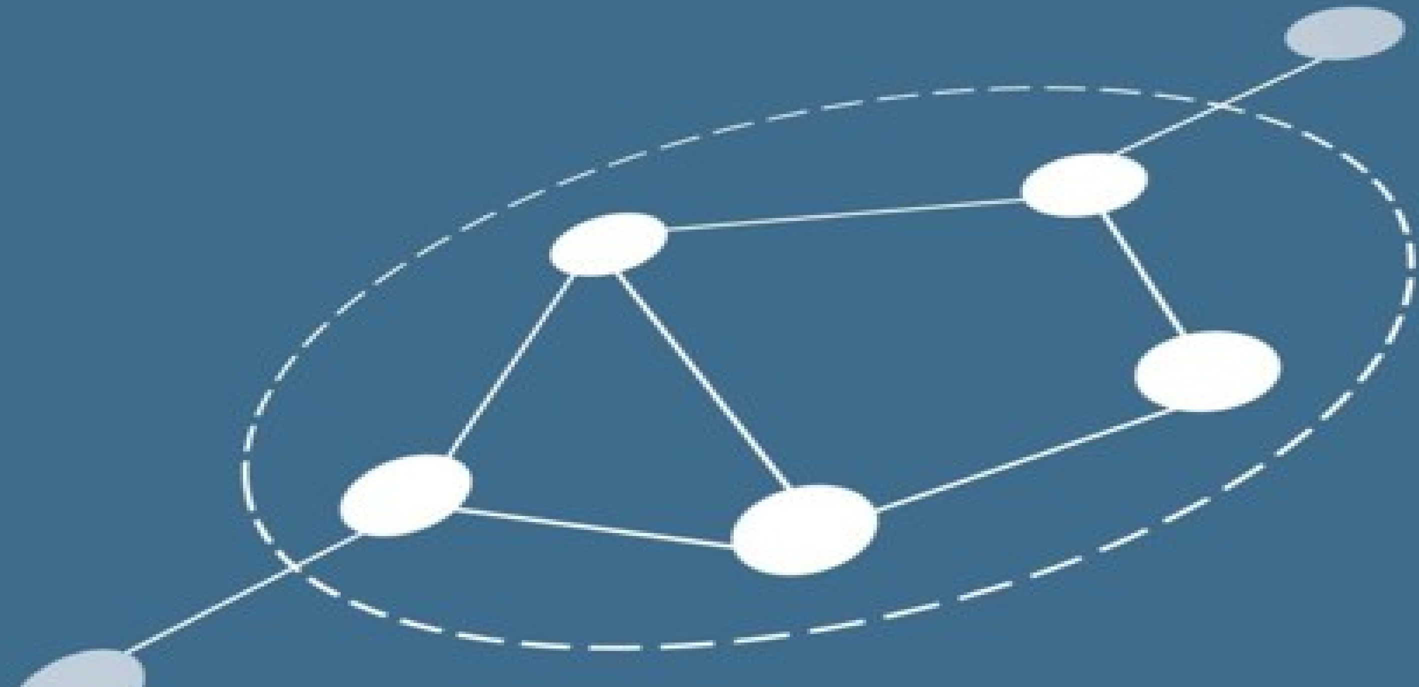
INTRODUCTION TO SYSTEMS ENGINEERING



Ian Faulconbridge
Michael Ryan

ArgosPress

INTRODUCTION TO SYSTEMS ENGINEERING

Ian Faulconbridge
Michael Ryan

ArgosPress

INTRODUCTION TO SYSTEMS ENGINEERING

R. Ian Faulconbridge
Michael J. Ryan

ISBN: 9781921138089

Cover design by Tim Ryan

Published by Argos Press, Canberra, Australia

www.argospress.com

PREFACE

This e-book provides a basic coverage of the discipline known as *systems engineering*. The e-book supports the Massive Open Online Course (MOOC) that the authors deliver regularly through the Coursera platform and is an overview of our more-comprehensive physical text *Systems Engineering Practice*. Here, we offer a framework encapsulating the entire systems engineering discipline, clearly showing where the multitude of associated activities fits within the overall effort, providing an ideal vehicle for understanding the complex discipline.

Chapter 1 introduces systems engineering and related issues and Chapter 2 provides an overview of requirements-engineering and its application to systems development. Chapters 3 and 4 examine in more detail the issues associated with Conceptual Design and Preliminary Design. Emphasis is given to these early activities, as they have the greatest impact on the system development. Chapters 5, 6 and 7 deal with respectively Detailed Design and Development, Construction and/or Production, and Operational Use and Support. Chapter 8 deals with the broad topic of systems engineering management and details some of the associated activities.

Ian Faulconbridge
Mike Ryan
Canberra, 2015

CONTENTS

1 SYSTEMS AND SYSTEMS ENGINEERING

Systems engineering provides the framework within which complex systems can be adequately defined, analysed, specified, manufactured, operated, and supported. 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.

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. 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. 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

ISO/IEC 15288 defines a system as *a combination of interacting elements organized to achieve one or more stated purposes* [1]. As illustrated in Figure 1-1, the very act of identifying the system that we are interested in, implies an external system *boundary* and a *system of interest (SOI*).

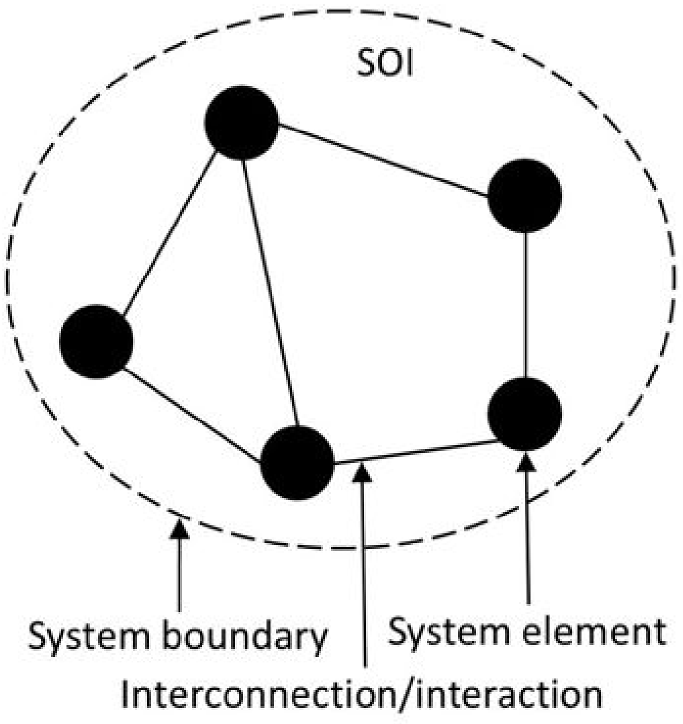


Figure 1-1. An SOI: its elements, interconnections, and boundary.

The purpose of the system is called its *mission*—in the broadest sense, the mission of the system is to provide a solution to a business problem. 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*. Further, 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).

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 this book:

- *Closed/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. We are only interested in useful systems, which are therefore open.
- *Natural/human-made/human-modified systems.* Natural systems contain natural elements that result from 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. Natural systems that have been modified for human purposes are 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/conceptual systems.* Physical systems exist in a physical form; conceptual systems do not have a physical form. We focus here on physical systems.
- *Precedented/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. In systems engineering we are interested in 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 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.

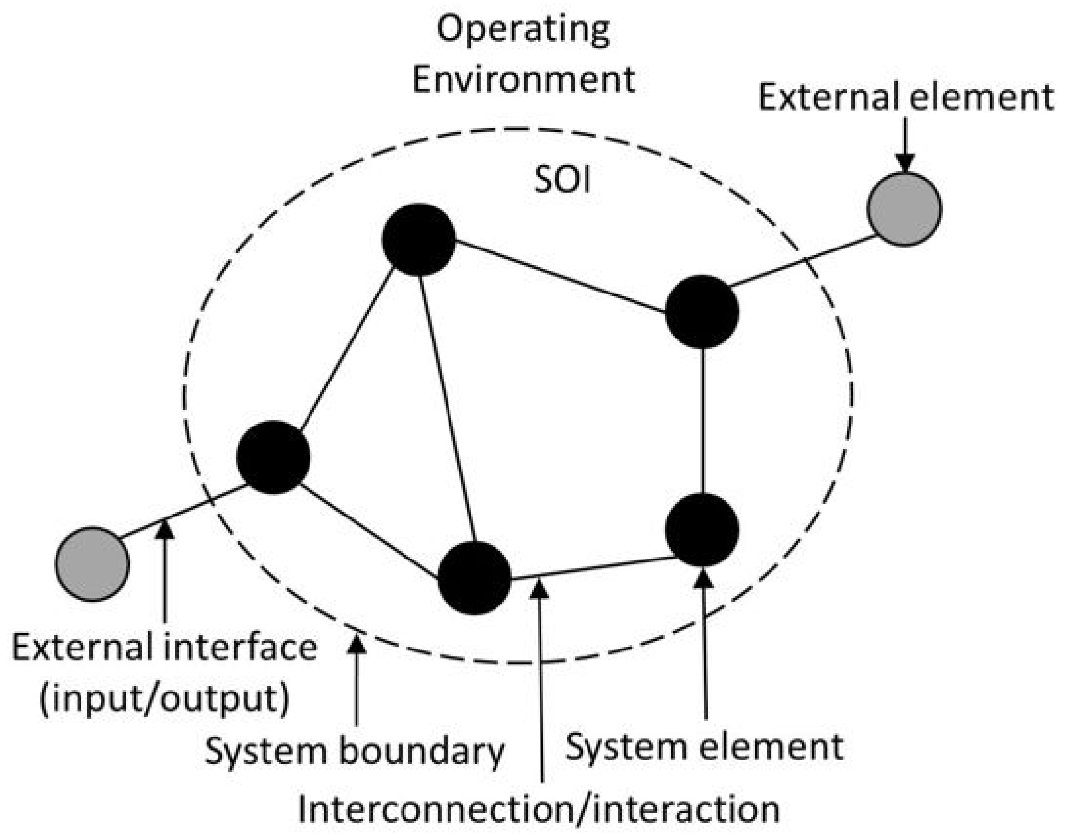


Figure 1-2. An SOI: its elements, interconnections, boundary, and interfaces to external elements in the operating environment.

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 such as operational products (end products) and enabling products (such as test, training, and disposal products).

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 and must also 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 its 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, to refer to the system at this level as a *capability system*.

Having acknowledged all of the elements of a capability system, it must also be recognised 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.

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 tested, 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 tested. 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*.

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 configuration items contained in the work breakdown structure (WBS).

1.1.7 Hierarchical Descriptions of a System

We saw earlier that ISO/IEC 15288 defines a *system* as a combination of *system elements* which interact to achieve a defined mission. The system elements 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, 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*.

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.

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 will be 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).

1.1.8 System-of-systems (SoS)

When the SOI consists only of system elements that are systems in their own right—the system-of-interest is called a *system-of-systems (SoS)*. An SoS has a similar architecture to that of a system, in that both comprise elements that are interconnected. The difference is that the SoS elements are systems in their own right so that they are managerially independent and operationally independent and have been optimised for their own purpose before contributing to the purpose of the SoS. 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 optimised, not the constituent subsystems.

The major distinction between systems as elements of an SoS and subsystems as elements of a system is therefore that the SoS comprises elements (systems) that are optimised for their own purposes before joining the SoS, whereas the system comprises elements (subsystems) that are optimised for the system’s purpose (not necessarily their own). Or, from the higher perspective, an SoS is most likely not optimised because the elements (the systems) are first optimised for their own purposes, whereas a system is optimised because the subsystems are designed for the system’s optimisation, not their own.

As we observed earlier, the systems in an SoS are managerially and operationally independent and will no doubt have independent life cycles (they will almost certainly have been procured separately). In a system, on the other hand, the system elements (the subsystems) will be procured at the same time as the system, have the same life cycle and be designed to serve the system’s purpose, rather than their own. In this text, we focus on systems whose elements are all subsystems.

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 [2]. 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 in the organisation. 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 the system, 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 will invariably mark the start of another and the process begins again.

1.2.1 Parties Involved

Throughout the system life cycle, there are a number of parties involved. The *customer* organization is managed by *enterprise management* who set the direction for the organisation and *business management* who are responsible for the activities conducted by the *operations* element of the organisation (run by the *operators*—sometimes called *users*). The systems used within the organisation are acquired by the *acquisition* element of the organisation under the auspices of a *project* (typically 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* who may deliver the system off-the-shelf or may develop it, in which case they are often called the *developer*. 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. 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.

1.3 ACQUISITION AND UTILIZATION PHASES

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 life-cycle activities based on those defined by Blanchard and Fabrycky [3]. 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.

The significance of focusing on the system life cycle is that decisions made early in Conceptual Design are informed by the intended activities later in Acquisition Phase 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 or operators to operate it in the intended environment.

1.3.1 Acquisition Phase

The Acquisition Phase comprises 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.

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; or where focus shifts from the problem domain into the solution domain. The result of the Preliminary Design process 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 configuration item.

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 and Retirement 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 may 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. 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.

1.4 SYSTEMS ENGINEERING AND DEVELOPMENT APPROACHES

It should be noted that the generic system life cycle 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 presented 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 the *waterfall approach* to system development. There are, however, a number of other development approaches to implementing the activities of the system life cycle—such as the *incremental*, *spiral*, or *evolutionary acquisition* models, 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 activity early in a system life cycle.

However, for simplicity here, 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. A solid understanding of waterfall development is therefore useful.

1.5 WHAT IS SYSTEMS ENGINEERING?

There is a wide range of definitions of systems engineering, each of which is subtly different because it tends to reflect the particular focus of its source. 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 criteria. This approach is valid and extremely useful for relatively straightforward problems that are well defined. Unfortunately, complicated problems cannot be solved using 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).

It must be recognized, however, that while design is conducted top-down, the system is implemented using a bottom-up approach 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.

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 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. 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, not on the product.

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. It does not necessarily follow that the combination of optimized subsystems leads to an optimized system. Consequently, 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, 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.

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.

1.6 SYSTEMS ENGINEERING RELEVANCE

Systems engineering principles and processes are applicable (albeit to varying degrees) to a wide range of projects. 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. Because it is so widely applicable, however, 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.

