Estimating the Cost of Security for COTS Software

Donald J. Reifer, Barry W. Boehm, and Murali Gangadharan

University of Southern California
Center for Software Engineering
Los Angeles, CA 90089
d.reifer@ieee.org
{boehm, murali}@usc.edu

Abstract. This paper describes enhancements being made to the University of Southern California's **CO**nstructive **COTS** (COCOTS) integration cost model to address security concerns. The paper starts by summarizing the actions we have taken to enhance COCOMO II to model the impact of security on development effort and duration. It then relates the COCOMO II approach to the COCOTS estimating framework so that the enhancements proposed can be incorporated into the COCOTS model. After summarizing the team's progress in developing counterpart COCOTS security cost drivers and expert-consensus cost driver parameter values, the paper points to the steps that will be taken to validate the findings and calibrate the model.

1 Introduction

As the software engineering community incorporates more and more COTS software into its systems, security becomes a more important concern. The reason behind this worry is that firms are becoming increasingly apprehensive that their critical operations will be disrupted as they integrate and use more and more commercial components into their systems. For example, spyware inserted in commercial operating systems have made it easy for non-authorized parties to collect proprietary information about an individual's or firm's usage habits. Additionally, viruses, worms and Trojan horses inserted in COTS packages have infected entire systems and brought them down for prolonged periods of time. What's worse, according to the National Research Council's recent report of Trust in Cyberspace [1], many of the threats posed by COTS components are simply ignored by firms because they don't know how to deal with them.

Understandably, there are many factors that make COTS security hard to address [2]. Predominate among them is the fact that COTS software is provided by third parties to firms in executable form without full disclosure of what the software functionality and performance is. While it is relatively easy to analyze source code for security flaws, checking binaries is a labor intensive and arduous task. The situation is further confused by the fact that most COTS packages are in a variable state of flux much of the time. This is due to the fact that new versions of the package are released frequently and inconsistencies between vendor offerings only become apparent when usage patterns change over time during operations.

Many COTS users would like to do more when it comes to security. But, they don't understand what factors drive the costs of additional assurance and are concerned that the effort will be too time-consuming and expensive relative to the benefits derived. They need help in understanding the consequences of their actions in terms of dollars and cents. In response, the Center for Software Engineering at the University of Southern California (USC/CSE) has mounted an effort to address these concerns as part of its active COTS software integration and maintenance cost modeling efforts. This paper reports the initial results of our efforts to date in structuring and analyzing the effects of required security on COTS integration effort.

2 The COCOTS Model

USC/CSE has been researching the topic of COTS costs for several years. We have worked with clients to understand the factors that drive the cost. Using this information, USC has developed a cost-estimating model called COCOTS (**CO**nstructive **COTS** integration cost model [3]). COCOTS builds on the popular USC/CSE COCOMO II model [4] to predict the effort involved in integrating COTS software products into applications using the activity-based approach shown in Fig. 1. In the COCOTS model, the assessment activity refers to the process by which COTS components are selected for use. Tailoring refers to those activities undertaken to prepare the selected COTS packages for use. Glue code development refers to development and testing of the connector software, which integrates the COTS components into the larger application.

In Fig. 1, software systems are assumed to comprise of a mix of new, modified, reused and COTS components. The central block in the diagram indicates that to determine the total effort, the COCOTS estimate must be added to the numbers predicted for the non-COTS software by the COCOMO II model. The relative size of the effort in each of the blocks is a function of the size of the final application. The terms Life Cycle Objective (LCO), Life Cycle Architecture (LCA) and Initial Operational Capability (IOC) refer to project milestones in the USC MBASE life cycle model. The COTS part of the estimate is developed for each of these four activities using formulas that take component size and volume of work into account. Cost drivers like staff experience and desired reliability are used by the models to address variability of costs to operating conditions across organizations.

