1
INTRODUCTION TO SYSTEMS ENGINEERING

While elements of systems engineering [1] are recognizable in all major engineering ventures throughout history, the discipline is relatively young. The term was first used in Bell Telephone Laboratories in the 1940s in the context of taking a systems view when engineering networks. Conference and journal papers [2] began using the term in the 1950s but many were still more related to 'engineering a system' rather than 'systems engineering' as we would use the term today. Modern use began to emerge with three seminal papers from Cole [3] and then a number of authors [4] began to define the practice of not just engineering a system but also the need for management and integration of specialist disciplines. The first book on systems engineering was published in 1957 [5] and the discipline was taught at universities from the 1960s.

More-formal methodologies and practices began to emerge from experience gained in the US Department of Defense acquisition programs in the late 1940s and early 1950s when, for the first time, the scope of system acquisition began to outstrip the ability of traditional engineering practices to cope with complex and challenging user requirements that tended to be incomplete and poorly defined. Additionally, most programs entailed high technical risk because they involved a wide variety of technical disciplines and the use of high technology. Following a number of program failures, the discipline of systems engineering emerged to help avoid, or at least mitigate, some of the technical risks associated with complex system acquisition. Since that time, systems engineering processes and methodologies have continued to develop and are now widely applied throughout the life cycles of modern systems, not just in the traditional defence and aerospace domains but also in relatively new domains such as transportation and health.

Systems engineering provides the framework within which complex systems can be adequately defined, analyzed, specified, designed, manufactured, operated, supported, and ultimately retired. The focus of systems engineering is on the system as a whole, and the maintenance of a strong interdisciplinary approach. Project management, quality assurance, integrated logistics support, and a wide variety of engineering disciplines are but a few of the many disciplines that are part of a coordinated systems engineering effort.

Use of examples in this book. Throughout the following chapters we use a number of examples to illustrate and reinforce the theory being introduced. To aid an understanding of the whole systems engineering process, we use two system examples: a larger system based on the acquisition of an aircraft system, and a smaller system based on the development of a domestic security alarm. We must state at the outset, however, that we do not intend to replicate the design process for either system or their supporting elements. Rather, the systems have been chosen as convenient examples that can be readily recognized by readers from a wide variety of disciplines and specialties. That is, readers are not forced to become domain experts in a particular field just to understand the illustration—the majority of readers can immediately understand the system context, the business needs, stakeholder needs, and system needs; the subsequent requirements; the interface issues; technical performance measures; the logical-to-physical translation; broad trade-off analyses; as well as the physical configuration items involved in the final design. It should be noted that we do not at all suggest that the aggregation of examples throughout the text represents an adequate design for either system; the available space prohibits the inclusion of sufficient detail, which would also obscure the general lessons that are to be illustrated by the examples.

Example 1.1: Introduction to Aircraft Example

An aircraft operator (ACME Air) has identified a business need for a medium-sized aircraft to replace the aging platform that it currently operates over domestic routes and some short international routes. The company will use a systems engineering approach to ensure that the new aircraft system is ideally suited to the role and to ensure the overall commercial viability of the project.

Example 1.2: Introduction to Domestic Security Alarm Example

Another division of ACME Industries, ACME Alarms, has a business need to develop a domestic security alarm. The company proposes to sell the alarm to the domestic market to compete in price and functionality for all forms of domestic dwelling such as houses, flats, and apartments. The alarm is to be capable of being installed by the customer and must be able to operate in a back-to-base monitored mode as well as a stand-alone mode.

1.1 WHAT IS A SYSTEM?

Before we begin to address the discipline associated with engineering a system, we need to consider what is meant by a system—particularly since ‘system’ is perhaps one of the most over-used words in the English language. There are physical systems such a solar systems, river systems, railway systems, satellite systems, communication systems, information systems, pulley systems, nervous systems, just to name a few. There are philosophical systems, social systems, religious systems, gambling systems, banking systems, systems of government, and many more. The word is even used for more-esoteric examples such as the consideration of individual and social behaviour as a system of purposeful events [6]. Before we continue, therefore, we should briefly consider what we mean by a *system* in the context of *systems engineering*.

1.1.1 Definition of a System

The common aspect of the use of 'system' in these varied contexts stems from its early use (and its Greek root) to refer to the whole (or the set) that results when a number of things have been grouped together in a particular manner, for a particular reason. In systems engineering, ISO/IEC/IEEE 15288 therefore defines a system as *a combination of interacting elements organized to achieve one or more stated purposes* [7]. This definition implies that a system comprises internal *system elements* with *interconnections (interactions)* between elements and, by the very act of identifying the system that we are interested in, an external system *boundary* is implied. As illustrated in Figure 1-1, when we draw the boundary around selected system elements, we define the *system of interest (SOI*) which consists of those system elements and their interconnections that exist within the defined boundary.

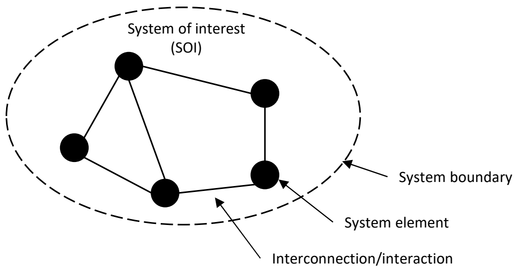

Figure 1-1. An SOI: its elements, interconnections, and boundary.

The purpose of the system is called its *mission*—clearly stated by business management and stakeholders—which represents the start point of the design process as well as providing the basis for the ultimate test of the system's fitness-for-purpose once it has been fielded. In the broadest sense, the mission of the system is to provide a solution to a business problem.

This narrowing of the general use of the term 'system' is very important because it has two major implications:

- When we refer to a system as comprising *system elements* that are *interconnected* in order to achieve the system's *mission*, we imply that all three of those principal aspects result from conscious choice. That is, we are referring to systems that have been deliberately *designed*, or *engineered*—hence our interest in *systems engineering*.
- A system that has been engineered to perform a specified mission must be able to perform that mission with relative autonomy—that is, it must be managerially and operationally independent (and may well have been procured independently). We return to this issue shortly

when we discuss the differences between systems and subsystems (and between systems and systems of systems).

1.1.2 Types of Systems

There are numerous ways to classify systems—here we identify the four main types in order to be clear as to which type of system we refer to in systems engineering (and therefore in the remainder of this book):

- *Closed or open systems.* An open system interacts with its operating environment—it accepts inputs from that environment across its boundary and returns outputs across the same boundary to the external environment. A closed system is isolated from its external environment and is not useful. We are therefore only interested in open systems.
- *Natural or human-made/human-modified systems.* Natural systems contain natural elements and are the result of natural processes; human-made systems come into existence through the efforts of humans and may contain human-made elements or natural elements adapted to human-designed purposes (called human-modified systems). The systems engineering for natural systems is certainly not conducted by humans, so we are only interested in human-made/modified systems.
- *Physical or conceptual systems.* Physical systems exist in a physical form; conceptual systems, such as economic, political and religious systems, do not have a physical form. We focus here on physical systems made up of combinations of integrated hardware and software items.
- *Precedented or unprecedented systems.* In a precedented system, similar such systems (or, at least, the majority of system elements) have been produced before. An unprecedented system is one that has not been previously produced. Systems that comprise mostly unprecedented elements are the result of research and development effort. Here we focus on systems that comprise largely precedented elements—that is, those to which engineering is appropriate.

A wide variety of combinations of the above (and other) characteristics can lead to a large number of types of systems, each of which has markedly difficult properties. It is important to recognize that this book and the majority of the standards discussed refer to *open*, *physical* systems that are *human-made/modified* from largely *precedented* elements.

1.1.3 A System and its Environment

Now, since we are interested in engineering physical systems that are open, our SOI in Figure 1-1 must accommodate *external interfaces* (inputs/outputs) across the system boundary to *external elements* that exist in an external *operating environment* (or perhaps in a related system)—see Figure 1-2.

Sometimes we need to be cognizant of an even wider context so, as illustrated in Figure 1-3, an SOI might be considered as part of a wider SOI (WSOI) within an operating environment, which can be conceived as being part of a wider environment [8].

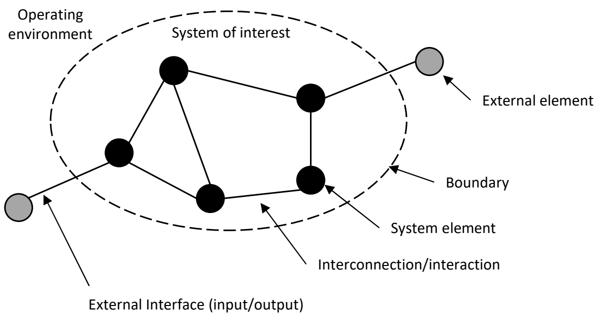

Figure 1-2. An SOI: its elements, interconnections, boundary, and interfaces to external elements in the operating environment.

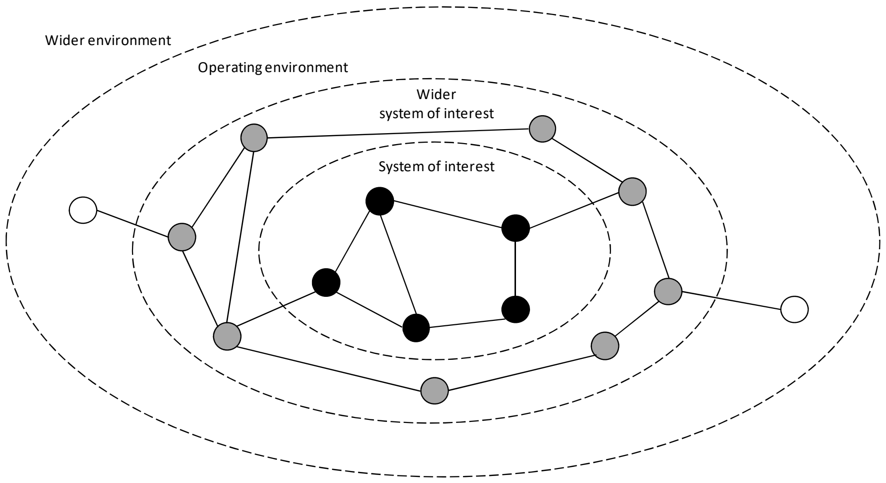

Figure 1-3. An SOI, which might be considered as part of a WSOI within an operating environment, which can be conceived as being part of a wider environment.

1.1.4 A System as a Product

In a physical sense, the term *system* is sometimes considered to be synonymous with *product*—that is, we say that the project is delivering a system, or is delivering a product. A system is normally, however, considered to comprise a number of products. Figure 1-4 shows that ANSI/EIA-632 defines a system as comprising operational products (end products) and enabling products (such as test, training, and disposal products).

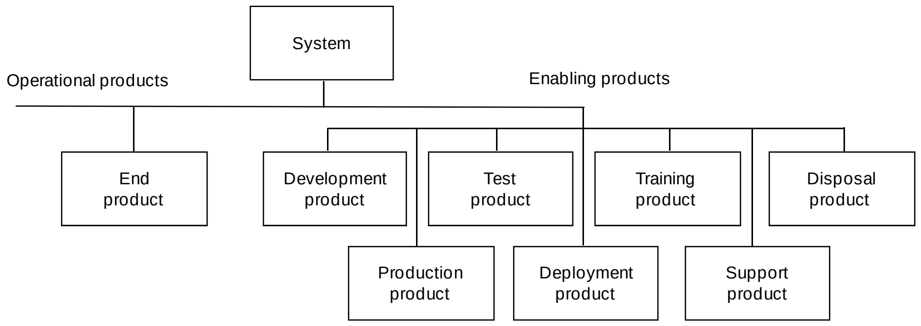

Figure 1-4. ANSI/EIA-632 building block concept of a system comprising operational products and enabling products [9].

1.1.5 A System as a Capability—A Capability System

Before we go any further, however, we must acknowledge that the systems we are interested in are much more than an aggregation of hardware or software products. Consequently, a system must be described in terms of all of its constituent elements, including: the major hardware and software products, the organisation within which it will be fielded, the personnel who will interact with it in many ways, the collective training systems required, as well as the facilities, data, and support (including supplies) required to keep the system in service, and the operating procedures and organisational policies. The system is fully defined by the combination of these resources operating in an operational environment in order to achieve some purpose. In that sense then, we could define a system as delivering an *operational capability*.

