
Systems Engineering: Three New Approaches

by Dr. Richard P. Evans

This paper describes three new systems engineering approaches: System Assessments, a Systems Integration (SI) program, and an Engineering Baseline System (EBS).

Some of the key features reported for System Assessments are an Assessment Control Board (ACB), as a critical complement to the traditional Configuration (or Change) Control Board (CCB), with associated one-page Assessment Plans (APs), and one-page Assessment Reports (ARs).

Primary characteristics of a Systems Integration (SI) program include the continuous acquisition of non-attribution System Reports (SRs); the structure of Candidate Program Initiatives (CPIs) for intermediate system planning; the application of small (3-5) consolidated customer/user/stakeholder and developer engineers in composite, non-attribution, altruistic Problem Area (PA) teams for system engineering review, and the use of Round Tables of participants in extracurricular roles, like INCOSE referees, to provide structured assessment support.

The Engineering Baseline System (EBS) addresses the opportunities/problems introduced by the recent widespread use of personal computers by engineers, the attendant separate and typically uncontrolled and non-standard structuring and naming of file-based system elements, and the accompanying associations, as new adjuncts to what had been exclusively a page-based environment. That uncontrolled and non-common creation and use of multiple separate file-based environments, and accompanying associations, brought on a loss of the standardized structure, naming, and change management that was previously maintained by the page-based environment—with its fixed and controlled page structure, page number, and page date.

The EBS paradigm includes standardized (thus common) system element structuring and naming (by a six-digit system number—that is a sequence number to sustain audits—and that has a six-digit suffix to support the assignment of unique system numbers to that span of separate system files. A system number has the format <<xxxxxx.yyyyyy>>. The xxxxxx prefix is a sequence number that is unique within the file or system component where the system element is maintained; and the yyyyyy suffix identifies that file. When a system element in a file changes, the next available system number prefix within that file is assigned; the suffix is fixed. All previous system numbers (prefixes) associated with a given system element are retained.

System numbers are unique for each system element, including specification elements, software code, drawings, and hardware elements. System numbers and associated tags, maintained in separate two-column ASCII-based index files, can be assigned by system developers when they create new system elements, without using specialized tools, or they can be assigned using database or CASE tool systems.

Added EBS features include the use of plain ASCII two-column index files for all manner of associations—even between code modules and user manuals—prepared, used and created by any and all engineers, anywhere, any time, for any reason—in contrast to the use of a central specialty database system.

That EBS element addresses shortcomings in the file-based approaches that are typically present in the current CASE-type environments. These approaches, while overcoming some of the page-based issues, but nevertheless based on the use of a few large, specialty tools, have also created problems and limitations of their own—particularly in the limited scope of those who are able to effectively participate.

Systems Engineering Principles

Three new methodologies are presented as systems engineering *approaches* in order to affect a shift from simply *engineering* to *methodologies for engineering*.

An example of one of the advantages of such a *transformation*—like a Laplace or Fourier transformation in addressing signal processing—is the following passage by Tully (1989) and Thome (1993), that is also illustrated in Figure 1, on the topic of systems engineering:

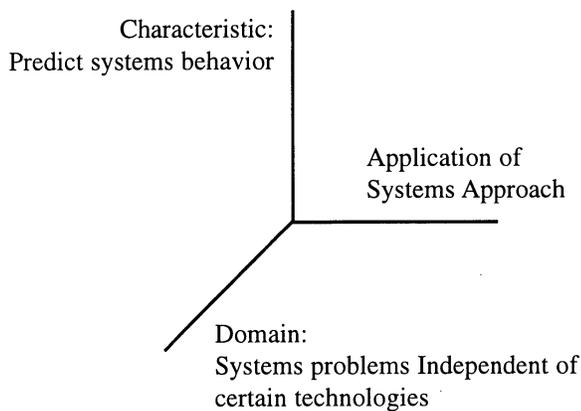


Figure 1. Systems Engineering—Three dimensions.