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 scales, 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 applied to each of these types of projects and the aspects we consider here must be tailored for each individual project.

1.7 SYSTEMS ENGINEERING BENEFITS

The principal causes of cost and schedule overrun on large-scale programs can be traced to overzealous advocacy, immature technology, lack of corporate roadmaps, requirements instability, ineffective acquisition strategy, unrealistic program baselines, program office personnel tenure and experience, and inadequate systems engineering [4]. In this book we focus on the latter 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.

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.

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. 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, 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.

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 makes for a quality system where quality is measured by the ability of the system to meet the documented requirements.

2
REQUIREMENTS ENGINEERING FRAMEWORK

2.1 INTRODUCTION

We saw in Chapter 1 that systems engineering (particularly during the early processes of Conceptual Design and Preliminary Design) has a focus on gathering requirements in a formal systematic way, which is referred to generically as *requirements engineering*. Before considering each of the major systems engineering processes in some detail in subsequent chapters, this chapter provides an overview of the extant requirements-engineering body of knowledge because it is such an essential aspect of system development.

2.1.1 Needs and Requirements Documentation

There are needs and requirements at a number of levels. There is an enterprise view (in which enterprise leadership sets the enterprise strategies and concepts of operations); a business management view (in which business management derive business needs and constraints as well as formalize their requirements); a business operations view (in which stakeholders define their needs and requirements); and a systems view (in which the system is defined in logical and physical views).

The enterprise, business management, and business operations views are in the problem domain; the system and subsystem (and lower) views are in the solution domain. The problem domain is generally considered to be the responsibility of those who have ownership of the problem to be solved, so the descriptions of the system are predominantly in the language of the customer's business management and business operations, focusing on what the system needs to be able to do, how well it should be done, and why—these descriptions are called *logical* (or often *functional*) descriptions. On the other hand, the solution domain is generally considered to be the responsibility of those implementing the solution, so the descriptions of the system in that domain are predominantly in engineering and physical terms, focusing on how the problem is to be solved—that is, how it will look once it has been implemented. These latter descriptions are called *physical* descriptions.

In addition to these vertical views of the process, there are two horizontal views: the *Needs View* and the *Requirements View*. While they tend to coexist at each level of consideration, needs and requirements are different:

- Needs are capabilities stated in the language of the business at the business management or business operations levels.
- Requirements are formal structured statements that can be validated—there may be more than one requirement defined for any one need.

So, at the upper levels, there are the dual views: at the business management level there are *Business Needs* and *Business Requirements*; and at the business operations level there are *Stakeholder Needs* and *Stakeholder Requirements*. The transformation of needs into requirements occurs at each level and involves three major processes:

- *Business Needs and Requirements (BNR) Definition Process.* The definition of needs and requirements begins with the organization's business vision, goals and objectives, and the Concept of Operations (ConOps), from which business management defines Business Needs, largely in the form of Preliminary Life-cycle Concept Documents (PLCD), which capture the Preliminary Acquisition Concept, Preliminary Operational Concept (OpsCon), Preliminary Deployment Concept, Preliminary Support Concept, and Preliminary Retirement Concept. Business Needs are elaborated and formalized into Business Requirements, which are documented in the Business Requirements Specification (BRS), which is also called the Business Requirement Document.
- *Stakeholder Needs and Requirements (SNR) Definition Process.* Using the ConOps and the other PLCD as guidance, requirements engineers lead stakeholders from the business operations level through a structured process to elicit Stakeholder Needs—in the form of a refined OpsCon document and other Life-cycle Concept Documents (LCD)—and transform them into the formal set of Stakeholder Requirements, which are documented in the Stakeholder Requirement Specification (StRS).

- *System Requirements Definition Process*. The requirements in the StRS are then transformed by requirements engineers into System Requirements, which are documented in the System Requirement Specification (SyRS) [5] (also referred to simply the System Specification). A number of SyRS may be derived from the StRS. Some organizations may prepare individual LCD for each of a number of systems that are developed to meet the Business Needs.

The ConOps is developed by the leadership at the enterprise level. The ConOps provides the context for the OpsCon, which is prepared at the business management level. Business management begins with the preparation of the Preliminary OpsCon, which summarizes the needs of the system within its business context—the Business Needs. The Preliminary OpsCon is then elaborated and refined into the OpsCon by engagement with stakeholders at the business operations level during the Stakeholder Needs and Requirements Definition Process—the OpsCon therefore contains Stakeholder Needs.

The OpsCon, however, is just one of the concepts required to address the Stakeholder Needs across the system life cycle. In addition to the operational aspects, other related life-cycle concept documents are required to address the following concepts:

- The *Acquisition Concept* describes the way the system will be acquired including aspects such as stakeholder engagement, requirements definition, solicitation and contracting issues, design, production, and verification.
- The *Deployment Concept* describes the way the system will be validated, delivered, and introduced into operations.
- The *Support Concept* describes the desired support infrastructure and manpower considerations for supporting the system after it is deployed. A support concept addresses operating support, engineering support, maintenance support, supply support and training support.
- The *Retirement Concept* describes the way the system will be removed from operation and retired, including the disposal of any hazardous materials used in or resulting from the process.

PLCD are prepared by business management and contain the Business Needs defined as part of the Business Needs and Requirements Definition

Process. Preliminary life-cycle concepts are elaborated and refined in much more detail in the LCD by stakeholders at the business operations level in the Stakeholder Needs and Requirements Definition Process.

2.1.2 What is a Requirement?

For our purposes here, a requirement can be defined as a statement of a system service, an attribute or a quality of the system, or a constraint placed on the system. A requirement is therefore:

- something that the system must do,
- a quality or attribute that it must possess, or
- a constraint under which it must operate or be developed.

As described earlier, there are three levels of requirement statement generated in the BRS, StRS and the SyRS as part of the development of requirements for a system:

- Requirement statements—statements of:
  - the services and functions that the system should provide, the things it should do, or some action it should take (often called functional requirements);
  - the qualities, properties, or attributes that the system must possess (often called non-functional requirements); and
  - any restrictions or bounds under which the system should operate, or on the way in which the system is to be developed (also called constraints and sometimes also included in the category of non-functional requirements).
- Performance, verification, and rationale statements supporting each requirement.
- Definitions of other systems with which the system must integrate and to which it must interface.
- Information about the application domain within which the system must operate.

The requirements in the BRS and the StRS are not as formal as those in the SyRS but they have the same structure and follow the same rules as presented in the remainder of this chapter.

2.1.3 What not How

In general, business, stakeholder and system requirements should be statements of *what* a system should do, rather than *how* it should do it. However, for a number of reasons, this is often too simplistic in practice. First, it may be essential that the system should perform a function in a particular way (such as for interoperability, or to meet extant standards). We could also specify a solution to meet broader concepts such as support and commonality with existing solutions. Additionally, a specific statement is often less confusing than an abstract statement of the problem—a specific statement is therefore the best guidance that a conceptual designer can give a lower-level designer. Further, for good business reasons, business management may well dictate not only what is required but also how it is to be implemented. Finally, the people who specify the system are often the domain experts—they are therefore best placed to state how the system should be developed and how it should operate. Still, unless one of those reasons prevails, we should ensure that the BRS, StRS and SyRS provide logical specifications (because they are in the solution domain), not physical specifications (which belong in the solution domain).

2.1.4 Emergent Properties

Not all properties of a system are exhibited separately by its constituent elements. That is, the system may have properties that only exist when the subsystems are integrated—these are called the *emergent properties* of a system. Emergent properties:

- are those that are possessed by the system as a whole;
- only emerge after all the individual subsystems have been integrated;
- cannot be exhibited by parts of the system in isolation; and
- depend on interactions between components, including their environment.

Examples of emergent properties are the ‘ilities’ such as: maintainability, reliability, availability, and usability. Other examples include safety, security, comfort, and ease of use. Emergent properties must therefore be defined from the top down—they cannot be obtained by specifying requirements for individual subsystems and then hoping that the desired properties will exist on integration.

Perhaps the bicycle is one of the best examples of a system that possesses emergent properties, since it can only perform its principal function when all subsystems, including the rider, have been integrated. If any one of the major subsystems is removed, the system cannot function correctly.

We also must consider the possibility of the completed system exhibiting emergent properties that were not considered as part of its design and, in some cases, may be undesirable in the end result.

2.2 WHAT IS REQUIREMENTS ENGINEERING?

Requirements engineering can be described most simply as the process by which we identify a problem's context, locate the business and stakeholder needs and requirements within that context, and deliver specifications that meets those needs [6]. That process requires a considerable reduction of the complexity of the real world, and that subset within which the customer organization operates, into a set of requirements that define the scope of the project. There is therefore considerable judgement required in identifying BNR and SNR and then converting them into a SyRS.

2.2.1 Why We Need Requirements

It may not always be obvious why requirements are necessary in the first place. An agreed set of requirements is essential, however, from a number of different perspectives.

From the point of view of the business management, the set of requirements provides a mechanism whereby they can agree that the organization's resources can be allocated to the project that has been established to bring the system into being. Unless the requirement set is well defined, the project has no firm base and the resources allocated to it will be inappropriate. Worse still, without sufficient oversight, the business may be held to ransom by stakeholders insisting on unrealistic requirements that business management have not approved.

From the project manager's perspective, the set of requirements is an essential part of the definition of the scope of the project—it is from this scope that a suitable system will be developed. Scope management is one of the nine major functions of a project manager [7], and a well defined scope (set of requirements) is necessary to able to justify any expenditure of funds or effort within the project, to be able to report adequately on the progress of work, and eventually to be able to determine when the project is complete. Clearly then, if there is not a well defined, universally agreed set of requirements at the outset of the project, the project is doomed to fail.

From the systems engineering perspective, the requirements set instantiated in the SyRS represents the transition from the business world into the systems engineering and engineering worlds. Once the set has been established, it is not possible to rectify a poor set of requirements with good design, good engineering, or good production/process control—particularly if that set of requirements is the basis of a fixed-price contract between the acquirer and the contractor. Consequently, a project can rarely recover from poor definition, regardless of how much good design work is performed subsequently.

2.2.2 Why We Need Requirements Engineering

An appropriate requirements-engineering process allows us to have the best chance that we begin the project with a set of requirements that is complete, balanced, comprehensible, feasible, and consistent. A useful process ensures that all relevant business managers and stakeholders have had input and the various points of view have been reconciled. To do that, we must therefore be able to trade off functionality for cost—which implies that we understand the required functionality, priorities, and costs. We also need to be able to use the process to manage changes in requirements for a wide variety of reasons.

Requirements engineering focuses on what is needed by the system, rather than how it is to be designed. As discussed in the previous section, poor requirements definition cannot be rectified by good design, so that it invariably follows that rigorous development of requirements is essential for the successful system—requirements engineering provides this rigour through the application of a formal methodology.

2.3 REQUIREMENTS ELICITATION AND ELABORATION

Requirements elicitation and elaboration involve working with business managers and stakeholders to investigate the problem to be solved, and to identify their needs and requirements. At the beginning of this chapter, we considered the three levels of requirement statement:

- business requirements in the BRS;
- stakeholder requirements in the StRS; and
- system requirements in the SyRS.

We then discussed requirements engineering by referring to ‘requirements’ in general terms. Before discussing in subsequent chapters how these requirements are part of systems engineering, it is useful here to discuss how the design moves through those three levels of requirement statement. To do so, we need to consider the hierarchical nature of the requirement statements and the processes of *elicitation* and *elaboration* (which involves *decomposition* and *derivation*) that is used to move between the levels in the requirements hierarchy—the process is also often referred to as *requirements flowdown*. In general, requirements in the BRS, StRS and SyRS fall into three categories:

- *Elicited requirements.* Elicited requirements can be attributed to the source (business manager or stakeholder), either directly or through negotiation. These requirements are normally gathered via interview or a structured workshop.
- *Decomposed requirements.* Elicited requirements tend to be at a high level because they were obtained from those directly involved in the business management and business operations supported by the system of interest. Decomposition entails breaking a higher-level requirement into those lower-level requirements that are explicitly required by it. For example, if an aircraft needs to be able to land on a certain class of airfield, then it must be able to land within the appropriate runway length, be less than the maximum allowable weight, be able to interface to the appropriate air traffic control systems, and so on. In decomposition, the need for the children requirements is obvious in the parent. Requirements engineers take an elicited requirement and decompose it into two or more subordinate requirements whose total meaning is equivalent

to that of the original requirement. It should be noted that the original requirement normally becomes redundant and is replaced by the lower-level requirements. The essence of the upper-level requirement often becomes a heading or is noted in the rationale for the decomposed requirements, although it is good practice to retain the hierarchical structure of the RBS by abstracting the original requirement to a heading in the specification structure.

- *Derived requirements.* In addition to being high level, elicited requirements also tend to be from the business management and operations perspectives and will not necessarily address all aspects of a systems design. Derivation entails requirements engineers drawing some inference from the higher-level requirements to obtain lower-level statements. That is, business management or the stakeholders did not state the requirement directly, but the derived requirement is a necessary part of the system design if one or more of the directly stated requirements are to be met. Unlike decomposed requirements, derived requirements do not replace the original parent requirement.

Business, stakeholder, and system requirements are not collated randomly. High-level requirements are elaborated further into a number of lower-level requirements which are then grouped, along with other elaborated requirements into the business-management-level descriptions in the BRS, the business-operations-level descriptions in the StRS, and the system-level description in the SyRS.

The requirements in the SyRS are then further elaborated into lower-level requirements which are then re-grouped from the logical groupings of the SyRS into physical groupings of requirements. These system elements (called generically subsystems at the first level) are documented in a number of Subsystem Specifications, which then define ‘how’ the system will be implemented. The mapping of the logical groupings of requirements in the SyRS into the physical groupings in the Subsystem Specifications informs the definition of project deliverables.

For well behaved systems, the requirements in each of the BRS, StRS, SyRS and Subsystem Specifications are normally grouped in a hierarchical fashion (albeit grouped logically in the first two, and physically in the last). This hierarchy is useful because it allows requirements engineers to decompose the system into tasks of a complexity that can be more easily be understood and communicated, as well as managed within the limitations of humans and our organizations. A hierarchical description of a system is also useful in that it supports requirements traceability, which is an essential element of effective systems engineering as well as project management. Traceability is assisted by a hierarchical numbering of requirements.

