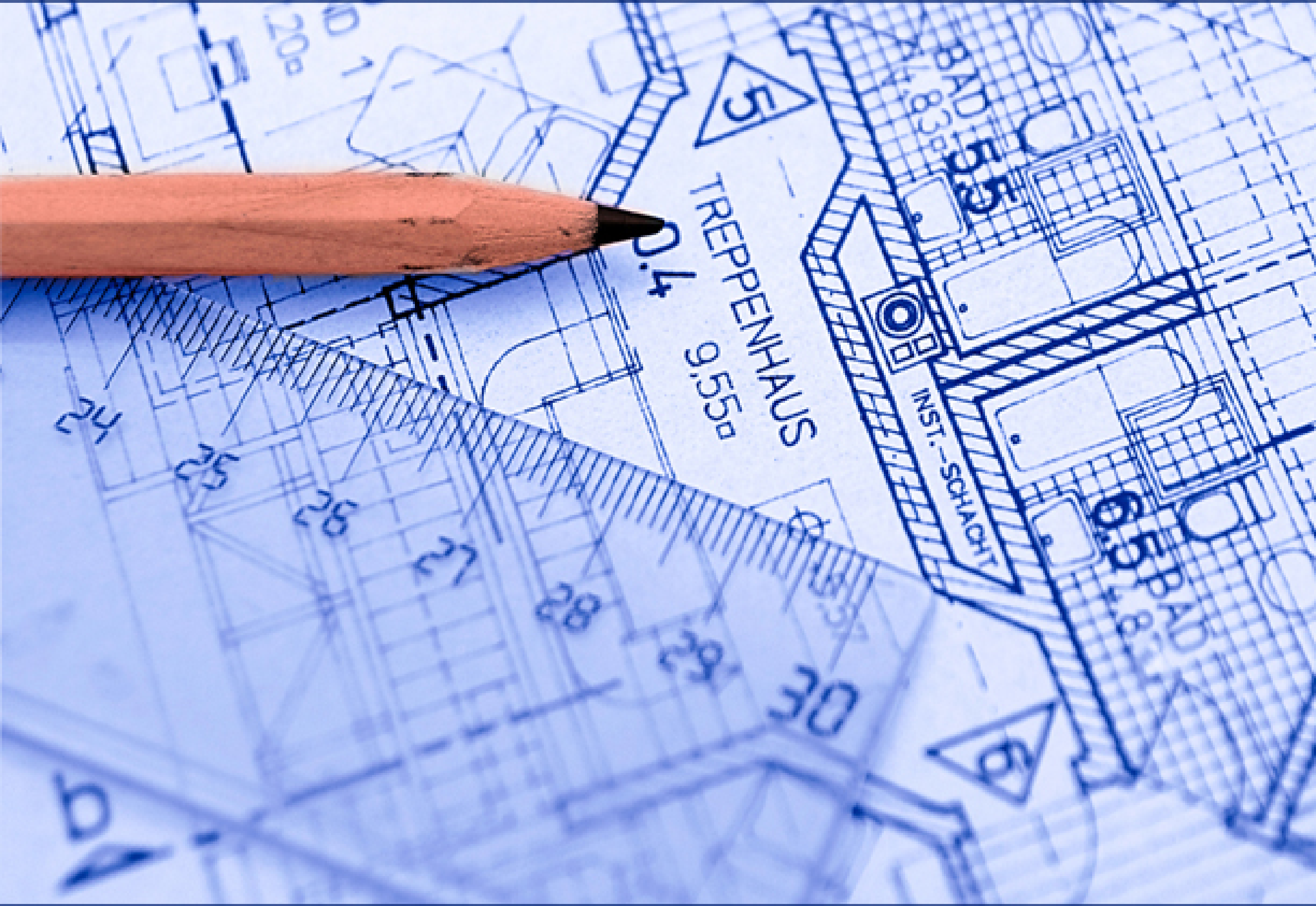
Advanced Systems Engineering



Trena Robles

First Edition, 2012

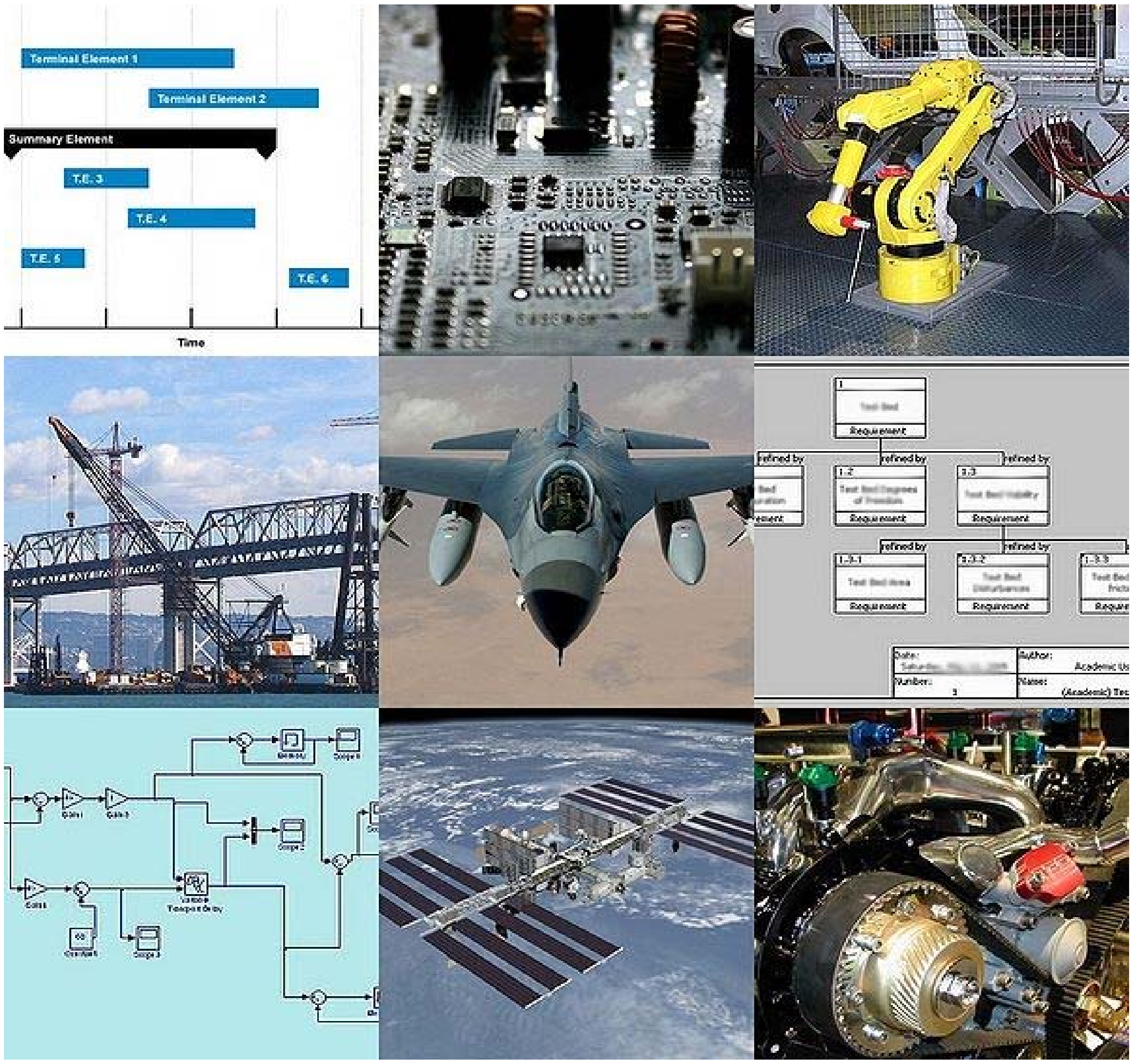
ISBN 978-81-323-2283-2

Published by:
Library Press
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter- 1

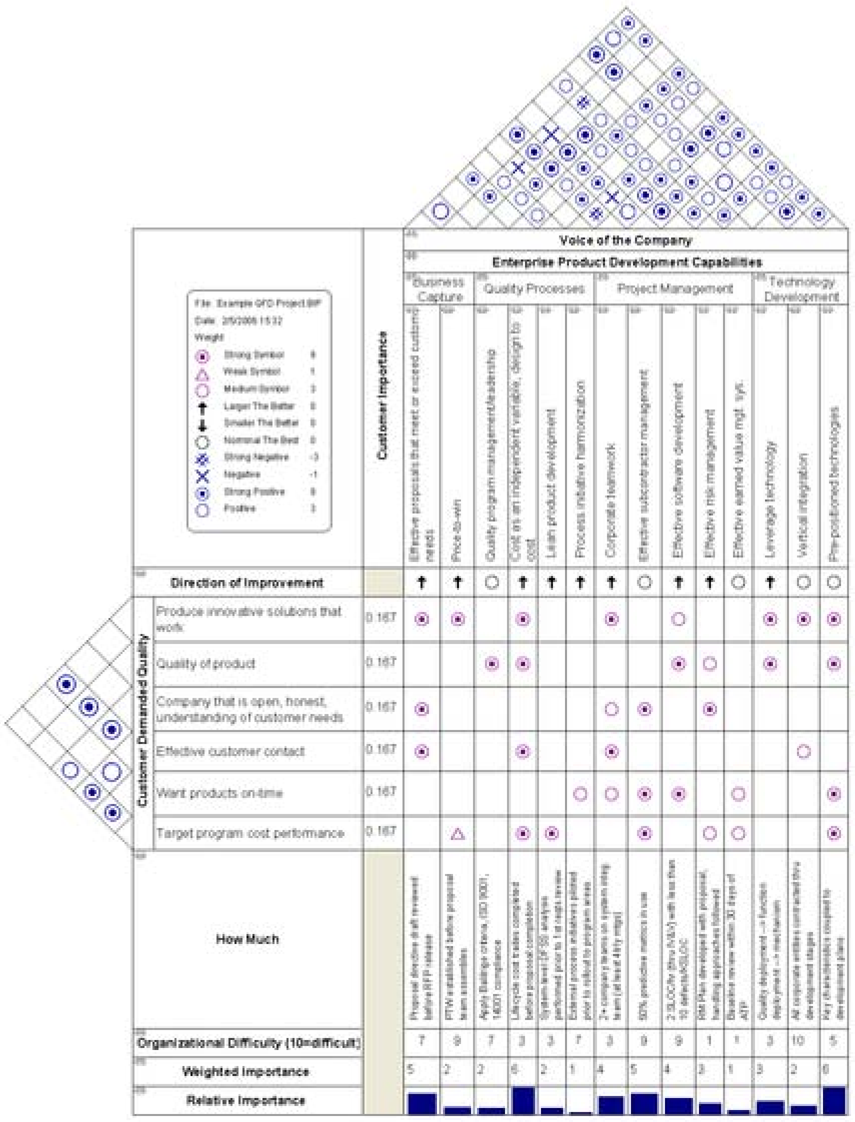
Systems Engineering



Systems engineering techniques are used in complex projects: spacecraft design, computer chip design, robotics, software integration, and bridge building. Systems engineering uses a host of tools that include modeling and simulation, requirements analysis and scheduling to manage complexity.

Systems engineering is an interdisciplinary field of engineering that focuses on how complex engineering projects should be designed and managed over the life cycle of the project. Issues such as logistics, the coordination of different teams, and automatic control of machinery become more difficult when dealing with large, complex projects. Systems engineering deals with work-processes and tools to handle such projects, and it overlaps with both technical and human-centered disciplines such as control engineering, industrial engineering, organizational studies and project management.

History



Voice of the Company
Enterprise Product Development Capabilities
Business Capture
Quality Processes
Project Management
Technology Development
Customer Importance
Direction of Improvement
Customer Demanded Quality
Produce innovative solutions that work
Quality of product
Company that is open, honest, understanding of customer needs
Effective customer contact
Want products on-time
Target program cost performance
How Much
Organizational Difficulty (10=difficult)
Weighted Importance
Relative Importance

QFD House of Quality for Enterprise Product Development Processes

The term *systems engineering* can be traced back to Bell Telephone Laboratories in the 1940s. The need to identify and manipulate the properties of a system as a whole, which in complex engineering projects may greatly differ from the sum of the parts' properties, motivated the Department of Defense, NASA, and other industries to apply the discipline.

When it was no longer possible to rely on design evolution to improve upon a system and the existing tools were not sufficient to meet growing demands, new methods began to be developed that addressed the complexity directly. The evolution of systems engineering, which continues to this day, comprises the development and identification of new methods and modeling techniques. These methods aid in better comprehension of engineering systems as they grow more complex. Popular tools that are often used in the systems engineering context were developed during these times, including USL, UML, QFD, and IDEF0.

In 1990, a professional society for systems engineering, the *National Council on Systems Engineering* (NCOSE), was founded by representatives from a number of U.S. corporations and organizations. NCOSE was created to address the need for improvements in systems engineering practices and education. As a result of growing involvement from systems engineers outside of the U.S., the name of the organization was changed to the International Council on Systems Engineering (INCOSE) in 1995. Schools in several countries offer graduate programs in systems engineering, and continuing education options are also available for practicing engineers.

Concept

Some definitions

"An interdisciplinary approach and means to enable the realization of successful systems" — *INCOSE handbook, 2004.*

"System engineering is a robust approach to the design, creation, and operation of systems. In simple terms, the approach consists of identification and quantification of system goals, creation of alternative system design concepts, performance of design trades, selection and implementation of the best design, verification that the design is properly built and integrated, and post-implementation assessment of how well the system meets (or met) the goals." — *NASA Systems Engineering*

Handbook, 1995.

"The Art and Science of creating effective systems, using whole system, whole life principles" OR "The Art and Science of creating optimal solution systems to complex issues and problems" — *Derek Hitchins, Prof. of Systems Engineering, former president of INCOSE (UK), 2007.*

"The concept from the engineering standpoint is the evolution of the engineering scientist, i.e., the scientific generalist who maintains a broad outlook. The method is that of the team approach. On large-scale-system problems, teams of scientists and engineers, generalists as well as specialists, exert their joint efforts to find a solution and physically realize it...The technique has been variously called the systems approach or the team development method." — *Harry H. Goode & Robert E. Machol, 1957.*

"The systems engineering method recognizes each system is an integrated whole even though composed of diverse, specialized structures and sub-functions. It further recognizes that any system has a number of objectives and that the balance between them may differ widely from system to system. The methods seek to optimize the overall system functions according to the weighted objectives and to achieve maximum compatibility of its parts." — *Systems Engineering Tools by Harold Chestnut, 1965.*

Systems engineering signifies both an approach and, more recently, as a discipline in engineering. The aim of education in systems engineering is to simply formalize the approach and in doing so, identify new methods and research opportunities similar to the way it occurs in other fields of engineering. As an approach, systems engineering is holistic and interdisciplinary in flavour.

Origins and traditional scope

The traditional scope of engineering embraces the design, development, production and operation of physical systems, and systems engineering, as originally conceived, falls within this scope. "Systems engineering", in this sense of the term, refers to the distinctive set of concepts, methodologies, organizational structures (and so on) that have been developed to meet the challenges of engineering functional physical systems of unprecedented complexity. The Apollo program is a leading example of a systems engineering project.

The use of the term " system engineer " has evolved over time to embrace a wider, more holistic concept of "systems" and of engineering processes. This evolution of the definition has been a subject of ongoing controversy and the term continues to be applied to both the narrower and broader scope.

Holistic view

Systems engineering focuses on analyzing and eliciting customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem, the system lifecycle. Oliver *et al.* claim that the systems engineering process can be decomposed into

- a *Systems Engineering Technical Process*, and
- a *Systems Engineering Management Process*.

Within Oliver's model, the goal of the Management Process is to organize the technical effort in the lifecycle, while the Technical Process includes *assessing available information*, *defining effectiveness measures*, to *create a behavior model*, *create a structure model*, *perform trade-off analysis*, and *create sequential build & test plan*.

Depending on their application, although there are several models that are used in the industry, all of them aim to identify the relation between the various stages mentioned above and incorporate feedback. Examples of such models include the Waterfall model and the VEE model.

Interdisciplinary field

System development often requires contribution from diverse technical disciplines. By providing a systems (holistic) view of the development effort, systems engineering helps mold all the technical contributors into a unified team effort, forming a structured development process that proceeds from concept to production to operation and, in some cases, to termination and disposal.

This perspective is often replicated in educational programs in that systems engineering courses are taught by faculty from other engineering departments which, in effect, helps create an interdisciplinary environment.

Managing complexity

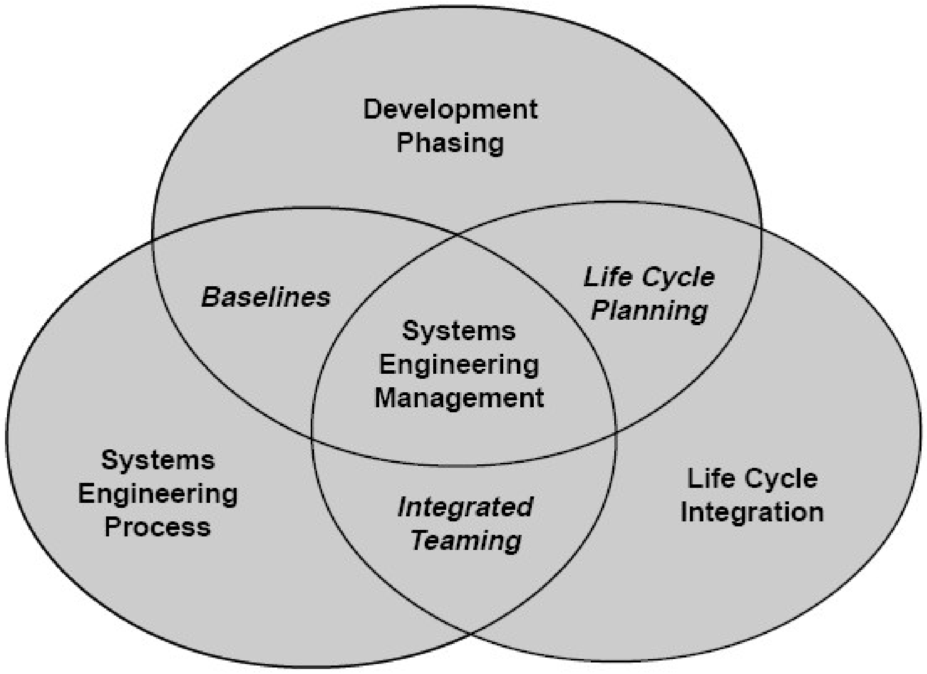
The need for systems engineering arose with the increase in complexity of systems and projects. When speaking in this context, complexity incorporates not only engineering systems, but also the logical human organization of data. At the same time, a system can become more complex due to an increase in size as well as with an increase in the amount of data, variables, or the number of fields that are involved in the design. The International Space Station is an example of such a system.

The development of smarter control algorithms, microprocessor design, and analysis of environmental systems also come within the purview of systems engineering. Systems engineering encourages the use of tools and methods to better comprehend and manage complexity in systems. Some examples of these tools can be seen here:

- *System model, Modeling, and Simulation*,
- System architecture,
- *Optimization*,
- *System dynamics*,
- *Systems analysis*,
- *Statistical analysis*,
- *Reliability analysis*, and
- *Decision making*

Taking an interdisciplinary approach to engineering systems is inherently complex since the behavior of and interaction among system components is not always immediately well defined or understood. Defining and characterizing such systems and subsystems and the interactions among them is one of the goals of systems engineering. In doing so, the gap that exists between informal requirements from users, operators, marketing organizations, and technical specifications is successfully bridged.

Scope



The scope of systems engineering activities

One way to understand the motivation behind systems engineering is to see it as a method, or practice, to identify and improve common rules that exist within a wide variety of systems. Keeping this in mind, the principles of systems engineering — holism, emergent behavior, boundary, et al. — can be applied to any system, complex or otherwise, provided systems thinking is employed at all levels. Besides defense and aerospace, many information and technology based companies, software development firms, and industries in the field of electronics & communications require systems engineers as part of their team.

An analysis by the INCOSE Systems Engineering center of excellence (SECOE) indicates that optimal effort spent on systems engineering is about 15-20% of the total project effort. At the same time, studies have shown that systems engineering essentially leads to reduction in costs among other benefits. However, no quantitative survey at a larger scale encompassing a wide variety of industries has been conducted until recently. Such studies are underway to determine the effectiveness and quantify the benefits of systems engineering.

Systems engineering encourages the use of modeling and simulation to validate assumptions or theories on systems and the interactions within them.

Use of methods that allow early detection of possible failures, in safety engineering, are integrated into the design process. At the same time, decisions made at the beginning of a project whose consequences are not clearly understood can have enormous implications later in the life of a system, and it is the task of the modern systems engineer to explore these issues and make critical decisions. There is no method which guarantees that decisions made today will still be valid when a system goes into service years or decades after it is first conceived but there are techniques to support the process of systems engineering. Examples include the use of soft systems methodology, Jay Wright Forrester's System dynamics method and the Unified Modeling Language (UML), each of which are currently being explored, evaluated and developed to support the engineering decision making process.

Education

Education in systems engineering is often seen as an extension to the regular engineering courses, reflecting the industry attitude that engineering students need a foundational background in one of the traditional engineering disciplines (e.g. mechanical engineering, industrial engineering, computer engineering, electrical engineering) plus practical, real-world experience in order to be effective as systems engineers. Undergraduate university programs in systems engineering are rare.

INCOSE maintains a continuously updated Directory of Systems Engineering Academic Programs worldwide. As of 2006, there are about 75 institutions in United States that offer 130 undergraduate and graduate programs in systems engineering. Education in systems engineering can be taken as *SE-centric* or *Domain-centric*.

- *SE-centric* programs treat systems engineering as a separate discipline and all the courses are taught focusing on systems engineering practice and techniques.
- *Domain-centric* programs offer systems engineering as an option that can be exercised with another major field in engineering.

Both these patterns cater to educate the systems engineer who is able to oversee interdisciplinary projects with the depth required of a core-engineer.

Systems engineering topics

Systems engineering tools are strategies, procedures, and techniques that aid in performing systems engineering on a project or product. The purpose of these tools vary from database management, graphical browsing, simulation, and reasoning, to document production, neutral import/export and more.

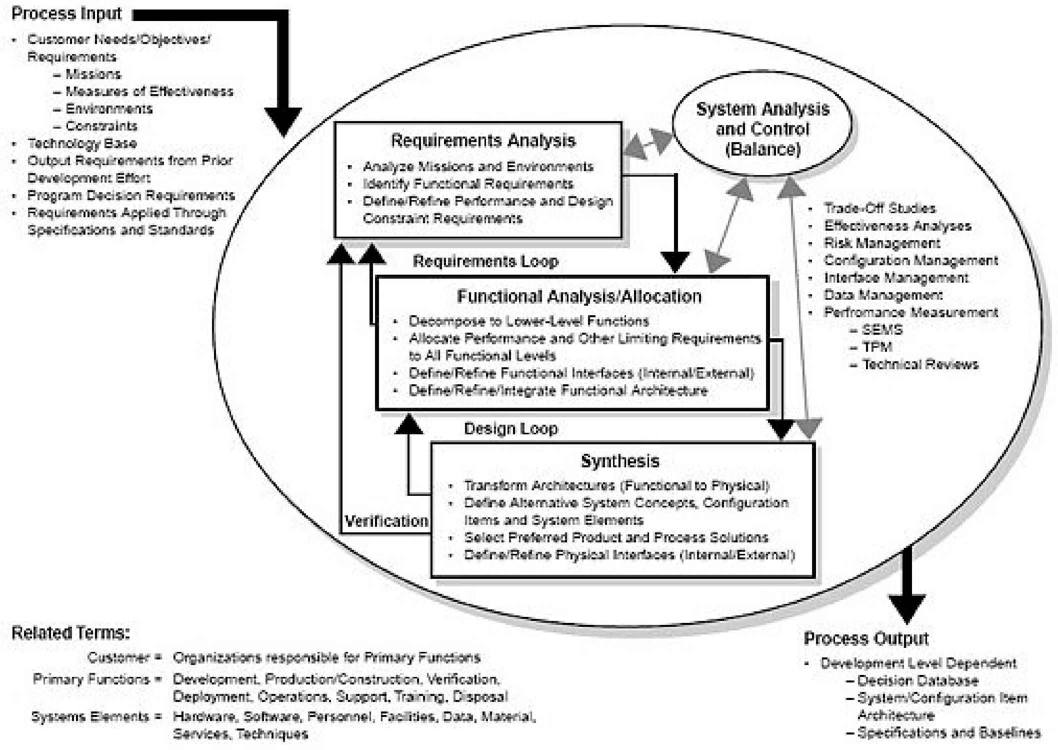
System

There are many definitions of what a system is in the field of systems engineering. Below are a few authoritative definitions:

- ANSI/EIA-632-1999: "An aggregation of end products and enabling products to achieve a given purpose."
- IEEE Std 1220-1998: "A set or arrangement of elements and processes that are related and whose behavior satisfies customer/operational needs and provides for life cycle sustainment of the products."
- ISO/IEC 15288:2008: "A combination of interacting elements organized to achieve one or more stated purposes."
- NASA Systems Engineering Handbook: "(1) The combination of elements that function together to produce the capability to meet a need. The elements include all hardware, software, equipment, facilities, personnel, processes, and procedures needed for this purpose. (2) The end product (which performs operational functions) and enabling products (which provide life-cycle support services to the operational end products) that make up a system."
- INCOSE Systems Engineering Handbook: "homogeneous entity that exhibits predefined behavior in the real world and is composed of heterogeneous parts that do not individually exhibit that behavior and an integrated configuration of components and/or subsystems."
- INCOSE: "A system is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce systems-level results. The results include system level qualities, properties, characteristics, functions, behavior and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected."

The systems engineering process

Depending on their application, tools are used for various stages of the systems engineering process:


Process Input
• Customer Needs/Objectives/ Requirements
– Missions
– Measures of Effectiveness
– Environments
– Constraints
• Technology Base
• Output Requirements from Prior Development Effort
• Program Decision Requirements
• Requirements Applied Through Specifications and Standards
Requirements Analysis
• Analyze Missions and Environments
• Identify Functional Requirements
• Define/Refine Performance and Design Constraint Requirements
System Analysis and Control (Balance)
• Trade-Off Studies
• Effectiveness Analyses
• Risk Management
• Configuration Management
• Interface Management
• Data Management
• Perfromance Measurement
– SEMS
– TPM
– Technical Reviews
Requirements Loop
Functional Analysis/Allocation
• Decompose to Lower-Level Functions
• Allocate Performance and Other Limiting Requirements to All Functional Levels
• Define/Refine Functional Interfaces (Internal/External)
• Define/Refine/Integrate Functional Architecture
Design Loop
Synthesis
• Transform Architectures (Functional to Physical)
• Define Alternative System Concepts, Configuration Items and System Elements
• Select Preferred Product and Process Solutions
• Define/Refine Physical Interfaces (Internal/External)
Verification
Related Terms:
Customer = Organizations responsible for Primary Functions
Primary Functions = Development, Production/Construction, Verification, Deployment, Operations, Support, Training, Disposal
Systems Elements = Hardware, Software, Personnel, Facilities, Data, Material, Services, Techniques
Process Output
• Development Level Dependent
– Decision Database
– System/Configuration Item Architecture
– Specifications and Baselines

Using models

Models play important and diverse roles in systems engineering. A model can be defined in several ways, including:

- An abstraction of reality designed to answer specific questions about the real world
- An imitation, analogue, or representation of a real world process or structure; or
- A conceptual, mathematical, or physical tool to assist a decision maker.

Together, these definitions are broad enough to encompass physical engineering models used in the verification of a system design, as well as schematic models like a functional flow block diagram and mathematical (i.e., quantitative) models used in the trade study process. This section focuses on the last.

The main reason for using mathematical models and diagrams in trade studies is to provide estimates of system effectiveness, performance or technical attributes, and cost from a set of known or estimable quantities. Typically, a collection of separate models is needed to provide all of these outcome variables. The heart of any mathematical model is a set of meaningful quantitative relationships among its inputs and outputs. These relationships can be as simple as adding up constituent quantities to obtain a total, or as complex as a set of differential equations describing the trajectory of a spacecraft in a gravitational field. Ideally, the relationships express causality, not just correlation.

Tools for graphic representations

Initially, when the primary purpose of a systems engineer is to comprehend a complex problem, graphic representations of a system are used to communicate a system's functional and data requirements. Common graphical representations include:

- Functional Flow Block Diagram (FFBD)
- VisSim
- Data Flow Diagram (DFD)
- N2 (N-Squared) Chart
- IDEF0 Diagram
- UML Use case diagram
- UML Sequence diagram
- USL Function Maps and Type Maps.
- Enterprise Architecture frameworks, like TOGAF, MODAF, Zachman Frameworks etc.

A graphical representation relates the various subsystems or parts of a system through functions, data, or interfaces. Any or each of the above methods are used in an industry based on its requirements. For instance, the N2 chart may be used where interfaces between systems is important. Part of the design phase is to create structural and behavioral models of the system.

Once the requirements are understood, it is now the responsibility of a systems engineer to refine them, and to determine, along with other engineers, the best technology for a job. At this point starting with a trade study, systems engineering encourages the use of weighted choices to determine the best option. A decision matrix, or Pugh method, is one way (QFD is another) to make this choice while considering all criteria that are important. The trade study in turn informs the design which again affects the graphic representations of the system (without changing the requirements). In an SE process, this stage represents the iterative step that is carried out until a feasible solution is found. A decision matrix is often populated using techniques such as statistical analysis, reliability analysis, system dynamics (feedback control), and optimization methods.

At times a systems engineer must assess the existence of feasible solutions, and rarely will customer inputs arrive at only one. Some customer requirements will produce no feasible solution. Constraints must be traded to find one or more feasible solutions. The customers' wants become the most valuable input to such a trade and cannot be assumed. Those wants/desires may only be discovered by the customer once the customer finds that he has overconstrained the problem. Most commonly, many feasible solutions can be found, and a sufficient set of constraints must be defined to produce an optimal solution. This situation is at times advantageous because one can present an opportunity to improve the design towards one or many ends, such as cost or schedule. Various modeling methods can be used to solve the problem including constraints and a cost function.

