COCOMO Suite Methodology and Evolution

Dr. Barry Boehm, Ricardo Valerdi, Jo Ann Lane, and A. Winsor Brown
University of Southern California

Over the years, software managers and software engineers have used various cost models such as the Constructive Cost Model (COCOMO) to support their software cost and estimation processes. These models have also helped them to reason about the cost and schedule implications of their development decisions, investment decisions, client negotiations and requested changes, risk management decisions, and process improvement decisions. Since that time, COCOMO has cultivated a user community that has contributed to its development and calibration. COCOMO has also evolved to meet user needs as the scope and complexity of software system development has grown. This eventually led to the current version of the model: COCOMO II.2000.3. The growing need for the model to estimate different aspects of software development served as a catalyst for the creation of derivative models and extensions that could better address commercial off-the-shelf software integration, system engineering, and system-of-systems architecting and engineering. This article presents an overview of the models in the COCOMO suite that includes extensions and independent models, and describes the underlying methodologies and the logic behind the models and how they can be used together to support larger software system estimation needs. It concludes with a discussion of the latest University of Southern California Center for Software Engineering effort to unify these various models into a single, comprehensive, user-friendly tool.

In the late 1970s and the early 1980s as software engineering was starting to take shape, software managers found they needed a way to estimate the cost of software development and to explore options with respect to software project organization, characteristics, and cost/schedule. Along with a number of commercial and proprietary cost/schedule estimation models, one of the answers to this need was the open-internal Constructive Cost Model (COCOMO). This and other models allowed users to *reason* about the cost and schedule implications of their development decisions, investment decisions, established project budget and schedules, client negotiations and requested changes, cost/schedule/performance/functionality tradeoffs, risk management decisions, and process improvement decisions [1].

By the mid-1990s, software engineering practices had changed sufficiently to motivate a new version called COCOMO II, plus a number of complementary models addressing special needs of the software estimation community. Figure 1 shows the variety of cost models that have been developed at the University of Southern California (USC) Center for Software Engineering (CSE) to support the planning and estimating of software-intensive systems as the technologies and approaches have evolved since the development of the original COCOMO in 1981.

Figure 1: *Historical Overview of COCOMO Suite of Models*

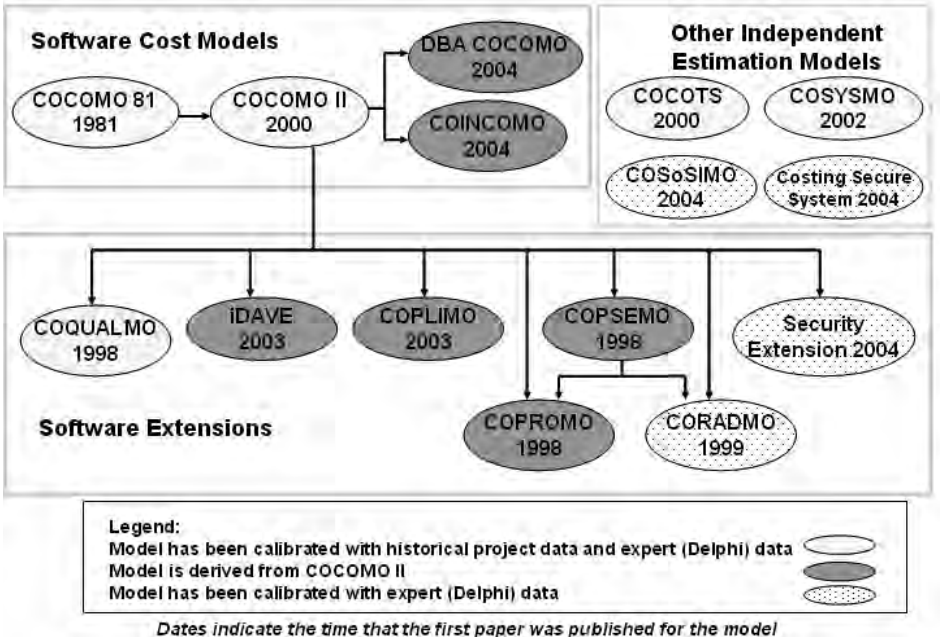


Figure 1 also shows the evolution of the COCOMO suite categorized by software models, software extensions, and independent models. The more mature models have been calibrated with historical project data as well as expert data via Delphi surveys. The newer models have only been calibrated by expert data.

