Measuring Systems Engineering Productivity

Gan Wang
BAE Systems
Reston, VA
gan.wang@baesystems.com

Lori Saleski
BAE Systems
Hudson, NH
lori.a.saleski@baesystems.com

Alex Shernoff
BAE Systems
Reston, VA
alex.shernoff@baesystems.com

John C. Deal
BAE Systems
San Diego, CA
john.deal@baesystems.com

Abstract. Productivity measurement has long been employed to guide economic activities. Engineering organizations, as labor-intensive economic entities, have used a variety of labor productivity metrics to capture the underlying productivity and efficiency (a reciprocal notion), in their ever-enduring attempts to produce more outputs with fewer human resources. Depending on the functional disciplines, some widely embraced and de facto standard measures range from hours per drawing in hardware engineering, to lines of code per hour or day practiced in software engineering. Systems engineering, a relatively young profession still struggling to find its identity at times, has not yet reached an agreement on how it should capture its productivity and efficiency. However, such a measure is critical in managing resources, improving processes and work output, and establishing its utility as a bona-fide field of engineering in production of goods and services.

This paper presents a proposed systems engineering productivity and efficiency (P&E) measure for system development projects, where there is usually the greatest amount of systems engineering content. This measure is defined by using parameters described in COSYSMO, a parametric estimating model for systems engineering. We provide the definition, recommendations for deployment, and relay our experience in applying this measure to organizational productivity and efficiency improvement.

Introduction

In general economic terms, *productivity* is the amount of output created, in terms of goods produced and services rendered, per unit input used. The general production function can be expressed as:

$$Q = f(K, L) \qquad (1)$$

Where, Q = level of output, K = quantity of the capital, and L = quantity of labor. Figure 1 illustrates the production model in a simple context diagram, where economic processes or activities transform certain input (material and labor) to certain output (product). Such a measure of the output vs. input is constrained by the scope or size of product so that the quantities of the input and the output can be meaningfully measured and correlated. From this base definition, a variety of measures can be developed to gauge productivity from different

angles and focuses [Coelli, Rao, and Battese, 1998]. Mathematically, taking partial derivatives of the above production function yields productivity measures for the desired perspective. Practical examples of some widely used metrics include GDP per capita for national economy, dollar revenue/income per employee for a business organization, gross margin for a product line; return on asset and return on invested capital for performance of an investment, and quantity of products per hour for a manufacturing line or facility.

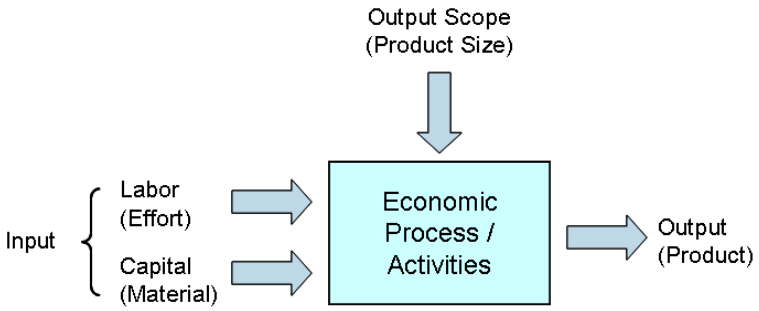

Figure 1 — General production model relating certain input to certain output under a defined scope.

Engineering, unlike manufacturing activities, is predominately labor-intensive and requires no capital on the recurring basis (or direct material). Labor productivity measures are most relevant. *Labor productivity* is typically measured as output per worker or output per labor-hour. Since it ignores the capital input, equation (1) becomes:

$$f = (L) \qquad (2)$$

And labor productivity can be expressed in a general form as:

$$\frac{dQ}{dL} = \frac{f(L)}{dL} \qquad (3)$$

As simple as it may seem, there have been noticeable difficulties in defining commonly accepted engineering productivity measures across industries. In certain engineering fields, some measures are relatively commonly recognized and used despite great variances. The most noticeable effort has been in software development productivity research [Wagner and Ruhe, 2008]. A leading example of such measures is source lines of code (SLOC) or function points (output) vs. development effort (input) for software engineering productivity [IEEE Standard 1045-1992]. In other fields, the variances seem to be much wider. The number of engineering drawings (output) vs. design and drafting effort (input) is less uniformly defined but frequently practiced in mechanical engineering, and the number of circuit boards (characterized by type, such as FPGA) vs. design and testing hours for electrical engineering productivity measurement seems to have a similar stature.