Systems Modeling Language (SysML), a modeling language used for systems engineering applications, supports the specification, analysis, design, verification and validation of a broad range of complex systems.

Universal Systems Language (USL) is a systems oriented object modeling language with executable (computer independent) semantics for defining complex systems, including software.

Related Fields and Sub-fields

Many related fields may be considered tightly coupled to systems engineering. These areas have contributed to the development of systems engineering as a distinct entity.

Cognitive systems engineering

Cognitive systems engineering (CSE) is a specific approach to the description and analysis of human-machine systems or sociotechnical systems. The three main themes of CSE are how humans cope with complexity, how work is accomplished by the use of artefacts, and how human-machine systems and socio-technical systems can be described as joint cognitive systems. CSE has since its beginning become a recognised scientific discipline, sometimes also referred to as Cognitive Engineering. The concept of a Joint Cognitive System (JCS) has in particular become widely used as a way of understanding how complex socio-technical systems can be described with varying degrees of resolution. The more than 20 years of experience with CSE has been described extensively.

Configuration Management

Like systems engineering, Configuration Management as practiced in the defence and aerospace industry is a broad systems-level practice. The field parallels the taskings of systems engineering; where systems engineering deals with requirements development, allocation to development items and verification, Configuration Management deals with requirements capture, traceability to the development item, and audit of development item to ensure that it has achieved the desired functionality that systems engineering and/or Test and Verification Engineering have proven out through objective testing.

Control engineering

Control engineering and its design and implementation of control systems, used extensively in nearly every industry, is a large sub-field of systems engineering. The cruise control on an automobile and the guidance system for a ballistic missile are two examples. Control systems theory is an active field of applied mathematics involving the investigation of solution spaces and the development of new methods for the analysis of the control process.

Industrial engineering

Industrial engineering is a branch of engineering that concerns the development, improvement, implementation and evaluation of integrated systems of people, money, knowledge, information, equipment, energy, material and process. Industrial engineering draws upon the principles and methods of engineering analysis and synthesis, as well as mathematical, physical and social sciences together with the principles and methods of engineering analysis and design to specify, predict and evaluate the results to be obtained from such systems.

Interface design

Interface design and its specification are concerned with assuring that the pieces of a system connect and inter-operate with other parts of the system and with external systems as necessary. Interface design also includes assuring that system interfaces be able to accept new features, including mechanical, electrical and logical interfaces, including reserved wires, plug-space, command codes and bits in communication protocols. This is known as extensibility. Human-Computer Interaction (HCI) or Human-Machine Interface (HMI) is another aspect of interface design, and is a critical aspect of modern systems engineering. Systems engineering principles are applied in the design of network protocols for local-area networks and wide-area networks.

Mechatronic engineering

Mechatronic engineering, like Systems engineering, is a multidisciplinary field of engineering that uses dynamical systems modeling to express tangible constructs. In that regards it is almost indistinguishable from Systems Engineering, but what sets it apart is the focus on smaller details rather than larger generalizations and relationships. As such, both fields are distinguished by the scope of their projects rather than the methodology of their practice.

Operations research

Operations research supports systems engineering. The tools of operations research are used in systems analysis, decision making, and trade studies. Several schools teach SE courses within the operations research or industrial engineering department, highlighting the role systems engineering plays in complex projects. Operations research, briefly, is concerned with the optimization of a process under multiple constraints.

Performance engineering

Performance engineering is the discipline of ensuring a system will meet the customer's expectations for performance throughout its life. Performance is usually defined as the speed with which a certain operation is executed or the capability of executing a number of such operations in a unit of time. Performance may be degraded when an operations queue to be executed is throttled when the capacity is of the system is limited. For example, the performance of a packet-switched network would be characterised by the end-to-end packet transit delay or the number of packets switched within an hour. The design of high-performance systems makes use of analytical or simulation modeling, whereas the delivery of high-performance implementation involves thorough performance testing. Performance engineering relies heavily on statistics, queueing theory and probability theory for its tools and processes.

Program management and project management.

Program management (or programme management) has many similarities with systems engineering, but has broader-based origins than the engineering ones of systems engineering. Project management is also closely related to both program management and systems engineering.

Proposal engineering

Proposal engineering is the application of scientific and mathematical principles to design, construct, and operate a cost-effective proposal development system. Basically, proposal engineering uses the "systems engineering process" to create a cost effective proposal and increase the odds of a successful proposal.

Reliability engineering

Reliability engineering is the discipline of ensuring a system will meet the customer's expectations for reliability throughout its life; i.e. it will not fail more frequently than expected. Reliability engineering applies to all aspects of the system. It is closely associated with maintainability, availability and logistics engineering. Reliability engineering is always a critical component of safety engineering, as in failure modes and effects analysis (FMEA) and hazard fault tree analysis, and of security engineering. Reliability engineering relies heavily onstatistics, probability theory and reliability theory for its tools and processes.

Safety engineering

The techniques of safety engineering may be applied by non-specialist engineers in designing complex systems to minimize the probability of safety-critical failures. The "System Safety Engineering" function helps to identify "safety hazards" in emerging designs, and may assist with techniques to "mitigate" the effects of (potentially) hazardous conditions that cannot be designed out of systems.

Security engineering

Security engineering can be viewed as an interdisciplinary field that integrates the community of practice for control systems design, reliability, safety and systems engineering. It may involve such sub-specialties as authentication of system users, system targets and others: people, objects and processes.

Software engineering

From its beginnings, software engineering has helped shape modern systems engineering practice. The techniques used in the handling of complexes of large software-intensive systems has had a major effect on the shaping and reshaping of the tools, methods and processes of SE.

Chapter- 2

Control Engineering

Control systems play a critical role in space flight

Control engineering or **Control systems engineering** is the engineering discipline that applies control theory to design systems with predictable behaviors. The practice uses sensors to measure the output performance of the device being controlled (often a vehicle) and those measurements can be used to give feedback to the input actuators that can make corrections toward desired performance. When a device is designed to perform without the need of human inputs for correction it is called automatic control (such as cruise control for regulating a car's speed). Multi-disciplinary in nature, control systems engineering activities focus on implementation of control systems mainly derived by mathematical modeling of systems of a diverse range.

Overview

Modern day control engineering (also called control systems engineering) is a relatively new field of study that gained a significant attention during 20th century with the advancement in technology. It can be broadly defined as practical application of control theory. Control engineering has an essential role in a wide range of control systems, from simple household washing machines to high-performance F-16 fighter aircraft. It seeks to understand physical systems, using mathematical modeling, in terms of inputs, outputs and various components with different behaviors; use control systems design tools to develop controllers for those systems; and implement controllers in physical systems employing available technology. A system can be mechanical, electrical, fluid, chemical, financial and even biological, and the mathematical modeling, analysis and controller design uses control theory in one or many of the time, frequency and complex-s domains, depending on the nature of the design problem.

History

Automatic control Systems were first developed over two thousand years ago. The first feedback control device on record is thought to be the ancient water clock of Ktesibios in Alexandria Egypt around the third century B.C. It kept time by regulating the water level in a vessel and, therefore, the water flow from that vessel. This certainly was a successful device as water clocks of similar design were still being made in ~Baghdad when the Mongols captured the city in 1258 A.D. A variety of automatic devices have been used over the centuries to accomplish useful tasks or simply to just entertain. The latter includes the automata, popular in Europe in the 17th and 18th centuries, featuring dancing figures that would repeat the same task over and over again; these automata are examples of open-loop control. Milestones among feedback, or "closed-loop" automatic control devices, include the temperature regulator of a furnace attributed to Drebbel, circa 1620, and the centrifugal flyball governor used for regulating the speed of steam engines by James Watt in 1788.

In his 1868 paper "On Governors", J. C. Maxwell (who discovered the Maxwell electromagnetic field equations) was able to explain instabilities exhibited by the flyball governor using differential equations to describe the control system. This demonstrated the importance and usefulness of mathematical models and methods in understanding complex phenomena, and signaled the beginning of mathematical control and systems theory. Elements of control theory had appeared earlier but not as dramatically and convincingly as in Maxwell's analysis.

Control theory made significant strides in the next 100 years. New mathematical techniques made it possible to control, more accurately, significantly more complex dynamical systems than the original flyball governor. These techniques include developments in optimal control in the 1950s and 1960s, followed by progress in stochastic, robust, adaptive and optimal control methods in the 1970s and 1980s. Applications of control methodology have helped make possible space travel and communication satellites, safer and more efficient aircraft, cleaner auto engines, cleaner and more efficient chemical processes, to mention but a few.

Before it emerged as a unique discipline, control engineering was practiced as a part of mechanical engineering and control theory was studied as a part of electrical engineering, since electrical circuits can often be easily described using control theory techniques. In the very first control relationships, a current output was represented with a voltage control input. However, not having proper technology to implement electrical control systems, designers left with the option of less efficient and slow responding mechanical systems. A very effective mechanical controller that is still widely used in some hydro plants is the governor. Later on, previous to modern power electronics, process control systems for industrial applications were devised by mechanical engineers using pneumatic and hydraulic control devices, many of which are still in use today.

Control theory

There are two major divisions in control theory, namely, classical and modern, which have direct implications over the control engineering applications. The scope of classical control theory is limited to single-input and single-output (SISO) system design. The system analysis is carried out in time domain using differential equations, in complex-s domain with Laplace transform or in frequency domain by transforming from the complex-s domain. All systems are assumed to be second order and single variable, and higher-order system responses and multivariable effects are ignored. A controller designed using classical theory usually requires on-site tuning due to design approximations. Yet, due to easier physical implementation of classical controller designs as compared to systems designed using modern control theory, these controllers are preferred in most industrial applications. The most common controllers designed using classical control theory are PID controllers.

In contrast, modern control theory is carried out strictly in the complex-s or the frequency domain, and can deal with multi-input and multi-output (MIMO) systems. This overcomes the limitations of classical control theory in more sophisticated design problems, such as fighter aircraft control. In modern design, a system is represented as a set of first order differential equations defined using state variables. Nonlinear, multivariable, adaptive and robust control theories come under this division. Being fairly new, modern control theory has many areas yet to be explored. Scholars like Rudolf E. Kalman and Aleksandr Lyapunov are well-known among the people who have shaped modern control theory.

Control systems

Control engineering is the engineering discipline that focuses on the modeling of a diverse range of dynamic systems (e.g. mechanical systems) and the design of controllers that will cause these systems to behave in the desired manner. Although such controllers need not be electrical many are and hence control engineering is often viewed as a subfield of electrical engineering. However, the falling price of microprocessors is making the actual implementation of a control system essentially trivial. As a result, focus is shifting back to the mechanical engineering discipline, as intimate knowledge of the physical system being controlled is often desired.

Electrical circuits, digital signal processors and microcontrollers can all be used to implement Control systems. Control engineering has a wide range of applications from the flight and propulsion systems of commercial airliners to the cruise control present in many modern automobiles.

In most of the cases, control engineers utilize feedback when designing control systems. This is often accomplished using a PID controller system. For example, in an automobile with cruise control the vehicle's speed is continuously monitored and fed back to the system which adjusts the motor's torque accordingly. Where there is regular feedback, control theory can be used to determine how the system responds to such feedback. In practically all such systems stability is important and control theory can help ensure stability is achieved.

Although feedback is an important aspect of control engineering, control engineers may also work on the control of systems without feedback. This is known as open loop control. A classic example of open loop control is a washing machine that runs through a pre-determined cycle without the use of sensors.

Control engineering education

At many universities, control engineering courses are taught in Electrical and Electronic Engineering, Mechatronics Engineering, Mechanical engineering, and Aerospace engineering; in others it is connected to computer science, as most control techniques today are implemented through computers, often as Embedded systems (as in the automotive field). The field of control within chemical engineering is often known as process control. It deals primarily with the control of variables in a chemical process in a plant. It is taught as part of the undergraduate curriculum of any chemical engineering program, and employs many of the same principles in control engineering. Other engineering disciplines also overlap with control engineering, as it can be applied to any system for which a suitable model can be derived.

Control engineering has diversified applications that include science, finance management, and even human behavior. Students of control engineering may start with a linear control system course dealing with the time and complex-s domain, which requires a thorough background in elementary mathematics and Laplace transform (called classical control theory). In linear control, the student does frequency and time domain analysis. Digital control and nonlinear control courses require z transformation and algebra respectively, and could be said to complete a basic control education. From here onwards there are several sub branches.

Recent advancement

Originally control engineering was all about continuous systems. Development of computer control tools posed a requirement of discrete control system engineering because the communications between the computer-based digital controller and the physical system are governed by a computer clock. The equivalent to Laplace transform in the discrete domain is the z-transform. Today many of the control systems are computer controlled and they consist of both digital and analogue components.

Therefore, at the design stage either digital components are mapped into the continuous domain and the design is carried out in the continuous domain, or analogue components are mapped in to discrete domain and design is carried out there. The first of these two methods is more commonly encountered in practice because many industrial systems have many continuous systems components, including mechanical, fluid, biological and analogue electrical components, with a few digital controllers.

Similarly, the design technique has progressed from paper-and-ruler based manual design to computer-aided design, and now to computer-automated design (CAutoD), which has been made possible by evolutionary computation. CAutoD can be applied not just to tuning a predefined control scheme, but also to controller structure optimisation, system identification and invention of novel control systems, based purely upon a performance requirement, independent of any specific control scheme.

Chapter- 3

Organizational Studies

Organizational studies, sometimes known as **organizational science**, encompass the systematic study and careful application of knowledge about how people act within organizations. Organizational studies sometimes is considered a sister field for, or overarching designation that includes, the following disciplines: industrial and organizational psychology, organizational behavior, human resources, and management. However, there is no universally accepted classification system for such subfields.

Overview

Organizational studies encompass the study of organizations from multiple viewpoints, methods, and levels of analysis. For instance, one textbook divides these multiple viewpoints into three perspectives: modern, symbolic, and postmodern. Another traditional distinction, present especially in American academia, is between the study of "micro" organizational behaviour — which refers to individual and group dynamics in an organizational setting — and "macro" strategic management and organizational theory which studies whole organizations and industries, how they adapt, and the strategies, structures and contingencies that guide them. To this distinction, some scholars have added an interest in "meso" scale structures - power, culture, and the networks of individuals and units in organizations — and "field" level analysis which study how whole populations of organizations interact.

Whenever people interact in organizations, many factors come into play. Modern organizational studies attempt to understand and model these factors. Like all modernist social sciences, organizational studies seek to control, predict, and explain. There is some controversy over the ethics of controlling workers' behavior, as well as the manner in which workers are treated. As such, organizational behaviour or OB (and its cousin, Industrial psychology) have at times been accused of being the scientific tool of the powerful. Those accusations notwithstanding, OB can play a major role in organizational development, enhancing organizational performance, as well as individual and group performance/satisfaction/commitment.

One of the main goals of organizational theorists is, according to Simms (1994) "to revitalize organizational theory and develop a better conceptualization of organizational life." An organizational theorist should carefully consider levels assumptions being made in theory, and is concerned to help managers and administrators.

History

Kurt Lewin attended the Macy conferences and is commonly identified as the founder of the movement to study groups scientifically.

The Greek philosopher Plato wrote about the essence of leadership. Aristotle addressed the topic of persuasive communication. The writings of 16th century Italian philosopher Niccolò Machiavelli laid the foundation for contemporary work on organizational power and politics. In 1776, Adam Smith advocated a new form of organizational structure based on the division of labour. One hundred years later, German sociologist Max Weber wrote about rational organizations and initiated discussion of charismatic leadership. Soon after, Frederick Winslow Taylor introduced the systematic use of goal setting and rewards to motivate employees. In the 1920s, Australian-born Harvard professor Elton Mayo and his colleagues conducted productivity studies at Western Electric's Hawthorne plant in the United States.

Though it traces its roots back to Max Weber and earlier, organizational studies is generally considered to have begun as an academic discipline with the advent of scientific management in the 1890s, with Taylorism representing the peak of this movement. Proponents of scientific management held that rationalizing the organization with precise sets of instructions and time-motion studies would lead to increased productivity. Studies of different compensation systems were carried out.

After the First World War, the focus of organizational studies shifted to analysis of how human factors and psychology affected organizations, a transformation propelled by the identification of the Hawthorne Effect. This Human Relations Movement focused on teams, motivation, and the actualization of the goals of individuals within organizations.

Prominent early scholars included Chester Barnard, Henri Fayol, Frederick Herzberg, Abraham Maslow, David McClelland, and Victor Vroom.

The Second World War further shifted the field, as the invention of large-scale logistics and operations research led to a renewed interest in rationalist approaches to the study of organizations. Interest grew in theory and methods native to the sciences, including systems theory, the study of organizations with a complexity theory perspective and complexity strategy. Influential work was done by Herbert Alexander Simon and James G. March and the so-called "Carnegie School" of organizational behavior.

In the 1960s and 1970s, the field was strongly influenced by social psychology and the emphasis in academic study was on quantitative research. An explosion of theorizing, much of it at Stanford University and Carnegie Mellon, produced Bounded Rationality, Informal Organization, Contingency Theory, Resource Dependence, Institutional Theory, and Organizational Ecology theories, among many others.

Starting in the 1980s, cultural explanations of organizations and change became an important part of study. Qualitative methods of study became more acceptable, informed by anthropology, psychology and sociology. A leading scholar was Karl Weick.

Elton Mayo

Elton Mayo, an Australian national, headed the Hawthorne Studies at Harvard. In his classic writing in 1931, Human Problems of an Industrial Civilization, he advised managers to deal with emotional needs of employees at work.

Mary Parker Follett

Mary Parker Follett was a pioneer management consultant in the industrial world. As a writer, she provided analyses on workers as having complex combinations of attitude, beliefs, and needs. She told managers to motivate employees on their job performance, a "pull" rather than a "push" strategy.

Douglas McGregor

Douglas McGregor proposed two theories/assumptions, which are very nearly the opposite of each other, about human nature based on his experience as a management consultant. His first theory was "Theory X", which is pessimistic and negative; and according to McGregor it is how managers traditionally perceive their workers. Then, in order to help managers replace that theory/assumption, he gave "Theory Y" which takes a more modern and positive approach. He believed that managers could achieve more if they start perceiving their employees as self-energized, committed, responsible and creative beings. By means of his Theory Y, he in fact challenged the traditional theorists to adopt a developmental approach to their employees. He also wrote a book, *The Human Side of Enterprise*, in 1960; this book has become a foundation for the modern view of employees at work.

Current state of the field

Organizational behaviour is currently a growing field. Organizational studies departments generally form part of business schools, although many universities also have industrial psychology and industrial economics programs.

The field is highly influential in the business world with practitioners like Peter Drucker and Peter Senge, who turned the academic research into business practices. Organizational behaviour is becoming more important in the global economy as people with diverse backgrounds and cultural values have to work together effectively and efficiently. It is also under increasing criticism as a field for its ethnocentric and pro-capitalist assumptions.

During the last 20 years organizational behavior study and practice has developed and expanded through creating integrations with other domains:

- Anthropology became an interesting prism to understanding firms as communities, by introducing concepts like Organizational culture, 'organizational rituals' and 'symbolic acts' enabling new ways to understand organizations as communities.
- Leadership Understanding: the crucial role of leadership at various level of an organization in the process of change management.
- Ethics and their importance as pillars of any vision and one of the most important driving forces in an organization.

Methods used in organizational studies

A variety of methods are used in organizational studies. They include quantitative methods found in other social sciences such as multiple regression, non-parametric statistics, time series analysis, Meta-analysis and ANOVA. In addition, computer simulation in organizational studies has a long history in organizational studies. Qualitative methods are also used, such as ethnography, which involves direct participant observation, single and multiple case analysis, grounded theory approaches, and other historical methods.

Systems framework

The systems framework is also fundamental to organizational theory as organizations are complex dynamic goal-oriented processes. One of the early thinkers in the field was Alexander Bogdanov, who developed his Tectology, a theory widely considered a precursor of Bertalanffy's General Systems Theory, aiming to model and design human organizations. Kurt Lewin was particularly influential in developing the systems perspective within organizational theory and coined the term "systems of ideology", from his frustration with behavioural psychologies that became an obstacle to sustainable work in psychology. The complexity theory perspective on organizations is another systems view of organizations.

The systems approach to organizations relies heavily upon achieving negative entropy through openness and feedback. A systemic view on organizations is transdisciplinary and integrative. In other words, it transcends the perspectives of individual disciplines, integrating them on the basis of a common "code", or more exactly, on the basis of the formal apparatus provided by systems theory. The systems approach gives primacy to the interrelationships, not to the elements of the system. It is from these dynamic interrelationships that new properties of the system emerge. In recent years, *systems thinking* has been developed to provide techniques for studying systems in holistic ways to supplement traditional reductionistic methods. In this more recent tradition, systems theory in organizational studies is considered by some as a humanistic extension of the natural sciences.

Theories and models

Decision making

- Mintzberg's managerial roles
- Rational Decision-Making Model
- Scientific management
- Garbage can model

Theories of decision making can be subdivided into three categories

- Normative (concentrates on how decision should be made)
- Descriptive (concerned with how the thinker came up with their judgement)
- Prescripted (aim to improve decision making)

Organization structures and dynamics

- Incentive theory is a concept of human resources or management theory. In the corporate sense, it states that firm owners should structure employee compensation in such a way that the employees' goals are aligned with owners' goals. As it applies to the operations of firms, it is more accurately called the principal-agent problem.
- Bureaucracy
- Complexity theory and organizations
- Contingency theory
- Evolutionary Theory and organizations
- Hybrid organisation
- Informal Organization

- Model of Organizational Citizenship behaviour
- Model of organizational justice
- Model of Organizational Misbehaviour
- Resource dependence theory
- Transaction cost

Personality traits theories

- Big Five personality traits
- Holland's Typology of Personality and Congruent Occupations
- Myers-Briggs Type Indicator
- Herrmann Brain Dominance Instrument

Control and stress modelling

- Herzberg's Two factor theory
- Theory X and Theory Y

Motivation in organizations

Motivation the forces either internal or external to a person that arouse enthusiasm and resistance to pursue a certain course of action. According to Baron et al. (2008): "Although motivation is a broad and complex concept, organizational scientists have agreed on its basic characteristics. Drawing from various social sciences, we define motivation as the set of processes that arouse, direct, and maintain human behavior toward attaining some goal"

There are many different motivation theories such as:

- Attribution theory
- Equity theory
- Maslow's hierarchy of needs
- Incentive theory (psychology)
- Model of emotional labor in organizations
- Frederick Herzberg two-factor theory
- Expectancy theory

Chapter- 4

Project Management

Project management is the discipline of planning, organizing, securing and managing resources to bring about the successful completion of specific project goals and objectives. It is sometimes conflated with program management, however technically that is actually a higher level construction: a group of related and somehow interdependent engineering projects.

A project is a temporary endeavor, having a defined beginning and end (usually constrained by date, but can be by funding or deliverables), undertaken to meet unique goals and objectives, usually to bring about beneficial change or added value. The temporary nature of projects stands in contrast to business as usual (or operations), which are repetitive, permanent or semi-permanent functional work to produce products or services. In practice, the management of these two systems is often found to be quite different, and as such requires the development of distinct technical skills and the adoption of separate management.

The primary challenge of project management is to achieve all of the engineering project goals and objectives while honoring the preconceived project constraints. Typical constraints are scope, time, and budget. The secondary—and more ambitious—challenge is to optimize the allocation and integration of inputs necessary to meet pre-defined objectives.

History

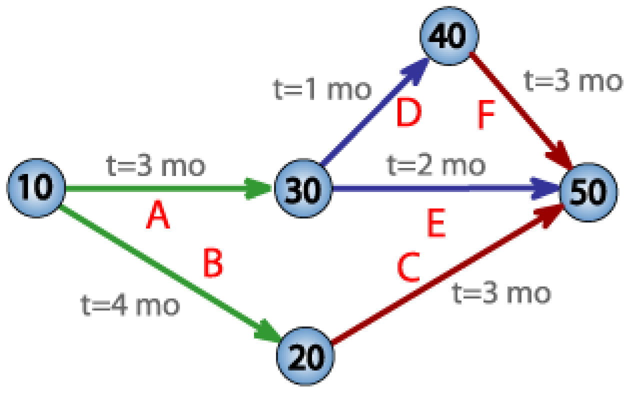
Roman Soldiers Building a Fortress, Trajan's Column 113 AD

Project management has been practiced since early civilization. Until 1900 civil engineering projects were generally managed by creative architects and engineers themselves, among those for example Vitruvius (1st century BC), Christopher Wren (1632–1723), Thomas Telford (1757–1834) and Isambard Kingdom Brunel (1806–1859). It was in the 1950s that organizations started to systematically apply project management tools and techniques to complex engineering projects.

Henry Gantt (1861-1919), the father of planning and control techniques

As a discipline, Project Management developed from several fields of application including civil construction, engineering, and heavy defense activity. Two forefathers of project management are Henry Gantt, called the father of planning and control techniques, who is famous for his use of the Gantt chart as a project management tool; and Henri Fayol for his creation of the 5 management functions which form the foundation of the body of knowledge associated with project and program management. Both Gantt and Fayol were students of Frederick Winslow Taylor's theories of scientific management. His work is the forerunner to modern project management tools including work breakdown structure (WBS) and resource allocation.

The 1950s marked the beginning of the modern Project Management era where core engineering fields come together working as one. Project management became recognized as a distinct discipline arising from the management discipline with engineering model. In the United States, prior to the 1950s, projects were managed on an *ad hoc* basis using mostly Gantt Charts, and informal techniques and tools. At that time, two mathematical project-scheduling models were developed. The "Critical Path Method" (CPM) was developed as a joint venture between DuPont Corporation and Remington Rand Corporation for managing plant maintenance projects. And the "Program Evaluation and Review Technique" or PERT, was developed by Booz Allen Hamilton as part of the United States Navy's (in conjunction with the Lockheed Corporation) Polaris missile submarine program; These mathematical techniques quickly spread into many private enterprises.



PERT network chart for a seven-month project with five milestones

At the same time, as project-scheduling models were being developed, technology for project cost estimating, cost management, and engineering economics was evolving, with pioneering work by Hans Lang and others. In 1956, the American Association of Cost Engineers (now AACE International; the Association for the Advancement of Cost Engineering) was formed by early practitioners of project management and the associated specialties of planning and scheduling, cost estimating, and cost/schedule control (project control). AACE continued its pioneering work and in 2006 released the first integrated process for portfolio, program and project management (Total Cost Management Framework).

The International Project Management Association (IPMA) was founded in Europe in 1967, as a federation of several national project management associations. IPMA maintains its federal structure today and now includes member associations on every continent except Antarctica. IPMA offers a Four Level Certification program based on the IPMA Competence Baseline (ICB). The ICB covers technical competences, contextual competences, and behavioral competences.

In 1969, the Project Management Institute (PMI) was formed in the USA. PMI publishes *A Guide to the Project Management Body of Knowledge* (PMBOK Guide), which describes project management practices that are common to "most projects, most of the time." PMI also offers multiple certifications.

The American Academy of Project Management (AAPM) International Board of Standards 1996 was the first to institute post-graduate certifications such as the MPM Master Project Manager, PME Project Management E-Business, CEC Certified-Ecommerce Consultant, and CIPM Certified International Project Manager. The AAPM also issues the post-graduate standards body of knowledge for executives.

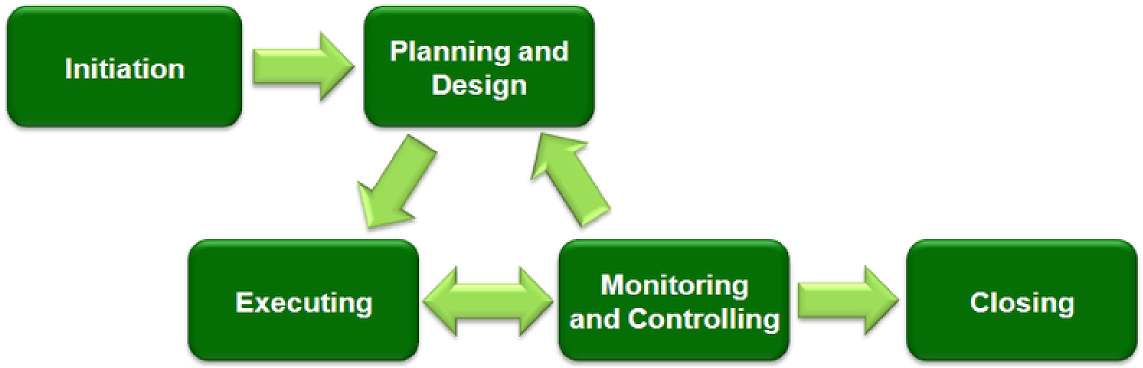
Approaches

There are a number of approaches to managing project activities including agile, interactive, incremental, and phased approaches.

Regardless of the methodology employed, careful consideration must be given to the overall project objectives, timeline, and cost, as well as the roles and responsibilities of all participants and stakeholders.

The traditional approach

A traditional phased approach identifies a sequence of steps to be completed. In the "traditional approach", we can distinguish 5 components of a project (4 stages plus control) in the development of a project:



Typical development phases of an engineering project

- Project initiation stage;
- Project planning and design stage;
- Project execution and construction stage;
- Project monitoring and controlling systems;
- Project completion.