Systems Engineering consists of applying a systems approach to the engineering of systems. Its domain is the engineering of solutions to systems problems independent of employing a certain technology for realizing systems functions and properties. A characteristic of systems engineering is that it has to predict systems behavior and to design systems structure so that emergent behavior can be provided for and controlled within acceptable and desirable bounds.

In that approach, the authors address systems engineering along three separate dimensions that are more amenable to understanding and insight, as they transform in a sense from only *engineering* per se. That *transform* approach enables a separate consideration of each of the three dimensions, rather than addressing *engineering* as a whole. In the case of the

dimension of *application of the systems approach*, for example, the transform effects a shift from the topic of engineering, to a separate consideration of the systems approach.

The authors, who also cite (Jenkins 1969 and Churchman 1989), then apply the same transform technique in considering the *systems approach* in the context of the following three primary *perspectives*, attributes of *systems thinking*, or *ways of thinking* about the engineering of computer-based systems, as depicted in Figure 2. The effective use of a three-dimensional framework for describing systems engineering and its various facets is similarly cited by Sage (1992), Hall (1969), and Warfield (1972).

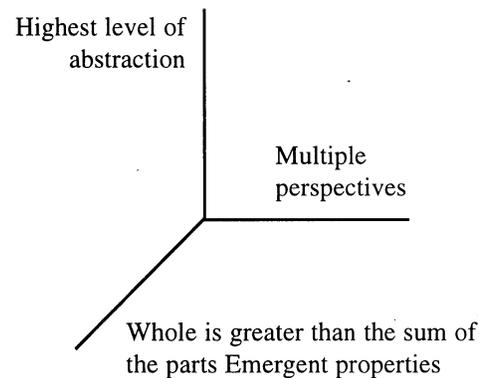


Figure 2. Systems Approach—Three dimensions (Perspectives, Ways of System Thinking)

The three new systems engineering approaches depicted in Figure 3, and presented in further detail in Sections 1, 2 and 3, respectively, are elements in this framework of systems engineering.

System Assessments and an ACB

Systems engineering approaches typically include a basic program control board known as the Configuration (or Change) Control Board (CCB). CCBs act after-the-fact in the sense that they receive formal change proposals in specific formats, some of which may have been in preparation for months. An Assessment Control Board (ACB) serves as a complementary and contrasting control board.

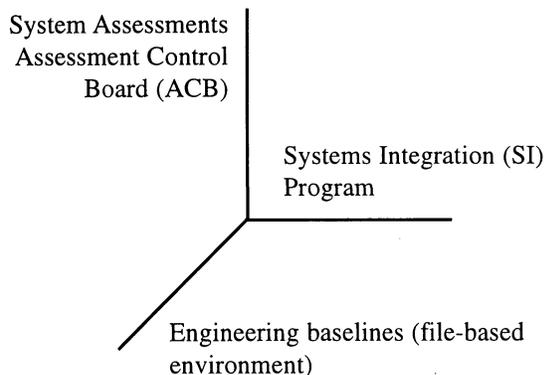


Figure 3. Three New Systems Engineering Approaches.

There is a need for both assessment control (ACB) and configuration control (CCB). Assessments make discoveries, a CCB disciplines the application of those discoveries. An ACB anticipates and plans, it operates up-front—that increases its leverage; a CCB operates after the fact and regulates. While CCBs are essential for change and implementation control, there is an equal need for the balance of assessment. An ACB, in contrast and as a complement to a CCB, is focused on the plans for the initiation of work, with a concentration on the plans for its assessment. In that sense, an ACB is focused on proposed plans and process, in contrast to a CCB emphasis on details of proposed change and the control of its implementation.