In fact, most of the productivity discussions to date pertaining to engineering development quickly turn to software productivity [Card 2006], because SLOC as the basic scope measure seems to be the only tangible thing people can grasp. In his work to explain the evolution and future trends for systems and software engineering, Boehm [Boehm 2005] discussed some expected transformations for both systems and software and examined the corresponding changes in processes that will be necessary to support them in order to improve productivity, but does not specifically identify any quantifiable measures of productivity besides those used in

software. A related effort [Boehm, et al, 2007] has attempted to quantify systems engineering tasks, with an emphasis on the architecture and risk resolution effort, for the purpose of calculating the return on investment for systems engineering, but goes no further in terms of measures.

A more recent study [Kasunic 2008] attributes the lack of standardization and common operational definitions for measurement as major obstacles to comparing software project performance. Attempts are made to define a comprehensive set of performance measures but, once again, they are mainly focused on software efforts and do not address systems engineering efforts.

The literature search effort by the authors in preparation of writing this paper has turned up little other than aspects of software productivity. There were very few that ventured into engineering productivity in general or systems engineering specifically.

One difficulty is the inability to define commonly accepted engineering *scope* or *size* of a product that consistently can be measured and applied to correlate the input and output measures [Card 2006]. Depending on the desired perspective and outcome, there is a wide range of possible measurable inputs and outputs and interpretations of the results. The same physical product is a different system to different engineers or engineering disciplines. In addition, there are enough discrepancies in measuring the same attribute that results can swing widely and fail to represent anything meaningful in the same field. While SLOC produced per unit labor hour seems to be straightforward enough, there are various definitions and interpretations in counting the lines of code, as no two lines of code are created equal. COCOMO II categorizes source lines of code into new, adapted, and reuse code [Boehm, et al, 2000]. It further examines the degree of modification in the adapted code in terms of amount of modification in design, code and integration, and adopts a notion of equivalent KSLOC. These are just some of the means to enforce consistency in measurement and to minimize the degree of variance in determining the scope or size of a product.

To date, as surprising as it may seem, there has been a void in systems engineering. There is no commonly embraced productivity definition for systems engineering. A major impediment, once again, has been the lack of a commonly accepted approach for sizing products for systems engineering. Perhaps due to the diverse job descriptions, it is difficult to define a single, across-the-board notion of a system for systems engineers. Without a quantifiable output, it is difficult to capture or identify the input that produces it. However, as the systems get bigger and more complex, there is an increase in the systems engineering content in developing a system. As all organizations strive to improve productivity and efficiency, there is an urgent need for a community-accepted definition of systems engineering productivity. The key is how to define the system and quantify its size for systems engineering.

It is worth noting that there are at least two prevalent views in considering the roles of systems engineering. One is the “production” or the “output” view, in which systems engineers produce their own products or outputs in the life cycle of a system. The other is the “outcome” or the “value-added” view, the one that says the work of systems engineering helps to shape or influence the product or outcome of other disciplines and facilitates better results, say, in quality or functionality. The “output” view logically supports the quantification of outputs, which endorses the measure of productivity. It is, however, less straightforward, if not contentious, to gauge the added portion of a product in the “value-added” view. This may be one of the reasons

that it has been difficult to determine the return on investment (ROI) for systems engineering. In this paper, we take the "output" view, where we consider that systems engineering, as a field of engineering, produces its own outputs.

In the following sections, we propose a definition of systems engineering productivity measures by applying a quantity called the system size defined by COSYSMO. We then discuss the use of such a measure through an organization pilot experience. We also provide practical advice for implementing such a measure for differing system development types of projects.