It is common, therefore, particularly in defence environments, to refer to the system at this level as a *capability system*. In the US DoD the acronym DOTMLPF refers to the capability system elements of: doctrine, organization, training, materiel, leadership, personnel, and facilities [10]. In Australia, the capability system is considered to comprise fundamental inputs to capability (FIC): command and management, organization, collective training, major systems, personnel, facilities and training areas, supplies, support, and industry [11]. In the United Kingdom, the defence lines of development (DLOD) refer to doctrine and concepts, organisation, training, equipment, personnel, infrastructure, logistics, and information [12]. In Canada, the acronym PRICIE refers to personnel, research and development, infrastructure, concepts and doctrine, information technology, equipment [13].

Having acknowledged all of the elements of a capability system, it must also be recognized that each of the elements will most probably have a different acquisition cycle—for example, people are ‘acquired’ in a different manner to that in which the major equipment will be developed—and each element of the capability may even be acquired through a different acquisition element in the organisation. In the remainder of this text, for ease of description, we focus on the acquisition of the major equipment element (often called the *materiel system*) of the capability system. We must always keep in mind, however, that this acquisition is being undertaken in parallel with the acquisitions of the other elements of the desired capability and that all the elements must be brought back together prior to introduction into service in order to field an operational capability.

Example 1.3: Capability System Elements for our Aircraft Example

Resources for our aircraft system example could include, but not be limited to:

- Major Equipment. *The most tangible part of the system is the hardware itself. The aircraft will be produced, distributed and sold to operators who will then use the aircraft in a number of different ways such as domestic and international operations. Software is also now a critical item within most systems. The aircraft is likely to use hardware and software to control a range of functions from engine management, through navigation and environmental control systems, to the communications and flight-control systems.*
- Personnel. *Air crew are required to operate the system. Ground crew are required to maintain and support the fleet of aircraft.*
- Support. *Maintenance facilities and equipment are required for routine maintenance and repairs throughout the aircraft's life. Materials are required to operate the system, including fuel, lubricants and other consumables such as tyres and spare parts.*
- Facilities. *Other facilities such as terminals are also necessary to operate the aircraft and its support systems.*
- Organisation, Policies and Procedures. *ACME will need to conform to a significant number of regulations and will need appropriate organisational structures, policies and procedures in order to be able to operate the aircraft effectively.*
- Collective Training. *Air crew and ground crew for the system will require training throughout the system life cycle.*
- Data. *Data is required to maintain and operate the aircraft. Data could include maintenance information such as specifications and drawings, and operational information such as user manuals and instructions.*

1.1.6 Logical and Physical Descriptions of a System

A system can be described in two broad ways—in *logical* terms and in *physical* terms. A logical description (historically often referred to as a *functional* description) of a system articulates what the system will do, how well it will do it, how it will be verified, under what conditions it will perform, and what other systems will be involved with its operation. A physical description relates to the system elements and explains what the elements are, how they look, and how they are to be manufactured, integrated, and verified. The logical description contains the 'whats' of the system, and the physical description contains the 'hows'. Both the logical and physical descriptions of a system comprise a series of statements called *requirements*.

The two descriptions are valid independent descriptions of a system, and it is very important that a system is described both logically and physically, focusing first on the logical description:

- In one sense, it is axiomatic that we develop the logical description first. In order to determine whether any particular physical implementation (that is, *how* we are going to implement the elements of the system) is appropriate, we first must understand (from the logical architecture) *what* it is that we want the system to do (that is, what purpose it serves). We therefore need to focus on the logical description (what) first, from which a series of candidate physical descriptions (how) can be developed, one of which can be selected as the preferred physical solution.
- We also must not allow the way in which we implement current physical systems to colour unnecessarily the way in which we might describe future systems. An initial focus on the logical description therefore allows us to provide novel solutions to new (or even old) problems—if we focused on the physical description initially, we would always tend to solve new problems with old physical building blocks.
- Upper-level trade-offs and feasibility analyses must be conducted at the logical level before deciding on the physical implementation—if not, significant waste may result from the selection of physical solutions that either perform unnecessary functions or do not possess critical functionality.
- A logical description is ideally suited to the interface between systems engineering and the business case. While it is often possible for the business case to be met directly by an obvious physical solution, it is better for business management to transition from the business case into a more-detailed logical description of what is required before considering how to achieve it in a physical sense. The definition of the logical description before the development of an appropriate physical description therefore moves from the business case to the final physical solution in controlled verifiable steps.
- The logical description changes slowly; the physical description changes much faster, particularly as the pace of technological change quickens. Arguably, for example, the need for, and upper-level logical

description of, an internal combustion engine have changed little (other than performance requirements) over the last two centuries, while the physical implementations have changed dramatically. That is, the purpose of the engine subsystem, as part of the car system, has not changed over the years, but the physical implementation is obviously very different.

In the development of a system, therefore, there are at least two architectural views: a system logical architecture, and a system physical architecture. Of course, these two descriptions are of the same system so they must be related. We will see later how the logical architecture, as outlined in the requirements breakdown structure (RBS), is mapped onto the physical architecture as represented by the hardware and software configuration items contained in the work breakdown structure (WBS).

1.1.7 Hierarchical Descriptions of a System

We saw earlier that ISO/IEC/IEEE 15288 defines a *system* as a combination of *system elements* which interact to achieve a defined mission. Since each of these system elements will need to perform functions allocated to it so that it can contribute to the systems mission, we can consider the system to be a hierarchical composition of system elements, as illustrated in Figure 1-5.

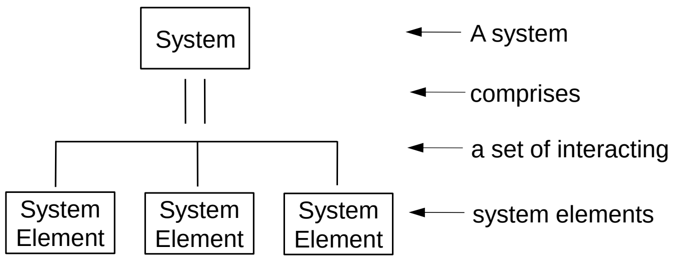

Figure 1-5. A system comprises a set of interacting system elements [14].

The system elements in Figure 1-5 can be logical elements or physical elements, which supports the concepts of a logical architecture and a physical architecture as we discussed in the preceding section.

1.1.7.1 Logical Hierarchy

In a logical description of a system (see Figure 1-6), the system's mission is broken down into a hierarchical structure of its major functions. The logical description or architecture is therefore often called a *functional hierarchy*, or a *functional architecture*.

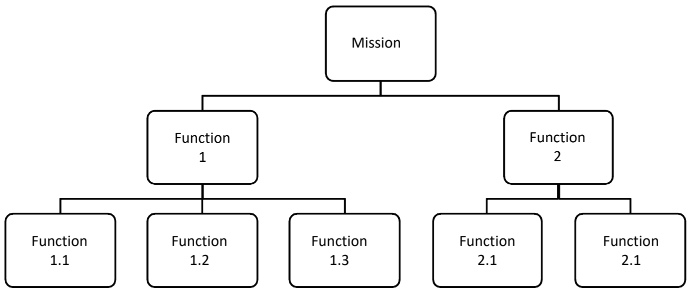

Figure 1-6. A logical (or functional) hierarchy of system functions.

1.1.7.2 Physical Hierarchy

In a physical sense, we saw earlier that a system can be considered to comprise operational (end) products and enabling products. The end products of systems are also normally described in a hierarchy—here we use a four-layer hierarchy. We describe a top-level entity known as the *system* that comprises a number of *subsystems* that comprise a number of *assemblies* that comprise a number of *components*. Although these terms are perhaps the most common, there are others in use. For example, Figure 1-7 illustrates what is perhaps the most complete physical hierarchy of system elements as defined by IEEE-STD-1220 [15], which also adds entities such as products, subassemblies, subcomponents, and parts to our simple four-layer taxonomy.

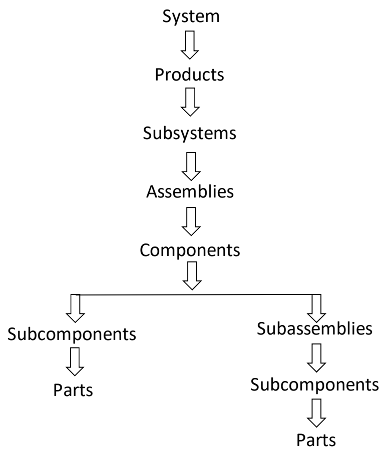

Figure 1-7. Hierarchy of elements of a system from IEEE-STD-1220.

The application of these terms to specific situations and examples also depends very much on the context of the situation and where within the overall project the system is being considered. For example, at the highest-level of an aircraft, we would consider that the aircraft system contains, among others, the engine subsystem. The engine subsystem may consist of assemblies such as fuel tanks, pumps and lines, turbines, compressors, gear boxes, and hydraulic pumps. From the viewpoint of an engine manufacturer, however, the engine is commonly considered to be the system, comprising fuel, power plant, and hydraulic subsystems, and so on.

The difficulty with considering the engine subsystem as a system in its own right is that an implicit part of the definition of a system is that it must be able to stand alone in its own right. By that definition, an engine is not able to be considered a system—it is only useful as an element of a system (that is, as a subsystem).

That is not a common view, however. For example, ISO/IEC/IEEE 15288 [16] considers an SOI to comprise a combination of interacting system elements, some of which may be systems in their own right—as illustrated in Figure 1-8. We will see later that the precise bounding of the SOI is a very important part of the system design.

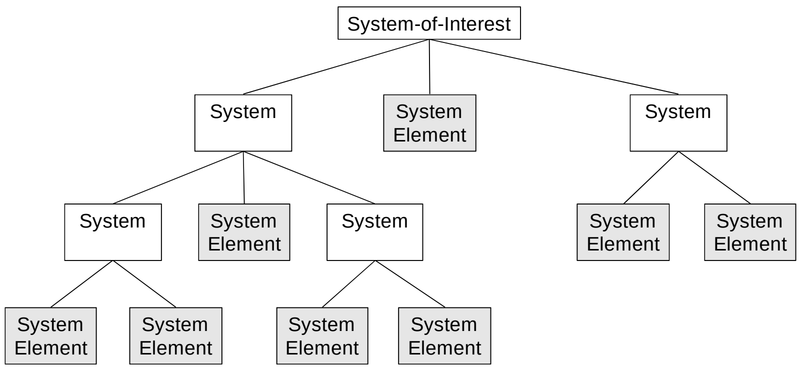

Figure 1-8. Hierarchy of elements of an SOI (after ISO/IEC/IEEE 15288).

When the SOI consists only of system elements that are systems in their own right—the system-of-interest is commonly called a *system-of-systems (SoS)* [17]. Figure 1-9 illustrates how the hierarchy of Figure 1-8 can be delineated into groupings of systems and SoS. However, to have a useful view, we need a better understanding of the ways in which system elements can be grouped.

1.1.8 Collections of Systems

If a system comprises a set of interacting elements, there are a number of types of ways in which those elements can be collected. Unfortunately, we are not always very careful about the language we use to describe those connections. First, we are not always careful to make the distinction between a system and a subsystem. The difference matters because we simply cannot refer to any grouping of elements as a system. For example, we saw earlier that an aircraft system may contain, among other elements, the engine, which may consist of assemblies such as fuel tanks, pumps and lines, turbines, compressors, gear boxes, and hydraulic pumps. Commonly, the engine manufacturer will refer to the engine as the *engine system*. If we continue such a use of the term to the lowest level of components, a nut and bolt can be considered a *fastening system* because those two elements interact for that purpose. Consequently, if we use the term in that manner, everything is a "system"—which is not very useful.

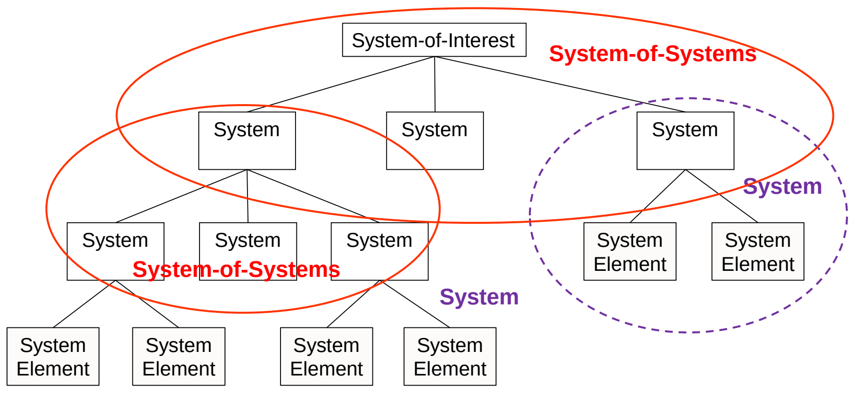

Figure 1-9. Hierarchy of elements of an SOI (after ISO/IEC/IEEE 15288).

