BACK TO BASICS

Brian W. Mar
Department of Civil Engineering
University of Washington
Seattle, WA 98195

Abstract. Anyone concerned with establishing world class systems engineering for the 21st Century needs to understand the basic concepts of systems engineering and to use these concepts in their practice of engineering. Too often, individuals involved with systems engineering focus on selected tasks and forget the basic concepts and purpose of systems engineering. A set of rules-of-thumb for system engineering practice are explored as a point of departure for the papers to follow.

INTRODUCTION

If the basic concepts of systems engineering are not fully understood by the work force, the practice of world class systems engineering in the 21st Century will be difficult to realized. Engineering in the 21st Century must be able to cope with (1) rapidly changing technologies, (2) many existing systems that must be modified rather than replaced, and (3) engineering disciplines that continue to fragment. In this paper I will draw from my observations and experiences and attempt to identify problems that can be traced to lack of understanding of basic systems engineering concepts.

SYSTEM ENGINEERING BASICS

Systems engineering requires (1) a structured and disciplined process that defines problems before seeking solutions, (2) a systematic search for solutions that examines tradeoffs between alternative solution sets, (3) a traceable and disciplined integration process that verifies that the product system meets the original requirements and performs the needed functions, and (4) an effective information management system that can provide each team member and the customer with any information concerning the system that has been generated. While DSMC (1983), Alford(1991), and Mar(1991) have discussed these basic concepts, many systems engineering texts concentrate on simulation, optimization or speciality engineering in their treatment of systems engineering. In practice, the system engineering process is applied throughout the life cycle and individuals tend to be assigned to a particular phase and given a special system engineering task. Soon, these specialist begin to view systems engineering as only their particular task. This is one of the major reasons why there are so many different definitions of systems engineering. The practitioners focus on single tasks and rarely see the whole process.

Many people do not realize that the systems engineering process is not limited to the translation of needs of major systems acquisition agencies into large complex systems. The system engineering process can be applied by anyone to any task. The translation of any problem statement into functional descriptions of the product should be the first step in any engineering activity. The generation of requirements that describe how well each function must be performed, and the identification of constraints that must be dealt with are also common sense engineering practice. Once a person has defined what the product system must do **(functions)** and how well it must do it **(requirements)**, the actual product system architecture or design can be developed **(architecture)**. Mar (1991) has labelled this basic process F-R-A. This simple three step process of defining functions before requirements, defining requirements before seeking answers, and examining all alternatives before selecting an answer is common sense and is a basic systems engineering concept.

The reason why the label systems engineering is used for this process rather than just engineering is that each function, requirement or architecture can be viewed as a system and decomposed into its parts. Decomposing functions, requirements, and architecture descriptions to lower levels with more detail, is an application of systems theory. This concept is that the whole can be decomposed into its parts and that parts should make up the whole. This vertical linkage of parents to children facilitates the tracing of functions to a parent function, requirements to a parent requirement, and architecture to a parent architecture. Another important system concept is that systems can have interfaces with other systems. System engineering involves three interacting systems at each level of detail -- the functional system, the requirement systems, and the architecture system. While the vertical relationship between parent and child are defined within each of the F-R-A systems, the horizontal or interface relationship between these three systems is defined by the F-R-A linkage. Functions are quantified by requirements, and requirements are satisfied by architecture. This simple set of pointers permit any function, requirement, or piece of architecture to be traced to its parent F, R, or A, and the horizontal F-R-A relationship is traced by the serial linkage of F to R and R to A. The failure to use these simple pointers in favor of complex numerical/decimal numbering systems leads to serious tracing problems during the development life cycle.

The selection of answers using the systems engineering process is based upon the systems approach. The basic premise of the system approach is that once the problem is defined, all possible alternatives must be equally evaluated in only enough detail to rank the alternatives. Once the ranking is achieved, the systems approach requires that a new solution be formulated that incorporates the strengths, but not the weaknesses of the better alternatives. The examination of a rich set of alternatives, and the search for a better solution built from the strengths of the original alternatives is often neglected in practice. The point design (considering only one solution) is common practice, where one alternative is examined in great detail and all others are ignored or dismissed with a cursory look.

Searches for solutions can be done randomly, it can be done by examining each potential solution in great detail , or it can be done parametrically. A parametric or trade study approach to problem solving has also been called the system approach. The major difference appears to be the evaluations of single alternative parameters and the consideration of different alternatives.