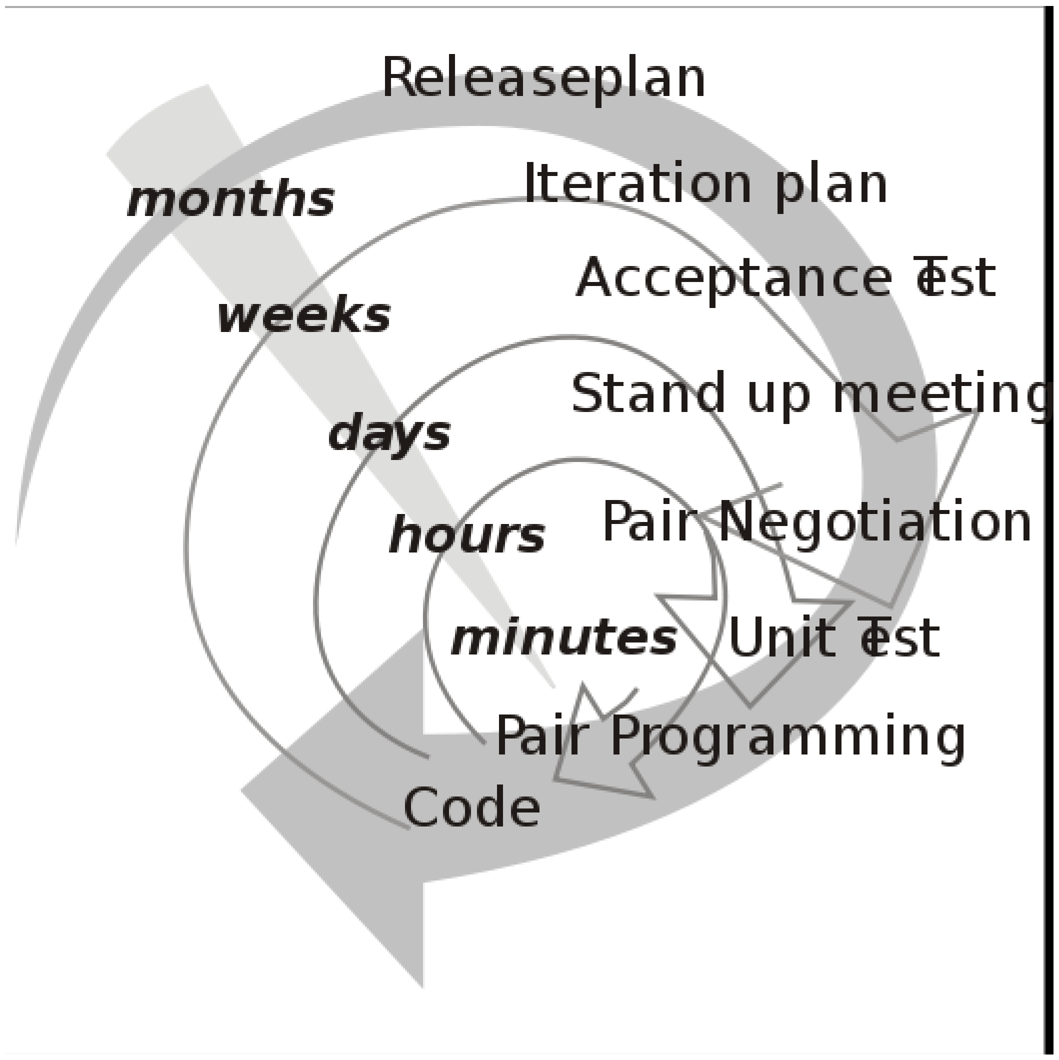
Not all the projects will visit every stage as projects can be terminated before they reach completion. Some projects do not follow a structured planning and/or monitoring stages. Some projects will go through steps 2, 3 and 4 multiple times.

Many industries use variations on these project stages. For example, when working on a brick and mortar design and construction, projects will typically progress through stages like Pre-Planning, Conceptual Design, Schematic Design, Design Development, Construction Drawings (or Contract Documents), and Construction Administration. In software development, this approach is often known as the waterfall model, i.e., one series of tasks after another in linear sequence. In software development many organizations have adapted the Rational Unified Process (RUP) to fit this methodology, although RUP does not require or explicitly recommend this practice. Waterfall development works well for small, well defined projects, but often fails in larger projects of undefined and ambiguous nature. The Cone of Uncertainty explains some of this as the planning made on the initial phase of the project suffers from a high degree of uncertainty. This becomes especially true as software development is often the realization of a new or novel product. In projects where requirements have not been finalized and can change, requirements management is used to develop an accurate and complete definition of the behavior of software that can serve as the basis for software development. While the terms may differ from industry to industry, the actual stages typically follow common steps to problem solving — "defining the problem, weighing options, choosing a path, implementation and evaluation."

Critical Chain Project Management

Critical Chain Project Management (CCPM) is a method of planning and managing projects that puts more emphasis on the resources (physical and human) needed in order to execute project tasks. The most complex part involves engineering professionals of different fields (Civil, Electrical, Mechanical etc) working together. It is an application of the Theory of Constraints (TOC) to projects. The goal is to increase the rate of throughput (or completion rates) of projects in an organization. Applying the first three of the five focusing steps of TOC, the system constraint for all projects is identified as are the resources. To exploit the constraint, tasks on the critical chain are given priority over all other activities. Finally, projects are planned and managed to ensure that the resources are ready when the critical chain tasks must start, subordinating all other resources to the critical chain.

Regardless of project type, the project plan should undergo Resource Leveling, and the longest sequence of resource-constrained tasks should be identified as the critical chain. In multi-project environments, resource leveling should be performed across projects. However, it is often enough to identify (or simply select) a single "drum" resource—a resource that acts as a constraint across projects—and stagger projects based on the availability of that single resource.



Planning and feedback loops in Extreme Programming (XP) with the time frames of the multiple loops.

Extreme Project Management

In critical studies of Project Management, it has been noted that several of these fundamentally PERT-based models are not well suited for the multi-project company environment of today. Most of them are aimed at very large-scale, one-time, non-routine projects, and nowadays all kinds of management are expressed in terms of projects.

Using complex models for "projects" (or rather "tasks") spanning a few weeks has been proven to cause unnecessary costs and low maneuverability in several cases. Instead, project management experts try to identify different "lightweight" models, such as Agile Project Management methods including Extreme Programming for software development and Scrum techniques.

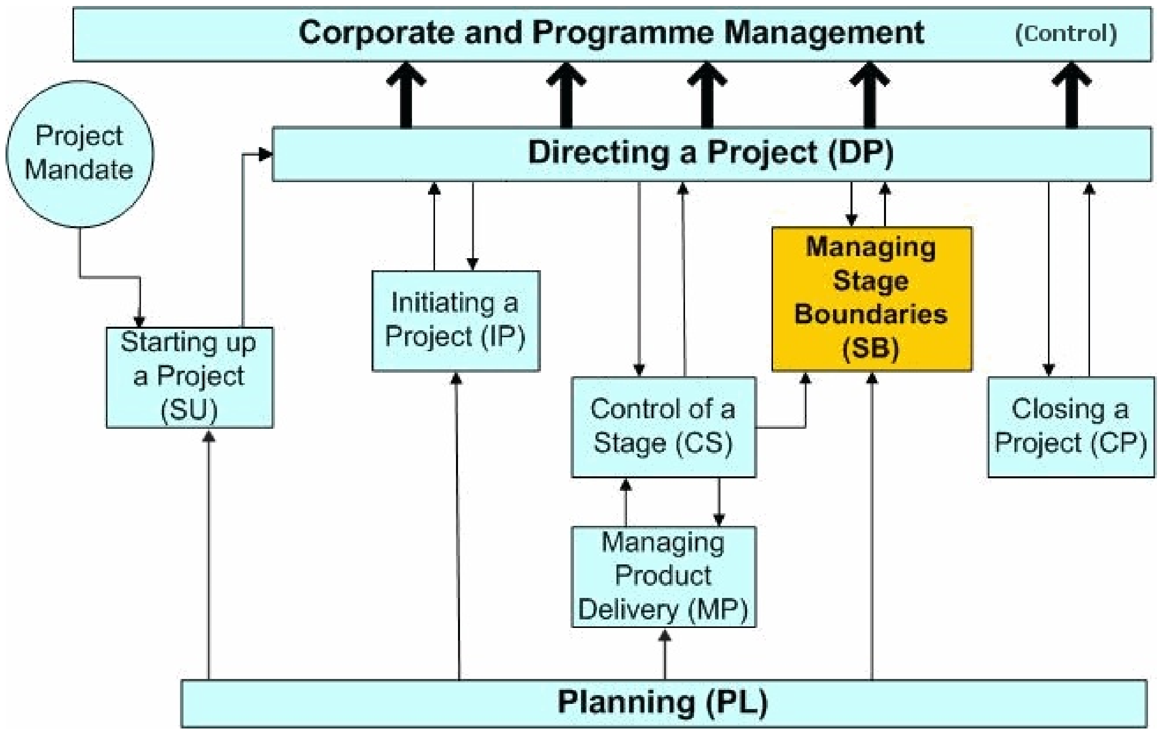
The generalization of Extreme Programming to other kinds of projects is extreme project management, which may be used in combination with the process modeling and management principles of human interaction management.

Event chain methodology

Event chain methodology is another method that complements critical path method and critical chain project management methodologies.

Event chain methodology is an uncertainty modeling and schedule network analysis technique that is focused on identifying and managing events and event chains that affect project schedules. Event chain methodology helps to mitigate the negative impact of psychological heuristics and biases, as well as to allow for easy modeling of uncertainties in the project schedules. Event chain methodology is based on the following principles.

- **Probabilistic moment of risk:** An activity (task) in most real life processes is not a continuous uniform process. Tasks are affected by external events, which can occur at some point in the middle of the task.
- **Event chains:** Events can cause other events, which will create event chains. These event chains can significantly affect the course of the project. Quantitative analysis is used to determine a cumulative effect of these event chains on the project schedule.
- **Critical events or event chains:** The single events or the event chains that have the most potential to affect the projects are the "critical events" or "critical chains of events." They can be determined by the analysis.
- **Project tracking with events:** Even if a project is partially completed and data about the project duration, cost, and events occurred is available, it is still possible to refine information about future potential events and helps to forecast future project performance.
- **Event chain visualization:** Events and event chains can be visualized using event chain diagrams on a Gantt chart.


Corporate and Programme Management
(Control)
Project Mandate
Directing a Project (DP)
Initiating a Project (IP)
Managing Stage Boundaries (SB)
Starting up a Project (SU)
Control of a Stage (CS)
Closing a Project (CP)
Managing Product Delivery (MP)
Planning (PL)

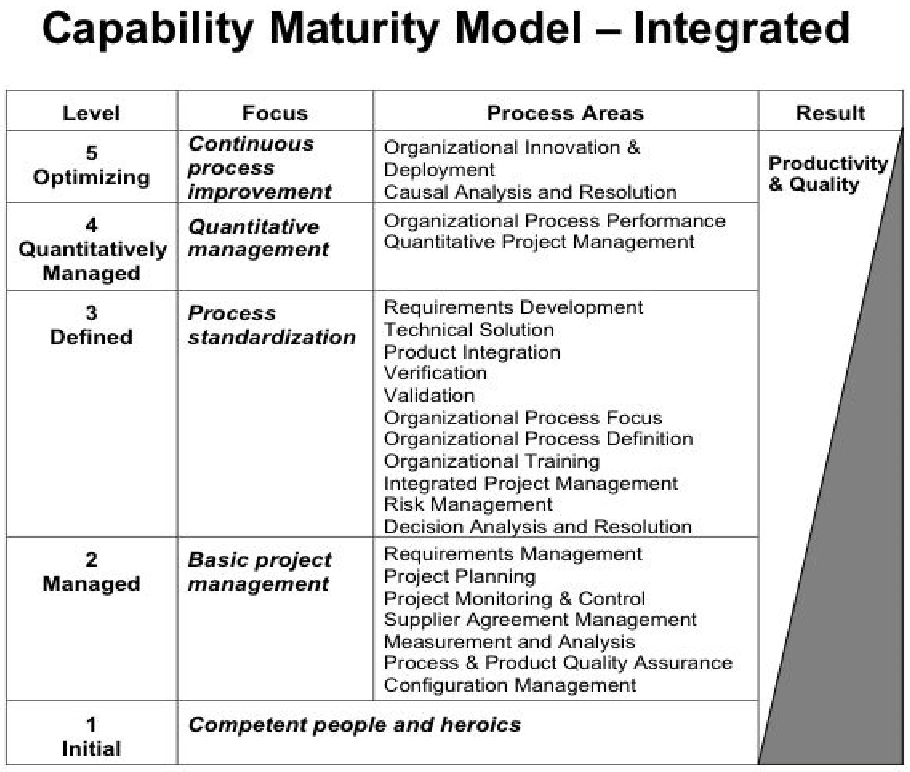
The PRINCE2 process model

PRINCE2

PRINCE2 is a structured approach to project management, released in 1996 as a generic project management method. It combined the original PROMPT methodology (which evolved into the PRINCE methodology) with IBM's MITP (managing the implementation of the total project) methodology. PRINCE2 provides a method for managing projects within a clearly defined framework. PRINCE2 describes procedures to coordinate people and activities in a project, how to design and supervise the project, and what to do if the project has to be adjusted if it does not develop as planned.

In the method, each process is specified with its key inputs and outputs and with specific goals and activities to be carried out. This allows for automatic control of any deviations from the plan. Divided into manageable stages, the method enables an efficient control of resources. On the basis of close monitoring, the project can be carried out in a controlled and organized way.

PRINCE2 provides a common language for all participants in the project. The various management roles and responsibilities involved in a project are fully described and are adaptable to suit the complexity of the project and skills of the organization.



Capability Maturity Model – Integrated

Level	Focus	Process Areas	Result
5 Optimizing	*Continuous process improvement*	Organizational Innovation & Deployment Causal Analysis and Resolution	Productivity & Quality
4 Quantitatively Managed	*Quantitative management*	Organizational Process Performance Quantitative Project Management	
3 Defined	*Process standardization*	Requirements Development Technical Solution Product Integration Verification Validation Organizational Process Focus Organizational Process Definition Organizational Training Integrated Project Management Risk Management Decision Analysis and Resolution	
2 Managed	*Basic project management*	Requirements Management Project Planning Project Monitoring & Control Supplier Agreement Management Measurement and Analysis Process & Product Quality Assurance Configuration Management	
1 Initial	*Competent people and heroics*		

Capability Maturity Model, predecessor of the CMMI Model

Process-based management

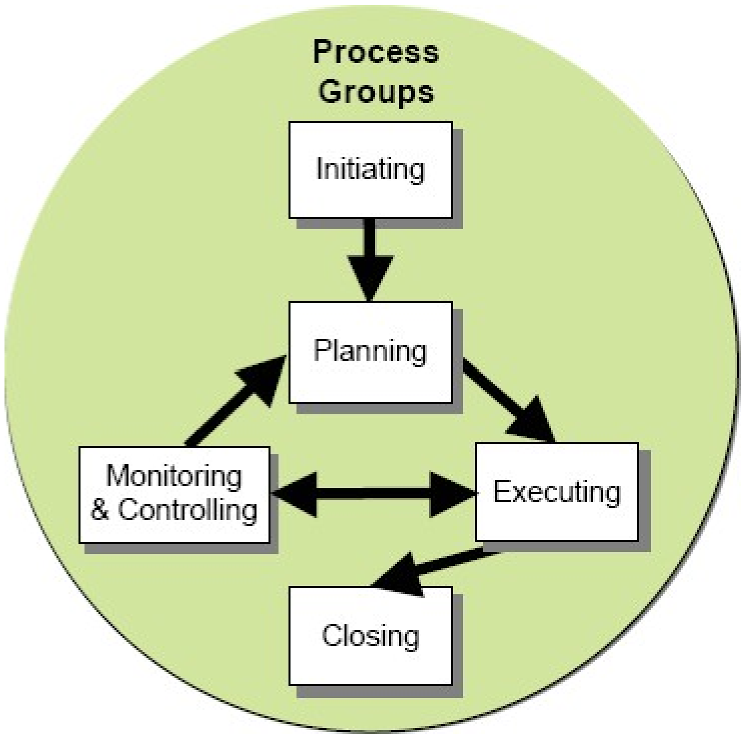
Also furthering the concept of project control is the incorporation of process-based management. This area has been driven by the use of Maturity models such as the CMMI (Capability Maturity Model Integration) and ISO/IEC15504 (SPICE - Software Process Improvement and Capability Estimation).

Agile Project Management

Agile Project Management approaches based on the principles of human interaction management are founded on a process view of human collaboration. This contrasts sharply with the traditional approach. In the agile software development or flexible product development approach, the project is seen as a series of relatively small tasks conceived and executed as the situation demands in an adaptive manner, rather than as a completely pre-planned process.

Processes

Traditionally, project management includes a number of elements: four to five process groups, and a control system. Regardless of the methodology or terminology used, the same basic project management processes will be used.



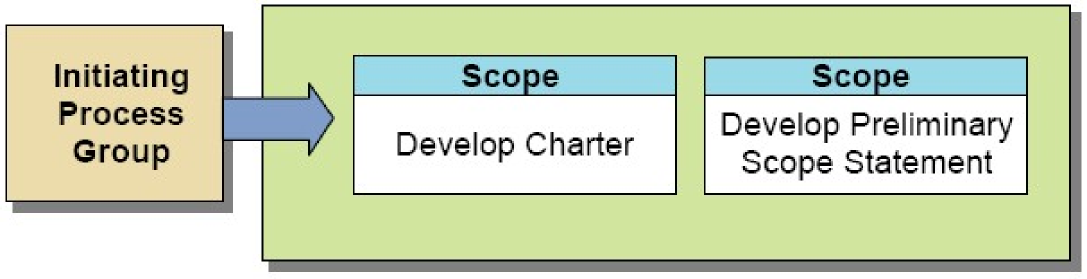
The project development stages

Major process groups generally include:

- Initiation
- Planning or development
- Production or execution
- Monitoring and controlling
- Closing

In project environments with a significant exploratory element (e.g., Research and development), these stages may be supplemented with decision points (go/no go decisions) at which the project's continuation is debated and decided. An example is the Stage-Gate model.

Initiation



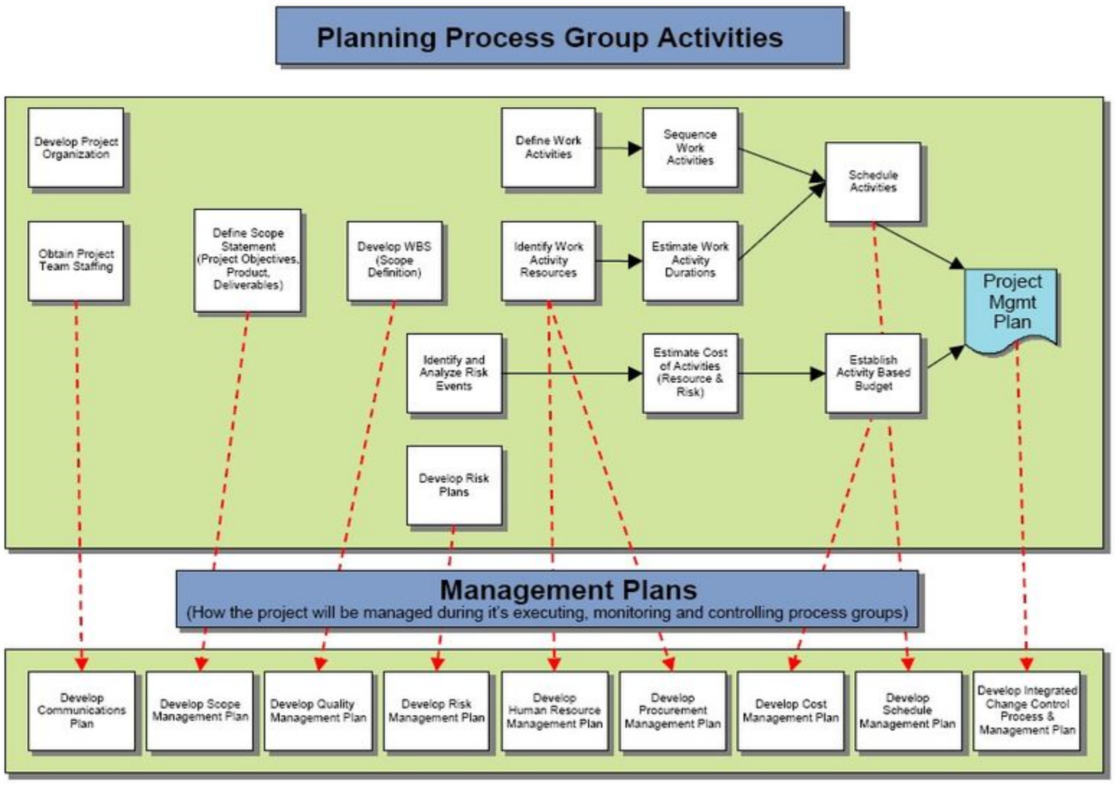
Initiating Process Group Processes

The initiation processes determine the nature and scope of the project. If this stage is not performed well, it is unlikely that the project will be successful in meeting the business' needs. The key project controls needed here are an understanding of the business environment and making sure that all necessary controls are incorporated into the project. Any deficiencies should be reported and a recommendation should be made to fix them.

The initiation stage should include a plan that encompasses the following areas:

- Analyzing the business needs/requirements in measurable goals
- Reviewing of the current operations
- Financial analysis of the costs and benefits including a budget
- Stakeholder analysis, including users, and support personnel for the project
- Project charter including costs, tasks, deliverables, and schedule

Planning and design


Planning Process Group Activities
Develop Project Organization
Obtain Project Team Staffing
Define Scope Statement (Project Objectives, Product, Deliverables)
Develop WBS (Scope Definition)
Define Work Activities
Sequence Work Activities
Schedule Activities
Identify Work Activity Resources
Estimate Work Activity Durations
Project Mgmt Plan
Identify and Analyze Risk Events
Estimate Cost of Activities (Resource & Risk)
Establish Activity Based Budget
Develop Risk Plans
Management Plans
(How the project will be managed during it's executing, monitoring and controlling process groups)
Develop Communications Plan
Develop Scope Management Plan
Develop Quality Management Plan
Develop Risk Management Plan
Develop Human Resource Management Plan
Develop Procurement Management Plan
Develop Cost Management Plan
Develop Schedule Management Plan
Develop Integrated Change Control Process & Management Plan

Planning Process Group Activities

After the initiation stage, the project is planned to an appropriate level of detail. The main purpose is to plan time, cost and resources adequately to estimate the work needed and to effectively manage risk during project execution. As with the Initiation process group, a failure to adequately plan greatly reduces the project's chances of successfully accomplishing its goals.

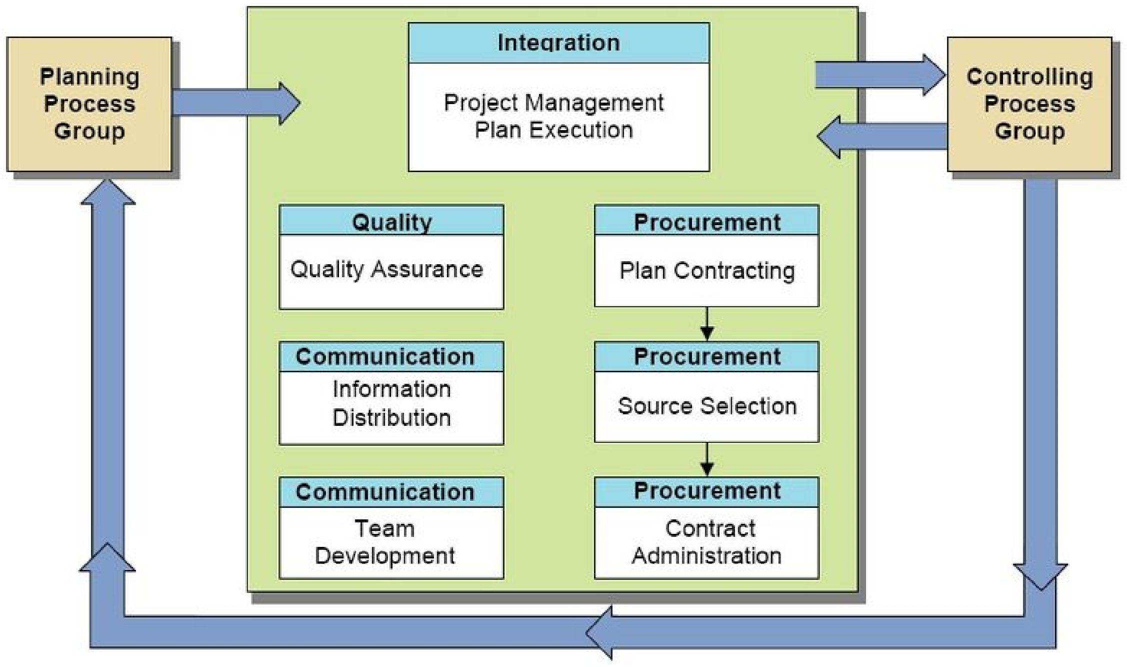
Project planning generally consists of

- determining how to plan (e.g. by level of detail or rolling wave);
- developing the scope statement;
- selecting the planning team;
- identifying deliverables and creating the work breakdown structure;
- identifying the activities needed to complete those deliverables and networking the activities in their logical sequence;
- estimating the resource requirements for the activities;
- estimating time and cost for activities;
- developing the schedule;
- developing the budget;
- risk planning;
- gaining formal approval to begin work.

Additional processes, such as planning for communications and for scope management, identifying roles and responsibilities, determining what to purchase for the project and holding a kick-off meeting are also generally advisable.

For new product development projects, conceptual design of the operation of the final product may be performed concurrent with the project planning activities, and may help to inform the planning team when identifying deliverables and planning activities.

Executing

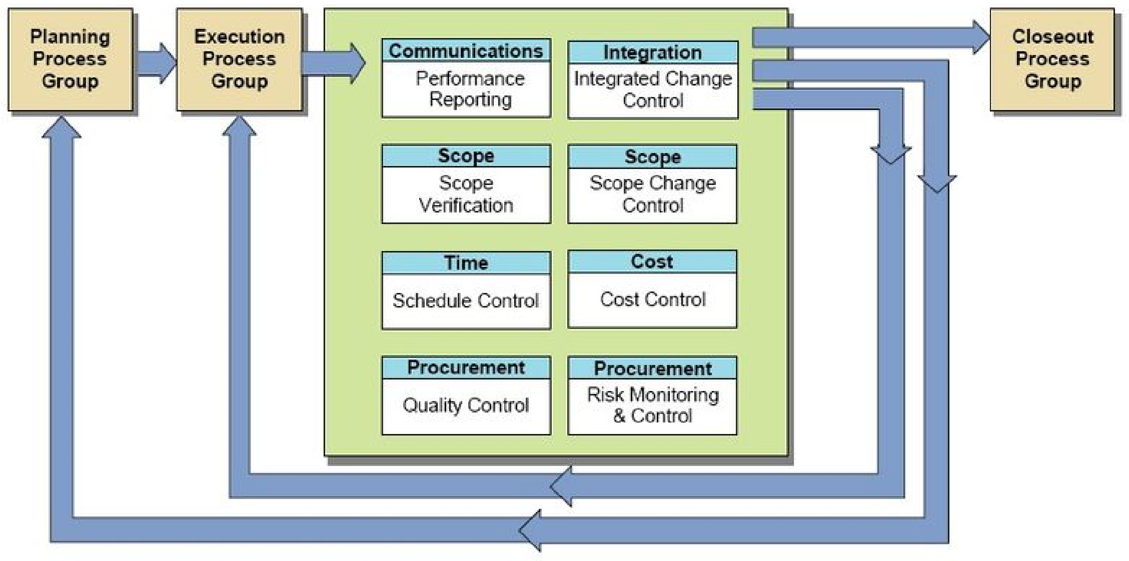


Executing Process Group Processes

Executing consists of the processes used to complete the work defined in the project management plan to accomplish the project's requirements. Execution process involves coordinating people and resources, as well as integrating and performing the activities of the project in accordance with the project management plan. The deliverables are produced as outputs from the processes performed as defined in the project management plan.

Monitoring and controlling

Monitoring and controlling consists of those processes performed to observe project execution so that potential problems can be identified in a timely manner and corrective action can be taken, when necessary, to control the execution of the project. The key benefit is that project performance is observed and measured regularly to identify variances from the project management plan.



Monitoring and Controlling Process Group Processes

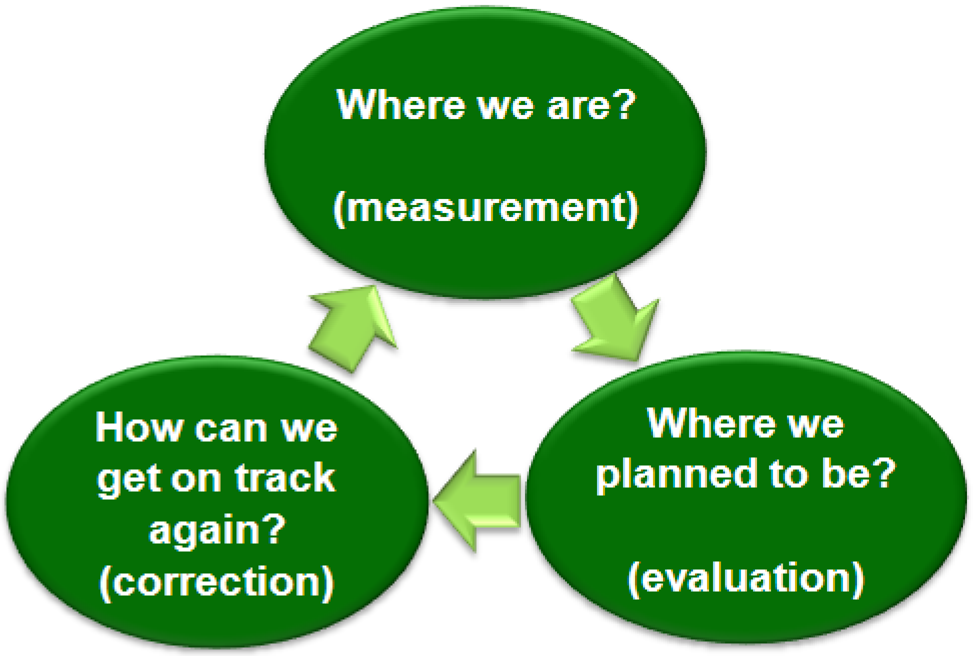
Monitoring and Controlling includes:

- Measuring the ongoing project activities ('where we are');
- Monitoring the project variables (cost, effort, scope, etc.) against the project management plan and the project performance baseline (*where we should be*);
- Identify corrective actions to address issues and risks properly (*How can we get on track again*);
- Influencing the factors that could circumvent integrated change control so only approved changes are implemented

In multi-phase projects, the monitoring and control process also provides feedback between project phases, in order to implement corrective or preventive actions to bring the project into compliance with the project management plan.