So, as we use the term in systems engineering, an implicit part of the definition of a system is that it must be able to stand alone in its own right—that is, it is managerially and operationally independent. An engine is therefore not a system—it is only useful as an element of a system (that is, as a subsystem). We can therefore be more careful about the use of the terms system and subsystem by noting:

- A *system* is a set of tightly coupled, co-dependent elements (subsystems) optimized at the system level for the permanent purpose of the system. Systems are managerially and operationally independent and will no doubt have independent life cycles (they will almost certainly have been procured separately).
- A *subsystem* is a set of tightly coupled, co-dependent elements (components) optimized at the system level for the permanent purpose of the system. Subsystems are not independent and only exist to serve the system—they will have been designed by the system designer to be part of the system. Further, since the system is to be optimized, the subsystems are invariably sub-optimal.

Since a system element can be a system in its own right, there are then three types of ways in which systems can be grouped [18]:

- *Portfolio-of-systems (PoS)*. A PoS is a collection of systems managed jointly with a unified budget to fulfil one or more missions (that may not be related). Member systems interact to share (compete for) resources. For example, ACME Air may group all facilities, buildings and infrastructure into a PoS for ease of management—further, if they operate in three capital cities, they may have three facilities PoS, one in each city.

- *System-of-systems (SoS).* As we saw at the end of the previous section, an SoS is a collection of independent systems that interact for a common purpose, normally to produce an operational effect. The constituent systems of an SoS are loosely coupled and, since they are independent and can join or leave the SoS over time, have been optimized for their own purpose before joining the SoS. An individual system may be a member of more than one SoS.
- *Family-of-systems (FoS).* A collection of systems (also commonly called a called a product line) that are grouped because they are jointly designed, developed, and manufactured even though they may not interact in operations.

Having introduced the various manners of grouping, in this course we are particularly interested in three levels of combination: subsystem, system and SoS. As illustrated simplistically in Figure 1-10, the three groupings have similar architectures, in that each comprises elements that are interconnected.

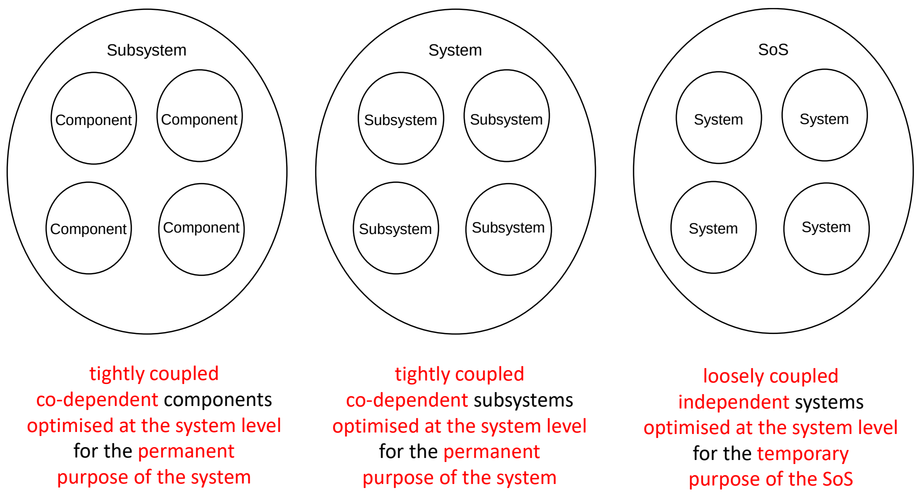

Figure 1-10. Elements of a subsystem, a system, and a system-of-systems.

However, as noted in the descriptions above, there are differences, despite a similar upper-level architecture:

- *SoS.* SoS elements are systems in their own right so that they are managerially independent and operationally independent and have been optimized for their own purpose before contributing to the purpose of the SoS. Further, since the constituent systems must be allowed to be optimized for their own purpose, the resultant SoS will invariably be sub-optimal.
- *System.* On the other hand, subsystems are not independent and only exist to serve the parent system—subsystems are therefore invariably

sub-optimal (from their perspective) since it is the system that is to be optimized, not the constituent subsystems.

- *Subsystem.* Components are not independent and only exist to serve the system as a good servant of the subsystem—again, it is the system that is to be optimized, not the constituent subsystems.

The attributes of subsystems, systems and SoS are summarized in Figure 1-11.

Attribute	Subsystem	System	SoS
Purpose	System	System	SoS
Element Type	Component	Subsystem	System
Integration	Tightly Coupled	Tightly Coupled	Loosely Coupled
Element Interaction	Co-dependent	Co-dependent	Independent
Period	Permanent	Permanent	Temporary
Optimisation Level	System	System	System

Figure 1-11. Attributes of a subsystem, a system, and an SoS.

An important observation from Figure 1-11 is the central position of system design—note that it is the system that is optimized at each level of subsystem, system and SoS. In this text, therefore, we focus on the system.

1.1.9 Problem Domain and Solution Domain

When introducing a system, we noted that a system can be considered to be the solution to a problem. As well as viewing the system descriptions in logical and physical terms, therefore, it is common to consider the activities being undertaken throughout the life of the system to be in either the *problem domain* (*problem space*) where we use predominantly logical descriptions, or the *solution domain* (*solution space*) where we use predominantly physical descriptions.

Activities in the problem domain (including production of the logical architecture) are generally considered to be the responsibility of the customer (the business owner); activities in the solution domain (including the physical architecture) are generally considered to be the responsibility of the organisation implementing the system (the developer).

1.2 SYSTEM LIFE CYCLE

As with almost anything else, a system has a life—at some point in time it doesn't exist, it is brought into being, it is used, and then it is disposed of once it can no longer serve the purpose for which it was created. Throughout the life of a system there are a number of phases and activities, each of which builds on the results of the preceding phase or activity. The sum all these activities is called a *system life cycle*, which can be described using a model that represents the conceptualization of the business needs for the system, its realization, utilization, evolution, and ultimate disposal [19].

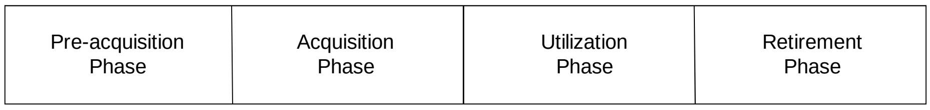

Figure 1-12. Phases of a generic system life cycle.

As shown in Figure 1-12, a generic system life cycle can be divided into four very broad phases:

- *Pre-acquisition Phase*. The life cycle begins in the Pre-acquisition Phase with an idea for a system being generated as a result of business planning. Early consideration of the possible options results in the confirmation of the early business needs for the system, which are elaborated by a business case that justifies expenditure of organizational resources on acquisition of the system. In some instances, the Pre-acquisition Phase may determine that it may not be feasible or cost-effective to proceed to acquisition (due to technology limitations or funding shortfalls, for example). In that sense then, the Pre-acquisition Phase is where organisations expend research and development funds to ensure that only feasible, cost-effective projects are taken forward to acquisition.
- *Acquisition Phase*. The business needs for the system provide the input for the Acquisition Phase which is focused on bringing the system into being and into service. This would normally involve defining the system in terms of business needs and requirements, stakeholder needs and requirements, and system requirements and then engaging a contractor to develop/deliver the system.
- *Utilization Phase.* The system remains in service during the Utilization Phase until the business has no further need for it, or it no longer can meet the functions required of it by the organisation, or it is no longer cost-effective to keep it in service. During utilization, the system may undergo a number of modifications and upgrades to rectify performance shortfalls, to meet changing operational requirements or external environments to enable ongoing support for the system to be maintained, or to enhance current performance or reliability.
- *Retirement Phase*. Following operational use and system support, the system is eventually phased out and retired from service. The system life cycle concludes with the Retirement Phase. If the business need for the capability still exists in the organisation, the conclusion of one system life cycle marks the start of another and the process begins again.

1.2.1 Parties Involved

There are a number of parties involved throughout the system life cycle. The *customer* organization is managed by *enterprise management* who set the direction for the organisation and for *business management* who are responsible for managing the activities conducted by the *operations* element of the organisation (undertaken by the *operators*—sometimes called *users*).

The systems employed within the organisation are acquired by the *acquisition* element (also called the *acquirer* [20], or *tasking activity* [21]) of the organisation under the auspices of a *project* (managed by a *project manager*). Project managers are supported by a number of related disciplines including *systems engineering*, *requirements engineering*, *specialist engineering disciplines*, *quality assurance*, and *integrated logistic support*. Operators are supported in their operation of the system by the *support* element of the organisation, which supports, sustains, and maintains the system throughout its life. In addition to the operational, acquisition, and support staff, there are many others within the customer organization who have a stake in the successful implementation of the project. These *stakeholders* can include representatives from the management, financial, operations, supply, maintenance, and facilities areas of the organisation.

The system is obtained from a *supplier* [22] (also called the *performing activity* [23]) who may deliver the system off-the-shelf or may develop it, in which case they are often called the *developer* [24]. The supplier (developer) may be an internal part of the customer (acquirer) organisation. If the development of the system is undertaken in-house, the acquisition element of the organisation (the acquirer) will engage with the development element (the developer) to develop the system. It is increasingly common these days for the supply or development to be undertaken by an outside organisation called a *contractor*, which is the entity responsible for supplying (perhaps by designing and developing) the system to meet the customer requirements. The relationship between the customer and the contractor varies with each project but, for each project, is defined by the terms and conditions of the *contract* between the two parties. In many cases the contractor is not able to perform all of the work required and devolves packages of work to a number of *subcontractors*. The terms and conditions relating to this work are described in the relevant *subcontract*.

Responsibility for the various phases of the system life cycle is spread across the enterprise (or organisation) within which the eventual system will operate. Figure 1-13 shows that the initial Pre-acquisition Phase is the responsibility of enterprise management, who conduct business planning and establish the business case for the projects required to support an organisation. A project is then established with a project charter providing authority to a project manager to expend organizational resources on the acquisition of the system. Systems engineering is an important discipline which is responsible to the project manager to perform the technical management of the project throughout acquisition and utilization. Once acquisition is complete, and the system is in-service, it is operated by the users and supported by the support element. Note that all parties are involved at all stages in the life cycle, with the roles and responsibilities of each party shifting in emphasis between stages.

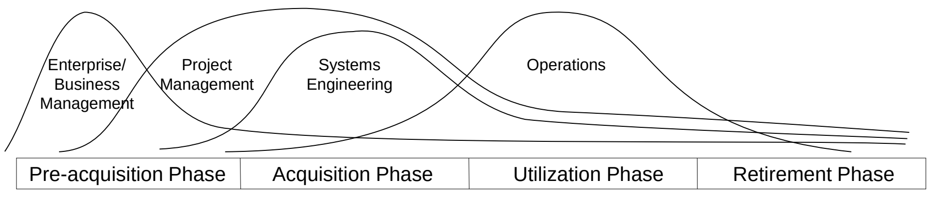

Figure 1-13. Responsibility for the phases of a generic system life cycle.

1.3 ACQUISITION AND UTILIZATION PHASES

As illustrated in Figure 1-13, systems engineering is predominantly related to the Acquisition Phase and, to a lesser extent, the Utilization Phase of the system life cycle. For these two major phases, we use the life-cycle activities in Figure 1-14, which are based on those defined by Blanchard and Fabrycky [25]. In the Acquisition Phase, the activities are *Conceptual Design*, *Preliminary Design*, *Detailed Design and Development*, and *Construction and/or Production*. In the Utilization Phase, the activities are *Operational Use* and *System Support*, which are undertaken in parallel. While there is no standard taxonomy, we choose these activities as a framework here because they are generally accepted in the systems engineering community over the past decade or so, and for the following reasons:

- These activities emphasize that a system begins with the perceived business needs and finishes with retirement and, ultimately, disposal of the system—the so-called cradle-to-grave approach.
- There is a clear delineation between the Acquisition and Utilization (in-service) Phases of a system, which recognizes that there is a period of transition required as the system is transferred from the acquirers to the operators/users.
- The activities show sufficient detail in the early stages of the Acquisition Phase (particularly in Conceptual Design and Preliminary Design) where the application of systems engineering methodologies and practices have the potential to make the most significant contribution.
- Importantly, within the Acquisition Phase, the activities also differentiate clearly between the problem domain which contains the logical description of the system (the product of Conceptual Design) and the solution domain which contains the physical description of the system (the products of Preliminary Design and Detailed Design and Development).