Elicitation involves engagement with the stakeholders in order to establish their requirements for the new system. There are a number of techniques for this engagement, such as facilitated structured workshops, brainstorming, interviews, surveys, questionnaires, process models, use cases and user scenarios, simulations, prototypes, observation of work studies (time and motion studies), participation in work activities, observation of the system’s organizational and political environment, technical documentation review, benchmarking, competitive system assessment, reverse engineering, and market analysis.

2.4 REQUIREMENTS VALIDATION

Requirements must be validated in order to ensure that individual requirements are appropriate and that the requirements set is an adequate description of the desired system. This process is intended to detect any problems and gaps in the requirements before they are used as a basis for system development. In particular, the SyRS must be validated to show that it meets the stakeholders' original requirements in the StRS, which must be validated to show that it meets the business requirements in the BRS.

Requirements validation is very difficult—somehow, we have to ensure that we have a complete set of requirements that, in their aggregation, represent the system required by the stakeholders and meets the business needs. This is difficult because there is no significant baseline we can refer back to for the validation. Later on in the life cycle, we will look back to the previous baseline—for example in Preliminary Design, we look back to the SyRS; in Detailed Design and Development, we look back to the Development Specifications; and in Construction and Production we look back to the Product Specifications. In Conceptual Design, however, we only have the high-level business descriptions as the start point. It is therefore not a straightforward matter to validate that the StRS and then the SyRS represent the complete set of requirements associated with meeting those needs. Validation is normally conducted by a formal requirements review, or a series of reviews.

2.5 REQUIREMENTS DOCUMENTATION

Writing well-formed requirements can be achieved by using the following guidelines 3:

- Ensure each requirement is a necessary, short, definitive statement.
- Define the appropriate conditions (qualitative or quantitative measures) for each requirement.
- Ensure that each requirement is verifiable through the discipline associated with writing a verification statement for every requirement statement.
- Avoid over-specification, unnecessary constraints and unbounded statements.
- Ensure readability by defining standard templates for describing requirements; using natural language simply, consistently and concisely; using short sentences and paragraphs, as well as lists and tables; supplementing natural language with other descriptions where appropriate, such as equations if they are the most unambiguous way of describing requirements; and using diagrams where appropriate to show relationships between entities.

Requirements must be placed into a structured format. For example, a paper-based format for a specification uses columns to contain most of the information required to be recorded along with the requirements. Note that it is normally not possible to include all the information on a single sheet of paper in a manageable way. The difficulties with paper-based formats for specifications can be overcome with the use of automated requirements management tools.

2.6 REQUIREMENTS MANAGEMENT

Requirements management provides the processes by which changes to requirements are managed throughout the system life cycle. Requirements management involves establishing and ensuring compliance with a formal procedure for the elicitation/generation, verification, and traceability of requirements. The many detailed requirements in lower-level specifications have been derived from a relatively simple statement made by a stakeholder; these lower-level requirements must be managed as they are developed to ensure that each of the lower-level requirements can be justified. Additionally, a formal procedure is required for the control of changes to requirements.

2.6.1 Requirements Change Management

Throughout the entire life cycle of the system of the system we are very interested in managing the configuration of the system elements. In the earliest stages we must manage the configuration of the requirements set—this process involves a formal change-management process to ensure that changes to the requirements set occur in a controlled way. Requirement change management is therefore a form of configuration management (which we discuss in Chapter 8). Later, we become interested in the configuration management of physical elements, which begins as soon as configuration items are defined at the end of Preliminary Design.

2.6.2 Change Management—Traceability

Requirements cannot be managed effectively without *requirements traceability*, which allows us to be able to identify where each requirement came from, what requirements are related to it, and what requirement or requirements stemmed from it (that is, which is its child, or are its children). If a requirement must change for some reason, good traceability allows us to identify all requirements affected by the change. Consequently, traceability is essential to support change management and supports the project management function of project scope management. Broadly, there are two main sorts of traceability:

· *Forward traceability.* Forwards traceability gives us confidence that each requirement has been addressed in the design. Through forward traceability, design decisions can be traced from any given system-level requirement (a parent requirement) down to a detailed design decision (a child requirement). For each requirement, we must be able to find at least one child in the subordinate design documents. If there is more than one child, we must be able to identify all of them.

- *Backward traceability.* We must be able to trace from each requirement back to at least one requirement in the parent document. This backward traceability ensures that additional requirements (not formally endorsed by the customer) have not crept (through requirements creep) into the design. 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. That is, an orphan requirement must, by definition in such a process, be out-of-scope.

Backwards and forwards traceability can mean that an enormous amount of information must be recorded. Consequently, one of the key decisions to be made early in the requirements-engineering process is the degree to which traceability is implemented and the point at which traceability and change management start. If change management starts too early (with the mission, goals and objectives, for example) the large number of changes in the early iterations are not important and tracking them leads to a huge amount of information that is relatively useless. It is generally better to allow the design to mature to a reasonable degree, than it is to insert traceability information as soon as the requirements are entered into a database.

2.6.3 Requirements Management Tools

Due to the large number of requirements elicited/elaborated during the development of the SyRS, a requirements management tool is generally necessary to assist in the management of requirements. In particular, it is almost impossible to have requirements traceability without implementing the requirements in some automated context.

2.7 REQUIREMENTS-ENGINEERING TOOLS

There are a large number of tools that may assist in requirements engineering, including the context diagram, functional flow block diagrams (FFBD), requirements breakdown structure (RBS), and N2 diagrams. Other tools include structured analysis, data flow diagrams (DFD), control flow diagrams (CFD), IDEF diagrams, behaviour diagrams, action diagrams, state/mode diagrams, process flow diagrams, function hierarchy diagrams, state transition diagrams (STD), entity relationship diagrams (ERD), structured analysis and design, object-oriented analysis (OOA), unified modelling language (UML), structured systems analysis and design methodology (SSADM), and quality function deployment (QFD). Each of these techniques focuses on gathering requirements in a formal systematic way.

In this text we focus on the philosophy of requirements engineering using the upper-level tools of the context diagram, FFBD, and RBS. This section provides a brief introduction to the RBS and FFBD, which are used in the remaining chapters to illustrate the processes, activities and steps necessary to implement the methodology presented. The use of a context diagram is discussed in more detail in Chapter 3.

2.7.1 Requirements Breakdown Structure (RBS)

The requirements-engineering process can be assisted by the use of a requirements framework around which the logical description of the system can be based. Here we call the requirements framework the *requirements breakdown structure* (RBS) because, as illustrated in Figure 2-1, the requirements are shown in a tree-structured elaboration of the mission. The words are deliberately chosen to differentiate this structure from the well-known project management document called the *work breakdown structure* (WBS) [8]—the RBS is grouped logically (functionally), the WBS is structured by physical work packages (including CIs) and contains other project-related work—the logical groupings of the RBS are then allocated to the physical groupings of the CIs in the WBS. The RBS is also called a *functional hierarchy*.

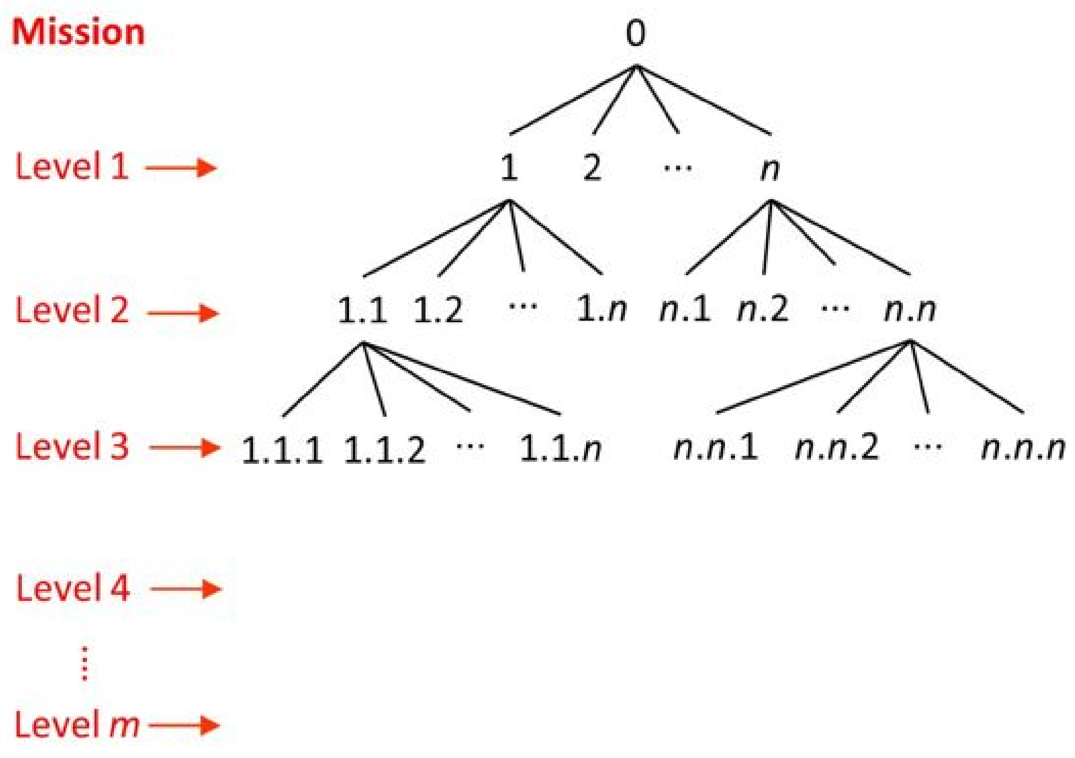

Mission
0
Level 1
1 2 ··· n
Level 2
1.1 1.2 ··· 1.n n.1 n.2 ··· n.n
Level 3
1.1.1 1.1.2 ··· 1.1.n n.n.1 n.n.2 ··· n.n.n
Level 4
Level m

Figure 2-1. A simple RBS showing the hierarchical elaboration of requirements for the mission statement down to m levels.

In addition to being a useful requirements analysis tool, the RBS can be used to capture the business requirements in the BRS, which can be further refined to reflect stakeholder requirements in the StRS, and then the system requirements in the SyRS. When completely populated, the RBS for a system provides an outline of the SyRS. In the first instance, the system is described by the words or short phrases of the RBS; in the SyRS, these short titles are elaborated into well formed requirement statements articulating functional, performance, and verification requirements.

There are a number of advantages associated with the establishment of a suitable framework such as the RBS early in the requirements analysis activity. First, the framework will act as a reference source during requirements analysis to ensure that all aspects of the system requirements are addressed and that important areas are not omitted. The framework allows multiple people to work on the analysis simultaneously as it facilitates the effective allocation to individuals of responsibility for sections of requirements. The framework assists in avoiding duplication of requirements in different sections, which is undesirable as it raises the probability of conflict between requirements and often leads to ambiguity and confusion.

It should be noted, of course, that not all complex systems can be represented adequately in a hierarchical decomposition and we must remain cognisant that we might need to extend the representation to describe fully the system to reflect its possible emergent properties. Still, almost all workable human-made systems are hierarchical, so the RBS is an adequate representation [9].

2.7.2 Functional Flow Block Diagrams (FFBDs)

The hierarchical representation of requirements in the RBS can be supported by FFBDs. While the RBS is very useful to show a hierarchical decomposition of the requirements, FFBDs are useful in showing the interrelationship of the functions to be performed by the system in accomplishing its mission. Some functions must be performed sequentially; some may be performed in parallel; sometimes alternative paths may be taken based on certain conditions—the FFBDs can show these relationships much more easily than the RBS.

As with the RBS, FFBDs are developed top-down through hierarchical decomposition. Each top-level block can be expanded (decomposed and derived) as necessary to ensure that the functionality of the system is adequately defined.

3
CONCEPTUAL DESIGN

3.1 INTRODUCTION

The systems engineering processes begin in earnest with the first activity in the Acquisition Phase: Conceptual Design (or logical design or problem-domain design), which aims to articulate the needs, to analyse and document the system-level requirements flowing from the needs, and to complete a logical design of the system. The major product of the Conceptual Design, the Initial *Functional Baseline (FBL)*, provides a system-level logical architecture that is the basis for subsequent lower-level (physical) design.

Conceptual Design is about the problem domain and is therefore normally the domain of the customer, who is responsible for and heavily involved in this activity. While other organizations may be involved, it is the customer who must determine what the system needs to do and how well it needs to do it—after all, it is their business from which the needs for the system arise.

Conceptual Design is perhaps the most critical of all of the life-cycle activities because:

- Conceptual Design is responsible for the expansion of the system definition from relatively brief business needs into a logical set of system-level requirements that may be hundreds of pages long.
- All subsequent aspects of the system design will be traced back to the FBL that ends this activity—any errors here will flow down to the remainder of the acquisition activities.
- Conceptual Design is concerned with the transition from the problem domain into the solution domain. This phase also often ends with the transition from the customer organization to a contractor—that is, as we will see later, the SyRS defined in Conceptual Design may form the basis of an agreement between the customer (the acquirer) and a developer. It is therefore essential that the output of Conceptual Design adequately represents the business and stakeholder needs and requirements.

Figure 3-1 illustrates that five major processes are conducted during Conceptual Design, including:

- definition of the BNR in the PLCD and the BRS;
- definition of SNR in the LCD and StRS;
- definition of system requirements in the draft SyRS;
- system synthesis to finalize the SyRS with the preferred solution in mind; and
- the System Design Review (SDR), at which the Initial FBL is established.

Note that, as with all systems engineering effort, the processes are iterative, even at the highest level of abstraction—at each stage the improved understanding of the system is used to revisit the previous process or processes if necessary. These Conceptual Design processes are described in more detail in this chapter.