Measuring System Size

The emergence of the *Constructive Systems Engineering Cost Model*, or *COSYSMO* [Valerdi 2005], in recent years and its adoption by industry presented an opportunity for defining a practical productivity measure for systems engineering. The model definition solves a key stumbling block for measuring productivity.

COSYSMO, developed at the University of Southern California with the support of a consortium of academic, industry, and government organizations, is a parametric model for estimating the systems engineering and integration effort required for the conceptualization, design, test, and deployment of software and hardware systems and executing projects that develop such a system. It estimates the standard systems engineering and integration activities typically on the top layers of the system "Vee Model" in a standard system life cycle [INCOSE, 2007]. The basic model definition specifies the following estimating relationship:

$$PM_{NS} = A \cdot (System\ Size)^{E} \cdot CEM \qquad (4)$$

Where:

PM_{NS} = effort in person months (nominal schedule)

A = calibration constant derived from historical project data

E = represents (dis)economies of scale

CEM = composite effort multiplier determined by 14 cost drivers

COSYSMO defines a quantity called *system size* based on four sizing parameters or size drivers: system requirements (REQ), system interfaces (INT), system algorithms (ALG), and operational scenarios (SCN). Further elaboration of the model definition introduced a *reuse model* in counting these size drivers that allows classification of each driver count in terms of up to five categories of reuse and three levels of difficulty. Therefore, the system size is defined by the following equation:

$$System\ Size = \sum_{k} \left(\sum_{r} w_r \left(w_{e,k} \Phi_{e,r,k} + w_{n,k} \Phi_{n,r,k} + w_{d,k} \Phi_{d,r,k} \right) \right) \qquad (5)$$

Where:

$\Phi_{x,r,k}$ = quantity of "k" size driver, in the reuse category "r" and with the level of difficulty "x"

k = {*REQ*, *IF*, *ALG*, *SCN*}

$\mathbf{r}$ = {*New*, *Modified*, *Deleted*, *Adopted*, *Managed*}

w_r = weight for defined categories of reuse

w_x = weight for "*Easy*," "*Nominal*," or "*Difficult*" size driver

System size has a basic unit of measure known as "eReq," or *equivalent requirement*, which is the equivalent of one new nominal requirement in terms of end-to-end systems engineering effort. This unit of measure allows the conversion of four heterogeneous quantities into a single homogeneous unit and, subsequently, establishes a common denominator in measuring systems of different types and quantifying diversified system engineering work.

The underlying premise of this model is that its four specified size drivers completely describe the direct product and capture the scope for the work that systems engineers produce, regardless of the type of systems to build. In other words, the defined system size quantity can reliably determine the size of the system output in terms of the total effort required to produce it. It is worth pointing out that the validity of this model is still being verified through practical applications and may take a long time to be established for the entire community. However, when accepted, it provides a foundation that enables the measurement of productivity across all system types.

Proposed SE Productivity and Efficiency Metrics

Therefore, following the general form of the productivity as the ratio of output produced over the effort consumed, we can define the *systems engineering (SE) productivity* as:

$$SE\ Productivity = \frac{System\ Size}{Total\ SE\ Hours} \quad (eReqs / hour) \qquad (6)$$

Where the system size is the quantity defined by COSYSMO using the four size drivers and the total SE hours are the end-to-end systems engineering effort in developing a system or executing a system development project, from concept definition to delivery or sell-off.

On an intuitive level, SE productivity can be interpreted as the amount of system (measured in equivalent requirement, or eReq) realized or output produced per unit of systems engineering labor (measured in engineering hours) worked, under a nominal product environment. This system output may be viewed as the work-in-progress inventory from an accounting point of view. A reciprocal measure is the *SE efficiency* and it captures the number of systems engineering hours required to produce or realize a unit of system (measured in eReq) as:

$$SE\ Efficiency = \frac{Total\ SE\ Hours}{System\ Size} \quad (hours / eReq) \qquad (7)$$

Theoretically, with these definitions, any two system development projects or any group can be evaluated comparatively in terms of how productive the teams are and how well the projects are executed. This can apply to both projects in execution (assuming one can estimate the system size and the effort at project completion) and post-mortem. There are, however, important caveats and constraints in practical implementation. First, the absolute numbers produced by the above equations in isolation have little practical meaning. It is because the quantity *system size* measured in eReq is somewhat arbitrary, as there is no universal definition of *requirement*. The correct use is to observe the trend as a project progresses through its life cycle or to compare peer

projects that are measured using the same definition over time. These observations should be made periodically or aligned with similar programmatic and technical milestones. It is trend data (increase or decrease in productivity) that provides valuable guidance to project managers — not absolute values by applying the above equation.