- Additionally, the separation of the early system design into Conceptual (what and why) and Preliminary (how) Design is very important since the responsibility in most programs transitions from the customer for Conceptual Design to the contractor for Preliminary Design.

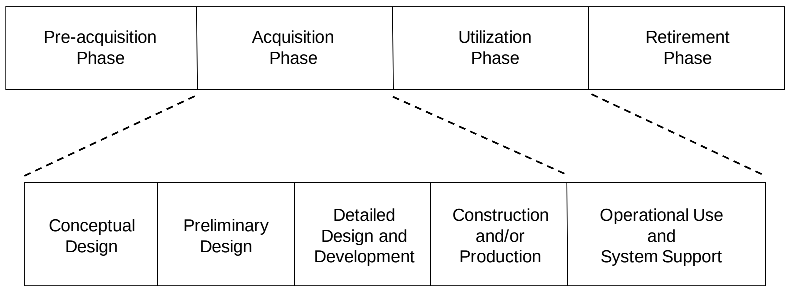

Figure 1-14. Activities in the Acquisition and Utilization Phases of the system life cycle (after Blanchard and Fabrycky [26]).

The significance of focusing on the system life cycle is that decisions made early in Conceptual Design are informed by the activities proposed to be conducted later in the Acquisition Phase and in the Utilization Phase. For example, the design of an aircraft airframe must take into account the maintenance and operation of that airframe during the Utilization Phase—it would be pointless to design the best airframe in the world if it did not have the necessary access points to allow maintenance personnel to service it, nor operators to operate it in the intended environment. We consider these issues in more detail in Section 1.5.3.

1.3.1 Acquisition Phase

Figure 1-14 shows that the Acquisition Phase comprises the four main activities of Conceptual Design, Preliminary Design, Detailed Design and Development, and Construction and/or Production. Each of these activities is described in more detail in the following sections, which outline the major tasks undertaken and the main artefacts produced in each (see Figure 1-15 for an overview).

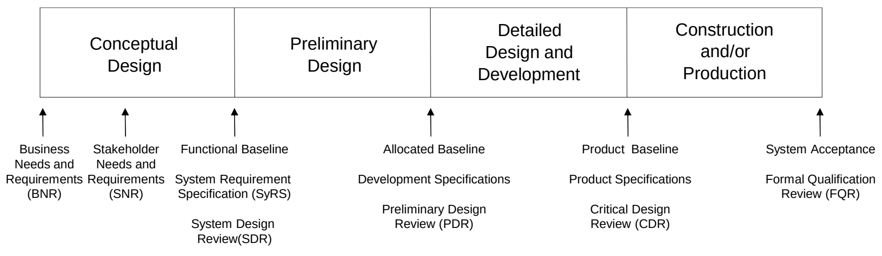
Conceptual Design
Preliminary Design
Detailed Design and Development
Construction and/or Production
Business Needs and Requirements (BNR)
Stakeholder Needs and Requirements (SNR)
Functional Baseline
System Requirement Specification (SyRS)
System Design Review(SDR)
Allocated Baseline
Development Specifications
Preliminary Design Review (PDR)
Product Baseline
Product Specifications
Critical Design Review (CDR)
System Acceptance
Formal Qualification Review (FQR)

Figure 1-15. Acquisition Phase activities and the major artefacts and reviews associated with each.

1.3.1.1 Conceptual Design

Conceptual Design is aimed at producing a set of clearly defined requirements, at the system level, and in logical terms. Although clearly defining the requirements of the system would seem a logical (and essential) first step, it is often poorly done and is commonly the direct cause of problems later in the development process. Business managers and stakeholders sometimes prefer to describe their requirements in loose and ambiguous terms to protect themselves from changes in their needs and their business environment. The Conceptual Design process aims to avoid this ambiguity by providing a formal process by which the *Business Needs and Requirements (BNR)* are articulated and confirmed by business management, and then elaborated by stakeholders at the business operations level into a set of *Stakeholder Needs and Requirements (SNR)*, which are further elaborated by requirements engineers into a set of system requirements in the *System Requirement Specification (SyRS)*. There may be one SyRS for the entire capability system, but it is more likely that there is one SyRS for each of the constituent elements of capability—the major materiel system, personnel, support, training, facilities, and so on. As noted earlier, each of these constituent capability elements may be developed independently, perhaps through separate contracts.

The SyRS is the key element of what is called the *Functional Baseline (FBL)*, which describes the whats and whys of the system. The FBL represents a system-level logical architecture that meets the business and stakeholder needs and requirements.

Conceptual Design ends with the *System Design Review (SDR)*, which finalizes the initial FBL. The SDR provides a formalized check of the logical design; communicates that design to the major stakeholders; confirms external interface and interoperability issues; confirms the BNR, SNR and the SyRS; and provides a formal record of design decisions and design acceptance.

1.3.1.2 Preliminary Design

The aim of Preliminary Design is to convert the FBL into an upper-level physical definition of the system configuration or architecture (the hows of the system). Preliminary Design is therefore the stage where logical design is translated into physical design; where focus shifts from the problem domain into the solution domain. The result of Preliminary Design is a subsystem-level design known as the *Allocated Baseline (ABL)* in which the logical groupings defined in the FBL have been defined in more detail, and then re-grouped and allocated to subsystem-level physical groupings (called *configuration items (CI)*), which combine to form the upper-level physical design of the system. At the centre of the ABL are a series of *Development Specifications*, which contain the subsystem-level requirements grouped by CI.

The ABL is so-called because the requirements that are logically grouped in the FBL are ‘allocated’ at this next baseline into physical groupings. The

ABL therefore represents a subsystem-level architecture (couched in physical terms) that meets the requirements of the system-level architecture (couched in logical terms) contained in the FBL.

The ABL is formalized at the *Preliminary Design Review (PDR)*. The PDR ensures the adequacy of the Preliminary Design effort prior to focusing on detailed design. PDR is designed to assess the technical adequacy of the proposed solution in terms of technical risk and the likely satisfaction of the FBL. PDR also investigates the identification of CI interfaces and the compatibility of each of the CIs.

1.3.1.3 Detailed Design and Development

The ABL developed during Preliminary Design is used in the Detailed Design and Development process to complete development of the individual subsystems, assemblies, and components in the system. Prototyping may occur and the system design is confirmed by test and evaluation. The result of the Detailed Design and Development process is the initial establishment of the *Product Baseline (PBL)* as the system is now defined by the numerous products (subsystems, assemblies, and components) making up the total system (as well as the requisite materials and processes for manufacturing and construction). The definition of the system at this stage should be sufficiently detailed to support the commencement of the Construction and/or Production activities.

The PBL is established at the *Critical Design Review (CDR)*. The CDR is the final design review resulting in the official acceptance of the design and the subsequent commencement of Construction and/or Production activities; CDR evaluates the detailed design; determines readiness for production/construction; and ensures design compatibility, including a detailed understanding of all external and internal interfaces.

1.3.1.4 Construction and/or Production

The final activity within the Acquisition Phase is Construction and/or Production. System components are produced in accordance with detailed design specifications in the PBL and the system is ultimately constructed in its final form. Formal test and evaluation activities (acceptance tests) will be conducted to ensure that the final system configuration meets the requirements in the SyRS.

Construction and/or Production, and the Acquisition Phase, ends with the *Formal Qualification Review (FQR)*, which provides the basis upon which the customer accepts the system from the contractor. The FQR is informed by the results of acceptance test and evaluation (AT&E).

1.3.2 Utilization Phase

On acceptance from the supplier, the system moves into the Utilization Phase. The major activities during this phase are Operational Use and System Support. Systems engineering activities will invariably continue during the Utilization Phase to support any modification activity that may be required. Modifications may be necessary to rectify performance shortfalls, to meet changing operational requirements or external environments to enable ongoing support for the system to be maintained, or to enhance current performance or reliability.

1.3.3 Retirement Phase

The system life cycle ends with retirement of the system in the Retirement Phase, which may well overlap with the introduction into service of the replacement system. Functions associated with phase-out and disposal include transportation and handling, decomposition, and processing of the retiring system. A Retirement Concept should be developed during the early stages of the Acquisition Phase. If considered early, disposal and phase-out issues will form some of the criteria against which the system is designed ('design for disposability'). It is important, however, that system designers focus on retirement, rather than the more limiting issues of disposal—planning for disposal is important, but a system may retire from a number of life cycles before it is ultimately disposed of at the end of life.

1.4 SYSTEMS ENGINEERING AND DEVELOPMENT APPROACHES

It should be noted that the generic system life cycle illustrated in Figure 1-12 shows the phases and activities in sequence and is not intended to represent any particular development or acquisition model. Throughout the early chapters of this book we describe systems engineering without discussing in great detail the development and acquisition context within which it might be undertaken. We have so far presented that the life-cycle activities are undertaken sequentially because it is the best way to explain the activities and artefacts of systems engineering. In doing so, we have assumed what is generally referred to as a *linear sequential* or *waterfall approach* to system development. There are, however, a number of other development approaches to implementing the activities of the system life cycle in Figure 1-12—such as the *incremental*, *spiral*, or *evolutionary acquisition* models [27], each of which has strengths and weaknesses depending on the nature of the system under development. The selection of a suitable development approach is a critical planning activity early in a system life cycle.

However, for simplicity in the early chapters, the waterfall approach system development is assumed in order to provide a logical, sequential flow of activities and deliverables that support teaching and explaining systems engineering. Additionally, the waterfall approach is generally considered to be the basic building block upon which the alternative approaches such as incremental, evolutionary, and spiral development are built [28,29]. A solid understanding of waterfall development is therefore useful.

We discuss these issues in much more detail in Chapter 11 in which we consider systems engineering as part of various acquisition and development approaches.

1.5 WHAT IS SYSTEMS ENGINEERING?

There are many definitions of systems engineering, each of which is subtly different because it tends to reflect the particular focus of its source. The following are some of the definitions from relevant standards and documents.

'Systems engineering is the management function which controls the total system development effort for the purpose of achieving an optimum balance of all system elements. It is a process which transforms an operational need into a description of system parameters and integrates those parameters to optimize the overall system effectiveness.' [30]

'An interdisciplinary collaborative approach to derive, evolve, and verify a life cycle balanced system solution which satisfies customer expectations and meets public acceptability.' [31]

'An interdisciplinary approach encompassing the entire technical effort to evolve and verify an integrated and life cycle balanced set of system, people, product, and process solutions that satisfy customer needs. Systems engineering encompasses: the technical efforts related to the development, manufacturing, verification, deployment, operations, support, disposal of, and user training for, system products and processes; the definition and management of the system configuration; the translation of the system definition into work breakdown structures; and development of information for management decision making.' [32]

'Systems engineering is the selective application of scientific and engineering efforts to: transform an operational need into a description of the system configuration which best satisfies the operational need according to the measures of effectiveness; integrate related technical parameters and ensure compatibility of all physical, functional, and technical program interfaces in a manner which optimizes the total system definition and design; and integrate the efforts of all engineering disciplines and specialties into the total engineering effort.' [33]

'Systems engineering is an interdisciplinary, comprehensive approach to solving complex system problems and satisfying stakeholder requirements.'[34]

Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the

> *complete problem: operations, cost and schedule, performance, training and support, test, manufacturing, and disposal. SE considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs. [35]*

More recently, the INCOSE Fellows have defined systems engineering as:

> *… a transdisciplinary and integrative approach to enable the successful realization, use and retirement of engineered systems, using systems principles and concepts, and scientific, technological and management methods. [36]*

Although each of these definitions has a slightly different focus, a number of common themes are evident and are described in the following sections.

