security than the instant messaging software that is the backbone of teenage culture.

"Despite the advances that the software industry has made in development practices and processes, the buffer overflow continues to be the most frequently cited security vulnerability ..."

As increasing security constraints will increase the cost of developing and maintaining any software system, it becomes an important part of the software acquisition process to understand the required level of assurance based on the functionality the software is intended to perform and the end users who will consume that functionality.

Software security requirements present themselves in two forms. The first are those requirements that impose additional functional requirements for features specifically related to security. Examples of these are encryption algorithms, password protection requirements, or remote access security procedures. The second form of security requirements relate to the additional levels of qualification and testing required to ensure that the software does not allow security breaches into the system on which it operates or the data it maintains. These requirements involve ensuring there are no back doors, buffer overflows, or defects that allow entry to hackers. They also require that patch releases be handled in such a way that the wise hacker cannot use them as road maps to the weaknesses in the software system.

The functional requirements related to security impact cost primarily through increased functional size of the application. When the software is being developed in-house, this additional size drives design, code, and test activities for these requirements. When the software is COTS, this additional size drives integration and test activities.

Before a conversation can take place about the cost impacts of compliance to specific security assurance levels, it is important to understand the various COTS selection strategies that might apply. The selection strategy applied determines where in the COTS implementation process the costs will be incurred and what the extent of those effects might be. There are basically three choices an integrator has when selecting COTS components for a software system:

- Buy and wrap.
- Buy only pre-certified components.
- Buy components and certify internally.

The *buy-and-wrap* option indicates that the integrator intends to purchase the COTS components that best meet the functional requirements for the software. The integrator will then develop a glue code wrapper that operates around this component, isolating it so that any security threats inherent in the COTS component cannot be reached through the resultant applications.

An integrator who opts to *buy only pre-certified components* intends to limit evaluations to COTS components from vendors that have certified their components to the required level of compliance. This reduces the integrator's development and integration effort, but limits the selections and is likely to increase the purchase and maintenance costs of the COTS components. It may also lead to the necessity to develop more in-house functionality if properly certified components are unavailable for some functional requirements.

The option to buy components and

certify internally seems the least desirable of all options. When this path is taken, the integrator purchases the components that best meet their requirements and then has these components certified to the required level of compliance. The integrator can select components that meet functional requirements, but then must bear the burden and expense of the certification process. If the certification fails, the integrator must then restart the evaluation and selection process to identify other potential solutions and must contend with the fact that they have purchased software that has no value.

Security breaches in software are caused by defective specification, design, and implementation [3]. Lack of quality in software creates openings through which threats can be realized. Despite the advances that the software industry has made in development practices and processes, the buffer overflow continues to be the most frequently cited security vulnerability according to a paper published by the Oregon Graduate Institute of Science and Technology [8]. Other defects that lead to security breaches include format bugs, resource leaks, hard-coded path names, and malformed inputs. Additionally, sloppy implementation of encryption algorithms, password transmission routines, and graceful failures lead to security breaches.

Clearly software that has been developed and integrated by organizations that have invested in putting proven software engineering practices and processes in place is going to reach security assurance level compliance with less cost (and less drama) than software developed in a more ad-hoc fashion. The institutionalization of good software development processes is an important factor to consider when planning a project with security constraints.

Another commonly cited cause for vulnerabilities in software is the training and expertise of the developers and integrators of software systems. The successful implementation of secure software systems requires that security and quality be considerations from day one. The skills required to do this are not innate and have not traditionally been stressed as part of a typical computer science or software engineering curriculum. Although this condition is improving with respect to good software development practices, those skills related specifically to security concerns continue to be overlooked in many instances. The cost impact of *fixing* a software system not designed and implemented to be a secure, defect-free system is substantially higher than the cost of developing it from the ground up.

Six Steps to a Successful COTS Implementation

To understand the cost impacts of security constraints on the development of a COTS-based system, it is important to first understand the activities that occur during any successful COTS-based software project. These activities are briefly outlined below. Also, [1] and [4] present more detailed descriptions of this process. Research has indicated the essential activities that must take place to ensure successful COTS-based projects are the following:

- Analyze software requirements.
- Evaluate and select COTS solution(s).
- Negotiate terms with vendors.
- Implement the COTS-based solution.
- Maintain license, subscription, and royalty fees.
- Maintain and upgrade the COTS-based solution.