The second implication is that the observation should be made to similar or comparable projects or systems only. One certainly would expect productivity differences between two systems of different types and scales. For example, it would not be a fair comparison between hardware-centric and software-centric systems, proprietary electronics board design and commercial-off-the-shelf (COTS)-based system integration, or a pharmaceutical product and a civil engineering project. Meaningful productivity comparisons can be made to like systems only.

Finally, two projects that measure to the same or similar system size can be expected to have different outcomes in terms of systems engineering effort consumed due to different project environments or system complexity factors. COSYSMO recognizes this by defining 14 cost drivers that act as effort multipliers and capture system-specific complexities and project-unique environments. The aggregate quantity of these drivers is known as the *composite effort multiplier* (CEM) and is defined by COSYSMO version 1.0 as:

$$CEM = \prod_{j=1}^{14} EM_j \qquad (8)$$

There are alternative implementations of this multiplier in practice. See details in the next section — "Applications of the SE Productivity and Efficiency Metrics."

The effort multipliers or the CEM help to compensate for environments unique to each system or project under consideration. When CEM is greater than 1.0, it adds a penalty to the project, which implies that for the system of the same size it would require greater SE effort. On the other hand, when CEM is less than 1.0, it adds a benefit and predicts smaller effort. Divided by CEM, the resulting SE hours represent the required effort for the same system size under the nominal project condition. On an intuitive level, a CEM of less than 1.0 implies that, had the project been conducted under the nominal condition, it would require greater effort. On the other hand, a project under a more-complicated environment (with CEM > 1.0) could have been completed with less effort.

We normalize the SE productivity metric using the same CEM factor, so that peer projects can be compared on a level playing field. Therefore, the *normalized SE productivity* can be defined as the actual productivity measured multiplied by CEM and it is expressed as below:

$$SE\ Productivity_{Norm} = \left(\frac{System\ Size}{Total\ SE\ Hours} \right) \cdot CEM \qquad (9)$$

And, consequently, the *normalized SE efficiency* can be expressed as:

$$SE\ Efficiency_{Norm} = 1/CEM \cdot \left(\frac{Total\ SE\ Hours}{System\ Size} \right) \qquad (10)$$

Once again, on the intuitive level, the normalized SE productivity represents, on average, the amount of a system in terms of equivalent requirement produced per SE hour under a *nominal* project environment. Similarly, the normalized SE efficiency represents the required SE hours to

realize one nominal requirement under the same project condition. These normalized measures provide a more objective or accurate comparison between or among projects.

Applications of the SE Productivity and Efficiency Metrics

Through an engineering productivity improvement initiative, the proposed set of P&E metrics was piloted at the authors' affiliated organizations. The authors helped to implement these metrics, which were applied to select projects in different business areas and product lines. This was the first time the same set of measures was applied to multiple product areas and results were evaluated at the organizational level. It also was the first time that systems engineering attributes (requirements, etc.) were directly tied to program performance metrics (actuals and ETC recorded using the EVM systems). It encouraged managers to analyze the effect in a quantitative manner and examine the root causes.

Over time, we observed the improvement of productivity in multiple product areas — perhaps due to the attention given by the data collection and causal analysis activities and the secondary effect of the measurement process. During the implementation of these metrics, we learned and matured the application methods, which we share in this section.

As with any measurement, the key is consistency. Practically, this boils down to consistency in collecting or estimating total systems engineering effort and assessing system size. It's not always clear what is included in systems engineering and a policy must be put in place to guide data collection [Wang, et. al, 2009]. We created a standard, three-dimensional construct of work breakdown structure, organization breakdown structure, and project life-cycle phases for data collection. We provided training and specific guidelines on the scope of each engineering function based on the "task-ownership" policy. In this way, we were able to better enforce consistency in collecting systems engineering effort across multiple programs.

