Fundamental Concepts, Elements and Applications of

Systems Engineering

Jeremy Ziegler

First Edition, 2012

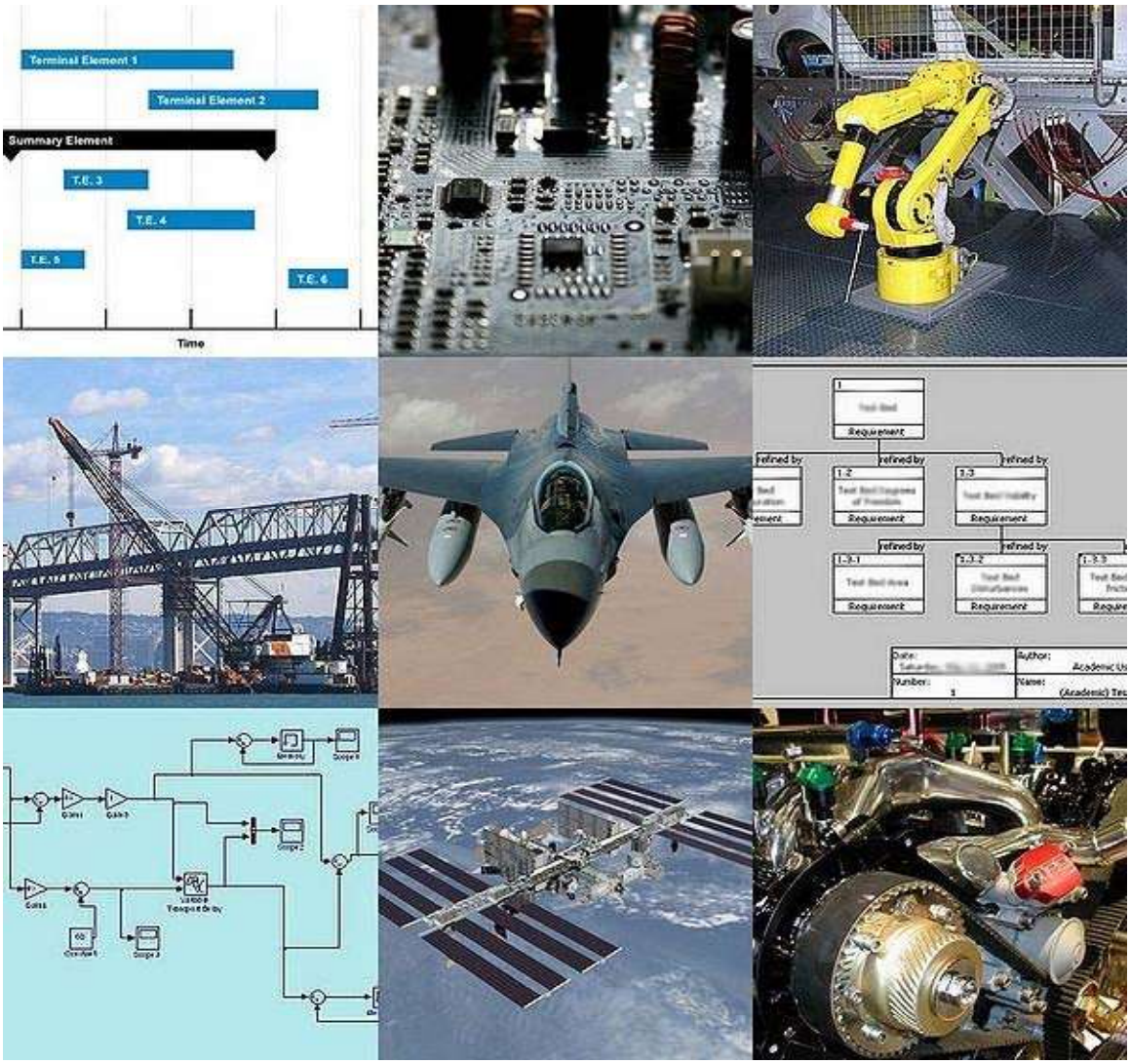
ISBN 978-81-323-0885-0

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

Table of Contents

Chapter- 1

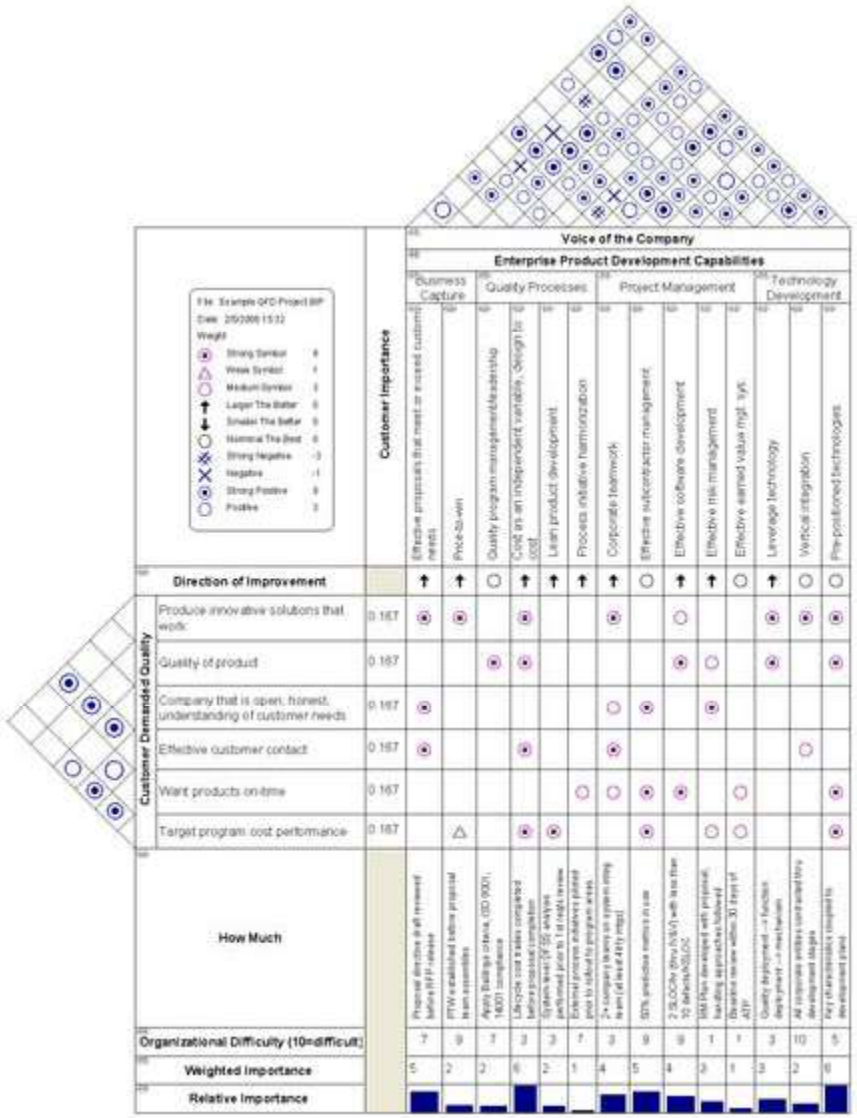
Systems Engineering



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



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

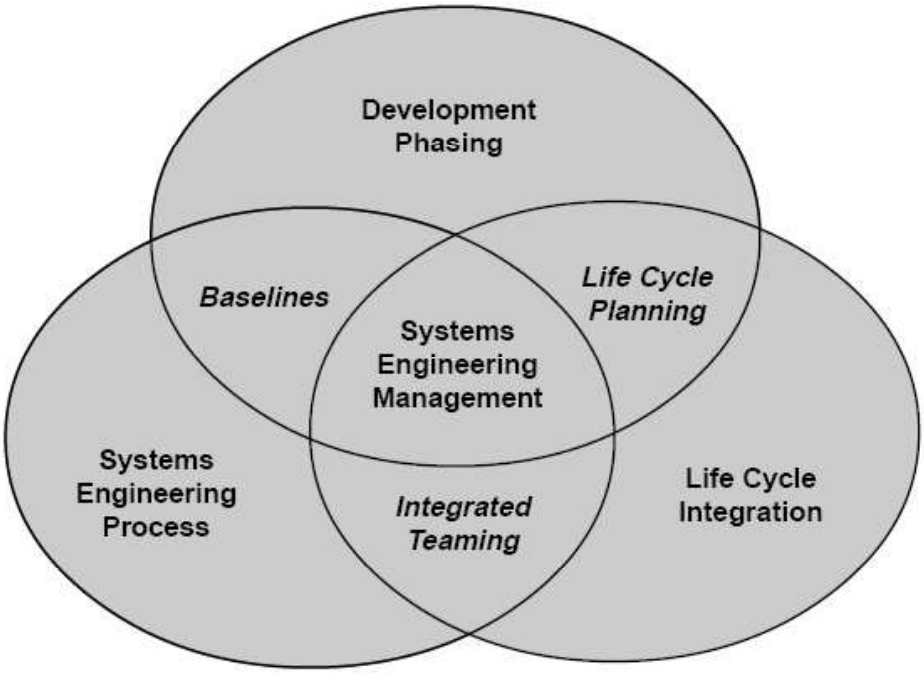
The need for systems engineering arose with the increase in complexity of systems and projects. When speaking in this context, complexity incorporates not only engineering systems, but also the logical human organization of data. At the same time, a system can become more complex due to an increase in 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.

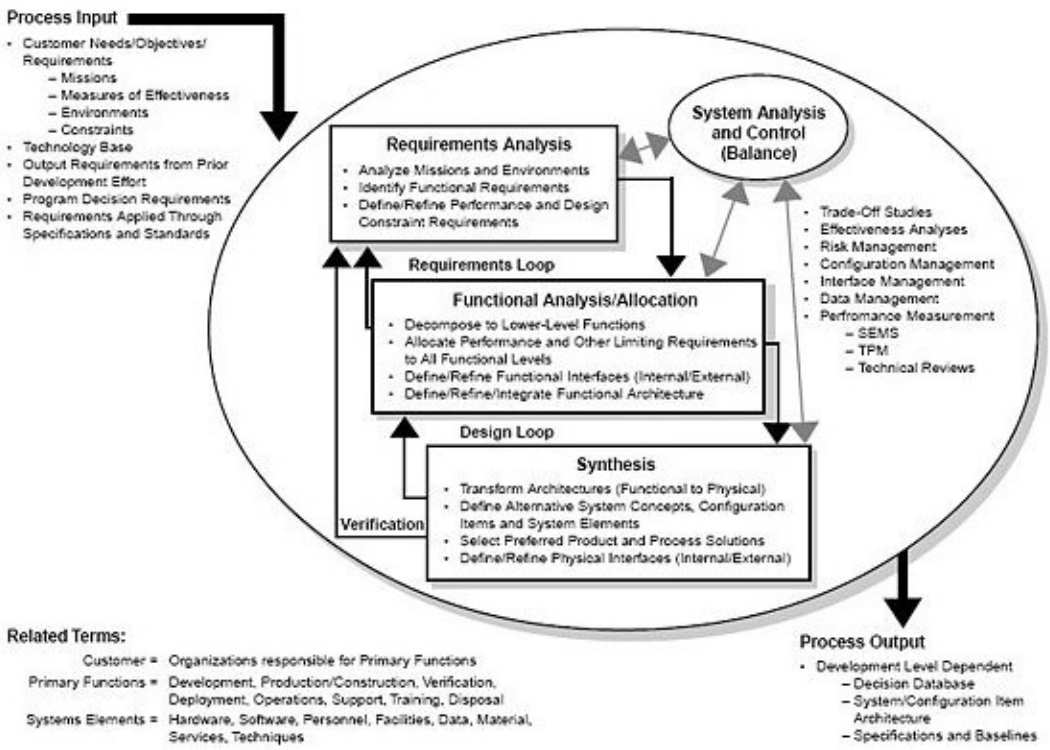
System

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:



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

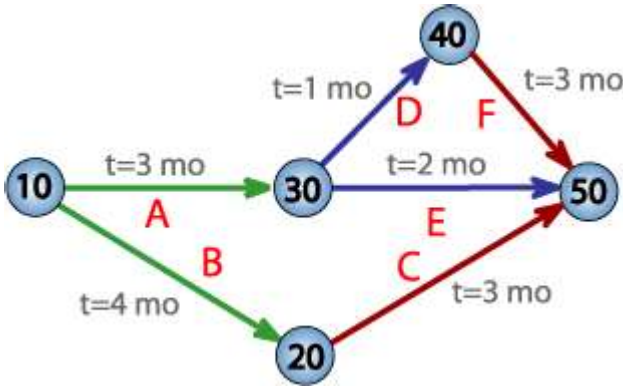
Roman Soldiers Building a Fortress, Trajan's Column 113 AD

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.

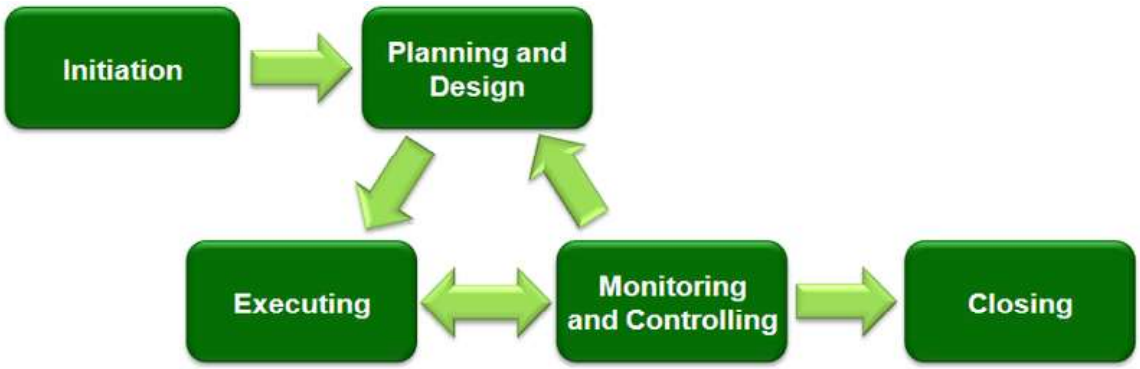
Approaches

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.

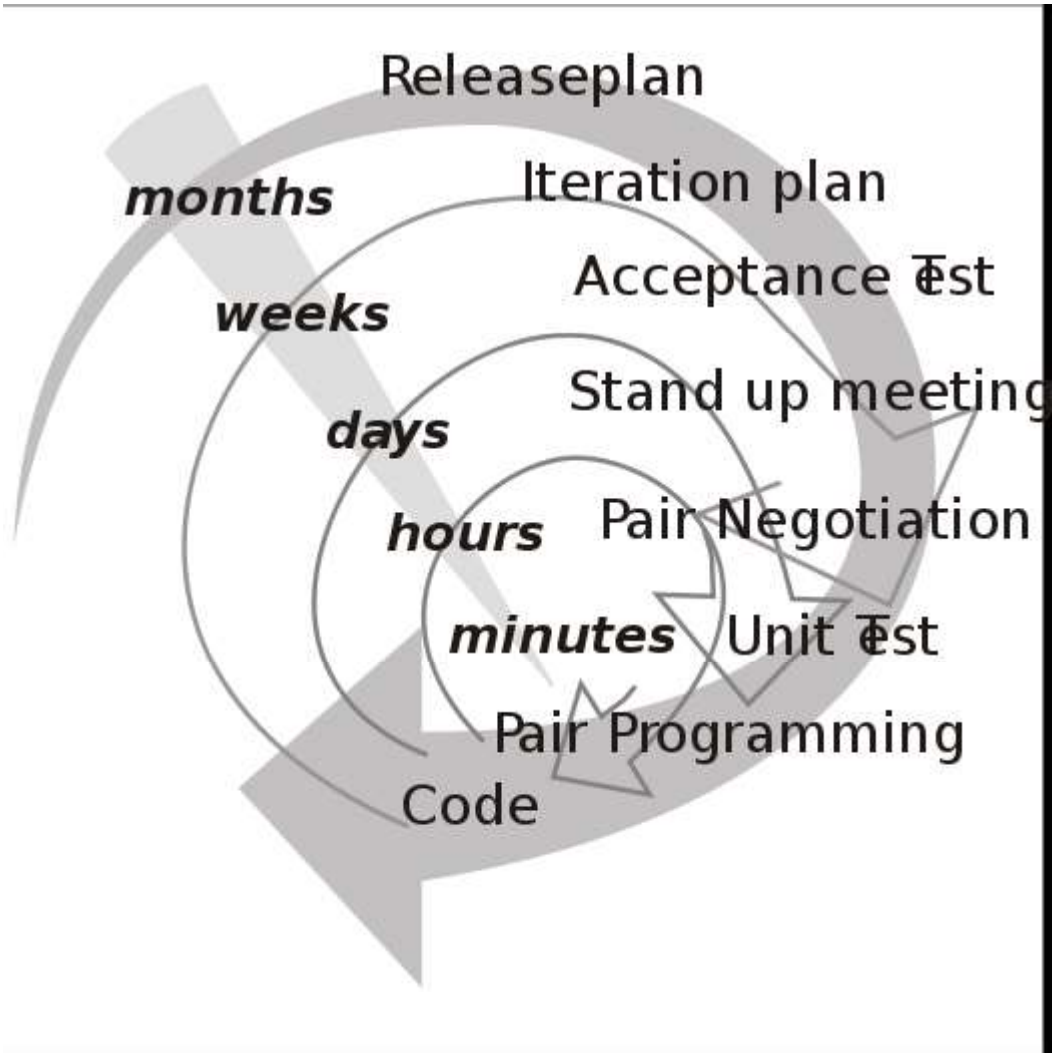
Many industries use variations on these project stages. For example, when working on a brick and mortar design and construction, projects will typically progress through stages like Pre-Planning, Conceptual Design, Schematic Design, Design Development, Construction Drawings (or Contract Documents), and 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.

