Chapter 2
Fundamentals of Systems Engineering—A Practitioner's Approach

John C. Hsu

Abstract In today's globalized environment, companies compete fiercely for business. They need world class product quality, no cost overrun and schedule slippage. Customer satisfaction is number 1 and cannot afford system development failures. Practicing systems engineering is the answer. It is an old subject but has been revitalized since the mid 90s. Systems engineering is now a major theme in this century has led to reduction in time-to-market, improving quality and reducing costs. However, systems engineering has not been sufficiently understood by the majority of workers (technical and nontechnical, professional and non-professional, and financial, etc.), evidenced by many failures been reported, specifically in US Government Accountability Office (GAO) reports. Therefore, continuous systems engineering education is still needed, which is the major theme of this chapter. In this chapter a history of systems engineering is introduced; including why it is needed, its evolution and revitalization. the fundamentals are presented. The requirement management process needs to be followed to analyze, derive, allocate and trace the requirements. Functional analysis can assist requirement hierarchy developed, vice versa; requirements can also feed function architecture development. This chapter is concluded by overview of the functional allocation and the system synthesis.

Keywords Systems engineering · Revitalization · Requirements management · Functional analysis and allocation · Trade studies · Interface management

J. C. Hsu (✉)
AIAA, Reston, USA
e-mail: john.hsu@csulb.edu

INCOSE ESEP, San Diego, USA

The Boeing Company, Long Beach, USA

California State University Long Beach, Long Beach, USA

System Management and Engineering Consulting Services, Cypress, USA

UK Royal Academy of Engineering, London, UK

Queens University, Belfast, UK

J. Stjepandić et al. (eds.), *Systems Engineering in Research and Industrial Practice*,
https://doi.org/10.1007/978-3-030-33312-6_2

2.1 Introduction

In today's globalized environment, manufacturing and designing companies compete for business. To be successful, companies need to practice strategies that minimize the possibility of degradation of product quality, cost overrun, schedule slippage, customer dissatisfaction and system development failures. Systems engineering is the answer to the above statement. It is different from mechanical, aeronautical, electrical and other engineering disciplines and yet bridges these traditional engineering disciplines. Systems engineering is focused on the system as a whole. While the primary purpose of systems engineering is to guide, it does not mean that systems engineers do not themselves play a key role in system design. Systems engineering is needed now and in the future to meet the following challenges:

- Constantly changing requirements
  - Requirements for new system are frequently changing
  - Changes in mission thrusts
  - Continuous introduction of new technologies
- More emphasis on "systems"
  - Greater emphasis on total system versus the components of a system
  - Functions need to be performed in an effective and efficient manner
  - Look at the system throughout its entire life cycle
- Increasing system complexities
  - System becomes more complex, such as, system-of-systems for network-centric applications
  - System design changes should be incorporated quickly, efficiently, and without causing a significant impact of the overall configuration of the system-of-systems, system, or subsystem
- Increasing globalization
  - More trading and dependency on different countries
  - Introduction of rapid and improvement communications
- Greater international competition
- Increasing globalization can also trigger more international competition
- More outsourcing
  - More suppliers associated with any given program
  - Needs early definition and allocation of system level requirements

Systems engineering fundamentals in this chapter will cover the following subjects:

1. Introduction to Systems Engineering.
2. Requirements Management.
3. Functional Analysis and Architecture.

Section 2.2 Introduction to Systems Engineering will explain why we need systems engineering; evolution of systems engineering; the history of systems engineering revitalization; and the role of systems engineers. The systems engineering methods and processes presented in this chapter can be directly applied to the job. Learn how to analyze and develop requirements in Sect. 2.3; and how to validate, trace and allocate requirements. Four (4) elements of functional analysis and Allocation are discussed in Sect. 2.3. In Sect. 2.3 the Functional Flow Block Diagrams and Integrated Definition for Functional Modeling are presented, as well as the relationships between functional allocation and system synthesis, and decision Analysis as needed during the design, development and manufacturing life cycle.

2.2 Introduction to Systems Engineering

In this section, we discuss in detail the poor performance of engineering projects in current industries, especially with respect to cost overruns and schedule delays. Practicing systems engineering (SE) may be the answer to correct these problems. The value of SE is presented. SE is not a new subject; therefore, the evolution of SE is discussed in this section. Many people may not know what SE revitalization is. It will be presented and discussed here. Finally, the role of systems engineers is explained.

2.2.1 Why Systems Engineering

This is often a question in people's mind. Why do I need to know and understand systems engineering? What is good for me? One obvious reason that we can give is [1]:

> Systems Engineering provides theory and methods for the management of complexity. Without Systems Engineering, we can expect additional developmental failures, cost overruns, schedule slippages, customer dissatisfaction, and environmental disasters.

Figure 2.1 is extracted from Metrics and Case Studies for Evaluating Engineering Designs, Moody et al. [2]. It shows that the cost overrun decreases as the systems engineering (SE) effort as a percentage of total program cost increases for most of the space programs in the past. Figures 2.2 and 2.3 [3] show that the cost ratio of actual to plan and schedule ratio of actual to plan, respectively, decreases as SE effort increases. The SE effort is defined as the product of SE quality and the ratio of SE cost to actual cost [4].

United States (US) Government Accountability Office (GAO) [5] have found problems related to quality that have resulted in major impacts to the 11 Department of Defense (DoD) weapons systems, billions of dollars in cost overrun and years-long delays, and decreased capabilities for the warfighter. GAO's analysis of 11 DoD

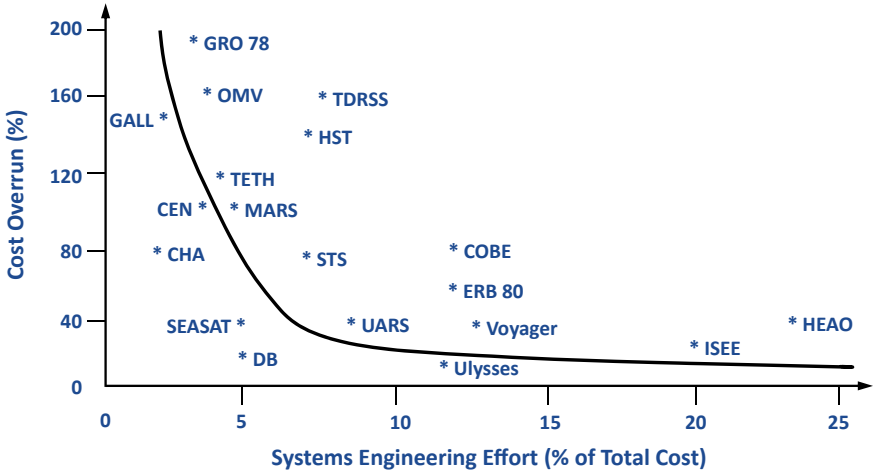

Fig. 2.1 Cost overrun versus systems engineering effort [2]

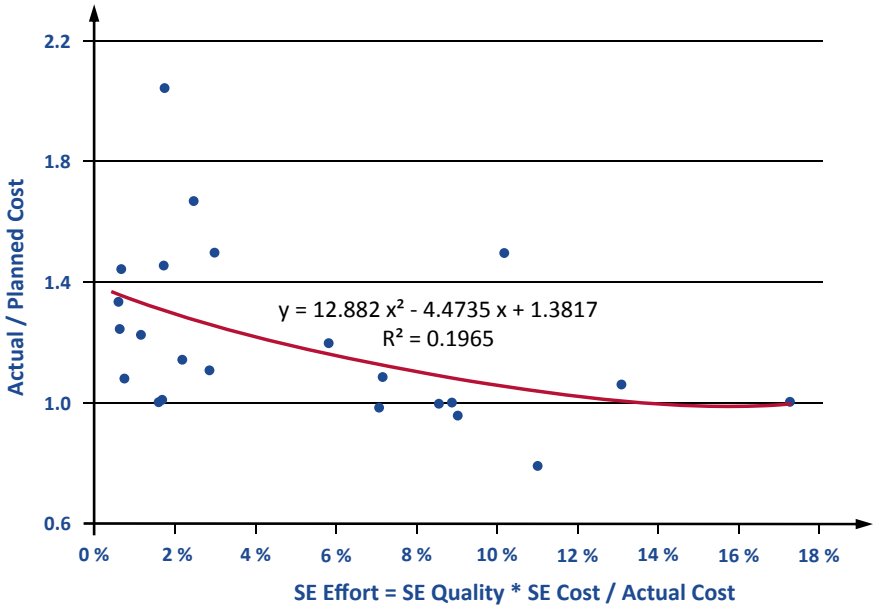

Fig. 2.2 Cost ratio of actual to plan [3]

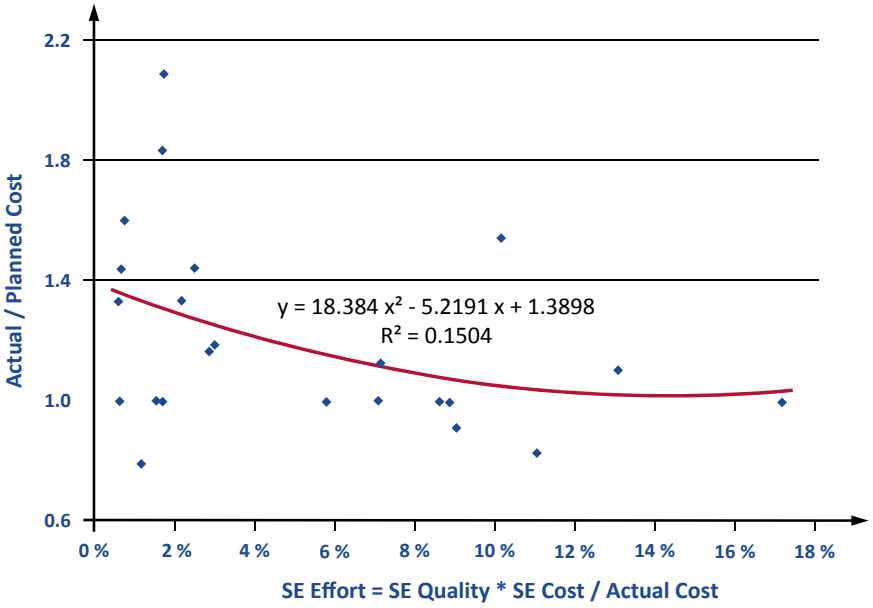

Fig. 2.3 Schedule ratio of actual to plan [3]

weapon systems illustrates those defense contractors' poor practices of systems engineering activities as well as manufacturing and supplier quality problems. A study [6] was undertaken why cost overrun and schedule delays have occurred and continue to occur in large-scale US federal defense and intelligence acquisition programs. One of the major reasons is inadequate systems engineering practices. Requirements redefinition and creep was discussed as a major problem. The acquisition strategy for programs must embrace the systems engineering processes and philosophies early in the program life cycle [7]. Systems engineering practices provide a program baseline where customer and stakeholder needs are satisfied, when diligently followed early and throughout the acquisition process.