Several considerations must be given to counting COSYSMO size drivers, as subjectivity can easily skew the calculation of system size. The following aspects have proven to be the major factors in ensuring consistency:

The four COSYSMO size drivers — system requirements, system interfaces, system-specific algorithms, and operational scenarios — all are system-level attributes. It is important to define what the "system-level" is. We adapted an ISO-15288-based model and defined system-level to be at the sell-off level of the prime-mission product (PMP) at the system delivery, which is not necessarily the same PMP of the overall end-user system (which can be at a higher level in the overall system hierarchy). A direct implication to this policy is that we must group like programs for productivity comparison. Putting a tier-2- or tier-3-level system with a tier-1 system would easily skew the measurement.

COSYSMO v2.0 adopts a reuse model in counting the size drivers [Wang, et. al, 2008], as the systems we build today are rarely from "scratch" but more likely an evolution of previously developed systems. The reuse model takes into account an amount of reuse in terms of systems engineering activities and divides the four size drivers into five categories of reuse. They are:

- *New*: Elements that are completely new.
- *Modified*: Elements that are inherited but require tailoring or interface-level changes and verification and validation testing.

- *Deleted*: Elements that are removed from a system, which requires tailoring or interface-level changes and verification and validation testing.
- *Adopted*: Elements that are incorporated unmodified, but require verification and validation testing. Also known as "black box" reuse.
- *Managed*: Elements that are incorporated unmodified and require no added systems engineering effort other than technical management.

To further ensure consistency, an activity-based framework is provided to guide the driver-classification exercises, where the five reuse categories are evaluated against six system-level processes, as shown in Figure 2. Given a driver, it is classified into the respective reuse category depending on the processes it goes through. For example, a system interface is classified as "*Modified*" if it incurs all the engineering activities except implementation or architecture-level changes, while as an operational scenario is considered "*Adopted*" when it is inherited and only goes through requirement definition and V&V testing steps but skips the entire system analysis and design, as well as implementation activities. This classification framework proves to be effective in reinforcing the required level of consistency across projects and individual estimators.

	Tech. Management	Requirement Definition	System Analysis & Design	Architecture & Implementation Change	Tailoring / Interface Change	V&V Testing
New	√	√	√	√	√	√
Modified	√	√	√		√	√
Deleted	√	√	√		√	√
Adopted	√	√				√
Managed	√					

Figure 2 — An activity-based reuse framework for classifying COSYSMO size drivers.

As we stated in the previous section, the composite effort multiplier, or CEM, is an important factor in compensating for differences in project environments and system complexities. However, our implementation of COSYSMO v1.0 has revealed a critical deficiency of the original model. The cost-driver ratings have an over-dramatic effect on the estimated effort, such that they unreasonably skew the results [Wang, et al, 2008]. We have modified the model parametric relationship and changed the CEM with the following expression:

$$CEM = \left(\prod_{i=1}^{8} AF_i \right)^{1/8} \cdot \left(\prod_{j=1}^{6} TF_j \right)^{1/6} \qquad (11)$$

Where CEM is a product of two geometric means. The first is more than the eight application factors and the second is more than the six team factors. The modified relationship has brought the expected impact of the 14 cost drivers into a reasonable range.

We have developed a number of internal tools for data collection and analysis. They include a standard data-collection tool to capture data cross the company, an SQL database for data storage

and preliminary data processing, and a number of normalization, calibration, and plotting tools for semi-automatic data processing and plotting.

