FRAT – A BASIC FRAMEWORK FOR SYSTEMS ENGINEERING

By

Brian W. Mar, Ph.D.
Emeritus Professor, University of Washington
10615 60th Avenue South
Seattle, WA 98178

Bernard G. Morais
SYNERGISTIC APPLICATIONS, Inc
333 Cobalt Way, Suite 107
Sunnyvale, CA 94085-5404

ABSTRACT

Systems engineering can be difficult to implement if the words and framework are not clearly understood by all parties involved. INCOSE has adopted a definition of systems engineering; yet a debate continues over what systems engineering is and how to do it. The authors have developed over a period of many years the FRAT framework for describing the process used to implement systems engineering for any task, the name for which is derived by the first letters of the series of words function, requirement, answer, and test. FRAT is based on the hypothesis that four views of a system are needed to completely define any system. These views are what it does (functions), how well the functions are performed (requirements), a physical or actual description of the system (answers), and verification and validation of the system when completed (tests). The FRAT framework has been validated by reviewing all the papers published at INCOSE proceedings. It is important to recognize that the system of interest (a product) is produced by another system (a production system). Both of these systems can be described by the four FRAT views when developing a shared vision of the development process as well as the product that is developed.

HISTORICAL PERSPECTIVE

The DMSC "Systems Engineering Management Guide" (SEMG) provided a summary and overview of the practice of systems engineering in 1983. Since then it has been revised and other publications appeared offering extensions and modifications of the systems engineering process and products. The basic terms introduced in the SEMG have created a confusion that exists to today.

The two most common misinterpretations of the SEMG are (1) functional analysis follows requirements analysis and (2) the term "system" is reserved for the product to be acquired.

A major source of confusion over functions, requirements and systems is the use of those words in the SEMG. Some readers understood that functional analysis must be performed prior to identifying requirements, while most read the SEMG verbatim and conclude that functional analysis follows requirements analysis. Most texts and recent publications continue to present this error. Many publications also reserve the term "system" for the product of interest and label the production system that creates the product "development program", "acquisition program", or other process type names.

What was intended was that mission level requirements (as well as functions, answers and tests) need to be decomposed into lower level functions, then lower level requirements, then answers and finally test. The condensation of this statement into requirements are developed prior to functions is the cause of the confusion that exists.

THE FRAT CONCEPT

The FRAT concept was developed over a decade of teaching systems engineering to university students who had problems understanding the examples and military jargon in the SEMG and to practicing engineers who were trying to understand how to implement systems engineering. Each year the authors have reviewed each paper published at the annual INCOSE symposium to determine new concepts in systems engineering and to validate that FRAT provides a framework for describing how systems engineering is implemented. These surveys are distributed via the INCOSE website or directly by the authors upon a request. The following steps simplified the complicated life cycle process descriptions and systems definitions presented in the SEMG:

1. Anything with parts that interact to achieve a common purpose whether it is a product (hardware, software, documents, instructions, etc.) a process, organization, or a thought, can be viewed as a system.
2. A system can be described by four views –

- what the system does (functions),
- how well the system performs its functions (all types of requirements including constraints),
- what the system actually is (answers), and
- verification and validation activities that provide the proof that the actual system

satisfies the intended functions and requirements (tests).

3. It is important to define and understand the three interacting systems;
   - the product system,
   - the program system that creates the product system, and
   - everything else that interacts with the product and program system.
4. To define a system at any level of decomposition, you need as an input a definition of the next higher level. If this upper level definition does not exist, the first step is to establish this in terms of the four views defined above. Once this is available, it can be decomposed into lower level functions. Once the functions are available the requirements for these functions can be established. Given the function/requirement descriptions the search for alternative answers can begin and trade studies used to select the better answer. Finally, definition and results of tests for verification and validation of the answer are generated.

Step 1 suggests that rather than reserving the word "system" for the product to be acquired, the word "system" can be used to describe anything. This allows anything to be described using the same four views (functions, requirements, answers and tests). Systems have boundaries as well as inputs and outputs. Any thing crossing those boundaries as an input goes through transformation to create an output. This is true even if the system is required to maintain a status quo. If state machine or transform thinking is used to analyze this transformation, the functions that are performed can be identified. These functions will have requirements that state how well they must be performed and will require some answer to perform them and also proof by some test that the answer is satisfactory.