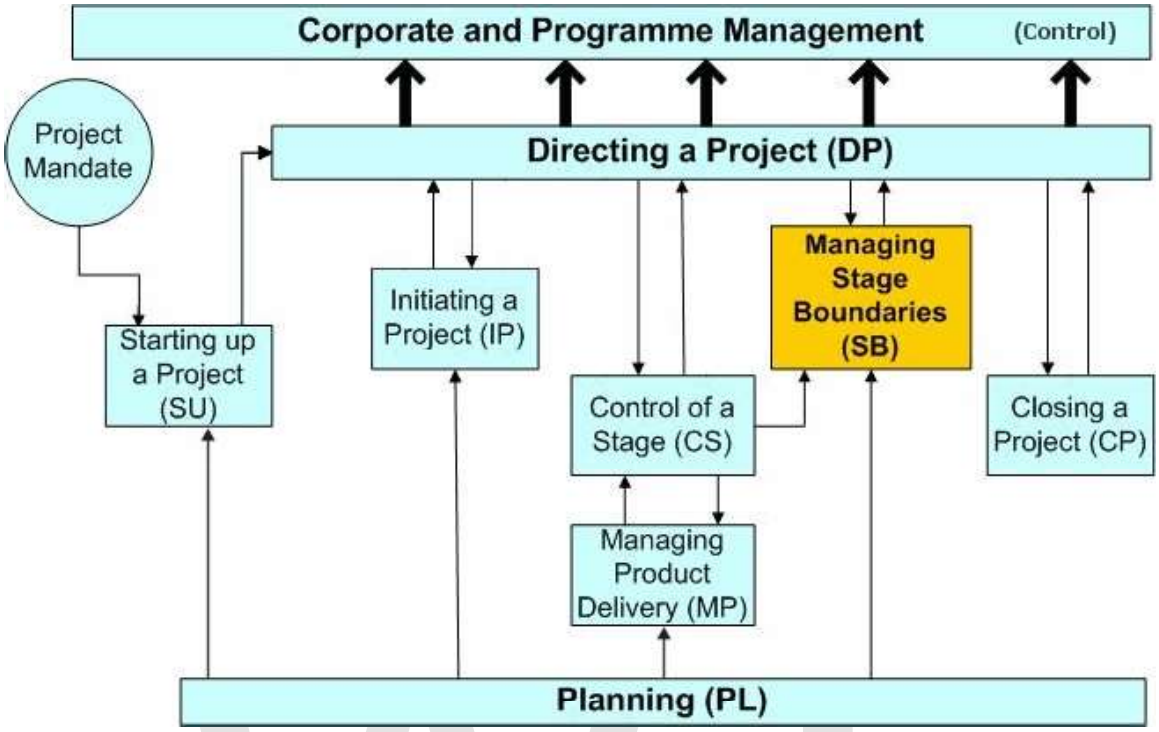
The generalization of Extreme Programming to other kinds of projects is extreme project management, which may be used in combination with the process modeling and management principles of human interaction management.

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.



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

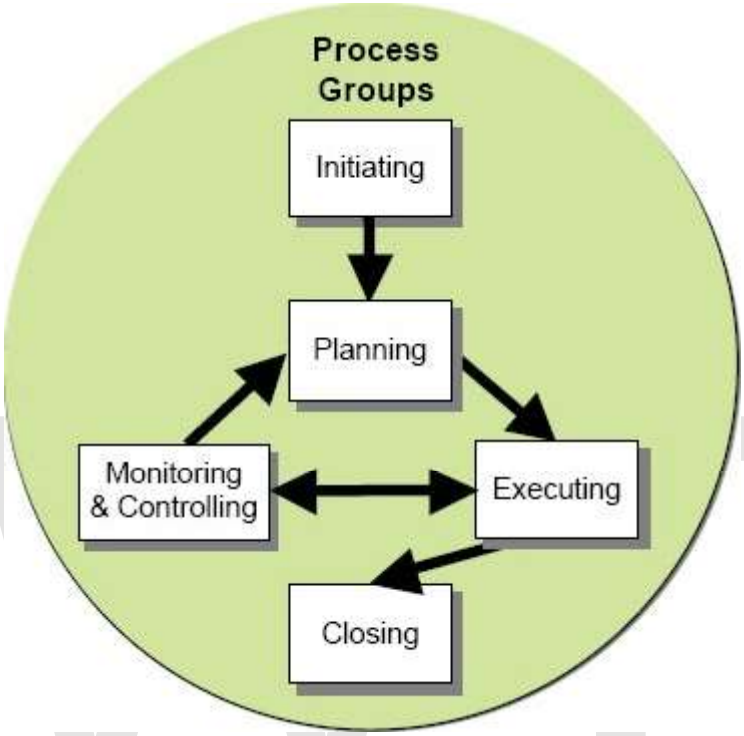
Also furthering the concept of project control is the incorporation of process-based management. This area has been driven by the use of Maturity models such as the CMMI (Capability Maturity Model Integration) and ISO/IEC15504 (SPICE - Software Process Improvement and Capability Estimation).

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

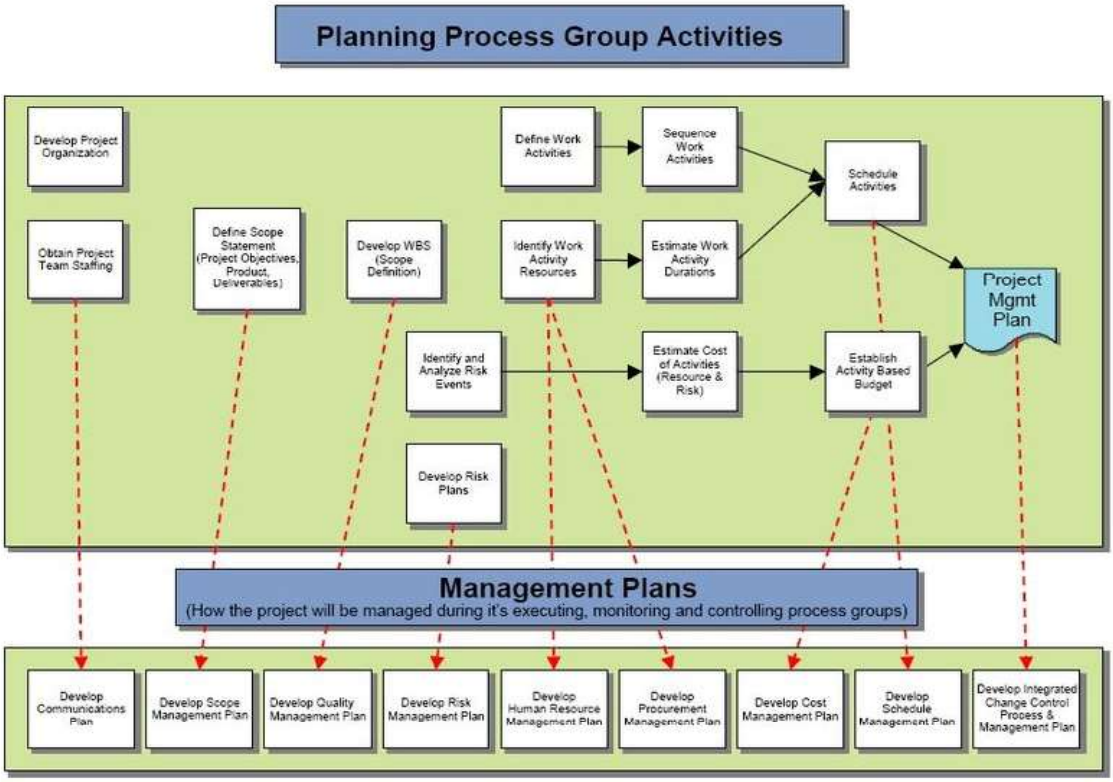
Initiating Process Group Processes

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

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.