Table 1 includes the status of the 12 models in the COCOMO suite. All of these models have been developed using the following seven-step methodology [2]: (1) analyze existing literature, (2) perform behavior analysis, (3) determine form of model and identify relative significance of parameters, (4) perform expert-judgment/Delphi assessment, (5) gather project data, (6) determine Bayesian A-Posteriori update, and (7) gather more data, refine model.

The checkmarks in Table 1 indicate the completion of that step for each model. Step 4 of the methodology can often involve multiple rounds of the Delphi survey that provide model developers some insight into the effects of the model parameters on development effort. The Delphi surveys attempt to capture what the experts believe has an influence on development effort.

Step 5 of the methodology involves collecting historical project data to validate the cost-estimating relationships in the model. This process depends on the support of the CSE affiliates to provide data that is relevant to the model being calibrated. The COCOMO model has more data than the other models com-

bined mostly because it has been around the longest, and it has been shown to be robust as well as accurate.

Step 6 involves combining the project data with the expert judgment captured in the Delphi survey to produce a calibrated model. This is done using Bayesian statistical techniques that provide the ability to balance expert data and historical data [2].

Model priorities, definitions, Delphi, and calibration data are collaboratively provided by the practical needs and experiences of USC CSE's supporting affiliates. These have included the major aerospace, computing, and telecommunications companies along with many of the major software and manufacturing companies, non-profits, professional societies, government organizations, and commercial cost model proprietors. For the list of CSE affiliates, visit <http://sunset.usc.edu/cse/pub/affiliate/general.html>.

The first three models (COCOMO II, COINCOMO, and DBA COCOMO) are fundamentally the same model but tailored for different development situations. In addition, commercial versions of COCOMO such as Costar <www.softstarsystems.com> and Cost Xpert <www.costxpert.com> provide further estimation-related capabilities. COQUALMO is used to estimate the number of residual defects in a software product and to provide insights into payoffs for quality investments. iDAVE estimates and tracks software dependability return on investment. COPLIMO supports software product line cost estimation and return on investment analysis. COPSEMO provides a phased distribution of effort to support incremental rapid application development and is typically used with CORADMO. COPROMO predicts the most cost effective allocation of investment resources in new technologies intended to improve productivity. All of the models described thus far are derivatives of the COCOMO model because they somehow depend on the output of COCOMO and modify it for certain situations.

The final three models are independent extensions of COCOMO that require their own inputs and can be used in conjunction with COCOMO, if desired. COCOTS estimates the effort associated with the integration of commercial off-the shelf (COTS) software products. COSYSMO estimates the systems engineering effort required over the entire system life cycle. COSOSIMO estimates the lead system integrator (LSI) effort associated with the definition and integration of software intensive system-of-systems (SoS) components.

Model	Description	Literature	Behavior	Significant Parameters	Delphi	Data
COCOMO II	Constructive Cost Model	✓	✓	✓	✓	>200
COINCOMO	Constructive Incremental COCOMO					
DBA COCOMO	DataBase (Access) Doing Business As COCOMO II					
COQUALMO	Constructive Quality Model	✓	✓	✓	✓	6
iDAVE	Information Dependability Attribute Value Estimation	✓	✓	✓		---
COPLIMO	Constructive Product Line Investment Model	✓	✓	✓		---
COPSEMO	Constructive Phased Schedule and Effort Model	✓	✓			---
CORADMO	Constructive Rapid Application Development Model	✓	✓	✓		16
COPROMO	Constructive Productivity-Improvement Model	✓	✓	✓	✓	---
COCOTS	Constructive Commercial Off-the-Shelf Cost Model	✓	✓	✓	✓	29
COSYSMO	Constructive Systems Engineering Cost Model	✓	✓	✓	✓	14
COSOSIMO	Constructive System-of-Systems Integration Cost Model*	✓	✓	✓		---

* Literature, behavior, and variable analysis limited due to number of available SoS to evaluate.

Table 1: *Status of the Models*

For more information on the COCOMO suite of models, visit: <http://sunset.usc.edu>.