The total cost of incorporating COTS packages into an application becomes the linear sum of the three sources of effort as follows:

Total COTS Cost in COCOTS (staff-months of labor) =
Assessment Effort + Tailoring Effort + Glue Code Effort

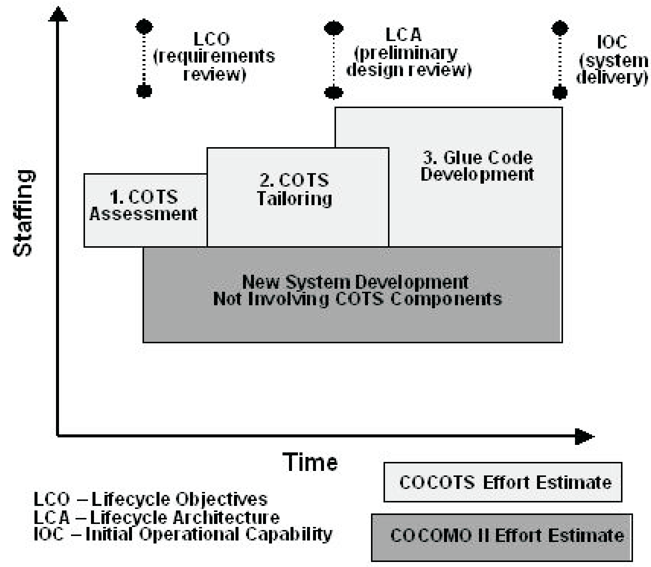
LCO (requirements review)
LCA (preliminary design review)
IOC (system delivery)
Staffing
1. COTS Assessment
2. COTS Tailoring
3. Glue Code Development
New System Development Not Involving COTS Components
Time
LCO – Lifecycle Objectives
LCA – Lifecycle Architecture
IOC – Initial Operational Capability
COCOTS Effort Estimate
COCOMO II Effort Estimate

Fig. 1. COCOTS Activity-Based Costing Elements

We have studied each of these activities in depth during the past three years. Using data collected on 20+ projects, we were able to identify the significant factors that influenced project effort and duration and determine their relative impacts on cost [5].

3 Security Enhancements for COCOMO II and COCOTS

Neither the COCOMO II nor the COCOTS estimating model currently includes security as a cost driver. This was not an oversight in either model's design. Until recently, security was considered a significant factor only for military projects. Several cost models addressed security from this vantage point. But, none of these models considers current threats posed by the current state-of-the-art of information munitions. We all are familiar with the threats posed by viruses, worms and denial-of-service attacks like those posed by Code Red. PC World magazine [6] reports that as many as 200 to 300 such viruses circle the globe in any given month. What's worse, the Software Engineering Institute reports that this number is rising with over 26,000 incidences reported just in the first quarter of this year as compared with 21,756 incidences reported in the year 2000 (see www.cert.org/stats/ for details). As the threat increases, our ability to deal with attacks diminishes. For example, a recent article [7] reports that the conventional "layered security" and "defense-in-depth" security strategies used by many firms to combat current threats have major holes. As another example, spyware can be covertly inserted into commercial applications via COTS software to perform industrial espionage. As a final example, tampering with binaries and COTS software represents yet another set of threats as adversaries try to exploit other's work and steal other's technology (key algorithms, protection schemes, make techniques, underlying frameworks, etc.). For these reasons, many commercial firms now are now interested in determining how the costs of implementing different security strategies.

To address security costs in traditional software developments, we have analyzed how existing models addressed security and found them deficient. In response, we proposed the use of an optional cost driver in COCOMO II [8] that we developed especially for security. As Table 1 illustrates, this driver builds on the large Body of Knowledge [9] that the security community has developed over the past decade and the Common Criteria [10]. The motivating force behind use of these techniques is the expansion of business into the world of systems of systems (i.e., network-centric processing of applications in a geographically distributed fashion). Such systems network organizations and applications together across the Internet, Virtual Private Networks, Intranets and Extranets and permit wireless and mobile access to provide subscribers with instant access to information through linkages with distributed, persistent knowledge bases.