It is emphasized by a customer, "…Imperative for all the programs is to focus more attention on the application of Systems Engineering principles and practices throughout the system life cycle" [8]. It is further directed, "Improve SE throughout the acquisition process, including workforce …education and training; tools … guidance; Provision for contractor's Board to consider contract performance when setting top executives' salaries/bonuses." One year later, "Application of rigorous systems engineering discipline is paramount to the department's ability to meet the challenge of developing and maintaining needed warfighting capability" [9].

The industries, especially defense and aerospace, are crying for the application of systems engineering practice in their companies, as described in the presentations of "Systems Engineering in Today's Competitive Environment" [10], and "Current and Future Trends in Systems Engineering" [11]. The four (4) key metrics for a company

to achieve business targets are: Revenue Growth, Net Margins, Operating Cash Flow, and Return on Net Assets. The actions required to meet these four (4) metrics are all related to systems engineering practices. They are:

- Meet customer requirements
- Validate and verify requirements to final products
- Mitigate risks
- Make the best decision
- Continuously measure performance
- Shorten Cycle Times
- Execute to plan and schedule
- Best organization and budget control

With world-wide web communications nowadays, the newest technology will be most likely spread within two year or less; therefore, technology is a common commodity in most cases. The winner is not determined by the possession of the technology. It is the one who can first deliver the product to the market with the best quality and sell at the cheapest price. Then what are the challenges? They are cost and schedule.

There are three (3) basic arguments for the value of systems engineering: First, assurance that the system will accomplish its objectives; Secondly, the cost-time trade-off as shown in Fig. 2.4 [12] that the later problems are discovered, the more it costs to fix them; and thirdly, insurance against serious low-probability consequences. Well-practiced SE is for adequate upfront planning, adequate scope definition, and understanding customers intent, expectations and requirements definition at the early phase of a project. A well-executed team work can achieve the value of systems engineering.

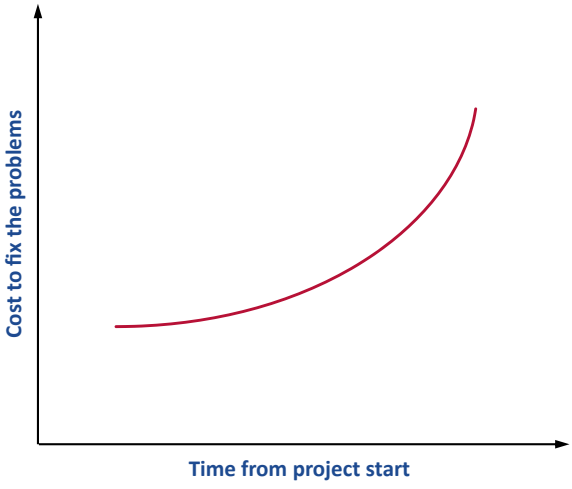

Fig. 2.4 Cost-time trade effect [12]

2.2.2 Evolution of Systems Engineering

SE is an old subject that started in ancient time; like Egyptian built pyramids, Chinese built the great wall, and Roman built the forum and bridge. During that time, it was not called systems engineering but they must used the system principle and method to build huge and durable structures, not to mention the tower of Pisa as a bad system example.

In 1957, the Soviet Union launched the human-made satellite orbiting around the earth and also had the intercontinental ballistic missile (ICBM), a guided ballistic missile with a minimum range of 5500 km (3400 mile) primarily designed for nuclear weapons delivery (delivering one or more thermonuclear warheads). The US suddenly fell behind. For national survival and space race, US government spent billions of dollars to build the satellite and ICBM. These are complicated systems that require a systematic approach with precision and sound management; therefore, SE was adopted and expanded rapidly. System performance for mission success was emphasized, as well as project management for technical performance, delivery schedule, and cost control. A driving force for high system reliability led to the development of parts traceability, materials and process control, change control, product accountability, formal interface control, and requirements traceability. These are all SE tasks and processes.

2.2.3 Systems Engineering Revitalization

After successful completion of the Apollo Program, US space race and ICBM program won over Soviet Union. The government significantly cut back funding for space and defense programs. The contractors suffered severely shortages of funding and forced major reductions in manpower. This was in the early 1970s. In the next ten (10) years, government slowly increased funding for space and defense programs. The contractors also gradually recovered and increased personnel hiring. In the late 1980s, more space and defense programs suffered cost overruns and schedule delays that raised government concerns. The newly hired employees were not aware of systems engineering principles, methods and processes. SE has faded away. In the mid 1990s, US Air Force started an initiative to revitalize systems engineering practices that were executed successfully in the 1950–1960 period to win the space race and ICBM competition. A year later DoD led the systems engineering revitalization until today. Michael W. Wynne, acting under the secretary of defense for acquisition, technology and logistics, and Mark D. Schaeffer, principal deputy, defense systems and director, systems engineering, Office of the USD (AT&L), called for the revitalization of systems engineering across the Department of Defense [13]. "Analyses of a sampling of major acquisition programs show a definite linkage between escalating costs and the ineffective application of systems engineering," Wynne and Schaeffer called for the "systemic, effective use of systems engineering as a key acquisition

management planning and oversight tool" and said that, in addition, DoD would "promote systems engineering training and best practices among our acquisition professionals."

In its present form, the systems engineering process is broadened and combines elements of many disciplines:

- Operations research and analysis
- System modeling and simulation
- Decision analysis
- Project management and control
- Software engineering
- Specialty engineering
- Industrial engineering.

2.2.4 Role of Systems Engineers

Systems engineering (SE) is not a rocket science but it may be harder than rocket science. It deals with people, management and engineering. SE is with a project or a program from womb to tomb, and from cradle to grave. SE principles and methods can also be applied to individual's daily life.

Systems engineering differs from traditional disciplines in the following ways: It is focused on the system as a whole; it is concerned with customer needs and operational environment; systems engineering leads system conceptual design; and bridges traditional engineering disciplines and gaps between specialties.

The systems engineers can be:

- Deputy to Project Manager
- Customer Interface
- Requirements Owner
- System Architect
- System Analyst
- IMP/IMS Generator and Keeper
- Risk Management Administrator
- Trade Study Facilitator
- Interface Manager
- Verification Plan Owner and Administrator
- Process Owner
- Coordinator.

A Successful Systems Engineer should be a good problem solver, and welcome challenges, well-grounded technically with broad interests, analytical and systematic, but also creative, and a superior communicator with leadership skills.

Recommended systems engineers' role in a program is shown in Fig. 2.5 [14].

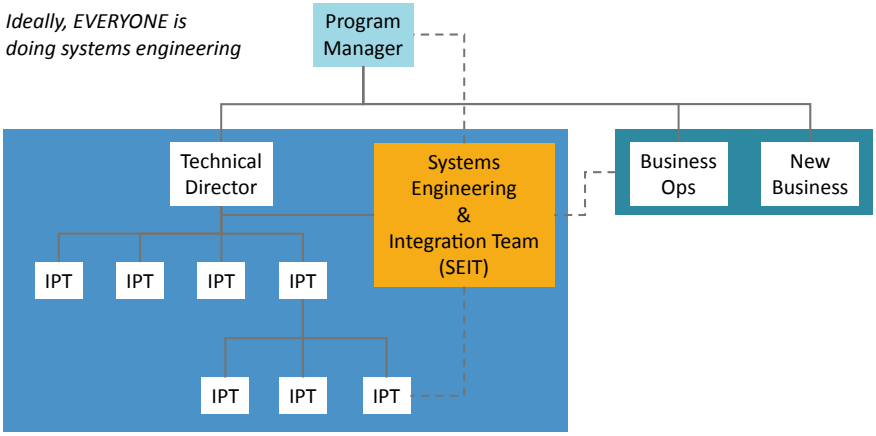

Fig. 2.5 Systems engineers' roles on the program [12]

2.3 Requirements Management

A general systems engineering process is shown in Fig. 2.6 [14]. On the left of the figure are customer requirements. As you can see that the customer requirements come in different forms, from highly sophisticated customers, like US government with fully developed user system specifications indicating performance, supportability, measures of effectiveness; and the constraints as affordability, interoperability, system evolution, and component reuse, etc., to casual customer requirements walking on the street with a few words. Regardless what forms customer requirements

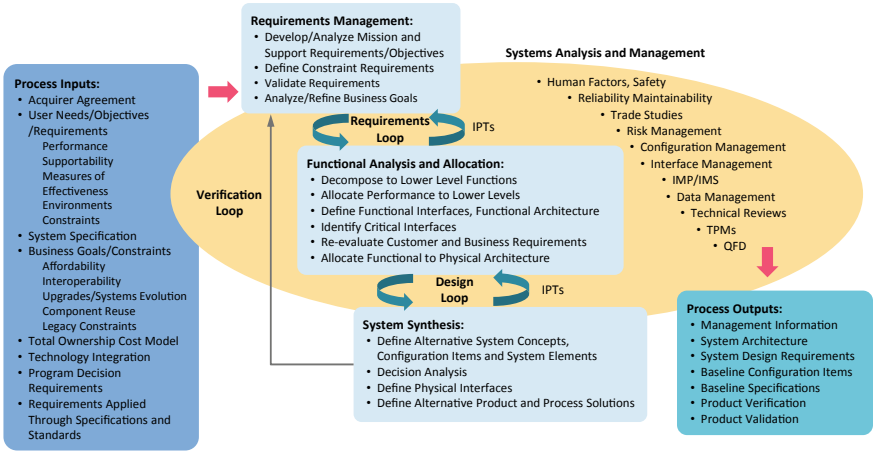

Fig. 2.6 The systems engineering process [14]

have, they have to be validated before deriving system level requirements. It leads to the middle top block "Requirements Management" which is the subject of this section.