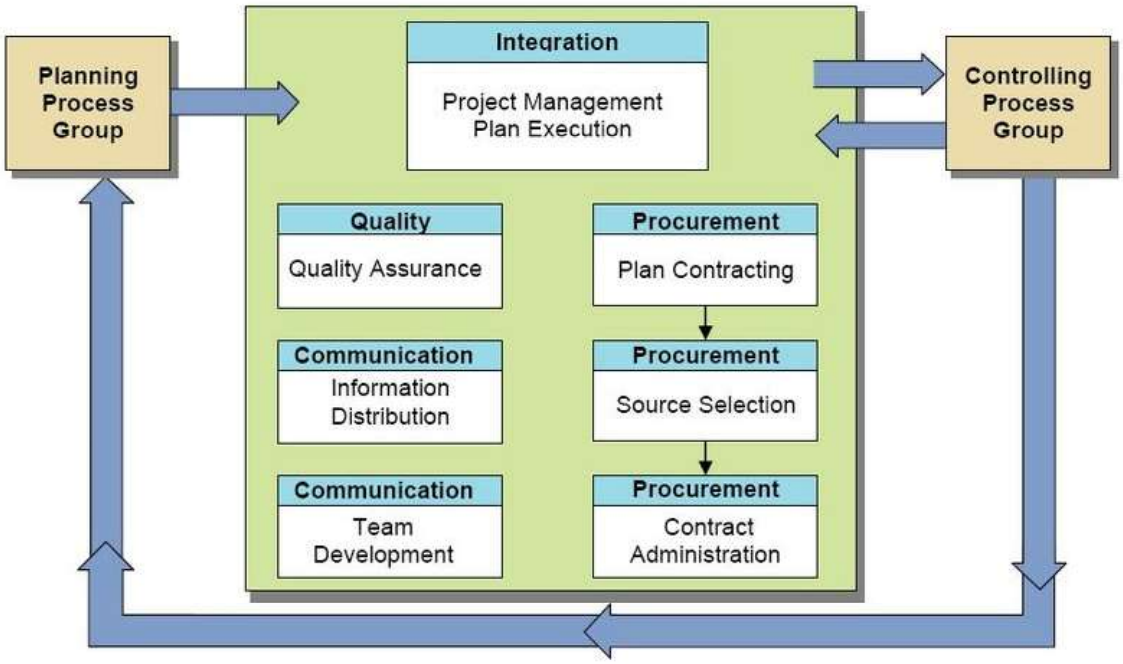
Project planning generally consists of

- 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

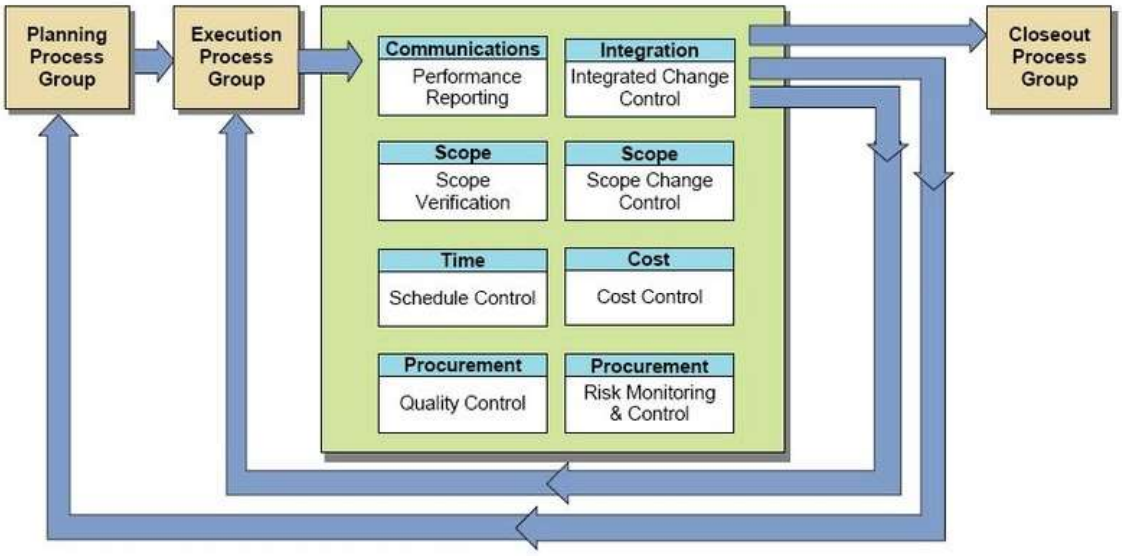


Executing Process Group Processes

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

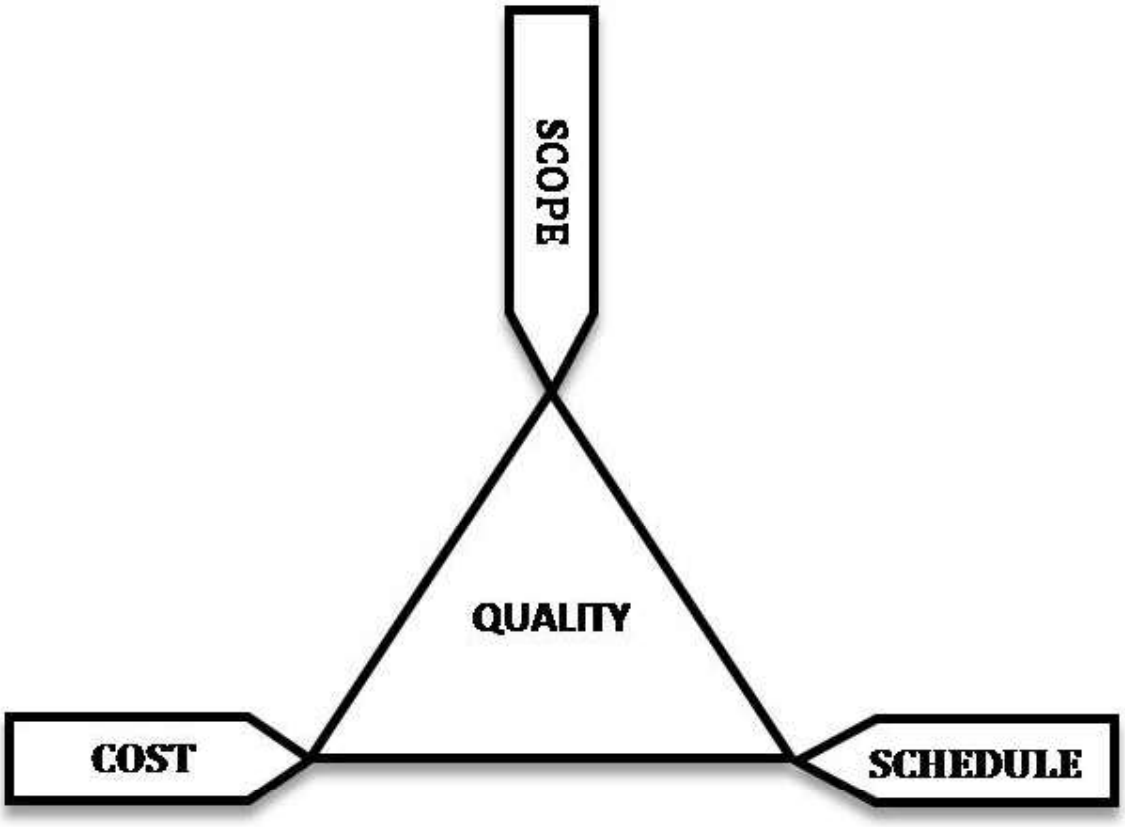
Project managers

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



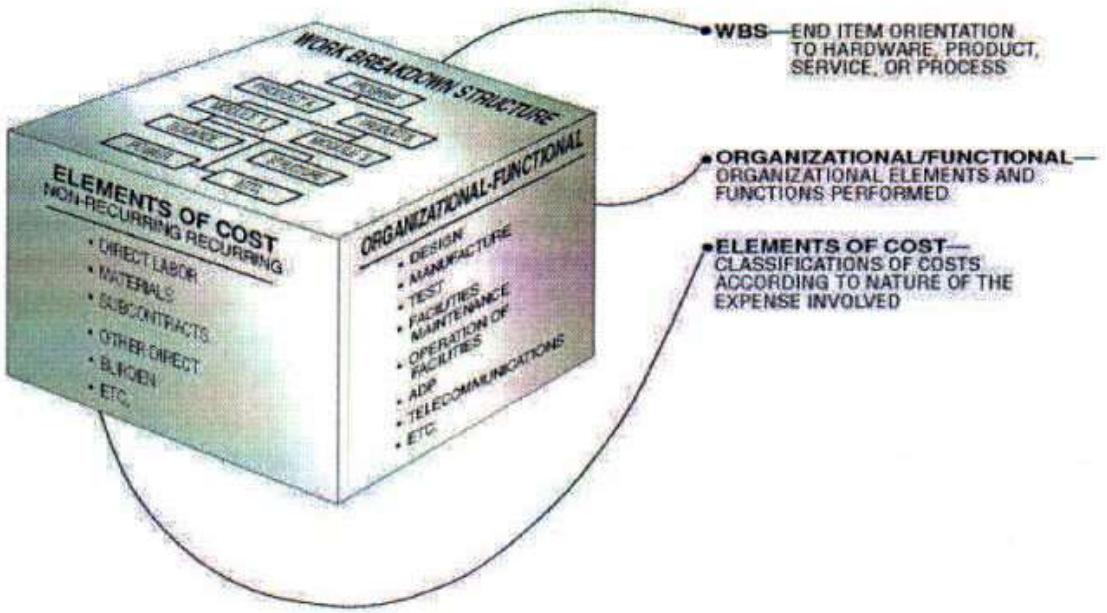
The Project Management Triangle

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



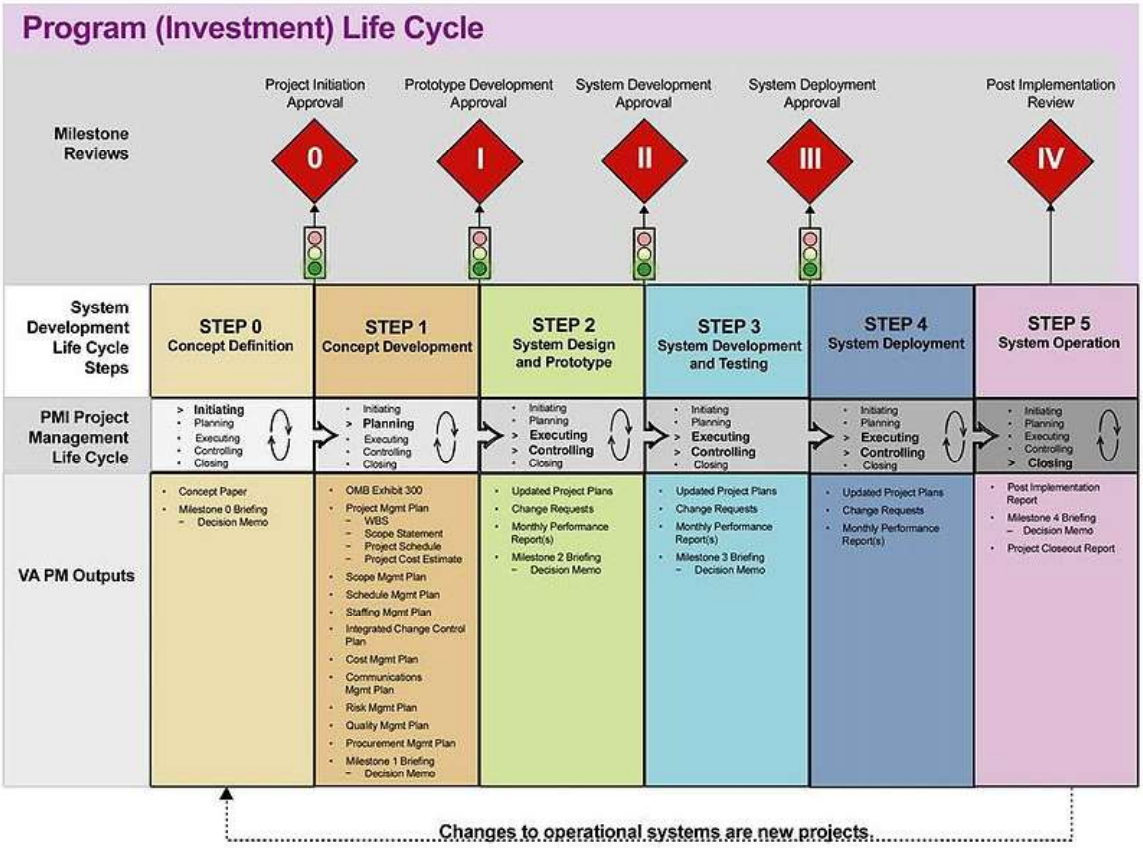
Example of a Work breakdown structure applied in a NASA reporting structure

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



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.

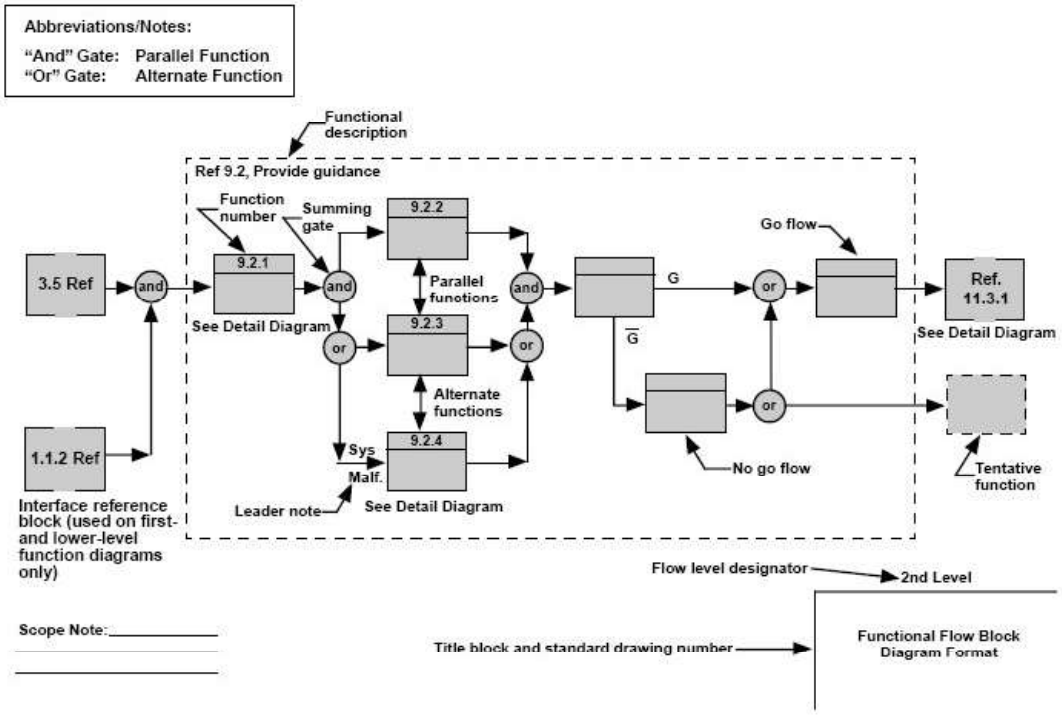
Project portfolio management

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

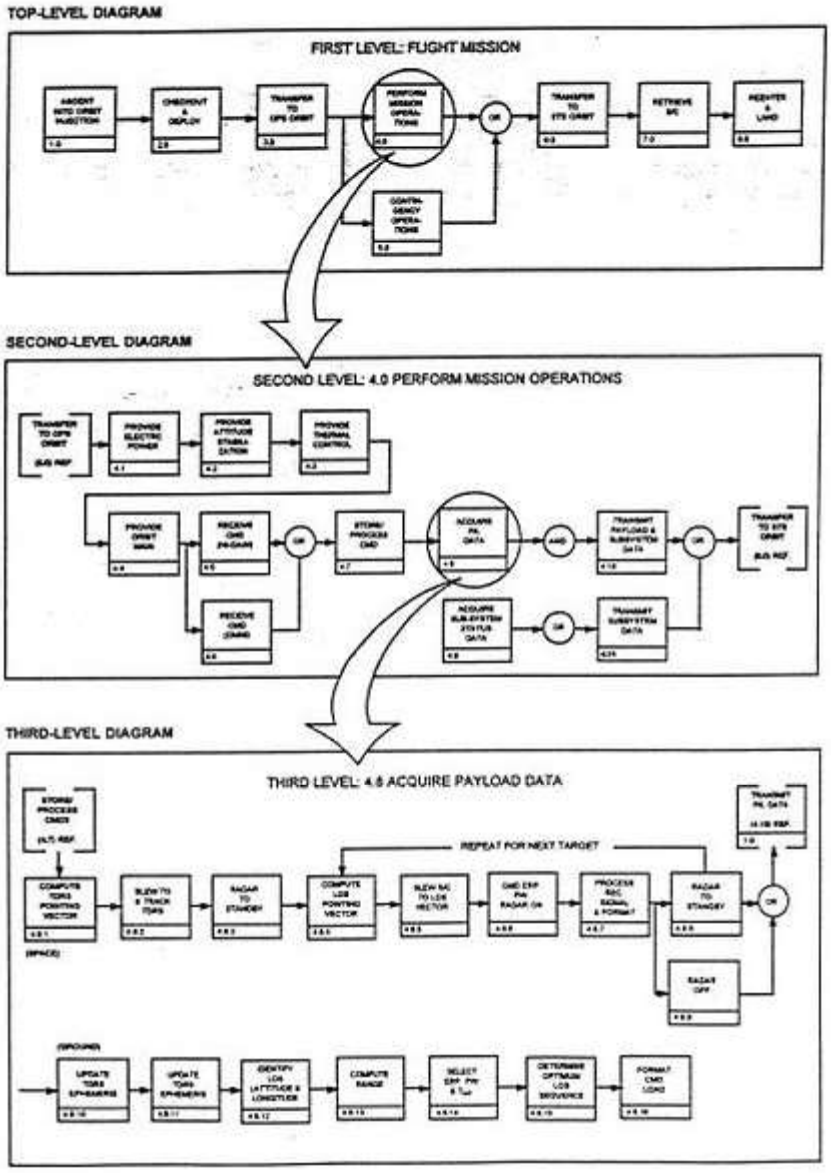


Figure 2: Development of Functional Flow Block Diagrams

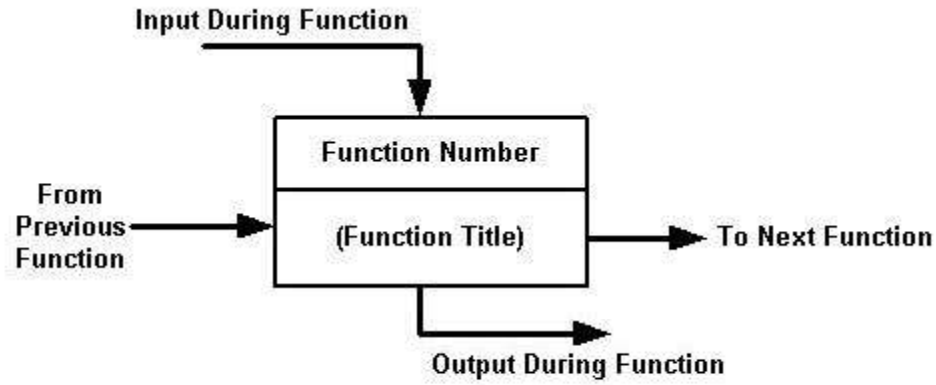
FFBDs can be developed in a series of levels. FFBDs show the same tasks identified through functional decomposition and display them in their logical, sequential relationship. For example, the entire flight mission of a spacecraft can be defined in a top level FFBD, as shown in Figure 2. Each block in 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.

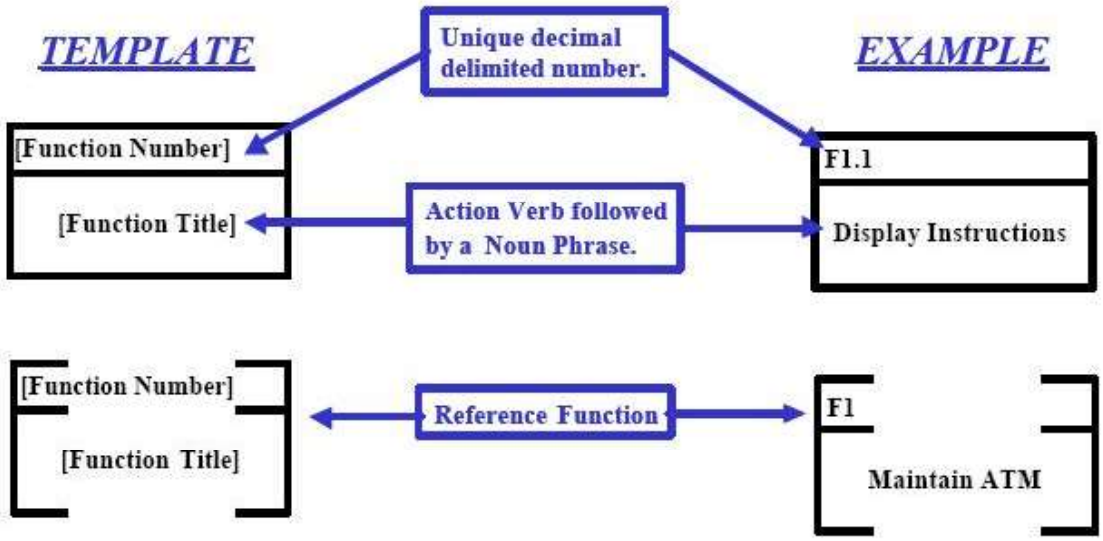


Figure 3. Function Symbol

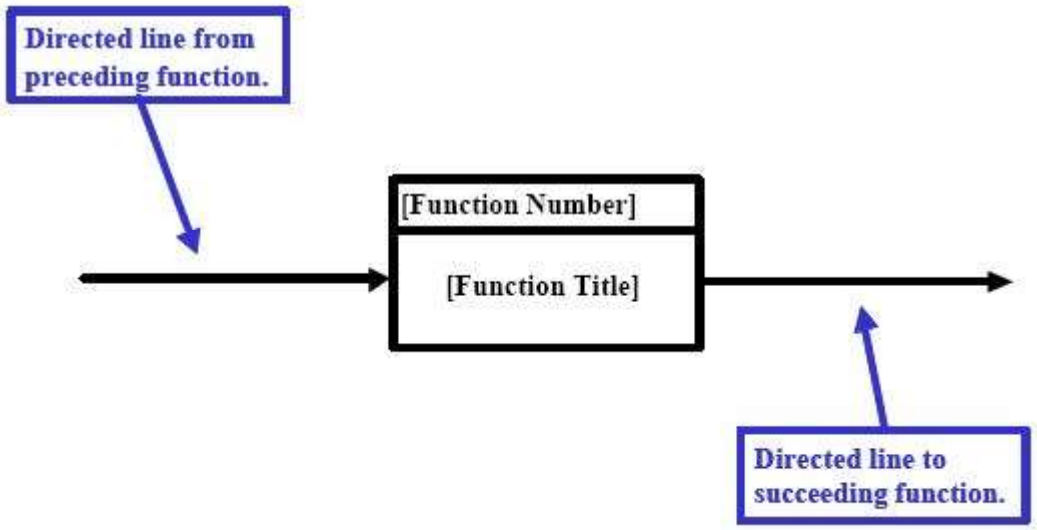


Figure 4. Directed Lines

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.