Using the framework in Table 1, we conducted a Delphi exercise to obtain some initial expert-based cost driver values for the relative cost of security for the COCOMO II model. This scheme views security from a design, protection and physical security perspective. We circulated a survey to USC-CSE's Affiliates to rate the cost of implementing different strategies within organizations, both aerospace and comercial. The goal was to develop and calibrate a security cost driver that firms and government agencies could use if they needed to. In the COCOMO II software that we freely distribute from our web site at http://sunset.usc.edu, we permit users to employ a user-specified cost driver to address the additional cost of parameters of interest. Because cost drivers are chosen to be relatively independent random variables, variation in one can be considered without considering conditional relationships with other parameters. Using the information presented in Tables 1 and 2, the expert consensus indicates that the delta cost per instruction for implementing software supporting a layered defensive security strategy would be forty-one percent higher than for taking the nominal passive defense strategy.

Unfortunately, the scheme we devised for COCOMO II will only apply to the glue code portion of COCOTS. For completeness, we will have to determine the delta effort associated with the two additional activities in this model as illustrated in Table 3 and sum them to develop an estimate for the total security effort associated with securing a COTS package in staff-months. As the Table indicates, our initial poll of experts believes that security adds a nominal fifty percent to the cost when of COTS. Furthermore, they believe the range of impact on effort is between ten and ninety-five percent and schedule between five and forty-three percent. Of course, there could be no impact. This would be true if security was not a concern and protection did not have to be built into the system to limit the negative impact.

Table 1. Proposed Security Cost Driver Rating Scheme for COCOMO II

Rating	**Design for Security**	**Operational Security**	**Development Constraints**
Extremely High (XH)	Defense-in-depth strategy is planned for. The design is formally verified. The resulting model is supplemented by a formal specification of the requirements and high-level design. Evidence of developer "white box" testing and complete independent confirmation of developer test results is provided along with correspondence proofs. Security controls are supported by a measurable life-cycle model that supports selected protection mechanisms.	The defense-in-depth strategy for protection is implemented. The security policy is represented as a formal model. Protection against insider and outsider attacks is provided using intrusion detection systems, proxy servers and firewalls. Hardware write-protected audit trails are maintained to capture forensic evidence. Independent vulnerability analysis and penetration tests performed by external teams. All communications and storage are encrypted.	Physical security is strengthened even more to include the most biometric devices and other modern access control.
Very High (VH)	Layered defense strategy is planned for. The design is verified using a semi-formal representation. Security is supported by a modular and layered approach to design, and a structured presentation of the implementation. Independent search for vulnerabilities is performed to ensure high resistance to penetration attack. Systematic analysis of covert channels is performed. Security controls are supported throughout development by a standardized life-cycle model that embraces selected protection mechanisms	A layered defense strategy is implemented to protect the system. Audit trails are maintained and access to the system are strictly controlled. All external communications are encrypted. Message digests and integrity verification programs used to monitor activities for penetration attempts. Intrusion detection systems, firewalls and proxy servers are used to protect the network.	Specialized facilities are used to provide even more physical protection (SCIFs). Facilities are secured against emissions of spurious radiation. High levels of vigilance are maintained.
High (H)	A perimeter defense strategy is planned for. The design is tested and reviewed in depth to ensure security requirements are satisfied. The low-level design is analyzed to ensure proper protection. Testing is supported by an independent search for obvious vulnerabilities. Security controls are supported by the life-cycle model selected, tools, and the configuration management system.	A perimeter defense strategy is implemented to protect the system. An unclassified network is used for external communications with no co-mingling of access with the project's secure networks. An incidence response team is put into practice. Audit trails are maintained to track incidences. Data redundancy is used to ensure continuous availability of sensitive information.	Guards, cameras and other perimeter defense measures are put into place to provide additional physical security protection.
Nominal (N)	Security requirements are formulated for the system and its design using high- level guidance documents. Developer tests validating that these requirements are satisfied are independently verified. Black box and vulnerability analysis are performed.	Security policies are well specified. Reasonable password and virus protection practices are put into place along with database integrity and privacy controls. Project has administrator to monitor security controls and improve them as needed. Proper guidance documentation is in place for both the administrators and the users	Development personnel co-located in single facility with cipher lock or other protection to guard against unauthorized access.
Low (L)	No security requirements. No security protection other than that provided by the vendors built into either the product or the software engineering environment.	No organization-wide security policies. Ad hoc use of security practices. Some use firewalls and virus protection, some don't.	No unique facility requirements