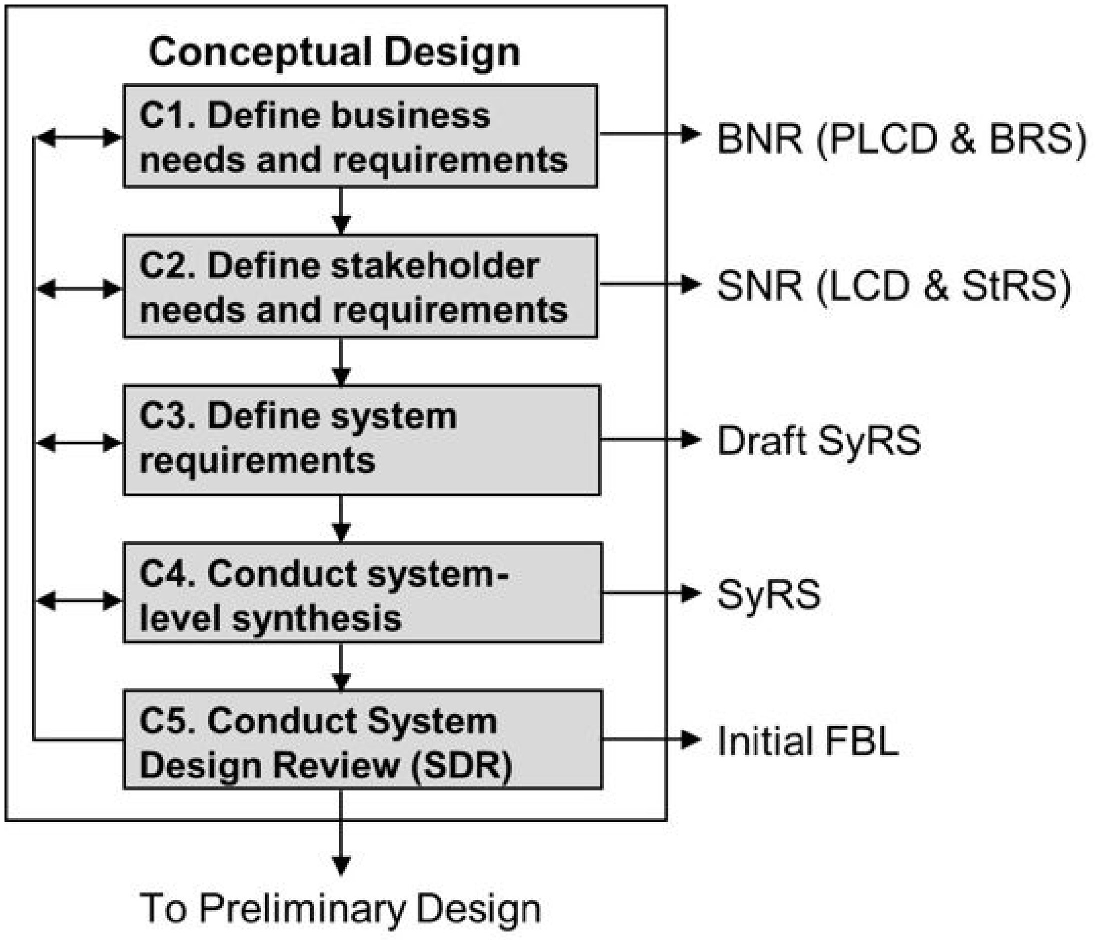

Conceptual Design
C1. Define business needs and requirements
BNR (PLCD & BRS)
C2. Define stakeholder needs and requirements
SNR (LCD & StRS)
C3. Define system requirements
Draft SyRS
C4. Conduct system-level synthesis
SyRS
C5. Conduct System Design Review (SDR)
Initial FBL
To Preliminary Design

Figure 3-1. The five major processes performed iteratively during Conceptual Design.

3.2 C1—DEFINE BUSINESS NEEDS AND REQUIREMENTS

Before any work can commence on developing the system, the basic BNR must be articulated clearly and completely by business management. It would appear obvious that an understanding of what is to be achieved must be clearly articulated by the business before any further work is undertaken. Yet, a surprising number of system developments commence without a clear and complete understanding of the fundamental business needs and requirements. Not surprisingly most of these developments founder or, at the very least, are introduced into service with poor levels of user acceptance.

3.2.1 Identify Major Stakeholders and Constraints

Business management begin the definition of BNR with the identification of the major stakeholders who are to be engaged to elicit business needs and requirements and to verify design artefacts as the design progresses. They also must identify the business, project, external, and design constraints that provide important context within which to consider the system requirements, as well as being the source of many requirements.

3.2.1.1 Identify Major Stakeholders

A stakeholder is commonly defined as someone who has a stake in the project—that is, someone who is affected by the system in some way, or can affect the system in some way. However, the identification of stakeholders cannot involve simply listing those who have, or perceive themselves to have, a stake in the project—in most systems this is not a useful definition since it is often difficult to find someone who is not affected by the system in some way. Even in a simple system such as an automatic teller machine (ATM) network for a bank, there may be millions of stakeholders by such a definition. This is also true in committee-based organizations such as public service organizations, where the number of potential stakeholders is almost limitless.

Business management must restrict stakeholders to the owners of the problem and of the solution presented by the new system. While definition of needs and requirements must take into account anyone who is affected by the new system, just because they are affected does not necessarily mean that they are stakeholders. For example, when considering the design of an ATM system, the armed guards who distribute the cash around the system are clearly important actors within the system—however, they are not stakeholders even though their requirements place considerable constraints on the design of the system. Similarly, there are a number of parties (such as the tellers whose employment may be threatened by the proliferation of ATMs and other competitor banks), are clearly affected by the system, but they are not necessarily stakeholders.

More usefully, a stakeholder could be defined as some individual (or some group or organization) who has a right to influence the outcome of the system, rather than someone who is simply affected by the system. Note, however, that a stakeholder is also invariably affected by the system.

3.2.1.2 Identify Business and Project Constraints

Constraints are requirements that are imposed on the system in some way. Before focusing on the detail of the desired system, it is therefore essential to identify the business and project constraints that are relevant to the system and its acquisition—a system will fail if it cannot operate under the constraints imposed upon it. This analysis provides essential information about the development environment for the system and begins the top-down approach to system development.

Business constraints include management guidance, organizational policies, procedures, standards, or guidelines that guide system development and procurement. These constraints can include partnering relationships with other companies, use of established life-cycle processes, contracting policies, human resource limitations, budget restrictions, and specific management guidance to the project. For example, an aircraft manufacturer may have a policy of buying engines only from a particular manufacturer so that through-life support issues are simplified across the entire fleet. This constraint must be articulated early because it has a significant effect on the acquisition and the design of the remainder of the system.

Project constraints include budget and schedule constraints, but also include the resource allocations within the project as well as any externally imposed deliverables and acquisition timeframes. Many companies have business-wide standards for processes such as quality assurance and systems engineering and these methodologies guide the manner in which projects can operate. Additionally, projects might be constrained to conform to particular engineering and technical standards; mandated toolsets; metrics; documentation sets and plan templates; technology use; and control and reporting mechanisms.

3.2.1.3 Identify External Constraints

External constraints on system development arise from the requirement for conformance to national and international laws and regulations, compliance with industry-wide standards, as well as ethical and legal considerations. Other external constraints include the requirement for interoperability and the capabilities required for interfacing with other systems. Additionally, the capability of competitors, as well as the availability of human resources, specific skill sets, technologies and tools might provide external constraints. Again, an important aspect of top-down design is to understand these constraints while considering needs and requirements.

3.2.1.4 Identify Design Constraints

Design constraints include those factors that directly affect the way in which the system design can be conducted. Typical constraints include the state-of-the-art of relevant technologies as well as extant methodologies and tools to assist in the design, development, construction and production of the system. Such issues must be addressed by business management who will address the risk associated with embracing new technologies, the impact of upgrading construction facilities, workforce re-skilling issues, and so on.

3.2.2 Elicit Business Needs

Having established the context by identifying major stakeholders and constraints, the next activity is to elicit stakeholder intentions. The major activities are the definition of the mission, goals and objectives; preliminary operational scenarios; preliminary validation criteria; and the preliminary life-cycle concepts to be documented in the PLCD.

3.2.2.1 Define Mission, Goals, and Objectives

It is axiomatic that every project should begin with a concise statement of the mission of the system, elaborated by statements of the upper-level goals and objectives for the system. The mission statement should be quite short (stated in a single sentence) and may be expressed in only a few lines, although it must have a word or phrase for every important aspect of the system. While business management often finds it difficult to state the mission in a single, short sentence, the project is invariably doomed to failure if the owners of the mission cannot articulate it succinctly at the outset. Interestingly, it will often take a small number of participants several hours to craft an agreed mission statement—the process is useful therefore, not only to start the design process, but also to serve as a mechanism for developing among stakeholders a common understanding of the system.

Once drafted, the mission statement is then expanded and qualified by short declarative statements of the system goals and objectives. Goals are normally relatively broad statements, each of which spawns a number of more-specific objectives (although in business planning these are sometimes treated in the reverse order and objectives are considered to lead to goals).

Once again it is clear that the mission statement performs an essential role since its contents must be sufficient to start the system design. While it is elaborated further by the high-level requirements that are gathered subsequently, the mission statement cannot be at odds with them, nor can it contain information which is not then elaborated on by at least one subsequent requirement.

Having satisfied ourselves that we have a draft mission statement that is a suitable start point for the design (recognising that we will invariably return later and modify the mission statement as we understand the system-of-interest better), we can now begin the process of identifying the requirements that are explicit in the draft mission statement (that is, the mission statement can be *decomposed* into those requirements) and those requirements that can be inferred (that is, *derived*) from it. We have now begun the elaboration process that continues throughout the remainder of the design.

We may again choose to confirm the draft mission, goals, and objectives by communicating them to the broader community of stakeholders, not just those immediately involved in their development. Once we have confirmed the artefacts with stakeholders, we finalise the mission, goals, and objectives into a formal set of statements that provide a complete, balanced description of the system, covering all aspects of operation, support, and maintenance. The set is the start point for the system design back to which every subsequent requirement must be traceable.

3.2.2.2 Define Preliminary Operational Scenarios

Once the mission, goals, and objectives have been articulated, business management in conjunction with stakeholders from business operations identify the range of operational scenarios proposed for the system. Scenarios help stakeholders explain how the system will be used. These scenarios, often expressed in terms of one or more *use cases*, provide valuable guidance to the system designers and also form the basis of major validation events in the Acquisition Phase such as testing of the system as it is introduced into service. Despite any more detailed technical verification and validation procedures, the system's fitness-for-purpose is fundamentally related to its ability to perform in accordance with the operational scenarios defined at this stage.

3.2.2.3 Define Preliminary Validation Criteria

An important top-level activity is the identification and definition of preliminary validation criteria. Broadly, validation criteria encompass any mechanism by which the customer will measure satisfaction with the products of the Acquisition Phase. Key criteria include performance in each operational scenario, safety, reliability, supportability, maintainability, ease of use, and time and cost to train.

3.2.2.4 Define Preliminary Life-cycle Concepts

Early in the Acquisition Phase, business management must give some guidance on the life-cycle concepts related to the acquisition, deployment, operation, support, and retirement of the system. While the systems engineering procedures that follow will ensure a life-cycle focus, it is important that there is an early business focus on the major cost drivers that will impact on all aspects of the system throughout its life. Additionally, there are a number of life-cycle-related trade-offs at a business-case level. Life-cycle concepts include the following.

Preliminary Operational Concept (OpsCon). The Preliminary OpsCon is written by business management and contains as much detail as they wish to communicate regarding (and sometimes constrain) the solution, which is to be refined by stakeholders at the business operations level.

Preliminary Acquisition Concept. Based on the operational needs for the proposed system, business management outline the proposed acquisition concept. Any business needs for acquisition are articulated, such as: budget and schedule, preferred (and perhaps prohibited) supply sources, types of contract desired, relevant existing contractual arrangements, any relevant existing acquisition support arrangements, program/project management considerations, reporting issues, and the relationship of the system-of-interest to other system acquisitions within the organization

Preliminary Deployment Concept. In this concept document, business management outline the issues surrounding the deployment of the system (to the extent that they are concerned with the introduction into service of this system). Additionally, the Preliminary Deployment Concept may address issues such as transition between facilities, training of operational personnel (and possibly engaging new staff and retiring others), and transition support arrangements.

Preliminary Support Concept. The Preliminary Support Concept will capture the needs of business management for the support of the system throughout its life cycle. Issues addressed would include relevant support policies and procedures; any relevant existing support arrangements; the support environment; desired maintenance levels and cycles; and the anticipated impact of the proposed system on support facilities, equipment, personnel, and training.

Preliminary Retirement Concept. When considering the retirement aspects of the system during its design, it is useful to undertake three broad tasks: identify the reasons for potential retirement, identify potential retirement methods for the system, which then allows conceptual designers to identify design issues that may arise from the consideration of each retirement method.

3.2.3 Scope System

The next phase in understanding business needs and requirements involves the development an understanding of the scope of the system development effort (called *scoping*, in the vernacular). This scoping activity helps to establish a clear understanding of what the system is expected to do by defining the system context, the system boundary, and any external interfaces.

3.2.3.1 Develop Context Diagram

To assist with the scoping process, a tool called a *context diagram* may be used to illustrate the related systems, relevant regulatory environments, major stakeholders, external systems, external interfaces, and so on. The context diagram is a very useful tool because, in this case, a picture is definitely worth a thousand words.

Figure 3-2 illustrates a simple context diagram for our domestic security alarm example.