Analyze Software Requirements

Software requirements analysis should occur regardless of whether the decision is made to build, buy, or borrow. A proper understanding of system requirements that are to be satisfied by software is essential to a successful software project. It is during the software requirements analysis activity that decisions can be made as to which requirements can be satisfied with off-the-shelf components.

Evaluate and Select COTS Solution(s)

Once a decision to pursue a COTS alternative is made, the first step is to determine the availability of COTS solutions that have the potential to provide needed functionality and evaluate these solutions. The evaluation needs to be focused on more than just product characteristics such as functionality, architecture, and technology, but also on vendor characteristics such as maturity, stability, and ability to provide adequate support, training, and documentation.

Negotiate Terms With COTS Vendors

Certainly it is important to negotiate the best deal possible when working with one or more vendors to craft a solution. Vendors are much more likely to address customer concerns with missing or incomplete functionality as well as bugs in the software before they sign on the dotted line.

Implement the COTS-Based Solution

Once an analysis, evaluation, and selection of a COTS-based solution is complete, implementation can commence. Implementation includes the tailoring of the COTS components; modifications to COTS software (if this is possible and desirable); design, code, and test of any glue code required; and higher level integrations of COTS components with other components.

Maintain License, Subscription, and Royalty Fees

License or maintenance fees need to be paid to ensure updates and upgrades as well as continuing support of the COTS components.

Maintain and Upgrade the COTS-Based Solution

Once the software is deployed, there are several ongoing activities required to keep it operational and keep end users happy. These include the ongoing evaluation of upgrades from the vendor, inclusion of those upgrades when desirable, bug fixes in glue code or to compensate for errors in the COTS components the vendor will not fix, and higher level integration as upgrades and bug fixes are deployed.

Security Implications for the Six Steps

The previous section outlined the activities that should take place, and thus should be part of the plan, for the implementation of any COTS-based system. This section looks at the additional factors that must be considered when the COTS implementation includes requirements to comply with security constraints, and highlights how those factors will impact activity costs as compared to the costs of those same activities with nominal security constraints.

Analyze Software Requirements

During the Requirements Analysis activity, functional security requirements need to be analyzed, along with the non-functional ones. From a cost and planning perspective, these additional requirements should be modeled as an additional functional size that needs to be either developed or purchased and integrated. Functional size can be any measure of software size that relates to the amount of user functionality that is being delivered such as function points or feature points. An additional cost consideration is the amount of skill and expertise the integration team has with implementing secure software systems. It is during the requirements analysis phase that the COTS selection strategy is likely to be determined.

Evaluate and Select COTS Solution(s)

The COTS selection strategy plays a big part in determining the impact of security constraints on this activity. If the strategy is to buy and wrap COTS components, then the evaluation and selection activity effort should focus entirely on the functional requirements for which COTS components are being considered, with little attention paid to the security requirements since this will be taken care of externally to the COTS components. The cost impacts of this activity would be a function of the total number of COTS solutions available for consideration, along with the number expected to be selected for more detailed consideration. The decision to build a wrapper also impacts the cost of glue code development (described below) as the need for wrapper code increases the amount and complexity of glue code that must be developed.

If the strategy is to buy pre-certified components, you would expect this to have substantial impact on the amount of time and effort associated with this activity. This strategy is likely to result in a broader initial search for qualified COTS components that meet the desired functionality and security constraints, but then a smaller set of candidate components available for detailed evaluation. The cost and effort impacts to this activity will be a function of the security assurance level required and the number of COTS components available for evaluation that are certified to that level.