1.5.1 Top-down Approach

Traditional engineering design methods are based on a bottom-up approach in which known components are combined into assemblies and then into the subsystems from which the system is then constructed. The system is then tested for the desired properties and the design is modified in an iterative manner until the system meets the desired needs. This approach is valid and extremely useful for relatively straightforward problems that are well defined. Unfortunately, complicated problems cannot be solved with the bottom-up approach.

Systems engineering begins by addressing the system as a whole, which facilitates an understanding of the system, its environment and its interfaces. Once system-level requirements are understood, the system is then broken down into subsystems and the subsystems further broken down into assemblies, and then into components until a complete understanding is achieved of the system from top to bottom. This top-down approach is a very important element of managing the development of complicated systems. By viewing the system as a whole initially and then progressively breaking the system into smaller elements, the interaction between the components can be understood more thoroughly, which assists in identifying and designing the necessary interfaces between components (internal interfaces) and between this and other systems (external interfaces). For example, Figure 1-16 illustrates the ANSI/EIA-632 approach to top-down development from a product perspective.

It must be recognized, however, that while design is conducted top-down, the system is implemented using a bottom-up approach—as illustrated in Figure 1-17 for a simple four-tier system. That is, one of the aims of system engineering is to provide a rigorous, reproducible process by which the complex system can be broken down into a series of simple components that can then be designed and built using the traditional bottom-up engineering approach. Importantly, therefore, the second principal facet of systems engineering is to provide a process by which the components, assemblies, and subsystems can be integrated to achieve the desired system purpose.

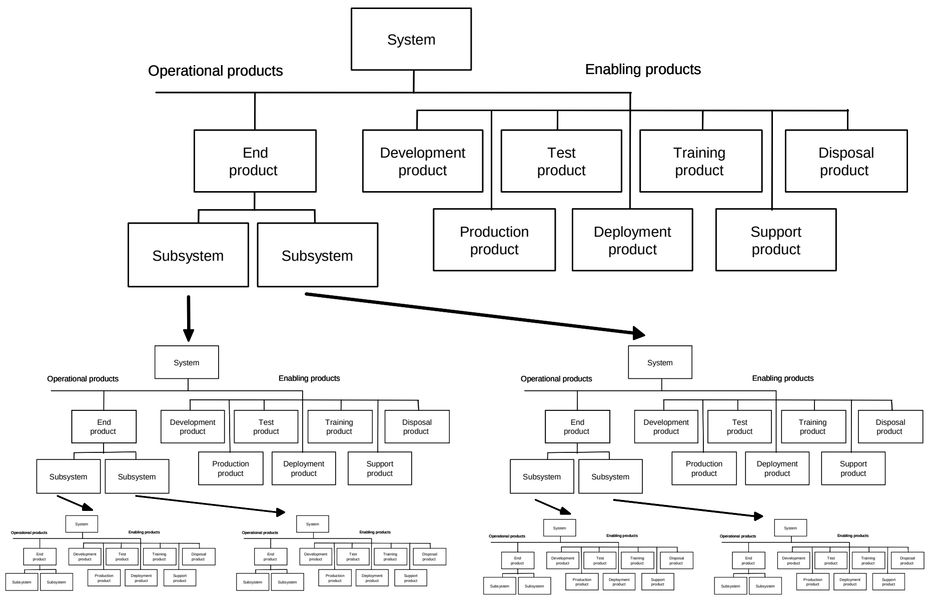

Figure 1-16. ANSI/EIA-632 building block concept for top-down development [37].

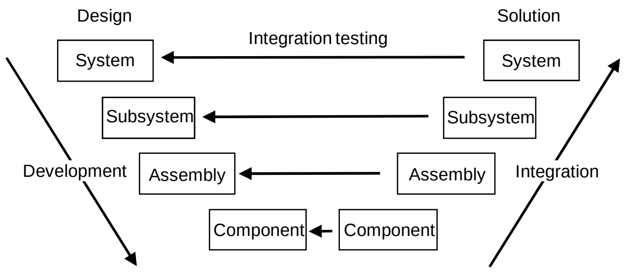

Figure 1-17. Bottom-up integration of a system.

Note that, as discussed in Section 1.1.7, the terms *system*, *subsystem*, *assemblies*, and *component* are relative. Each system comprises subsystems that consist of assemblies that consist of components. Each subsystem, however, can be considered in some areas to be a system in its own right, which has subsystems, assemblies, and components, and so on. IEEE-STD-1220 calls this the *system paradigm*. Consequently, while the building-block concept for top-down development is very useful, it is often a source of confusion among novices if the associated terms are allowed to be used in a relative manner. Here, as outlined earlier, we avoid that confusion by maintaining distinctions among the terms SoS, system and subsystem.

1.5.2 Requirements Engineering

The development of a complete and accurate definition of system requirements is fundamental to project success and is a primary focus of the early systems engineering effort (recognising, of course, that a complete description is not always possible). The life cycle of a system begins with business needs, which are ultimately translated into a large number of statements of requirement that form the basis for the logical design and subsequently elaborated further to form the physical architecture. These transitions must be managed by a rigorous process, called *requirements engineering*, which is aimed at ensuring that all relevant requirements are included (and all irrelevant requirements excluded). The establishment of correct requirements is fundamental to the success of the subsequent design activities. Poor requirements cannot be rectified by good design, so it invariably follows that rigorous development of requirements is essential for the acquisition to be successful.

Once requirements have been collected, the systems engineering process then focuses on the derivation and decomposition of these requirements from the system level right down to the lowest constituent component (sometimes referred to as *requirements flowdown*). This process involves elicitation, analysis, definition, validation, and management of requirements. Requirements engineering ensures that a rigorous approach is taken to the collection of a complete set of unambiguous requirements from the stakeholders.

Requirements traceability is also an essential element of effective management of complex projects. Through traceability, design decisions can be traced from any given system-level requirement down to a detailed design decision (called *forward traceability*). Similarly, any individual design decision must be able to be justified by being associated with at least one higher-level requirement (called *backward traceability*). This traceability is important since the customer must be assured that all requirements can be traced forward and can be accounted for in the design at any stage. Further, any aspect of the design that cannot be traced back to a higher-level requirement is likely to represent unnecessary work for which the customer is most probably paying a premium. Traceability also supports the configuration control (change management) process, especially the investigation of the impact of the change.

Support for requirements traceability is a feature of the top-down approach that provides a mechanism by which it can be guaranteed that requirements can be satisfied at any stage. A bottom-up approach cannot provide the same guarantee.

Chapter 2 provides more detail on the body of knowledge of requirements engineering.

1.5.3 Focus on Life Cycle

Systems engineering is focused on the entire system life cycle and takes this life cycle into consideration during decision-making processes. In the past it has been too common to consider design options only in the light of the issues associated with the Acquisition Phase and to pay little attention to through-life support aspects. It is proper for project managers and their teams to focus on the Acquisition Phase of the project and on the development of a system that meets the stakeholder requirements while minimizing cost and schedule. However, a lack of consideration of whole-of-life considerations can often lead to larger-than-expected costs in the Utilization Phase to be met from budgets that are insufficient to keep systems in service. A life-cycle focus requires a focus on the capability system throughout its life cycle, not on the product throughout its acquisition.

Other examples of Utilization Phase requirements that impact on equipment design or selection during the Acquisition Phase include *reliability* and *availability*. Reliability normally refers to the ability of the equipment to operate without failure for a given period of time. Availability is a measure of the degree to which a system is in an operable condition when required at some random point in time. Again, a superior design is pointless unless the system can meet specified minimum levels of reliability and availability.

Economic factors provide arguably the most compelling evidence to support the focus on life cycle as opposed to an emphasis on the end product itself. In short, a life-cycle focus can save money in the long term. Experience has shown that a large proportion of total life-cycle cost for a given system stems from decisions made early in the Acquisition Phase of the project. Some 60% of errors in system development originate in the requirements-analysis process [38]. To that end, the Acquisition Phase presents the maximum opportunity to reduce the total life-cycle cost of a system.

Example 1.4: Life-cycle Focus

As a simple example to demonstrate the concept (and problems) of a product focus as opposed to a life-cycle focus, ACME Air must take into account much more than the purchase price of the new aircraft. If a low purchase price has been achieved by a design that incorporates poor-quality components that require regular maintenance, the running costs may well be much higher than competing models. Clearly, the choice of an aircraft must focus on total cost of ownership of the aircraft which, in addition to the purchase price, must take into account the system life-cycle aspects of operation, maintenance, and support.

1.5.4 System Optimization and Balance

A system architecture must represent a balance between the large number of requirements and constraints that, as well as the technical considerations, cover a wide range of factors such as environmental, economic human factors, moral, ethical, social, cultural, psychological, and so on. A simple, but essential, example is the balance between just two design factors: cost and performance—if either is allowed to dominate, it will generally be at the expense of the other.

A balance in system design must also be struck across the life cycle. Metrics such as cost-effectiveness must be measured across all phases, not just acquisition. It is often the case that savings made in acquiring the system are then countered by significant maintenance and repair costs in service.

Systems engineering performs a very important role in system design in ensuring that there is a balance among the various system components. It is essential that optimization and balance are managed at the systems level. As we discuss in Chapter 4, it does not necessarily follow that the combination of optimized subsystems leads to an optimized system. It is not normally useful, therefore to allow the designers of subsystems to optimize their part of the system in isolation of system-level considerations. Consider the impact of incorporating an F1 engine in a small family saloon car—the engine may be optimized for performance but it will destroy the remainder of the drive train that has been designed for a much less powerful engine. Additionally, the F1 engine is capable of propelling the family car far faster than is safe given its suspension and brakes, and much faster than is allowed by the legal speed limits.

It follows, therefore, that a number of subsystems may need to be suboptimal (or at least constrained in some manner) to allow their combination (the system) to be optimal. A further advantage, therefore, of the top-down approach in systems engineering is that system optimization and balance can be achieved as a by-product of the design process, something that cannot be guaranteed in a bottom-up design method.

1.5.5 Integration of Disciplines and Specializations

Systems engineering aims to manage and integrate the efforts of a multitude of technical disciplines and specialties to ensure that all stakeholder requirements are adequately addressed. Rarely is it possible for a complicated system to be designed by a single discipline. Consider our aircraft example. While aeronautical engineers may be considered to have a major role, the design, development and production of a modern aircraft system requires a wide variety of other engineering disciplines including electrical/electronics, safety and assurance, EMI/EMC, production, metallurgical, and corrosion engineers. Of course, in system terms, other engineering disciplines are required for testing and for logistics and maintenance support as well as the design and building of facilities such as runways, hangars, refuelling facilities, embarkation and disembarkation facilities, and so on. Other non-engineering disciplines are involved in such aspects as marketing, finance, accounting, legal, and environmental. In short, there could be hundreds, even thousands, of engineers and members of other disciplines involved in the delivery of a single aircraft system. Commonly, complex systems can involve millions of hours of work by thousands of people from a wide range of disciplines and backgrounds spread across a dozens of countries.

The aim of systems engineering is to define the tasks that can be completed by these disparate disciplines and specialties, and then to provide the management to integrate their efforts to produce a system that meets the users' requirements. In modern system developments, this function is all the more important because of the complexity of large projects and their contracting mechanisms, and the geographic dispersion of contractor and subcontractor personnel across the country and around the world.

1.5.6 Management

While systems engineering clearly has a technical role and provides essential methodologies for systems development, it is not limited simply to technical issues and is not simply another engineering process to be adopted. Systems engineering has both a management and a technical role. Project management is responsible for ensuring that the system is delivered on-time and within-budget, and meets the expectations of customers. The trade-offs and compromises implicit in those functions are informed by the products of systems engineering. Additionally, the scope of the project is defined by the work breakdown structure, which is the result of requirements engineering. Systems engineering, requirements engineering, and project management are therefore inextricably linked. These issues are discussed in more detail in Chapter 10.

1.6 SYSTEMS ENGINEERING RELEVANCE

Systems engineering principles and processes are applicable (albeit to varying degrees) to a wide range of projects. For example, ANSI/EIA-632 [39] states that the standard itself is intended to be applicable to:

> *'the engineering or the reengineering of:*
>
> *a) commercial or non-commercial systems, or part thereof;*
>
> *b) any system, small or large, simple or complex, software-intensive or not, precedented or unprecedented;*
>
> *c) systems containing products made up of hardware, software, firmware, personnel, facilities, data, materials, services, techniques, or processes (or combinations thereof);*
>
> *d) a new system or a legacy system, or portions thereof.'*

It is difficult to imagine a project that does not fit into the above description, which can effectively be summarized to say that systems engineering is applicable to all projects. However, a project is defined as a unique activity, which means that the application of systems engineering to each project will also be unique to that project. Consequently, it is critical to understand the merits of systems engineering and apply them in a tailored manner, cognizant of the relative size, complexity, and risks associated with each undertaking. The most obvious application of systems engineering principles and methodologies is in projects that are large and complicated. However, smaller and less complex projects can also benefit from the tailored application of systems engineering.