Project Maintenance is an ongoing process, and it includes:

- Continuing support of end users
- Correction of errors
- Updates of the software over time



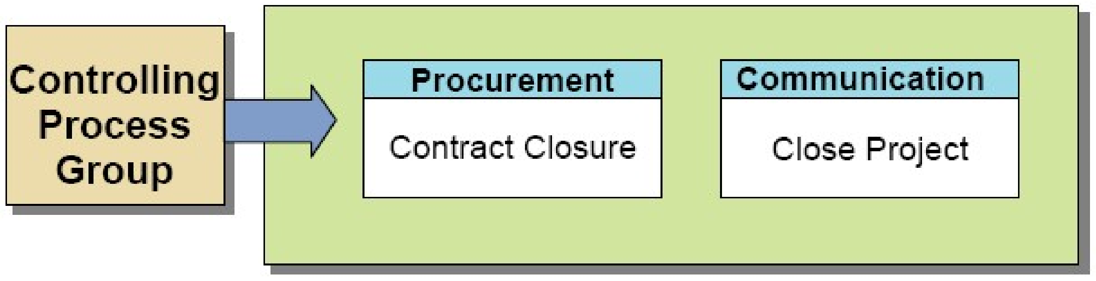
Monitoring and Controlling cycle

In this stage, auditors should pay attention to how effectively and quickly user problems are resolved.

Over the course of any construction project, the work scope may change. Change is a normal and expected part of the construction process. Changes can be the result of necessary design modifications, differing site conditions, material availability, contractor-requested changes, value engineering and impacts from third parties, to name a few. Beyond executing the change in the field, the change normally needs to be documented to show what was actually constructed. This is referred to as Change Management. Hence, the owner usually requires a final record to show all changes or, more specifically, any change that modifies the tangible portions of the finished work. The record is made on the contract documents – usually, but not necessarily limited to, the design drawings. The end product of this effort is what the industry terms as-built drawings, or more simply, "as built." The requirement for providing them is a norm in construction contracts.

When changes are introduced to the project, the viability of the project has to be re-assessed. It is important not to lose sight of the initial goals and targets of the projects. When the changes accumulate, the forecasted result may not justify the original proposed investment in the project.

Closing



Closing Process Group Processes

Closing includes the formal acceptance of the project and the ending thereof. Administrative activities include the archiving of the files and documenting lessons learned.

This phase consists of:

- **Project close**: Finalize all activities across all of the process groups to formally close the project or a project phase
- **Contract closure**: Complete and settle each contract (including the resolution of any open items) and close each contract applicable to the project or project phase.

Project control systems

Project control is that element of a project that keeps it on-track, on-time and within budget. Project control begins early in the project with planning and ends late in the project with post-implementation review, having a thorough involvement of each step in the process. Each project should be assessed for the appropriate level of control needed: too much control is too time consuming, too little control is very risky. If project control is not implemented correctly, the cost to the business should be clarified in terms of errors, fixes, and additional audit fees.

Control systems are needed for cost, risk, quality, communication, time, change, procurement, and human resources. In addition, auditors should consider how important the projects are to the financial statements, how reliant the stakeholders are on controls, and how many controls exist. Auditors should review the development process and procedures for how they are implemented. The process of development and the quality of the final product may also be assessed if needed or requested. A business may want the auditing firm to be involved throughout the process to catch problems earlier on so that they can be fixed more easily. An auditor can serve as a controls consultant as part of the development team or as an independent auditor as part of an audit.

Businesses sometimes use formal systems development processes. These help assure that systems are developed successfully. A formal process is more effective in creating strong controls, and auditors should review this process to confirm that it is well designed and is followed in practice. A good formal systems development plan outlines:

- A strategy to align development with the organization’s broader objectives
- Standards for new systems
- Project management policies for timing and budgeting
- Procedures describing the process
- Evaluation of quality of change

Topics

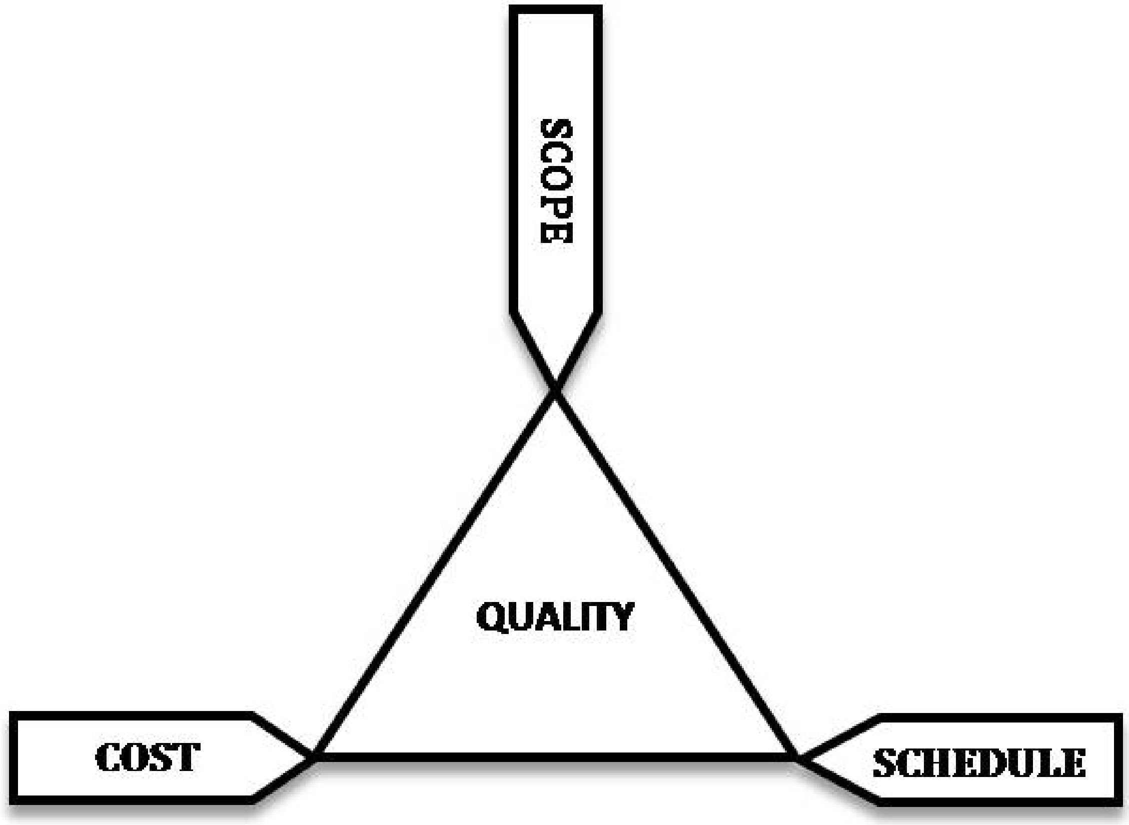
Project managers

A project manager is a professional in the field of project management. Project managers can have the responsibility of the planning, execution, and closing of any project, typically relating to construction industry, engineering, architecture, computing, or telecommunications. Many other fields in the production engineering and design engineering and heavy industrial also have project managers.

A project manager is the person accountable for accomplishing the stated project objectives. Key project management responsibilities include creating clear and attainable project objectives, building the project requirements, and managing the triple constraint for projects, which is cost, time, and scope.

A project manager is often a client representative and has to determine and implement the exact needs of the client, based on knowledge of the firm they are representing. The ability to adapt to the various internal procedures of the contracting party, and to form close links with the nominated representatives, is essential in ensuring that the key issues of cost, time, quality and above all, client satisfaction, can be realized.

Project Management Triangle



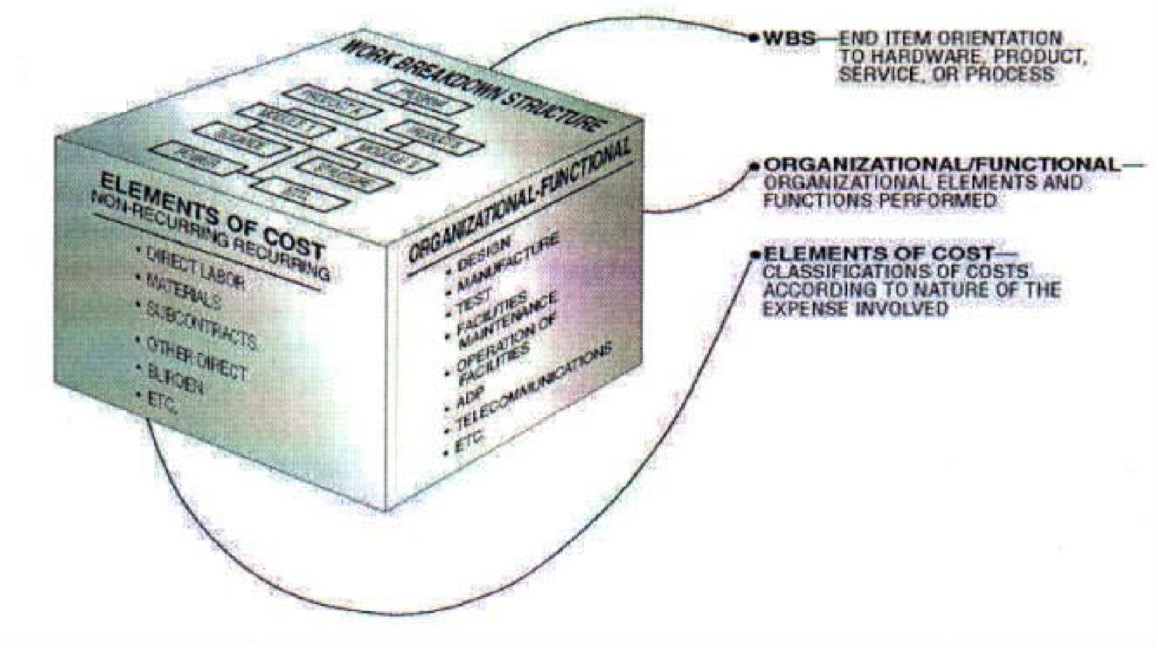
The Project Management Triangle

Like any human undertaking, projects need to be performed and delivered under certain constraints. Traditionally, these constraints have been listed as "scope," "time," and "cost". These are also referred to as the "Project Management Triangle", where each side represents a constraint. One side of the triangle cannot be changed without affecting the others. A further refinement of the constraints separates product "quality" or "performance" from scope, and turns quality into a fourth constraint.

The time constraint refers to the amount of time available to complete a project. The cost constraint refers to the budgeted amount available for the project. The scope constraint refers to what must be done to produce the project's end result. These three constraints are often competing constraints: increased scope typically means increased time and increased cost, a tight time constraint could mean increased costs and reduced scope, and a tight budget could mean increased time and reduced scope.

The discipline of Project Management is about providing the tools and techniques that enable the project team (not just the project manager) to organize their work to meet these constraints.

Work Breakdown Structure


WORK BREAKDOWN STRUCTURE
ELEMENTS OF COST
NON-RECURRING RECURRING
• DIRECT LABOR
• MATERIALS
• SUBCONTRACTS
• OTHER DIRECT
• BURDEN
• ETC.
ORGANIZATIONAL-FUNCTIONAL
• DESIGN
• MANUFACTURE
• TEST
• FACILITIES MAINTENANCE
• OPERATION OF FACILITIES
• ADP
• TELECOMMUNICATIONS
• ETC.
• WBS—END ITEM ORIENTATION TO HARDWARE, PRODUCT, SERVICE, OR PROCESS
• ORGANIZATIONAL/FUNCTIONAL—ORGANIZATIONAL ELEMENTS AND FUNCTIONS PERFORMED
• ELEMENTS OF COST—CLASSIFICATIONS OF COSTS ACCORDING TO NATURE OF THE EXPENSE INVOLVED

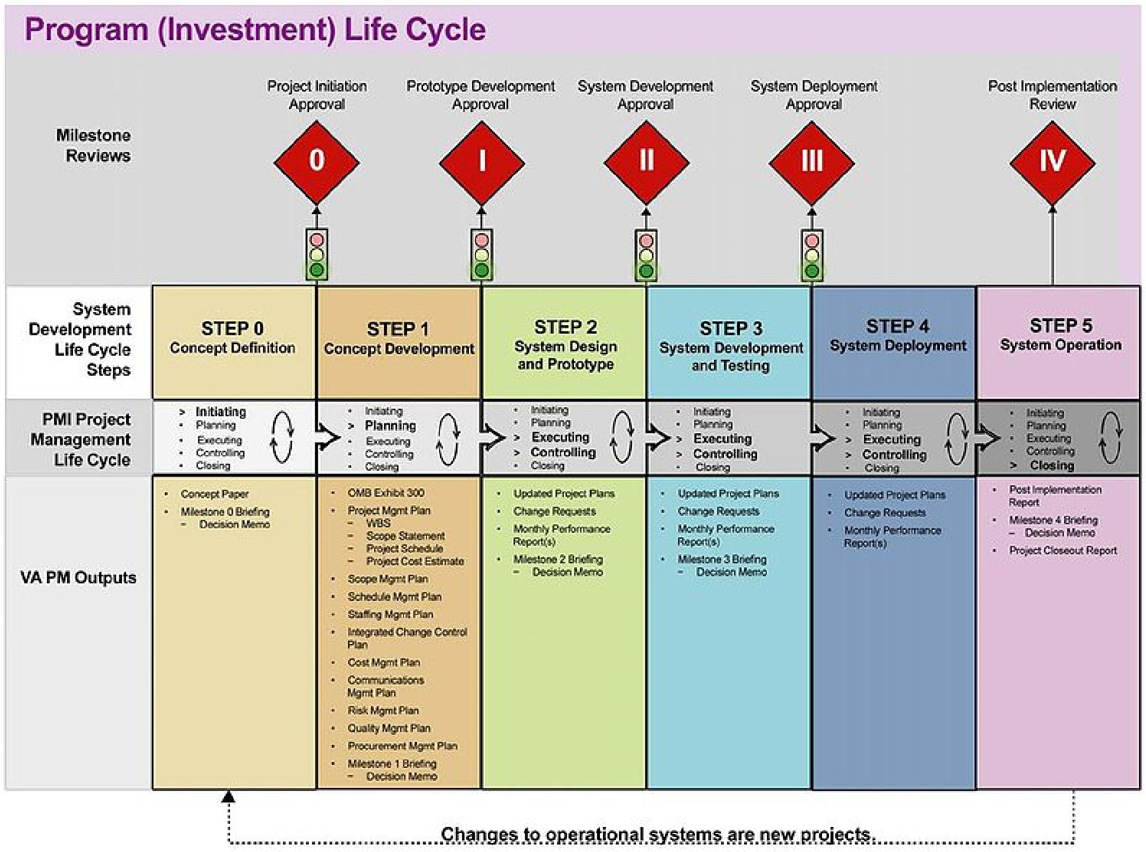
Example of a Work breakdown structure applied in a NASA reporting structure

The Work Breakdown Structure (WBS) is a tree structure, which shows a subdivision of effort required to achieve an objective; for example a program, project, and contract. The WBS may be hardware, product, service, or process oriented.

A WBS can be developed by starting with the end objective and successively subdividing it into manageable components in terms of size, duration, and responsibility (e.g., systems, subsystems, components, tasks, subtasks, and work packages), which include all steps necessary to achieve the objective.

The Work Breakdown Structure provides a common framework for the natural development of the overall planning and control of a contract and is the basis for dividing work into definable increments from which the statement of work can be developed and technical, schedule, cost, and labor hour reporting can be established.

Project Management Framework



Program (Investment) Life Cycle
Milestone Reviews
Project Initiation Approval
Prototype Development Approval
System Development Approval
System Deployment Approval
Post Implementation Review
0
I
II
III
IV
System Development Life Cycle Steps
STEP 0 Concept Definition
STEP 1 Concept Development
STEP 2 System Design and Prototype
STEP 3 System Development and Testing
STEP 4 System Deployment
STEP 5 System Operation
PMI Project Management Life Cycle
> Initiating • Planning • Executing • Controlling • Closing
• Initiating > Planning • Executing • Controlling • Closing
• Initiating • Planning > Executing > Controlling • Closing
• Initiating • Planning > Executing > Controlling • Closing
• Initiating • Planning > Executing > Controlling • Closing
• Initiating • Planning • Executing • Controlling > Closing
VA PM Outputs
• Concept Paper • Milestone 0 Briefing – Decision Memo
• OMB Exhibit 300 • Project Mgmt Plan – WBS – Scope Statement – Project Schedule – Project Cost Estimate • Scope Mgmt Plan • Schedule Mgmt Plan • Staffing Mgmt Plan • Integrated Change Control Plan • Cost Mgmt Plan • Communications Mgmt Plan • Risk Mgmt Plan • Quality Mgmt Plan • Procurement Mgmt Plan • Milestone 1 Briefing – Decision Memo
• Updated Project Plans • Change Requests • Monthly Performance Report(s) • Milestone 2 Briefing – Decision Memo
• Updated Project Plans • Change Requests • Monthly Performance Report(s) • Milestone 3 Briefing – Decision Memo
• Updated Project Plans • Change Requests • Monthly Performance Report(s)
• Post Implementation Report • Milestone 4 Briefing – Decision Memo • Project Closeout Report
Changes to operational systems are new projects.

Example of an IT Project Management Framework

The Program (Investment) Life Cycle integrates the project management and system development life cycles with the activities directly associated with system deployment and operation. By design, system operation management and related activities occur after the project is complete and are not documented within this guide.

For example, see figure, in the US United States Department of Veterans Affairs (VA) the program management life cycle is depicted and describe in the overall VA IT Project Management Framework to address the integration of OMB Exhibit 300 project (investment) management activities and the overall project budgeting process. The VA IT Project Management Framework diagram illustrates Milestone 4 which occurs following the deployment of a system and the closing of the project. The project closing phase activities at the VA continues through system deployment and into system operation for the purpose of illustrating and describing the system activities the VA considers part of the project. The figure illustrates the actions and associated artifacts of the VA IT Project and Program Management process.

International standards

There have been several attempts to develop Project Management standards, such as:

- Capability Maturity Model from the Software Engineering Institute.
- GAPPS, Global Alliance for Project Performance Standards- an open source standard describing COMPETENCIES for project and program managers.
- A Guide to the Project Management Body of Knowledge
- HERMES method, Swiss general project management method, selected for use in Luxembourg and international organizations.
- The ISO standards ISO 9000, a family of standards for quality management systems, and the ISO 10006:2003, for Quality management systems and guidelines for quality management in projects.
- PRINCE2, PRojects IN Controlled Environments.
- Team Software Process (TSP) from the Software Engineering Institute.
- Total Cost Management Framework, AACE International's Methodology for Integrated Portfolio, Program and Project Management)
- V-Model, an original systems development method.
- The Logical framework approach, which is popular in international development organizations.
- IAPPM, The International Association of Project & Program Management, guide to Project Auditing and Rescuing Troubled Projects.

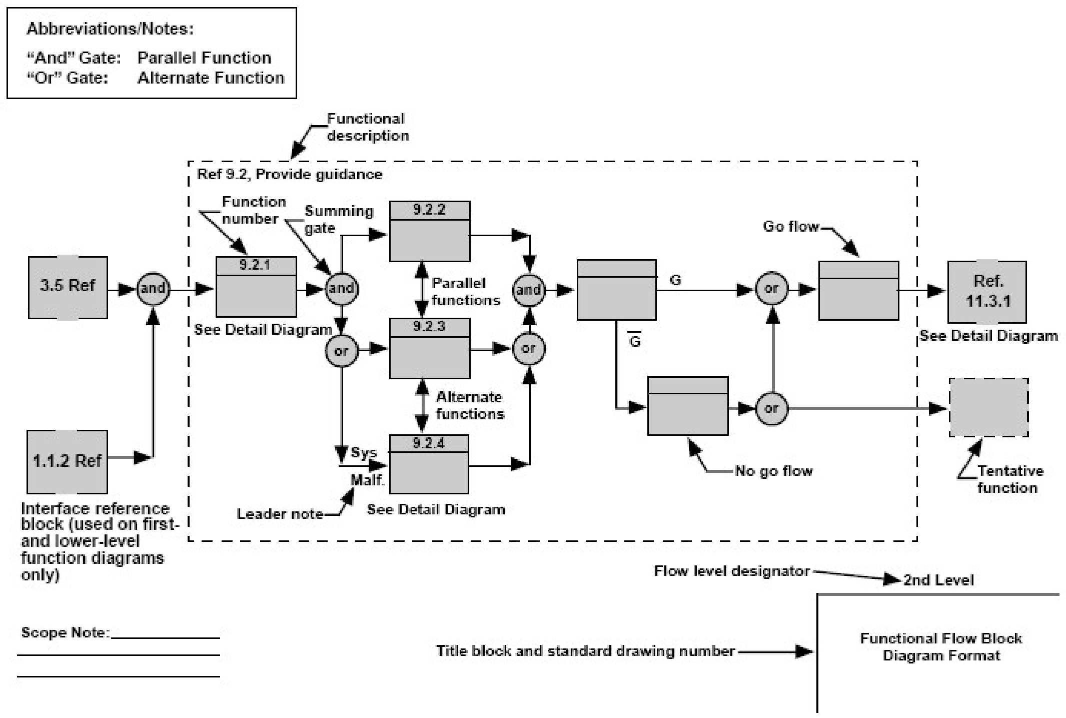
Project portfolio management

An increasing number of organizations are using, what is referred to as, project portfolio management (PPM) as a means of selecting the right projects and then using project management techniques as the means for delivering the outcomes in the form of benefits to the performing private or not-for-profit organization.

Project management methods are used 'to do projects right' and the methods used in PPM are used 'to do the right projects'. In effect PPM is becoming the method of choice for selection and prioritising among resource inter-related projects in many industries and sectors.

Chapter- 5

Functional Flow Block Diagram



Functional Flow Block Diagram Format

A **Functional Flow Block Diagram** (FFBD) is a multi-tier, time-sequenced, step-by-step flow diagram of a system's functional flow.

The FFBD notation was developed in the 1950s, and is widely used in classical systems engineering. FFBDs are one of the classic business process modeling methodologies, along with flow charts, data flow diagrams, control flow diagrams, Gantt charts, PERT diagrams, and IDEF.

FFBDs are also referred to as *Functional Flow Diagrams*, *functional block diagrams*, and *functional flows*.

History

The first structured method for documenting process flow, the flow process chart, was introduced by Frank Gilbreth to members of American Society of Mechanical Engineers (ASME) in 1921 as the presentation "Process Charts—First Steps in Finding the One Best Way". Gilbreth's tools quickly found their way into industrial engineering curricula. In the early 1930s, an industrial engineer, Allan H. Mogensen began training business people in the use of some of the tools of industrial engineering at his Work Simplification Conferences in Lake Placid, New York. A 1944 graduate of Mogensen's class, Art Spinanger, took the tools back to Procter and Gamble where he developed their Deliberate Methods Change Program. Another 1944 graduate, Ben S. Graham, Director of Formcraft Engineering at Standard Register Corporation, adapted the flow process chart to information processing with his development of the multi-flow process chart to displays multiple documents and their relationships. In 1947, ASME adopted a symbol set as the ASME Standard for Operation and Flow Process Charts, derived from Gilbreth's original work.

The modern FFBD was developed by TRW Incorporated, a defense-related business, in the 1950s. In the 1960s it was exploited by NASA to visualize the time sequence of events in space systems and flight missions. FFBDs became widely used in classical systems engineering to show the order of execution of system functions.

Development of functional flow block diagrams

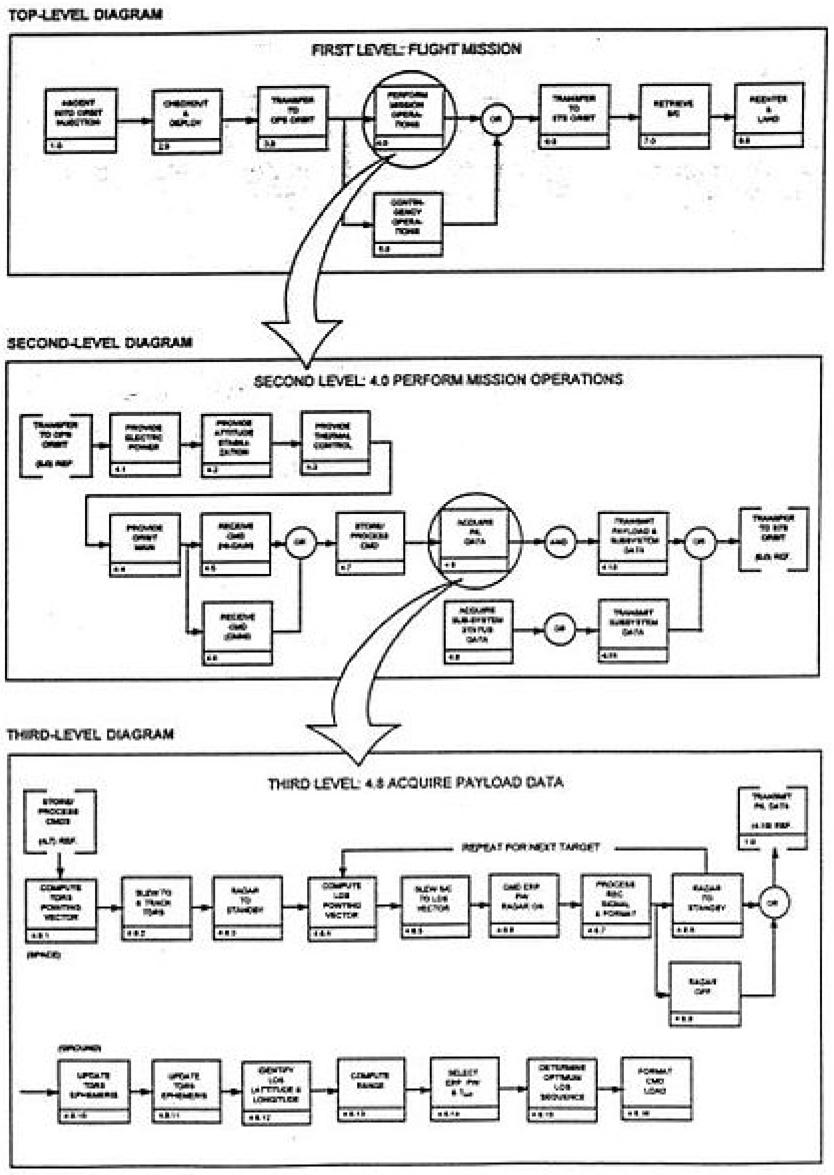

TOP-LEVEL DIAGRAM
FIRST LEVEL: FLIGHT MISSION
OR
SECOND-LEVEL DIAGRAM
SECOND LEVEL: 4.0 PERFORM MISSION OPERATIONS
OR
AND
THIRD-LEVEL DIAGRAM
THIRD LEVEL: 4.8 ACQUIRE PAYLOAD DATA
REPEAT FOR NEXT TARGET
OR

Figure 2: Development of Functional Flow Block Diagrams

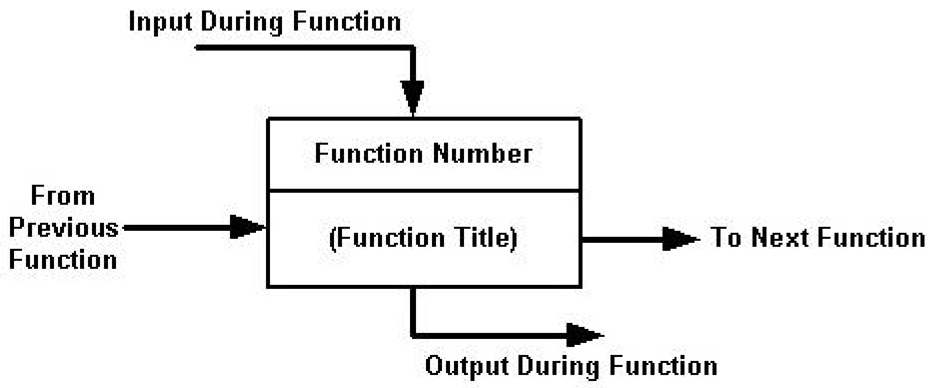
FFBDs can be developed in a series of levels. FFBDs show the same tasks identified through functional decomposition and display them in their logical, sequential relationship. For example, the entire flight mission of a spacecraft can be defined in a top level FFBD, as shown in Figure 2. Each block in the first level diagram can then be expanded to a series of functions, as shown in the second level diagram for "perform mission operations." Note that the diagram shows both input (transfer to operational orbit) and output (transfer to space transportation system orbit), thus initiating the interface identification and control process. Each block in the second level diagram can be progressively developed into a series of functions, as shown in the third level diagram on Figure 2.

These diagrams are used both to develop requirements and to identify profitable trade studies. For example, does the spacecraft antenna acquire the tracking and data relay satellite (TDRS) only when the payload data are to be transmitted, or does it track TDRS continually to allow for the reception of emergency commands or transmission of emergency data? The FFBD also incorporates alternate and contingency operations, which improve the probability of mission success. The flow diagram provides an understanding of total operation of the system, serves as a basis for development of operational and contingency procedures, and pinpoints areas where changes in operational procedures could simplify the overall system operation. In certain cases, alternate FFBDs may be used to represent various means of satisfying a particular function until data are acquired, which permits selection among the alternatives.

Building blocks

Key attributes

An overview of the key FFBD attributes:



Graphical explanation of a "function block" used in these diagrams. Flow is from left to right.

- *Function block*: Each function on an FFBD should be separate and be represented by single box (solid line). Each function needs to stand for definite, finite, discrete action to be accomplished by system elements.
- *Function numbering*: Each level should have a consistent number scheme and provide information concerning function origin. These numbers establish identification and relationships that will carry through all Functional Analysis and Allocation activities and facilitate traceability from lower to top levels.
- *Functional reference*: Each diagram should contain a reference to other functional diagrams by using a functional reference (box in brackets).
- *Flow connection*: Lines connecting functions should only indicate function flow and not a lapse in time or intermediate activity.