Another basic systems engineering task is to manage the increasing volume of information describing functions, requirements, and architecture as lower and lower levels of system descriptions are generated. System descriptions will include functional descriptions, design descriptions, product descriptions, and operational descriptions. In addition, records must be kept of each decision and the basis for that decision. Traditionally, the engineering groups participating in a program develop have different document styles and format. It is common to observe the same set of information being reentered into data bases or analyses by each group. This duplication of effort can lead to errors of entry, loss of data, and added effort.

Traditionally, systems engineering outputs have been document oriented. The F-R-A information is organized and sorted in different document formats. The classic systems engineering document is the Specification, where F-R-A data is sorted by parts of the architecture (A) and functions, requirements, constraints, and tests are presented for each of these parts. Prior to the generation of specifications, functional descriptions (F) are documented in functional flow diagrams, N square diagrams, behavior diagrams, and other documents describing the systems functions. The relationship between functions and requirements are often reported in Requirement Allocation Sheets (RAS). RAS documents are later modified to include the identification of configuration items (A) that will perform the functions. 21st Century system engineering will not focus on documents, but the information contained in these documents. Managing the information, and providing automation tool to create any type of document of interest will provide better configuration management of information in documents, and reduce the high costs of document preparation.

The basic concept that two systems, not one needs to be system engineered in any program is difficult to grasp. The activities and tasks associated with the development of system engineering descriptions of the product, are not the same activities and tasks associated with the system engineering of the program that creates the product. In the classic document oriented system engineering process, the information describing the product is contained in the set of specifications, while the information describing the program is contained in the systems engineering management plan (SEMP). The current efforts to modify MIL-STD 499 is a recognition of the difference between specification information and system engineering management plan information.

Another weakness of the document oriented systems engineering process is that all the descriptions are static. The ability to describe the dynamic interactions between any system and its environment, and the interactions between internal system functions will be characteristic of 21st Century systems engineering. Thus, simulation models of the functional as well as the physical behavior of the system must be created to support the assessment of the top-down decomposition of problems and solutions, as well as the bottom up verification of the product performance. While simulation has been a key element of design and staff level engineering, it has not be widely accepted as part of the systems engineering process.

Once the basic F-R-A type information has been generated at any level, management tasks such as risk assessment and management, life cycle cost analysis, scheduling, technical performance measures, specialty engineering can linked and initiated. Each of these task require the basic F-R-A system information as a point of departure.

In summary, the basic system engineering concepts include descriptive activities that translate a customer input or task assignment into a consistent F-R-A knowledge base. Managing the knowledge base (information) and reformating it into customer or other party formats such as specifications, review documents, etc, and provide traceability for impact assessment of proposed changes should be automated. Finally, system engineering provides a common structure and discipline to guide and support the top- down definition and bottom-up synthesis and test activities for any program development.

LESSONS LEARNED

A problem with the current practice of systems engineering is that even those who advocate its practice tend to deviate from these basic concepts because of job pressures, and underestimate the seriousness of such neglect. The following sections examines common heuristic statements (placed in italics) that are encountered in the normal course of systems engineering practice. These statements are organized as they relate to the role of systems engineerings, the role of any engineer, and the implementation of the systems engineering process.

Role of systems engineers

Understanding the basic concepts of systems engineering can help resolve the growing conflicts between the different types of systems engineers. Mar(1991) suggests that the common types of systems engineering are system architects, product systems engineers, program systems engineering, systems analysts, and those that approach any problem with the F-R-A process. Most organizations seem to seek the systems architect and when none can be found, seek concurrent engineering solutions. Rechtin (1991) implies that system architects do not have to be systems engineers, but many organizations insists that:

System architects must practice systems engineering

Systems architects are those rare individuals that can conceive new products and provide new architectural solutions to customer needs. The basic systems engineering concepts of generating functional descriptions prior to defining architect, or defining actual problems before seeking solutions is difficult for many high technology engineers and architects to accept. Innovators and architects may create for the sake of creating, not to respond to existing problems. If the system architects generate only architecture and ignore functional descriptions, reverse systems engineers will be needed to provide the F-R-A descriptions at the upper level.