At one end of the spectrum are large complex projects making use of leading-edge developmental technology. These projects typically involve large sums of money, long time frames, and significant risks. At the other end of the spectrum are small projects making use of extant techniques and existing technology. These projects typically involve short periods, low costs, and a minimum of risk. Clearly different levels of systems engineering are applicable to each of these types of projects and the aspects we consider here must be tailored for each individual project.

The failures of system developments are often attributed to misunderstandings, ambiguities, misinterpretations, errors, and omissions in what the contractor is attempting to deliver. The result is a system that fails to solve the customer’s problem leading to a breakdown in the relationship between contractor and customer. Systems engineering specifically targets these problems and therefore is relevant to all stakeholders in any system development.

Within a customer’s project office, systems engineering is particularly relevant to the technical personnel who are responsible for the application of systems engineering principles as part of the overall project management effort. In most customer organizations, systems engineering personnel report to the project manager. The systems engineer is in an excellent position to apply the tools of the systems engineering to assist the project manager in each of the ten project management knowledge areas (discussed in more detail in Chapter 10). Systems engineering is therefore an essential element of the project manager’s ability to acquire a quality system within budget, time, and scope constraints.

Contractors should also be focused on developing processes and best practices to support the delivery of superior products and services to their customers. Systems engineering offers tools, processes, and methodologies that support the consistent development of quality products and services.

1.7 SYSTEMS ENGINEERING BENEFITS

The principal causes of cost and schedule overrun on large-scale projects can be traced to overzealous advocacy, immature technology, lack of corporate roadmaps, requirements instability, ineffective acquisition strategy, unrealistic project baselines, project office personnel tenure and experience, and inadequate systems engineering [40]. In this book we focus on the latter (which, incidentally, is the only one of those causes directly within the project manager’s control) because there are a number of potential benefits from the successful implementation of systems engineering processes and methodologies.

The first and most visible benefit is the scope for saving money during all phases of the system life cycle—life-cycle cost (LCC) savings. While some may argue that the additional requirements imposed by systems engineering can increase costs, these increases are comparatively small and are generally felt in the very early design phases. If applied appropriately, systems engineering can ensure that the savings achieved far outweigh the cost of implementing appropriate procedures and methodologies. Experience indicates that an early emphasis on systems engineering can result in significant cost savings later in the construction and/or production, operational use and system support, and disposal phases of the life cycle [41].

Systems engineering should also assist in reducing the overall schedule associated with bringing the system into service. Systems engineering ensures that the user requirements are accurately reflected in the design of the system helping to minimize costly and time-consuming changes to requirements later in the life cycle. If changes are required, they can be incorporated early in the design and in a controlled manner. The rigorous consideration and evaluation of feasible design alternatives during the design phases of the project promote greater design maturity earlier.

System failures, cost overruns, and schedule problems are often the direct result of poor requirements-engineering practices—poor requirements cannot be rectified by good design. The systems engineering discipline aims to put in place a rigorous process of requirements engineering to produce well-defined requirements, adequate levels of traceability between the different levels of technical design documentation back to the original user requirements, and requirements which are both verifiable and consistent. This requirements-management process must achieve these results without pre-supposing a particular technical solution or placing unnecessary technical constraints on the solution.

Figure 1-18 provides a simplistic illustration of the impact of systems engineering on the system life cycle. Note that systems engineering has its greatest impact through the rigorous application of processes and methodologies during the early stages of the project where the ease of change and cost of modification is the lowest. In fact, the curve in Figure 1-18 could be relabelled as the ease with which changes can be made throughout the system life cycle. Consequently, systems engineering provides the ideal opportunity to have the greatest impact on a project at a time when changes are easiest to make. Additionally, as illustrated in Figure 1-19, the greatest impact of requirements engineering comes at a time when the cost of implementing changes is the lowest.

Systems engineering leads to a reduction in the technical risks associated with the product development. Risks are identified early and monitored throughout the process using a system of technical performance measures, and design reviews and audits. Design decisions can be traced back to the original user requirements and conflicting user requirements can be identified and clarified early, significantly reducing the risk of failure later in the project.

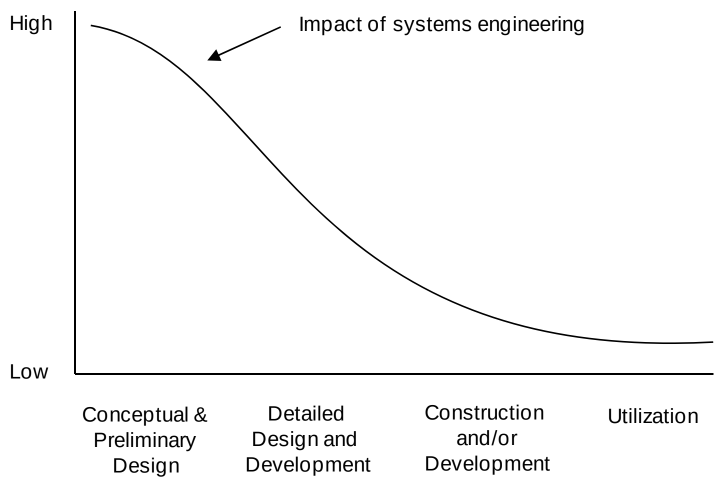

Figure 1-18. Impact of systems engineering on the system life cycle.

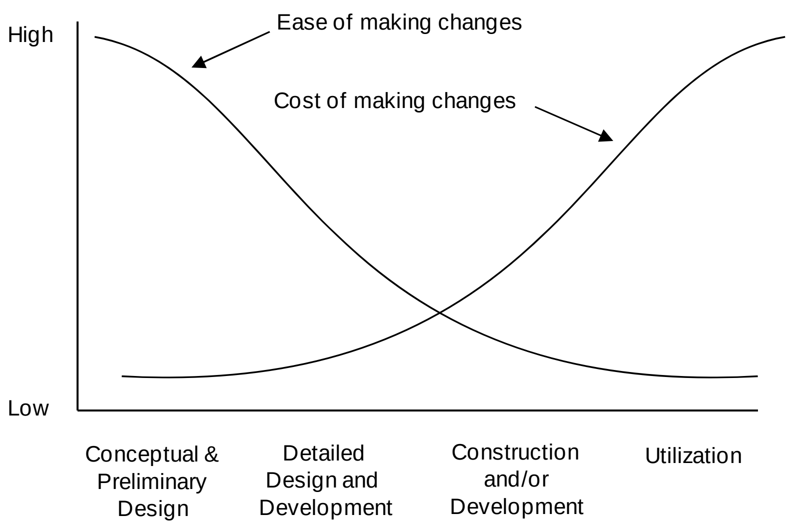

Figure 1-19. Ease and cost of making changes throughout the system life cycle.

Finally, and probably most importantly, the disciplined approach to systems engineering leads to a product that meets the original intended purpose more completely. This improved performance leads to a quality system where quality is measured by the ability of the system to meet the documented requirements.

1.8 ANALYSIS, SYNTHESIS, AND EVALUATION

All extant systems engineering standards and practices extol processes that are built around an iterative application of *analysis*, *synthesis*, and *evaluation* [42]. The iterative nature of the application is critical to systems engineering processes. Initially the process is applied at the systems level; it is then re-applied at the subsystem level, and then the assembly level, and so on until the entire development process is complete. During the earlier stages the customer is heavily involved; in the latter stages, the contractor is mainly responsible for the continuing effort, which is monitored by the customer.

Prior to detailing the individual activities within the systems engineering processes, it is worth considering the basic foundations of the analysis-synthesis-evaluation loop illustrated in Figure 1-20. This concept is not complex; it is simply a good, sound approach to problem solving that is applicable in any domain but is particularly fundamental to systems engineering.

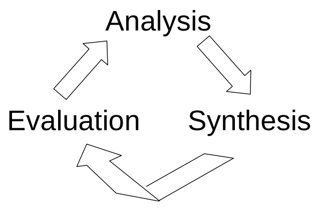

Figure 1-20. The analysis-synthesis-evaluation iteration.

1.8.1 Analysis

Analysis commences with the business needs for the system. During Conceptual Design, analysis investigates these needs and identifies the essential requirements of the system in order to meet the needs. Analysis at the system level aims to answers the *what*, *how well*, and the *why* questions relative to the system design. Analysis activities continue throughout the subsequent stages of the life cycle to help in defining the lower-level requirements associated with physical aspects (the *hows*) of the system design.

The requirements for the system should identify what functions the system should perform; the associated performance parameters such as speed, altitude, accuracy; constraints under which the system is to operate and be developed; interoperability requirements detailing other systems with which the system under development must operate; and interface requirements to describe the necessary outputs expected from the system and the inputs to the system. Depending on the particular design phase, these requirements may be grouped in accordance with some logical criteria and then allocated to a particular physical component of the system. That is, the component becomes responsible for the satisfaction of those requirements by performing the functions assigned to it. The allocation of requirements forms a description of the system elements and architecture and therefore assists in the process of synthesis or design (answering the *how* questions).

1.8.2 Synthesis

The analysis activity resolves what is required, as well as how well, and why. Synthesis, or design, now determines how. Synthesis is possibly the most widely recognized role of a professional engineer. Synthesis is the process where creativity and technology are combined to produce a design that best meets the stated system requirements. The term synthesis is more appropriate than design in the systems engineering context as it hints at the evolutionary nature of design and development.

In the early systems engineering processes, synthesis is limited to defining completely the logical design of the system and then considering all possible technical approaches using the results from the requirements-analysis effort. From this consideration, the best approach is selected and the process moves to the next level of detail. Later in the systems engineering processes, the selected design concept is synthesized further until the complete system design is finalized.

1.8.3 Evaluation

Evaluation is the process of investigating the trade-offs between requirements and design, considering the design alternatives, and making the necessary decisions. The process of evaluation continues throughout all stages of the systems engineering effort, ultimately determining whether the system satisfies the original needs and requirements. Trade-off analysis is one of the tools available to the system designer in performing evaluation of competing requirements or designs—a detailed treatment of trade-off analysis is provided in Chapter 4.

The outcome of the evaluation is the selection or confirmation of the desired approach to design. Discrepancies are also identified if applicable and may result in further analysis and synthesis. The basic analysis-synthesis-evaluation loop described in Figure 1-20 is applied iteratively throughout the system life cycle, as illustrated in Figure 1-21.

1.9 A SYSTEMS ENGINEERING FRAMEWORK

Discussions on systems engineering become complicated due to the broad mandate of the discipline, the wide range of types of systems, the complexity and interrelationship of the many systems engineering activities, and the relationships with other disciplines throughout the entire system life cycle.

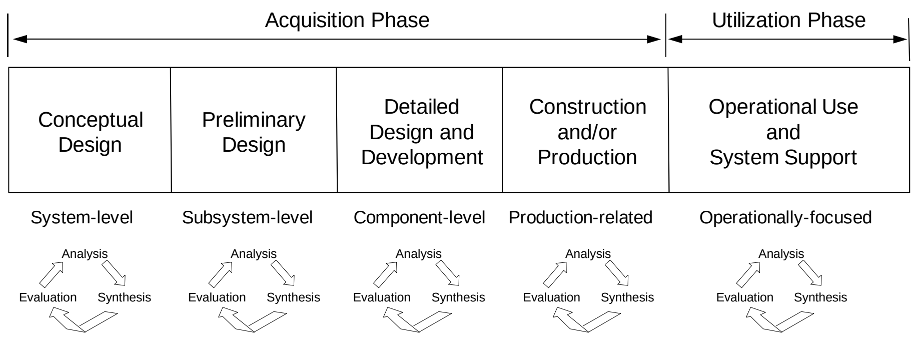

Figure 1-21. Iterative application of the analysis-synthesis-evaluation loop throughout acquisition and utilization.