- *Flow direction*: Diagrams should be laid out so that the flow direction is generally from left to right. Arrows are often used to indicate functional flows.
- *Summing gates*: A circle is used to denote a summing gate and is used when AND/OR is present. AND is used to indicate parallel functions and all conditions must be satisfied to proceed. OR is used to indicate that alternative paths can be satisfied to proceed.
- *GO and NO-GO paths*: “G” and “bar G” are used to denote “go” and “no-go” conditions. These symbols are placed adjacent to lines leaving a particular function to indicate alternative paths.

Function symbolism

A function shall be represented by a rectangle containing the title of the function (an action verb followed by a noun phrase) and its unique decimal delimited number. A horizontal line shall separate this number and the title, as shown in see Figure 3 above. The figure also depicts how to represent a reference function, which provides context within a specific FFBD. See Figure 9 for an example regarding use of a reference function.

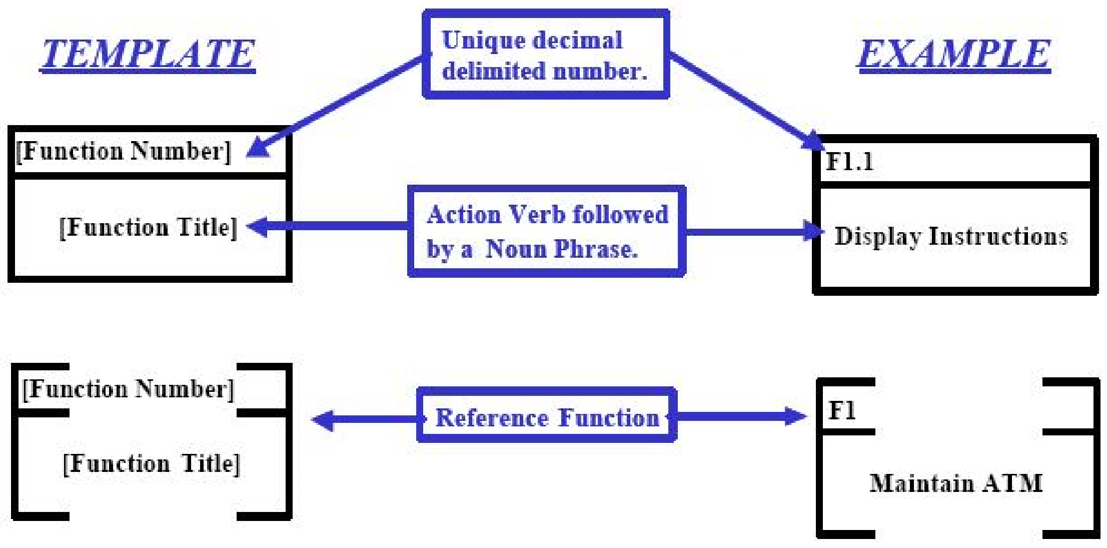


Figure 3. Function Symbol

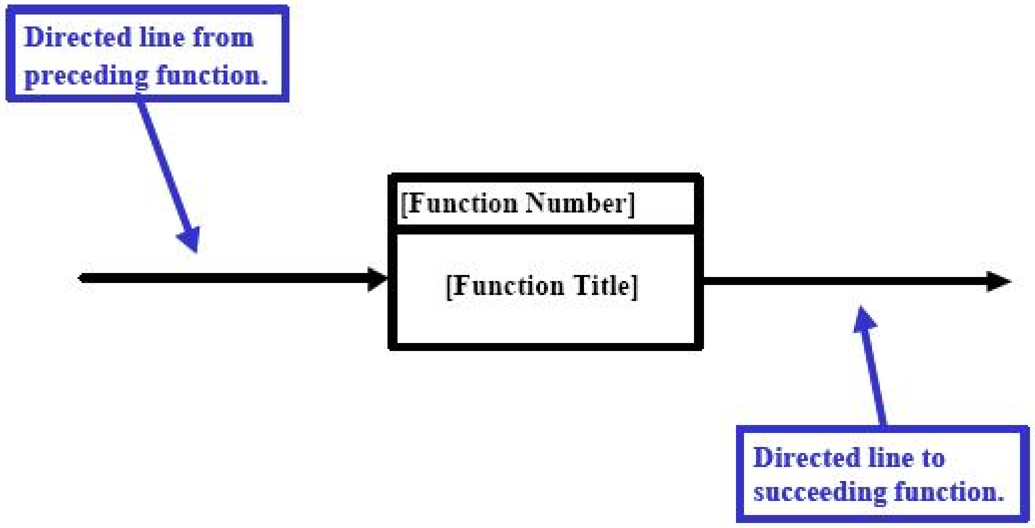

Directed line from preceding function.
[Function Number]
[Function Title]
Directed line to succeeding function.

Figure 4. Directed Lines

Directed lines

A line with a single arrowhead shall depict functional flow from left to right, see Figure 4.

Logic Symbols

The following basic logic symbols shall be used.

- AND: A condition in which all preceding or succeeding paths are required. The symbol may contain a single input with multiple outputs or multiple inputs with a single output, but not multiple inputs and outputs combined (Figure 5). Read the figure as follows: F2 AND F3 may begin in parallel after completion of F1. Likewise, F4 may begin after completion of F2 AND F3.

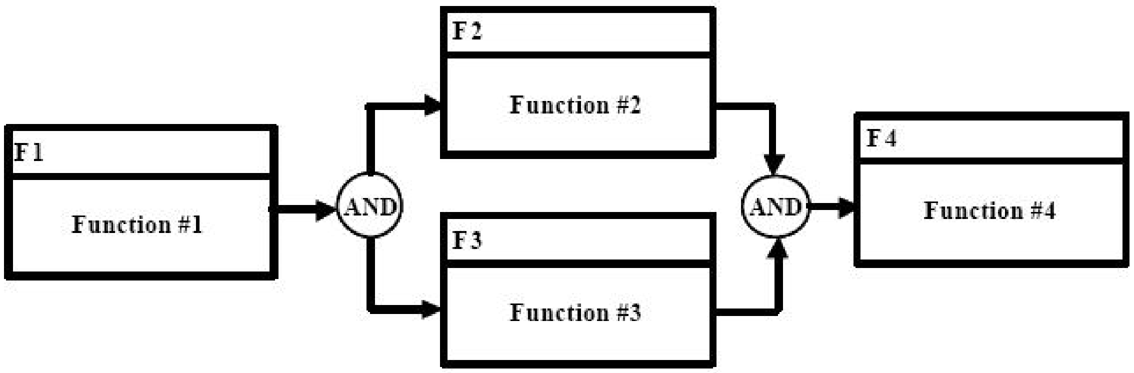


Figure 5. "AND" Symbol

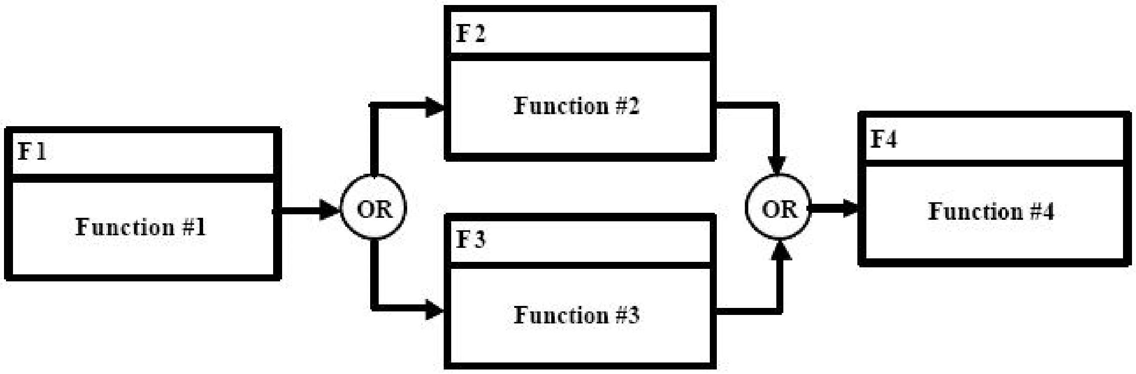


Figure 6. "Exclusive OR" Symbol

- Exclusive OR: A condition in which one of multiple preceding or succeeding paths is required, but not all. The symbol may contain a single input with multiple outputs or multiple inputs with single output, but not multiple inputs and outputs combined (Figure 6). Read the figure as follows: F2 OR F3 may begin after completion of F1. Likewise, F4 may begin after completion of either F2 OR F3.

- Inclusive OR: A condition in which one, some, or all of the multiple preceding or succeeding paths are required. Figure 7 depicts Inclusive OR logic using a combination of the AND symbol (Figure 5) and the Exclusive OR symbol (Figure 6). Read Figure 7 as follows: F2 OR F3 (exclusively) may begin after completion of F1, OR (again exclusive) F2 AND F3 may begin after completion of F1. Likewise, F4 may begin after completion of either F2 OR F3 (exclusively), OR (again exclusive) F4 may begin after completion of both F2 AND F3

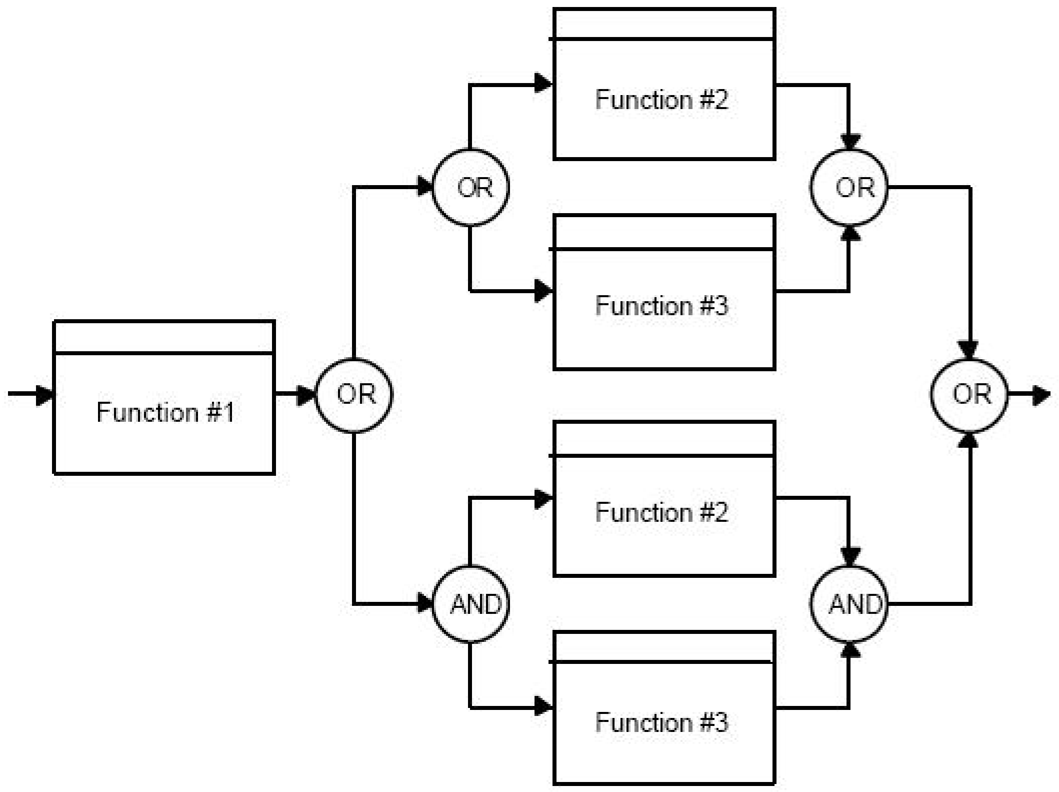


Figure 7. “Inclusive OR” Logic

Contextual and Administrative Data

Each FFBD shall contain the following contextual and administrative data:

- Date the diagram was created
- Name of the engineer, organization, or working group that created the diagram
- Unique decimal delimited number of the function being diagrammed
- Unique function name of the function being diagrammed.

Figure 8 and Figure 9 present the data in an FFBD. Figure 9 is a decomposition of the function F2 contained in Figure 8 and illustrates the context between functions at different levels of the model.

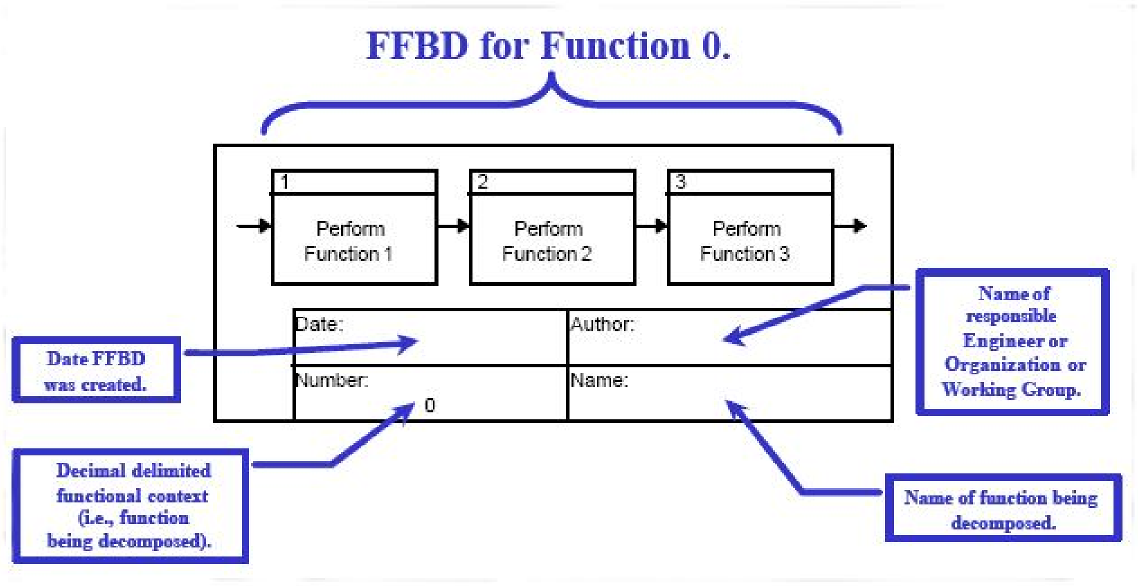

FFBD for Function 0.
1
Perform Function 1
2
Perform Function 2
3
Perform Function 3
Date:
Author:
Number:
0
Name:
Date FFBD was created.
Name of responsible Engineer or Organization or Working Group.
Decimal delimited functional context (i.e., function being decomposed).
Name of function being decomposed.

Figure 8. FFBD Function 0 Illustration

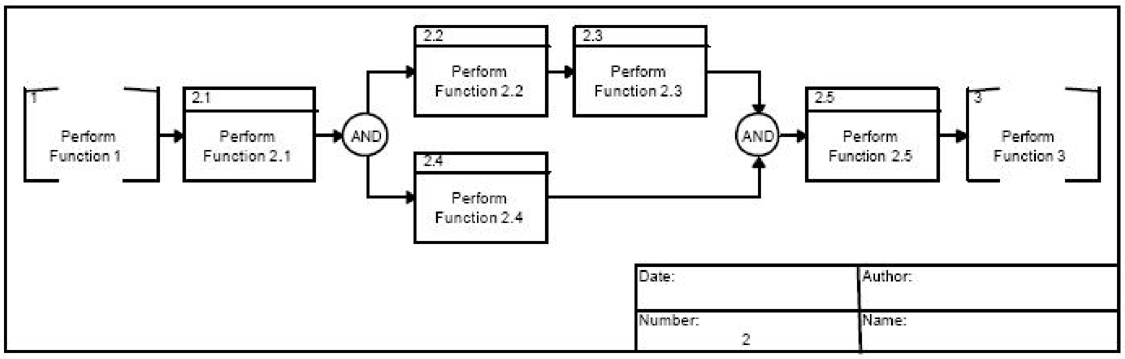
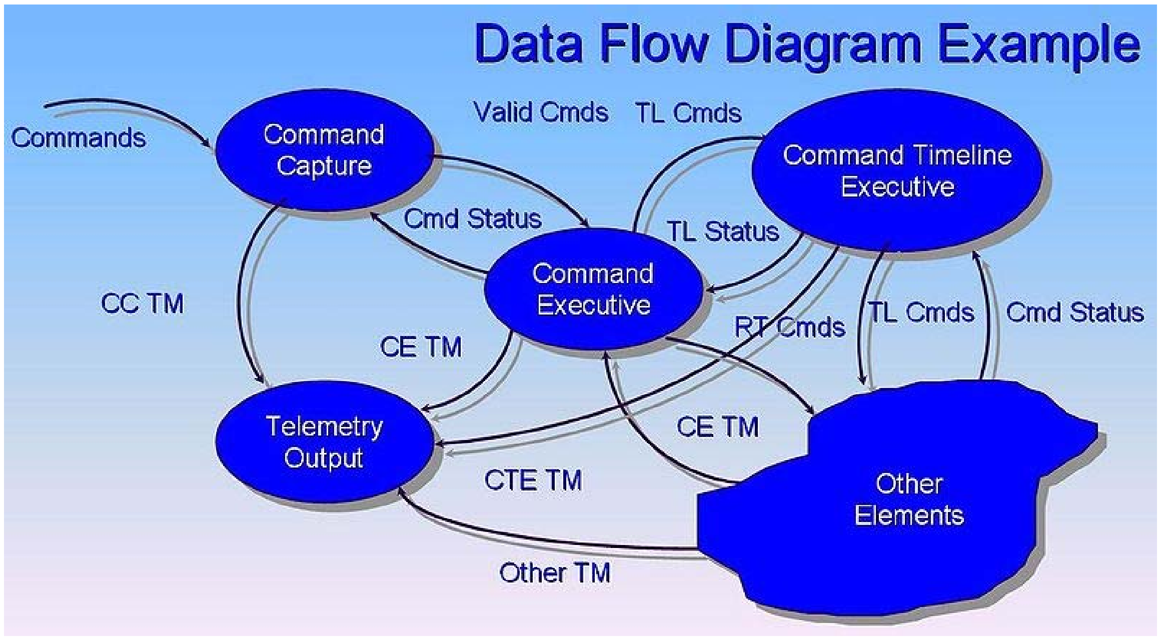


Figure 9. FFBD Function 2 Illustration

Chapter- 6

Data Flow Diagram and N2 Chart

Data flow diagram



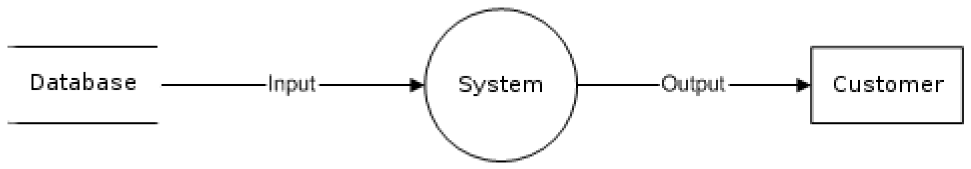
Data flow diagram example

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

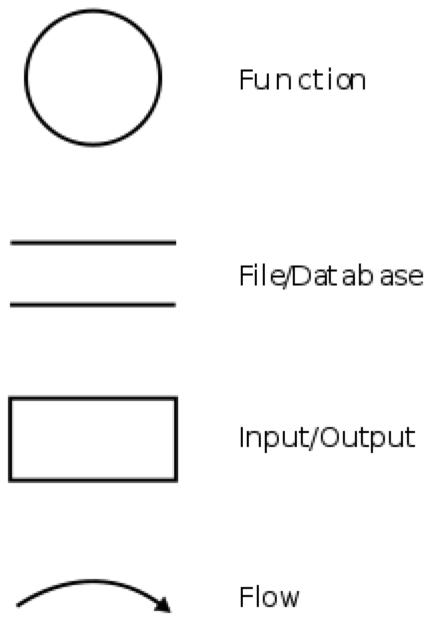
On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD).

Overview



Data flow diagram example



Data flow diagram - Yourdon/DeMarco notation

It is common practice to draw a context-level data flow diagram first, which shows the interaction between the system and external agents which act as data sources and data sinks. On the context diagram (also known as the 'Level 0 DFD') the system's interactions with the outside world are modelled purely in terms of data flows across the *system boundary*. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization.

This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

Data flow diagrams were proposed by Larry Constantine, the original developer of structured design, based on Martin and Estrin's "data flow graph" model of computation.

Data flow diagrams (DFDs) are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's dataflow diagrams can be drawn up and compared with the new system's data flow diagrams to draw comparisons to implement a more efficient system. Data flow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to report. How any system is developed can be determined through a data flow diagram.

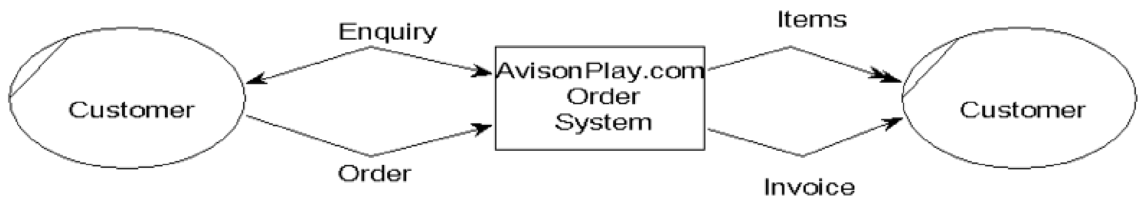
In the course of developing a set of *levelled* data flow diagrams the analyst/designers is forced to address how the system may be decomposed into component sub-systems, and to identify the transaction data in the data model.

There are different notations to draw data flow diagrams (Yourdon & Coad and Gane & Sarson), defining different visual representations for processes, data stores, data flow, and external entities.

Developing a data flow diagram

Event partitioning approach

Event partitioning was described by Edward Yourdon in *Just Enough Structured Analysis*.



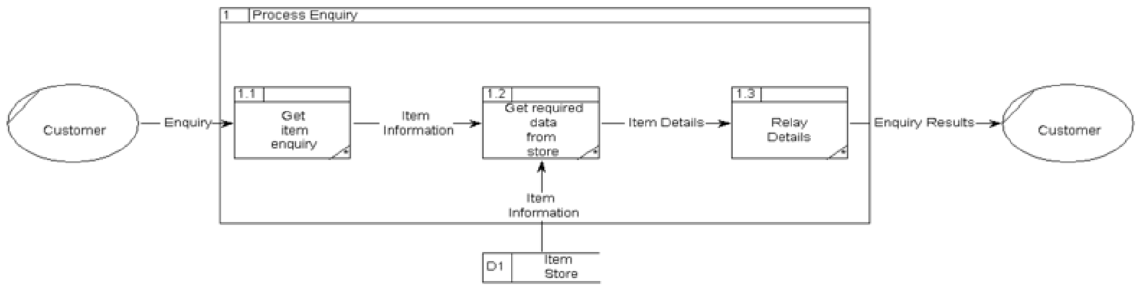
A context level Data flow diagram created using Select SSADM

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. It does not usually show data stores, unless they are "owned" by external systems, e.g. are accessed by but not maintained by this system, however, these are often shown as external entities.

Level 1 (high level diagram)

This level (level 1) shows all processes at the first level of numbering, data stores, external entities and the data flows between them. The purpose of this level is to show the major and high-level processes of the system and their model will have one, and only one, level-1 diagram. A level-1 diagram must be balanced with its parent context level diagram, i.e. there must be the same external entities and the same data flows, these can be broken down to more detail in the level 1, example the "enquiry" data flow could be split into "enquiry request" and "enquiry results" and still be valid. This is all about using your creativity.

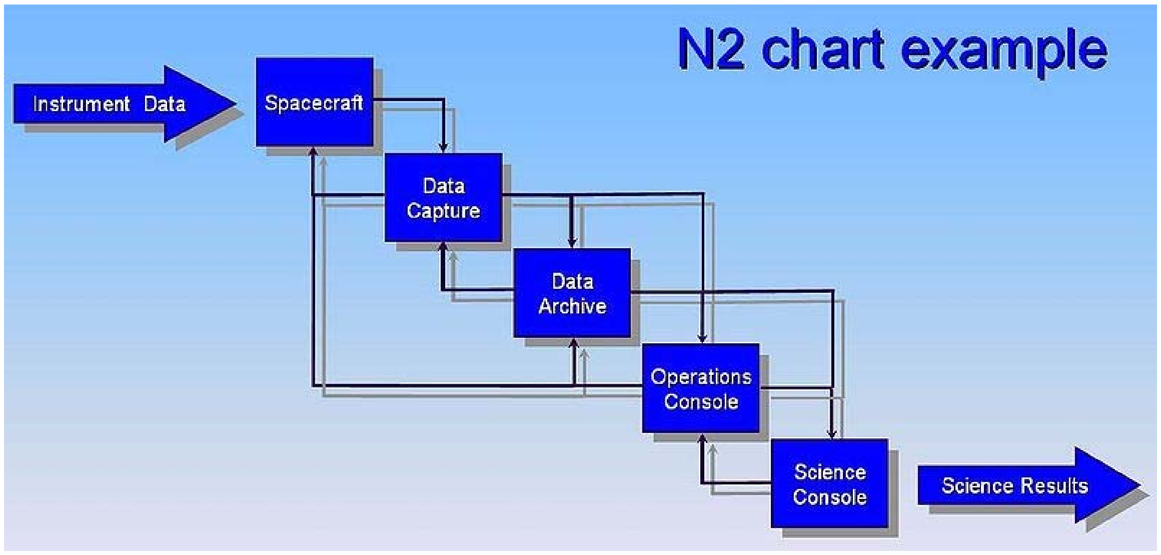
Level 2 (low level diagram)



A Level 2 Data flow diagram showing the "Process Enquiry" process for the same system.

This level is a decomposition of a process shown in a level-1 diagram, as such there should be a level-2 diagram for each and every process shown in a level-1 diagram. In this example, processes 1.1, 1.2 & 1.3 are all vimal of process 1. Together they wholly and completely describe process 1, and combined must perform the full capacity of this parent process. As before, a level-2 diagram must be balanced with its parent level-1 diagram.

N2 chart



N^2 chart example

The N^2 **chart**, also referred to as N^2 **diagram**, ***N*-squared diagram** or ***N*-squared chart**, is a diagram in the shape of a matrix, representing functional or physical interfaces between system elements. It is used to systematically identify, define, tabulate, design, and analyze functional and physical interfaces. It applies to system interfaces and hardware and/or software interfaces.

The *N*-squared chart was invented by the systems engineer Robert J. Lano, while working at TRW in the 1970s and first published in a 1977 TRW internal report.

Overview

The N^2 diagram has been used extensively to develop data interfaces, primarily in the software areas. However, it can also be used to develop hardware interfaces. The basic N^2 chart is shown in Figure 2. The system functions are placed on the diagonal; the remainder of the squares in the $N \times N$ matrix represent the interface inputs and outputs.