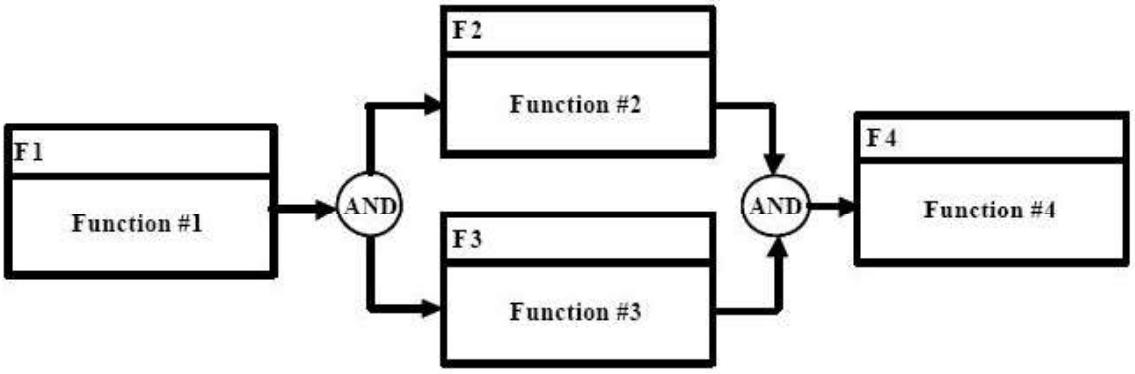


Figure 5. "AND" Symbol

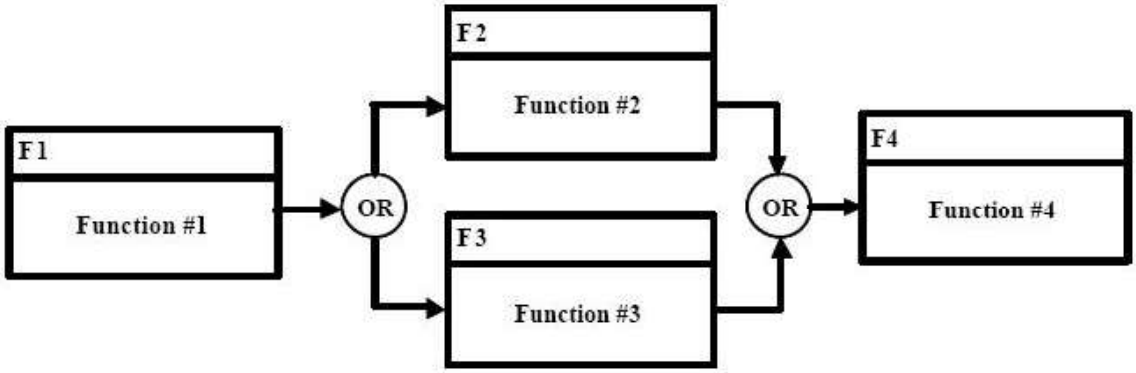


Figure 6. "Exclusive OR" Symbol

- 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

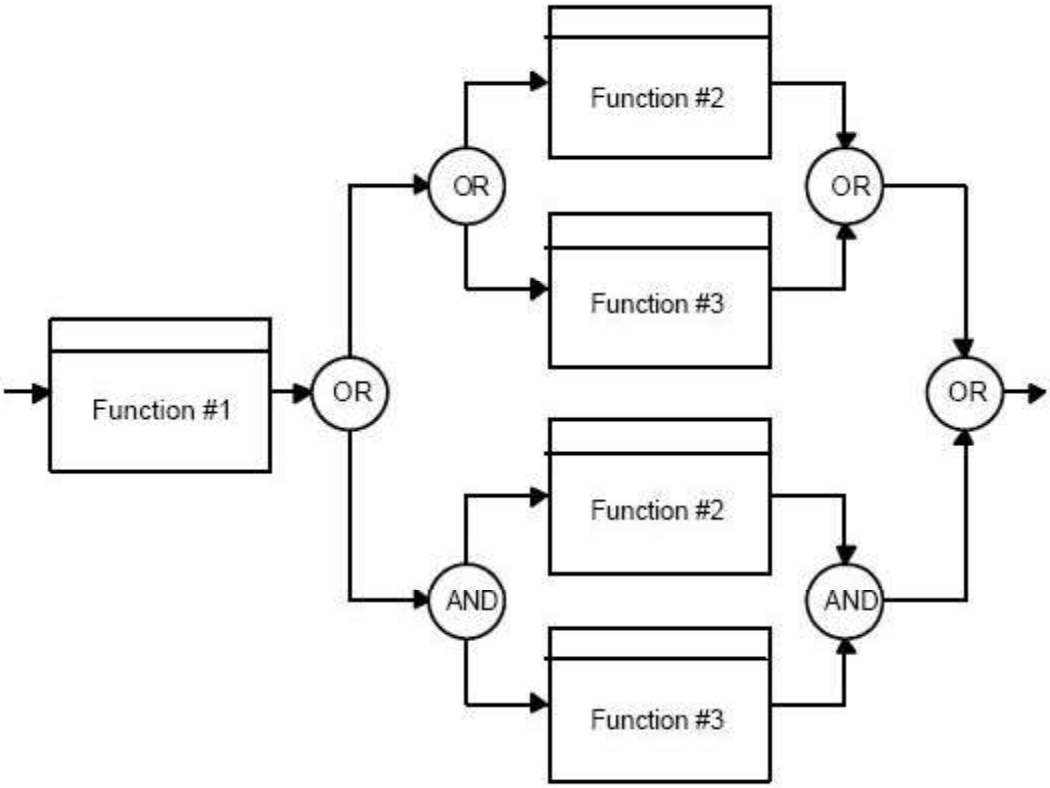


Figure 7. "Inclusive OR" Logic

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.

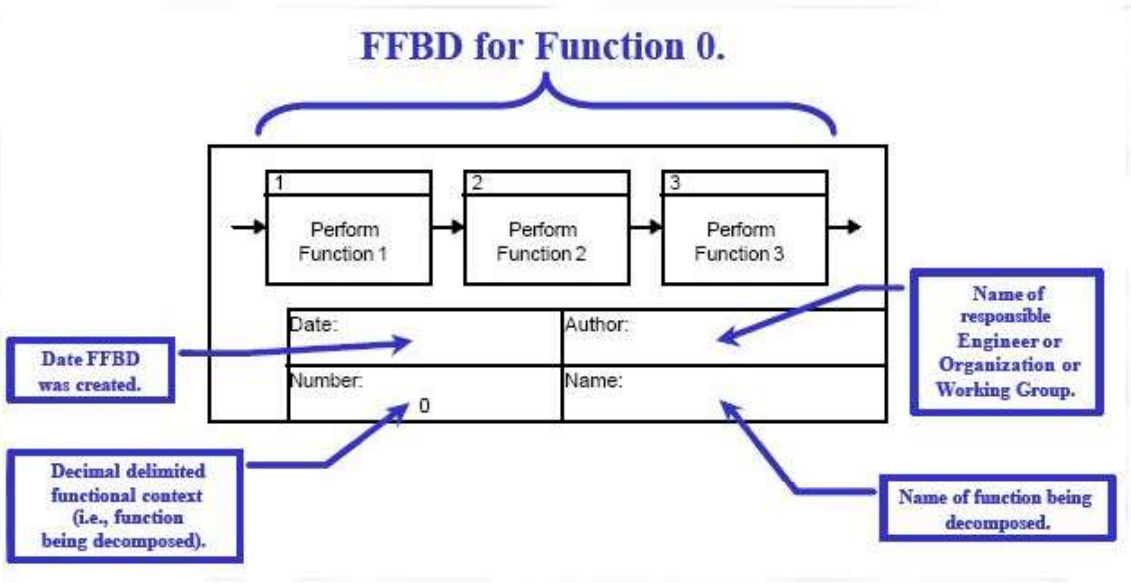


Figure 8. FFBD Function 0 Illustration

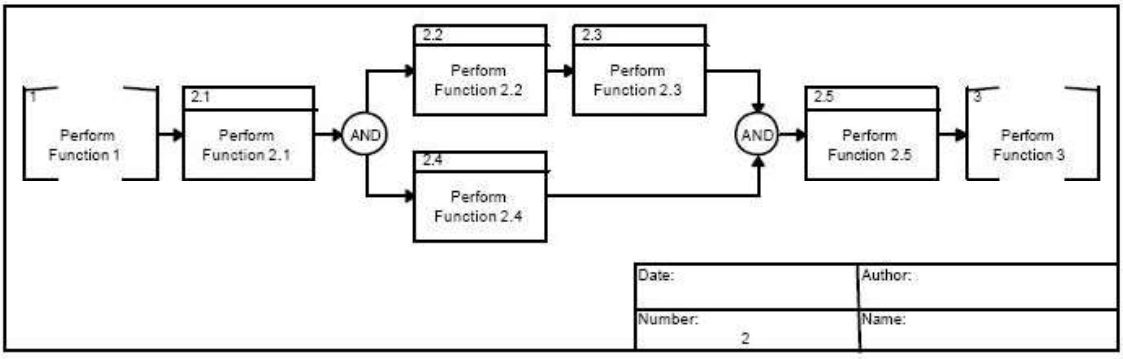
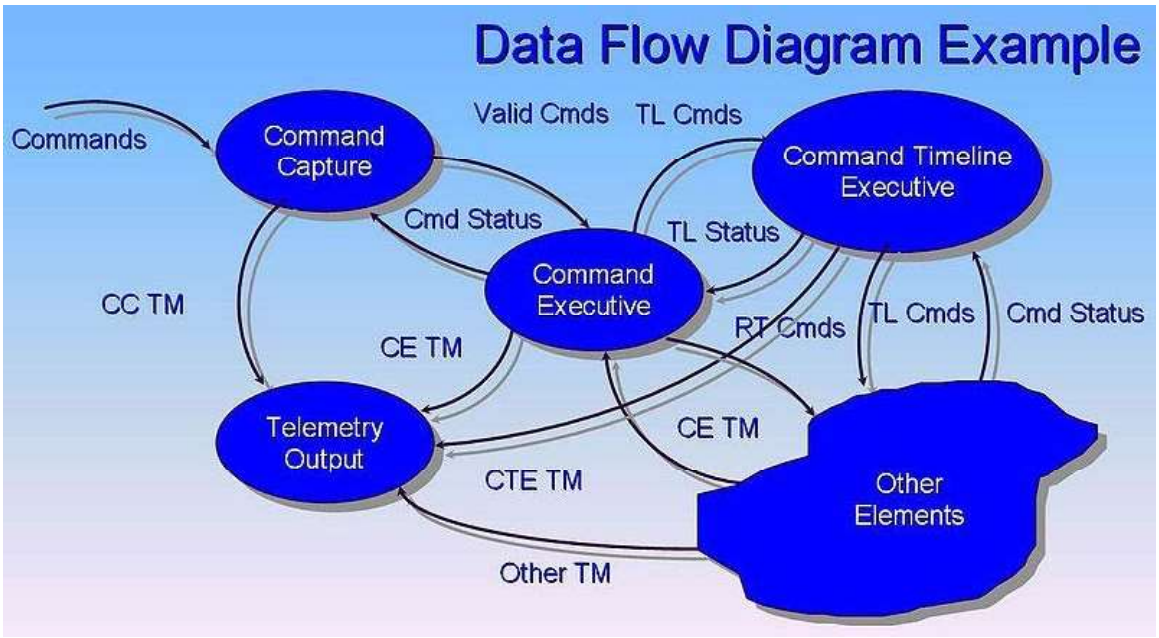


Figure 9. FFBD Function 2 Illustration

Chapter- 6

Data Flow Diagram and N2 Chart

Data flow diagram



Data flow diagram example

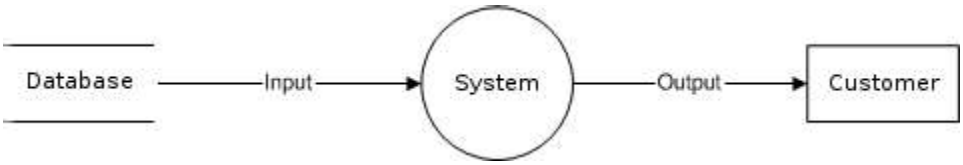
A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

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.

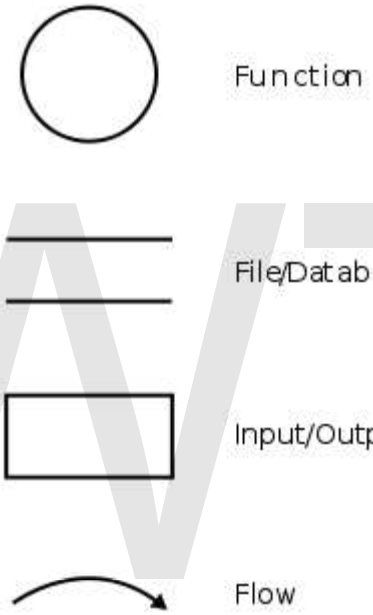
A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in 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.

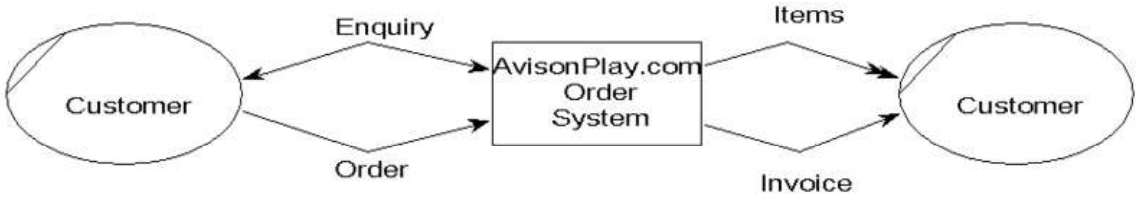
In the course of developing a set of *levelled* data flow diagrams the analyst/designers is forced to address how the system may be decomposed into component sub-systems, and to identify the transaction data in the data model.

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*.



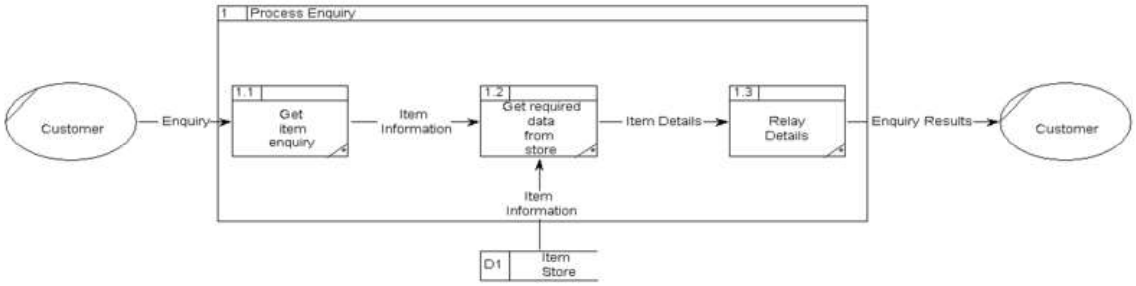
A context level Data flow diagram created using Select SSADM

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.

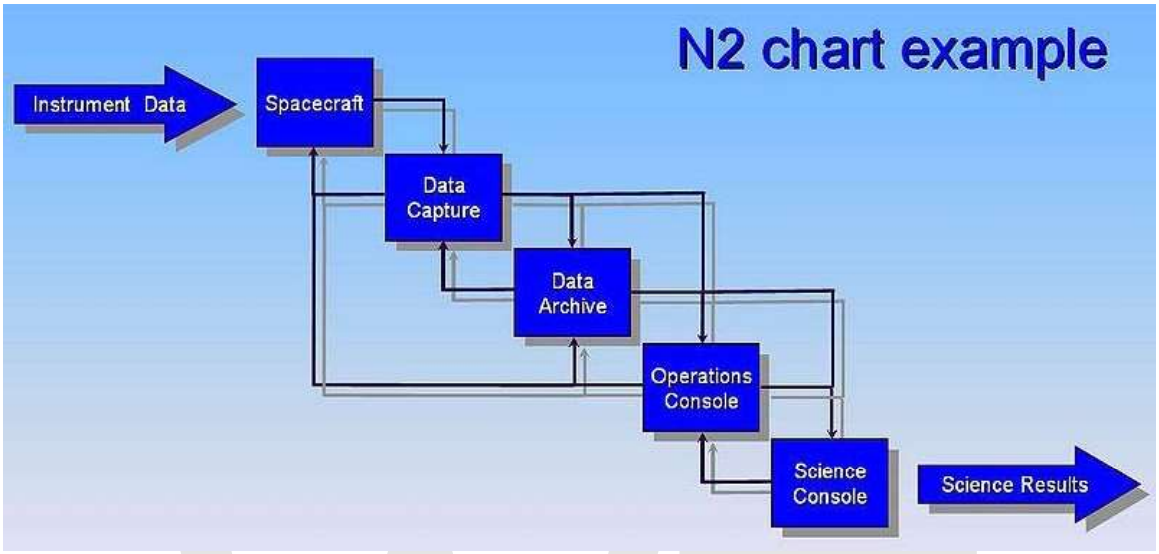
Level 2 (low level diagram)



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.