The ability to understand a complex subject such as systems engineering is greatly enhanced by a solid framework within which concepts can be considered. Examples of such a structure include the Project Management Body of Knowledge (PMBOK) [43], which provides a clear framework within which to consider the many facets of project management, and the Software Engineering Body of Knowledge (SWEBOK) [44] that performs a similar role for the software engineering discipline. Without an equivalent framework, the broad scope of systems engineering soon becomes confusing given the complexity of its components and their many interrelationships. There are a number of excellent systems engineering standards available today that contribute to the elements of a suitable framework, but each standard contains complexity, terminology and detail that requires substantial interpretation. The entry level of many students, junior engineers, and project managers therefore does not allow the use of such standards as effective frameworks within which to examine systems engineering.

A systems engineering framework [45] (illustrated in Figure 1-22) has been synthesized by the authors through a thorough survey of existing systems engineering publications and standards, and through experience in teaching systems engineering at a range of levels. The main aim of the framework is to provide a simple construct within which the systems engineering discipline can be understood and implemented.

The framework illustrates the relationship of the three main elements of *systems engineering processes*, *systems engineering management*, and *systems engineering tools*, and places them in context with *related disciplines* such as traditional engineering disciplines, project management, integrated logistics support, and quality assurance. The terms systems engineering management and systems engineering processes are sometimes used interchangeably. Here we make a distinction between the two.

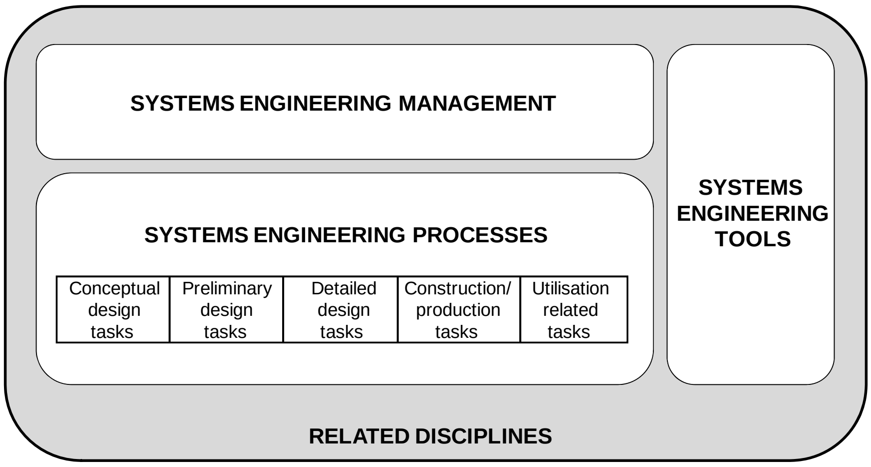
SYSTEMS ENGINEERING MANAGEMENT
SYSTEMS ENGINEERING PROCESSES
Conceptual design tasks
Preliminary design tasks
Detailed design tasks
Construction/ production tasks
Utilisation related tasks
SYSTEMS ENGINEERING TOOLS
RELATED DISCIPLINES

Figure 1-22. A framework for the consideration of systems engineering.

We present the systems engineering processes as the hows of systems engineering, the application of which forms the foundation of the systems engineering effort. Over the top of these processes sits the systems engineering management function, which is responsible for directing the systems engineering effort, monitoring and reporting that effort to the appropriate areas, and reviewing and auditing the effort at critical stages in the entire process. These two elements are supported by a range of tools and all elements exist in the context of related disciplines.

The systems engineering framework provides an excellent structure within which to examine and explain the complex discipline of systems engineering. Experience in undergraduate and graduate courses as well as commercial short courses [46] has shown that the framework provides an excellent means of communicating the complexities and interrelationships of the systems engineering discipline, particularly to those who do not have a significant amount of project experience. Students new to the discipline can successfully grasp the fundamental concepts of systems engineering within a relatively short timeframe.

1.9.1 Systems Engineering Processes

Systems engineering processes and tasks are divided into the life-cycle stages within which they typically occur. In this book we do not attempt to detail exhaustively all systems engineering processes. Instead, we concentrate on the intent and main aim of each phase of the system life cycle, and examine some of the likely techniques that may be used to arrive at that aim. We place particular emphasis on the Acquisition Phase of the life cycle, as it is the phase during which systems engineering has the ability to have the most impact on a system.

1.9.2 Systems Engineering Management

Systems engineering management is an overarching activity responsible for directing the systems engineering effort, monitoring and reporting that effort to the appropriate areas, and reviewing and auditing the effort at critical stages in the entire process—such management is the key to success of the entire systems engineering effort. In Chapter 8, we address the major systems engineering management elements of technical reviews and audits, system test and evaluation, technical risk management, configuration management, the use of specifications and standards, and systems engineering management planning.

1.9.3 Systems Engineering Tools

Many tools exist to assist systems engineering processes and management, including a range of techniques and methods and a number of standards. Here we describe the most popular tools without repeating information contained elsewhere in standards and other documents.

Throughout the book we present generic process tools such as requirements breakdown structures (RBS), functional flow block diagrams (FFBD), work breakdown structures (WBS), trade-off analyses, and prototyping and simulation as examples of tools that may be applied to the systems engineering process effort. We also describe the systems engineering management tools of standards and capability maturity models. In Chapter 9, the major systems engineering standards are reviewed and summarized including MIL-STD-499B [47], EIA/IS-632 [48], IEEE-STD-1220 [49], ANSI/EIA-632 [50], SAE1001 [51], and ISO/IEC/IEEE 15288 [52].

1.9.4 Related Disciplines

There are many disciplines (both technical and non-technical) related to systems engineering. Examples include project management, logistics management, quality assurance, requirements engineering, software engineering, hardware engineering, and interface engineering (or integration engineering).

The relationship between the related disciplines and the other facets of systems engineering depends very much upon the discipline in question. Some (such as project management) oversee the whole systems engineering discipline, while others (such as hardware and software engineering) sit between systems engineering management and the processes, and others (such as quality assurance) sit alongside the systems engineering effort. Chapter 8 discusses these disciplines and their relationship with systems engineering.

1.10 SUMMARY

In this chapter we introduced the nature of systems, the logical and physical descriptions of a system in the problem domain and the solution domain, the system life cycle and its constituent activities, and the parties involved in system acquisition. We also examined some definitions of systems engineering and extracted the common themes of top-down design, requirements engineering, focus on life cycle, system optimization and balance, integration of related disciplines and specialties, and management. We then considered the benefits of systems engineering and examined the analysis-synthesis-evaluation loop which is applicable from the basic engineering activities to the systems engineering activities right across the system life cycle.

In the next chapter we consider the body of knowledge of requirements engineering which is an essential aspect of system development. Subsequent chapters then consider each of the major systems engineering processes in some detail.

1.11 REVISION QUESTIONS

1. Briefly define a *system*.
2. What is meant by a *system of interest (SOI)*?
3. What is meant by the terms *operating environment* and *wider SOI (WSOI)*?
4. Briefly describe the classify system classifications: *open* or *closed*; *natural* or *human-made* or *human-modified*, *physical* or *conceptual*, *precedented* or *unprecedented*.
5. List the major resources that make up a *capability system* in the context of systems engineering.
6. A system can be described in both *logical* and *physical* terms. Briefly explain why the two descriptions must co-exist, what each description provides, and the relationship between the two descriptions.
7. Briefly outline how and why a system is described hierarchically (both logically and physically).
8. Briefly describe the three types of collection of *system*: *portfolio-of-systems (PoS)*, *system-of-systems (SoS)*, and *family-of-systems (FoS)*.
9. Briefly describe the differences among a *subsystem*, a *system* and a *system-of-systems*.
10. Briefly describe the difference between the *problem domain* and the *solution domain*.
11. Draw a diagram and briefly describe the four broad phases of a generic *system life cycle* (*Pre-acquisition Phase*, *Acquisition Phase*, *Utilization Phase*, and *Retirement Phase*).

12. Identify the major responsible parties (*enterprise management*, *business management, project management*, *systems engineering*, the *developer (contractor)*, and *users/support*) for the major phases of the life cycle.
13. Briefly describe the *Acquisition Phase* and the *Utilization Phase* of the system life cycle (as proposed by Blanchard and Fabrycky, and used in this text) and explain briefly the activities that occur within each, and the major artefacts and reviews associated with each.
14. Definitions of systems engineering abound, but all agree on a number of key themes of the discipline. List and briefly describe these key themes of systems engineering (top-down approach, requirements engineering, focus on life cycle, system optimization and balance, integration of disciplines and specializations, management).
15. Describe the benefits of using systems engineering discipline during the development of a system.
16. Describe diagrammatically the *analysis-synthesis-evaluation* process that is applied iteratively throughout the system life cycle, and explain each of the components.

ENDNOTES

[1] Further reading on systems engineering can be obtained from the following major sources:

Aslaksen E.W., *Designing Complex Systems: Foundations of Design in the Functional Domain*, Boca Raton: CRC Press, 2008.

Aslaksen, E. and R. Belcher, *Systems Engineering*, Upper Saddle River, N.J.: Prentice-Hall, 1992.

ANSI/EIA-632-1998, *EIA Standard—Processes for Engineering a System*, Arlington, V.A.: Electronic Industries Association, 1999.

Beam, W., *Systems Engineering: Architecture and Design*, New York: McGraw-Hill Book Co., 1990.

Blanchard, B.S. and J.E. Blyler, *Systems Engineering Management*, Hoboken, NJ: John Wiley & Sons, Inc, 2016.

Blanchard, B. and W. Fabrycky, *Systems Engineering and Analysis*, Upper Saddle River, N.J.: Prentice-Hall, 2011.

Boardman, J., *Systems Engineering: An Introduction*, Upper Saddle River, N.J.: Prentice-Hall, 1990.

Buede, D.M. and W.D. Miller, *The Engineering Design of Systems: Models and Methods*, New York: John Wiley & Sons, 2016.

Chestnut, H., *Systems Engineering Tools*, New York: John Wiley & Sons, 1965.

Chestnut, H., *Systems Engineering Methods*, NY: John Wiley & Sons, 1967.

Defense Systems Management College (DSMC), *Systems Engineering Management Guide*, Washington, D.C.: U.S. Government Printing Office, 1990.

Dickerson, C.E. and D.N. Marvis, *Architecture and Principles of Systems Engineering*, Baco Raton, CRC Press, 2010.

EIA/IS-632, *EIA Interim Standard—Systems Engineering*, Washington D.C.: Electronic Industries Association, 1994.

Eisner, H., *Essentials of Project and Systems Engineering Management*, New

York: John Wiley & Sons, 2008.

Faulconbridge R., and M. Ryan, *Engineering a System: Managing Complex Technical Projects*, Boston: Artech House, 2003.

Forrester, J., *Principles of Systems*, Cambridge, M.A.: The MIT Press, 1968.

Grady, J.O., *Systems Integration*, Boca Raton, F.L.: CRC Press, 1994.

Grady, J.O., *System Engineering Planning and Enterprise Identity*, Boca Raton, F.L.: CRC Press, 2010.

Haberfellner, R., O. de Weck, et al, *Systems Engineering: Fundamentals and Applications*, Birkhauser, 2019.

Hall, A., *A Methodology for Systems Engineering*, Princeton, N.J.: D. Van Nostrand Co. Ltd., 1962.

Hitchins, D., *Putting Systems to Work*, Chichester, England: John Wiley & Sons, 1992.

Hitchins, D., *Advanced Systems Thinking, Engineering, and Management*, Norwood, M.A.: Artech House, 2003.

Hitchins, D., *Systems Engineering: A 21st Century Systems Methodology*, Chichester, England: John Wiley & Sons, 2007.

Hubka, V. and W. Eder, *Theory of Technical Systems*, Berlin: Springer-Verlag, 1988.

Hunger, J., *Engineering the System Solution: A Practical Guide to Developing Systems*, Upper Saddle River, N.J.: Prentice-Hall, 1995.

IEEE-STD-1220-2005, *IEEE Standard for Application and Management of the Systems Engineering Process*, New York: IEEE Computer Society, 2005.

INCOSE, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, Walden D.D, et al, (eds), INCOSE-TP-2003-002-04 2015, Wiley, 2015.

INCOSE, *Journal of the International Council on Systems Engineering*, Seattle, W.A.: International Council on Systems Engineering.

INCOSE, *Proceeding of Annual Conference*, Seattle, W.A.: International Council on Systems Engineering.

Jackson, S., *Systems Engineering for Commercial Aircraft*, Surrey, England, Ashgate, 2015.