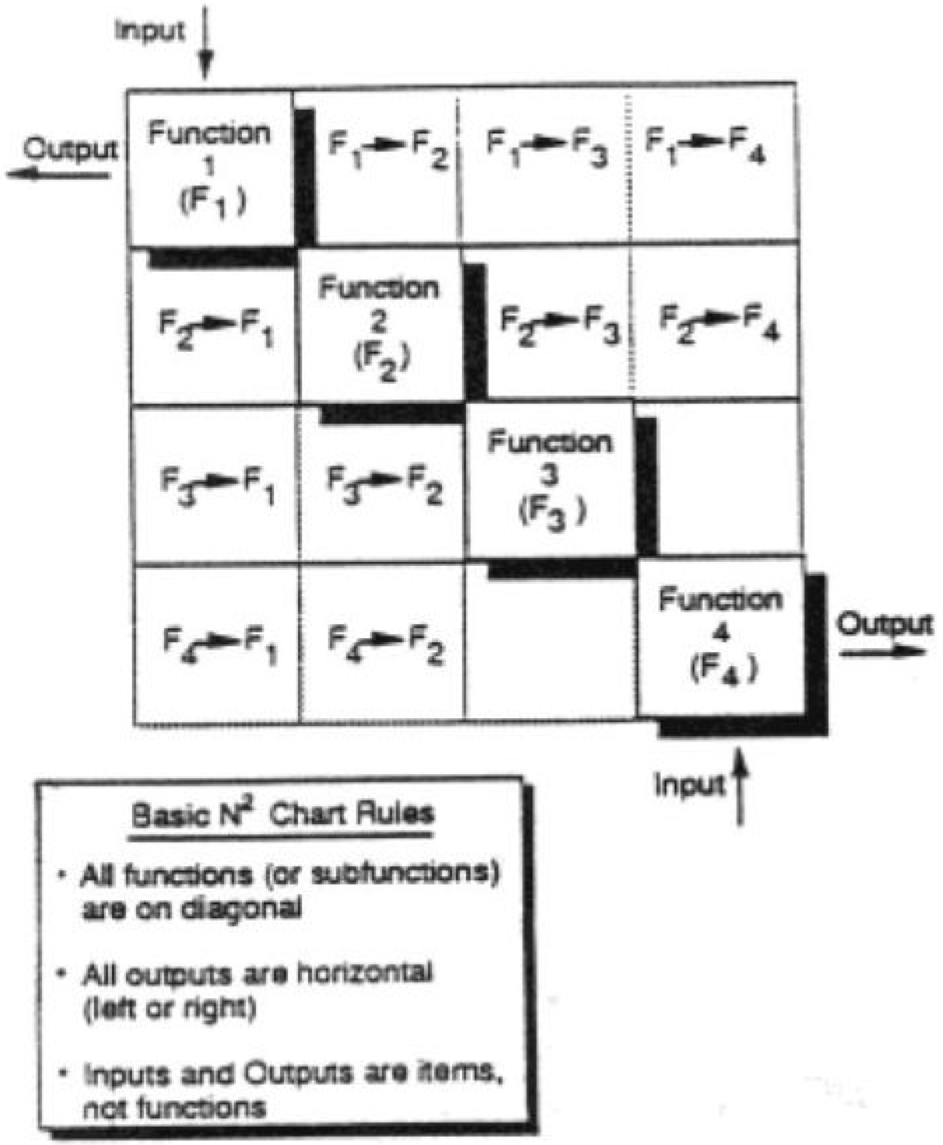


Figure 2. N2 chart definition

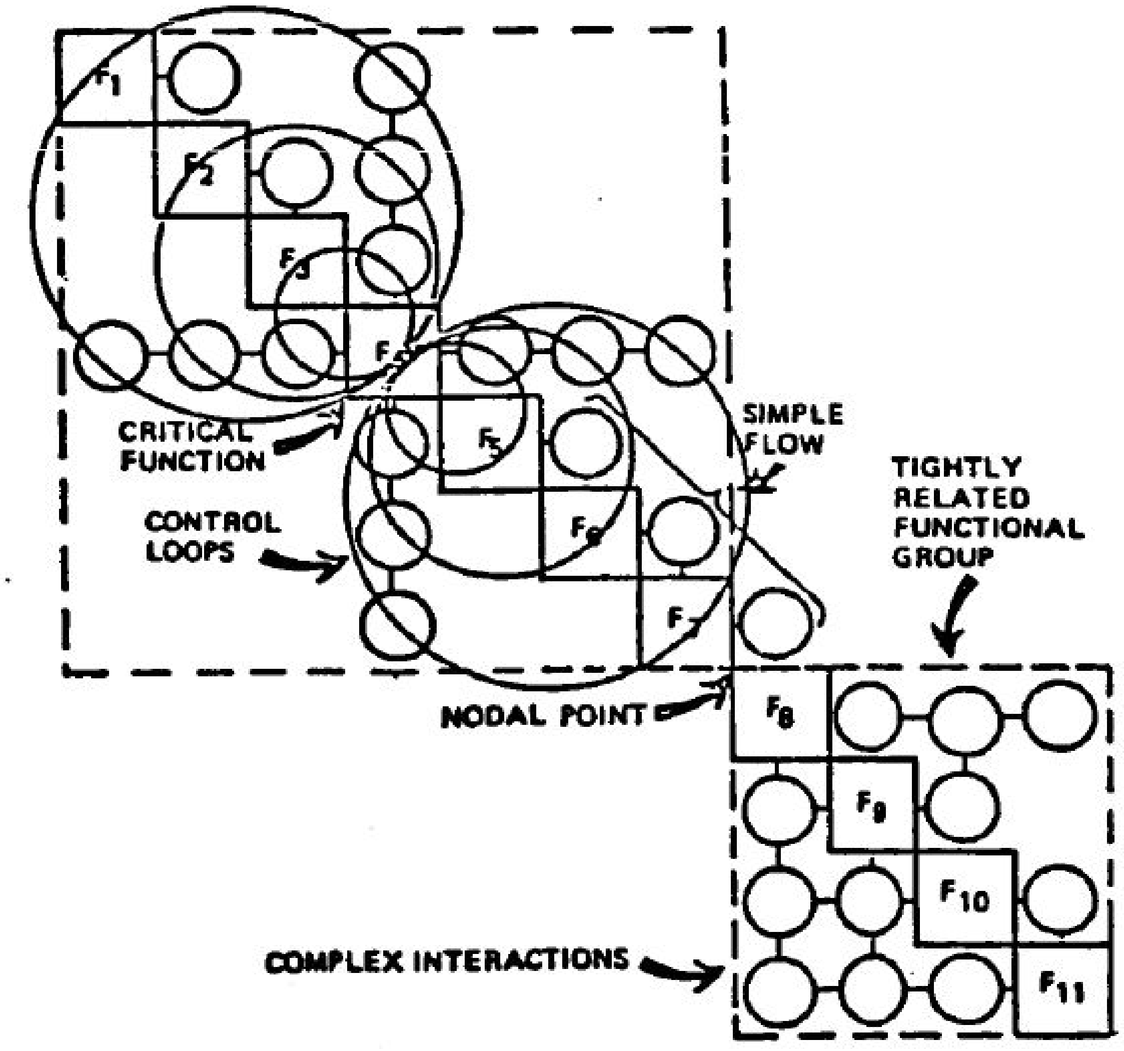


Figure 3. N2 Chart Key Features

Where a blank appears, there is no interface between the respective functions. Data flows in a clockwise direction between functions (e.g., the symbol F1 F2 indicates data flowing from function F1, to function F2). The data being transmitted can be defined in the appropriate squares. Alternatively, the use of circles and numbers permits a separate listing of the data interfaces. The clockwise flow of data between functions that have a feedback loop can be illustrated by a larger circle called a control loop. The identification of a critical function is also shown in Figure 3, where function F4 has a number of inputs and outputs to all other functions in the upper module. A simple flow of interface data exists between the upper and lower modules at functions F7 and F8. The lower module has complex interaction among its functions. The N2 chart can be taken down into successively lower levels to the hardware and software component functional levels. In addition to defining the data that must be supplied across the interface, the N2 chart can pinpoint areas where conflicts could arise.

N^2 *charts building blocks*

Number of entities

The "*N*" in an N^2 diagram is the number of entities for which relationships are shown. This $N \times N$ matrix requires the user to generate complete definitions of all interfaces in a rigid bidirectional, fixed framework. The user places the functional or physical entities on the diagonal axis and the interface inputs and outputs in the remainder of the diagram squares. A blank square indicates that there is no interface between the respective entities. Data flows clockwise between entities (i.e., the symbol F1 ? F2 in Figure 1 indicates data flowing from function F1 to function F2; the symbol F2 = F1 indicates the feedback). That which passes across the interface is defined in the appropriate squares.

The diagram is complete when the user has compared each entity to all other entities. The N2 diagram should be used in each successively lower level of entity decomposition. Figure 1 illustrates directional flow of interfaces between entities within an N^2 diagram. (In this case, the entities are functions.)

Functions on the diagonal

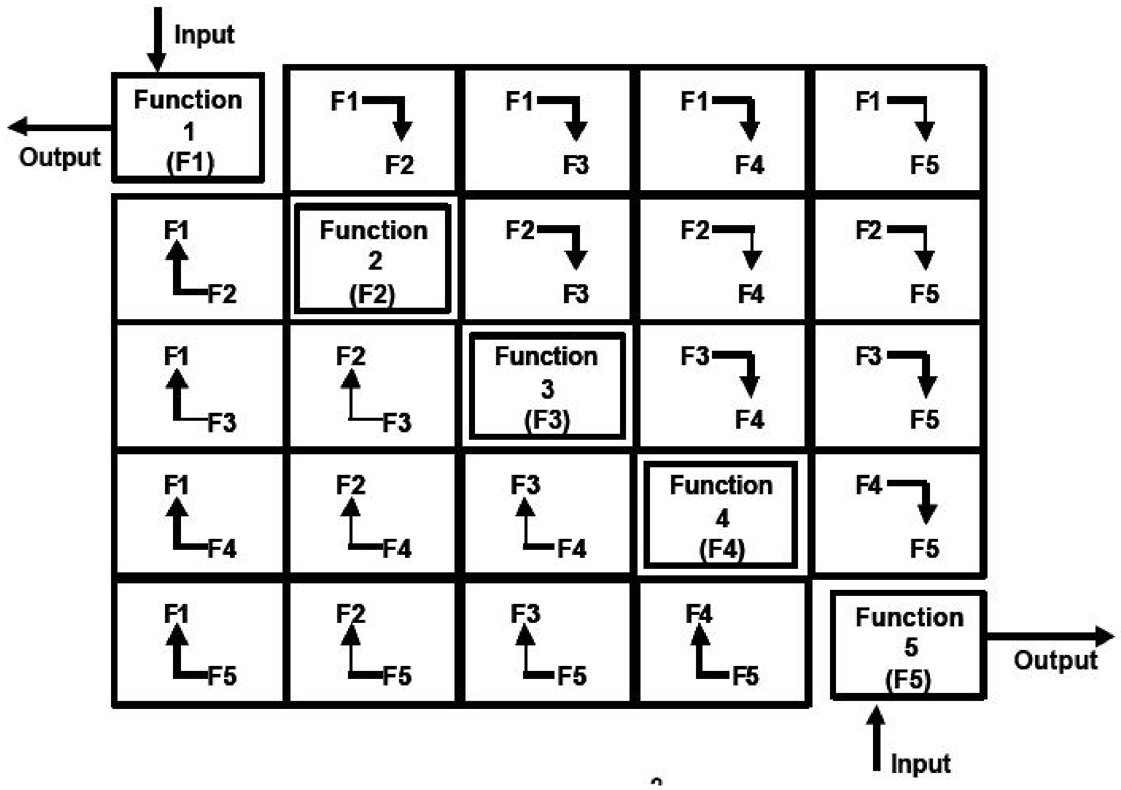


Figure 4. N^2 diagram

In the example on the right, *N* equals 5. The five functions are on the diagonal. The arrows show the flow of data between functions. So if function 1 sends data to function 2, the data elements would be placed in the box to the right of function 1. If function 1 does not send data to any of the other functions, the rest of the boxes to right of function 1 would be empty. If function 2 sends data to function 3 and function 5, then the data elements would be placed in the first and third boxes to the right of function 2. If any function sends data back to a previous function, then the associated box to the left of the function would have the data elements placed in it. The squares on either side of the diagonal (not just adjacent squares) are filled in with appropriate data to depict the flow between the functions. If there is no interface between two functions, the square that represents the interface between the two functions is left blank. Physical interfaces would be handled in the same manner, with the physical entities on the diagonal rather than the functional entities.

Contextual and administrative data

Each N^2 diagram shall contain at a minimum the following contextual and administrative data:

- Date the diagram was created
- Name of the engineer, organization, or working group that created the diagram
- Unique decimal delimited number of the functional or physical entity being diagrammed
- Unique name for the functional or physical entity being diagrammed

N2 diagrams are a valuable tool for not only identifying functional or physical interfaces, but also for pinpointing areas in which conflicts may arise with interfaces so that system integration proceeds smoothly and efficiently.

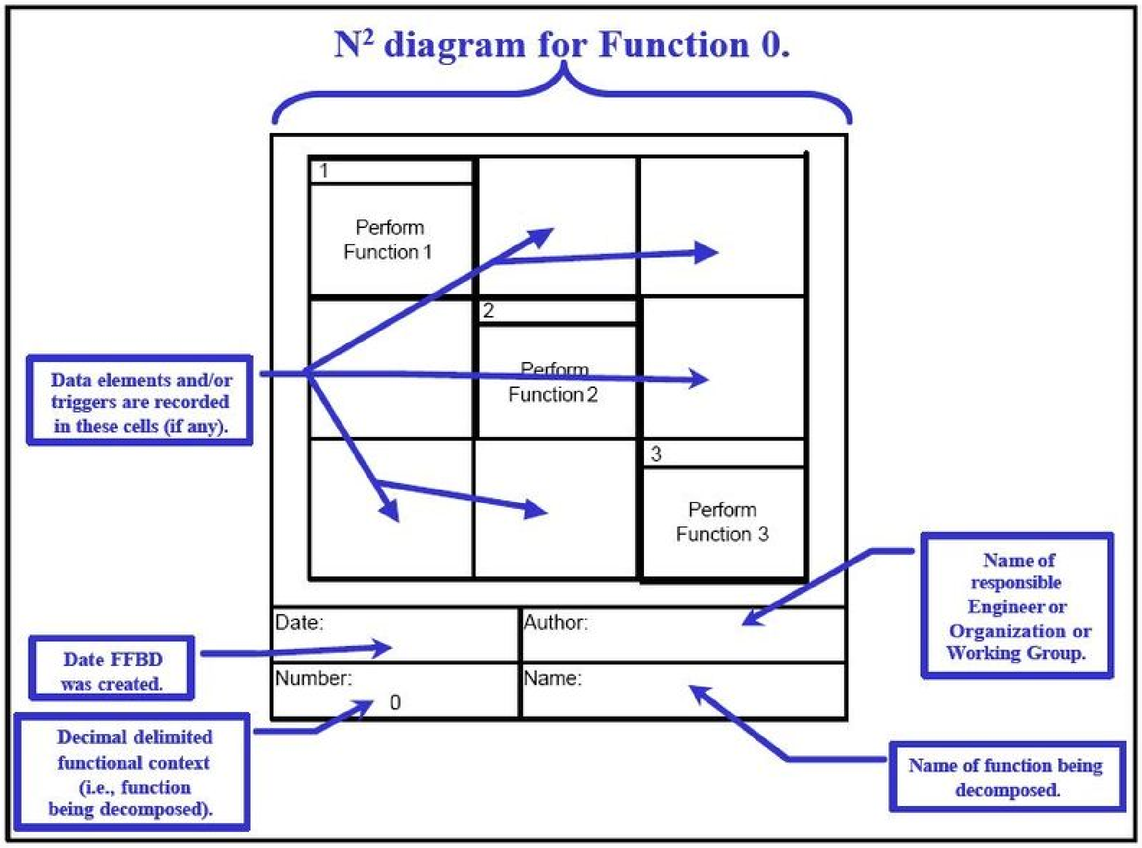
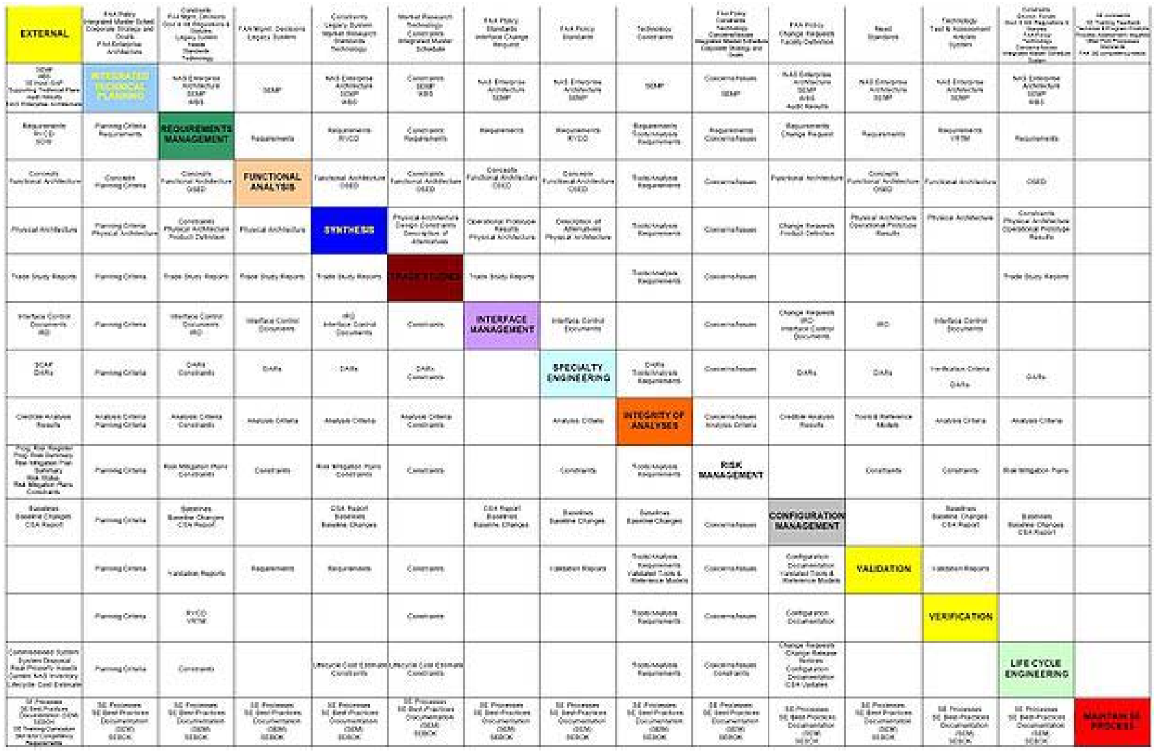

N² diagram for Function 0.
1
Perform Function 1
2
Perform Function 2
3
Perform Function 3
Data elements and/or triggers are recorded in these cells (if any).
Date:
Author:
Number:
0
Name:
Date FFBD was created.
Decimal delimited functional context (i.e., function being decomposed).
Name of responsible Engineer or Organization or Working Group.
Name of function being decomposed.

Figure 5 presents information in an N2 diagram, which complements the Functional flow block diagram. Notice that in this illustration, there are no data elements or triggers. The figure illustrates the context between functions at different levels of the model.

Examples

Figure 6 is an example of the diagram’s appearance when cells are populated with data.



Chapter- 7

Industrial Engineering

Industrial engineering is a branch of engineering dealing with the optimization of complex processes or systems. It is concerned with the development, improvement, implementation and evaluation of integrated systems of people, money, knowledge, information, equipment, energy, materials, analysis and synthesis, as well as the mathematical, physical and social sciences together with the principles and methods of engineering design to specify, predict, and evaluate the results to be obtained from such systems or processes. Its underlying concepts overlap considerably with certain business-oriented disciplines such as Operations Management, but the engineering side tends to emphasize extensive *mathematical* proficiency and usage of quantitative methods.

Depending on the sub-speciality(ies) involved, industrial engineering may also be known as operations management, management science, operations research, systems engineering, or manufacturing engineering, usually depending on the viewpoint or motives of the user. Recruiters or educational establishments use the names to differentiate themselves from others. In health care, industrial engineers are more commonly known as health management engineers or health systems engineers.

While the term originally applied to manufacturing, nowadays the term "industrial" in industrial engineering can be somewhat misleading. It has grown to encompass any methodical or quantitative approach to optimizing how a process, system, or organization operates. Some engineering universities and educational agencies around the world have changed the term “industrial” to the broader term “production”, leading to the typical extensions noted above. In fact, the primary U.S. professional organization for Industrial Engineers, the Institute of Industrial Engineers (IIE) has been considering changing its name to something broader (such as the Institute of Industrial & Systems Engineers), although the latest vote among membership deemed this unnecessary for the time being.

The various topics of concern to industrial engineers include management science, financial engineering, engineering management, supply chain management, process engineering, operations research, systems engineering, ergonomics, cost and value engineering, quality engineering, facilities planning, and the engineering design process. Traditionally, a major aspect of industrial engineering was planning the layouts of factories and designing assembly lines and other manufacturing paradigms. And now, in so-called lean manufacturing systems, industrial engineers work to eliminate wastes of time, money, materials, energy, and other resources.

Examples of where industrial engineering might be used include designing an assembly workstation, strategizing for various operational logistics, consulting as an efficiency expert, developing a new financial algorithm or loan system for a bank, streamlining operation and emergency room location or usage in a hospital, planning complex distribution schemes for materials or products (referred to as Supply Chain Management), and shortening lines (or queues) at a bank, hospital, or a theme park.

Industrial engineers typically use computer simulation (especially discrete event simulation), along with extensive mathematical tools and modeling and computational methods for system analysis, evaluation, and optimization.

History

Efforts to apply science to the design of processes and of production systems were made by many people in the 18th and 19th centuries. They took some time to evolve and to be synthesized into disciplines that we would label with names such as industrial engineering, production engineering, or systems engineering. For example, precursors to industrial engineering included some aspects of military science; the quest to develop manufacturing using interchangeable parts; the development of the armory system of manufacturing; the work of Henri Fayol and colleagues (which grew into a larger movement called Fayolism); and the work of Frederick Winslow Taylor and colleagues (which grew into a larger movement called scientific management). In the late 19th century, such efforts began to inform consultancy and higher education. The idea of consulting with experts about process engineering naturally evolved into the idea of teaching the concepts as curriculum.

Industrial engineering courses were taught by multiple universities in Europe at the end of the 19th century, including in Germany, France, the United Kingdom, and Spain. In the United States, the first department of industrial and manufacturing engineering was established in 1909 at the Pennsylvania State University.

The first doctoral degree in industrial engineering was awarded in the 1930s by Cornell University.

University programs

Many universities have BS, MS, M.Tech and PhD programs available. US News and World Report's article on "America's Best Colleges 2010" lists schools offering Undergraduate engineering specialities in Industrial or Manufacturing. The Georgia Institute of Technology has been ranked as having the best Industrial Engineering program in the United States consecutively for the last twenty years according to this survey.

Postgraduate curriculum

The usual postgraduate degree earned is the Master of Science in Industrial Engineering/Production Engineering/Industrial Engineering & Management/Industrial Engineering & Operations Research. The typical MS in IE/PE/IE&M/IE & OR/Management Sciences curriculum includes:

- Operations research & Optimization techniques
- Engineering economics
- Supply chain management & Logistics
- Systems Simulation & Stochastic Processes
- System Dynamics & Policy Planning
- System Analysis & Techniques
- Manufacturing systems/Manufacturing engineering
- Human factors engineering & Ergonomics
- Production planning and control
- Management Sciences
- Computer aided manufacturing
- Facilities design & Work space design
- Quality Engineering
- Reliability Engineering & Life Testing
- Statistical process control or Quality control
- Time and motion study
- Operations management
- Corporate planning
- Productivity improvement
- Materials management

Undergraduate curriculum

In the United States, the usual undergraduate degree earned is the Bachelor of Science or B.S. in Industrial Engineering (BSIE). Like most undergraduate engineering programs, the typical curriculum includes a broad math and science foundation spanning chemistry, physics, engineering design, calculus, differential equations, statistics, materials science, engineering mechanics, computer science, circuits and electronics, and often additional specialized courses in areas such as management, systems theory, ergonomics/safety, stochastics, advanced mathematics and computation, and economics. Some Universities require International credits to complete the BS degree.

Salaries and workforce statistics

The total number of engineers employed in the U.S. in 2006 was roughly 1.5 million. Of these, 201,000 were industrial engineers (13.3%), the third most popular engineering specialty. The average **starting** salaries being $55,067 with a bachelor's degree, $64,759 with a master's degree, and $77,364 with a doctorate degree. This places industrial engineering at 7th of 15 among engineering bachelors degrees, 3rd of 10 among masters degrees, and 2nd of 7 among doctorate degrees in average annual salary. The median annual income of industrial engineers in the U.S. workforce is $68,620.

Often, within a few years at a company, industrial engineers will become strong candidates for technical supervisory or engineering management positions because their work is more related to management than most other engineering disciplines.

Chapter- 8

Operations Research

Operations research (also referred to as **decision science**, or **management science**) is an interdisciplinary mathematical science that focuses on the effective *use* of technology by organizations. In contrast, many other science & engineering disciplines focus on technology giving secondary considerations to its use.

Employing techniques from other mathematical sciences --- such as mathematical modeling, statistical analysis, and mathematical optimization --- operations research arrives at optimal or near-optimal solutions to complex decision-making problems. Because of its emphasis on human-technology interaction and because of its focus on practical applications, operations research has overlap with other disciplines, notably industrial engineering and management science, and draws on psychology and organization science. Operations Research is often concerned with determining the maximum (of profit, performance, or yield) or minimum (of loss, risk, or cost) of some real-world objective. Originating in military efforts before World War II, its techniques have grown to concern problems in a variety of industries.

Overview

Operational research encompasses a wide range of problem-solving techniques and methods applied in the pursuit of improved decision-making and efficiency. Some of the tools used by operational researchers are statistics, optimization, probability theory, queuing theory, game theory, graph theory, decision analysis, mathematical modeling and simulation. Because of the computational nature of these fields, OR also has strong ties to computer science and analytics. Operational researchers faced with a new problem must determine which of these techniques are most appropriate given the nature of the system, the goals for improvement, and constraints on time and computing power.

Work in operational research and management science may be characterized as one of three categories:

- Fundamental or foundational work takes place in three mathematical disciplines: probability, optimization, and dynamical systems theory.
- Modeling work is concerned with the construction of models, analyzing them mathematically, implementing them on computers, solving them using software

tools, and assessing their effectiveness with data. This level is mainly instrumental, and driven mainly by statistics and econometrics.
- Application work in operational research, like other engineering and economics' disciplines, attempts to use models to make a practical impact on real-world problems.

The major subdisciplines in modern operational research, as identified by the journal *Operations Research*, are:

- Computing and information technologies
- Decision analysis
- Environment, energy, and natural resources
- Financial engineering
- Manufacturing, service sciences, and supply chain management
- Policy modeling and public sector work
- Revenue management
- Simulation
- Stochastic models
- Transportation

History

As a formal discipline, operational research originated in the efforts of military planners during World War II. In the decades after the war, the techniques began to be applied more widely to problems in business, industry and society. Since that time, operational research has expanded into a field widely used in industries ranging from petrochemicals to airlines, finance, logistics, and government, moving to a focus on the development of mathematical models that can be used to analyze and optimize complex systems, and has become an area of active academic and industrial research.

Historical origins

In the World War II era, operational research was defined as "a scientific method of providing executive departments with a quantitative basis for decisions regarding the operations under their control." Other names for it included operational analysis (UK Ministry of Defence from 1962) and quantitative management.

Prior to the formal start of the field, early work in operational research was carried out by individuals such as Charles Babbage. His research into the cost of transportation and sorting of mail led to England's universal "Penny Post" in 1840, and studies into the dynamical behaviour of railway vehicles in defence of the GWR's broad gauge. Percy Bridgman brought operational research to bear on problems in physics in the 1920s and would later attempt to extend these to the social sciences. The modern field of operational research arose during World War II.

Modern operational research originated at the Bawdsey Research Station in the UK in 1937 and was the result of an initiative of the station's superintendent, A. P. Rowe. Rowe conceived the idea as a means to analyse and improve the working of the UK's early warning radar system, Chain Home (CH). Initially, he analyzed the operating of the radar equipment and its communication networks, expanding later to include the operating personnel's behaviour. This revealed unappreciated limitations of the CH network and allowed remedial action to be taken.

Scientists in the United Kingdom including Patrick Blackett later Lord Blackett OM PRS, Cecil Gordon, C. H. Waddington, Owen Wansbrough-Jones, Frank Yates, Jacob Bronowski and Freeman Dyson, and in the United States with George Dantzig looked for ways to make better decisions in such areas as logistics and training schedules. After the war it began to be applied to similar problems in industry.

Second World War

Patrick Blackett

During the Second World War close to 1,000 men and women in Britain were engaged in operational research. About 200 operational research scientists worked for the British Army.

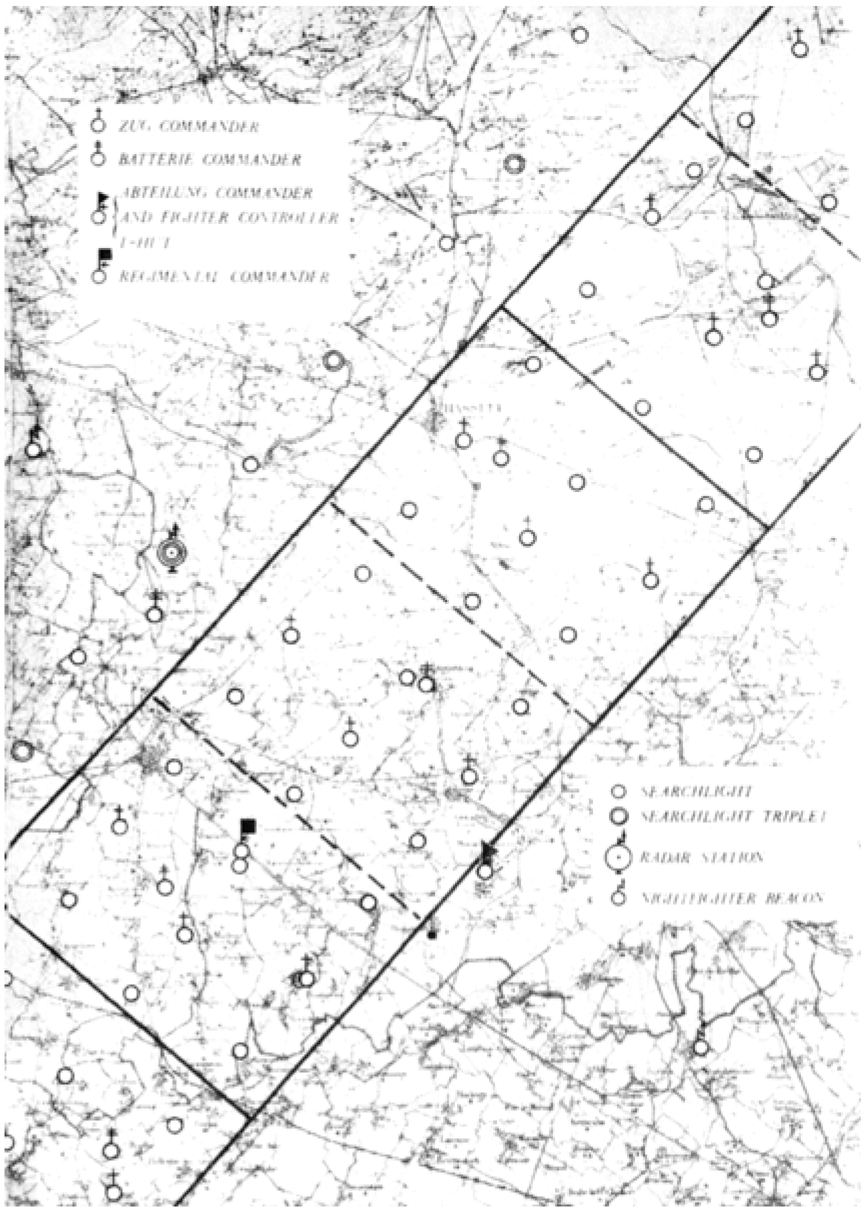
Patrick Blackett worked for several different organizations during the war. Early in the war while working for the Royal Aircraft Establishment (RAE) he set up a team known as the "Circus" which helped to reduce the number of anti-aircraft artillery rounds needed to shoot down an enemy aircraft from an average of over 20,000 at the start of the Battle of Britain to 4,000 in 1941.