The definition of Requirements Management is

> The identification, derivation, allocation, and control in a consistent, traceable, correlatable, verifiable manner of all the system functions, attributes, interfaces, and verification methods that a system must meet including customer, derived (internal), and specialty engineering needs.

The objectives of Requirements Management are:

- Ensure specified requirements have been completely decomposed and met in the design
- Ensure that any impacts to the design due to Requirements modification are completely understood
- Ensure no extraneous requirements (or components) have been introduced
- Ensure requirements have been completely verified.

Refer to the Requirements Management Process as shown in Fig. 2.7 [14], in the first-round of the requirement loop should work with customers to derive the top-level system requirements. For commercial products, derive the top-level system requirements based on validated customer requirements. Each system requirement needs to be allocated to function, organization or individual. This is called requirements allocation to ensure the ownership for each requirement. Under Implementation Process, there is no system design and supplier hardware/software development for the first-round. Each system requirement needs to have one or more verification requirements. For the second-round the systems engineers should also work with customers to derive the second-level system requirements. In sum, each requirement needs to be allocated and has one or more verification requirements. The second-round implies

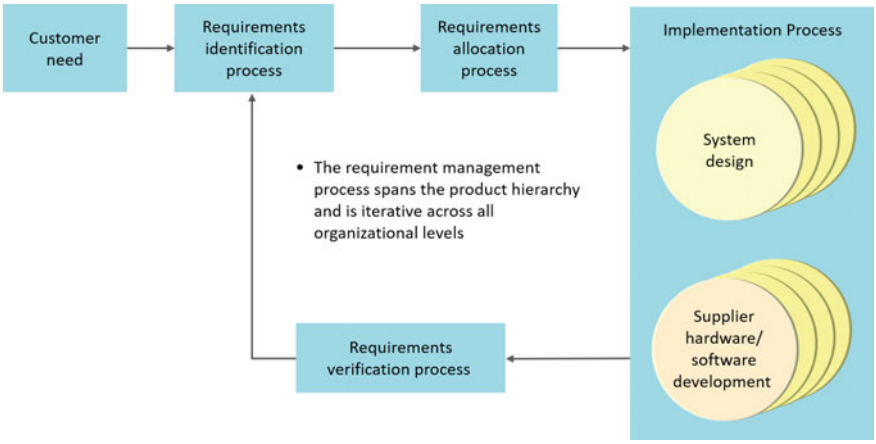

Fig. 2.7 Requirements management process [14]

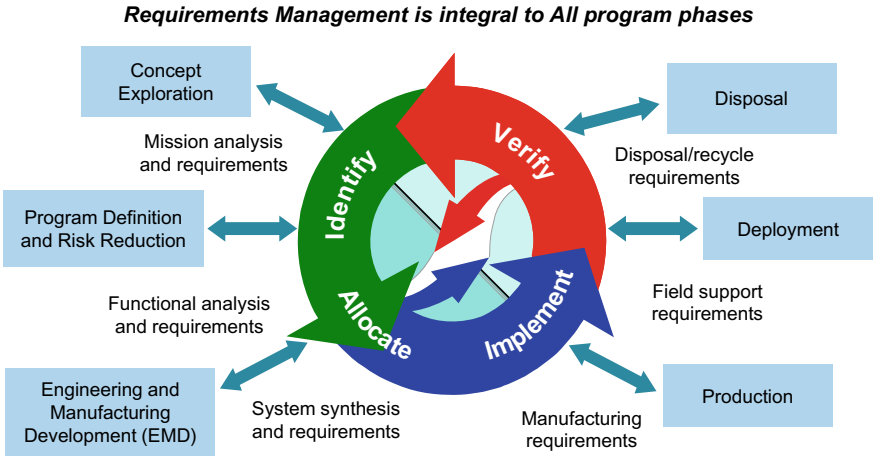

Fig. 2.8 Requirements management role in a system life cycle

conceptual design but not yet the hardware/software development under Implementation Process. With each more round, more lower-level requirements are derived for lower-level designs and hardware/software developments. Then the requirements hierarchy is formed and the requirement management process spans the product hierarchy and is iterative across all organizational levels. Requirements are central to all facets of the product. Requirements can be found everywhere and will depend upon each other; therefore, the continuous involvement and coordination with all interested parties must be accomplished. All impacted areas must be tied together to ensure an optimal solution.

The Requirements Management role in a system life cycle is illustrated in Fig. 2.8. The outside circle is a system life cycle from Concept Exploration to Engineering and Manufacturing Development (EMD) to Disposal. The middle cycle is the requirements for different phases in a system life cycle. The inner circle is the steps for each requirement from identification to allocation for the requirement, implementing the requirement by designers and verification requirement for insuring the product satisfying the requirement. At the beginning of developing requirements for Concept Exploration, the project team is formed on day 1, a concurrent engineering practice. All the required team members are gathered including representatives from each life cycle phase. The requirements will be developed for all phases of the life cycle at the beginning. Use X to represent the consideration and inclusion during the design of a product, such as, reliability, maintainability, survivability, safety, test, fabrication, assembly, operation and product support, etc. The goal of applying SE is to fulfil "Design for X" to cover the development and usage of a product in its entire life cycle.

2.3.1 Validate Requirements

We need to fully understand the customer requirements with no ambiguity. If we do not understand even with slightly doubt, we cannot derive and develop requirements. A better way to convey the concept of validating requirements is to use a cup of coffee as an example as shown in Table 2.1. A cup of coffee is a simple and small system. It does not need system requirements and derive design requirements are derived directly from customer requirements. Commercial customers have very loose requirements. If customers are asked what kind of coffee they would like to drink, they will say "taste good". You would scratch your head what does "taste good" means? You need to validate the requirement of "taste good". One way is to make ten (10) cups of coffee and you know exactly the flavour percentage in each cup of coffee. You need to get 90% of general population to taste your ten (10) cups of coffee. The number and kinds of people selected to satisfy the 90% is similar to polling process. If majority of the population like the taste of No. 5 cup of coffee. Then you have validated the "taste good" requirement that is the flavour percentage of No. 5 cup. The next customer requirement is "be hot". You can make five (5) cups of coffee with 70, 75, 80, 85, and 90 °C. Again, gather 90 percentile of general population to taste these five (5) cups of coffee. If the majority population think No. 4 cup is "be hot". Then you know 85 °C is the validate requirement for "be hot". Customers would like the coffee "wake me up" especially in the morning. You know "wake me up" relates to "quantity of caffeine"; however, the "quantity of caffeine" competes with "level of acidity", which corresponds to customers' "not upset my stomach". You make ten (10) cups of coffee. Each has different levels of caffeine and acidity. You know that the higher level of caffeine, the higher level of acidity. With 90 percentile of population to taste these ten (10) cups of coffee, you find out that the No. 2 cup of coffee the majority of customers feel enough "quantity of caffeine" to "wake me up" and yet "not upset my stomach". Then you have validated these two customer requirements with exact "quantity of caffeine" and "level of acidity". The last customer requirement "be cheap" competes with all the above customer requirements since better taste, higher temperature, and more quantity of caffeine will increase the base cost of a cup of coffee. You can validate this requirement by comparing the price of your cup of coffee with the price of other equivalent brand(s). When all these customer requirements are validated, you need to verify that each cup of your coffee will be made to No. 5 cup of flavour percentage, No. 4 cup 85 °C,

Table 2.1 Customer requirements driving the design requirements

Customer requirements	Design requirements
Taste good	Flavor components
Be hot	Serving temperature
Wake me up	Quantity of caffeine
Not upset my stomach	Level of acidity
Be cheap	Cost per cup

No. 2 cup of coffee "quantity of caffeine" and "level of acidity", and the price of "be cheap". You may have an idea now how to validate customer requirements and verify design requirements.

2.3.2 Requirements Analysis

The requirement and different kinds of requirements are defined below:

- Requirement—A statement of required performance or design constraint to which a product must conform.
- Customer Requirements—Statements of fact and assumptions that define the expectations of the system by customer.
- System Requirements—The necessary task, action or activity that must be accomplished. System Requirements identified in requirements analysis will be used as the top-level requirements.
- Design Requirements—The "build to", "code to", and "buy to" requirements for products and "how to execute" requirements for processes expressed in technical data packages and manuals.
- Derived Requirements—Requirements that are implied or transformed from higher-level requirement.

A requirement must be related to the function on which is to be performed, verifiable, which means testable, analyzable, demonstrable and inspectable (not applicable to customer requirements), precisely worded, and be unique.

First, we need to analyze customer requirements (CRs). After validation of customer requirements, we need to define customer functional and performance requirements. Also need to identify, understand, and define customer constraints. The specialty engineering requirements, such as, reliability, maintainability, availability, safety, survivability, and human factor. of the system are important to know at the onset of requirements analysis.

The next step is to analyze and derive the top-level system requirements. The sophisticated customers, like DoD (Department of Defense), NASA (National Aeronautics and Space Administration), and other government agencies, will provide systems specifications to contractors. If this is the case, we need to break down paragraphs in the system specification down to each sentence as a requirement. If the requirements are mandatory, remember to use the word of "shall" in front of the verb; otherwise, the designer or owner of this requirement does not have to follow the requirement. If we received compound and complex Statements from customers, we need to decompose these statements into a set of single requirements.

The design requirement drives the design as a one-way street. The design should be developed from these requirements. If a requirement is derived from the existing design, it is called reverse engineering that defeats the purpose of deriving requirements from CRs. Most likely, the design will not meet the CRs. Each requirement

shall be a complete, simple, specific, and straightforward sentence with concise quantifiable meaning. Without quantity, how can the requirement be used for design or verification? It is often to read "TBD" (to be determined) included in the requirement writing. This is not allowed since quantity value have to be indicated clearly in the requirement statement. For compliance with military and commercial standards, specifications, or any documents, the section number of the document shall be cited in the requirement. The requirement sentence cannot contain any adjectives which will be interpreted differently by different people, for example, beautiful, comfortable, etc. Margin in a requirement will provide design flexibility and tolerance for ease in manufacturing. As a result, margin in requirements can reduce cost for design and manufacturing. How can you determine the margin in a requirement? The answer is to use trade-off studies if necessary to determine the optimum margin.