Table 2. Security Cost Driver Values

	Ad hoc Defense (L)	**Active Defense (N)**	**Perimeter Defense (H)**	**Layered Defense (VH)**	**Defense in Depth (XH)**
Average Rating	0.94	1.02	1.27	1.43	1.75
Range of Ratings	0.91 to 1.0	1.0 to 1.05	1.1 to 1.4	1.2 to 1.6	1.4 to 2.0

Table 3. Range of Delta Effort and Duration Due to Security

Activity	**Nominal Effort**	**Delta Effort**	**Nominal + Effort**	**Range of Impact**	
				Effort	**Duration**
Assessment	Done in two passes: * Initial filtering effort * Detailed assessment effort	Adds a third pass: * Try before you buy	15%	+ 12 to 20%	+ 5 to 10%
Tailoring	Estimated assuming an average effort adjusted for complexity	Adds a second pass: * Assess vulnerabiliti es during setup	10%	+ 8 to 18%	+ 5 to 10%
Glue code development	COCOMO II-like model that uses cost drivers to address parametric variations	Add a new cost driver for security to glue code model	30%	+ 0 to 75%	+ 0 to 33%
+5 to 43%			+ 50%	+10to95 %	+5to43%

The COCOTS assessment effort estimation model includes security as one of 17 attributes being assessed for COTS suitability. For each attribute, the assessment effort is determined as (Number of Products assessed)*(Average assessment effort per product), without further guidance on the latter factor, using Table 1, we can provide a graded sequence of assessment activities needed for increasing security assurance. At the "Low" level, for example, organizations would tap into user communities to see if the COTS has any reported security holes. They would next follow up with the vendors to see if the security problems have been fixed in a timely manner. At the next level, they would embrace a "try the package, before buying it" philosophy. Evalutators would then acquire the COTS package using an evaluation license and test it for viruses, worms, covert channels and a variety of other known security issues. Next, they would try to identify vulnerabilities, isolate problems and determine potential damage inherent to COTS software by running a series of regression tests that they have prepared for such an examination. Further, they would search for dependencies and try to identify configuration-sensitive parameters and platform-sensitive behavior. As expected, this extra analysis and testing takes additional time and effort. Based upon our analyses as illustrated in Table 3, the range of the deltas involved in assessment due to security can add between twelve to twenty percent to the effort and five to ten percent to the duration of this activity.

The tailoring activity configures the COTS package for use within a specific context. Tailoring refers to the normal actions that would be undertaken to adapt a product to the specific host computing environment's configuration like parameter initialization, screen layout and/or report setup. Again, security can add effort and time to the job especially if those involved have to do more than scan each file before incorporating it into the configuration to determine whether or not it is free from viruses, worms, Trojan horses and other types of information munitions. The more secure the application, the more work involved. For example, those involved might perform sophisticated dynamic scans of the COTS software using some a threat simulator to test it for hijackers, hitchhikers or masquerades during setup if you were using it in a critical application. They might also have to examine the behavior of active applets with another threat simulator if you were using Java or other software with known security issues. Such scans are context sensitive and time-consuming to run. Again, as shown in Table 3, based upon our analysis the range of deltas for the extra time and effort involved in tailoring can add between eight to eighteen percent to effort and five to ten percent to the duration of this activity.

The glue code development activity is also impacted when security becomes a concern in the system. Because protection must be built into the system, generic wrappers and bindings build to interface COTS components to the operating system or middleware are hardened. This also takes more effort and time to accomplish. These resources can be easily assessed using an additional cost driver for security. Like in the COCOMO II model, such a security driver can be employed as a user-defined parameter in COCOTS because it uses a like mathematical formulation to predict effort and duration. However, driver ratings for COTS may be rated differently than those developed for COCOMO II to prevent a mismatch between the security level of the product and the application into which it is integrated. In addition, security ratings must be done in a manner that minimizes overlap between contributing parameters. For these reasons, we have developed the separate rating scheme in Table 4 for the proposed additional security cost driver that we are adding to the COCOTS model.