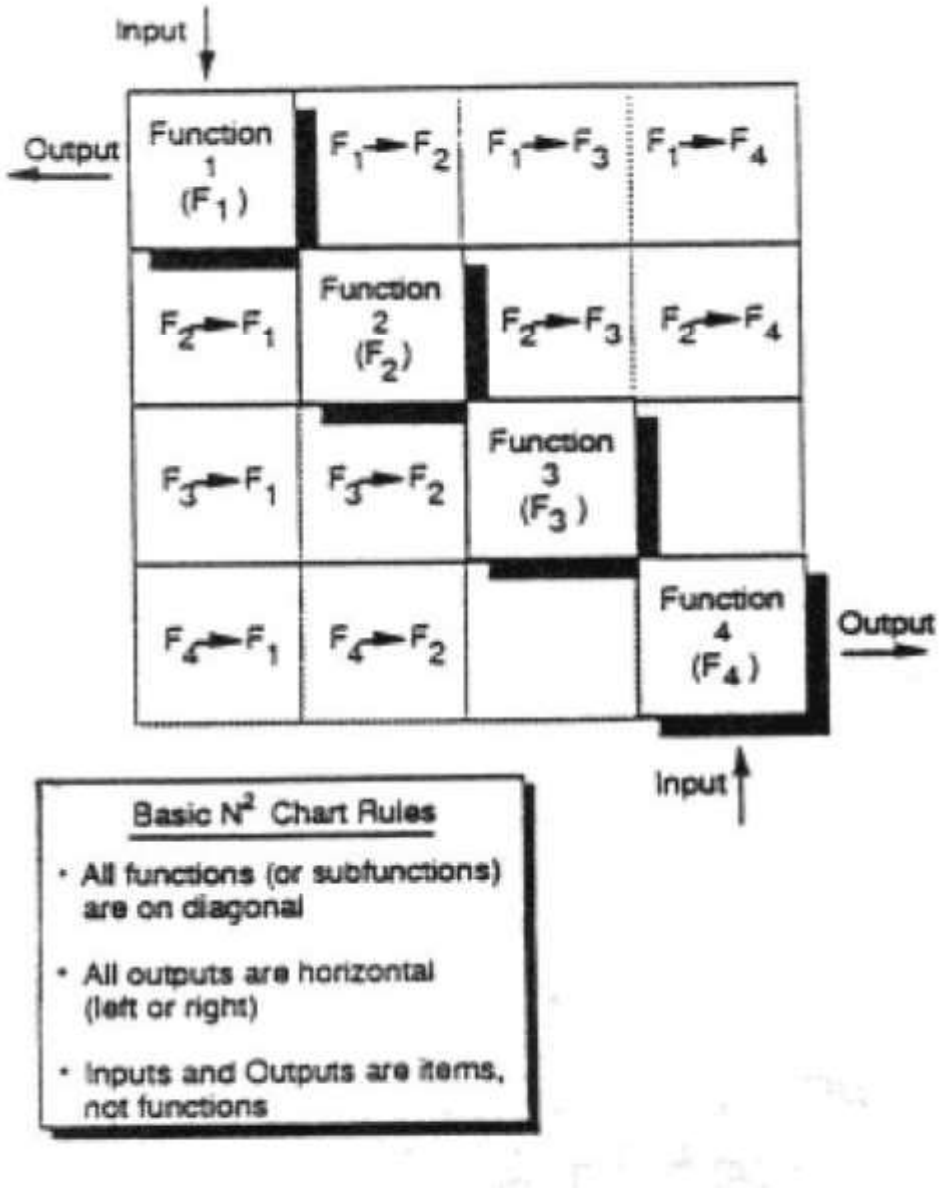


Figure 2. N2 chart definition

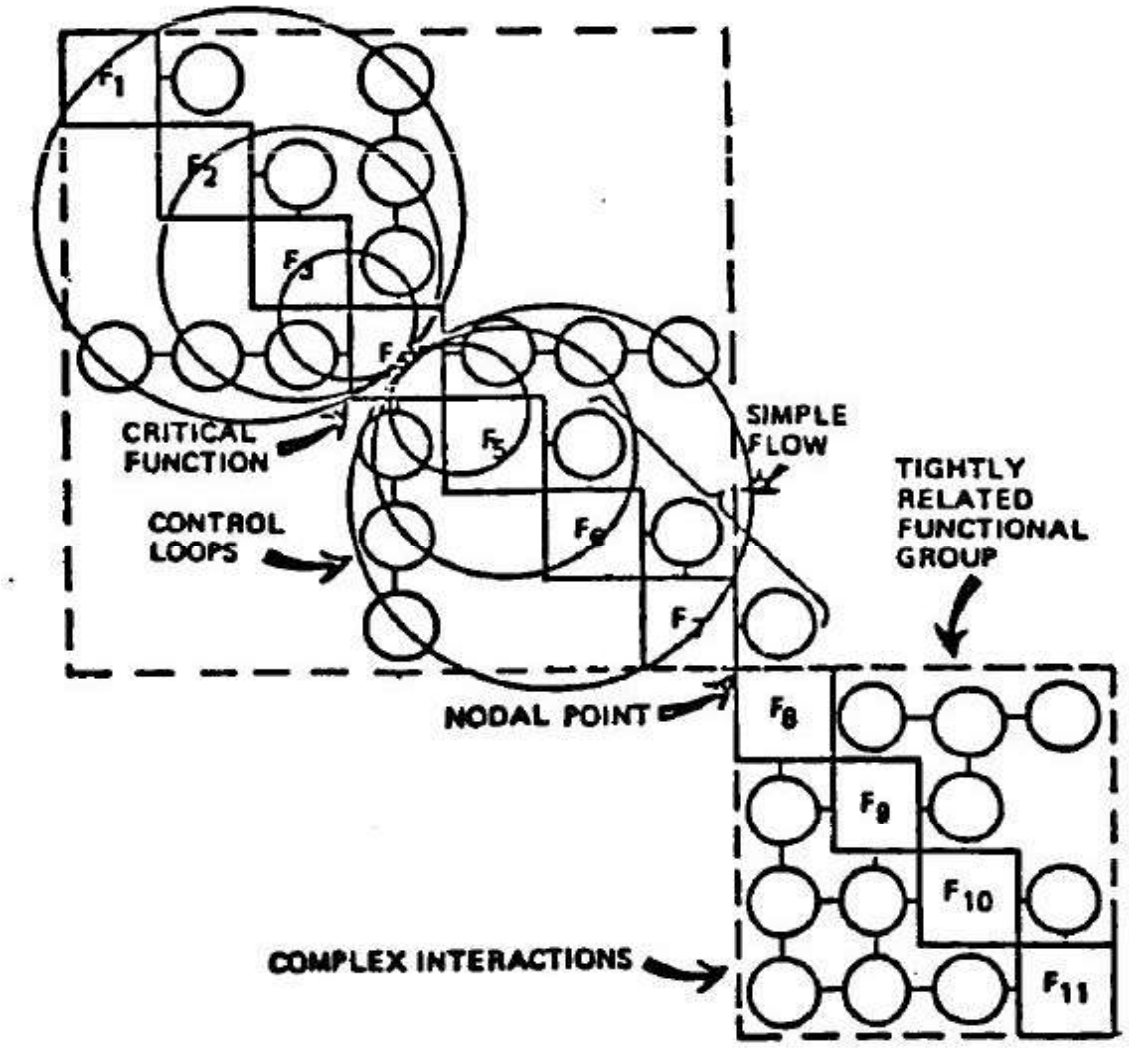


Figure 3. N2 Chart Key Features

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

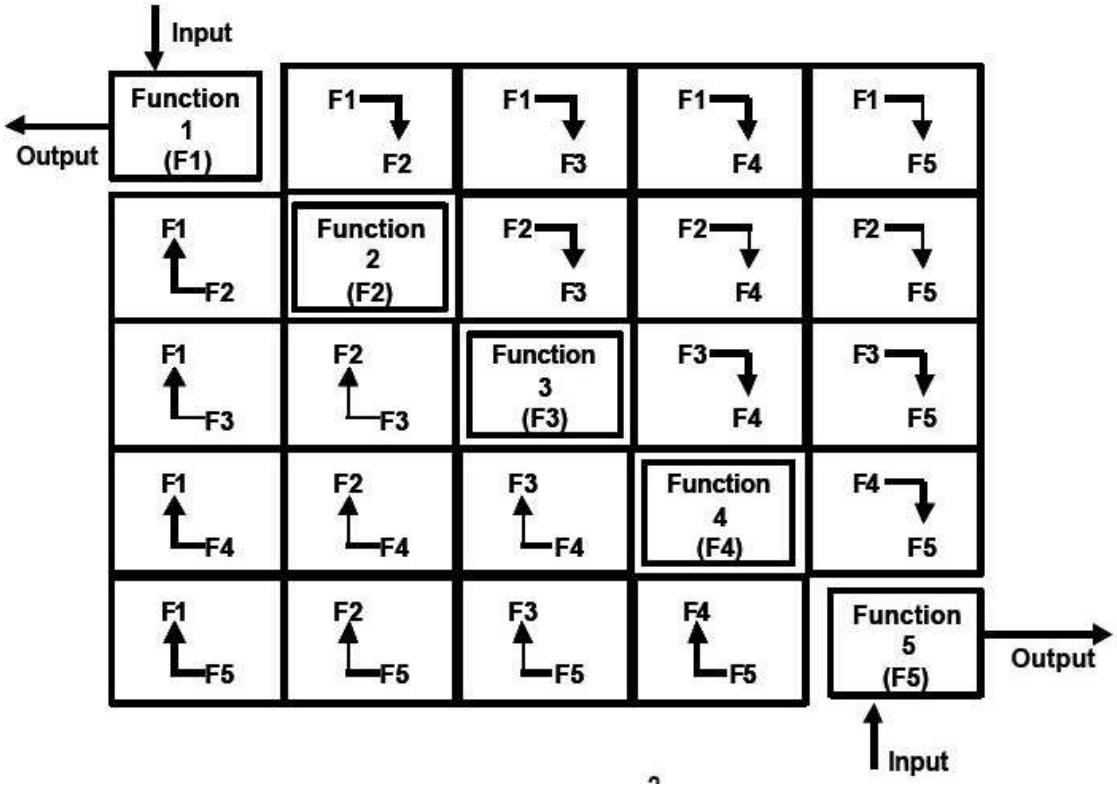


Figure 4. N^2 diagram

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.

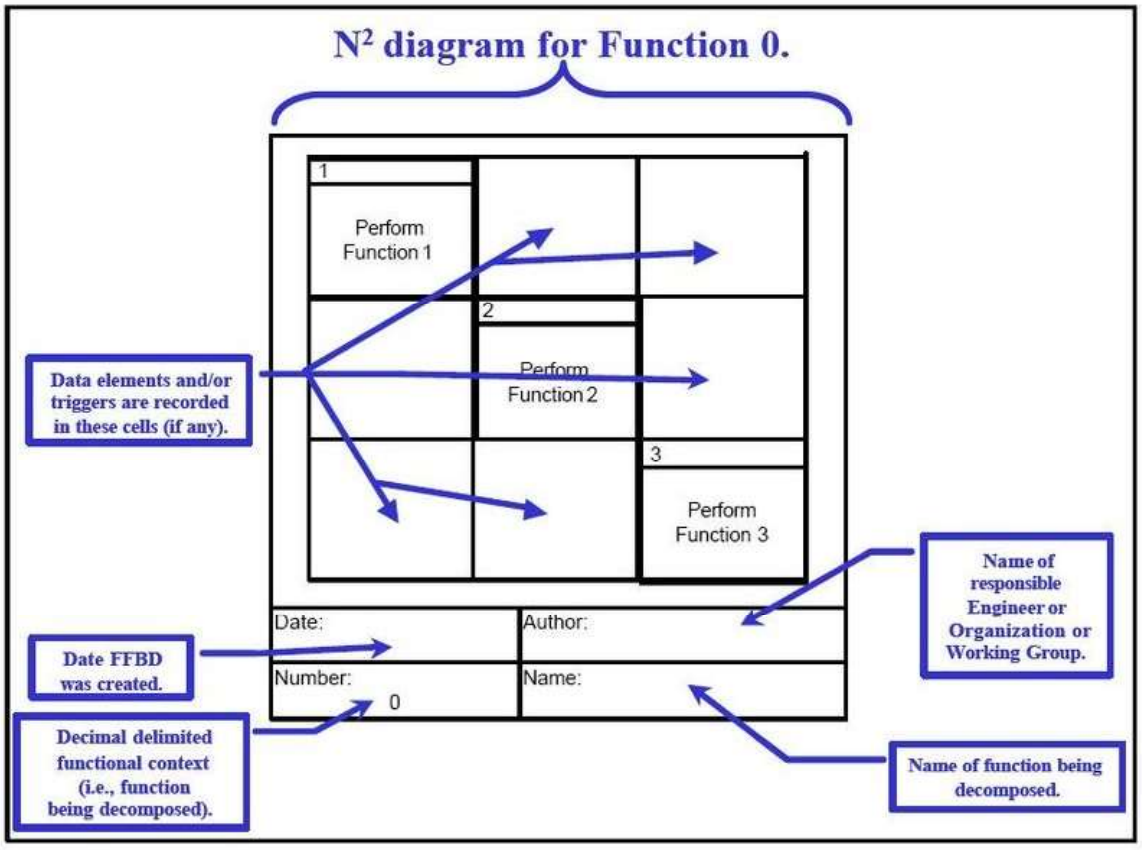
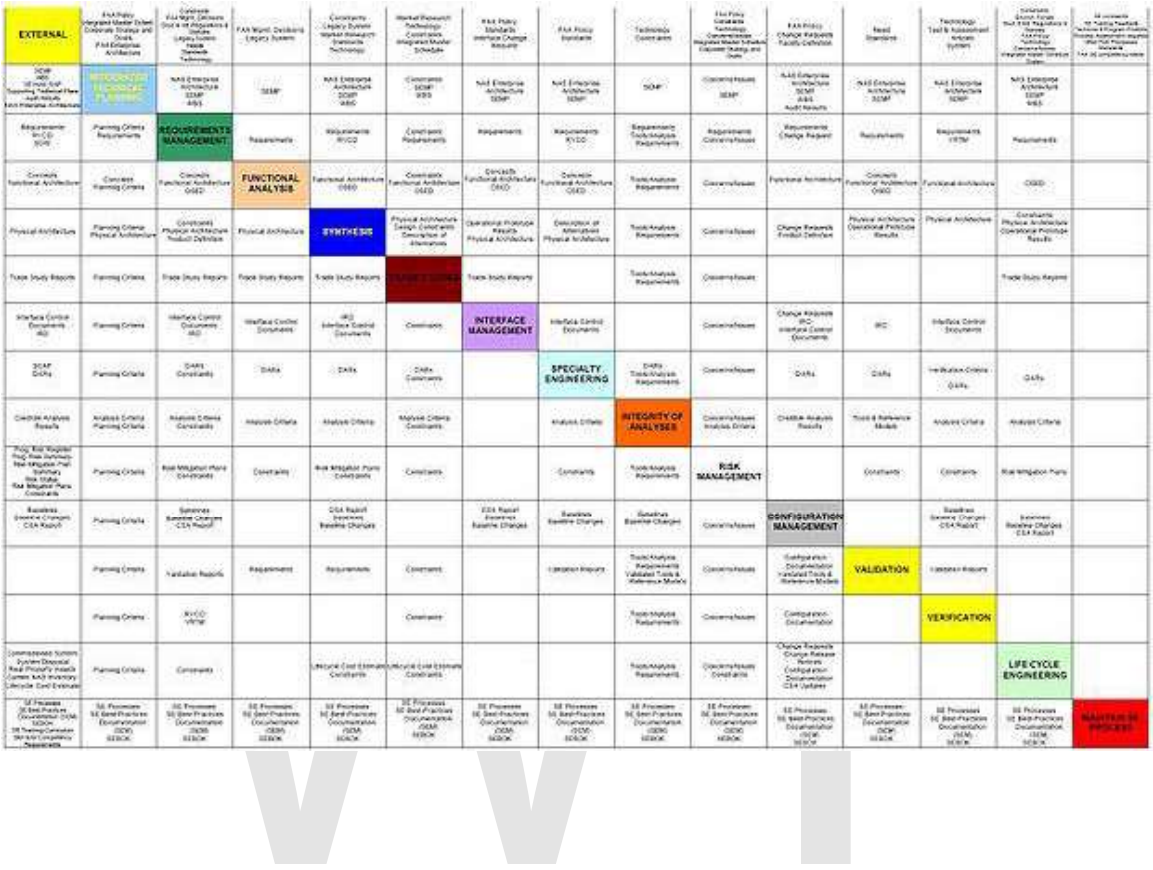


Figure 5 presents information in an N2 diagram, which complements the Functional flow block diagram. Notice that in this illustration, there are no data elements or triggers. The figure illustrates the context between functions at different levels of the model.