Underlying Methodologies and Logic

The key to understanding the model outputs and how to use multiple models together is by comprehending the underlying methodologies and logic. In the development of a software-related cost model, the general COCOMO form is:

$$PM = A \times (\Sigma\, Size)^{\Sigma B} \times \Pi(EM)$$

where,

PM = person months.
A = calibration factor.
Size = measure(s) of functional size of a software module that has an additive effect on software development effort.
B = scale factor(s) that has an exponential or nonlinear effect on software development effort.
EM = effort multipliers that influence software development effort.

Each factor in the equation can be represented by a single value or multiple values, depending on the purpose of the factor. For example, the size factor can be used to characterize the functional size of a software module via either software lines of code *or* function points, but not both. Alternatively, the project characteristics can be characterized by a set of effort multipliers, *EM*, that describe the development environment. These could include software complexity *and* software reuse. COCOMO II has one additive, five exponential, and 17 multiplicative factors. Other models have a different number of factors that depend on the scope of the effort being estimated by that model. The number of factors in each of the models is shown in Table 2 (see next page).

The general rationale for whether a factor is additive, exponential, or multiplicative comes from the following criteria:

1. A factor that has effect on only one part of the system – such as software size – has a local effect on the system. For example, adding another source instruction, function point entity, module, interface, operational scenario, or algorithm to a system has mostly local additive effects on project effort.
2. A factor is multiplicative or exponential if it has a global effect across the overall system. For example, adding another level of service requirement, development site, or incompatible customer has mostly global multiplicative or exponential effects. If the size of

Model Name	Scope of Estimate	Number of Additive Factors	Number of Exponential Factors	Number of Multiplicative Factors
COCOMO	Software development effort and schedule	1	1	15
COCOMO II	Software development effort and schedule	1	5	17
COSYSMO	Systems engineering effort	4	1	14
COCOTS	COTS assessment, tailoring, and integration effort	3	1	13
COSOSIMO	SoS architecture and integration effort	4	6	---

Table 2: *Model Factor Types*

the product is doubled and the proportional effect of that factor is also doubled, then it is a multiplicative factor. If the effect of the factor is more influential or less influential for larger projects because of the amount of rework due to architecture and risk resolution, team compatibility, or readiness for SoS integration, then it is treated as an exponential factor.

These rules have been applied to the development of the COCOMO model as well as the associated models that have been developed at the CSE. The assumptions made about the cost estimating relationships in these models require that they be not only developed but also validated by historical projects. A crucial part of developing these models is finding representative data that can be used to calibrate the size, multiplier, and exponential factors contained in the models. The COCOMO form is a hypothesis that is tested by the data. For example, COCOTS data analysis showed that the COCOMO form applied to COTS integration, but that other forms were needed for COTS assessment and tailoring.

Table 2 summarizes the factors for the various COCOMO-independent models. The decision to have a different number of factors is determined by the Delphi process and confirmed by the data analysis, either of which can add or subtract factors from a model. However, the same criteria for factor type are used in all of the models. The COCOMO II extensions (shown in Figure 1) are based on the initial COCOMO II estimates with additional factors incorporated for the software characteristic of interest.

Understanding the scope of each model is also a key element in understanding the output it provides. The models in the COCOMO suite provide a specialized set of estimates that address specific aspects of development effort for software-intensive systems. COCOMO users are now beginning to use multiple models in parallel to develop cost estimates that cover a broader scope that exceeds the boundaries of traditional software development. In this case, the models in the COCOMO suite provide a set of tools that enable more comprehensive cost estimates. However, there are some limitations that exist when using multiple models together. These limitations are discussed in the next section.

Using Current Models Together

Many benefits exist when using multiple models in parallel. For one, they provide a more comprehensive set of estimates that better reflect the true effort associated with developing a software system. The effort that is not accounted for in COCOMO may be covered by other models such as COCOTS, COSYSMO, and COSOSIMO. Secondly, they enable the estimator to characterize the system in terms of multiple views.

However, some complications can arise when any two of these models are used in parallel since each of the models was initially developed as an independent entity. Just as the process model community has found that software engineering, software development, system engineering, and other activities are integrated, have dependencies, and cannot be adequately performed and optimized independently of each other, the estimation community has also found that these activities cannot be estimated independently for many of the larger software-intensive systems and SoS. Activities need to be planned and estimated at a program or project level.

Feedback from USC CSE affiliates and other COCOMO model users [3, 4] indicates that users would like a single tool in which they can do the following:

- Identify system and software components comprising the software system of interest.
- Easily evaluate various development approaches and alternatives and their impacts to cost and schedule.
- Understand the overlaps between models, if any.