In 1941 Blackett moved from the RAE to the Navy, first to the Royal Navy's Coastal Command, in 1941 and then early in 1942 to the Admiralty. Blackett's team at Coastal Command's Operational Research Section (CC-ORS) included two future Nobel prize winners and many other people who went on to be preeminent in their fields. They undertook a number of crucial analyses that aided the war effort. Britain introduced the convoy system to reduce shipping losses, but while the principle of using warships to accompany merchant ships was generally accepted, it was unclear whether it was better for convoys to be small or large. Convoys travel at the speed of the slowest member, so small convoys can travel faster. It was also argued that small convoys would be harder for German U-boats to detect. On the other hand, large convoys could deploy more warships against an attacker. Blackett's staff showed that the losses suffered by convoys depended largely on the number of escort vessels present, rather than on the overall size of the convoy. Their conclusion, therefore, was that a few large convoys are more defensible than many small ones.

While performing an analysis of the methods used by RAF Coastal Command to hunt and destroy submarines, one of the analysts asked what colour the aircraft were. As most of them were from Bomber Command they were painted black for nighttime operations. At the suggestion of CC-ORS a test was run to see if that was the best colour to camouflage the aircraft for daytime operations in the grey North Atlantic skies. Tests showed that aircraft painted white were on average not spotted until they were 20% closer than those painted black. This change indicated that 30% more submarines would be attacked and sunk for the same number of sightings.

Other work by the CC-ORS indicated that on average if the trigger depth of aerial delivered depth charges (DCs) was changed from 100 feet to 25 feet, the kill ratios would go up. The reason was that if a U-boat saw an aircraft only shortly before it arrived over the target then at 100 feet the charges would do no damage (because the U-boat wouldn't have time to descend as far as 100 feet), and if it saw the aircraft a long way from the target it had time to alter course under water so the chances of it being within the 20 feet kill zone of the charges was small. It was more efficient to attack those submarines close to the surface when these targets' locations were better known than to attempt their destruction at greater depths when their positions could only be guessed. Before the change of settings from 100 feet to 25 feet, 1% of submerged U-boats were sunk and 14% damaged. After the change, 7% were sunk and 11% damaged. (If submarines were caught on the surface, even if attacked shortly after submerging, the numbers rose to 11% sunk and 15% damaged). Blackett observed "there can be few cases where such a great operational gain had been obtained by such a small and simple change of tactics".

Bomber Command's Operational Research Section (BC-ORS), analysed a report of a survey carried out by RAF Bomber Command. For the survey, Bomber Command inspected all bombers returning from bombing raids over Germany over a particular period. All damage inflicted by German air defences was noted and the recommendation was given that armour be added in the most heavily damaged areas. Their suggestion to remove some of the crew so that an aircraft loss would result in fewer personnel loss was rejected by RAF command. Blackett's team instead made the surprising and counter-intuitive recommendation that the armour be placed in the areas which were completely untouched by damage in the bombers which returned. They reasoned that the survey was biased, since it only included aircraft that returned to Britain. The untouched areas of returning aircraft were probably vital areas, which, if hit, would result in the loss of the aircraft.



Map of *Kammhuber Line*

When Germany organised its air defences into the Kammhuber Line, it was realised that if the RAF bombers were to fly in a bomber stream they could overwhelm the night fighters who flew in individual cells directed to their targets by ground controllers. It was then a matter of calculating the statistical loss from collisions against the statistical loss from night fighters to calculate how close the bombers should fly to minimise RAF losses.

The "exchange rate" ratio of output to input was a characteristic feature of operational research. By comparing the number of flying hours put in by Allied aircraft to the number of U-boat sightings in a given area, it was possible to redistribute aircraft to more productive patrol areas. Comparison of exchange rates established "effectiveness ratios" useful in planning. The ratio of 60 mines laid per ship sunk was common to several campaigns: German mines in British ports, British mines on German routes, and United States mines in Japanese routes.

Operational research doubled the on-target bomb rate of B-29s bombing Japan from the Marianas Islands by increasing the training ratio from 4 to 10 percent of flying hours; revealed that wolf-packs of three United States submarines were the most effective number to enable all members of the pack to engage targets discovered on their individual patrol stations; revealed that glossy enamel paint was more effective camouflage for night fighters than traditional dull camouflage paint finish, and the smooth paint finish increased airspeed by reducing skin friction.

On land, the operational research sections of the Army Operational Research Group (AORG) of the Ministry of Supply (MoS) were landed in Normandy in 1944, and they followed British forces in the advance across Europe. They analysed, among other topics, the effectiveness of artillery, aerial bombing, and anti-tank shooting.

After World War II

With expanded techniques and growing awareness of the field at the close of the war, operational research was no longer limited to only operational, but was extended to encompass equipment procurement, training, logistics and infrastructure.

Academic Denis Bouyssou describes the historical development of operational research from the 1940s to the 1970s as follows. "The historical development of Operational Research (OR) is traditionally seen as the succession of several phases: the 'heroic times' of the Second World War, the 'Golden Age' between the fifties and the sixties during which major theoretical achievements were accompanied by a widespread diffusion of OR techniques in private and public organisations, a 'crisis' followed by a 'decline' starting with the late sixties, a phase during which OR groups in firms progressively disappeared while academia became less and less concerned with the applicability of the techniques developed".

Individuals such as Stafford Beer and George Dantzig pioneered early academic efforts in operational research.

Problems addressed with operational research

- critical path analysis or project planning: identifying those processes in a complex project which affect the overall duration of the project
- floorplanning: designing the layout of equipment in a factory or components on a computer chip to reduce manufacturing time (therefore reducing cost)
- network optimization: for instance, setup of telecommunications networks to maintain quality of service during outages
- allocation problems
- Bayesian search theory : looking for a target

- optimal search
- routing, such as determining the routes of buses so that as few buses are needed as possible
- supply chain management: managing the flow of raw materials and products based on uncertain demand for the finished products
- efficient messaging and customer response tactics
- automation: automating or integrating robotic systems in human-driven operations processes
- globalization: globalizing operations processes in order to take advantage of cheaper materials, labor, land or other productivity inputs
- transportation: managing freight transportation and delivery systems (Examples: LTL Shipping, intermodal freight transport)
- scheduling:
    - personnel staffing
    - manufacturing steps
    - project tasks
    - network data traffic: these are known as queueing models or queueing systems.
    - sports events and their television coverage
- blending of raw materials in oil refineries
- determining optimal prices, in many retail and B2B settings, within the disciplines of pricing science

Operational research is also used extensively in government where evidence-based policy is used.

Management science

In 1967 Stafford Beer characterized the field of management science as "the business use of operations research". However, in modern times the term management science may also be used to refer to the separate fields of organizational studies or corporate strategy. Like operational research itself, management science (MS), is an interdisciplinary branch of applied mathematics devoted to optimal decision planning, with strong links with economics, business, engineering, and other sciences. It uses various scientific research-based principles, strategies, and analytical methods including mathematical modeling, statistics and numerical algorithms to improve an organization's ability to enact rational and meaningful management decisions by arriving at optimal or near optimal solutions to complex decision problems. In short, management sciences help businesses to achieve their goals using the scientific methods of operational research.

The management scientist's mandate is to use rational, systematic, science-based techniques to inform and improve decisions of all kinds. Of course, the techniques of management science are not restricted to business applications but may be applied to military, medical, public administration, charitable groups, political groups or community groups.

Management science is concerned with developing and applying models and concepts that may prove useful in helping to illuminate management issues and solve managerial problems, as well as designing and developing new and better models of organizational excellence.

The application of these models within the corporate sector became known as Management science.

Techniques

Some of the fields that have considerable overlap with Management Science include:

- Data mining
- Decision analysis
- Engineering
- Forecasting
- Game theory
- Industrial engineering
- Logistics
- Mathematical modeling
- Optimization
- Probability and statistics
- Project management
- Simulation
- Social network/Transportation forecasting models
- Supply chain management
- Financial engineering

Applications of management science

Applications of management science are abundant in industry as airlines, manufacturing companies, service organizations, military branches, and in government. The range of problems and issues to which management science has contributed insights and solutions is vast. It includes:.

- scheduling airlines, including both planes and crew,
- deciding the appropriate place to site new facilities such as a warehouse, factory or fire station,
- managing the flow of water from reservoirs,
- identifying possible future development paths for parts of the telecommunications industry,
- establishing the information needs and appropriate systems to supply them within the health service, and
- identifying and understanding the strategies adopted by companies for their information systems

Management science is also concerned with so-called ”soft-operational analysis”, which concerns methods for strategic planning, strategic decision support, and Problem Structuring Methods (PSM). In dealing with these sorts of challenges mathematical modeling and simulation are not appropriate or will not suffice. Therefore, during the past 30 years, a number of non-quantified modelling methods have been developed. These include:

- stakeholder based approaches including metagame analysis and drama theory
- morphological analysis and various forms of influence diagrams.
- approaches using cognitive mapping
- the Strategic Choice Approach
- robustness analysis

Chapter- 9

Performance Engineering

Performance engineering within systems engineering, encompasses the set of roles, skills, activities, practices, tools, and deliverables applied at every phase of the Systems Development Life Cycle which ensures that a solution will be designed, implemented, and operationally supported to meet the non-functional performance requirements defined for the solution.

It may be alternatively referred to as *software performance engineering* within software engineering; however since performance engineering encompasses more than just the software, the term performance engineering is preferable. Adherence to the non-functional requirements is validated by monitoring the production systems. This is part of IT service management.

Performance engineering has become a separate discipline at a number of large corporations, and may be affiliated with the enterprise architecture group. It is pervasive, involving people from multiple organizational units; but predominantly within the information technology organization.

Performance Engineering Objectives

- Increase business revenue by ensuring the system can process transactions within the requisite timeframe
- Eliminate system failure requiring scrapping and writing off the system development effort due to performance objective failure
- Eliminate late system deployment due to performance issues
- Eliminate avoidable system rework due to performance issues
- Eliminate avoidable system tuning efforts
- Avoid additional and unnecessary hardware acquisition costs
- Reduce increased software maintenance costs due to performance problems in production
- Reduce increased software maintenance costs due to software impacted by ad hoc performance fixes
- Reduce additional operational overhead for handling system issues due to performance problems

Performance Engineering Approach

Because this discipline is applied within multiple methodologies, the following activities will occur within differently specified phases. However if the phases of the rational unified process (RUP) are used as a framework, then the activities will occur as follows:

Inception

During this first conceptual phase of a program or project, critical business processes are identified. Typically they are classified as critical based upon revenue value, cost savings, or other assigned business value. This classification is done by the business unit, not the IT organization.

High level risks that may impact system performance are identified and described at this time. An example might be known performance risks for a particular vendor system.

Finally performance activities, roles, and deliverables are identified for the Elaboration phase. Activities and resource loading are incorporated into the Elaboration phase project plans.

Elaboration

During this defining phase, the critical business processes are decomposed to critical use cases. Such use cases will be decomposed further, as needed, to single page (screen) transitions. These are the use cases that will be subjected to script driven performance testing.

The type of requirements that relate to Performance Engineering are the non-functional requirements, or NFR. While a functional requirement relates to **what** business operations are to be performed, a performance related non-functional requirement will relate to **how fast** that business operation performs under defined circumstances.

The concept of "defined circumstances" is vital. This will be illustrated by example:

- Invalid – the system should respond to user input within 10 seconds.
- Valid – for use case ABC the system will respond to a valid user entry within 5 seconds for a median load of 250 active users and 2000 logged in users 95% of the time; or within 10 seconds for a peak load of 500 active users and 4000 logged in users 90% of the time.

Note the critical differences between the two specifications. The first example provides no conditions. The second clearly identifies the conditions under which the system is to perform. The second example may have a service level agreement, the first should not. The capacity planners and architects can actually design and build a system to meet the criteria for the valid nonfunctional requirement – but not for the invalid one. Testers may build a reliable performance test for the second example, but not for the invalid example.

Each critical use case must have an associated NFR. If, for a given use case, no existing NFR is applicable, a new NFR specific to that use case must be created.

Non functional requirements are not limited to use cases. The overall **system volumetrics** must be specified. These will describe the overall system load over a specified time period, defining how many of each type of business transaction will be executed per unit of time. Commonly volumetrics describe a typical business day, and then are broken down for each hour. This will describe how system load will vary over the course of the day. For example: 1200 of transaction A, 300 of transaction B, 3300 of transaction C, etc. for a given business day; then in hour 1 so many executions of A, B, C etc., in hour 2 so many transaction executions, and so on. The information is often formatted in a tabular form for clarity. If different user classes are executing the transactions, this information will also be incorporated in the NFR documentation. Finally, the transactions may be classified as to general type, normally being user interaction, report generation, and batch processing.

The system volumetrics documented in the NFR documentation will be used as inputs for both load testing and stress testing of the system during the performance test.

At this point it is suggested that performance modeling be performed using the use case information as input. This may be done using a performance lab, and using prototypes and mockups of the "to be" system; or a vendor provided modeling tool may be used; or even merely a spreadsheet workbook, where each use case is modeled in a single sheet, and a summary sheet is used to provide high level information for all of the use cases.

It is recommended that Unified Modeling Language sequence diagrams be generated at the physical tier level for each use case. The physical tiers are represented by the vertical object columns, and the message communication between the tiers by the horizontal arrows. Timing information should be associated with each horizontal arrow; this should correlate with the performance model.

Some performance engineering activities related to performance testing should be executed in this phase. They include validating a performance test strategy, developing a performance test plan, determining the sizing of test data sets, developing a performance test data plan, and identifying performance test scenarios.

For any system of significant impact, a monitoring plan and a monitoring design are developed in this phase. Performance engineering applies a subset of activities related to performance monitoring, both for the performance test environment as well as for the production environment.

The risk document generated in the previous phase is revisited here. A risk mitigation plan is determined for each identified performance risk; and time, cost, and responsibility is determined and documented.

Finally performance activities, roles, and deliverables are identified for the Construction phase. Activities and resource loading are incorporated into the Construction phase project plans. These will be elaborated for each iteration.

Construction

Early in this phase a number of performance tool related activities are required. These include:

- Identify key development team members as subject matter experts for the selected tools
- Specify a profiling tool for the development/component unit test environment
- Specify an automated unit (component) performance test tool for the development/component unit test environment; this is used when no GUI yet exists to drive the components under development
- Specify an automated tool for driving server-side unit (components) for the development/component unit test environment
- Specify an automated multi-user capable script-driven end-to-end tool for the development/component unit test environment; this is used to execute screen-driven use cases
- Identify a database test data load tool for the development/component unit test environment; this is required to ensure that the database optimizer chooses correct execution paths and to enable reinitializing and reloading the database as needed
- Deploy the performance tools for the development team
- Presentations and training must be given to development team members on the selected tools

A member of the performance engineering practice and the development technical team leads should work together to identify performance-oriented best practices for the development team. Ideally the development organization should already have a body of best practices, but often these do not include or emphasize those best practices that impact system performance.

The concept of application instrumentation should be introduced here with the participation of the IT Monitoring organization. Several vendor monitoring systems have performance capabilities, these normally operate at the operating system, network, and server levels; e.g. CPU utilization, memory utilization, disk I/O, and for J2EE servers the JVM performance including garbage collection.

But this type of monitoring does not permit the tracking of use case level performance. To reach this level of monitoring capability may require that the application itself be instrumented. Alternatively, a monitoring toolset that works at the switch level may be used. (Examples might be TeaLeaf's Cx technology, Quest Software's Foglight, Hewlett-Packard's RUM, NetQoS's SuperAgent, or Compuware's agentless ClientVantage.) The monitoring group should have specified the requirements in a previous phase, and should work with the development team to ensure that use case level monitoring is built in.

The group responsible for infrastructural performance tuning should have an established "base model" checklist to tune the operating systems, network, servers (application, web, database, load balancer, etc.), and any message queueing software. Then as the performance test team starts to gather data, they should commence tuning the environment more specifically for the system to be deployed. This requires the active support of subject matter experts, for example, database tuning normally requires a DBA who has special skills in that area.

The performance test team normally does not execute performance tests in the development environment, but rather in a specialized pre-deployment environment that is configured to be as close as possible to the planned production environment. This team will execute performance testing against test cases, validating that the critical use cases conform to the specified non-functional requirements. The team will execute load testing against a normally expected (median) load as well as a peak load. They will often run stress tests that will identify the system bottlenecks. The data gathered, and the analyses, will be fed back to the group that does performance tuning. Where necessary, the system will be tuned to bring nonconforming tests into conformance with the non-functional requirements.

If performance engineering has been properly applied at each iteration and phase of the project to this point, hopefully this will be sufficient to enable the system to receive performance certification. However, if for some reason (perhaps proper performance engineering working practices were not applied) there are tests that cannot be tuned into compliance, then it will be necessary to return portions of the system to development for refactoring. In some cases the problem can be resolved with additional hardware, but adding more hardware leads quickly to diminishing returns.

For example: suppose we can improve 70% of a module by parallelizing it, and run on 4 CPUs instead of 1 CPU. If α is the fraction of a calculation that is sequential, and $(1-\alpha)$ is the fraction that can be parallelized, then the maximum speedup that can be achieved by using P processors is given according to Amdahl's Law: $\frac{1}{\alpha + \frac{1-\alpha}{P}}$

In this example we would get: 1/(.3+(1-.3)/4)=2.105. So for quadrupling the processing power we only doubled the performance (from 1 to 2.105). And we are now well on the way to diminishing returns. If we go on to double the computing power again from 4 to 8 processors we get 1/(.3+(1-.3)/8)=2.581. So now by doubling the processing power again we only got a performance improvement of about one fifth (from 2.105 to 2.581).

Transition

During this final phase the system is deployed to the production environment. A number of preparatory steps are required. These include:

- Configuring the operating systems, network, servers (application, web, database, load balancer, etc.), and any message queueing software according to the base checklists and the optimizations identified in the performance test environment
- Ensuring all performance monitoring software is deployed and configured
- Running Statistics on the database after the production data load is completed

Once the new system is deployed, ongoing operations pick up performance activities, including:

- Validating that weekly and monthly performance reports indicate that critical use cases perform within the specified non functional requirement criteria
- Where use cases are falling outside of NFR criteria, submit defects
- Identify projected trends from monthly and quarterly reports, and on a quarterly basis, execute capacity planning management activities

Service Management

In the operational domain (post production deployment) performance engineering focuses primarily within three areas: service level management, capacity management, and problem management.

Service Level Management

In the service level management area, performance engineering is concerned with service level agreements and the associated systems monitoring that serves to validate service level compliance, detect problems, and identify trends. For example, when real user monitoring is deployed it is possible to ensure that user transactions are being executed in conformance with specified non-functional requirements. Transaction response time is logged in a database such that queries and reports can be run against the data. This permits trend analysis that can be useful for capacity management. When user transactions fall out of band, the events should generate alerts so that attention may be applied to the situation.

Capacity Management

For capacity management, performance engineering focuses on ensuring that the systems will remain within performance compliance. This means executing trend analysis on historical monitoring generated data, such that the future time of non compliance is predictable. For example, if a system is showing a trend of slowing transaction processing (which might be due to growing data set sizes, or increasing numbers of concurrent users, or other factors) then at some point the system will no longer meet the criteria specified within the service level agreements. Capacity management is charged with ensuring that additional capacity is added in advance of that point (additional CPUs, more memory, new database indexing, et cetera) so that the trend lines are reset and the system will remain within the specified performance range.

Problem Management

Within the problem management domain, the performance engineering practices are focused on resolving the root cause of performance related problems. These typically involve system tuning, changing operating system or device parameters, or even refactoring the application software to resolve poor performance due to poor design or bad coding practices.

Monitoring

To ensure that there is proper feedback validating that the system meets the NFR specified performance metrics, any major system needs a monitoring subsystem. The planning, design, installation, configuration, and control of the monitoring subsystem is specified by an appropriately defined Monitoring Process. The benefits are as follows:

1. It is possible to establish service level agreements at the use case level.
2. It is possible to turn on and turn off monitoring at periodic points or to support problem resolution.
3. It enables the generation of regular reports.
4. It enables the ability to track trends over time – such as the impact of increasing user loads and growing data sets on use case level performance.

The trend analysis component of this cannot be undervalued. This functionality, properly implemented, will enable predicting when a given application undergoing gradually increasing user loads and growing data sets will exceed the specified non functional performance requirements for a given use case. This permits proper management budgeting, acquisition of, and deployment of the required resources to keep the system running within the parameters of the non functional performance requirements.

Chapter- 10

Software Engineering

The Airbus A380 uses a substantial amount of software to create a "paperless" cockpit

Software engineering (**SE**) is a profession dedicated to designing, implementing, and modifying software so that it is of higher quality, more affordable, maintainable, and faster to build. It is a "systematic approach to the analysis, design, assessment, implementation, test, maintenance and reengineering of software, that is, the application of engineering to software." The term *software engineering* first appeared in the 1968 NATO Software Engineering Conference, and was meant to provoke thought regarding the perceived "software crisis" at the time. The IEEE Computer Society's *Software Engineering Body of Knowledge* defines "software engineering" as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software. It is the application of Engineering to software because it integrates significant mathematics, computer science and practices whose origins are in Engineering.

Software development, a much used and more generic term, does not necessarily subsume the engineering paradigm. Although it is questionable what impact it has had on actual software development over the last more than 40 years, the field's future looks bright according to Money Magazine and Salary.com, which rated "software engineer" as the best job in the United States in 2006.

History

When the first modern digital computers appeared in the early 1940s, the instructions to make them operate were wired into the machine. Practitioners quickly realized that this design was not flexible and came up with the "stored program architecture" or von Neumann architecture. Thus the first division between "hardware" and "software" began with abstraction being used to deal with the complexity of computing.

Programming languages started to appear in the 1950s and this was also another major step in abstraction. Major languages such as Fortran, ALGOL, and COBOL were released in the late 1950s to deal with scientific, algorithmic, and business problems respectively. E.W. Dijkstra wrote his seminal paper, "Go To Statement Considered Harmful", in 1968 and David Parnas introduced the key concept of modularity and information hiding in 1972 to help programmers deal with the ever increasing complexity of software systems. A software system for managing the hardware called an operating system was also introduced, most notably by Unix in 1969. In 1967, the Simula language introduced the object-oriented programming paradigm.

These advances in software were met with more advances in computer hardware. In the mid 1970s, the microcomputer was introduced, making it economical for hobbyists to obtain a computer and write software for it. This in turn led to the now famous Personal Computer (PC) and Microsoft Windows. The Software Development Life Cycle or SDLC was also starting to appear as a consensus for centralized construction of software in the mid 1980s. The late 1970s and early 1980s saw the introduction of several new Simula-inspired object-oriented programming languages, including Smalltalk, Objective-C, and C++.

Open-source software started to appear in the early 90s in the form of Linux and other software introducing the "bazaar" or decentralized style of constructing software. Then the World Wide Web and the popularization of the Internet hit in the mid 90s, changing the engineering of software once again. Distributed systems gained sway as a way to design systems, and the Java programming language was introduced with its own virtual machine as another step in abstraction. Programmers collaborated and wrote the Agile Manifesto, which favored more lightweight processes to create cheaper and more timely software.

The current definition of *software engineering* is still being debated by practitioners today as they struggle to come up with ways to produce software that is "cheaper, better, faster". Cost reduction has been a primary focus of the IT industry since the 1990s. Total cost of ownership represents the costs of more than just acquisition. It includes things like productivity impediments, upkeep efforts, and resources needed to support infrastructure.

Profession

Legal requirements for the licensing or certification of professional software engineers vary around the world. In the UK, the British Computer Society licenses software engineers and members of the society can also become Chartered Engineers (CEng), while in some areas of Canada, such as Alberta, Ontario, and Quebec, software engineers can hold the Professional Engineer (P.Eng)designation and/or the Information Systems Professional (I.S.P.) designation; however, there is no legal requirement to have these qualifications. In Israel a person with an appropriate engineering degree has the right to be listed in Israel's Registry of Engineers and Architects, and Israeli engineering law says that a person calling themselves an engineer without the proper license / registration could be sentenced to up to 6 months in jail.

The IEEE Computer Society and the ACM, the two main professional organizations of software engineering, publish guides to the profession of software engineering. The IEEE's *Guide to the Software Engineering Body of Knowledge - 2004 Version*, or SWEBOK, defines the field and describes the knowledge the IEEE expects a practicing software engineer to have. The IEEE also promulgates a "Software Engineering Code of Ethics".

Employment

In 2004, the U. S. Bureau of Labor Statistics counted 760,840 software engineers holding jobs in the U.S.; in the same time period there were some 1.4 million practitioners employed in the U.S. in all other engineering disciplines combined. Due to its relative newness as a field of study, formal education in software engineering is often taught as part of a computer science curriculum, and many software engineers hold computer science degrees.

Many software engineers work as employees or contractors. Software engineers work with businesses, government agencies (civilian or military), and non-profit organizations. Some software engineers work for themselves as freelancers. Some organizations have specialists to perform each of the tasks in the software development process. Other organizations require software engineers to do many or all of them. In large projects, people may specialize in only one role. In small projects, people may fill several or all roles at the same time. Specializations include: in industry (analysts, architects, developers, testers, technical support, middleware analysts, managers) and in academia (educators, researchers).

Most software engineers and programmers work 40 hours a week, but about 15 percent of software engineers and 11 percent of programmers worked more than 50 hours a week in 2008. Injuries in these occupations are rare. However, like other workers who spend long periods in front of a computer terminal typing at a keyboard, engineers and programmers are susceptible to eyestrain, back discomfort, and hand and wrist problems such as carpal tunnel syndrome.

Certification

The Software Engineering Institute offers certifications on specific topics like Security, Process improvement and Software architecture. Apple, IBM, Microsoft and other companies also sponsor their own certification examinations. Many IT certification programs are oriented toward specific technologies, and managed by the vendors of these technologies. These certification programs are tailored to the institutions that would employ people who use these technologies.

Broader certification of general software engineering skills is available through various professional societies. As of 2006, the IEEE had certified over 575 software professionals as a Certified Software Development Professional (CSDP). In 2008 they added an entry-level certification known as the Certified Software Development Associate (CSDA). In the U.K. the British Computer Society has developed a legally recognized professional certification called *Chartered IT Professional (CITP)*, available to fully qualified Members (*MBCS*). In Canada the Canadian Information Processing Society has developed a legally recognized professional certification called *Information Systems Professional (ISP)*. The ACM had a professional certification program in the early 1980s, which was discontinued due to lack of interest. The ACM examined the possibility of professional certification of software engineers in the late 1990s, but eventually decided that such certification was inappropriate for the professional industrial practice of software engineering.

Impact of globalization

The initial impact of outsourcing, and the relatively lower cost of international human resources in developing third world countries led to the dot com bubble burst of the 1990s. This had a negative impact on many aspects of the software engineering profession. For example, some students in the developed world avoid education related to software engineering because of the fear of offshore outsourcing (importing software products or services from other countries) and of being displaced by foreign visa workers. Although statistics do not currently show a threat to software engineering itself; a related career, computer programming does appear to have been affected. Nevertheless, the ability to smartly leverage offshore and near-shore resources in an efficient fashion has improved the overall operational capability of many organizations. When Europeans are leaving work, Asians are just arriving to work. When Asians are leaving work, Europeans are arriving to work. This provides a continuous ability to have human oversight on business-critical processes 24 hours per day, without paying overtime compensation or disrupting key human resource sleep patterns.

Education

A knowledge of programming is a pre-requisite to becoming a software engineer. In 2004 the IEEE Computer Society produced the SWEBOK, which has been published as ISO/IEC Technical Report 19759:2004, describing the body of knowledge that they believe should be mastered by a graduate software engineer with four years of experience. Many software engineers enter the profession by obtaining a university degree or training at a vocational school. One standard international curriculum for undergraduate software engineering degrees was defined by the CCSE, and updated in 2004. A number of universities have Software Engineering degree programs; as of 2010, there were 244 Campus programs, 70 Online programs, 230 Masters-level programs, 41 Doctorate-level programs, and 69 Certificate-level programs in the United States.

In addition to university education, many companies sponsor internships for students wishing to pursue careers in information technology. These internships can introduce the student to interesting real-world tasks that typical software engineers encounter every day. Similar experience can be gained through military service in software engineering.

Sub-disciplines

Software engineering can be divided into ten subdisciplines. They are:

- Software requirements: The elicitation, analysis, specification, and validation of requirements for software.
- Software design: The design of software is usually done with Computer-Aided Software Engineering (CASE) tools and use standards for the format, such as the Unified Modeling Language (UML).
- Software development: The construction of software through the use of programming languages.
- Software testing
- Software maintenance: Software systems often have problems and need enhancements for a long time after they are first completed. This subfield deals with those problems.
- Software configuration management: Since software systems are very complex, their configuration (such as versioning and source control) have to be managed in a standardized and structured method.
- Software engineering management: The management of software systems borrows heavily from project management, but there are nuances encountered in software not seen in other management disciplines.
- Software development process: The process of building software is hotly debated among practitioners; some of the better-known processes are the Waterfall Model, the Spiral Model, Iterative and Incremental Development, and Agile Development.
- Software engineering tools
- Software quality

Related disciplines

Software engineering is a direct subfield of computer science and has some relations with management science. It is also considered a part of overall systems engineering.

Systems engineering

Systems engineers deal primarily with the overall system design, specifically dealing more with physical aspects which include hardware design. Those who choose to specialize in computer hardware engineering may have some training in software engineering.

Computer Software Engineers

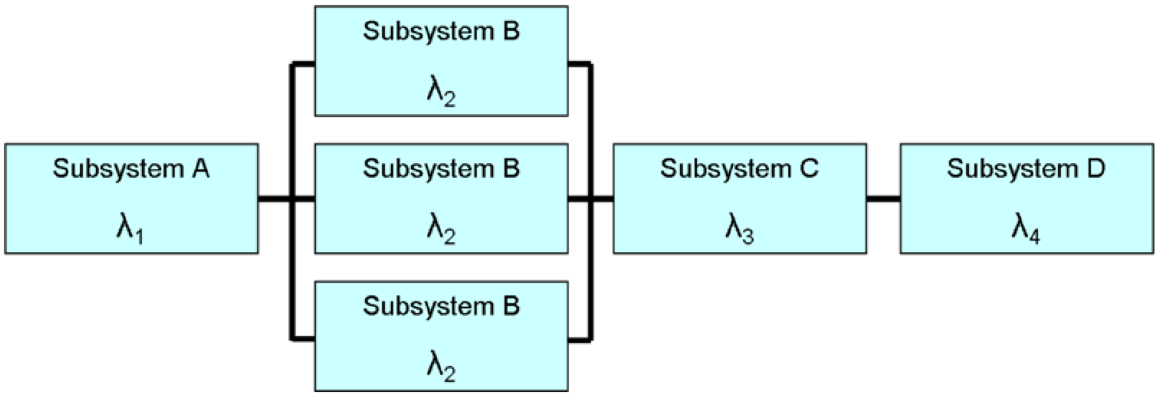
Computer Software Engineers are usually systems level (software engineering, information systems) computer science or software level computer engineering graduates. This term also includes general computer science graduates with a few years of practical on the job experience involving software engineering.