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.

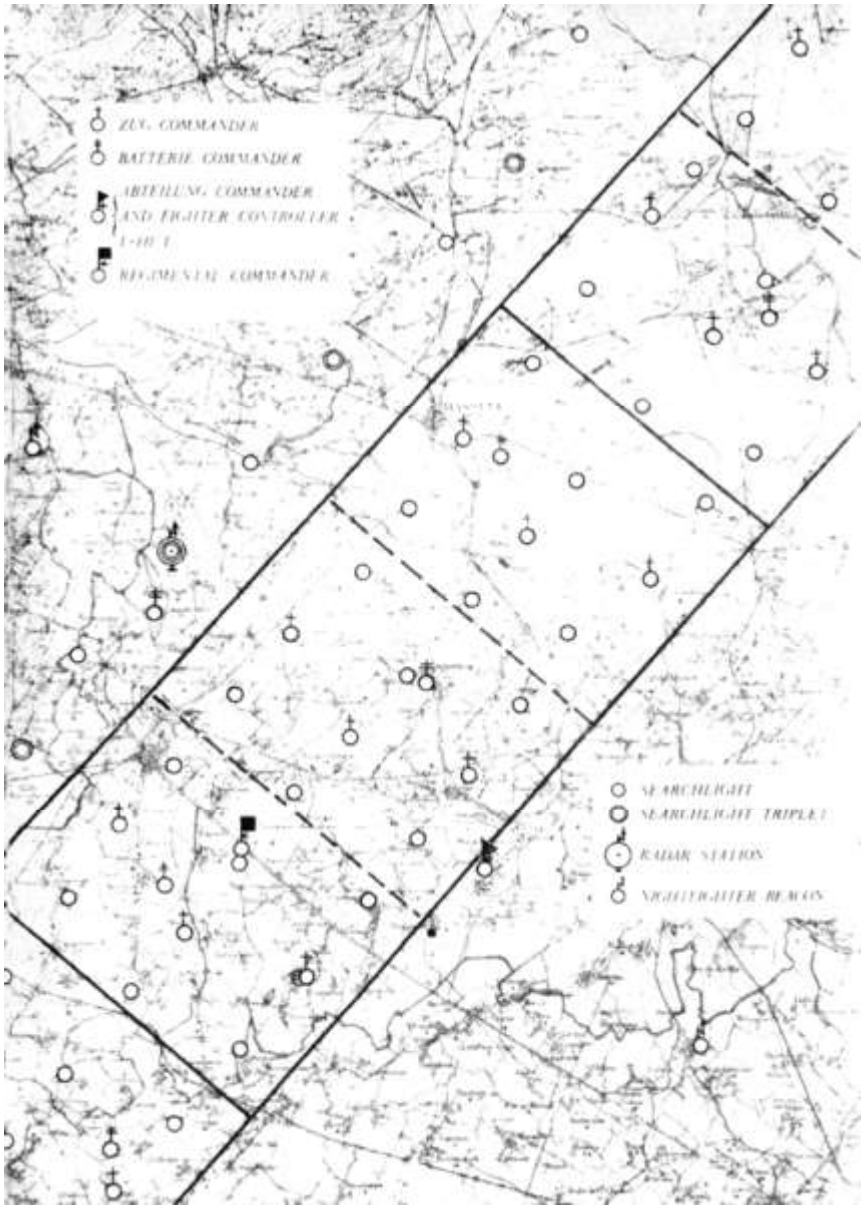
In 1941 Blackett moved from the RAE to the Navy, first to the Royal Navy's Coastal Command, in 1941 and then early in 1942 to the Admiralty. Blackett's team at Coastal Command's Operational Research Section (CC-ORS) included two future Nobel prize winners and many other people who went on to be preeminent 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-α) 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

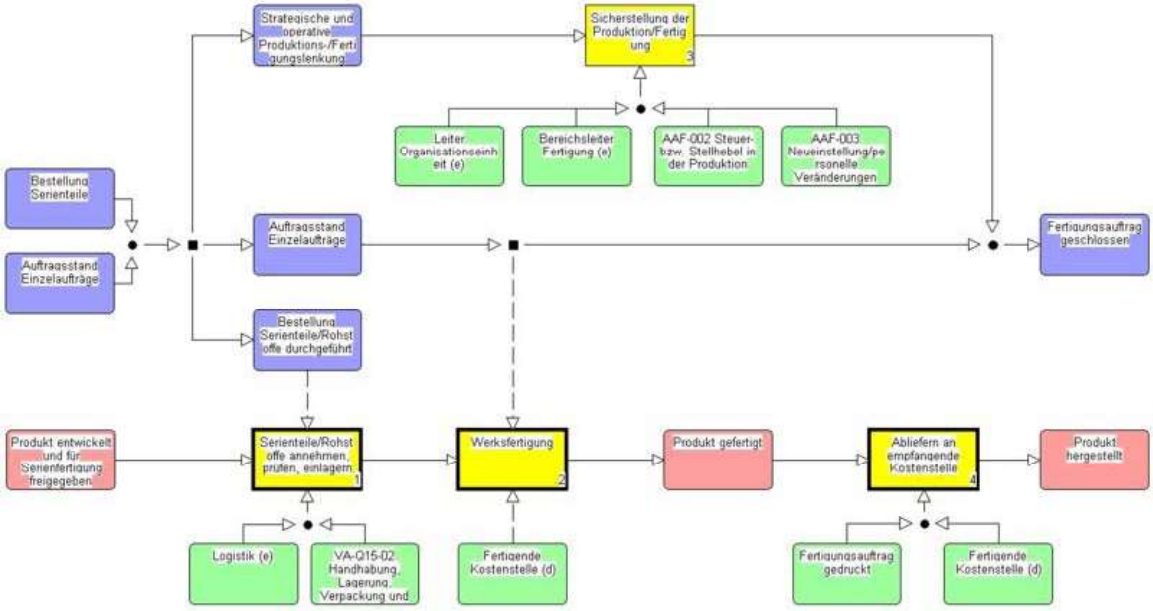
To ensure that there is proper feedback validating that the system meets the NFR specified performance metrics, any major system needs a monitoring subsystem. The planning, design, installation, configuration, and control of the monitoring subsystem is specified by an appropriately defined Monitoring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

Integrated Enterprise Modeling



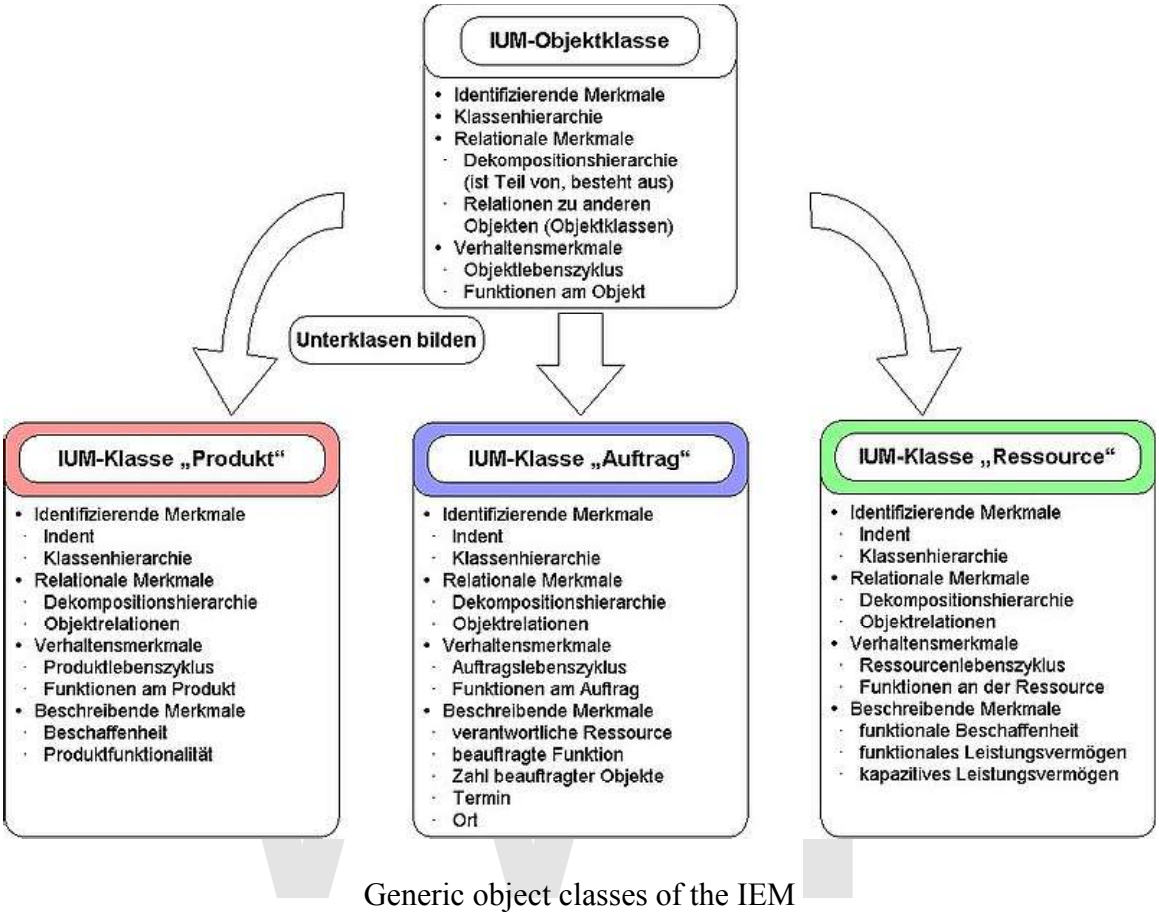
Model example

The **Integrated Enterprise Modeling** (IEM) is an enterprise modeling method used for the admission and for the reengineering of processes both in producing enterprises and in the public area and service providers. In the Integrated Enterprise Modeling different aspects as functions and data become described in one model. Furthermore the method supports analyses of business processes independently of the available organizational structure.

The Integrated Enterprise Modeling is developed at the *Fraunhofer Institute for Production Systems and Design Technology (German: IPK) Berlin, Germany*.

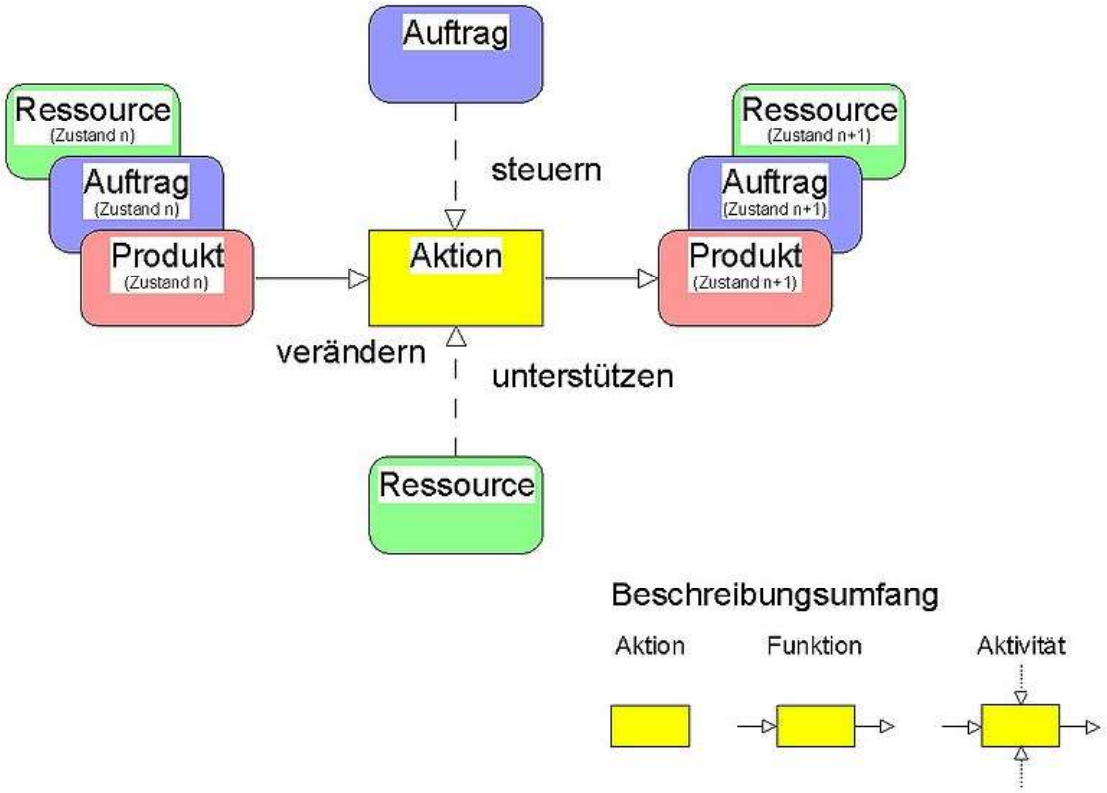
Integrated Enterprise Modeling Topics

Base constructs



Generic object classes of the IEM

Overview of the object classes product, order and resource



Generic activity model

The Integrated Enterprise Modeling (IEM) method uses an object-oriented approach and adapts this for the enterprise description. An application-oriented division of all elements of an enterprise forms the core of the method in generic object classes *"product"*, *"resource"* and *"order"*.

Product

The object class "product" represents all objects whose production and sale are the aim of the looked-at-enterprise as well as all objects which flow into the end product. Raw materials, intermediate products, components and end products as well as services and the describing data are included.

Order

The object class "order" describes all types of a commissioning in the enterprise. The objects of the class "order" represent the information which is relevant from the point of view of planning, control and supervision of the enterprise processes. One understands by it what, when, at which objects, in whose responsibility and with which resources it will be executed.

Resource

The IEM class "resource" contains all necessary key players which are required in the enterprise for the execution or support of activities. Among other things these are employees, business partner, all kinds of documents as well as information systems or operating supplies.

The classes "product", "order", and "resource" can gradually be given full particulars and specified. Through this it is possible to show both line of business typical and enterprise specific product, order and resource subclasses. Structures (e.g. parts lists or organisation charts) can be shown as relational features of the classes with the help of being-part-of- and consists-of-relations between different subclasses.

Action

The activities which are necessary for the production of products and to the provision of services can be described as follows: an activity is the purposeful change of objects. The aim orientation of the activities causes an explicit or implicit planning and control. The execution of the activities is incumbent by the capable key players. From these considerations the definitions can be derived for the following constructs:

- An *action* is an object neutral description of activities: a verbal description of a work task, a lawsuit or proceeding;
- A *function* describes the change of state of a defined status into another defined one of objects of a class by using an action;
- An *activity* specifies necessary resources for the state transformation of objects of a class the controlling order described by a function and these for the execution of this transformation in the enterprise, in each case represented by an object state description.

Views

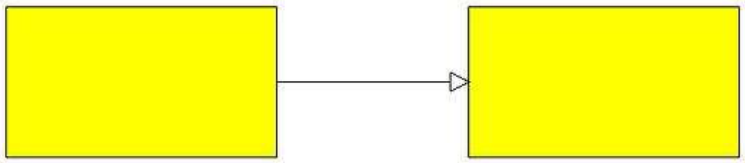
All modeled data of the looked-at-enterprise are recorded in the model core of an Integrated Enterprise Modeling (IEM) model in two main views:

- the "information model" and
- the "business process model".