As illustrated in Figure 4, an ACB complements a CCB by exercising control of the initiation of work, including trade studies that lead to proposed changes for CCB consideration. The control of work initiation by an ACB includes the plans for, and the results of, the work assessment. An ACB focus is on assuring the operation of ACBs at all levels of the engineering effort, not just at the program office level. An ACB's goal is to assure a whole set of ACBs so that every engineer has the privilege to undertake their labors in the context of an Assessment Plan approved at an appropriate level by those to whom they also have the opportunity to provide reports of its application efficiently and effectively.

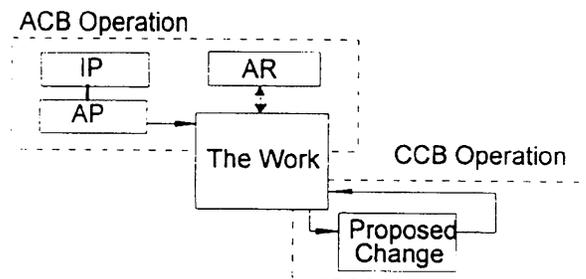


Figure 4. ACB and CCB Operations.

The initiation of work is controlled by an Initiation Plan (IP) approved by the ACB. An attachment to the IP is the one-page Assessment Plan (AP). Assessment results are similarly reported in typically one-page Assessment Reports (ARs). APs typically address the following:

Scope: The work and the associated products to be assessed.

Assessment Criteria: The criteria to be applied in assessing the work and the products. This is one of the hardest elements of a plan to devise, and accordingly one of the most critical program controls.

Approach: How will the assessment itself be assessed, how will the assessment be conducted—the format and process: who will be on the separate/independent assessment team—their names?

Schedule and cost: the assessment milestones and the proposed investment in assessment.

System Integration (SI) Program

A parallel methodology that can be applied to strengthen the CCB is for the ACB to also sponsor an SI program, as a complement to final CCB program control. The objective of an SI Program is to assure that proposed changes are well prepared for CCB consideration. Changes may be changes to the configuration of the program architecture and schedule, as well as a change to the design. There are three primary dimensions of an SI program as illustrated in

Figure 5, Identification, Investigation, and Implementation:

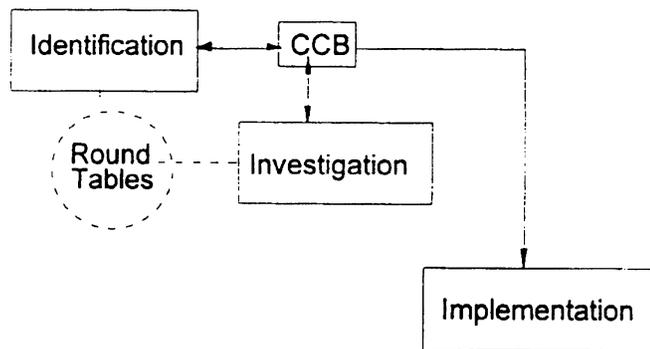


Figure 5. ACB Sponsored and CCB-Controlled SI Program

The driving influence of the SI Program is in the first two “I”s: Identification and Investigation—those that are the most *up-front*. The Investigation process also has, as a central feature, the use of Round Tables (RTs), as a panel of three to five experts, to serve like INCOSE referees as an unfunded assessment team for planned investigations.

The SI Program structure includes four elements: *System Reports (SRs)*, *Candidate Program Initiatives (CPIs)*, *Program Objectives (POs)*, and *Problem Area (PA) Teams*. All are supported, as depicted in Figure 6, by an SI Database,

System Reports (SRs) are individually numbered records of every problem, suggestion, insight, or idea. An SI Database is built on the ever-accumulating set of SRs maintained throughout the life of the system. SRs are recorded as symptoms, so to speak, without prejudice. They are not filtered by any criteria, such as who said, or how they were reported, or whether they were validated. They are accumulated and honored by a unique SR Number that is never reused. Thus, while the SR may be placed in an inactive file, its identity, its number, always remains unique to that SR.