Moving Forward – COCOMO Suite Unification

Efforts have been initiated at the USC CSE to develop a framework in which the key cost models can be integrated to provide a comprehensive software-system development effort to users. Once the models that are most likely to be used together are integrated, efforts will focus on the integration of other more specialized models. We will also begin with the models that have a high degree of maturity.

The purpose of this unification effort is similar to that of the individual cost models [2], that is, to help software-intensive system and SoS developers and their customers reason about the cost and schedule implications of their development decisions, investment decisions, risk management decisions, and process improvement decisions.

Key to our approach is distinguishing between an *integrated* set of models versus a truly *unified* model. When a set of models is integrated, typically each model becomes an entity in the integrated set with inputs into one model creating outputs that are then fed into subsequent models. However, when a unified model is developed, there is a reengineering of the set of models to come up with an architecture where the whole of the unified set is greater than the sum of the parts. Developing a unified COCOMO suite model will support the goals to minimize or eliminate overlap between the models, provide a relatively comprehensive coverage of the SoS, system engineering, and software development activities, and develop a relatively simple interface for specifying inputs as well as a well-integrated set of outputs.

Key Unification Issues

In August 2004, the CSE held an internal workshop to identify key issues for model unification. The outcome of the workshop was the identification of four areas of focus for unification: (1) selection of models that must be unified to support various types of development, (2) identification of the overlap between these models, (3) identification of missing activities not covered by any of the current models, and (4) specification of the required parameters and outputs for the related models in a user-friendly, consistent, and usable manner. The following sections describe some of the more detailed issues identified as part of the four focus areas.

Model Selection

Many of today's large software-intensive systems integrate legacy capabilities, COTS software products, and new custom software subsystems. No single COCOMO model covers the full life-cycle effort for the development of these types of sys-

tems. The new software development effort is easily estimated using COCOMO II. COTS customization effort might be estimated using another COCOMO suite model: COCOTS. COSYSMO would typically be used to estimate the system-level engineering activities such as feasibility analysis to support the integration concept, functional analysis of the new requirements, trade-off studies, prototyping, performance evaluation, synthesis, and system verification and validation activities. And finally, COSOSIMO might be used to estimate the effort associated with the integration of the legacy system with the COTS system and the new custom software system. CSE corporate affiliates have identified potential combinations of cost models that would be of value to them, including COCOMO/COSYSMO/COCOTS and COCOMO/COSYSMO/COSOSIMO [4].

Model Overlap

Further analysis is required to determine the extent of any overlap between the various COCOMO models. Potential overlap issues were identified with respect to various combinations of the primary cost models as well as with respect to the general integration of software and system components.

- **COCOMO II and COSYSMO Model Overlap:** Currently, COCOMO II is designed to estimate the software effort associated with the analysis of software requirements and the design, implementation, and test of software. COSYSMO estimates the system engineering effort associated with the development of the software system concept, overall software system design, implementation, and test. Key to understanding the overlap is deciding which activities are considered *system engineering* and which are considered *software engineering/development*, and how each estimation model handles these activities.
- **COSYSMO and COSOSIMO Model Overlap:** COSOSIMO aims to estimate the effort associated with the architecture definition of a *SoS* as well as the effort associated with the integration of the highest level SoS components. On the other hand, COSYSMO estimates are done in the context of a single system and include the effort needed to define a single, system-level architecture, the design of the system components, and the integration of those components. COSYSMO also includes the effort required for the system development to support the integration of the system component in the target environment. Further work is required to understand the subtleties of these models and exact extent of any overlap between these models.

Missing Activities

Are there any key activities missing when the key models are viewed together? How are specialty engineering tasks for secure or sensitive systems handled? How are non-software system development tasks handled? What about logistics planning for operational support? Can effort from activities not supported by any current COCOMO model be easily integrated?

Effort Outputs

What granularity should be provided? One effort value? An effort value for each of the key models? By software component? By system component? By engineering category (e.g., software, systems engineering, LSI)? By phase/stage of development?

Understanding Unification Issues

To begin to understand these four unification issues better and to start developing a candidate approach for the unified COCOMO model, efforts were initiated to better understand the following:

- Current model boundaries.
- How the current models are typically used today.
- The activities associated with software development, system engineering, and SoS integration work performed by LSIs.
- What activities are included in each of the current primary cost models.