Concurrent engineering takes the opposite approach to creativity and attempts to collate the inputs of many subdomain experts to synthesize domain expertise. The other extreme view of systems engineering that is often associated with the concurrent engineering process is :

Systems engineering orchestrate smore than create

This heuristic suggests that systems engineers role in concurrent engineering is to provide a discipline and structure that creates functional descriptions and requirements prior to seeking solutions using a top down sequence of descriptions. This role focuses on orchestration of the process and management of the information. In the perfect world, systems engineers should provide the creative spark as well as the glue for any program. In the imperfect world, two parties one supplying the spark (the architect) and the other supplying the glue (systems engineers) are needed.

Another view is that

Everyone must be a system engineer sometimes

where everyone contributes a little to the creative spark and everyone must cooperate and communicate using system engineering process for communication and structure of the activities. The original need for systems engineering can be traced to the inability of domain specialist to recognized all functions that the final product must perform. The solution was to create another group to speak for the system as a whole.

The definition of systems engineering found in most references stresses that translation of customer needs into functions, requirements and architecture., and to not omit needed functions. This implies:

Translation will be a greater problem than creativity

The failure to identify all functions early in the development process is the source of many cost overruns, poor product performance, and schedule delays. The inability to communicate real needs to those creating the products is recognized by everyone. Very few recognize the importance of using the basic F-R-A concepts to reduce misunderstanding. The increased complexity of new systems has spawned many new disciplines that use different nomenclature and paradigms compounding the communication problem.. The generation of the right solution to the wrong problem is more common than the inability to generate a solution for the right problem. Thus more important task for systems engineering is to improve communication and understanding and facilitate orchestration of the developmental effort. This should help to ensure that functional descriptions and requirement definitions are generated and any interested party can access this information in a usable format.

Role of any engineer

Members of any engineering discipline tends to view their discipline as the critical factor in any program development. The heuristics:

Your system is someone else's subsystem and someone else's system is your subsystem

is not well understood by many engineers. Their view of the world seems to center on their product, and does not include the rest of the system. Understanding basic systems concepts and that the total system is controlled by more than one part is necessary is necessary to achieve world class systems engineering. Otherwise a large fraction of the systems engineering effort is wasted fighting other team members for resources and control of the program. When your contribution to the program is critical, then you need to lead the effort. When your contribution is not binding the system performance, then you need not lead. The heuristics:

There is a time to lead and a time to follow

is must be accepted by those participating in the systems engineering process..

Every engineer should understand the need to implement F-R-A. If possible, all engineer involved in any activity should participate in problem definition or at least understanding their customer's needs. The heuristics:

Problem definition is a group activity, but problem solving may be done alone

is important for each engineer to understand. Domain experts can solve problems, but they must participate in the formulation of system problems. Too often, providing point designs and assuming a understanding of the problem leads to the right answer to the wrong problem. System engineering should focus on identifying the right problems to solve.

Implementation of systems engineering

Even when all engineers understand the basic systems engineering concepts, misunderstanding of its implementation can lead to program cost overruns, poor system performance or schedule delays. The most common implementation heuristic than is ignored is:

Pay me now or pay me later - you can not spend too much up front money.

Too often, in new program development resources for systems engineering are allocated by program managers that are rewarded for short term management of their resources during the proposal and start up phases of a program. Under such ground rules, minimize of the resources allocated to system engineering activities is logical, since there are few near term rewards for good systems engineering, the payoffs are downstream since careful F-R-A and effective trade studies result in fewer restarts and interface problems in the long run, but add cost to the start up activities. Given their incentives to save money in the short term, buying more system engineering early in a program seems to be a difficult concept for most managers to accept. The fundamental concept that system engineering is attempting to avoid surprises, rather than to cope with them when they are discovered is not well understood.

The introduction of systems engineering must begin with a common set of nomenclature that is accepted by all parties. Many attempts to implement systems engineering did not discover that participants had assumed different meanings for basic words such as functions and requirements. This misunderstanding of common terms can result in many person years of wasted efforts. As a result of this experience, It is important that:

A data dictionary is needed for every program

The practice of assigning the system engineering task to a separate group labelled systems engineering creates a polarization between system engineering and design. Too often the designers do not wait for the systems engineers to generate specifications and designs are completed before specifications (right answers to wrong problems). If the systems engineering group lacks domain expertise, the specifications include excessive, impossible or impractical requirements. One goal of concurrent engineering is to have all domain experts contribute to a functional description of the product prior to seeking the architectural description. This is consistent with proper implementation of systems engineering. Thus the heuristics:

Everyone in a given activity must use the system engineering process to make it work well

is derived from the observed tensions and conflicts that are encountered with systems engineering is separated from design, production and operations. Unless all participants in a program activity understand and embrace the basic system engineering concepts, the introduction of the discipline and structure of system engineering can become counterproductive. More time and effort is loss resisting system engineering than can be saved by proper system engineering.