Problem Area (PA) teams assess the overall program handling of the SRs. The team members are drawn from both the user and the developer. They serve as professional collateral assignments, not as

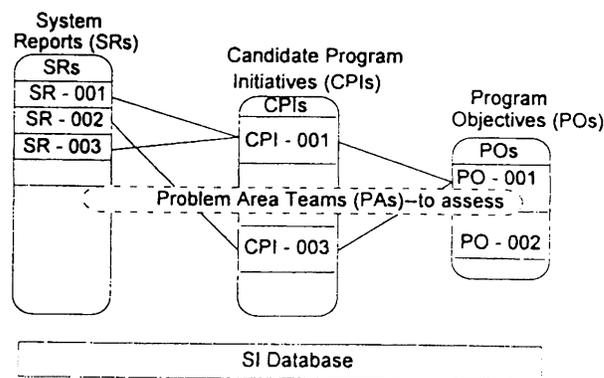


Figure 6. Four-part SI Program.

representatives of their parent organization’s management priorities or interests. The PA teams assess; they do not have responsibility for solutions. They recommend initiatives, but they do not sponsor changes—with the attendant responsibility to implement approved changes. The PAs monitor the process design and operation.

Candidate Program Initiatives (CPIs) are temporary homes for potential program initiatives. CPIs are unfunded and do not have a designated management responsibility. They are the initial planning framework, a neutral territory, for the allocation of SRs. Note that SRs are allocated redundantly, with one primary allocation and multiple secondary assignments.

Program Objectives (POs) are funded, have assigned implementation responsibility, and are the formal vehicles for configuration change. POs are assembled as the implementation packages from the array of CPIs. They may be one entire CPI or include portions of many.

Engineering Baseline System (EBS)

The EBS methodology provides a new paradigm for system element identification, application, association, and control in the engineering of computer-based systems. Prior to the increasingly widespread use of computers by all engineers, system elements were only defined and controlled in a *page-based*

environment where the *page* structure, number, and date established system elements. With computers now available to, and in use by, essentially all engineers, a *file-based* environment is being added to the *page-based* foundation. The added *file-based* capability has both new promise as well as new risk; the EBS methodology addresses both. The EBS paradigm capitalizes on the file approach while addressing shortcomings in the typical CASE-type approaches to a *file-based* capability. Those systems, based on the use of a few large, central, specialty tools have, while overcoming some of the page-based-only issues, also created problems and limitations of their own. EBS features include:

1. *File-based engineering baselines*: prepared and controlled day-by-day in a distributed management framework.
2. *Standard common structure* of all system elements: controlled and defined at the basic primitive level as stand-alone, machine-processable elements. These are *file-based* structures that are structured *from* the *page-based* foundation.
3. Centrally assigned blocks of standard-format six-digit (auditable) *system numbers* that are maintained automatically in strict journal number sequence for every system element—whether requirements specifications, designs, test cases, maintenance documents, code modules, hardware components, budget elements, schedule milestones, or user manuals.
4. *Engineering baseline (eb) numbers, and engineering change (ec) numbers*, with associations to system numbers maintained in plain, two-column, ASCII index files for each primitive system element.
5. *Plain ASCII two-column index files* for all types of associations that are prepared, used and created by any and all engineers, anywhere, anytime, and for any purpose. These contrast to the use of a central specialty database system. EBS index files, prepared as individual two-column ASCII files, are thus not only amenable to being aggregated into larger sets of other plain ASCII files,

they may also be aggregated into centralized, specialty, database-oriented software packages. Therefore, while not in any way constraining the use of specialty database-oriented tracing approaches, the index files actually enable them by enabling wide preparation and use outside of, and thus in support of, central database-oriented systems. On the other hand, using only specialty software applications, rather than *ASCII index files* to create as well as maintain associations, restricts visibility into those associations to either hard copy tables, or by direct use of the specialty software that created the table. Individual *index files*, however, remain visible to any and all for use, modification, extension, and review, and on any machine, and simultaneously also provide the needed inputs for a central database repository or report generator, as may be desired.