The requirement should be written in positive tense, not passively. Requirements engineers who write requirements can be categorized as "words engineer" since the words used in the requirement dominate the design and verification. Requirement documents are legal documents as part of the contract.

2.3.3 Requirements Allocation

Requirements should be allocated. Allocation means to set apart, assign, or allot for a particular purpose. Requirements allocation is the assignment of requirements to a responsible party. There are several ways of allocation. Partition a value assigned to a parent requirement into parts that are assigned to child requirements. Requirements can also be allocated to functions, organizations, or experts. Sometimes, a requirement is for a specific time period, for example, most of the people witnessed the countdown of rocket launch the last 10 s. In reality, the countdown starts at years or months ago. There are requirements associated with different year or time period.

Requirements allocation can be tabulated in a table, with requirement numbers, requirement sentences, associated verification requirements, verification methods, and allocation. This table is called Requirements Allocation Sheets (RAS).

2.3.4 Requirements Traceability

Each requirement should be traceable to the higher level requirement(s) all the way to customer requirements (CRs). If not traceable to a CR, it could be a new CR or this requirement is not needed. The top-level of requirements traceability hierarchy is CRs, followed by top-level systems requirements, then the next level systems requirements and continue to the lowest level requirements, usually component requirements. This total traceability is necessary to insure all the CRs are incorporated and complied. Requirements traceability for the top two levels of systems requirements should be developed jointly by customer and contractor. Requirements

traceability is two-way protection between customers and contractors. The customer will know which CR or CRs are ignored; on the other hand, if the upward trace finds new CR or CRs, it will be out-of-scope and the customer has to increase the scope in cost and schedule; otherwise, the customers should delete the extra CR or CRs. Once the traceability hierarchy tree is established, any impact of top-level CR or CRs on the lower-level requirements can be easily found. Conversely, the impact of any middle-level requirement changes or removals can be easily found on higher level as well lower level requirements. Parent requirements flow down the hierarchy to the immediate lower level child requirements. Traceability means having clear knowledge of the ancestry of every requirement in terms of the parent requirements that make it necessary. CRs have child requirement. The lower level requirements have both parents and children requirements. Orphan requirement(s) that has no parent requirement is not allowed. Through a traceability tree one can spot those orphan requirements. The orphan requirements need parent requirement(s) or have to be deleted. Software is usually the source of problem for hardware systems as well as software-intensive system; therefore, it is important to have total traceability from CRs down to lines of coding.

There are three (3) ways to develop a requirements traceability tree. One way is to use block diagrams to show the traceability from top-level to lower-level requirements. A second way is a dedicated traceability matrix. The third way is to use computer tools. The most commonly one used in the systems engineering communities is DOORS [15].

2.3.5 An Example

After discussions of requirements analysis, allocation, and traceability in the above, an example may be necessary to reinforce the understanding. The example is to develop CRs, system requirements, design requirements, RAS and a traceability tree for a Mouse Trap [16].

The Mouse Trap CRs are shown in Table 2.2. As discussed above, CRs have no rules and restrictions. It could come in different formats and styles, sometimes, not even a sentence, just words or phrases. But shown here in Table 2.2, CRs are nicely written, at least understandable; therefore, the validation of CRs is not necessary. The

Table 2.2 Mouse trap customer requirements

No.	Requirement	Description
CR1	Simple device	There is a need to kill mice with a simple device
CR2	Simple to operate	This device should be simple to operate and affordable by the general population
CR3	Harmed instantaneously	The mouse will be harmed instantaneously
CR4	Price cheap	The price for the Mouse Trap device should be cheap

Table 2.3 Mouse trap system requirements

No.	Requirement	Description
SR1	Cost $\leq$ \$0.50	The cost of the simple device shall not be more than \$0.50
SR2	Mechanical design	This device shall be mechanically designed without any sensors and software
SR3	Attracted to enter	A mouse shall be attracted by cheese to enter the device
SR4	Harm when bait disturbed	The device shall harm the mouse in less than 1/2 s when the bait is disturbed
SR5	Operated by 90 percentile	The device shall be simple to be operated at least by 90 percentile of the general population

system requirements (SRs) are shown in Table 2.3. As can be seen from Table 2.3, the complete sentences of SRs are simple, specific and straightforward, using the word "shall". SR1 is derived from CR4 and CR1 by comparing with the market prices of mouse traps. SR1 contains quantity of \$0.50. SR2 is derived from CR1 and CR2. Mechanical design is corresponding to simple device and simple to operate. SR3 is derived from CR3, the need to kill mice. SR3 specifically refers to cheese (requirement engineer's choice). SR4 is derived from CR2 and CR3 to harm the mice in less than one-half second (this is a quantity) when the bait is disturbed and simple to operate. Less than one-half second is determined by trade-off studies or by building a prototype. We should strive to get the slowest possible action time since the slower the action time the less the cost. It is because a lower spring strength will give a higher action time with lower spring costs. This satisfies the margin for a requirement discussed above. SR5 is derived from CR2 simple to operate by meeting 90 percentile (a quantity) of general population. The design requirements (DRs) are shown in Table 2.4. DR1 is derived from SR1, SR2 and SR5 to hold the cheese firmly and in the meantime to set up the Lock Wire in its place. DR2 is derived from SR1, SR2, and SR4 to release the Kill Mechanism in one-half second or less. DR3 is derived from SR1, SR2, and SR4 to hit the mouse with 20 lb. force. We should strive to get the smallest possible impact force since the smaller the impact force the

Table 2.4 Mouse trap design requirements

No.	Requirement	Description
DR1	Hold cheese	The Bait System shall be designed to firmly hold the cheese and lock wire in its place in one step by meeting 90 percentile of population abilities
DR2	Release $\leq$ ½ s	The Trigger Systems shall be designed to release the Kill Mechanism in less than one-half second
DR3	20 lb force	The Power System shall be designed to hit the mouse with 20 lb force
DR4	Platform support	A platform shall be designed to house all the subsystems of this device

lowest the costs. It is because the lower spring strength will give a smaller impact force with lower spring costs. This is, again, to satisfy the margin for a requirement as discussed above. DR4 is derived from SR1, SR2, SR3, and SR4 to support the subsystems. Different individuals or teams may develop different set of SRs and DRs. For example, peanut butter may be used for bait rather than cheese; action time could be different from one-half second; and the impact force could be different from 20 lb. force, as may be known later in functional analysis. Different functional and system architectures are possible. We can apply synthesis analysis, basically through trade-off studies, to choose the best conceptual design from these different architectures. In a systems engineering process one can select the best conceptual design objectively and directly from CRs.

The Requirements Allocation Sheet (RAS) for the mouse trap SRs is shown in Table 2.5. RAS applies to any levels of requirements. For the example shown here the RAS applies to SRs. DRs can also have its RAS. Every SR has its verification requirement (VR). A VR follows the same requirement rules with a complete sentence. One may notice that a VR repeats words as used in SR, especially concerning quantities and compliance standards. This is necessary since a VR will verify these quantities and compliances with standards. Later on, the VRs and SRs will be dissected. SRs are given to a designer and VRs will be distributed to verification specialists. The verification methods are only for recommendation. Final methods will be determined

Table 2.5 Requirements allocation sheet (RAS)

No.	Requirement	Verification requirement	Method	Department
SR1	The cost of the simple device shall not be more than $0.50	VR1: it shall be verified when the mouse trap is assembled that the cost for manufacturing shall be less than $0.50	Demonstration	Manufacturing and design
SR2	This device shall be mechanically designed without any sensors and software	VR2: every component of the mouse trap shall be mechanically designed	Inspection	Quality
SR3	A mouse shall be attracted by cheese to enter the device	VR3: a cheese shall be placed on a holder with a clear visibility without any obstruction and easy to be reached by 90 percentile of mouse population	Demonstration	Human factor

(continued)

Table 2.5 (continued)

No.	Requirement	Verification requirement	Method	Department
SR4	The device shall harm the mouse in less than 1/2 s when the bait is disturbed	VR4: it shall be verified by test that a mouse shall be harmed in less than one-half second	Test	Design
SR5	The device shall be simple to be operated at least by 90 percentile of the general population	VR5: it shall be verified by demonstration that placing bait on Bait Holder and locking wire to the bait holder can be handled by 90 percentile of population	Demonstration	Human factor

by verification specialists. In this example the SRs are allocated to departments for ownership. One should be aware that requirements may also be allocated to functions or individuals, etc.

Both block diagrams and traceability matrices are applied to the mouse trap. Block diagrams are shown in Table 2.6. The traceability matrix is shown in Table 2.7. For a simple device like a mouse trap, the block diagrams for only three levels already appear complicated. For a larger device or system, it will be overwhelmingly complicated. A traceability matrix can contain more levels, i.e., more columns. If there are more requirements in each level, just add more rows can be added. A traceability matrix is more convenient and adaptable for large systems with more requirements and hierarchy levels.

Table 2.6 Mouse trap requirements traceability using block diagram

Customer requirements	System requirements	Design requirements
CR1: simple device	SR1: price ≤ $1.00	DR1: hold cheese
CR2: simple to operate	SR2: mechanical design	DR2: release ≤ ½ second
CR3: harmed instantaneously	SR3: attract to enter	DR3: 20 lb force
CR4: price cheap	SR4: harm when bait disturbed	DR4: platform support
	SR5: operated by 90 percentile	

Table 2.7 Mouse trap requirements traceability matrix

Customer requirement	System requirement	Design requirements
CR1	SR1	DR1, DR2, DR3
CR1	SR2	DR1, DR2
CR2	SR2	DR1, DR2
CR2	SR4	DR3, DR4
CR2	SR5	DR1
CR3	SR3	DR1, DR4
CR3	SR4	DR3, DR4
CR4	SR1	DR1, DR2, DR3

2.4 Functional Analysis and Allocation

When referring to Fig. 2.6, Requirements Analysis is followed by Functional Analysis in an iterative way. If one recalls the RAS in which each system requirement can be allocated to a function, more than one system requirement can be allocated to the same function. After the two top-level system requirements have been developed, the top-level functional analysis can be performed using the functions allocated from the top-level system requirements. If there are inconsistencies with top-level system requirements, requirements analysis and functional analysis will be iterated to correct the inconsistencies. functional analysis will be continued to the next level using the functions allocated from the next level system requirements. Again, if there are inconsistencies between the next level functional analysis and the next level system requirements, they should be corrected iteratively. Then the top and next level functional architecture is formed. The lower level functional analysis and architecture can be continuously developed without waiting for the corresponding lower level requirements developed. The lower level functional analysis and architecture can assist the requirements analysis to develop lower level requirements. Some of the organizations, such as the Commercial Satellite Division of The Boeing Company, develop the functional analysis and architecture first. Then the developed functional architectures were used to develop requirements for all levels. The iteration between requirement analysis and functional analysis is called requirements loop as shown in Fig. 2.6.