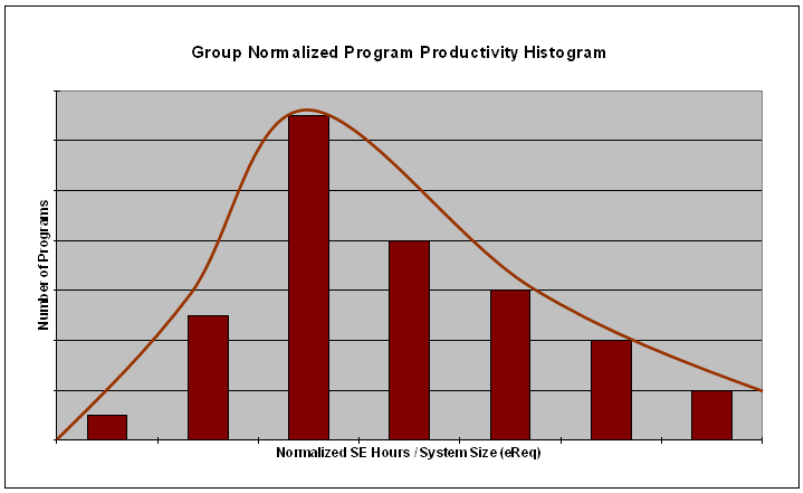

Figure 3 — Histogram of SE productivity measures for a group of selected projects represents a snapshot of the distribution at one point in time.

The recorded data is reviewed in tabular form or in plots for management briefings. One useful plot is the histogram that can be graphed with bar charts, as shown in Figure 3. The histogram is typically a bell-shaped curve and represents program concentrations in different productivity ranges. Each histogram is a snapshot of one point in time. The histogram representation is convenient and gives one a direct sense of where the mode is and how tight the distribution is. It is easy to derive quantitative statistics such as mean, mode, and standard deviation. By observing the mode's movement between measurements, we get a gauge on the productivity trend. The change in standard deviation over time also indicates the group's productivity-variance trend.

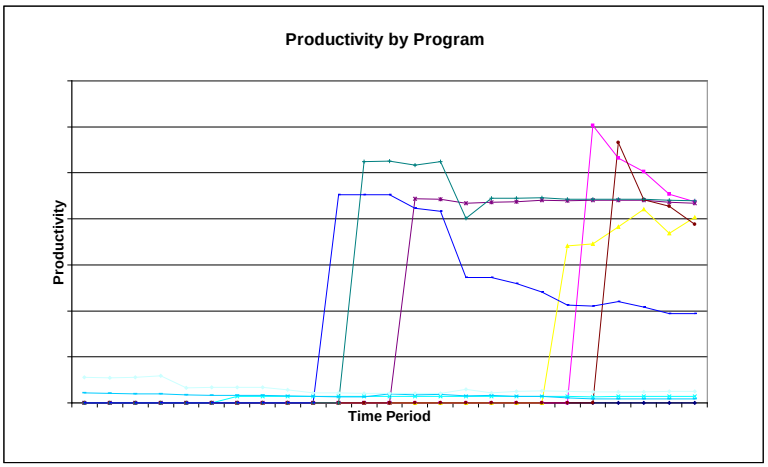

Figure 4 — Time traces of SE productivity measures for select projects, displaying the history of project evolution and maturity.

Another useful display is to plot the trace of productivity over time. Figure 4 shows an example of such a display, where each trace represents the history of a single project or program. These

"time traces" are intuitive and readily show the individual and relative behavior of projects in progress and the productivity changes. When multiple projects are overlaid together, as in Figure 4, one can observe the group behavior of multiple projects over time and access the similarity and dissimilarity of these projects in terms of maturity and productivity trends. However, it is best to align project trends by technical milestones rather than by calendar time for better comparison of project maturity.

Practical Lessons Learned

There are number of practical lessons learned from our implementation of the proposed SE P&E measures, which are:

- Apply the measures to a select group of projects throughout the development life cycle. Measure them periodically — for example, on a three- or six-month cycle, or at the significant technical milestones such as System Requirements Review (SRR), Preliminary Design Review (PDR), Critical Design Review (CDR), Test Readiness Review (TRR), etc. Measuring at the milestones has the advantage of comparing programs at similar levels of system maturity.
- When using estimate at completion for total SE hours, be sure to estimate using a different model or method from the COSYSMO model applied to measuring system size. Otherwise, the derived measures using the same model simply follow the calibration curve and, therefore, are not meaningful.
- Compare like-projects only. As explained in the previous section, comparing projects of different natures (HW-intensive or SW-intensive) can have the effect of "comparing apples and oranges" and may yield misleading results.
- The productivity metrics are best applied to system-development projects, for which the required COSYSMO-defined technical attributes can be readily captured and the model is best-suited. The model is less-relevant to operations and sustainment programs.
- Expect scattering of data, especially at first. But if you apply the model (and parameter definitions) consistently, the data should converge over a few periods.
- Training is critical and can mitigate the risks of inconsistency. As stated before, consistency is of the utmost importance in measurement. A good foundation for collecting any and all data is an investment well-spent.
- It is helpful to have the same reviewer (or reviewers) who understands the model and provides consistent guidance in collecting the data, especially in assessing system size, for all monitored projects.
- Use measured data constructively to help projects improve. Use it as the catalyst for additional in-depth analysis of what is happening. It is not conclusive. Do not use it as a "whip" or a label ("green," "yellow," or "red" programs, for example). Do not discourage participation of projects. Separate and protect the identity of an individual project's data, recognizing its sensitivity when compared to other projects in the company. Look at global behavior rather than individual projects.

Conclusion

"If you can't measure it, you can't manage it," as stated in a management mantra. The proposed systems productivity and efficiency measure represents the first independent metric for systems engineering as an engineering discipline. It bridges an apparent gap and provides a potentially effective enabler in our ever-enduring quest to produce more output (or provide more added value) at lower costs or to improve productivity and efficiency.

In this paper, we proposed a set of systems engineering productivity and efficiency measures, including definitions and barriers to achieving consistent measurement data. We include a discussion of system size as a key piece of measurement data to support productivity and efficiency measures. We also provide practical advice on implementing these metrics to measure multiple system-development projects in an organization. Application of these metrics and the necessary consideration for reuse within projects also is discussed. Some key lessons learned from our implementation experience also are provided.

This effort is still in the early phase of discovery and evaluation. There are several areas for possible improvements. One of the known deficiencies of this model is that it still supports the Waterfall development process only and assumes a pre-deterministic set of requirements and system architecture. There are no effective extensions for Agile models, for which some premises do not hold. This is rooted in the nature of COSYSMO, which uses Waterfall as its basic process. As more programs require Agile Development, such extensions soon will become necessary.

It would also be an interesting and useful exercise to explore ways of measuring the added value of a product by systems engineering for the "value-added" view, as mentioned in the introduction section. If we can quantify the effect upon a product, whether in quantity, quality or functionality, that other engineering disciplines produce in a normalized manner, we can establish the baseline for capturing the change to the product or the value enhancement. It would then open up the doors to meaningfully measure productivity and to better quantify the ROI for systems engineering from the "value-added" perspective.

While still at an early stage, the proposed measure provides an alternative to engineering leads and project managers to better gauge the effort of systems engineers and to more effectively manage the activities and resources. As a profession, systems engineering could use more compelling quantitative evidence to justify its value in this ever more competitive marketplace.

References

IEEE Standard for Software Productivity Metrics, *IEEE Std. 1045-1992*, IEEE Standards Board, 1993

INCOSE Systems Engineering Handbook, version 3.1, INCOSE, 2007

Boehm, B., *Software Engineering Economics*, Prentice Hall PTR, Upper Saddle River, NJ, 1981

Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D. J. and Steece, B., *Software Cost Estimation With COCOMO II*, Upper Saddle River, NJ: Prentice Hall, 2000

Boehm, B., *Some Future Trends and Implications for Systems and Software Engineering*

Processes, *Wiley InterScience*, October 2005

Boehm, B., Valerdi, R., Honour, E., The ROI of Systems Engineering: Some Quantitative Results for Software –Intensive Systems, *Wiley InterScience*, December 2007

Card, D. N., The Challenges of Productivity Measurement, *Proceedings: Pacific Northwest Software Quality Conference*, 2006

Coelli, T., Rao, D. S. P., Battese, G, An Introduction to Efficiency and Productivity Analysis, *Kluwer Academic Publishers*, 1998

Kasunic Mark, A Data Specification for Software Project Performance Measure: Results of a Collaboration on Performance Measurement, *Software Engineering Institute*, July 2008

Valerdi, R., *The Constructive Systems Engineering Cost Model (COSYSMO): Quantifying the Costs of Systems Engineering Effort in Complex Systems*, VDM Verlag, 2008.