Problem Areas—Criteria for EBS Methodology

Evaluation: The problem areas in current practice for the engineering of computer-based systems may be summarized in the following top-ten set of inter-related attributes. They are separate, but, as shown in Figure 7, they aggregate along three dimensions of system engineering needs (those that support, enable, and sustain all three dimensions are listed at the focus of the three axes):

- **Associations**—paired linkages of system elements.

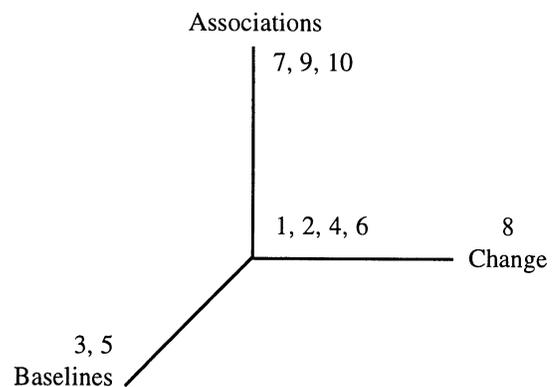


Figure 7. Three dimensions of systems engineering need as addressed by an EBS.

- **Change management support**, the identification and recording of changes, the associated rationale, and the specifics of new, changed and deleted system elements.
 - **Engineering baselines**—multiple controlled baselines, maintained in distributed management environments, by, for and of the engineering.
1. Structure and granularity—common structure and system element number
 2. Autonomy—stand-alone system elements
 3. Timeliness—controlled engineering baselines as needed
 4. Machine processability—ASCII files of systems elements and paired association
 5. Distributed management—engineering groups with control of their own baselines—yet all integratable
 6. Auditability—system numbers as sequence numbers
 7. Self-rule—creation of paired association index files on the spot
 8. Independence—non-dependence on hard copy only change definition
 9. Aggregations—integration to one common database of separately controlled files—enabled by suffix block allocations
 10. Associations—integration of all associations—ASCII paired index files
1. **Structure and Granularity:** The need is for controlled standardization of structure and naming/numbering to the lowest level; individual, uniquely numbered system elements that can also be separately processed in machines; CASE environments. Current ECBS controls are typi-

cally applied solely to formatted pages that are not machine processable without uncontrolled changes in structure. Current control practice also uses a framework of sections, such as 3.2.4.2.6, that often span sets of many otherwise separate requirements, specifications, and design elements. Further, current practice generally employs compound statements and bulleted and tabular data that are thus neither lowest-level system elements nor autonomous and stand-alone, as discussed below.

2. **Autonomy:** The ECBS methodology need is for stand-alone (as well as granular) system elements, that carry, with their unique name/number, all associated context and also the associated system/management information, including changes, allocations, associations/integration, and other system associations.
3. **Timeliness:** Effective engineering typically needs controlled *file-based* engineering baselines day-by-day. Current ECBS practice generally only provides formal *page-based* controlled baselines, and at *release* intervals that often span months, even years. Individual engineering activities usually need day-by-day controlled baselines for the interactions among their personnel, who are daily working on many tentative what-if type alternative assessments, designs, trade-offs, and other systems engineering considerations. They need day-to-day *engineering* baselines that are typically controlled among themselves. While those baselines are not the final contract type baselines that are eventually formally established by a CCB, equally formal control within their particular engineering activity is needed by them as they conduct their own iterative assessments and planning: the engineering.
4. **Machine processability:** The need is for system descriptions, whether specifications, designs, hardware components, software modules, etc., in ASCII non-formatted files—without dependence on features that are not machine-processable in ASCII files—such as tables, graphics, footnotes,

endnotes, italics, bold and indents. Tabular data is particularly susceptible to lack of machine processability as well as the attendant loss of automated auditability and change control. The same or similar data are often included in a variety of tables, with differing scope, format, and content. Thus change control, and even interface control, are difficult, if not precluded altogether. A controlled change to one table is not readily carried over to the needed changes in other tables as well as non-tabular system elements that address similar data, but in different formats and contexts.