When the top two levels of functional architectures are developed, through functional allocation, the two top levels system (product) architectures are developed. As discussed in Sect. 2.3.5, several system architectures may be developed. system synthesis can be performed to select the best system architecture to develop the best conceptual design. The developed system architectures can be checked against the functional architectures for consistencies since functional architectures are derived from customer requirements. These consistency checks will be iterative throughout the system architectural hierarchical levels between system synthesis and functional analysis and allocation. It is called design loop iteration as shown in Fig. 2.6.

2.4.1 Functional Analysis

Functional analysis is an important first step in determining system performance. It includes functions necessary for the product or service to operate properly. It is a structured approach for describing how a system might be used. The functional blocks will be defined through a series of functional analysis in all levels. The contractually specified usage modes are also included in functional blocks which are usually in the top or higher levels. These functional blocks in all levels hierarchically form a functional architecture for which system products and services can be designed. The operational sequence in time steps, for example, the last ten (10) seconds to launch a rocket, can be arranged as time sequence in functional blocks, that can be used to analyze time-critical requirements. The functional blocks, like the requirements, are arranged in a traceable and logical sequence.

Functions describe how users use a product or service. A functional statement begins with a verb and follows with a direct object, for example, fly airplane, surf internet, or enter password. As one moves away from user-interface level and into lower levels of details, functional descriptions become statements about what the system does, for example, compute coordinates, sense hydraulic pressure, or track target. Function name should identify the action or transformation accomplished by the function. Avoid the pitfalls of "provide" and "accept" functions since these two words cannot send out clear message. What does the function mean with either of these two words? For example, "provide diagnostics", what kind of diagnostics? A better way to write this function is "Perform BIT (Built in Test)". "Provide aircraft position", a more clear message is "Compute aircraft position."

2.4.1.1 Functional Flow Block Diagram

One of the often used functional analysis methods is Functional Flow Block Diagram (FFBD). Refer to Fig. 2.9 [17], the top-level function blocks are transformed from top level system requirements. There are two ways of transforming system requirements to functional blocks. One way to convert system requirements to functional blocks is through interpretation and judgement. If the system requirements have already been allocated to functions, it will be simply using the allocated functions as top-level functional blocks. The function blocks are connected in certain sequences, as shown in Fig. 2.9, Functions A to B to C to D, and Function A also to E to C to F. You would not want to be overwhelmed by too many blocks to perform complicated sequential relationships, one to one, one to many, and many to one, etc. It is recommended between five (5) to nine (9) blocks up to your preference. Each functional block is numbered, Function A as 1.0, Function B as 2.0, Function C as 3.0, Function D as 4.0, Function E as 5.0, and Function F as 6.0. Each block will have the next level functions. Let us choose Function E, 5.0. Continue to Fig. 2.10 for the next level FFBD. Since the Functions are all under Function E, aka 5.0, they will be numbered as 5.x. The functional sequences will be Function 5.1 to Function 5.3 to Function

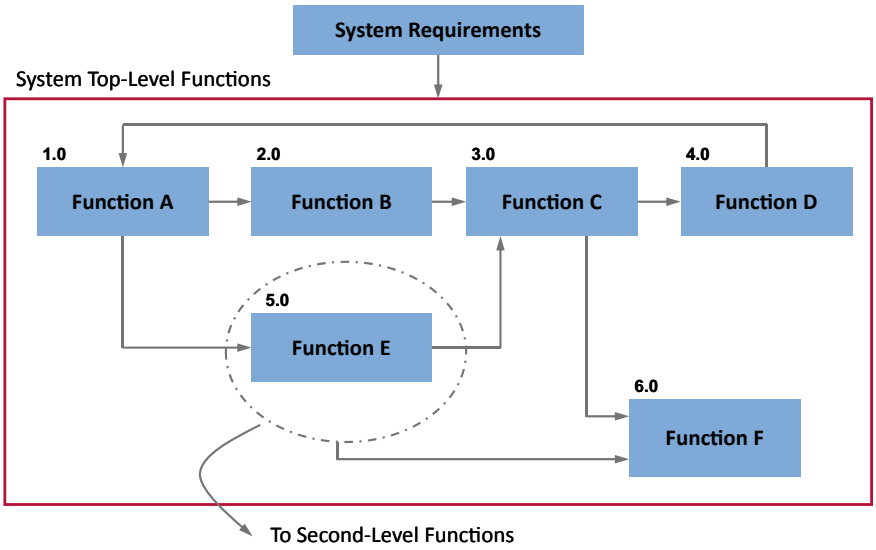

Fig. 2.9 Top-level functional flow block diagram [17]

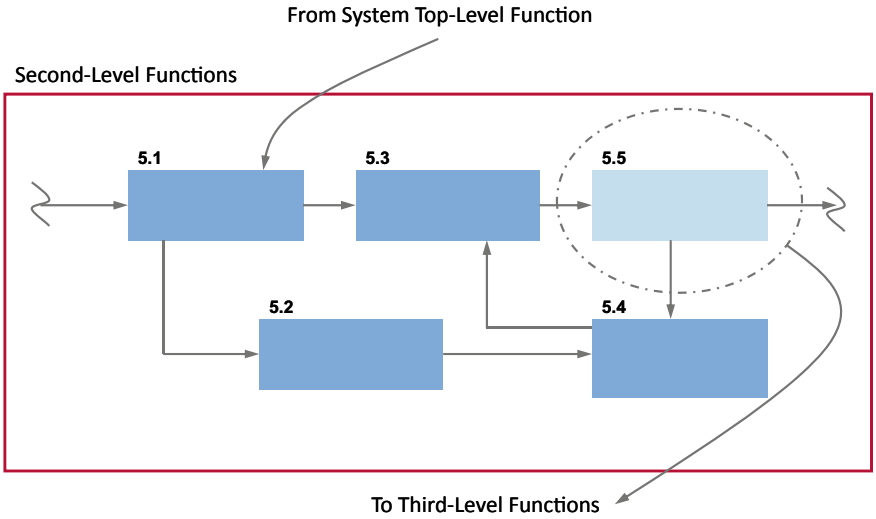

Fig. 2.10 Second-level functional flow block diagram [17]

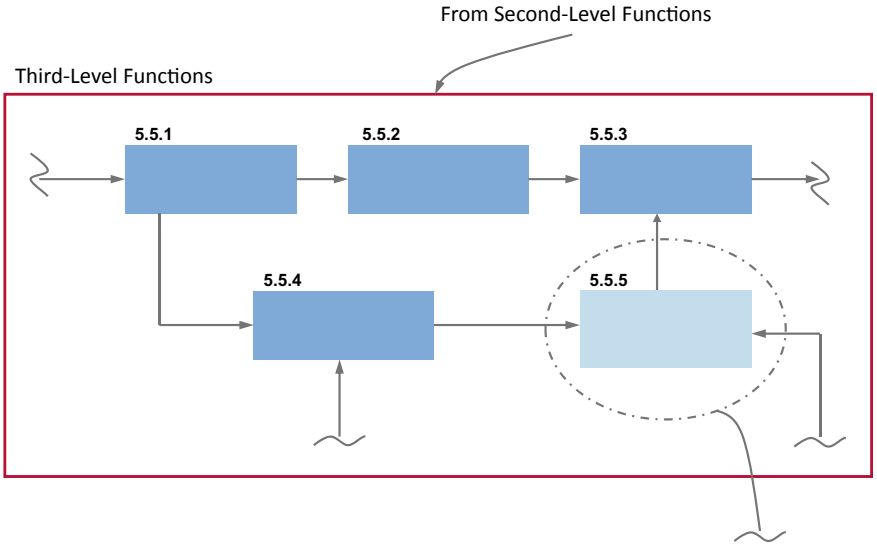

Fig. 2.11 Third-level functional flow block diagram [17]

5.5 to Function 5.4, Function 5.1 to Function 5.2 to Function 5.4, and Function 5.4 to Function 5.3. Now you may begin to appreciate why the functions are numbered. Each of 5.x Functions can have lower level functions. Let us choose Function 5.5, as shown in Fig. 2.10, to develop lower level functions. Continue to Fig. 2.11 to view the next lower level FFBD. The functional sequences will be Function 5.5.1 to Function 5.5.2 to Function 5.5.3, and Function 5.5.1 to Function 5.5.4 to Function 5.5.5 to Function 5.5.3. Then we can continue to next lower level functions, for example, from Function 5.5.5, as shown in Fig. 2.11. All the functional blocks 1.0, 2.0, 3.0, 4.0, 5.0, and 6.0 at the top-level are grand-parents level; all the functional blocks 5.1, 5.2, 5.3, 5.4, and 5.5 are parents level under grand-parent 5.0; all the functional blocks 5.5.1, 5.5.2, 5.5.3, 5.5.4, and 5.5.5 are children level under parent level 5.5. Therefore, from the function numbers can identify the hierarchy level under which function and above which functions. The top-level has only one FFBD. The second-level can have six (6) FFBDs under each grand-parents functional block. The numbers of FFBD in third-level will depend on how many functions under each functional blocks of second-level. For example, under second-level Function 5.5 will have five (5) FFBDs. It could be as many as thirty (30) FFBDs if each second-level function has five (5) FFBDs. All these functional blocks in each level piled in hierarchical layers, it forms functional architecture as shown partially in Fig. 2.12 focused on the branch of Function 5.0 to Function 5.5. As you can see that the functional architecture is fanned out from top-level to lower levels.