Furthermore the process used to generate these views can be standardized, and repeated at any level of system description. One of the weaknesses of classic systems engineering is that a classification taxonomy for different types of systems was not developed. Many other disciplines applying systems theory classify different systems by the structure of their hierarchies, the relationship between parts or functions that exist at any level, or the complexity of the components or functions. For example, systems can be classified by their control mechanisms – those that have no feedback (deterministic, for example a bicycle), those that have simple feedback (a thermostat controlled heating system), those with anticipatory feedback (a thermostat with a clock control), and those that can learn or change goals. Once the type of control for a system is defined or identified, the general features of that system will be similar to all systems with that type of control. Object oriented descriptions of things utilize the system classification concepts. General characteristics of a particular group of things are identified as classes, and specific objects in this group inherit these characteristics. For example, all animals can be defined as objects that have four legs, warm blood, hair, etc. A dog is a specific type of animal that is a subset of the animal object. Objects are usually identified by answer views, and functions and requirements are described as characteristics. An effective exercise to understand system and object descriptions is to define a system as an object and an object as a system.

Step 2 suggests that the four views of any system are needed to completely describe a system. Any single view is necessary but alone is not sufficient. Tools developed to support capture of systems engineering information allow for many different types of function, requirement, answer and test views. They can be a hierarchy of requirement views that may exceed one hundred different classes increasing the risk that a requirement may be classified incorrectly or even classified as a function, answer or test. It is important to recognize the four basic FRAT views when using these complex models.

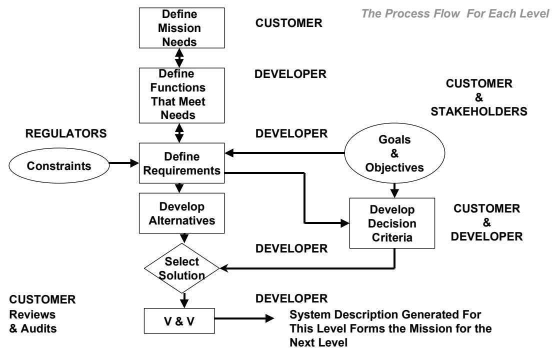
Step 3 identifies three interacting systems, the Product system, the Program system (that system which creates the Product), and the rest of the world (or universe). Each of these systems must have a clear definition of its boundaries. One of the Program system outputs is the Product system, and inputs from the rest of the world provide resources and constraints for both the Product and Program system. Each of these three systems has different functions, requirements, answers and tests. For example answers for the Program system tend to be tasks or activities needed to create and operate the Product system. The Work Breakdown Structure (WBS) descriptions are answers associated with a Program system. Each of these systems require the definition of all four FRAT views as well as graphical diagrams that define the parent/child hierarchy (vertical trace), the Requirements Allocation Sheet (RAS) type relationships (horizontal trace), and all the interface, sequence and processing dynamics (flow diagrams). Thus the FRAT views support the generation of all forms of systems engineering descriptions for both the product definition (specifications) and program management (Systems Engineering Management Plans -SEMP and WBS).

Step four is an important systems engineering concept of system decomposition. In classic systems engineering, there is a tendency to separate the generation of the four views of a system. Some people will generate functional descriptions, others will generate requirements, and others select answers and tests. Often designers developing answers did not wait for the systems engineers to complete the generation of specifications containing function and requirement descriptions. This leads to restarts when the functions and requirements were completed. Another problem, when the views are not integrated, is that tests were not defined until answers were selected. This has resulted in serious schedule slips if development of the test facilities required longer time periods than the product development. There is a relationship between the requirements and the verification and validation activities that must be understood. Requirements must be measurable and provide a means for proving that the selected answer can satisfy the functions and meet the objectives specified for that level of decomposition.

THE PROCESS ENGINE

As an approach to developing the four views of the three systems, the authors present the Process Engine. The process engine is a flow diagram of the sequence of activities needed to completely describe a system at any level of resolution. This description can be subsequently used to initiate decomposition of this description to a lower level (more detail). The process engine is initiated with a review of all of the information provided by the customer for the definition of the Mission Need. It requires developing the four views – Functions, Requirements, Answers and Tests – that help define the Developer's understanding of the Customer's need. (Rechtin, 1991) has advanced the heuristic "Don't assume that the original statement of the problem is necessarily the best, or even, the right one". This requires the communication of the developer's understanding of the need to the Customer and the Stakeholders. This also requires understanding that the Customer may dictate a solution, an answer (the A), at this top level. This also means completing the full FRAT definition of that need at the top level and identifying any lower level FRAT information that may be contained in the Customer's definition for use in subsequent analysis. This also requires that the Customer validate the Developer's understanding of the Mission Need. This is part of developing a Shared Vision between the Customer, the Developer and all the Stakeholders.