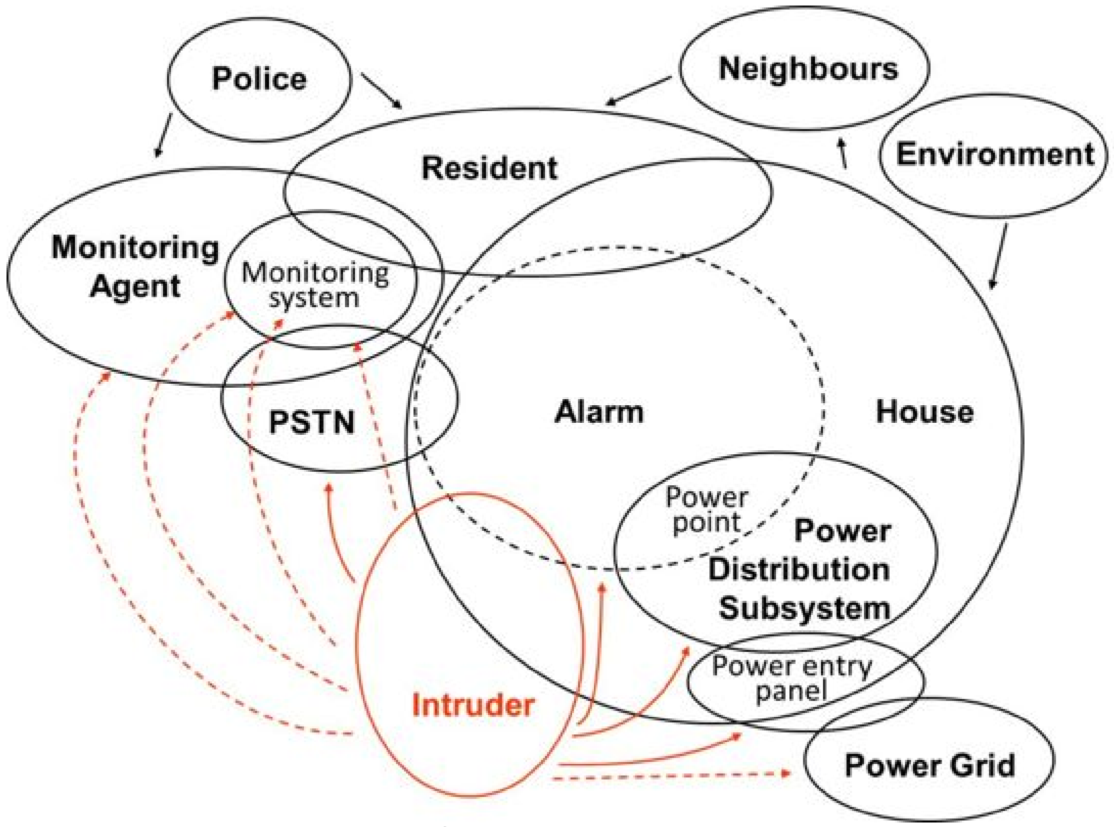


Figure 3-2. A simple context diagram for our domestic security alarm example.

3.2.3.2 Define System Boundary

Definition of the system boundary is essential early in the Acquisition Phase so that it is clear which elements are included in the system and which are outside. This activity also identifies which aspects are inside the systems acquisition and which are not, which is also particularly important to the project manager who is principally responsible for defining project scope—that is, what is to be included in the project as well as what is to be excluded. The boundary is also important in the next activity, which is to define the external interfaces—that is, those interfaces between the system of interest and to those other external systems to which it is interconnected.

3.2.3.3 Define External Interfaces

External interfaces are between the system of interest and each of the other existing or future external systems to which it is interconnected. The interfaces describe the inputs and the outputs of the system—the interfaces to its external environment. The specification, development, test, and management of these external interfaces will place considerable requirements on the system. While the external systems are not directly related to the project, the success of the fielded system is often dictated by its ability to interface to those systems in its external environment.

Figure 3-3 shows how the external interfaces might be enumerated on the context diagram for our domestic security alarm example. The nomenclature will be defined by the organization, but the example shows one simple method of listing each type of interface, from E01 to E06.

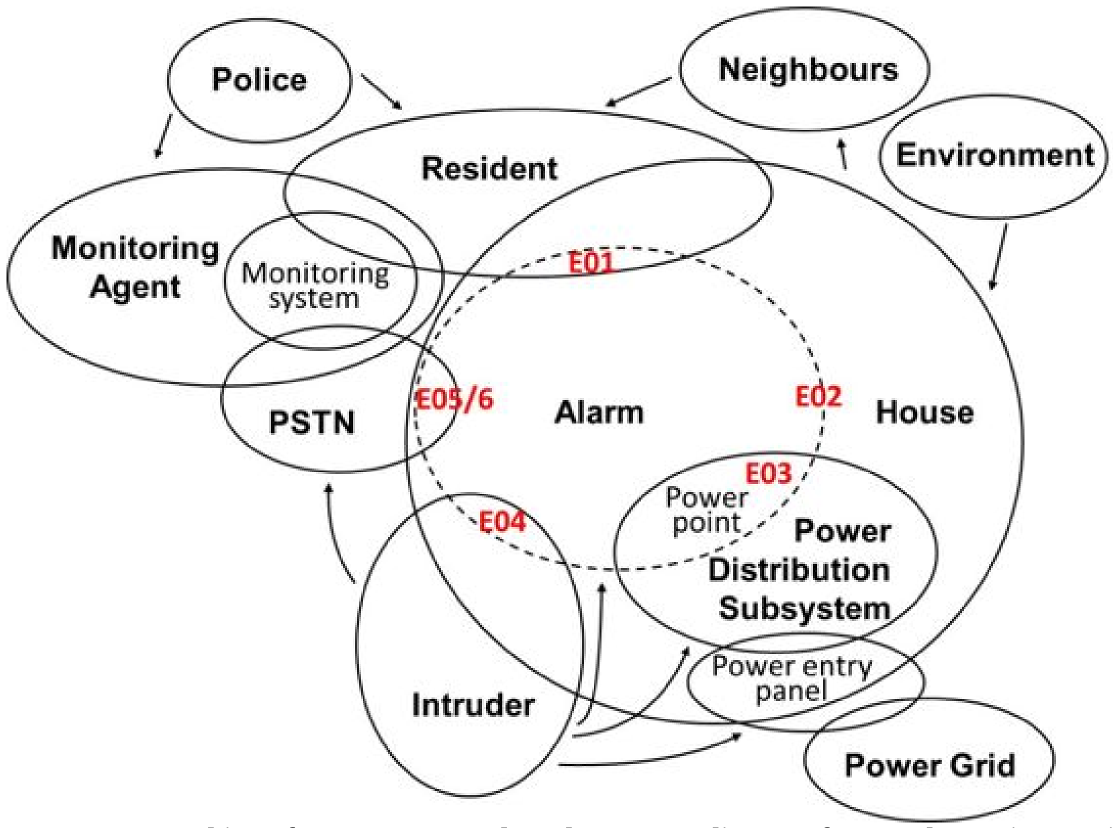


Figure 3-3. External interfaces enumerated on the context diagram for our domestic security alarm example.

3.2.4 Define Business Requirements

Once business management have defined their needs—the next major activities are related to the transformation of those needs into formal requirements.

3.2.4.1 Feasibility Analysis

At this stage of Conceptual Design, we have been careful to ensure that business management have stated their needs in logical terms—that is, in terms of the problem domain (the management and operational environment of the business). For any problem domain, however, there are normally a range of potential classes of solutions within the solution domain. Each alternative solution class may represent a completely different type of project—the feasibility analysis is therefore an essential step to narrowing the solution domain so that the subsequent project can be managed effectively. If we continue without narrowing down the design into one of those classes, we will have great difficulty coping with the breadth of the design. Consequently, we next undertake a feasibility analysis, the aim of which is to select a desired solution class (which is then to be elaborated further by the stakeholders at the business operations level).

The feasibility analysis narrows the solution domain in order to reduce the scope of the resultant requirements that are recorded in the BRS. It should also be noted that the narrowing of the solution classes to one preferred class is a business management decision, and not one that should normally be made by stakeholders at the business operations level.

3.2.4.2 Define Business Requirements (BRS)

The feasibility analysis and selection of the resultant desired solution class may require the draft business needs to be revisited in light of the more-detailed investigations of the feasibility analysis. The hierarchical representation of business needs (mission, goals, objectives) is now further elaborated (decomposed and derived) and formalised into a balanced set of business requirements, which are recorded in the BRS.

Development of the business requirements in the BRS can be assisted by a requirements framework around which the logical description of the system can be based. Here we call the requirements framework the *requirements breakdown structure* (RBS). The words are deliberately chosen to differentiate this structure from the well-known project management document called the *work breakdown structure* (WBS)—the RBS is grouped by function, the WBS is structured by physical work packages (including configuration items) and contains other project-related work.

In addition to being a useful requirements analysis tool, the RBS can be further refined to reflect stakeholder requirements in the StRS and then the system requirements in the SyRS. When completely populated, the RBS for a system provides an outline of the SyRS. A representative requirements framework for our domestic security alarm example is shown in Figure 3-4.

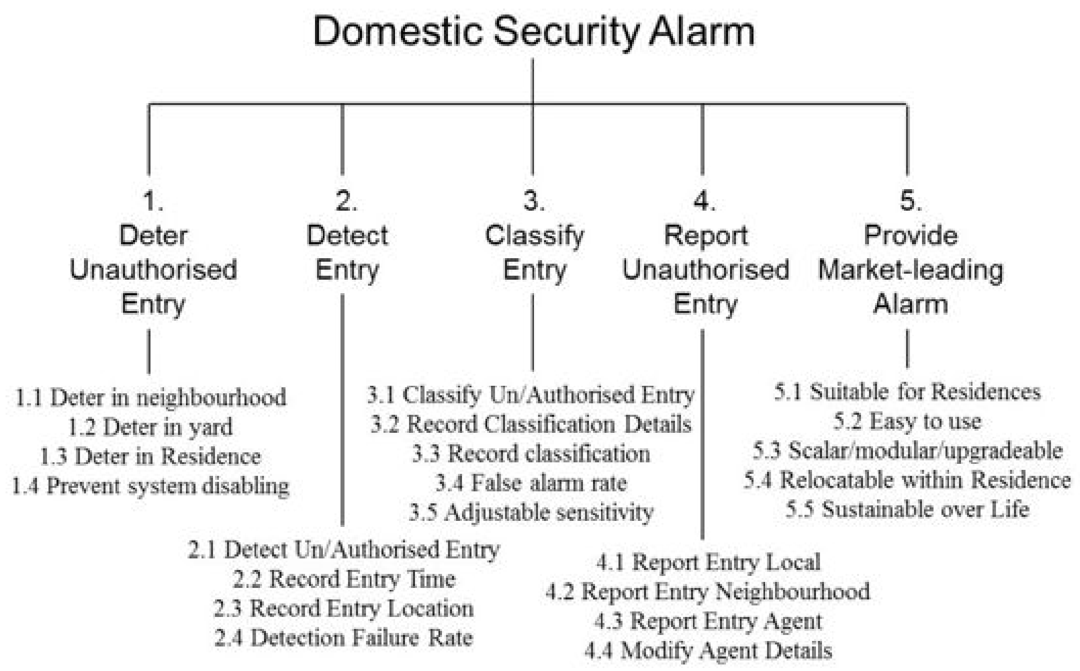

Domestic Security Alarm
1. Deter Unauthorised Entry
2. Detect Entry
3. Classify Entry
4. Report Unauthorised Entry
5. Provide Market-leading Alarm
1.1 Deter in neighbourhood
1.2 Deter in yard
1.3 Deter in Residence
1.4 Prevent system disabling
3.1 Classify Un/Authorised Entry
3.2 Record Classification Details
3.3 Record classification
3.4 False alarm rate
3.5 Adjustable sensitivity
5.1 Suitable for Residences
5.2 Easy to use
5.3 Scalar/modular/upgradeable
5.4 Relocatable within Residence
5.5 Sustainable over Life
2.1 Detect Un/Authorised Entry
2.2 Record Entry Time
2.3 Record Entry Location
2.4 Detection Failure Rate
4.1 Report Entry Local
4.2 Report Entry Neighbourhood
4.3 Report Entry Agent
4.4 Modify Agent Details

Figure 3-4. Example RBS for our domestic security alarm example.

3.2.5 Finalise Business Needs and Requirements (BNR)

At this stage, the business now understands the system much better than it did when it first drafted the mission statement—not only has much more detail been elicited but a solution class has been identified, which provides a much more concrete view of the system and its operation. Consequently, the BNR can be finalized by revisiting earlier artefacts and revising them to incorporate the increased understanding. The preliminary operational scenarios are revised; as are the PLCD; the solution scope in terms of the mission, goals, objectives, context diagram, boundary and external interfaces; and the preliminary validation criteria. Finally, the BNR are endorsed by business management, most likely at a major review.

3.3 C2—DEFINE STAKEHOLDER NEEDS AND REQUIREMENTS

The endorsement of the BNR signifies that business management are confident that they have defined the problem domain in sufficient detail to communicate their needs and requirements to the stakeholders at the business operations level. These stakeholders then go on to develop their needs and requirements (the SNR) within the context of the BNR.

The activities involved with the process of definition of SNR results in the development of the LCD and the StRS. Note once again that the process is iterative, allowing current understanding to improve the quality of previously generated artefacts.

3.3.1 Define Stakeholder Needs

The preliminary operational scenarios defined by business management are now defined in more detail, and other lower-level scenarios (often called vignettes) are developed as the design progresses. Similarly, the other PLCD are developed further and fleshed out with details from the various stakeholders to be more complete concepts in the LCD. Stakeholder needs are endorsed first by the stakeholders at the business operations level and then by business management. Again, endorsement is most likely achieved through a major review.

3.3.2 Define Stakeholders Requirements

Once the stakeholder needs have been endorsed, the process of transforming their needs into formal stakeholder requirements is undertaken. As noted earlier, the RBS is a useful framework for refining the BRS into the detail of the StRS while also assisting in the maintenance of traceability.

3.3.2.1 Trade Studies

In a similar manner to that which business management made use of a feasibility analysis to narrow down the alternative solution classes, stakeholders at the business operations level will have a number of alternative ways in which their needs can be met. The selection between various options is best undertaken by some form of formal trade (trade-off) study.

3.3.2.2 Define Stakeholder Requirements (StRS)

The development of the StRS is an essential first step towards a successful system development and is given a number of titles, including simply *stakeholder requirements* or *user requirements*. A reader of the StRS should be able to understand completely the likely applications or missions for which the system is intended; the major operational characteristics to be exhibited by the system; the operational constraints that limit the design and development of the system; the external systems and interfaces with which the system under development must operate; the operational and support environment within which the system must exist; and the support concept to be employed to support the system and enable it to continue performing in accordance with customer expectations.

3.4 C3—DEFINE SYSTEM REQUIREMENTS

The next process in Conceptual Design is the definition of system requirements, which starts with the endorsed SNR and then decomposes/derives them into system requirements. The aim of system requirements definition is to determine what the system must do in order to meet the stakeholder needs and requirements—that is, to describe the requirements at the system level and be able to relate the logical design back to the SNR, particularly the StRS. System requirements definition therefore uses elaboration to bridge from stakeholder requirements to system requirements.