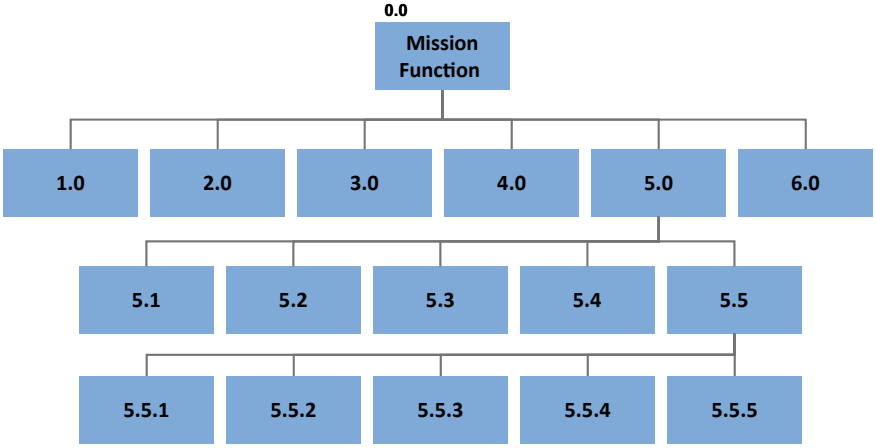

Fig. 2.12 Partial functional architecture [17]

Example 1 Use "Drive a Car", as shown in Table 2.8, as an example for FFBD. In the table lists the functions of driving a car. Each individual will drive the car in different ways. Shown in Fig. 2.13 is one way of driving a car. There are many functional sequences for driving a car pending on who is driving.

Example 2 Use "Dishwasher", as shown in Figs. 2.14, 2.15, 2.16, 2.17, 2.18 and 2.19 [18], as another example for FFBD. It is shown in Fig. 2.14 that through interpretation and judgement, top-level system requirements for dishwasher has been transformed to top-level FFBD for which Function 1.0 to Function 2.0 to Function 3.0 to Function 4.0. From the top-level four (4) functions generate four (4) second-level FFBDs. The

Table 2.8 Functions for driving a car

Drive a car
Accelerate car
Decelerate car
Turn car
Start car
Stop car

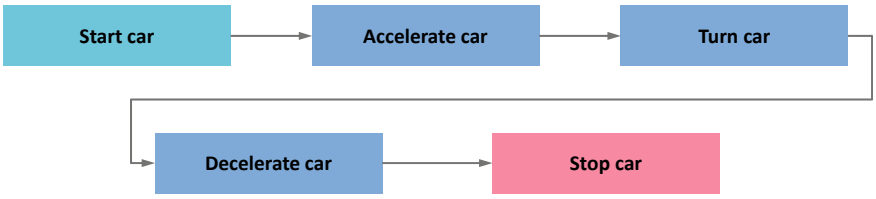

Fig. 2.13 Functional flow block diagrams—car

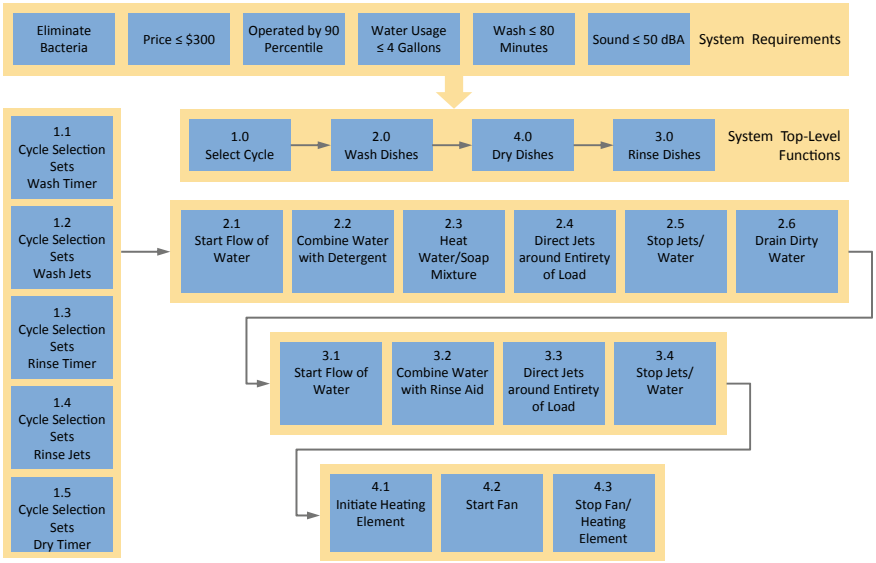

Fig. 2.14 Top-level and second-level functional flow block diagram—dishwasher [18]

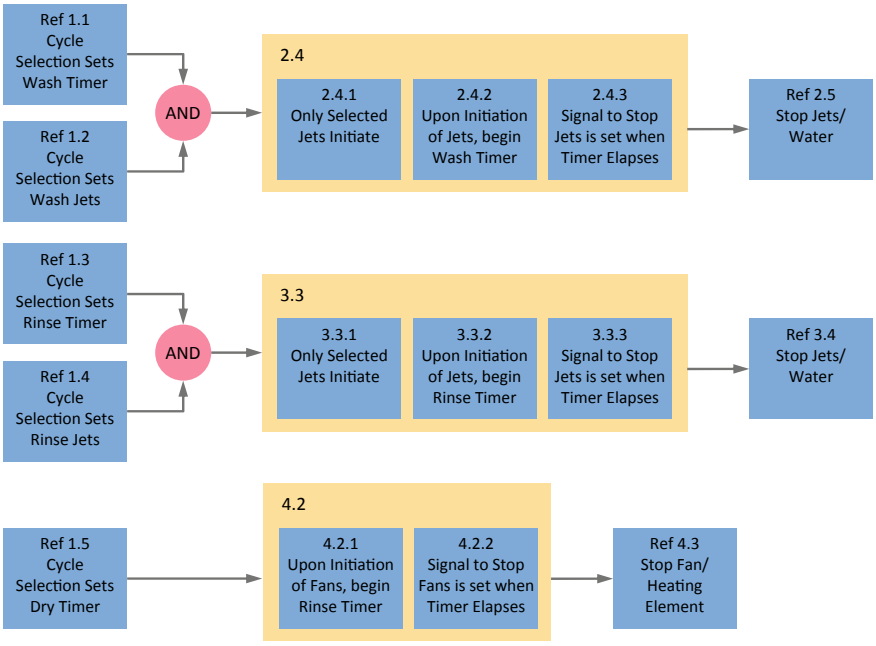

Fig. 2.15 Third-level functional flow block diagram—dishwasher [18]

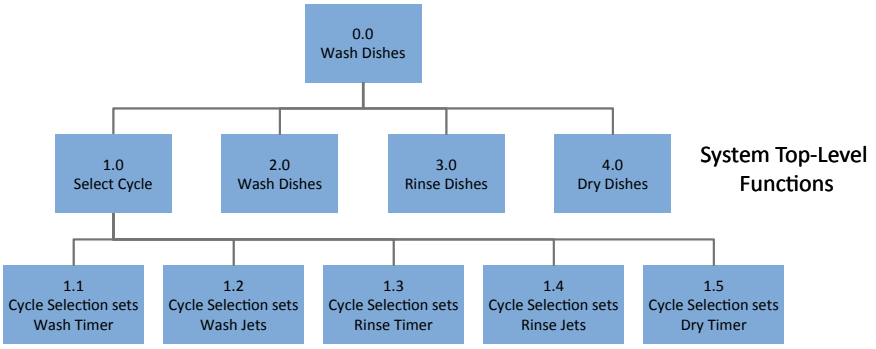

Fig. 2.16 Function 1.0 functional architecture—dishwasher [18]

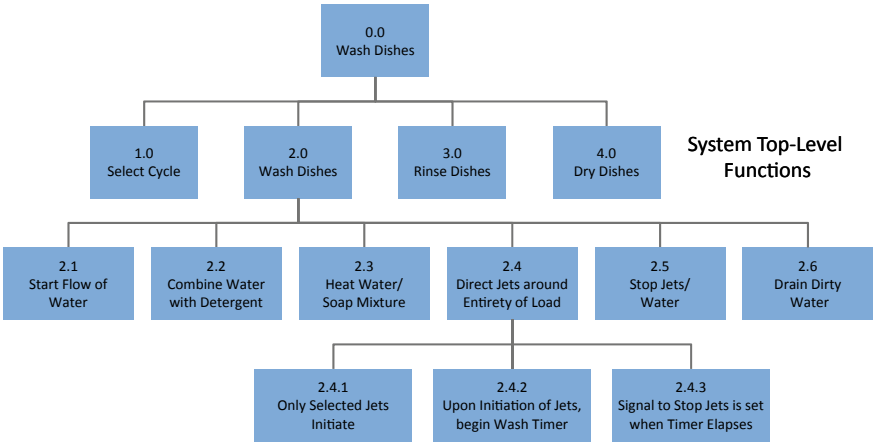

Fig. 2.17 Function 2.0 functional architecture—dishwasher [18]

functional sequences are shown in each of the second-level FFBD. The third-level FFBDs for the second-level Function 2.4, Function 3.3, and Function 4.2 are shown in Fig. 2.15. The external functions connected to each third-level FFBD are also shown in Fig. 2.15. It can be seen that the external interfaced functions can be in different levels. The word of "Ref" does not have to be included. When the FFBDs are completed, the generated functional blocks can be piled hierarchically in different levels to establish functional architecture. Function 1.0 branch, Function 2.0 branch, Function 3.0 branch, and Function 4.0 branch functional architectures are shown in Figs. 2.16, 2.17, 2.18, and 2.19, respectively.