THE PROCESS ENGINE



The Developer can now develop the Program that will produce the Product. This will use the same process of developing the FRAT views for those things that have to be done to produce that product. For government missions there are standards and procedures that have to be met that will provide data and metrics to support the government's needs for data. For commercial entities, internal company processes and procedures may dictate these data requirements. The thrust of this activity is to define the tasks, resources, and schedules needed for developing the next level of Product definition. The standards or the internal processes provide the definitions of the functions that the Program system must perform and will give the requirements of how well they must be performed within the constraints dictated by resources and schedules that may be levied on delivering the Product. The tasks will be the answers for the Program systems.

The next step is developing the Functions at the next lower Product level that will satisfy the Customer's need as defined by the Mission Analysis that determined the top level F's, R's, and the A that may have been customer directed. It will become evident that the FRAT descriptions of a Program are task descriptions, while the FRAT descriptions of a product are descriptions of hardware, software, or other types of products. This step should identify all of the inputs and outputs for each of the functions at the level of the Mission Need and identify the children of those functions for each of the system level inputs and outputs and their required transformations. The Developer should also create an Operational Concept Document that describes how the System will be operated through all of the modes. These modes include, initiating, operating, maintaining and, at the end of life, retiring the system. This should be validated against the vision that the Customers and users of the system have for operations during these various modes. The operational scenarios, that are a product of this effort, will also reveal other functions that may not be evident in the customer's initial statement of the mission need. This also includes developing diagrams that show the interactions between functions. If the generation of a system description does not proceed by first identifying the functions, and then, given the function, identifying the performance requirements, the description will be incomplete. The function/requirement description pair forms a problem statement or specification of the desired product. A key difference between a systems engineering approach and a conventional design approach is that the specifications provide~~s~~ a comprehensive problem statement capturing the needs of customers and stakeholders as well as the constraints of nature (scientific laws). The systems engineering process requires that alternative solutions be identified and traded before selecting a solution, while the classic designer usually provides a point solution dictated by personal experience rather than a comprehensive trade study. Often the point solution will not satisfy all the functions and requirements in the specification, since the designer was unaware of all the needs of the customers and stakeholders. The decision criteria that are needed to support the trade studies are determined from the understanding of the requirements that have to be met, the Constraints (requirements that are owned by someone else), and the Goals and Objectives desired by the customers and stakeholders. These goals and objectives typically take the form of preferences such as "use this technology if possible", " we would like to have the system development completed within the next two years", or "we would like to minimize the impact that this system will have on the local environment". These goals and objectives provide weighting for the decision criteria, wherein the Constraints and requirements have to be satisfied.

The use of the process engine for the development of system description has a convention that requires that the development of the FRAT data in the sequence of functions before requirements, requirements before answers, and answers before test. Thus a system should not be decomposed until all FRAT views for that level are established and base-lined. Only then should the next level of FRAT description begin. If this sequence if not followed, functions could be decomposed independently of requirements, answers or test. The answer that was selected to provide the functions (the actions) and the requirements (how well those functions will be performed) determine the next lower level functions at the level being decomposed. This allows the tracing of the parent/child relationships of the Functions, Requirements, Answers and Tests. This also allows for the creation of trees showing FRAT relationships for any kind of system. The development of the four views at each level allow for the depiction of the horizontal relationship of what the systems must do (the functions), how well it will do it (the requirements), what will do it (the answer) and how it will be proven that answer can satisfy the functionality and requirements (the tests). A requirement allocation sheet (RAS) was classically used in the SEMG concept to package FRAT information for each functional description. Requirements management tools allow more complex descriptions revealing the relationship between the FRAT views at each level of system description as well as traceability between levels of system descriptions.