Current Model Boundaries and Usage

To address this first aspect, we developed a table to indicate when each model (or set of models) is typically used (Table 3). As part of this effort, we developed descriptions that tried to capture information about the current boundaries of each model and how those boundaries expand as the current models are used in an integrated manner.

Types of Effort Currently Estimated

The next step was to identify a comprehensive set of high level, software-intensive system life-cycle activities, the typical development organizations responsible for the performance of these activities, and the scope of the activity typically performed by each development organization. Then each activity covered by each of the primary cost models was identified. For example, the system engineering organization is typically responsible for the system/subsystem requirements and design, and the software development organization participates in a support or review role. Other activities, such as management, are often performed at various levels with each development organization having primary responsibility at their respective levels.

The results of this effort are shown in Table 4 (see next page). The shaded activities under Software Development are currently covered in COCOMO II and COCOTS. The shaded activities under System Engineering are currently estimated by COSYSMO. The shaded activities under LSI are currently estimated by COSOSIMO. The activities that are not shaded are currently not covered by any of the models in the COCOMO suite. And, since the focus of the COCOMO suite is on software-intensive systems, none of the items under the hardware development column are currently covered.

Some activities such as management and support, involve several organizations

Table 3: *How Current Primary Cost Models Are Typically Used*

Use ...	When scope of work to be performed is ...
COCOMO II	Development of software components (software development).
COCOTS	Assessment, tailoring, and integration of COTS products.
COSYSMO	Design, specification, and integration (system engineering) of system components to be separately developed for a single system.
COSOSIMO	Specification, procurement, and integration of two or more separately system-engineered and developed systems.
COCOMO II with COCOTS	Development of software components (software development), and a software system, including assessment, tailoring and glue-code for integration of COTS.
COSYSMO and COCOMO II	System engineering and software development for a single system with software-intensive components.
COSYSMO and COSOSIMO	System engineering of individual systems and integration of the multiple systems.
COCOMO II, COSYSMO, COCOTS, and COSOSIMO	System engineering, software development, and integration of multiple software-intensive systems and COTS products.

Activity	Responsibilities			
	Software Development (COCOMO II and COCOTS)	Hardware Development	System Engineering (COSYSMO)	LSI (COSOSIMO)
Management	Primary for Software Level	Primary for Hardware Level	Primary for System Level	Primary for SoS Level
Support Activities (e.g., Configuration Management and Quality Assurance)	Software Level	Hardware Level	System Level	SoS Component Level
SoS Definition			SoS Component	SoS Level
Source Selection and SoS Component Procurement				Lead
Subsystem Requirements	Review	Review	Elaboration* Lead	Inception Lead
System/Subsystem Design	Support	Support	Lead	Review
Hardware/Firmware Development		Lead		
Software Requirements Analysis	Elaboration* Lead		Inception Lead	
Software Product Design	Lead		Review	
Software Implementation/ Programming	Lead		Support	
Software Test Planning	Lead		Review/Support	
Software Verification and Validation	Lead		Review/ Support	
System Integration/Test	Support	Support	Lead	Review
System Acceptance Test	Support	Support	Lead	Review
SoS Integration/Test	Support	Support	Review/Support	Lead
SoS Acceptance Test	Support	Support	Review/Support	Lead
Manuals (User, Operator, Maintenance)	Software Lead	Hardware Lead	System Lead	SoS Level Lead
Transition (Deploy and Maintain)	Support	Support	System Lead	SoS Level Lead

* Model Based (System) Architecting and Software Engineering/Rational Unified Process phase of development.

Table 4: *Life Cycle Activities*

at different layers of the system. Extreme care needs to be taken when developing models that cover activities that have shared responsibilities with hardware, software, and other players.