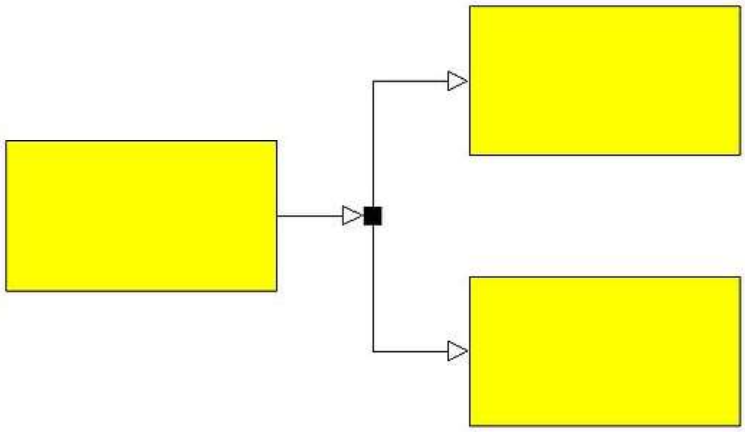
All relevant objects of an enterprise, their qualities and relations are shown in the "information model". It is class trees of the object classes "product", "order" and "resource" here. The "business process model" represents enterprise processes and their relations to each other. Activities are shown in their interaction with the objects.

Process modeling

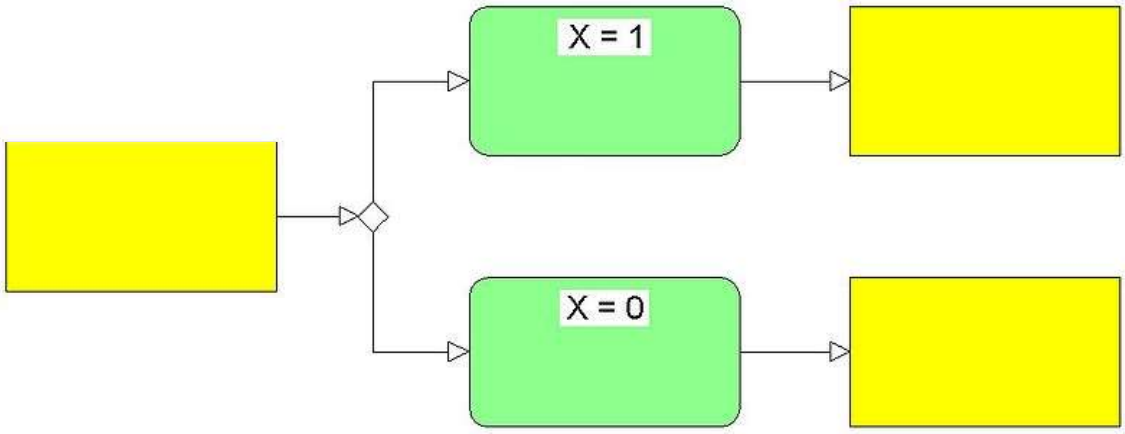
The structuring of the enterprise processes in Integrated Enterprise Modeling (IEM) is reached by its hierarchical subdivision with the help of the decomposition. Decomposition means the reduction of a system in a partial system which respectively contains components which are in a logical cohesion. The process modeling is a partitioning of processes into its threads. Every thread describes a task completed into itself. The decomposition of single processes can be carried out long enough until the threads are manageable, i.e. appropriately small. They may turn out also not too rudimentary because a high number of detailed processes increases the complexity of a business process model. A process modeling person therefore have to find a balance between the effort complexity degree of the model and possible detailed description of the enterprise processes. A model depth generally recommends itself with at most three to four decomposition levels (model levels).



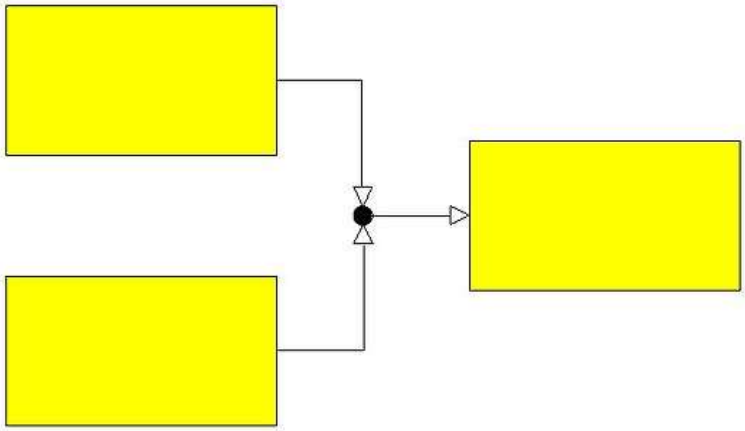
Sequential order



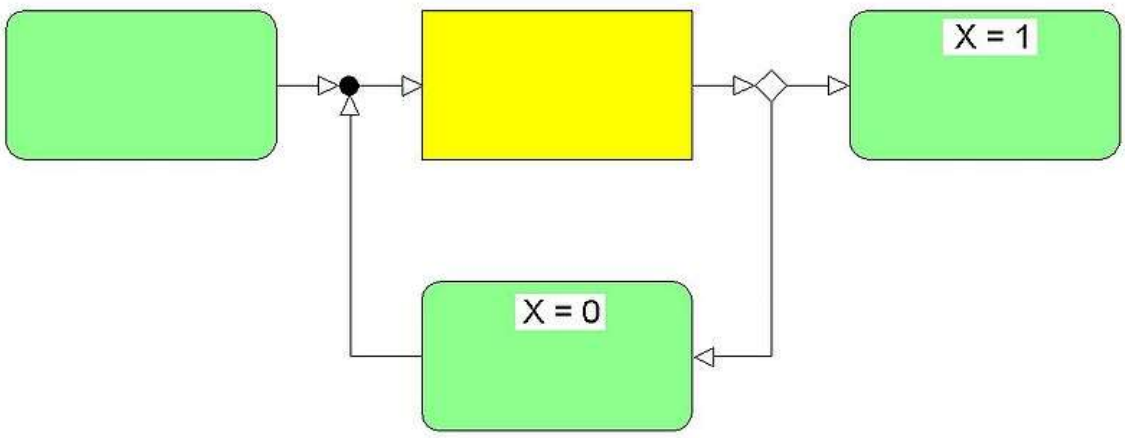
Parallel branching



Case distinction



Uniting



Loop

On a model level business process flows are represented with the aid of illustrated combination elements. There are these five basic types of combinations between the activities:

- Sequential orde: At a sequential order the activities are executed after each other.
- Parallel branching: A parallel branching means that all parallel branched activities to be executed have to be completed before the following activity can be started with. It is not necessary that the parallel activities are executed at the same time. They can be deferredly carried out, too.
- Case distinction: Decision either or. The case distinction is a branching in alternative processes depending on definition of the subsequent conditions.
- Uniting: The end of a parallel as the case may be alternative execution or also an integration of process chains is indicated by the uniting.
- Loop: A repatriation (loop, cycle) is represented by means of case distinction and uniting. The activities included in the loop are executed as long as the condition for the continuation is given.

Modeling proceeding

The modeling procedure for the illustration of business processes in IEM covers the following steps:

- System delimitation,
- Modeling,
- Model evaluation and use,
- Model change.

The *system delimitation* is the base of an efficient modeling. Starting out from a conceptual formulation the area of the real system to be shown is selected and interfaces will be defined to an environment. In addition, the detail depth of the model is also

determined, i.e. the depth of the hierarchical decomposition relations in the view "business process model".

The delimited real system is convicted with help of the IEM method in an abstract model. IEM is the construction of the two main positions "information model" and "business process model". The "information model" is made by the specification of the object classes to be modeled for "product", "order" and "resource" with the class structures as well as descriptive and relational features. By identification and description of functions, activities and its combination to processes the "business process model" is formed. As a general rule the construction of the "information model" follows first in which the modeling person can go back to available reference class structures. The reference classes which do not correspond to the real system or were not found to be relevant at the system delimitation are deleted. The missing relevant classes are inserted. After the object base is fixed, the activities and functions are joined together at the objects according to the "generic activity model" and with the help of combination elements to business processes. A model is made which can be analysed and changed if it's required. It often happens, that during the construction of the "business process model" new relevant object classes are identified so that the class trees getting completed. The construction of the two positions is therefore an iterative process.

Afterwards weak points and improvement potentials can be identified in the course of the *model evaluation*. This can cause the *model changes* whose realization should clear the weak points and make use of the improvement potentials in the real system.

Modeling tool MO²GO

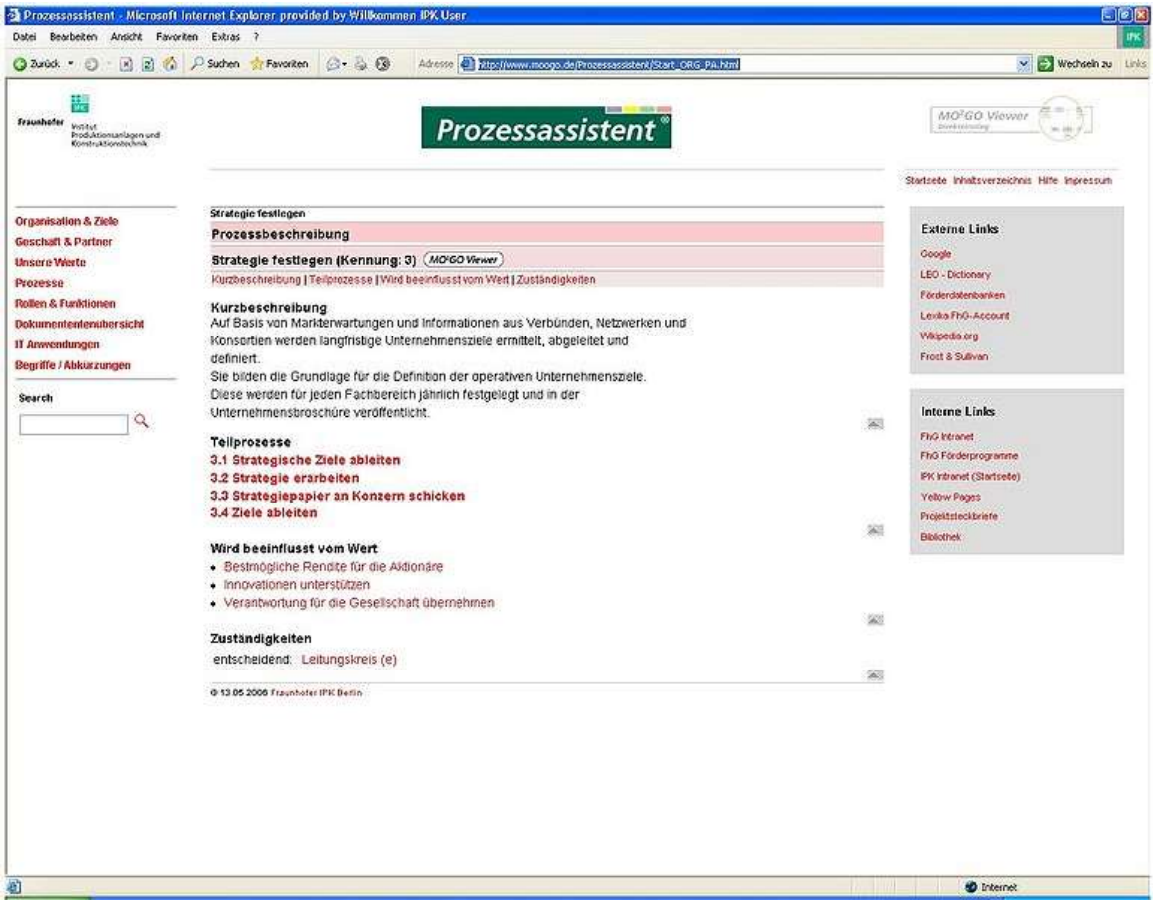
The software tool MO²GO (method for an object-oriented business process optimization) supports the modeling process based on the integrated enterprise modeling (IEM). Different analyses of a given model are available like the planning and implementation of information systems. The MO²GO system is expandable easily and makes a high-speed modeling approach possible.

The currently used MO²GO system consists of the following components:

- MO²GO version 2.4: This component offers modeling functions for class structures, process chains and mechanism for analysis of IEM.
- MO²GO Macro editor version 2.1 : The macro editor supports the outline of MO²GO macros for user-defined evaluation procedures.
- MO²GO Viewer version 1.07 : The Java based and licence free MO²GO Viewer is a user interface to be used easily to navigate process chains through MO²GO.
- MO²GO XML converter version 1.0 : Nowadays the IT implementation works mainly with UML diagrams. MO²GO supports a component for a model based XML file which can be imported in UML tools.
- MO²GO Web publisher version 2.0 : The web Publisher is a mechanism of analysis to be started directly out of MO²GO 2.4. A process assistant is the result of the evaluation of the model contents based on texture and hyperlink

representation. To be able to adapt the process assistant to the user requirements flexibly, the web Publisher contains a configuration component.

MO²GO Process assistant



Prozess assistant

The IEM business process models contain much information which can not only be used by system analysts but also be helpful for the employees at their daily work. To provide this model information for the staff and to enable the participation of the employees for the results of the modeling, a special tool was developed at the Fraunhofer IPK. This is a web based process assistant whose contents are generated automatically from the IEM business process model of the enterprise. The process assistant provides all users the information of the business process model in a HTML based form by intranet of the enterprise. For its implementation no special methods or tool knowledge is required besides the basic EDP and Internet experiences.

The process assistant has been developed so that the employees can find answers to the questions fast and precisely: e.g.

- What are the processes in the enterprise?
- In which way are they structured as?

- Who and with which responsibility is involved in the certain process?
- Which documents and application systems are used?

Or also:

- A certain organisation unit is involved at which processes?
- Or in which processes a certain document or an application system is used?

To make an informative process assistant from the business process model, certain modeling rules must be followed. The means e.g. that the individual actions must be deposited with its descriptions, the responsibility of the organisation units must be indicated explicitly or the paths also must be entered to the documents in the class tree. The fulfilment of these conditions means an additional time expenditure at the modeling, if these conditions are met, all employees are able to *"surf"* online through the intranet with the help of the process assistant by an informative enterprise documentation. They have the possibility between a graphic view and a texture based description according to their preferences and methodical previous knowledge. The graphic view is provided by the MO²GO Viewer, a viewer tool for MO²GO models. The process assistant and the MO²GO Viewer are connected so that the graphic representation of the process looked at can be accessed context sensitively from the process assistant.

Users can call on all templates, specifications and documents for the working sequence both from the process assistant and from the MO²GO Viewer online. Therefore the process assistant cannot only be employed for the tracing of the modeling results but also in the daily business for the training of new employees as well as execution of process steps. To improve the usability in the daily routine, the process assistant can be adapted to the needs of the users flexibility. This customization can be carried out both concerning the layout and concerning the main content emphases of the process assistant.

Areas of application of the IEM

Knowledge is used in organisations as a resource to render services for customers. The service preparation performs along actions which are described as processes or business processes. The analysis and improvement in dealing with knowledge presupposes a common idea about this context. An explicit description of the processes therefore is required because they represent the context for the respective knowledge contents. The process modeling represents a powerful instrument for the design and a conversion of a process-oriented knowledge management. In the context of the method of the business process-oriented knowledge management (GPO KM) developed at the Fraunhofer IPK the method of the "integrated enterprise modeling" (IEM) is accessed. It makes it possible to be able to show, to describe, to analyse and to form organisational processes. The IEM features few object classes, is ascertainable easily understandable and fast. Furthermore the object orientation of the IEM opens up the possibility of showing knowledge as an object class. For the knowledge-oriented modeling of the business processes according to the IEM method the relevant knowledge contents have to be specified after knowledge

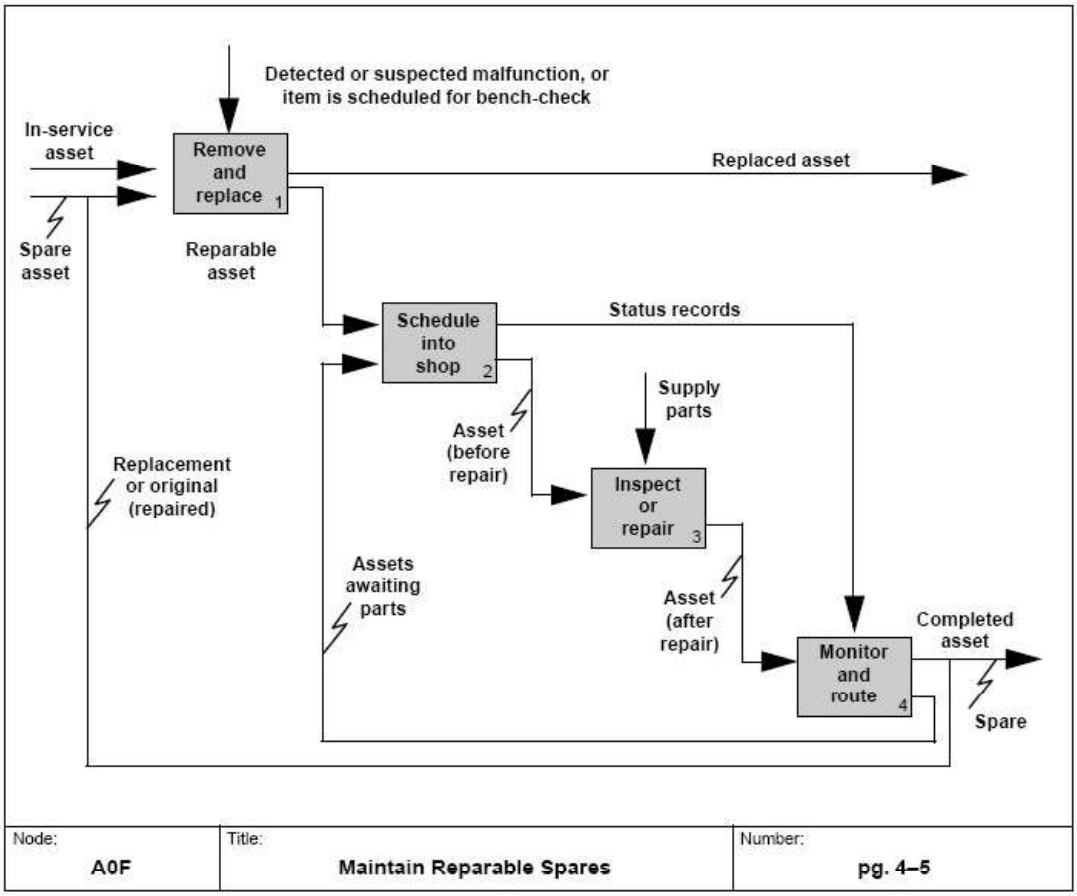
domains and know-how bearers and represented as resources in the business process model.