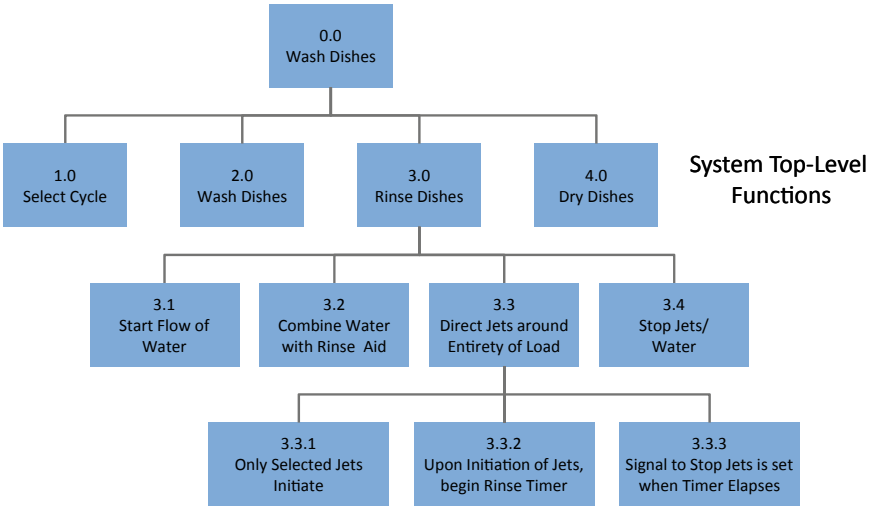

Fig. 2.18 Function 3.0 functional architecture—dishwasher [18]

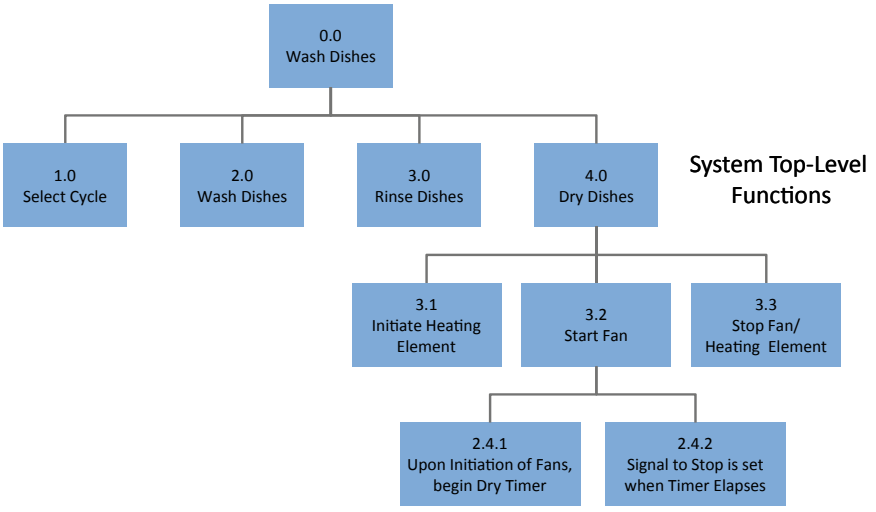

Fig. 2.19 Function 4.0 functional architecture—dishwasher [18]

2.4.1.2 Integrated Definition for Functional Modeling

There are as many as fourteen (14) Integrated Definition for Functional Modeling diagrams, i.e., IDEF0, IDEF1, 2, 3, 4, and 5, up to 14. IDEF0 is another commonly used functional analysis method. It is a functional model or process model of a system; a method designed to model the decisions, actions, and activities of an organization or

system. It is useful in establishing the scope of an analysis, especially for a functional analysis. As a communication tool, IDEF0 enhances domain expert involvement and consensus decision-making through simplified graphical devices. As an analysis tool, IDEF0 assists the modeller in identifying what functions are performed, what is needed to perform those functions, what the current system does right, and what the current system does wrong.

The IDEF0 modeling diagram is shown in Fig. 2.20 [19]. There are two additional inputs as compared with FFBD. The Control enters the top of the box. The Mechanism points up to the bottom of the box to show the supporting means for performing the function. Use "Perform detail design" function as an example, shown in Fig. 2.21, to show how to form the IDEF0 diagram. The input data to the left-hand side of the box is Preliminary Design Data; the output data from the right-hand side of the box is Recommended Detailed Design; the control data from the top of the box is Design Requirements; and the mechanism data enters the bottom of the box is Design Engineer. The control input can also include the standards (industrial, commercial, and military), constraints, and processes, etc. The mechanism data can also include resources (facilities, computer, etc.), people, and tools, etc. IDEF0 can have functions at different levels same as FFBD, as shown in Fig. 2.22. The output data does not

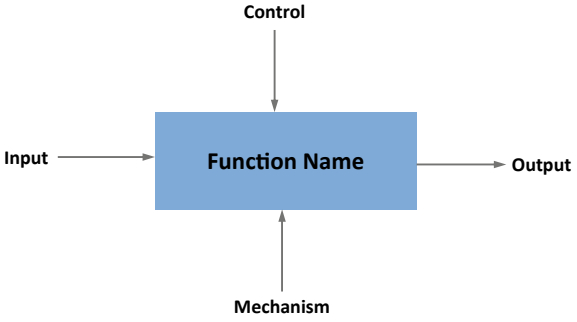

Fig. 2.20 IDEF0 modeling diagram [19]

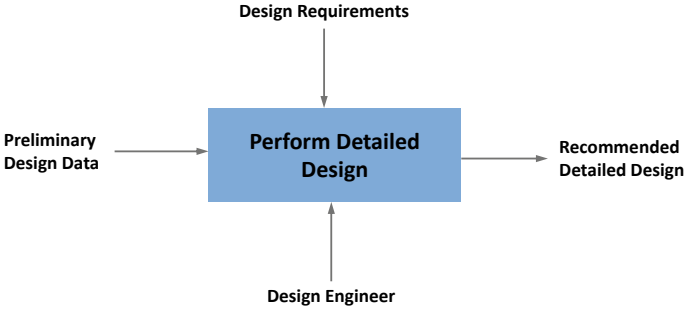

Fig. 2.21 IDEF0 modeling diagram—example

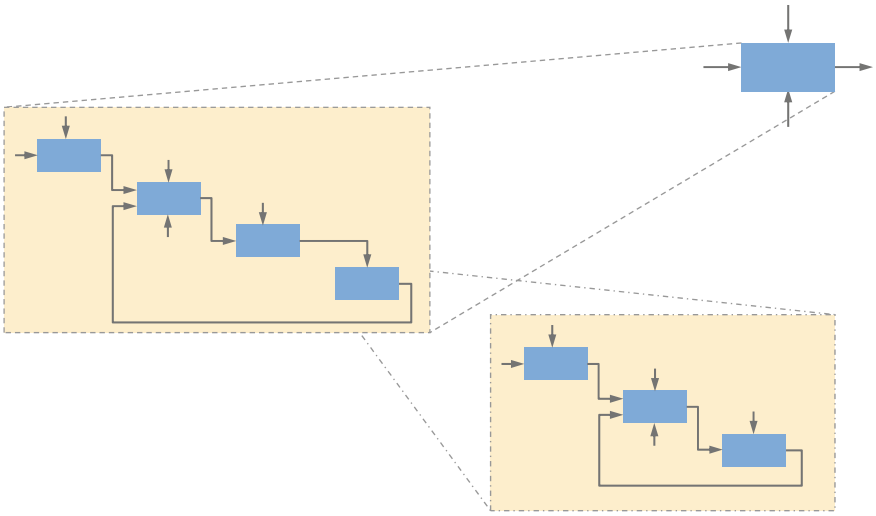

Fig. 2.22 Three levels of IDEF0 diagrams [19]

have to be the input data to the next connected diagram; instead, it can input as control data or mechanism data, as shown in Fig. 2.22. An example of IDEF0 diagram is shown in Fig. 2.23 [19]. IDEF0 can also use numbering system for each function same as that in FFBD, as shown in Fig. 2.24 [20], another IDEF0 diagram example included in Architecting the Communication and Navigation Networks for NASA's Space Exploration Systems. The IDEF0 diagram Function A1.3.1 to Function A1.3.2 to Function A1.3.3 sequence are children functions of A1.3.

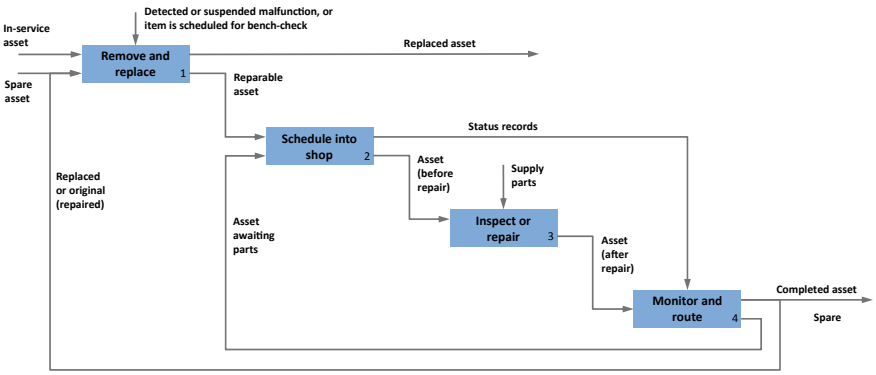

Fig. 2.23 IDEF0 diagram example—maintain reparable spares [19]

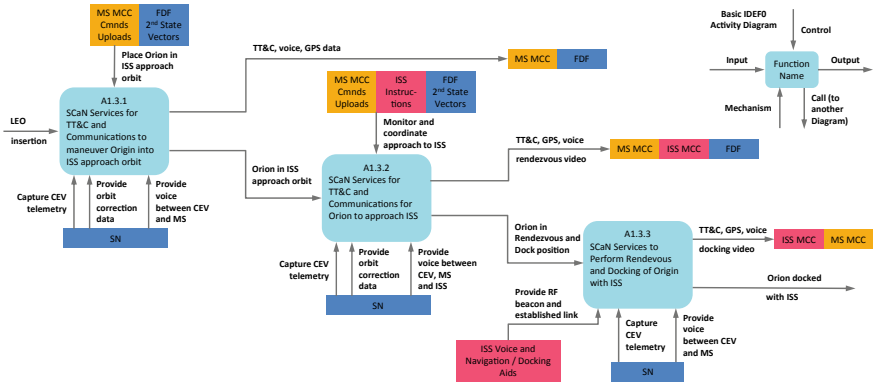

Fig. 2.24 IDEF0 diagram for operational activities flow diagram in support of Rendezvous and Dock [20]

2.4.1.3 Functional Decomposition

As discussed above, functions from top-level down to the lowest levels are developed by using FFBD or IDEF0 functional analysis method. Functions developed through this process are time consuming, especially for a large and complex system. A simpler way is to decompose top-level functions to the lowest levels in its most basic form a simple hierarchical decomposition of the functions. Functional decomposition is the breaking down of a high level function into smaller pieces of function that can be more easily managed and understood. Decomposing the top-level functions into sub-functions (i.e. Level 1 and 2, etc.) can also form functional architecture.

The primary steps for functional decomposition are:

1. Brainstorm functions performed.
2. Pick out the five to ten truly top level functions and arrange in sequence (if appropriate).
3. Place the other functions below the top-level functions.