The sequence of developing systems is to first define the product that will satisfy the Mission Need after determining that the need description is complete. That is determined by defining the top level Mission Need FRAT. Then the Program is defined to get the Product. (This creates a chicken and egg situation since resources are needed to define the Product, and some type of program description is needed to define this. Since the customer traditionally performed this task and allocated resources to develop the mission need statement, the developer usually used the customer defined mission as the starting point for his or her product definition effort. The program to do this was then defined in the proposal to win the product definition work.) The Program definition requires the determination of how that top level Product FRAT will be decomposed to get the next level of the definition. When that is complete, the Program must now be decomposed to generate the next lower level of the Product FRAT. When that level of Product definition is complete, the next level of decomposition for the Program can be started. This process of defining the Product in more detail then defining the Program to get the next lower level of Product detail continues the decomposition process down to point that the developer wants to manage. At that point, the Specification, SEMP and WBS hierarchies are set for adding details for both the Product and the Program. This will set the stage for the orderly development of a system that will meet the Customer's needs, satisfy the Stakeholder community and be within the constraints of schedule and costs for the development. All of this by defining first the FRAT for the Product, then the FRAT for the Program that will define the next lower level of FRAT for the Product. The relationship between the Product and Program are shown in the following figure using the process engine.

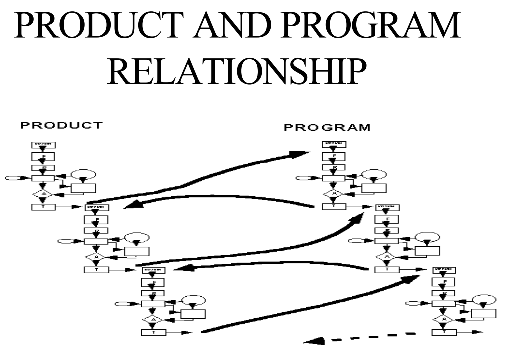


Figure 2. The Relationship Between Product and The Program to produce that product

Another representation of this relationship between the Product and the Program and the development of the Product Specifications and Program Documentation, the SEMP, is shown in the following figure. The block labeled "Analyze Problem" for the Product and "Analyze Organization" for the Program reflects the activities to determine the function/requirement pairs for each. The blocks with the labels "Synthesize" are to determine the answers that will satisfy these function/requirement pairs. The blocks with the labels "Verify" are the tests that prove that the selected answers will meet the objectives that are recorded in either the Specification or the SEMP. These represent the documentation developed for the Product as well as the Program that will provide that Product.

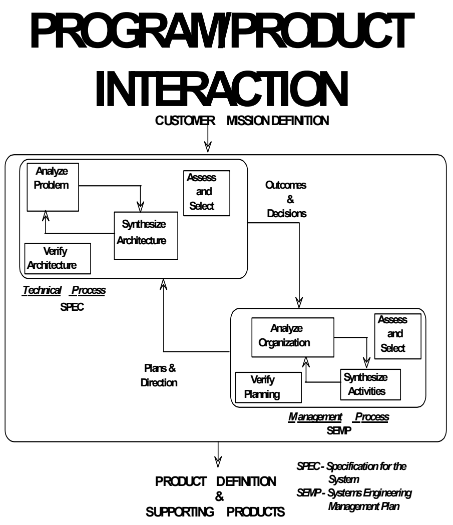


Figure 3. The Product Program Interaction

The following figure shows the flow of the decomposition of FRAT data adding more detail as the hierarchy is developed for both the Product and the Program descriptions. The apex of the pyramid is the mission at the top level. For the Product, this is the Customer's definition for the need. For the Program, what is needed to develop the next level of the Product definition. The pyramid has four faces, each representing the F, R, A and T that is required to finish the definition of each level. There are parallel pyramids, one for the Product and one for the Program that develops the next level of definition for the Product. These are each defined at each level before proceeding to subsequent decomposition for each level of the Product and Program.

SORTING AND USING FRAT DATA

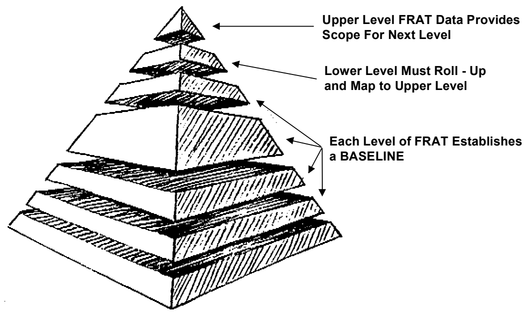

Upper Level FRAT Data Provides Scope For Next Level
Lower Level Must Roll - Up and Map to Upper Level
Each Level of FRAT Establishes a BASELINE