During system requirements definition it is important to concentrate on what is required rather than how to do it. For example, a system-level performance requirement of the aircraft system might be a specified cruising speed. The physical subsystems that would be needed to perform this function (such as engines, airframes, and flight controls) may not be considered at this stage. The latter stages of the systems engineering processes focus on the physical subsystems (and lower) and how system-level requirements will be met best through a disciplined design approach.

Specifying how to meet the requirements at this embryonic stage may result in ill-informed design decisions leading to sub-optimal designs. Because the customer is most likely to be performing the Conceptual Design tasks, specifying “how” will also have the undesirable effect of shifting some responsibility and risk for achieving system performance from the contractor to the customer.

System requirements must be complete and must take into account all of the needs of the major stakeholders. The requirements should be objective, designable, measurable, verifiable (or testable), and traceable. In contrast, system requirements are often ambiguous and incomplete, conflict with other requirements, and are not traceable to any endorsed customer requirement.

3.4.1 Establish Requirements Framework

The first step in defining system requirements is to establish a requirements framework around which the logical design of the system is to be based. As noted earlier, the RBS is a useful framework for refining the StRS into the detail of the SyRS while also assisting in traceability. Since we have started with a hierarchical decomposition of mission, goals, and objectives, we can use that information to begin our development of a logical hierarchy, which can be captured in an RBS.

3.4.2 Perform Requirements Analysis and Allocation

This activity involves the elaboration of major requirements for the system (with associated performance, verification and rationale) including operations, maintenance, support, and other necessary functions. As noted earlier, requirements analysis is the bridge from stakeholder requirements in the StRS to the system requirements in the SyRS.

3.4.2.1 Define Functional/Non-functional Requirements

The central step in requirements definition (and in Conceptual Design, for that matter) is the definition of the requirements (functional, non-functional, constraints, and interfaces) for the system. While users are understandably focused on operational requirements, maintenance and support requirements also need to be defined during Conceptual Design.

3.4.2.2 Define Performance Requirements

Once the functions have been identified and grouped according to the agreed RBS, design focuses on the performance-related parameters that the new system must achieve. Having decided what the system must do, for example, the designer must now determine how well the system is to perform each of those requirements. A good discipline is to ensure that, every time a requirement is articulated, at least one corresponding performance statement is made.

3.4.2.3 Define Verification Requirements

The definition of requirements is not complete until verification requirements are also included. It is good practice to ensure that, every time a requirement is articulated, a corresponding verification statement is made. It is often difficult to write verification requirements for system-level functions, but the discipline of doing so is important since there is little point in stating a requirement for a function or constraint without consideration of how the function is to be verified.

3.4.2.4 Assign Rationale

It is also useful to record the rationale behind each requirement, which will further assist with the removal of ambiguity but more importantly will assist in situations where more than one person is involved in the requirements analysis over the course of the project. The rationale explains why each requirement is necessary as well as the logic behind performance levels assigned to the requirement.

3.4.2.5 Analysis and Allocation

The requirements articulated by the stakeholder are generally at a high level and often tend to be more qualitative than quantitative. These broader functions need to be *analyzed* to identify the lower-level requirements necessary to achieve the parent requirement. In addition to the derivation of requirements during requirements analysis and allocation, system designers also *group* and *allocate* requirements. Requirements are allocated to the appropriate group of the RBS.

3.4.3 Draft System Requirement Specification (SyRS)

The complete and populated RBS forms the framework for the SyRS which, when approved, becomes the centrepiece of the system's FBL. At this stage, however, the populated RBS only forms the basis for the *draft* SyRS because the requirements collected and analysed so far have yet to be synthesized into an architectural solution. There will therefore be some degree of latitude allowed in certain requirements and priority labels (such as mandatory, important, and desirable) leave some flexibility for subsequent design.

The SyRS may take any one of many forms and the most appropriate form depends on the application. It is safe to assume, however, that the SyRS should contain at least the information described in the augmented RBS that has been established at SDR.

3.4.4 Define Technical Performance Measures (TPM)

Some of the requirements from the set determined in the preceding activities may be considered to be useful as key indicators of system performance (or may represent particular risk)—a sort of 'health check'.

These selected requirements are called *technical performance measures* (TPMs).

3.4.5 Conduct System Requirements Reviews (SRR)

System Requirements Reviews (SRR) may be conducted periodically throughout Conceptual Design to verify and approve versions of system-level requirements. The aim of the SRRs is to monitor and approve progressively the system-level requirements that are developed on the way to the Initial FBL. Progressive reviews allow the requirements analysis effort to continue to lower levels in the logical hierarchy in the RBS by providing validation of the higher levels of abstraction, providing a firm start point for the subsequent analysis. SRRs may or may not be considered formal reviews. While there is a natural flow to the activities, iteration is required.

3.5 C4—CONDUCT SYSTEM-LEVEL SYNTHESIS

The system design has now progressed to the stage where some of the system-level design decisions can be made. System requirements analysis has identified requirements. Synthesis (at the Conceptual Design level) establishes a system configuration that is representative of the final system form. The configuration established at this stage is not assumed to be final as the design is very immature and may go through significant changes later in the design process.

Based on the results of the requirements engineering and analysis, a range of architectural options is developed that represent potential system solutions. The selection of one of these options as the preferred solution requires information to be collected from the systems engineering process, life-cycle costings, quality assurance, test and evaluation, maintenance, integrated logistics support, and so on. The potential solutions are then evaluated, which requires the development of suitable evaluation criteria as well as an evaluation framework. Any discrepancies in the systems engineering products are noted) and fed back to previous activities. The preferred solution is then chosen.

The draft SyRS must to be refined to remove the ranges of acceptable performance and to reflect exactly what is being offered by the preferred system solution and the minimum acceptable level of performance. The main products of synthesis are the refined SyRS and a broad system solution for achieving the specification requirements.

The SyRS is perhaps the most important of all systems engineering documents because it becomes the source of reference for all the subordinate specifications that are produced during later stages of the design process. If the SyRS is solid, it forms an effective foundation for the remainder of the design and development effort. Errors or omissions in the SyRS flow into the remaining design effort. The later these errors are discovered, the more expensive and time-consuming the rectification will be.

3.6 C5—CONDUCT SYSTEM DESIGN REVIEW (SDR)

At the end of Conceptual Design the *System Design Review* (SDR) provides the following from a systems engineering perspective:

- formal confirmation that the logical design meets the business and stakeholder requirements;
- a formal record of design decisions and acceptance;
- a formalized communication of the intended design approach to the major players in the design effort; and
- approval of the V&V plans for the system.

The SyRS is approved and baselined as part of the Initial FBL. It is rare that SDR ends with all outstanding issues resolved during the review since many will take some time to resolve. Rather than hold up subsequent design, action items are agreed with agreed timelines to account for any outstanding action from the review (providing, of course, that resolution of those issues will not affect the remainder of the design). These actions are completed in parallel with the early Preliminary Design activities and are reviewed for completeness in conjunction with a later review or audit.

4
PRELIMINARY DESIGN

4.1 INTRODUCTION

Preliminary Design starts with the Initial FBL—as defined during Conceptual Design—and continues to translate system-level requirements into design requirements for the system elements that will combine to form the system. This translation requires a continuation of the requirements analysis started in Conceptual Design. Specific requirements for the elements making up the system need to be determined. Trade-off studies are conducted and the result of the Preliminary Design effort is the establishment of an *Allocated Baseline* (ABL), in which requirements are 'allocated' to specific physical system elements that combine to form the system.

Responsibility for performing Preliminary Design normally rests with the developer (the contractor), who develops the system to meet the requirements of the FBL (normally prepared by the customer). The customer's role now increasingly becomes one of monitoring, reviewing and supporting contractor progress. Although the customer normally avoids becoming actively involved in design decisions made during Preliminary Design (or any subsequent activity, for that matter), they remain very interested in the outcomes at each point. In some cases, the customer may opt to add an additional level of rigour to the process by engaging independent systems engineering consultants to provide independent review of the many and varied engineering documents and products produced by contractors during the course of a systems development—often called independent verification and validation (IV&V).

The activities conducted during the Preliminary Design effort include subsystem requirements analysis; requirements allocation; interface identification/design; subsystem-level synthesis; and the Preliminary Design Review (PDR). Subsystem Requirements Analysis

In the preceding chapters, we described requirements analysis as the process through which requirements are progressively elaborated (decomposed and derived) from the BNR to the system level. In

Preliminary Design we are now interested in the elaboration from the system level to the subsystem level and so on until all functions, parameters and interfaces have been defined and the necessary resources have been identified.

The next step is to begin defining requirements at the subsystem level by decomposing/deriving requirements from the SyRS to provide more-detailed requirement statements that can be allocated to system elements. Performance and verification requirements are then assigned, as is a rationale for each requirement.

It is not possible to prescribe the level of detail that needs to be decomposed/derived during subsystem requirements analysis. The detail must be sufficient to define completely the functions required of the major subsystems comprising the system, and must be able to be used either to procure, or to design and produce, the major subsystems and their assemblies and components during subsequent stages of the systems engineering process.

4.2 REQUIREMENTS ALLOCATION

Requirements allocation refers to the process of grouping or combining similar requirements (identified during the requirements analysis) into logical subdivisions. We have done this before during Conceptual Design where the groups were based on the RBS headings. During Preliminary Design, the groups are based around a preliminary physical architecture formulated by the designers, rather than logical RBS headings. The groups of similar functions then assist the designer to determine the design of the major system elements that are required to make up the system. This represents the translation from logical design to physical design. These elements are selected and/or designed to perform the group of requirements that are assigned to them.

4.3 INTERFACE IDENTIFICATION AND DESIGN

During the selection of the elements comprising the system, the interfaces between the elements are identified. Identification of interfaces is a critical part of Preliminary Design because interfaces not only determine successful operation of the system once integrated, but they also place additional limitations and requirements on the design of the individual elements. There are a number of types of interface.

Physical interfaces. Physical interfaces are normally the most obvious of interfaces because they need to exist in real terms. Examples include pipes through which a fluid or gas flows, fibre-optic cables passing digital information, or a structured interface such as a rope or a chain.

Electronic interfaces. An electronic interface is the name given to the flow of electronic signals (analogue or digital) between two points. These signals may flow over a physical interface such as a cable or fibre, or via a wireless connection.

Electrical interfaces. Electrical interfaces are normally associated with a physical interface of some sort such as a power cable or data cable. The type of information required in this sort of interface will include voltage levels, voltage type (for example 110V, 60 Hz), and any associated factors such as fault tolerance requirements.

Hydraulic/pneumatic interfaces. Hydraulic and pneumatic interfaces are the mechanical equivalent of the electronic or electrical interfaces already described. A physical interface definition will coexist with this type of interface. Additional information such as flow rate, pressure, temperature, and fluid type will be required to define completely the hydraulic and pneumatic interfaces.

Software interfaces. Software interfaces refer to passing of data between different computer software. This may occur within the one piece of hardware using internal hardware buses or it may be between two different pieces of hardware.

4.4 SUBSYSTEM-LEVEL SYNTHESIS AND EVALUATION

Once the allocation of requirements has been performed, it is time to continue the familiar loop of analysis, synthesis and evaluation and concentrate on synthesis. Subsystem-level synthesis results in a Preliminary Design that meets the subsystem-level requirements.

4.4.1 Investigate Design Alternatives

The initial synthesis is conducted to determine a preliminary design that satisfies the specification using one of the three broad design options available. There are three broad design options available including commercial-off-the-shelf (COTS), modified COTS, and developmental items. Factors that impact on the designer's decision to use COTS, modified COTS or specially developed items include the specific requirements to be performed by the item, the availability and stability of the current technology, size of the market, supportability and cost, and any contractual directives. The following sections discuss each of these options.

4.4.1.1 Commercial-off-the-shelf (COTS)

COTS equipment, as the name suggests, refers to pieces of equipment that are commercially available. Military-off-the-shelf (MOTS) equipment is another category of off-the-shelf equipment that has been included in the category of COTS for the purposes of this discussion.

If the COTS equipment meets or exceeds the requirements detailed in the Development Specifications, there may be some considerable advantages to using COTS equipment.

- The most obvious advantage is that the equipment is likely to be readily available with limited or no delay.
- Technical risk may be reduced since the COTS product is a known quantity whose performance can be readily verified.
- A COTS option is most likely cheaper than the alternative (on the assumption that the cost of design and development has been amortised over a large number of items).
- If the COTS item has a large user base, maintenance and support will probably be already in place for the item. If not, at least data

may be available to facilitate the planning of maintenance and support (to facilitate the quantification of spares holdings, for example).

- The COTS item may already be validated in the operational environment, saving considerable costs in validation and verification.

The disadvantages often associated with COTS items include:

- The COTS product may not be perfectly suitable.
- The COTS item may be based on out-dated technology and may soon face technical obsolescence—if the item has been in the market place for several years the technology incorporated will have been developed several years before that.
- Use or support of the COTS product may be constrained by warranty, ownership, or intellectual property issues.
- COTS products may come with very limited documentation.
- The COTS product may come with additional functionality which may be undesired, or may be unknown and represent a security risk.
- Support of COTS will most likely need to be undertaken through the supplier—any repair/modification by the customer may void warranty.

4.4.1.2 Modified COTS

COTS equipment that meets a majority of requirements contained in the Development Specification for a system element may be able to be modified to better suit the needs of the designer. Similar advantages will exist with modified COTS as for COTS, however there are the following additional disadvantages:

- Maintenance and support may be voided if the COTS item is modified.
- The effort involved in modifying the COTS item can be easily underestimated and the modification process can sometimes take much longer and cost much more than initially estimated.

4.4.1.3 Developmental Items

If suitable COTS equipment is not available, the designer may opt to design and develop the item ‘from the ground up’ to meet the specific requirements and characteristics detailed in the relevant Development Specifications. Developmental items have the following advantages:

- An item developed from the ground up is likely to match the desired criteria precisely in terms of form, fit and function.
- All aspects of the developed items will be understood (from a security perspective, for example, we will have much more confidence in software written by our programmers).