A practical approach is to use a roll of white paper with 22 inches or similar width. Roll out the roll of white paper. Draw the top-level, second-level, third-level, and fourth-level regions, etc. on the roll of white paper. Team members write the names of functions on post-it notes. Remember to use the verb-noun naming function format. Team members can lay the post-it notes on different regions which already drawn on the roll of white paper. If there is a contention about where a function belongs, make a duplicate post-it note and put in both regions. When you get uncomfortable about further decomposition, it is usually the end of decomposition. Then the team members will align the functions in different regions. The top-level region can have only 5–9 functions. In the second-level region, group of 5–9 functions should be aligned to each function in the top-level region. In the third region, group of 5–9 functions should be aligned to each function in the second-level region. In the fourth region, group of 5–9 functions should be aligned to each function in the third-level

region. In the same manner, work to further lower regions if there are more. The groups under each function should be between 5 and 9. The outcome from functional decomposition is a functional architecture.

2.4.2 Elements of Functional Analysis

There are four elements:

- Functional Decomposition
- Functional Sequencing
- Information/Data Flow
- Interface Definition

Function decomposition was discussed above in Sect. 2.4.1.3. Referring to Figs. 2.9, 2.10, 2.11, 2.13, 2.14 and 2.15, it can be seen that the functional blocks are connected in certain directions and sequences. This is the element of functional sequencing. When the two functional blocks are connected, information/data is transferred from one block to another block. It is directionally oriented. For example, Block A connects in the direction to Block B, asking "have you had dinner yet?" Block B connects in the direction back to Block A, answering "yes, I had dinner." This is the element of information/data flow. The information/data between the two blocks can be expanded to include all the necessary interface information/data. This is the element of interface definition where the interface requirement is defined and developed. The functional definition (requirement) shall be fulfilled by physical interface definition (requirement) that will be discussed under the systems engineering subject of Interface Management.

2.4.3 Functional Allocation and System Synthesis

The functions will ultimately be performed or accomplished through the use of equipment, personnel, facilities, software, or a combination. The functional architecture will need to be transformed into a physical, or software architecture by defining physical or software components needed to perform the functions. Functional partitioning is the process of grouping functions that logically fit with the components likely to be used, and to minimize functional interfaces. Functions at system, sub-system, segments and components levels should be allocated to the corresponding levels of physical or software system, sub-system, segments, and components. Functional analysis and allocation is repeated to define successively lower level functions and allocations, as shown in Fig. 2.25. When the allocations are performed to the lowest level, a system architecture, physical or software, is formed. The functional/physical matrix, shown in Fig. 2.26 [21], can be used to assist the functional allocation.

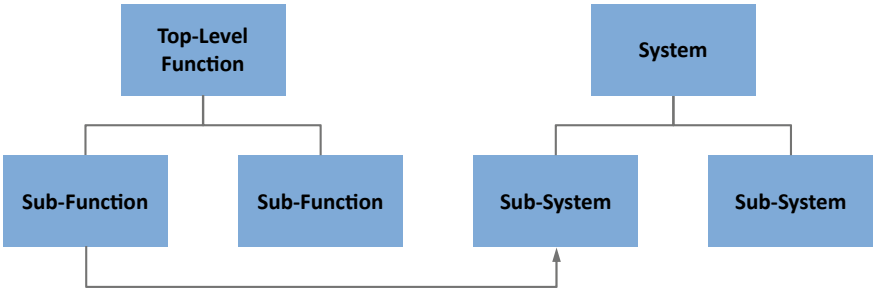

Fig. 2.25 Functional allocation

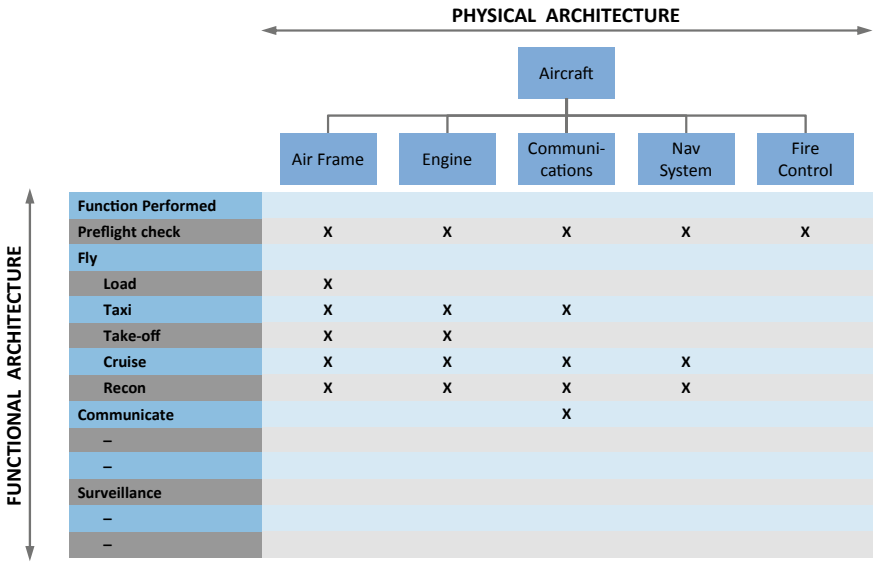

Fig. 2.26 Functional/physical matrix

Since there may be several physical or software architectures developed from different functional architectures or from the same functional architecture through different ways of allocations, design synthesis sets the stage for trade studies to select the best among the candidate architectures.

2.5 Summary and Prospective

Referring to Fig. 2.6, there are more systems analysis and management subjects than what are presented here in this chapter, for example, risk management [22], Work Breakdown System (WBS) [23], Integrated Master Planning (IMP)/Integrated

Master Scheduling (IMS) [24, 25], Technical Performance Measurement (TPM) [26], Verification and Validation [27], and System Integration, etc. [28]. One could write a whole book and more on systems engineering. As these are fundamentals and important subjects they may be presented in another book.

It is worth discussing here how to educate people to become systems engineers. In the past seventeen (17) years, the author has taught systems engineering. In the author's opinion, every employee should be a systems engineer in addition to their domain knowledge. In Japan, there is no Quality Assurance Department in the company organization since every employee, whether engineer, manufacturing worker, software programmer, or subcontractor employee, etc., is equipped with quality knowledge, methods, and conscientiousness. The same strategy can be applied to systems engineering.

References

1. Friedman G (1994) Statement made as president of INCOSE
2. Moody JA et al (1997) Metrics and case studies for evaluating engineering designs. Prentice Hall PTR, Upper Saddle River, NJ
3. Honour E, Mar B (2002) Value of systems engineering – SECOE research project progress report. In: INCOSE international symposium
4. Honour E (2004) Technical report value of systems engineering. Lean Aerospace Initiative (LAI)
5. United States Government Accountability Office (GAO) (2008) Best practices – increased focus on requirements and oversight to improve DoD's acquisition environment and weapon systems quality
6. Meier S (2010) Casual inferences on the cost overruns and schedule delays of large-scale U.S. federal defense and intelligence acquisition programs. National Reconnaissance Office, Chantilly, VA, USA
7. Eiler J (2008) Systems engineering. Aerospace America, AIAA (American Institute of Aeronautics and Astronautics), Reston, VA, USA
8. Assistant Secretary of the United States Air Force (2003) Released letter to defense contractors, USA
9. Under Secretary of the United States Department of Defense (2004) Released letter to defense contractors, USA
10. Vice-President of The Boeing Company (2013) Systems engineering in today's competitive environment. Presentation for aerospace engineering seminar at California State University Long Beach
11. Senior Manager of Systems Engineering of The Boeing Company (2014) Current and future trends in systems engineering. Presentation for aerospace engineering seminar at California State University Long Beach
12. Jackson S, Hsu J (2002) Systems engineering lecture notes. University of Southern California, California State University Long Beach and University of California, Irvine
13. Defense AT&L (Acquisition, Technology and Logistics) (2005) Revitalizing systems engineering - how six components are meeting the acting USD (AT&L) imperatives
14. Hsu J (2016) Systems engineering lecture notes. California State University Long Beach, University of California Irvine, AIAA continuing education, USA
15. IBM Rational Dynamic Object Oriented Requirements System (DOORS) (formerly Telelogic DOORS) is a requirement management tool

16. Hsu J (2006) Applying systems modeling language to a simple hardware systems. In: INCOSE international symposium, Orlando, FL, USA
17. Jackson S (2003) Systems engineering lecture notes. University of Southern California, Los Angeles, USA
18. Lessmueller B, Scribner V, Vilchez H, Rodriguez Y (2006) Systems engineering class projects. California State University Long Beach
19. Defense Acquisition University (2001) Systems engineering fundamentals. Supplement 5-B
20. NASA (2007) Architecting the communication and navigation networks for NASA's space exploration systems. In: IEEE international conference on system of systems engineering
21. Defense Acquisition University (2001) Systems engineering fundamentals, chapter 6
22. Aven T (2016) Risk assessment and risk management: review of recent advances on their foundation. Eur J Oper Res 253:1–13
23. Sharon A, Dori D (2015) A project-product model-based approach to planning work breakdown structures of complex system projects. IEEE Syst J 9(2), 6748857:366–376
24. Sofian S, Li X, Kusumawardhani P, Widiyani W (2015) Sustainable systems integration model-metrics in design process. Procedia Soc Behav Sci 184:297–309
25. Hsu J, Raghunathan S (2007) Systems engineering for CDIO - conceive, design, implement and operate. In: 45th AIAA aerospace meeting and exhibit, 8–11 Jan 2007, Reno, Nevada, AIAA 2007-591. https://doi.org/10.2514/6.2007-591
26. Sun J-F, Yuan J-H, Zhao Y, Pu H-B (2013) Technical performance measurement method for technical management of system development projects. Binggong Xuebao/Acta Armamentarii 34(suppl. 2):44–47
27. Dabney JB, Arthur JD (2019) Applying standard independent verification and validation techniques within an agile framework: identifying and reconciling incompatibilities. Syst Eng 22(4):348–360
28. Wognum N, Bil C, Elgh F, Peruzzini M, Stjepandić J, Verhagen WJC (2019) Transdisciplinary systems engineering: implications, challenges and research agenda. Int J Agile Syst Manage 12(1):58–89
29. Kepner CH, Tregoe BB (1997) The new rational manager. Kepner-Tregoe Inc, Princeton, NJ