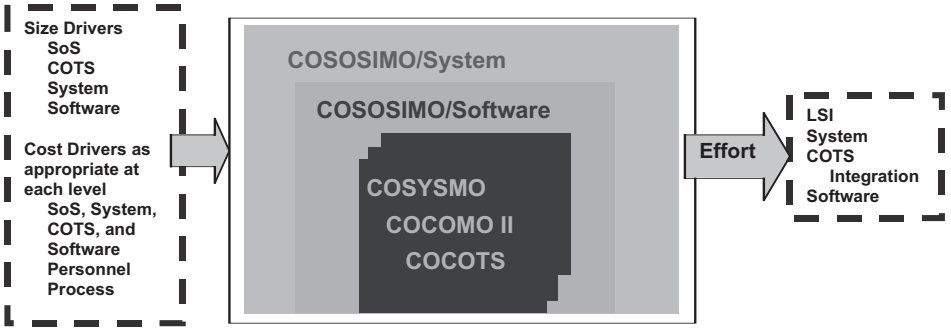
The identification of such activities is the first step in identifying possible overlaps between models. Further difficulties arise when dealing with different organizations that use customized work breakdown structures. These, along with the aforementioned challenges, will continue to be addressed as the model unification efforts continue at the CSE.

As seen from the discussions above, there is still much work to be done in order to support the unification of the COCOMO models. These include the following:

1. Develop a more complete description of activities covered by each model. These descriptions will allow us to identify, minimize, or eliminate any overlap between the models and identify software system-related activities not covered by any of the models.
2. Determine more precisely how traditional phase activities and Model Based (System) Architecting and Software Engineering/Rational Unified Process [1] phases map to cost-model activities and how these phases are integrated at the SoS, system, and software levels. Work in this area has already begun [5] but some unresolved issues remain in the context of unified models.
3. Refine counting rules/definitions for model inputs and outputs and then determine how they can be combined into an efficient, user-friendly unified model.
4. Determine typical distribution profiles for effort across all of the activities/ phases in a unified environment.

The initial goal of this effort is to develop a unified model that includes COCOMO II, COSYSMO, COCOTS and COSOSIMO as shown in Figure 2. As we learn from this process, we will begin to add other models from the COCOMO suite.

Figure 2: *Early Unification Goal*



The current unification effort will help establish a framework and define the context for the evolution of the unified model into something that can provide a comprehensive estimate for the development of software systems and software-intensive SoS. We will continue to collaborate with CSE affiliates with the goal of evolving the COCOMO suite so that it can help users make better decisions about the development of software-intensive systems.◆

References

1. Boehm, B. Software Engineering Economics. Prentice Hall, 1981.
2. Boehm, B., et al. Software Cost Estimation with COCOMO II. Prentice Hall, 2000.
3. Annual Research Review, Corporate Affiliate Survey. University of Southern California Center for Software Engineering, 16 Mar. 2004.
4. University of Southern California Center for Software Engineering. "Unification Workshop Minutes." 19th Forum on COCOMO and Software Cost Modeling, 26 Oct. 2004.
5. Boehm, B., A.W. Brown, V. Basili, and R. Turner. "Spiral Acquisition of Software-Intensive Systems of Systems." CROSSTALK May 2004: 4-9.

About the Authors

Barry Boehm, Ph.D., is the TRW professor of software engineering and director of the Center for Software Engineering at the University of Southern California. He was previously in technical and management positions at General Dynamics, Rand Corp., TRW, and the Defense Advanced Research Projects Agency, where he managed the acquisition of more than $1 billion worth of advanced information technology systems. Boehm originated the spiral model, the Constructive Cost Model, and the stakeholder win-win approach to software management and requirements negotiation.

E-mail: boehm@usc.edu

Jo Ann Lane is currently a doctorate student at the University of Southern California in systems architecting. Prior to this, she was a key technical member of Science Applications International Corporation's Software and Systems Integration Group. She has over 28 years of experience in the areas of software project management, software process definition and implementation, and metrics collection and analysis. Lane has a Master of Science degree in computer science from San Diego State University.

E-mail: jolane@usc.edu

Ricardo Valerdi is a member of the Technical Staff at the Aerospace Corporation. Previously, he worked as a systems engineer at Motorola and General Instruments. He is a doctorate candidate at the University of Southern California (USC) in the systems architecting program and is a research assistant at USC's Center for Software Engineering. Valerdi has a Bachelor of Science in electrical engineering from the University of San Diego and a Master of Science in systems architecting from USC.

E-mail: rvalerdi@sunset.usc.edu

A. Winsor Brown is a senior research scientist and assistant director of the University of Southern California Center for Software Engineering. As an engineer with decades of experience in large and small commercial and government contracting companies, he started his career in computer hardware design but shifted to software within months and remains there today. He has a Bachelor of Science in engineering science from Rensselaer Polytechnic Institute and a Master of Science in electrical engineering from California Institute of Technology.

E-mail: awbrown@usc.edu