Chapter- 11

Reliability Engineering

Reliability engineering is an engineering field, that deals with the study of reliability: the ability of a system or component to perform its required functions under stated conditions for a specified period of time. It is often reported as a probability.

Overview



A Reliability Block Diagram

Reliability may be defined in several ways:

- The idea that something is fit for a purpose with respect to time;
- The capacity of a device or system to perform as designed;
- The resistance to failure of a device or system;
- The ability of a device or system to perform a required function under stated conditions for a specified period of time;
- The probability that a functional unit will perform its required function for a specified interval under stated conditions.
- The ability of something to "fail well" (fail without catastrophic consequences)

Reliability engineers rely heavily on statistics, probability theory, and reliability theory. Many engineering techniques are used in reliability engineering, such as reliability prediction, Weibull analysis, thermal management, reliability testing and accelerated life testing. Because of the large number of reliability techniques, their expense, and the varying degrees of reliability required for different situations, most projects develop a reliability program plan to specify the reliability tasks that will be performed for that specific system.

The function of reliability engineering is to develop the reliability requirements for the product, establish an adequate reliability program, and perform appropriate analyses and tasks to ensure the product will meet its requirements. These tasks are managed by a reliability engineer, who usually holds an accredited engineering degree and has additional reliability-specific education and training. Reliability engineering is closely associated with maintainability engineering and logistics engineering, e.g. Integrated Logistics Support (ILS). Many problems from other fields, such as security engineering, can also be approached using reliability engineering techniques.

Many types of engineering employ reliability engineers and use the tools and methodology of reliability engineering. For example:

- System engineers design complex systems having a specified reliability
- Mechanical engineers may have to design a machine or system with a specified reliability
- Automotive engineers have reliability requirements for the automobiles (and components) which they design
- Electronics engineers must design and test their products for reliability requirements.
- In software engineering and systems engineering the **reliability engineering** is the subdiscipline of ensuring that a system (or a device in general) will perform its intended function(s) when operated in a specified manner for a specified length of time. Reliability engineering is performed throughout the entire life cycle of a system, including development, test, production and operation.

Reliability theory

Reliability theory is the foundation of reliability engineering. For engineering purposes, reliability is defined as:

> **the probability that a device will perform its intended function during a specified period of time under stated conditions.**

Mathematically, this may be expressed as,

$$R(t) = Pr\{T > t\} = \int_t^{\infty} f(x)\,dx ,$$

where $f(x)$ is the failure probability density function and *t* is the length of the period of time (which is assumed to start from time zero).

Reliability engineering is concerned with four key elements of this definition:

- First, reliability is a probability. This means that failure is regarded as a random phenomenon: it is a recurring event, and we do not express any information on individual failures, the causes of failures, or relationships between failures, except that the likelihood for failures to occur varies over time according to the given probability function. Reliability engineering is concerned with meeting the specified probability of success, at a specified statistical confidence level.
- Second, reliability is predicated on "intended function:" Generally, this is taken to mean operation without failure. However, even if no individual part of the system fails, but the system as a whole does not do what was intended, then it is still charged against the system reliability. The system requirements specification is the criterion against which reliability is measured.
- Third, reliability applies to a specified period of time. In practical terms, this means that a system has a specified chance that it will operate without failure before time t. Reliability engineering ensures that components and materials will meet the requirements during the specified time. Units other than time may sometimes be used. The automotive industry might specify reliability in terms of miles, the military might specify reliability of a gun for a certain number of rounds fired. A piece of mechanical equipment may have a reliability rating value in terms of cycles of use.
- Fourth, reliability is restricted to operation under stated conditions. This constraint is necessary because it is impossible to design a system for unlimited conditions. A Mars Rover will have different specified conditions than the family car. The operating environment must be addressed during design and testing. Also, that same rover, may be required to operate in varying conditions requiring additional scrutiny.

Reliability program plan

Many tasks, methods, and tools can be used to achieve reliability. Every system requires a different level of reliability. A commercial airliner must operate under a wide range of conditions. The consequences of failure are grave, but there is a correspondingly higher budget. A pencil sharpener may be more reliable than an airliner, but has a much different set of operational conditions, insignificant consequences of failure, and a much lower budget.

A reliability program plan is used to document exactly what tasks, methods, tools, analyses, and tests are required for a particular system. For complex systems, the reliability program plan is a separate document. For simple systems, it may be combined with the systems engineering management plan or integrated Logistics Support management plan. The reliability program plan is essential for a successful reliability program and is developed early during system development. It specifies not only what the reliability engineer does, but also the tasks performed by others. The reliability program plan is approved by top program management.

Reliability requirements

For any system, one of the first tasks of reliability engineering is to adequately specify the reliability requirements. Reliability requirements address the system itself, test and assessment requirements, and associated tasks and documentation. Reliability requirements are included in the appropriate system/subsystem requirements specifications, test plans, and contract statements.

System reliability parameters

Requirements are specified using reliability parameters. The most common reliability parameter is the mean time between failures (MTBF), which can also be specified as the failure rate or the number of failures during a given period. These parameters are very useful for systems that are operated frequently, such as most vehicles, machinery, and electronic equipment. Reliability increases as the MTBF increases. The MTBF is usually specified in hours, but can also be used with other units of measurement such as miles or cycles.

In other cases, reliability is specified as the probability of mission success. For example, reliability of a scheduled aircraft flight can be specified as a dimensionless probability or a percentage. refer to system safety engineering.

A special case of mission success is the single-shot device or system. These are devices or systems that remain relatively dormant and only operate once. Examples include automobile airbags, thermal batteries and missiles. Single-shot reliability is specified as a probability of success, or is subsumed into a related parameter. Single-shot missile reliability may be incorporated into a requirement for the probability of hit.

For such systems, the probability of failure on demand (PFD) is the reliability measure. This PFD is derived from failure rate and mission time for non-repairable systems. For repairable systems, it is obtained from failure rate and mean-time-to-repair (MTTR) and test interval. This measure may not be unique for a given system as this measure depends on the kind of demand. In addition to system level requirements, reliability requirements may be specified for critical subsystems. In all cases, reliability parameters are specified with appropriate statistical confidence intervals.

Reliability modelling

Reliability modelling is the process of predicting or understanding the reliability of a component or system. Two separate fields of investigation are common: The physics of failure approach uses an understanding of the failure mechanisms involved, such as crack propagation or chemical corrosion; The parts stress modelling approach is an empirical method for prediction based on counting the number and type of components of the system, and the stress they undergo during operation.

For systems with a clearly defined failure time (which is sometimes not given for systems with a drifting parameter), the empirical distribution function of these failure times can be determined. This is done in general in an accelerated experiment with increased stress. These experiments can be divided into two main categories:

Early failure rate studies determine the distribution with a decreasing failure rate over the first part of the bathtub curve. Here in general only moderate stress is necessary. The stress is applied for a limited period of time in what is called a censored test. Therefore, only the part of the distribution with early failures can be determined.

In so-called zero defect experiments, only limited information about the failure distribution is acquired. Here the stress, stress time, or the sample size is so low that not a single failure occurs. Due to the insufficient sample size, only an upper limit of the early failure rate can be determined. At any rate, it looks good for the customer if there are no failures.

In a study of the intrinsic failure distribution, which is often a material property, higher stresses are necessary to get failure in a reasonable period of time. Several degrees of stress have to be applied to determine an acceleration model. The empirical failure distribution is often parametrised with a Weibull or a log-normal model.

It is a general praxis to model the early failure rate with an exponential distribution. This less complex model for the failure distribution has only one parameter: the constant failure rate. In such cases, the Chi-square distribution can be used to find the goodness of fit for the estimated failure rate. Compared to a model with a decreasing failure rate, this is quite pessimistic (important remark: this is not the case if less hours / load cycles are tested than service life in a wear-out type of test, in this case the opposite is true and assuming a more constant failure rate than it is in reality can be dangerous). Sensitivity analysis should be conducted in this case.

Reliability test requirements

Because reliability is a probability, even highly reliable systems have some chance of failure. However, testing reliability requirements is problematic for several reasons. A single test is insufficient to generate enough statistical data. Multiple tests or long-duration tests are usually very expensive. Some tests are simply impractical. Reliability engineering is used to design a realistic and affordable test program that provides enough evidence that the system meets its requirement. Statistical confidence levels are used to address some of these concerns. A certain parameter is expressed along with a corresponding confidence level: for example, an MTBF of 1000 hours at 90% confidence level. From this specification, the reliability engineer can design a test with explicit criteria for the number of hours and number of failures until the requirement is met or failed.

The combination of reliability parameter value and confidence level greatly affects the development cost and the risk to both the customer and producer. Care is needed to select the best combination of requirements. Reliability testing may be performed at various levels, such as component, subsystem, and system. Also, many factors must be addressed during testing, such as extreme temperature and humidity, shock, vibration, and heat. Reliability engineering determines an effective test strategy so that all parts are exercised in relevant environments. For systems that must last many years, reliability engineering may be used to design an accelerated life test as well.

Reliability prediction

A prediction of reliability is an important element in the process of selecting equipment for use by telecommunications service providers and other buyers of electronic equipment. Reliability is a measure of the frequency of equipment failures as a function of time. Reliability has a major impact on maintenance and repair costs and on the continuity of service. Reliability predictions:

- Help assess the effect of product reliability on the maintenance activity and on the quantity of spare units required for acceptable field performance of any particular system. For example, predictions of the frequency of unit level maintenance actions can be obtained. Reliability prediction can be used to size spare populations.
- Provide necessary input to system-level reliability models. System-level reliability models can subsequently be used to predict, for example, frequency of system outages in steady-state, frequency of system outages during early life, expected downtime per year, and system availability.
- Provide necessary input to unit and system-level Life Cycle Cost Analyses. Life cycle cost studies determine the cost of a product over its entire life. Therefore, how often a unit will have to be replaced needs to be known. Inputs to this process include unit and system failure rates. This includes how often units and systems fail during the first year of operation as well as in later years.
- Assist in deciding which product to purchase from a list of competing products. As a result, it is essential that reliability predictions be based on a common procedure.
- Can be used to set factory test standards for products requiring a reliability test. Reliability predictions help determine how often the system should fail.
- Are needed as input to the analysis of complex systems such as switching systems and digital cross-connect systems. It is necessary to know how often different parts of the system are going to fail even for redundant components.
- Can be used in design trade-off studies. For example, a supplier could look at a design with many simple devices and compare it to a design with fewer devices that are newer but more complex. The unit with fewer devices is usually more reliable.
- Can be used to set achievable in-service performance standards against which to judge actual performance and stimulate action.

The telecommunications industry has devoted much time over the years to concentrate on developing reliability models for electronic equipment. One such tool is the Automated

Reliability Prediction Procedure (ARPP), which is an Excel-spreadsheet software tool that automates the reliability prediction procedures in SR-332, Reliability Prediction Procedure for Electronic Equipment. FD-ARPP-01 provides suppliers and manufacturers with a tool for making Reliability Prediction Procedure (RPP) calculations. It also provides a means for understanding RPP calculations through the capability of interactive examples provided by the user.

The RPP views electronic systems as hierarchical assemblies. Systems are constructed from units that, in turn, are constructed from devices. The methods presented predict reliability at these three hierarchical levels:

1. *Device*: A basic component (or part)
2. *Unit*: Any assembly of devices. This may include, but is not limited to, circuit packs, modules, plug-in units, racks, power supplies, and ancillary equipment. Unless otherwise dictated by maintenance considerations, a unit will usually be the lowest level of replaceable assemblies/devices. The RPP is aimed primarily at reliability prediction of units.
3. *Serial System*: Any assembly of units for which the failure of any single unit will cause a failure of the system.

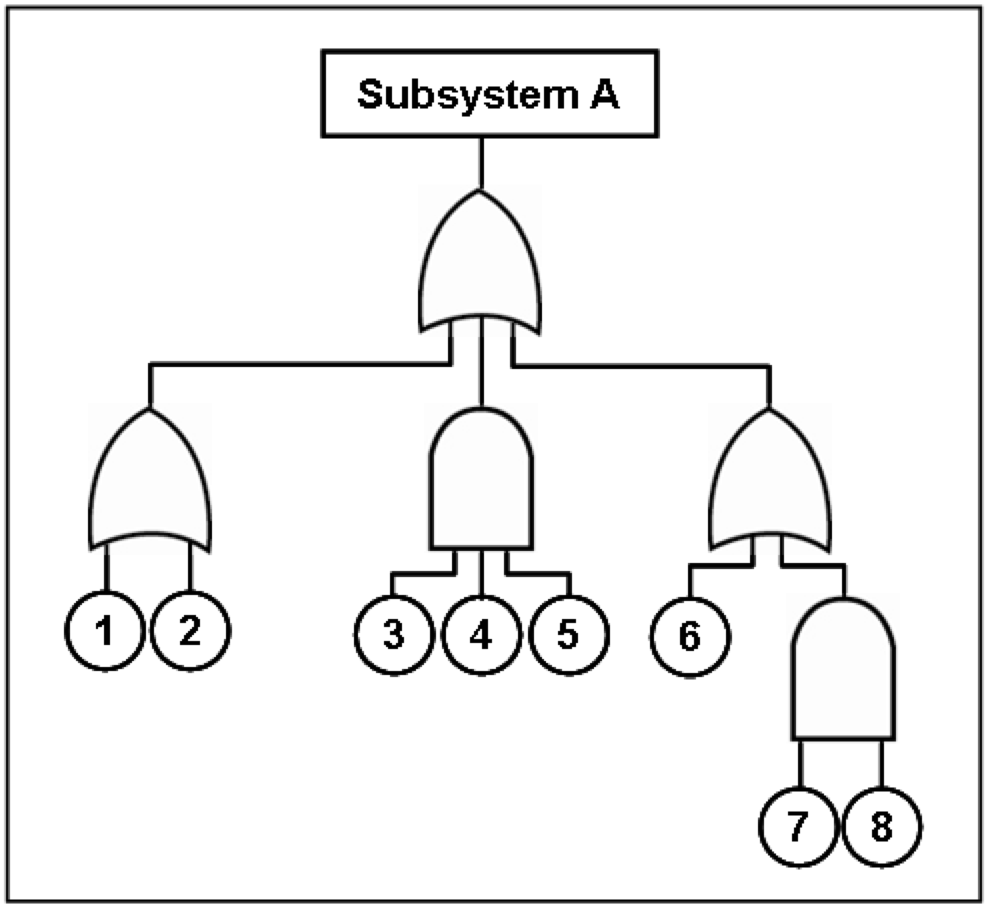
Requirements for reliability tasks

Reliability engineering must also address requirements for various reliability tasks and documentation during system development, test, production, and operation. These requirements are generally specified in the contract statement of work and depend on how much leeway the customer wishes to provide to the contractor. Reliability tasks include various analyses, planning, and failure reporting. Task selection depends on the criticality of the system as well as cost. A critical system may require a formal failure reporting and review process throughout development, whereas a non-critical system may rely on final test reports. The most common reliability program tasks are documented in reliability program standards, such as MIL-STD-785 and IEEE 1332. Failure reporting analysis and corrective action systems are a common approach for product/process reliability monitoring.

Design for reliability

Design For Reliability (DFR), is an emerging discipline that refers to the process of designing reliability into products. This process encompasses several tools and practices and describes the order of their deployment that an organization needs to have in place to drive reliability into their products. Typically, the first step in the DFR process is to set the system's reliability requirements. Reliability must be "designed in" to the system. During system design, the top-level reliability requirements are then allocated to subsystems by design engineers and reliability engineers working together.

Reliability design begins with the development of a model. Reliability models use **block diagrams** and **fault trees** to provide a graphical means of evaluating the relationships between different parts of the system. These models incorporate predictions based on parts-count failure rates taken from historical data. While the predictions are often not accurate in an absolute sense, they are valuable to assess relative differences in design alternatives.



A Fault Tree Diagram

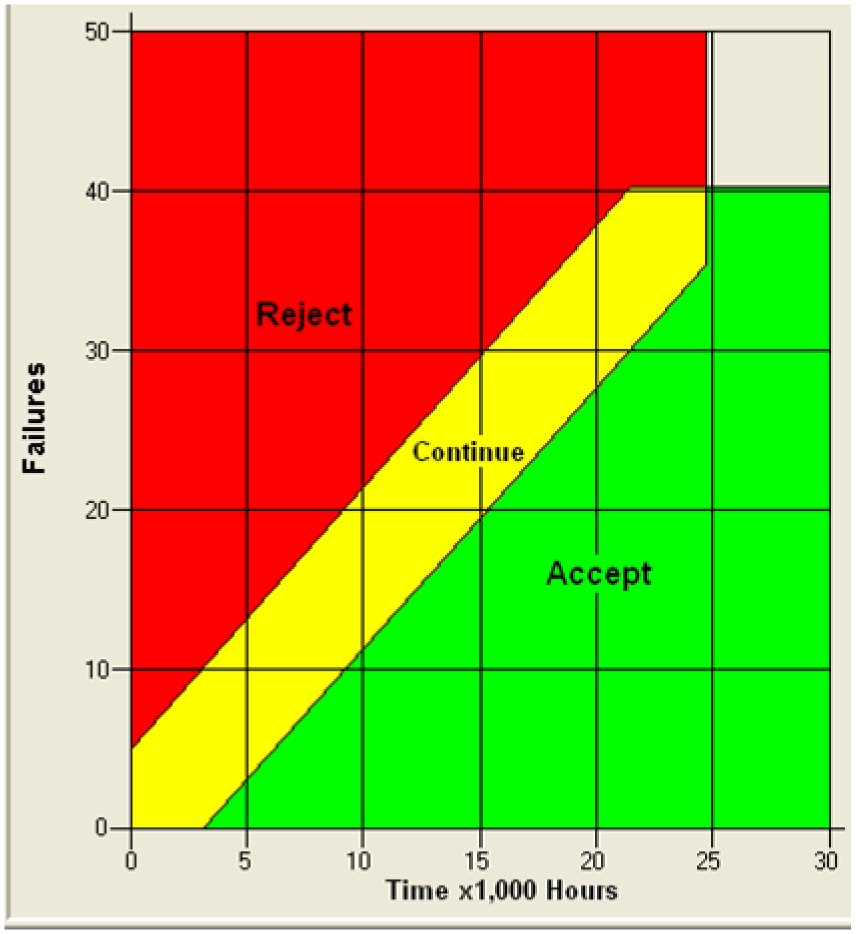
One of the most important design techniques is **redundancy**. This means that if one part of the system fails, there is an alternate success path, such as a backup system. An automobile brake light might use two light bulbs. If one bulb fails, the brake light still operates using the other bulb. Redundancy significantly increases system reliability, and is often the only viable means of doing so. However, redundancy is difficult and expensive, and is therefore limited to critical parts of the system. Another design technique, **physics of failure**, relies on understanding the physical processes of stress, strength and failure at a very detailed level. Then the material or component can be re-designed to reduce the probability of failure. Another common design technique is **component derating**: Selecting components whose tolerance significantly exceeds the expected stress, as using a heavier gauge wire that exceeds the normal specification for the expected electrical current.

Many tasks, techniques and analyses are specific to particular industries and applications. Commonly these include:

- Built-in test (BIT)
- Failure mode and effects analysis (FMEA)
- Reliability simulation modeling
- Thermal analysis
- Reliability Block Diagram analysis
- Fault tree analysis
- Root cause analysis
- Sneak circuit analysis
- Accelerated Testing
- Reliability Growth analysis
- Weibull analysis
- Electromagnetic analysis
- Statistical interference
- Avoid Single Point of Failure

Results are presented during the system design reviews and logistics reviews. Reliability is just one requirement among many system requirements. Engineering trade studies are used to determine the optimum balance between reliability and other requirements and constraints.

Reliability testing



A Reliability Sequential Test Plan

The purpose of reliability testing is to discover potential problems with the design as early as possible and, ultimately, provide confidence that the system meets its reliability requirements.

Reliability testing may be performed at several levels. Complex systems may be tested at component, circuit board, unit, assembly, subsystem and system levels. (The test level nomenclature varies among applications.) For example, performing environmental stress screening tests at lower levels, such as piece parts or small assemblies, catches problems before they cause failures at higher levels. Testing proceeds during each level of integration through full-up system testing, developmental testing, and operational testing, thereby reducing program risk. System reliability is calculated at each test level. Reliability growth techniques and failure reporting, analysis and corrective active systems (FRACAS) are often employed to improve reliability as testing progresses. The drawbacks to such extensive testing are time and expense. Customers may choose to accept more risk by eliminating some or all lower levels of testing.

Another type of tests are called Sequental Probability Ratio type of tests. These tests use both a statistical type 1 and type 2 error, combined with a discrimination ratio as main input (together with the R requirement). This test sets - Independently - before the start of the test both the risk of incorrectly accepting a bad design (Type 2 error) and the risk of incorrectly rejecting a good design (type 1 error) together with the discrimination ratio and the required minimum reliability parameter. The test is therefore more controllable and provides more information for a quality and business point of view. The number of test samples is not fixed, but it is said (reference needed...) that this test is in general more efficient (requires less samples) and provides more information than for example zero failure testing.

It is not always feasible to test all system requirements. Some systems are prohibitively expensive to test; some failure modes may take years to observe; some complex interactions result in a huge number of possible test cases; and some tests require the use of limited test ranges or other resources. In such cases, different approaches to testing can be used, such as accelerated life testing, design of experiments, and simulations.

The desired level of statistical confidence also plays an important role in reliability testing. Statistical confidence is increased by increasing either the test time or the number of items tested. Reliability test plans are designed to achieve the specified reliability at the specified confidence level with the minimum number of test units and test time. Different test plans result in different levels of risk to the producer and consumer. The desired reliability, statistical confidence, and risk levels for each side influence the ultimate test plan. Good test requirements ensure that the customer and developer agree in advance on how reliability requirements will be tested.

A key aspect of reliability testing is to define "failure". Although this may seem obvious, there are many situations where it is not clear whether a failure is really the fault of the system. Variations in test conditions, operator differences, weather, and unexpected situations create differences between the customer and the system developer. One strategy to address this issue is to use a **scoring conference** process. A scoring conference includes representatives from the customer, the developer, the test organization, the reliability organization, and sometimes independent observers. The scoring conference process is defined in the statement of work. Each test case is considered by the group and "scored" as a success or failure. This scoring is the official result used by the reliability engineer.

As part of the requirements phase, the reliability engineer develops a test strategy with the customer. The test strategy makes trade-offs between the needs of the reliability organization, which wants as much data as possible, and constraints such as cost, schedule, and available resources. Test plans and procedures are developed for each reliability test, and results are documented in official reports.

Accelerated testing

The purpose of accelerated life testing is to induce field failure in the laboratory at a much faster rate by providing a harsher, but nonetheless representative, environment. In such a test the product is expected to fail in the lab just as it would have failed in the field—but in much less time. The main objective of an accelerated test is either of the following:

- To discover failure modes
- To predict the normal field life from the high stress lab life

An **Accelerated testing** program can be broken down into the following steps:

- Define objective and scope of the test
- Collect required information about the product
- Identify the stress(es)
- Determine level of stress(es)
- Conduct the Accelerated test and analyse the accelerated data.

Common way to determine a life stress relationship are

- Arrhenius Model
- Eyring Model
- Inverse Power Law Model
- Temperature-Humidity Model
- Temperature Non-thermal Model

Software reliability

Software reliability is a special aspect of reliability engineering. System reliability, by definition, includes all parts of the system, including hardware, software, operators and procedures. Traditionally, reliability engineering focuses on critical hardware parts of the system. Since the widespread use of digital integrated circuit technology, software has become an increasingly critical part of most electronics and, hence, nearly all present day systems. There are significant differences, however, in how software and hardware behave. Most hardware unreliability is the result of a component or material failure that results in the system not performing its intended function. Repairing or replacing the hardware component restores the system to its original unfailed state. However, software does not fail in the same sense that hardware fails. Instead, software unreliability is the result of unanticipated results of software operations. Even relatively small software programs can have astronomically large combinations of inputs and states that are infeasible to exhaustively test. Restoring software to its original state only works until the same combination of inputs and states results in the same unintended result. Software reliability engineering must take this into account.

Despite this difference in the source of failure between software and hardware — software does not wear out — some in the software reliability engineering community believe statistical models used in hardware reliability are nevertheless useful as a measure of software reliability, describing what we experience with software: the longer you run software, the higher the probability you will eventually use it in an untested manner and find a latent defect that results in a failure (Shooman 1987), (Musa 2005), (Denney 2005).

As with hardware, software reliability depends on good requirements, design and implementation. Software reliability engineering relies heavily on a disciplined software engineering process to anticipate and design against unintended consequences. There is more overlap between software quality engineering and software reliability engineering than between hardware quality and reliability. A good software development plan is a key aspect of the software reliability program. The software development plan describes the design and coding standards, peer reviews, unit tests, configuration management, software metrics and software models to be used during software development.

A common reliability metric is the number of software faults, usually expressed as faults per thousand lines of code. This metric, along with software execution time, is key to most software reliability models and estimates. The theory is that the software reliability increases as the number of faults (or fault density) goes down. Establishing a direct connection between fault density and mean-time-between-failure is difficult, however, because of the way software faults are distributed in the code, their severity, and the probability of the combination of inputs necessary to encounter the fault. Nevertheless, fault density serves as a useful indicator for the reliability engineer. Other software metrics, such as complexity, are also used.

Testing is even more important for software than hardware. Even the best software development process results in some software faults that are nearly undetectable until tested. As with hardware, software is tested at several levels, starting with individual units, through integration and full-up system testing. Unlike hardware, it is inadvisable to skip levels of software testing. During all phases of testing, software faults are discovered, corrected, and re-tested. Reliability estimates are updated based on the fault density and other metrics. At system level, mean-time-between-failure data are collected and used to estimate reliability. Unlike hardware, performing exactly the same test on exactly the same software configuration does not provide increased statistical confidence. Instead, software reliability uses different metrics such as test coverage.

Eventually, the software is integrated with the hardware in the top-level system, and software reliability is subsumed by system reliability. The Software Engineering Institute's Capability Maturity Model is a common means of assessing the overall software development process for reliability and quality purposes. However, actual software reliability is served through SAE standards JA1002 and JA1003.

Reliability Operational Assessment

After a system is produced, reliability engineering monitors, assesses, and corrects deficiencies. Monitoring includes electronic and visual surveillance of critical parameters identified during the fault tree analysis design stage. The data are constantly analyzed using statistical techniques, such as Weibull analysis and linear regression, to ensure the system reliability meets requirements. Reliability data and estimates are also key inputs for system logistics. Data collection is highly dependent on the nature of the system. Most large organizations have quality control groups that collect failure data on vehicles, equipment, and machinery. Consumer product failures are often tracked by the number of returns. For systems in dormant storage or on standby, it is necessary to establish a formal surveillance program to inspect and test random samples. Any changes to the system, such as field upgrades or recall repairs, require additional reliability testing to ensure the reliability of the modification. Since it is not possible to anticipate all the failure modes of a given system, especially ones with a human element, failures will occur. The reliability program also includes a systematic root cause analysis that identifies the causal relationships involved in the failure such that effective corrective actions may be implemented. When possible, system failures and corrective actions are reported to the reliability engineering organization.

One of the most common methods to apply a Reliability Operational Assessment are Failure Reporting, Analysis and Corrective Action Systems (FRACAS). This systematic approach develops a reliability, safety and logistics assessment based on Failure / Incident reporting, management, analysis and corrective/preventive actions. Organizations today are adopting this method and utilize commercial systems such as a Web based FRACAS application enabling and organization to create a failure/incident data repository from which statistics can be derived to view accurate and genuine reliability, safety and quality performances.

Some of the common outputs from a FRACAS system includes: Field MTBF, MTTR, Spares Consumption, Reliability Growth, Failure/Incidents distribution by type, location, part no., serial no, symptom etc.

Reliability organizations

Systems of any significant complexity are developed by organizations of people, such as a commercial company or a government agency. The reliability engineering organization must be consistent with the company's organizational structure. For small, non-critical systems, reliability engineering may be informal. As complexity grows, the need arises for a formal reliability function. Because reliability is important to the customer, the customer may even specify certain aspects of the reliability organization.

There are several common types of reliability organizations. The project manager or chief engineer may employ one or more reliability engineers directly. In larger organizations, there is usually a product assurance or specialty engineering organization, which may include reliability, maintainability, quality, safety, human factors, logistics, etc. In such case, the reliability engineer reports to the product assurance manager or specialty engineering manager.

In some cases, a company may wish to establish an independent reliability organization. This is desirable to ensure that the system reliability, which is often expensive and time consuming, is not unduly slighted due to budget and schedule pressures. In such cases, the reliability engineer works for the project day-to-day, but is actually employed and paid by a separate organization within the company.

Because reliability engineering is critical to early system design, it has become common for reliability engineers, however the organization is structured, to work as part of an integrated product team.

Certification

The American Society for Quality has a program to become a Certified Reliability Engineer, CRE. Certification is based on education, experience, and a certification test: periodic recertification is required. The body of knowledge for the test includes: reliability management, design evaluation, product safety, statistical tools, design and development, modeling, reliability testing, collecting and using data, etc.

Another highly respected certification program is the CRP (Certified Reliability Professional). To achieve certification, candidates must complete a series of courses focused on important Reliability Engineering topics, successfully apply the learned body of knowledge in the workplace and publicly present this expertise in an industry conference or journal.