The following disadvantages may result from a developmental approach:

- The effort involved in this development will not be insignificant.
- The desired advantages may never accrue due to the increased technical risk of developing a novel item.
- Maintenance and support issues will also need to be developed and deployed when items have been designed specifically for the system.

4.4.2 Make Optimal Use of Design Space

During the selection of the different system elements making up the system, it is important to remember that it is the system performance that is of vital importance, not the performance of the individual elements and components. That is, in order to achieve optimal system performance, the performance of the elements may need to be constrained and balanced (that is, have sub-optimal performance). In many cases, none of the elements may perform as well as they might be able to so that their combined performance (the performance of the system) is optimised.

4.5 PRELIMINARY DESIGN REVIEW (PDR)

The Preliminary Design Review (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. At the conclusion of a successful PDR, the participants should be confident that all of the FBL requirements are being addressed adequately by the design, and that the logical and physical interfaces that need to be in place between the elements and between the elements and the external environment have been identified and documented ready for detailed design.

5
DETAILED DESIGN AND DEVELOPMENT

5.1 INTRODUCTION

The Detailed Design and Development activity continues the development effort and makes use of the FBL and ABL developed during Conceptual Design and Preliminary Design. The detailed design effort takes these definitions of the overall system (as contained in the FBL) and of the major system elements (contained in the ABL) and finalizes the design of specific components that make up the CIs (and subsystems). The realization and documentation of individual components used to support production is referred to as the *Product Baseline (PBL)*.

A number of tasks are undertaken during Detailed Design and Development. The major technical activities include:

- describing the lower-level assemblies and components making up the CIs (and their interrelationships);
- defining the characteristics of the above items through specifications and design data;
- finalizing the design of all interfaces necessary to support system integration;
- either procuring the above items off-the-shelf, or designing them if they are unique to the system under development;
- developing prototypes or engineering models of the CIs (or even the final system) by integrating the above items (for the purpose of design verification, and system-level test and evaluation);
- redesigning and retesting (as required); and
- conducting the Critical Design Review (CDR) to confirm that the design is ready for construction and production.

5.2 DETAILED DESIGN OF HARDWARE

It is not possible to cover all of the categories of hardware associated with complex system developments. It may be useful to consider what the term may mean to engineers from different engineering disciplines as follows:

- *Electrical/electronic engineer.* Hardware may be an electrical or electronic system of some description that relies on a combination of electronic components arranged in such a way as to manage, condition, or distribute electrical power or signals.
- *Civil engineer.* Hardware may be the foundations and structural components of a bridge or other piece of physical infrastructure.
- *Mechanical engineer.* Hardware may be a physical system such as an engine, chassis, or drive train.
- *Aeronautical engineer.* Hardware may be the structures and components associated with the wings and fuselage of an aircraft.
- *Software engineer.* Hardware may be considered to be anything that is not software including computer processors, network infrastructure, and human-machine interfaces.

5.3 DETAILED DESIGN OF SOFTWARE

In parallel with the effort associated with determining, proving and documenting the hardware, software engineers will be completing their own detailed design process for the software elements of the design. Software development process standards exist to guide software developers in their quest to design successful software. Software development methodologies are an area of rapid change, significant research and publication, and heated debate. There appears to be supporters and detractors associated with just about every software development methodology or strategy. Software development, like systems engineering, does not have a ‘one size fits all’ solution. The characteristics of the system within which the software is to be developed should help software engineers to choose the most appropriate methodology for the situation.

5.4 INTEGRATING SYSTEM ELEMENTS

Detailed Design has now reached a stage where all system elements (hardware and software) have been designed completely, as have the interfaces between the elements and between the elements and external systems. In the case of both software and hardware, the design teams have proven their design by releasing elements of the design and testing their design against the requirements. This process may have resulted in a series of redesigns and tests, but the result is a series of designs that have been completed. In many cases, subsystem and system-level design will need to be confirmed by a series of integration and test stages. Low-level items of the design (although individually tested and confirmed) may need to be integrated together to form the next higher level of assembly in the system hierarchy and tested against the relevant requirements documents.

5.5 DETAILED DESIGN REVIEWS

As with each of the preceding design phases, a design review (or a series of reviews) is normally conducted at the conclusion of Detailed Design and Development to ensure that the system design is complete prior to Construction and/or Production.

A number of low-level design reviews may be conducted throughout detailed design. These design reviews, sometimes called equipment or software design reviews, are focused on particular items of software and hardware to ensure that the specific design approach meets the requirements. All equipment/software design reviews will be completed prior to the major formal review of the phase, which is called the *Critical Design Review* (CDR).

CDR is the final design review resulting in the official acceptance of the design and the subsequent commencement of Construction and/or Production activities. The result of the successful completion of CDR is the establishment of the Product Baseline, which is the effective freezing of all design activity. Only discrepancies in the system design identified during testing result in further design activity following CDR.

6
CONSTRUCTION AND/OR PRODUCTION

6.1 INTRODUCTION

At the end of Detailed Design and Development, the PBL has been established and the production process is in place and has been proven (most likely with the trial production of selected system elements). The system can now move into Construction and/or Production.

Some system development projects are aimed at producing only one copy of the system. There are many examples of one-off system developments including high-rise buildings and large ships such as cruise liners. In these projects, the construction process may blend with the detailed design and integration process. For example, when building a single ship, a prototype may be developed to support detailed design, development and integration. This prototype may pass through stages of test and evaluation during which the design is refined. Ultimately, however, the design and development will be finished by which time the 'prototype' may have matured into the final system that is offered to the customer for acceptance.

With other system developments, many copies of the system may be produced once one or two prototypes have been proven. An aircraft system is an example of a system where many copies (aircraft) are likely to be produced after an initial aircraft (or even a small batch of aircraft) has been produced as prototypes to support design verification and user validation before the production is allowed to proceed.

6.2 PRODUCTION REQUIREMENTS

Production requirements need to be considered early in the Acquisition Phase to ensure that production risks are identified and addressed as early as possible. This is particularly true when there are one or more of the system elements that make use of novel technologies (with which existing production processes may be unfamiliar). As with all other technical requirements, the earlier that production issues are identified in the acquisition, the easier they can be addressed. To that end, although Construction and/or Production is being considered following detailed design, production engineers should be working with the rest of the design team from the earliest possible stages of the system development to ensure that production and construction issues are appropriately addressed. In short, the systems engineering effort must make sure that the design that flows out of the preceding stages is able to be produced.

Typical construction and production issues that need to be addressed and monitored throughout the entire systems engineering process include:

- material availability (lead times), ordering, and handling;
- availability of skill sets (including any training);
- availability of production tools and equipment;
- fabrication requirements including production requirements, assembly drawings and instructions;
- processing and process control;
- assembly, inspection and test; and
- packaging, storage, and handling.

6.3 ENGINEERING MANAGEMENT ISSUES

During the production and construction stage, the system will be subjected to a number of important systems engineering management activities. These include test and evaluation (T&E) activities, final reviews, and configuration audits.

Before the system completes Construction and/or Production, it should be subjected to a range of acceptance tests aimed at confirming the ability of the system to satisfy the original user requirements that started the whole systems engineering process.

Acceptance testing, alone, is insufficient. Audits should be conducted during production and/or construction to ensure that the system has been built in accordance with the approved specifications and project documentation—these are called generically *configuration audits*. The Construction and/or Production stage is often the only opportunity to conduct these audits.

There are two main types of configuration audit—the *functional configuration audit* (FCA) and the *physical configuration audit* (PCA) [10]:

- FCA is used to verify and certify that the performance of the system element meets the specified requirements. That is, it is a check to ensure that the final as-built element functionality as demonstrated by the test and evaluation effort is the same as the specified functionality for that element in the relevant documentation.
- The objective of the PCA is to provide confidence that the as-built system elements match the low-level specifications such as the Product Specifications, assembly specifications, drawings and technical data. PCAs are conducted on the final versions of the elements (that is, the version of the elements representative of the as-built items).

7
OPERATIONAL USE AND SYSTEM SUPPORT

Once the system has passed the necessary testing and audits, it is ready to enter operational service or use—the Utilization Phase. The major activities during the Utilization Phase include operational use, system support and modifications. The influence of systems engineering over these activities is relatively minor and is normally confined to modifications, which should be made to the system in accordance with sound configuration management procedures.

Configuration management continues to play a role during the Utilization Phase to ensure that the configuration of the system is managed and updated, even if modifications occur. Differences between system documentation and the physical system tend to result when systems are operated and maintained in an environment void of configuration management practices. These differences make maintenance and operation difficult and potentially dangerous. Accurate documentation supports the need to understand the design and capability of the system especially if there are slight differences in configuration across a fleet of systems (for example, an aircraft fleet). Modifications made in accordance with configuration management practices enhance supportability and operations, but unmanaged modifications adversely impact training, support, safety and operations in the long run.

Modifications may be required to ensure that the system continues to meet operational and support requirements. Modifications may be made for a number of reasons:

- Modifications may be required to rectify discrepancies with the performance of the system that were not identified during the Acquisition Phase. These discrepancies are often discovered during the OT&E effort when the system is placed in its operational environment and exercised in its intended purpose.

- Failures identified as part of the FRACAS process described in the next section may result in engineering changes to the system via the modification process.
- Modifications may also become necessary due to changing system-level requirements caused by a range of factors including a changing operational or support environment.
- Opportunities may arise to increase the efficiency of the system, or perhaps reduce maintenance costs, through the replacement of system elements with improved designs or through the insertion of new technologies.

The Retirement Phase is the final stage in the system life cycle. Functions associated with phase-out and disposal include transportation and handling, decomposition, and processing of the retiring system. A Retirement Concept should be developed as one of the LCD during the early stages of the Acquisition Phase of a system. 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.

There can be any number of reasons why the system may require retirement.

- The system may have reached the end of its life and no longer be usable and/or supportable.
- The system may still largely be useful, but one or more critical components may be unusable or unsupportable.
- The system may still be perfectly useful, but it (or a significant subsystem of it) is being retired because:
  - the business need for the system may have disappeared (for example, the owner's business direction may change, or the owner may be retiring and winding up the business); or
  - the business need is still valid but the business owner is forced to retire the system (due to issues such as cash flow, the operating license is revoked, scandal, public pressure, or disputes between business partners).

- The system (or one of its critical components) may be damaged beyond economical repair (by accident, natural disaster, act of war, or vandalism).

Once the reasons for potential retirement have been identified, each can be examined for potential retirement methods for the system (although not all may apply to all reasons for retirement):

- Sold as a second-hand item.
- Traded in on a replacement system.
- Re-deployed as a training aid.
- Destroyed and disposed as waste.
- Disassembled—that is, broken up and sold as working sub-systems, assemblies, or components.
- Scrapped—that is, broken up and sold as sub-systems, assemblies, or components for their value as scrap.
- Placed in storage—that is, mothballed because none of the other methods are currently acceptable or tenable.

Once the reasons for potential retirement and the potential retirement methods have been identified with the stakeholders, designers can identify any design issues that may arise, such as:

- observance of recycling regulations;
- avoidance of the use of potentially hazardous/toxic materials;
- environmental impact;
- cost of disposal/destruction/uninstallation;
- cost of refurbishment for resale or trade-in;
- desirability of salvage or material recovery for resale;
- cost of transportation for disposal;
- need for specialized personnel/equipment required for disposal;
- observance of any caveats on disposal (such as might be placed on military equipment by international arms trade agreements, or might be placed on high technology by national sanctions);
- time taken to arrange disposal; and
- availability of design data and drawings to support disposal.

8
SYSTEMS ENGINEERING MANAGEMENT

8.1 INTRODUCTION

Systems engineering management is responsible for planning and directing the systems engineering effort, monitoring and reporting on that effort to the appropriate areas, and reviewing and auditing the effort at critical stages in the system life cycle. In this chapter we consider the major systems engineering management elements of technical review and audit management, test and evaluation, technical risk management, configuration management, the use of specifications and standards, and systems engineering management planning.

8.2 TECHNICAL REVIEW AND AUDIT MANAGEMENT

As we have discussed in the preceding chapters, the technical review and audit considerations of a project can be quite extensive and the management of these issues helps ensure that the overall aim of reducing technical risks and enhancing confidence in the design process is achieved. In the previous chapters we have considered the major reviews and audits in each stage of the Acquisition Phase. This section discusses the management considerations of technical reviews and audits.

Technical reviews and audits provide both the customer and contractor with a measure of progress toward the goal of successfully introducing a system into service, and reduce the technical risks associated with the system development. Reviews and audits achieve this by:

- providing a formal evaluation of the design maturity,
- measuring and reporting on planned and actual performance,
- clarifying and prioritizing design requirements,
- evaluating and establishing the system baselines at discrete points in the design process,
- providing an effective means of formal communications between the stakeholders, and
- recording design decisions and rationales for later reference.

Obviously, the number of reviews required and their respective scope depends on the complexity and size of the system in question—so the reviews and audits must be tailored accordingly. The technical risk associated with the system also has an impact on the number of reviews scheduled.

It is clear that reviews and audits can provide significant assurance of the technical progress of the system development. However, these reviews and audits also take considerable time and effort to plan, prepare and conduct. While reviews and audits are being conducted, attention is being diverted from the design and development effort. It is important that the costs and benefits associated with conducting technical reviews and audits are considered fully prior to planning and mandating the reviews and audits to ensure that an appropriate level of review and audit is placed on the system development effort.

The scheduling of reviews and audits is also very important. Reviews held too early in the design process are unable to determine technical adequacy of the design because the design is immature and subject to change. Reviews held too late in the process may miss the opportunity to avoid costly and time-consuming rectifications if the design proves inadequate

As well as specifying the reviews and audits required, the contract should also specify how the reviews and audits are to be conducted. The requirement for chair person, agenda, data packages, and minute-taking will help in making the reviews and audits as effective as possible.

Decisions, action items and agreements should be noted in the minutes that are distributed following the meeting. Each action item should also be assigned to an individual who is responsible for following the action through to fruition by an agreed date. Unassigned action items invariably remain unactioned.