The strategy to buy and certify components is a bit riskier to predict with respect to effort or cost because there is an inherent risk that the evaluation process may be revisited if internal certification fails for selected components. Integrators should take steps to mitigate this risk by only evaluating components from vendors with documented software development practices and proven quality records. With such risk mitigation strategies in place, this activity would consist of a broader initial search, as all available components are contenders followed by a more detailed evaluation of the few that appear to meet high quality standards. Additionally, certification comes with its own costs, including fees for laboratory testing, modifications identified as the result of this testing (if modifications are possible and desired), collecting documentation required for assurance, and fees to certification agencies.

The cost and effort impacts to this activity will be driven by the security assurance level required, the number of COTS components likely to be certified to that level, the functional size and complexity of the COTS components that require certification, and the state and availability of documentation.

Negotiate Terms With COTS Vendors

If the strategy is buy and wrap, then negotiations with vendors should not be impacted by security constraints. Vendor pre-certification is unlikely to impact the negotiations process either, unless the pre-certification is being done as part of the negotiated terms of the contract. If the certification is being performed by the integrating organization, the terms of the contract should include actions to be taken if COTS components fail to meet any promises made with respect to quality or security.

"It is during the software requirements analysis activity that decisions can be made as to which requirements can be satisfied with off-the-shelf components."

Implement the COTS-Based Solution

The implementation activities for a COTS-based solution can be affected by security constraints in various ways. As with previous activities, the selection strategy determines which activities are affected and to what extent.

- **Tailoring of COTS solution.** Security-specific tailoring tasks should be handled as part of the functional requirements for the system. The impact to the cost of this activity would be a function of the functional size imposed through those requirements.
- **Modification of COTS software.** If the buy and wrap strategy is in place, the costs of modifications should not change regardless of security requirements as the wrapper has isolated the COTS component from the rest of the system in a secure cocoon. If the COTS components are pre-certified or internally certified and then modified by the integrator, an entire recertification must be accomplished, most likely at the cost of the integrator. The cost and effort for this activity would be driven by the security assurance level required and the functional size of the COTS component(s) that are modified. Institutionalized software development practices and security expertise would mitigate some of the costs of this process.
- **Design, code, and test of glue code.** When a buy-and-wrap strategy is employed, the glue code size should be expanded to include the amount of wrapper code that must be developed. The cost of the development of this wrapper code, along with any other glue code that is written, will be impacted by the security assurance level required. This impact can be substantially mitigated when software development practices are institutionalized and when the integration team has training and experience in developing secure software.
- **Integration of COTS components with other COTS or custom components.** Integration and test are where the rubber meets the road. As with other non-functional requirements, security requirements are not fully testable until all the pieces of the system are working together. It is possible that certain combinations of components create security vulnerabilities where none existed within the individual components. Regardless of selection strategy, this activity will be impacted. Extent of this impact is determined by the security assurance level required, but is mitigated when good software development practices are institutionalized and the integration team contains members with security training and experience.

Maintain License, Subscription, and Royalty Fees

If the strategy is buy and wrap, this activity is unlikely to be impacted by security assurance requirements. If COTS components are pre-certified or internally certified, renewal time is the time to ensure that promises with respect to security and quality are being maintained as the COTS products are upgraded over time.

Maintenance and Upgrade of the COTS-Based Solution

Maintenance of any COTS-based system can be problematic, particularly as the number of different vendors increases. Security constraints further complicate the maintenance and upgrade activities.

- **Evaluation and inclusion of updates and upgrades from the vendor.** When a buy-and-wrap strategy is employed, the inclusion of upgrades may require modifications to wrapper code to accommodate new or changed interfaces. Upgrades for internally certified components would require recertification. One would expect that precertified components would recertify with each upgrade and update. Regardless of selection strategy, upgrades and updates require some level of integration and test at the system level to ensure that security constraints are still met by the system. The cost of this activity is driven by the security assurance level, but is mitigated by good development practices, training, and expertise.
- **Bug Fixes.** Regardless of selection strategy, bug fixes will require some level of reintegration and test, which requires reverification at the system (or subsystem) level that these fixes have not introduced security vulnerabilities. Additionally, if fixes are made to COTS components or to modifications made to those components, the entire certification process for that component must be recertified. The cost and effort of this activity is driven by the security assurance level, as well as the amount of glue code and modified code being maintained. This cost can be mitigated with the institutionalization of good software development practices and an integration team with security training and expertise.