Wagner, S., Ruhe, M., A Systematic Review of Productivity Factors in Software Development, *2nd International Workshop on Software Productivity Analysis and Cost Estimation (SPACE 2008), Technical Report ISCAS-SKLCS-08-08*, State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, 2008.

Wang, G., Valerdi, R., Ankrum, A., Millar, C., Roedler, G., COSYSMO Reuse Extension, *Proceedings of 18th INCOSE International Symposium*, June 2008.

Wang, G., Valerdi, R., Boehm, B., Shernoff, A., *Proposed Modification to COSYSMO Estimating Relationship, Proceedings of 18th INCOSE International Symposium*, June 2008.

Wang, G., Valerdi, R., Roedler, G., Ankrum, A., Gaffney, J. E., Harmonizing Systems and Software Estimation, *Proceedings of 19th INCOSE International Symposium*, 2009.

Wang, G., Shernoff, A., Turnidge, J., Valerdi, R., Towards a Holistic, Total Engineering Cost Model, *Proceedings of 19th INCOSE International Symposium*, 2009.

Biographies

Gan Wang, Ph.D., is an Engineering Fellow at BAE Systems. He has been engaged in the research and development of systems engineering processes, engineering cost-modeling and life-cycle cost estimation, decision support methods, and system-of-systems engineering and management methodologies. Prior to joining BAE Systems, Dr. Wang spent many years developing real-time geospatial data visualization applications for mission planning and rehearsal, battlefield management, command-and-control (C2), flight simulation, and aircrew physiological training systems. He also developed control systems and aircraft simulation models for various man-in-the-loop flight training systems. He has more than 20 years of experience in systems engineering, software development, research and development, and engineering management involving complex, software-intensive systems.

Lori Saleski is a project engineering manager at BAE Systems, where she has been working on the development of the COSYSMO-based system engineering model and process and the total engineering estimation model and process over the past 3 years. She is an engineering manager for mission management programs. Lori has worked in software engineering for 25 years and holds a bachelor's degree in computer science with a mathematics minor from the University of Maine.

Alex Shernoff is a systems engineer at BAE Systems. He has been working on the deployment and implementation of COSYSMO at BAE Systems to localize it to organizational platforms and processes and to support systems engineering productivity and efficiency analysis. Prior to this effort, Alex assisted in the planning and logistics of BAE Systems' continuing support for the annual INCOSE Conference. Prior to joining BAE Systems, Alex graduated with a bachelor's degree in electrical engineering from The Ohio State University. He specialized in high-voltage power and electromagnetism.

John C. Deal serves as the Vice President for Maritime C4I Programs in Defense Systems & Solutions, Information Solutions. Prior to his current assignment he served as the Vice President of Systems Engineering in the E&IS OG, and as both Director for C4ISR Systems Engineering and Program Director for WIN-T, AMF JTRS and other defense agency programs within the Communications and Information Systems Business Area of the Communications, Navigation, Identification and Reconnaissance Division, BAE Systems, Inc. John served in the US Army for 29 years retiring as a Colonel. His final active duty assignments included Commander, U.S. Army Information Systems Engineering Command, and Executive Officer, Office of the Director of Information Systems for Command, Control, Communications, and Computers, Army CIO, Headquarters, Department of the Army. He was selected and served in two senior level fellowships; as a member of the Secretary of Defense Strategic Studies Group II and as a member of the Ridgeway Center for International Security Studies, University of Pittsburgh, PA. John holds a Bachelor of Arts degree in Physics from the University of Alaska, a Master of Science degree in Electrical Engineering from the Naval Postgraduate School, a Master of Arts degree in National Security and Strategic Studies from the Naval War College, and a Master of Arts degree in International Relations from Salve Regina University.