Jenney, J., M. Gangl, R. Kwolek, D. Melton, N. Ridenour, and M. Coe, *Modern Methods of Systems Engineering*, CreateSpace Independent Publishing Platform, 2011.

Kamrani, A.K. and M. Azimi, *Systems Engineering Tools and Methods*, Boca Raton, FL: CRC Press, 2011.

Khisty, C.J. and J. Mohammadi, *Fundamentals of Systems Engineering With Economics, Probability, and Statistics,* Upper Saddle River, N.J.: Prentice-Hall, 2001.

Kossiakoff, A., W.N. Sweet, S.J. Seymour, and S.M. Biemer *Systems Engineering Principles and Practice*, Hoboken, New Jersey: John Wiley & Sons, 2011.

Lacy, J., *Systems Engineering Management: Achieving Total Quality*, New York: McGraw-Hill Book Co., 1992.

Larson, W., D. Kirkpatrick, J. Sellers, L. Thomas, and D. Verma, *Applied Space Systems Engineering*, McGraw-Hill, 2009.

Machol, R., *System Engineering Handbook*, New York: McGraw-Hill, 1965.

Maier, M. and E. Rechtin, *The Art of Systems Architecting*, Boca Raton, F.L.: CRC Press, 2009.

Martin, J., *Systems Engineering Guidebook: A Process for Developing Systems*

and Products, Boca Raton, F.L.: CRC Press, 1997.
MIL-STD-499B, *Military Standard—Systems Engineering—Draft*, Washington D.C.: United States of America Department of Defense, 1994.
Rechtin, E., *Systems Architecting: Creating and Building Complex Systems*, Upper Saddle River, N.J.: Prentice-Hall, 1991.
Reilly, N., *Successful Systems Engineering for Engineers and Managers*, New York: Van Nostrand Reinhold, 1993.
Sage, A., *Decision Support Systems Engineering*, New York: John Wiley & Sons, 1991.
Sage, A., *Systems Engineering*, New York: John Wiley & Sons, 1992.
Sage, A., *Systems Management for Information Technology and Software Engineering*, New York: John Wiley & Sons, 1995.
Sage, A., and J. Armstrong, *Introduction to Systems Engineering*, New York: John Wiley & Sons, 2000.
Sage, A. and W. Rouse (eds), *Handbook of Systems Engineering and Management*, New York: John Wiley & Sons, 2009.
SECMM-95-01, *Systems Engineering Capability Maturity Model*, Version 1.1, Carnegie Mellon University, Pittsburgh, P.A.: Software Engineering Institute, 1995.
Shishko, R., *ASA Systems Engineering Handbook*, Washington, D.C.: NASA, 1995.
Stevens, R., P. Brook, K. Jackson, and S. Arnold, *Systems Engineering: Coping with Complexity*, Hertfordshire, England: Prentice Hall Europe, 1998.
Sydenham, P.H., *Systems Approach to Engineering Design*, Norwood, MA: Artech House, 2004.
Thome, B. (ed), *Systems Engineering: Principles and Practice of Computer-Based Systems Engineering*, New York: John Wiley & Sons, 1993.
Truxal, J., *Introductory Systems Engineering Management*, New York: McGraw-Hill Book Co., 1972.
Wasson, C., *System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices*, Hoboken, NJ: John Wiley & Sons, 2015.
Westerman, H., *Systems Engineering Principles and Practice*, Boston, M.A.: Artech House, 2001.
Wymore, A., *Model-Based Systems Engineering*, Boca Raton, F.L.: CRC Press, 1993.

[2] See for example:
McDonald, G.R. and J.C. Price, "Economics of Trolley Coach Power Supply". *Transactions of the American Institute of Electrical Engineers*, 1951. 70(1): p. 687-689.
Rothstein, J., "An Informational Approach to Organization and System Engineering Design", *Transactions of the IRE Professional Group on Engineering Management*, 1954. PGEM-1: p. 25-29.
Vonpechmann, W., "Method and Systems Engineering", *Industrial and Engineering Chemistry*, 1950. 42(4): p. A73-A74.

[3] See:
Cole, R.I., "Management's Role in Research and Development of Electronic Systems", *Proceedings of the I.R.E.*, November 1950, pp. 1252-1253.
Cole, R.I., "Management Aspects of Electronic Systems Engineering", *Proceedings of the Institute of Radio Engineers*, 1951. 39(12): p. 1492-1493.

Cole, R.I., “The Electronic ‘Systems’ Engineer (Systems Project Director)”, *Proceedings of the I.R.E.*, November 1952, pp. 260-261.

[4] See for example:
Quarles, G.G., “The Modern Torpedo: A Case Study in Systems Engineering”, *Journal of the American Society for Naval Engineers*, 1956. 68(4): p. 801-804.
Ryan, F.M., “Systems Engineering”, *IRE Transactions on Communications Systems*, 1956. 4(3): p. 1 .
Schlager, K.J., “Systems Engineering—Key to Modern Development”, *IRE Transactions on Engineering Management*, 1956. EM-3(3): p. 64-66.
Montgomery, B.E., “Air-ground Communications: the ANDB System Development Plan”, *IRE Transactions on Communications Systems*, 1956. 4(2): p. 48-53.
Monsanto, “Training for Automation”, *Chemical and Engineering News*, 1957. 35(28): p. 38.
Noll, A.H., “Airborne Weapons System Development through Simulation”, *SAE National Aeronautic Meeting*, 1957.

[5] Goode, H. and Machol R., *Systems Engineering: An Introduction to the Design of Large Scale Systems*, New York: McGraw-Hill, 1957.

[6] Ackoff, R.L. and F.E. Emery, *On Purposeful Systems: An Interdisciplinary Analysis of Individual and Social Behavior as a System of Purposeful Events*, London: Tavistock Publications, 1972.

[7] ISO/IEC/IEEE 15288-2015, *Systems and Software Engineering—System Life Cycle Processes*, 2015.

[8] Flood, R.L. and E.R. Carson, *Dealing With Complexity: An Introduction to the Theory and Application of Systems Science*, New York: Plenum Press, 1993.

[9] ANSI/EIA-632-1998, *Processes for Engineering a System*, Washington, D.C.: Electronic Industries Association (EIA), 1999.

[10] US Department of Defence, *Joint Capabilities Integration and Development System*, CJCSI 3170.01G, 2009.

[11] Commonwealth of Australia, *The Defence Capability Development Handbook*, 2012.

[12] UK Ministry of Defence, *Defence Lines of Development*, 2009.

[13] Canadian Department of National Defence, *Designing Canada’s Army of Tomorrow*, 2011.

[14] ISO/IEC/IEEE 15288-2015, *Systems and Software Engineering—System Life Cycle Processes*, 2015.

[15] IEEE-STD-1220-2005, *IEEE Standard for Application and Management of the Systems Engineering Process*, New York: IEEE Computer Society, 2005.

[16] ISO/IEC/IEEE 15288-2015, *Systems and Software Engineering—System Life Cycle Processes*, 2015.

[17] ISO/IEC/IEEE 21839-2019, *Systems and Software Engineering — System of Systems*, 2019.

[18] Maier, M.W., “Architecting a Portfolio of Systems”, *Systems Engineering*, 1(4), 1-13, 2019.

[19] ISO/IEC/IEEE 15288-2015, *Systems and Software Engineering—System Life Cycle Processes*, 2015.

[20] ANSI/EIA-632-1998, *EIA Standard—Processes for Engineering a System*, Arlington, V.A.: Electronic Industries Association, 1999.

[21] MIL-STD-499B, *Military Standard—Systems Engineering—Draft*, Washington D.C.: United States of America Department of Defense, 1994.

[22] ANSI/EIA-632-1998, *EIA Standard—Processes for Engineering a System*, Arlington, V.A.: Electronic Industries Association, 1999.

[23] MIL-STD-499B, *Military Standard—Systems Engineering—Draft*, Washington D.C.: United States of America Department of Defense, 1994.

[24] ANSI/EIA-632-1998, *EIA Standard—Processes for Engineering a System*, Arlington, V.A.: Electronic Industries Association, 1999.

[25] Blanchard, B. and W. Fabrycky, *Systems Engineering and Analysis*, Upper Saddle River, N.J.: Prentice-Hall, 1998.

[26] Blanchard, B. and W. Fabrycky, *Systems Engineering and Analysis*, Upper Saddle River, N.J.: Prentice-Hall, 2011.

[27] Further detail on these models can be found in, inter alia:
Blanchard, B. and W. Fabrycky, *Systems Engineering and Analysis*, Upper Saddle River, N.J.: Prentice-Hall, 2011.
Martin, J., *Systems Engineering Guidebook: A Process for Developing Systems and Products*, Boca Raton, F.L.: CRC Press, 1997.
Young, R., *The Requirements Engineering Handbook*, Norwood, M.A, Artech House, 2004, pp. 73−78.

[28] Forsberg, K. and H. Mooz, *Systems Engineering Overview*, Center for Systems Management, Cupertino, CA., 1996.

[29] Forsberg, K. and H. Mooz, *'Application of the 'VEE' to Incremental and Evolutionary Development'*, *Proceedings of the NCOSE Conference*, St. Louis, MO., July 1995.

[30] Defense Systems Management College (DSMC), *Systems Engineering Management Guide*, Washington, D.C.: U.S. Government Printing Office, 1990.

[31] IEEE-STD-1220-1994, *IEEE Trial-Use Standard for Application and Management of the Systems Engineering Process*, New York, N.J.: IEEE Computer Society, 1995.

[32] EIA/IS-632-1998, *Systems Engineering*, Washington, D.C.: Electronic Industries Association (EIA), 1994.

[33] SECMM-95-01, *Systems Engineering Capability Maturity Model*, Version 1.1, Carnegie Mellon University, Pittsburgh, P.A.: Software Engineering Institute, 1995.

[34] Lake, J., *Unraveling the Systems Engineering Lexicon*, Proceedings of the INCOSE Symposium, 1996.

[35] INCOSE, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, Walden D.D, et al, (eds), INCOSE-TP-2003-002-04 2015, Wiley, 2015.

[36] INCOSE Fellows, *Systems Engineering and Systems Definitions*, 2018.

[37] ANSI/EIA-632-1998, *Processes for Engineering a System*, Washington, D.C.: Electronic Industries Association (EIA), 1999.

[38] Weinberg, J., *Quality Software Management. Volume 4 Anticipating Change*, Dorset House, 1997.

[39] ANSI/EIA-632-1998, *Processes for Engineering a System*, Washington, D.C.: Electronic Industries Association (EIA), 1999.

[40] Meier, S., “Best Project Management and Systems Engineering Practices in the Preacquisition Phase for Federal Intelligence and Defence Agencies”, *Project Management Journal*, pp. 59–71, March 2008.

[41] Blanchard, B. and W. Fabrycky, *Systems Engineering and Analysis*, Upper Saddle River, N.J.: Prentice-Hall, 1998.

[42] Asimov, M., *Introduction to Design,* Englewood Cliffs, N.J.: Prentice-Hall, 1962.

[43] *PMBOK, A Guide to the Project Management Body of Knowledge*, Upper Darby: Project Management Institute, 2017.

[44] Abran, A. and J. Moore (eds), *Guide to the Software Engineering Body of Knowledge*, Los Alamitos, C.A., IEEE Computer Society, 2004.

[45] Faulconbridge, R., *Systems Engineering Body of Knowledge*, Canberra: Magpie Applied Technology, 2000.

[46] Faulconbridge, R., “A Systems Engineering Framework”, *Journal of Battlefield Technology*, vol. 3, no. 2, July 2000.

[47] MIL-STD-499B, *Military Standard—Systems Engineering—Draft*, Washington D.C.: United States of America Department of Defense, 1994.

[48] EIA/IS-632-1994, *Systems Engineering*, Washington, D.C.: Electronic Industries Association (EIA), 1994.

[49] IEEE-STD-1220-2005, *IEEE Standard for Application and Management of the Systems Engineering Process*, New York: IEEE Computer Society, 2005.

[50] ANSI/EIA-632-1998, *Processes for Engineering a System*, Washington, D.C.: Electronic Industries Association (EIA), 1999.

[51] SAE1001 *Integrated Project Processes for Engineering a System*, SAE International, 2018.

[52] ISO/IEC/IEEE 15288-2015, *Systems and Software Engineering—System Life Cycle Processes*, 2015.