5. **Distributed control:** Each engineering activity/organization needs to be enabled and responsible, to maintain a separate set of their own engineering baselines, yet integratable into a system whole. Current practice typically limits the authorization to establish baselines to a few centralized personnel using a large and unique specialty CASE tool or database.
6. **Auditability:** Names/numbers are needed that are centrally controlled, in a standard format (six digits) and strictly sequential—so that any missing or redundant number is clearly visible. Current ECBS practice relies on numbering/naming of system elements only by sections. They may include as many as 50 separate stand-alone system elements, with variable size numbers such as 3.4.2.1.3.7, and without separate, individual system numbers, of a fixed size number of characters such as 000357. In that framework, the only available names, for each system element is, for example, neither specific to each separate system element, nor is it a sequenced number to support audits. It is never assured, for example, that 3.1.6 would immediately follow 3.1.5.3.7.2; thus numbers may be missed. A sample of that inadequate *page-based* approach, along with its other association deficiencies, is presented in Table 1.
7. **Self-rule:** All engineers need to be both enabled as well as responsible to establish and maintain associations and dependencies—using the stan-

dard six-digit name/number—for all system elements they create and use. Current practice typically limits the establishment of associations to those entered by a few centralized personnel using a large and unique specialty CASE tool or database.

Document.	Function	Associated Segment
3.1.7.2	Provide on-line help	3.2.2.5 3.2.6.2.2

Table 1. *Sample Page-based (Non-EBS-based) Traces.*

8. **Independence from page-only change control:** Association of change data in each granular stand-alone name/number is needed. Current change management is typically based solely on change pages, without change definition embedded (by index files) with each separate stand-alone system element. Current controls are typically applied solely to formatted documentation that is not machine processable without uncontrolled changes in structure and associated change history.

As possibly one of the most significant benefits of the EBS paradigm for the engineering of computer-based system, change information is explicitly established and recorded for each system element, and that is maintained in individual machine-processable files and the associated two-column ASCII index files.

In the present practice on several large-scale systems currently in development, a major deficiency exists in the processing of formally approved changes, called RFCs, for Requests for Change. RFCs are allocated in composite sets to new *Versions* of formally controlled specifications, designs, budgets, schedules, test plans, installation manuals, etc. Each *Version* or *Release*, typically issued only after months of review by a CCB, normally includes several RFCs, with each RFC containing up to 10 pages, and with as many as 20 separate changes (system elements) per page. The RFCs are not structured to

primitive system elements for machine processing, and there is no unique identifier (like a system number) for each such basic change element. Further, there is no association index file (two-column paired associations between system numbers) for the individual changes in each RFC and each new revised system element in the composite *Version/Release*.

That deficiency is aggravated when the engineers remove the new information from the *pages* and enter them into machines. At that point, the non-association is compounded by the loss of the *page* date, number and structuring.

The following is an example of both the EBS approach to automatically recording (in two-column index files) changes to a given system element, and typical optional *display* formats. All system element information, including descriptions and index-file associations, is maintained in individual two-column index files. But various displays, such as the following sample, may be generated with various data aggregated on a page/report. In this case, para 3.2.1 was structured from the original in the *page-based* environment into two system elements in the *file-based* environment. Each was assigned a separate system number: 000002 and 000003, respectively. The second of those elements was altered by an engineering change (ec) action designated as <ec0827>. Please note that ec's may refer to formal RFCs or to any other controlled change process—especially those operated by the engineering staff as interim what-if changes. In the process, the engineering baseline (eb) increased from <eb0002> to <eb0003>. In addition, that new element was assigned the additional system number of <<000643.900001>>. Please note that the 2. <n2885> are for file ID (line number) “2”, in the file named <n2885>.