Conclusions

More and more, integrators are being asked to deliver software systems that meet high-level security constraints while delivering most (if not all) of their functionality using off-the-shelf components. As integrators are just beginning to understand all of the issues associated with a successful COTS-based software project, they now have to understand how security assurance requirements will impact those issues and what new issues will be introduced.

Security constraints impact a project in two ways. Functional security requirements increase the functional size of the software system being developed and need to be treated in the same way as all other functional requirements being met by COTS components or homegrown code. Nonfunctional security requirements to attain a specific level of security assurance require additional processes, documentation, testing, and verifications. The additional focus on these activities will result in additional cost to COTS integrators. Most security vulnerability in software is a result of poor quality and poor or inconsistent software development practices. By improving these practices and employing people with expertise in security and excellent software development skills, integrators can reduce the cost increases driven by requirements for security.

This article reviewed the impacts of security constraints on the cost of a project to deliver COTS-based software systems. It reviewed these impacts in the context of the activities required to successfully implement a COTS-based system and highlighted those areas where security constraints apply. The security-related factors found most likely to impact costs were the COTS selection strategy employed and the security assurance level required. In organizations where investments have already been made to institutionalize good software development processes and hire people with the training and experience to deliver secure, defect-free software, the cost impact of security requirements is less severe.◆

References

1. Minkiewicz, A. "The Six Steps to a Successful COTS Implementation." CrossTalk Aug. 2005 <www.stsc.hill.af.mil/crosstalk/2005/08/0508Minkiewicz.html>.
2. The White House. "National Strategy to Secure Cyberspace." Feb. 2003 <www.whitehouse.gov/pcipb>.
3. Software Process Subgroup of the Task Force on Security Across the Software Development Life Cycle. "Process to Produce Secure Software." National Cyber Security Summit, Mar. 2004.
4. The National Institute of Standards and Technology and the National Security Agency. "Common Criteria for Information Technology Security Evaluation, Vers. 2.1." National Institute of Standards and Technology. Aug. 1999 <http://csrc.nist.gov/cc>.
5. Minkiewicz, A. "The Real Costs of Developing a COTS-Based System." Proc. for IEEE Conference on Aerospace and Defense, Big Sky Montana, Mar. 2004.
6. Center for Software Engineering. "CoCOTS White Paper." Los Angeles, CA: University of Southern California, June 1997 <http://sunset.usc.edu/research/COCOTS/cocots_main.html>.
7. Reifer, D., et al. "Estimating the Cost of Security for COTS Software." 2nd International Conference on COTS-Based Software Systems. Los Angeles, CA. Mar. 2003.
8. Festa, P. "Study Says Buffer Overflow Is Most Common Security Bug." C-Net news.com 23 Nov. 1999 <http://news.com.com>.

About the Author

Arlene F. Minkiewicz is chief scientist of the Cost Research Department at PRICE Systems. She is responsible for the research and analysis necessary to keep the suite of PRICE estimating products responsive to current cost trends. In her 20-year tenure with PRICE, Minkiewicz has researched and developed the software cost estimating relationships that were the cornerstone for PRICE's commercial software cost estimating model, ForeSight, and invented the Cost Estimating Wizards originally used in ForeSight that walk the user through a series of high-level questions to produce a quick cost analysis. As part of this effort, she invented a sizing measurement paradigm for object-oriented analysis and design that allows estimators a more efficient and effective way to estimate software size. She recently received awards from the International Society of Parametric Analysts and the Society of Cost Estimating and Analysis. Minkiewicz contributed to a new parametric cost estimating book with the Consortium for Advanced Manufacturing International called "The Closed Loop: Implementing Activity-Based Planning and Budgeting," and she frequently publishes articles on software estimation and measurement. She has been a contributing author for several books on software measurement and speaks frequently on this topic at numerous conferences.

PRICE Systems
17000 Commerce PKWY STE A
Mt. Laurel, NJ 08054
Phone: (856) 608-7222
Fax: (856) 608-7247
E-mail: arlene.minkiewicz@pricesystems.com