Since it is impossible in most existing organization to implement a culture change that would cause an entire program or corporate entity to embrace systems engineering,

Incremental complete implementation on related tasks is more effective than partial implement on *unrelated tasks*

Better results are achieved when a entire group working on a part of the system uses system engineering, than to have many groups partially use systems engineering.

Incremental system engineering can lead to CDRL (contractor data requirements list) engineering, which only creates systems engineering products required by the contract. This process usually addresses each product independently of the others. Failure to recognize that system engineering products are chained together by the system engineering process, results in costly duplication of effort, and lack of integration of the system descriptions. The more successful incremental implementation of systems engineering selects particular lower level architecture for complete F-R-A description and analysis, rather than to focus on higher levels applications. In many cases, it is the upper level engineers that seek point designs.

Upper level managers seem too willing to allow TBD (to be determined) and orphan requirements to exist at upper levels, while encouraging lower level descriptions to continue. The risks associated with making lower level F-R-A decisions without resolving upper level TDB's and orphan is a major indicator that the basic concepts of systems engineering are not clearly understood. All lower level decisions on functionality and architecture are in response to upper level definitions. If the upper level decisions are missing, then costly mistakes will be made. In place of TBD's, the range of possible functions, requirements, or architecture must be established. These can be refined at a later date, but leaving a void is an invitation for disaster. Understanding by both the customer and contractor of the real risk of a allowing TBDs or orphans is missing in many programs.

Systems engineering is not limited to program or system level activities. Not all advocates of systems engineer appreciate this generalized use of systems engineering, but any task at any level can benefit from the use of systems engineering.

Systems engineering can improve the performance of any task at any level.

The generalized applicability of systems engineering provides any alternative implementation strategy:

You can tailor the system engineering process to fit any organization or any task

is important when considering the use of systems engineering.

Managing information generated by the system engineering process must be automated. Managing the documents containing these data and information has created a large and cumbersome bureaucracy which frustrates attempts to trace relationships between functions, requirements, and architecture. Recent advances in automation of system engineering information are shifting the emphasis to the control of information rather than documents.

System engineer your system engineering

The final implementation issue that has been neglected by many in practice is that system engineering activities must be system engineered. Many organizations have approached the costly issue of system engineering automation without defining the needed, requirements, and evaluate aleternatives using the system approach. The result have been poor performance and lost time. The failure to system engineer system engineering must be corrected.

SUMMARY

The heuristics discussed in this paper addressed (1) what is systems engineering?:

System architects must practice systems engineering
Systems engineering orchestrate more than create
Everyone must be a system engineer sometimes
Translation will be a greater problem than creativity

(2) what every engineering needs to know about systems engineering:

There is a time to lead and a time to follow
Your system is someone else's subsystem
Someone else's system may be your subsystem
Problem definition is a group activity, but problem solving may be done alone

and (3) how to implement systems engineering:

Pay me now or pay me later- you can not spend too much money upfront
A data dictionary must be created for each application
Everyone must use the system engineering process to make it work well
Incremental implementation on related tasks is more effective than partial implementation on unrelated tasks
System engineering can improve the results of any task
You can tailor the system engineering process to fit an organization's culture
Systems engieer your systems engineering

These concepts and issues may provide a point of departure for the papers to follow in this session.

REFERENCES

Alford, Mack, 1991, "Strengthening the System Engineering Process", Proceedings of the ASME/NCOSE Annual meetings, Chattanooga, TN.

DSMC, 1983, "System Engineering Management Guide", Defense Systems Management College, Fort Belvoir, Virginia

Mar, B. W., 1991, "What is Systems Engineering", NCOSE January meeting, Los Angeles

Rechtin, E, 1991 "Systems Architecting", Prentice Hall, Englewood Cliffs, NJ 07632

B.W. Mar is Professor of Civil and Systems Engineering at the University of Washington. He has been teaching systems engineering at the University for over 20 years. During the last decade he has been developing teaching materials and offering courses for government and industrial in-house systems engineering training programs. His research addresses systems engineering automation using COTS tools, and management of complex systems. He is president elect of NCOSE.