At the conclusion of the review, the customer team should meet to determine the level of satisfaction with the meeting. Successful technical reviews are a good indication of project status and schedule. Significant milestone payments are often tied to the successful completion of design reviews. Technical reviews and audits are, therefore, an important part of both the systems engineering and the project management effort.

8.3 VERIFICATION AND VERIFICATION

The entire systems engineering process aims to produce a system that is both verified against the documentation produced during the systems engineering process, and validated against the original needs, goals and objectives that initiated the system development in the first case. Often these two associated aims are combined into the term *verification and validation* (V&V). V&V ensures not only that we have 'built the system right' (verify) but we have also 'built the right system' (validate). A well-managed approach to test and evaluation aims to support the delivery of a system that is both verified and validated. Other systems engineering management functions including technical reviews and configuration audits also support the overall V&V objective.

There are three major categories of T&E that are applied to coincide roughly with the Acquisition Phase, the transition between the Acquisition and Utilization Phases, and the Utilization Phase.

- *Developmental test and evaluation (DT&E).* DT&E refers to the T&E activities undertaken during the Acquisition Phase of the system life cycle to support the design and development effort. DT&E activities may also occur during the Utilization Phase to support activities such as modification development.
- *Acceptance test and evaluation (AT&E).* As DT&E completion approaches, AT&E activities become increasingly relevant. AT&E represents the formal acceptance testing conducted on the system to enable the customer to accept the system from the contractor. AT&E effectively forms the boundary or transition between the Acquisition Phase and the Utilization Phase. Unlike DT&E and OT&E, AT&E tends to be a discrete testing activity (with a defined start and a defined end).
- *Operational test and evaluation (OT&E).* OT&E is the term sometimes associated with the T&E effort that is focused on the functional or operational testing of the system and its components, conducted under realistic operational conditions by operational personnel. OT&E is normally conducted for a period of time following acceptance of the system by the customer, although limited OT&E activities are possible during acquisition especially

where a long production cycle means that some systems have been accepted prior to other systems being produced, or when concept demonstrators are being used.

8.4 TECHNICAL RISK MANAGEMENT

Risk is defined as the possibility of a loss or injury, or the possibility of some disadvantage or destruction. Systems engineering management is concerned with the management of technical risk—that is, the risk associated with the technical aspects of the system life cycle. Broadly speaking, there are two major categories of risk; internal risk and external risk. Internal risks are those that are within the control of the customer's organisation; external risks are beyond the control of the project office. An example of an internal technical risk may be the risks associated with inappropriately staffing the project office with qualified technical staff. This risk can normally be managed from within the project office. An external risk may be the introduction of some new legislation that places more stringent requirements on the system, leading to some changes to the user requirements.

The concept of risk is subjective in that risk identification and assessment is based on the perception of an individual and their interpretation of the term risk. What one individual may find risky, another individual may not. To that end, risk management must be performed by personnel with a broad and experienced knowledge of the subject area to ensure that risks are assessed realistically. Given the extremely broad nature of risk, it is unlikely that a single person will be able to perform the entire risk management function, nor is that recommended—risk management is a team effort.

ISO 31000 [11] identifies that the broad risk management activities are:

- *Establish the context.* The first step in the risk management process involves understanding the strategic, organizational and risk management context within which the risk management is to be conducted. It is very important that risk management is not conducted without understanding the context—for example, the context for the analysis of the risk of injury to employees in the workplace is a very different issue for a warehouse than it is for a football franchise, or a marine battalion. This step also involves establishing the criteria against which risks will be evaluated, as well as determining the structure of the following analysis.

- *Risk identification.* The next step in the risk management process involves the identification of the potential risks relating to the system. Once the risks have been identified, they can be analyzed and managed accordingly.
- *Risk assessment.* Risk assessment consists of two parts: risk analysis and risk evaluation. Risk analysis involves determining the risk controls that already exist in the organization and then analyzing risks in terms of their level of risk should they actually occur. The level of risk is normally considered to result from the product of the probability, or likelihood, of the risk occurring and the consequence of its occurrence. Risk evaluation involves comparing the estimated levels of the risk to the criteria determined when establishing the risk context.
- *Risk treatment.* Risk treatment involves identifying the options available for the dealing with the risks identified. Classic means of treating risks include avoidance, mitigation, transfer (sharing), and acceptance.
- *Communicate and consult.* Risk management is not an isolated exercise in either time or scope. Communication and consultation is essential to ensure that all risks are identified and treated appropriately.
- *Monitor and review.* This activity monitors and reviews the performance of the risk management system to ensure that it is always current and relevant.

Technical risk management is not a distinct systems-engineering activity assigned to an individual risk manager, but rather an integrated part of a sound systems engineering management effort. This is demonstrated by the fact that a large number of tools and concepts described in this text as systems engineering processes or management functions contribute to technical risk management. Examples include the process of TPMs, technical reviews and audits, T&E, and configuration management. Nor is technical risk management a one-off exercise conducted early in the Acquisition Phase of the system. Technical risk management is an on-going concern. New risks may arise, existing risks may change and old risks may disappear during the course of a system life cycle. To that end, technical risk management must continue throughout the system life cycle.

8.5 CONFIGURATION MANAGEMENT

The term *configuration* can be defined as the relative disposition or arrangement of parts of something. In the context of systems engineering, the ‘something’ is a system element, which is normally a *configuration item* (CI) that forms a part of the overall system and the ‘relative disposition or arrangement of parts’ is called a *baseline*. To that end, *configuration management* (CM) is the act of controlling and managing the physical and functional make-up (defined in the baselines) of the configuration items that comprise the system.

The main aims of CM are to identify the functional and physical characteristics of selected system components, designated as configuration items, during the Acquisition Phase; to control changes to those characteristics; and to record and report on the change processing and implementation status.

An effective CM system provides an accurate ‘snapshot’ of the state of each CI (and through these, the system). CM allows an effective history of changes to the system design to be maintained including the alternatives considered and reason for selecting the preferred alternative. For CM to be effective, the following functions must be performed throughout the life cycle:

- CM planning and management is an essential CM function that is conducted over the life cycle of the system to ensure that the appropriate tasks are performed and the necessary resources are available at the right times.
- Configuration identification identifies the items (the CIs) to be placed under configuration management.
- Configuration change management ensures that changes to CIs can be conducted in accordance with an effective, systematic, measurable, and documented change management process.
- Configuration status accounting provides accurate and up-to-date status information on the configuration of the items placed under control.
- Configuration verification and audit establishes that the information relating to the CI is complete, accurate and current; that the allocated physical, functional and interface requirements

are met by the CI; and that an adequate process is in place to confirm the correct operation of the CM system.

8.6 SYSTEMS ENGINEERING MANAGEMENT PLANNING

Systems engineering is a broad subject and to cover the entire systems engineering effort, a Systems Engineering Management Plan (SEMP), is formulated to detail the required effort. The contract SEMP is normally constructed by the customer in accordance with the requirements in the contractual documentation and reviewed and approved by the customer. Once approved, the SEMP becomes the governing plan controlling the entire systems engineering effort and all technical aspects of the project. Changes to the SEMP must be reviewed and approved as they occur and the SEMP is normally reviewed at each of the formal design reviews.

The SEMP should cover all of the major systems engineering functions. It may do so by referring to other subordinate plans such as the Configuration Management Plan (CMP), the RMP, the TRAP, and the TEMP, specification lists/trees, and CI lists. Internal company design plans and processes should also be referenced by the SEMP if applicable. In this way, the SEMP completely defines the engineering management and processes to be applied to the project.

In addition to engineering management and processes, the SEMP should also detail positions of particular responsibility within the design team, including chief designers, software and hardware team leaders, project managers, and testing personnel. Part of the SEMP approval process should include an assessment of the skills and qualifications of these key personnel. Naturally, if this information is found in other management plans, the SEMP needs to refer to that plan.

The content of the SEMP should be maintained throughout the system design and development effort. Changes to the SEMP must be approved by the customer organization as this ensures visibility into changes that may expose the project to unexpected risks (such as a change in key personnel).

Due to its coverage of the entire systems engineering effort for the project, the SEMP is a fundamental source of information when evaluating the ability and capacity of a contractor to perform the technical activities associated with the system development (as well as providing a good indication of their understanding of the system and what is needed to bring it into being). It is common, therefore, for customers to request for a draft SEMP to be submitted as part of the solicitation process for a project.

GLOSSARY

ABL	Allocated baseline
ANSI	American National Standards Institute
AT&E	Acceptance test and evaluation
CAD	Computer-aided design
CAE	Computer-aided engineering
CAM	Computer-aided manufacturing
CASE	Computer-aided software engineering
CCB	Configuration control board
CCP	Contract change proposal
CDR	Critical design review
CFD	Control flow diagram
CI	Configuration item
CIDS	Critical item development specification
CM	Configuration management
CMM	Capability maturity model
CMMI	Capability maturity model integration
COD	Concept of operations document
COTS	Commercial-off-the-shelf
CPM	Critical path method
CSC	Computer software component
CSCI	Computer software configuration item
CSU	Computer software unit
CWBS	Contract work breakdown structure
DDP	Design-dependent parameters
DFD	Data flow diagram
DID	Data item description
DoD	(U.S.) Department of Defense
DSMC	Defense Systems Management college
DSN	Design support network
DT&E	Developmental test and evaluation
ECP	Engineering change proposal
EIA	Electronics Industry Association

EMC	Electromagnetic compatibility
EMI	Electromagnetic interference
ERD	Entity relationship diagram
FAIT	Fabrication, assembly, integration and testing
FBL	Functional Baseline
FCA	Functional configuration audit
FEA	Finite element analysis
FFBD	Functional flow block diagram
FMECA	Failure mode effects and criticality assessment
FQR	Formal qualification review
FRACAS	Failure reporting, analysis and corrective action system
GPS	Global Positioning System
HMI	Human-machine interface
HWCI	Hardware configuration item
ICD	Interface control document
ICWG	Interface control working group
IEEE	Institute of Electrical and Electronics Engineers
ILS	Integrated logistic support
INCOSE	International Council on Systems Engineering
IPPD	Integrated production and process development
IS	Interim standard
LCC	Life-cycle cost
LSC	Logistic support concept
MIL-HDBK	(U.S.) Military handbook
MIL-STD	(U.S.) Military Standard
MOE	Measure of effectiveness
MOP	Measure of performance
MTP	Master test plan
NCOSE	National Council on Systems Engineering
NDI	Non-developmental item
OCD	Operational concept document / description
OFP	Operational flight program
OOA	Object oriented analysis
OSA	Open systems architecture
OT&E	Operational test and evaluation
PA	Process area

PBL	Product baseline
PCA	Physical configuration audit
PDR	Preliminary design review
PHST	Packaging, handling, storage and transportation
PIDS	Prime item development specification
PMBOK	Project management body of knowledge
PWBS	Program work breakdown structure
QA	Quality assurance
QFD	Quality function deployment
RAAF	Royal Australian Air Force
RBS	Requirements breakdown structure
RFT	Request for tender
RMP	Risk management plan
SBS	System breakdown structure
SDD	System design document
SDR	System design review
SE	Systems engineering
SECAM	Systems engineering capability assessment model
SECM	Systems engineering capability model
SE-CMM	Systems engineering CMM
SEDS	Systems engineering detailed schedule
SEI	Software Engineering Institute
SEMP	Systems engineering management plan
SEMS	Systems engineering master schedule
SOP	Standard operating procedure(s)
SOW	Statement of work
SRD	Stakeholder requirement document
SRR	Systems requirements review
SSADM	Structured systems analysis and design methodology
STD	State transition diagram
SW	Software
SWEBOK	Software Engineering Body of Knowledge
T&E	Test and evaluation
TEMP	Test and evaluation master plan
TPM	Technical performance measure
TRAP	Technical review and audit plan

TRR	Test readiness review
U.S.	United States
UML	Unified markup language
URD	User requirements document
V&V	Verification and validation
WBS	Work breakdown structure

particularly in nomenclature and emphasis. The field of systems engineering is often viewed as dry, detailed, complicated, acronym-intensive and uninteresting. Yet, the discipline holds the solution to delivering complex systems on time and within budget, and avoiding many of the failures of the past. The intention of this book is both to cover all aspects of the discipline and to provide a framework for the consideration of the many issues associated with engineering complex systems.

Our secondary purpose is to describe a complex field in a simple, easily digested manner that is accessible to a wide spectrum of readers, from students to professionals, from novices to experienced practitioners. It is directed at a wide audience and aims to be a valuable reference for all professions associated with the management of complex technical projects: project managers, systems engineers, quality assurance representatives, integrated logistic support practitioners, maintainers, and so on.

In line with the top-down approach of systems engineering, we focus in this book on the early stages of the system life cycle since the activities in these stages have the greatest impact on the successful acquisition and fielding of a system. In the interests of balance, however, we use the systems engineering framework to provide an overview of all other aspects related to systems engineering.

Download a sample chapter of *Systems Engineering Practice* (PDF).

ENDNOTES

[1] ISO/IEC 15288-2008, *Systems and Software Engineering—System Life Cycle Processes*, 2008.

[2] ISO/IEC 15288-2008, *Systems and Software Engineering—System Life Cycle Processes*, 2008.

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

[4] 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.

[5] ISO/IEC, *ISO/IEC 29148 FDIS Systems and Software Engineering—Life Cycle Processes—Requirements Engineering, 2011*.

[6] Verner, J., K. Cox, S. Bleistein, and N. Cerpa, Requirements Engineering and Software Project Success: An Industrial Survey in Australian and the U.S.”, *Australasian Journal of Information Systems*, Vol. 13, No. 1, September 2005.

[7] Project Management Institute Standards Committee, *A Guide to the Project Management Body of Knowledge*, Upper Darby P.P: Project Management Institute, 2008.

[8] Project Management Institute (PMI), *Guide to the Project Management Body of Knowledge*, Newtown Square, PA: Project Management Institute (PMI), 2012.

[9] Simon, H., *The New Science of Management Decision*, Englewood Cliffs, NJ: Prentice-Hall, 1977.

[10] ANSI/EIA-649-B-2011, *EIA Standard - National Consensus Standard for Configuration Management*, Arlington, VA.: Electronic Industries Association, 2011.

[11] ISO 31000:2009, *Risk Management—Principles and Guidelines*, International Standards Organization, 2009.