Figure 4. FRAT Hierarchy

The definition needs to be completed before decomposing to the next level. These should be base-lined with the Customers and Stakeholders before the next level of analysis is started. There should also be a review that includes a roll-up that makes sure that the top-level objectives have not been modified in the subsequent analysis. The result is one where the analysis continues down to that level where the specifications and management plans have been documented and validated and are sufficient to provide the development/design organization with the definition of the product to be delivered and how it will be procured. The FRAT framework can also be used to describe all other lifecycle tasks such as design, production, V&V, delivery, operations, maintenance, modification, replacement and decommission. While not all these tasks recognize the need for systems engineering, it is implicitly used.

FRAT Versus Systems Engineering Standards

FRAT is not a substitute for systems engineering standards or other definition of systems engineering. It is a framework to discuss systems engineering products and efforts. For example the EIS/IS 731 definition of systems engineering does not satisfy the FRAT framework for describing systems engineering related issues. The key areas and processes defined by EIA/IS 731 shown in the following are actions that the systems engineering process should perform.

EIA/IS 731 Key Areas and Processes

ENGINEERING, TECHNICAL, AND SYSTEMS ENGINEERING AREAS

Define Stakeholders and System Level Requirements
Define Technical Requirements
Assess and Select
Integrate System
Verify System
Validate System

PROJECT OR MANAGEMENT AREAS

Plan and Organize
Monitor and Control
Integrate Disciplines
Coordinate With Suppliers
Manage Risk
Manage Data
Manage Configuration
Ensure Quality

In terms of the FRAT framework these only provide the _function_ view of the system defined as systems engineering standards. Without the _requirements, answer, and test_ views the definition of the SE standards system is incomplete. For example the function "Define stakeholders and system level requirements" needs a requirement statement that defines the quality, quantity, timeliness, and other measures of effectiveness that can be used to determine if the definition is acceptable. The answer view is the actual set of stakeholders and system level requirements developed; the test view defines how to determine if the answer satisfies the established function and requirements. In many attempts to define systems, the author assumes the missing views are common knowledge and need not be documented, or that someone else will provide such information. The FRAT framework provides a check-list and a constant reminder that any systems description must include all four FRAT views, and the FRAT engine provides a sequence of activities to generate these views. When dealing with existing systems, the FRAT views are developed in a reverse systems engineering sequence since the answer view exists. In the case of reverse systems engineering, the answer must be used to define the functions and requirements provided by the answers. The test views document the test activities and results that were used to verify and validate that the answers will perform the defined functions and meet the requirements.

In summary, the FRAT framework is the least common denominator to define all the information generated by systems engineering activities, and provides a process that can be used to develop such data.

REFERENCES

Defense Systems Management College, *Systems Engineering Management Guide*, Fort Belvoir, VA, 1983,

Rechtin, E., *Systems Architecting - Creating and Building Complex Sys*tems, Prentice Hall, Englewood Cliffs, NJ, 1991

EIA/IS 731, *Systems Engineering Capability Model*, Electronics Industries Affiliates, 1999

BIOGRAPHIES

Brian W. Mar, Ph.D.
Dr. Mar is an Emeritus Professor of the University of Washington. Prior to his retirement, he was at the University of Washington for over 30 years and was a Professor of Civil and Systems Engineering. The Boeing Company employed him for 10 years prior to his joining the University of Washington. He holds a Ph.D. in Chemical/Nuclear Engineering as well as several degrees in Civil and Chemical Engineering and has served on International and National councils and advisory boards. He is a member of Phi Beta Kappa and Tau Beta Pi Honor Societies. He is one of the founders, a Past President and a Fellow of INCOSE. He has published several books and over 100 papers.

Bernard G. Morais
Mr. Morais is the President of Synergistic Applications, Inc. and has over thirty five years of Program management and Systems Engineering experience in industry. He holds a BSEE from California State Polytechnic University and M.S. in Systems Management from the University of Southern California. One of the more significant achievements in his career was leading the development of the first *Systems Engineering Management Guide* for the Defense Systems Management College. He has been an Adjunct Professor at San Jose State University where he stressed the multi-disciplinary aspects of designing systems. He has provided lectures in Systems Engineering at many other universities. He provides consulting and training support for National and International government agencies as well as communications and energy companies. He is a founding member of INCOSE, was the first Treasurer and is the holder of the 2001 Founders Award.