Based upon our analysis, the range of impact as shown in Table 3 is between zero and seventy-five percent for effort and zero and thirty-three percent for duration depending on the extent to which wrappers and bindings have to be hardened and the operational software has to be protected.

Table 4. Proposed Security Cost Driver Rating Scheme for COCOTS

Rating	Protect with	Value
Very High (VH)	Agents used to dynamically search for potential security breaches as connectivity is established and maintained across system in real-time. All of "High" plus top level anti-tamper and data integrity analysis.	1.45
High (H)	Hardened wrappers and bindings used to isolate access to operating system and middleware. All of "Nominal" plus lots of vendor liaison to ensure that adequate security protection is built-in the product.	1.29
Nominal (N)	COTS wrapped and layered for "plug-and-play" to minimize potential threats. COTS scanned both statically and dynamically for information munitions. Vendor error reports checked to ensure that there are no open security holes and all patches have been installed.	1.15
Low (L)	No protection. COTS scanned statically for viruses, worms, Trojan horses and other information munitions	1.00

4 Next Steps: Model Validation and Calibration

We have initiated an effort to validate statistically that the model accurately predicts the impact of added security on effort and duration. We have started with the values developed by experts via our Delphi exercise. Next, we will survey the 20 projects that we have used to initially calibrate the model to collect and analyze the actual impact of security considerations. In parallel, we will use the enhanced data collection forms to gather data from the additional projects that we are working with to determine whether the initial security calibration that we have derived makes sense and holds up when estimates are compared with actual costs. As the database grows, we will glean greater insights into what security strategies work for COTS, under what conditions, and at what cost. Our goal is to refine the initial model developed by experts using actual data provided by projects. As our work progresses, we will provide additional reports summarizing our results.

Acknowledgements. We would like to thank the following groups for supporting the development of COCOTS: the USAF, the FAA, the Office of Naval Research, the Software Engineering Institute, the USC-CSE Affiliates, the members of the COCOMO II research group and most especially the organizations and individuals that have participated in our Delphi exercises and supplied the data upon which our model is calibrated. Particularly valuable contributions have come from Arthur Pyster, Marshall Potter, Roger Cooley, Pat Lewis, Jim Thomas, and Pat Heyl of the FAA; Chris Abts of Texas A&M University, and Betsy Bailey of Software Metrics, Inc.

References

1. Committee on Information Systems Trustworthiness, Trust in Cyberspace. National Academy Press (1999)
2. Lindqvist, U., and Jonsson, E.: A Map of Security Risks Associated with Using COTS. In: IEEE Computer, June (1998) 60–66.
3. Abts, C., Boehm, B., and Clark, E. B.: COCOTS: A Software COTS-Based System (CBS) Cost Model—Evolving Towards Maintenance Phase Modeling. In: Proceedings of ESCOM (2001)
4. Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R., Reifer, D., and Steece, B.: Software Cost Estimation with COCOMO II. Prentice-Hall (2000)
5. Abts, C., Boehm, B., and Clark, E. B.: COCOTS: A COTS Software Integration and Cost Model—Model Overview and Preliminary Data Findings. In: Proceedings of ESCOM (2000)
6. Luhn, R., and Spanbauer, S.: Protect Your PC. In: PC World, July (2002) page 92
7. Mackey, R: Layered Insecurity. Information Security, June (2002) 61–68.
8. Reifer, D. J.: Security: A Rating Concept for COCOMO II. Center for Software Engineering, University of Southern California, May (2002)
9. Allen, J. H.: The CERT Guide to System and Network Security Practices. Addison-Wesley (2001)
10. Common Criteria for Information Technology Security Evaluation—Part 3: Security Assurance Requirements. Version 2.1, CCIMB-99-033, August (1999)