2. <n2885> <eb0002> 3.2.1 The segment shall provide communications with the network through the Front End (FE) <<000002.900001>>.
2. <n2885> <eb0003> <ec0827> 3.2.1 The segment shall provide communications with the network through the Back End (BE) <<000003.900001>> <<000643.900001>>.

9. Aggregations: Use of system numbers with a six-digit suffix is needed to enable distributed baseline generation and control—yet integration (no conflicts in system numbers) into a single program database—aggregation of all management information into composite database sets of any needed scope. Allocation of “blocks” of suffixes (the “y”) sustains this need: xxxxxx.yyyyyy

10. Associations—integrations: Each separate system element needs to be associated with all other related system elements by reference to its standard and unique six-digit system number/name—in paired ASCII index files—that engineers create without reference to any database.

Summary

System Assessment: An Assessment Control Board (ACB), with a focus on Initiation Plans (IPs), their associated one-page Assessment Plans (APs) and Assessment Reports (ARs), can be essential complements to CCB operations. CCBs are essentially totally after the fact. The resources (both time and money) to prepare proposed changes for CCB consideration have generally already been invested by the time the CCB receives the results. The operation of an ACB is *management working up front*, where the leverage is greatest. The use of IPs, APs and ARs at all organizational levels, operated in essence by increasingly lower-level ACBs, is a key feature of the ACB approach. The ACB influence of how work is to be assessed is a prime lever on what is done. The criteria for goodness and the names of those who will prepare assessment reports are key areas for management influence.

Systems Integration (SI) Program: Operation of an engineering planning process, as an SI Program, based on SRs as the primitives for all planning, is a potential added aid that the ACB can sponsor as a further complement to the CCB. An SI Program uses bipartisan Problem Area (PA) teams that include both customer and developer members. The PAs, working on a low duty cycle, an hour a week or less, concern themselves with the planning to address their assigned SRs.

Engineering Baseline System (EBS): Engineering baselines represent a new and needed paradigm for controlled visibility and traceability in systems engineering. The EBS addresses the opportunities and problems introduced by the recent advent of personal computers in use by all engineers and the attendant separate and typically uncontrolled and non-standard structuring and naming of *file-based* system elements and associated associations as new adjuncts to what was previously exclusively maintained as a *page-based* environment.

The EBS is more a methodological framework than a toolset. The EBS software is but one implementation of the approach. It exists to make real the principles; but the idea, the approach, and the concepts are essential. Not only is most of the so-called EBS conducted outside of any special software (engineers creating their own two-column index files in any type of tool—including paper and pencil), system development firms may quite readily construct their own software implementations of an EBS, once they appreciate and determine to employ the principles.

Acknowledgments: While there have been many who have assisted in the development of the methodologies described here, this article is the sole responsibility of the author, it does not reflect the views or opinions of any organizational affiliations.

References

- Churchman, C. W. "Der Systemanatz und seine Feind," Translated from the American "The Systems Approach and its Enemies," commented and introduced by Werner Ulrich, Verlag Paul Hapt, Bern and Stuttgart, 1981.
- Hall, Arthur D., "Three-Dimensional Morphology of Systems Engineering," IEEE Transactions on Systems, Science, and Cybernetics, Vol SSC-5, pp 156-160, Apr 1969.
- Jenkins, G. M., "The Systems Approach," *Journal of Systems Engineering*, 1 (1969) 3-49.
- Sage, Andrew P., *Systems Engineering*, John Wiley & Sons, New York, 1992.
- Thome, Bernhard, (ed.), *Systems Engineering: Principles and Practices of Computer-based Systems Engineering*, John Wiley & Sons, New York, 1993.
- Tully, C. J., "Position Statement on Systems Engineering," ATM/WP4.4/CJT7/Issue 1, 12 December 1989, Commercial in Confidence.
- Warfield, John N., and Hill, Douglas J., *Unified Systems Engineering Concept*, Battelle Memorial Institute, Columbus, Ohio, 1972.