In further applications IEM is used to create models across organisations (e.g. companies)to archive a common understanding between the involved stakeholders and derive services (create software and define the ASP). In this context the object oriented basis of IEM has been used to create a common semantic across the single company models and to archive compliant enterprise models (predefined classes - terminology, model templates, etc.). The reason is that the terminology used within a model has to be understandable independent of the modeling language.

Chapter- 11

Function Model

A **function model** or *functional model* in systems engineering and software engineering is a structured representation of the functions (activities, actions, processes, operations) within the modeled system or subject area.



Example of a function model of the process of "Maintain Reparable Spares" in IDEF0 notation.

A function model, also called an activity model or process model, is a graphical representation of an enterprise's function within a defined scope. The purposes of the function model are to describe the functions and processes, assist with discovery of

information needs, help identify opportunities, and establish a basis for determining product and service costs.

History

The function model originates in the 1950s, after in the first half of the 20th century other types of management diagrams had already been developed. The first known Gantt Chart was developed in 1896 by Karol Adamiecki, who called it a *harmonogram*. Because Adamiecki did not publish his chart until 1931 - and in any case his works were published in either Polish or Russian, languages not popular in the West - the chart now bears the name of Henry Gantt (1861–1919), who designed his chart around the years 1910-1915 and popularized it in the West. 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 (ASMR) 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.

One of the first well defined function models, was the Functional Flow Block Diagram (FFBD) developed by the defense-related TRW Incorporated in the 1950s. In the 1960s it was exploited by the NASA to visualize the time sequence of events in a space systems and flight missions. It is further widely used in classical systems engineering to show the order of execution of system functions.

Functional modeling topics

Functional perspective

In systems engineering and software engineering a function model is created with a functional modeling perspective. The functional perspective is one of the perspectives possible in business process modelling, other perspecifives are for example behavioural, organisational or informational.

A functional modeling perspective concentrates on describing the dynamic process. The main concept in this modeling perspective is the process, this could be a function, transformation, activity, action, task etc. A well-known example of a modeling language employing this perspective is data flow diagrams.

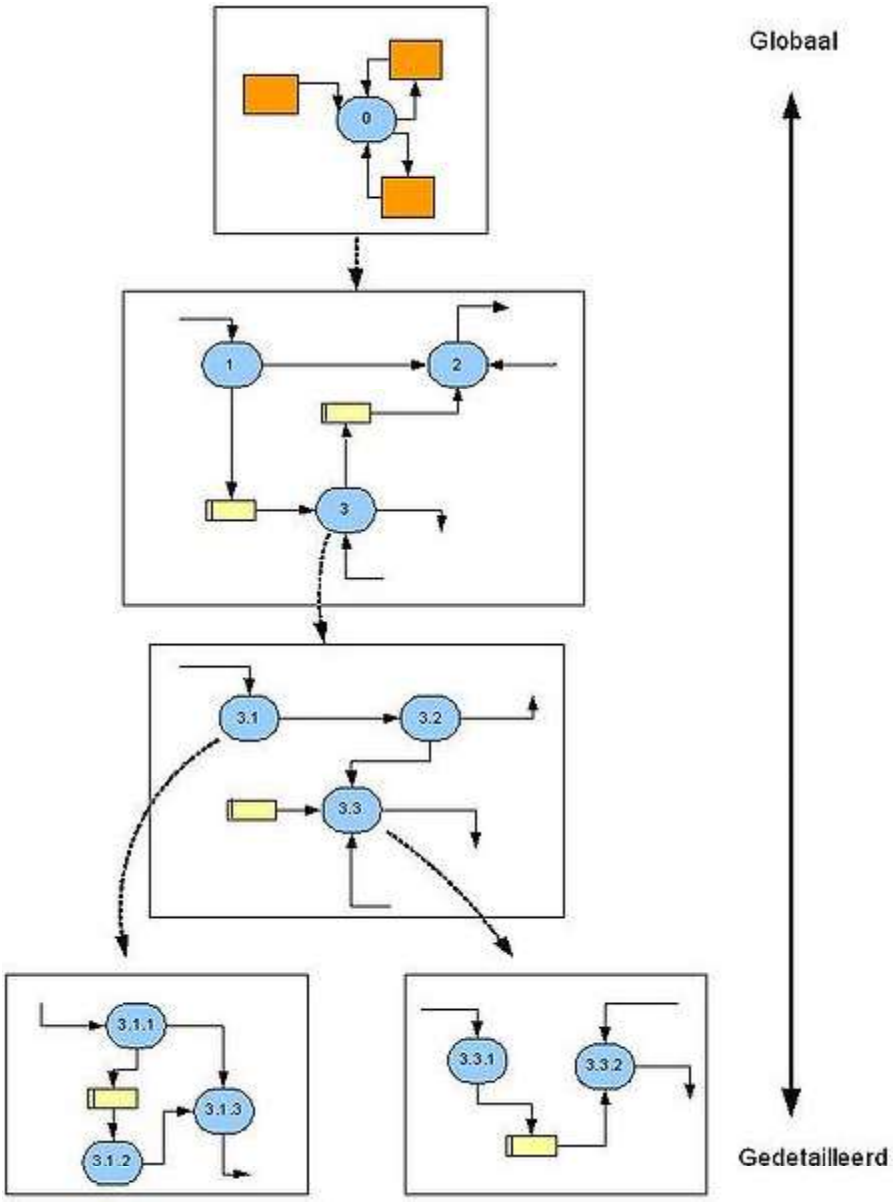
The perspective uses four symbols to describe a process, these being:

- Process: Illustrates transformation from input to output.
- Store: Data-collection or some sort of material.
- Flow: Movement of data or material in the process.
- External Entity: External to the modeled system, but interacts with it.

Now, with these symbols, a process can be represented as a network of these symbols. This decomposed process is a DFD, data flow diagram.

In Dynamic Enterprise Modeling a division is made in the Control model, Function Model, Process model and Organizational model.

Functional decomposition



Example of functional decomposition in a systems analysis.

Functional decomposition refers broadly to the process of resolving a functional relationship into its constituent parts in such a way that the original function can be reconstructed from those parts by function composition. In general, this process of decomposition is undertaken either for the purpose of gaining insight into the identity of the constituent components, or for the purpose of obtaining a compressed representation of the global function, a task which is feasible only when the constituent processes possess a certain level of *modularity*.

Functional decomposition has a prominent role in computer programming, where a major goal is to *modularize* processes to the greatest extent possible. For example, a library management system may be broken up into an inventory module, a patron information module, and a fee assessment module. In the early decades of computer programming, this was manifested as the "art of subroutining," as it was called by some prominent practitioners.

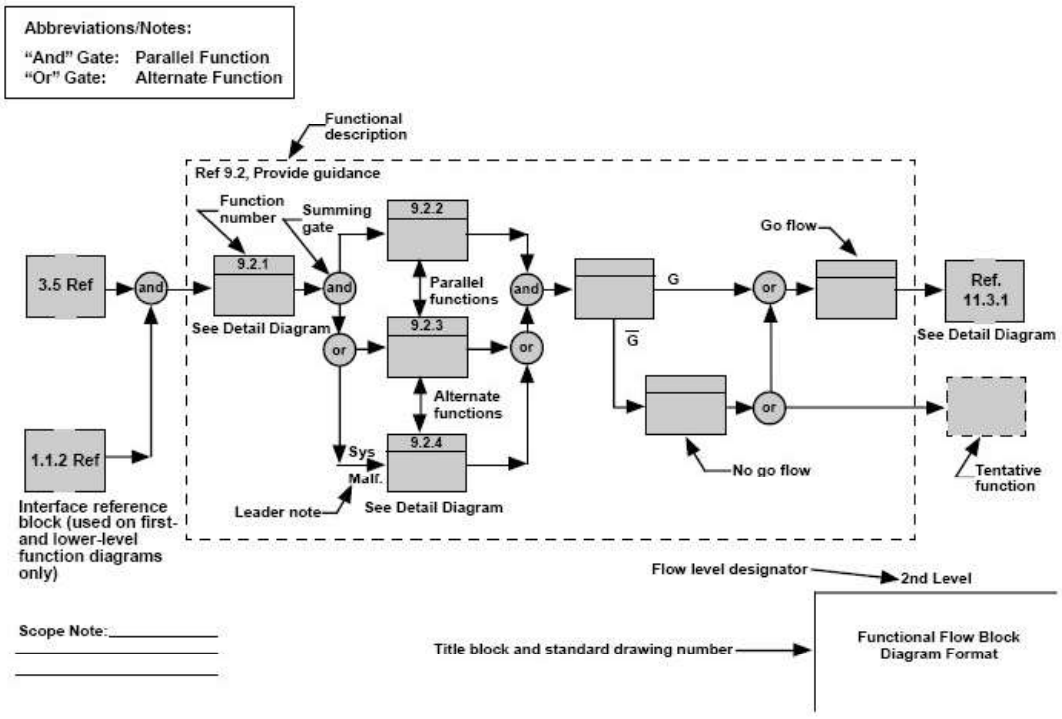
Functional decomposition of engineering systems is a method for analyzing engineered systems. The basic idea is to try to divide a system in such a way that each block of the block diagram can be described without an "and" or "or" in the description.

This exercise forces each part of the system to have a pure function. When a system is composed of pure functions, they can be reused, or replaced. A usual side effect is that the interfaces between blocks become simple and generic. Since the interfaces usually become simple, it is easier to replace a pure function with a related, similar function.

Functional modeling methods

The functional approach is extended in multiple diagrammic techniques and modeling notations.

Functional Flow Block Diagram

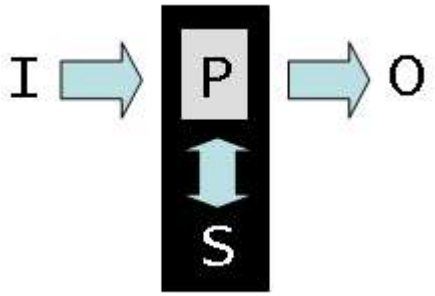


Functional Flow Block Diagram Format.

The Functional flow block diagram (FFBD) is a multi-tier, time-sequenced, step-by-step flow diagram of the system's functional flow. The diagram is developed in the 1950s and widely used in classical systems engineering. The Functional Flow Block Diagram is also referred to as *Functional Flow Diagram*, *functional block diagram*, and *functional flow*.

Functional Flow Block Diagrams (FFBD) usually define the detailed, step-by-step operational and support sequences for systems, but they are also used effectively to define processes in developing and producing systems. The software development processes also use FFBDs extensively. In the system context, the functional flow steps may include combinations of hardware, software, personnel, facilities, and/or procedures. In the FFBD method, the functions are organized and depicted by their logical order of execution. Each function is shown with respect to its logical relationship to the execution and completion of other functions. A node labeled with the function name depicts each function. Arrows from left to right show the order of execution of the functions. Logic symbols represent sequential or parallel execution of functions.

HIPO and IPO



An expanded IPO Model.

HIPO for *hierarchical input process output* is a popular 1970s systems analysis design aid and documentation technique for representing the modules of a system as a hierarchy and for documenting each module.

It was used to develop requirements, construct the design, and support implementation of an expert system to demonstrate automated rendezvous. Verification was then conducted systematically because of the method of design and implementation.

The overall design of the system is documented using HIPO charts or structure charts. The structure chart is similar in appearance to an organizational chart, but has been modified to show additional detail. Structure charts can be usedto display several types of information, but are used most commonly to diagram either data structures or code structures.

N2 Chart

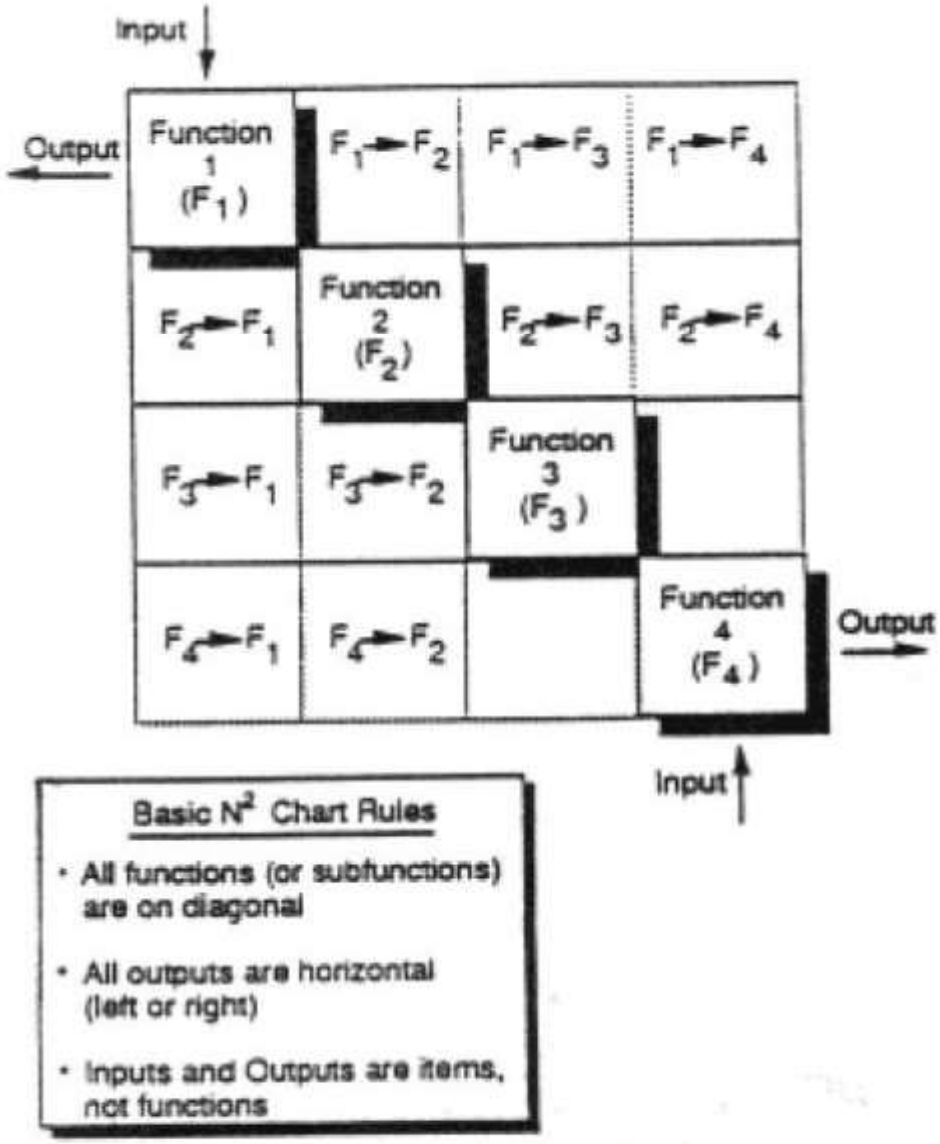
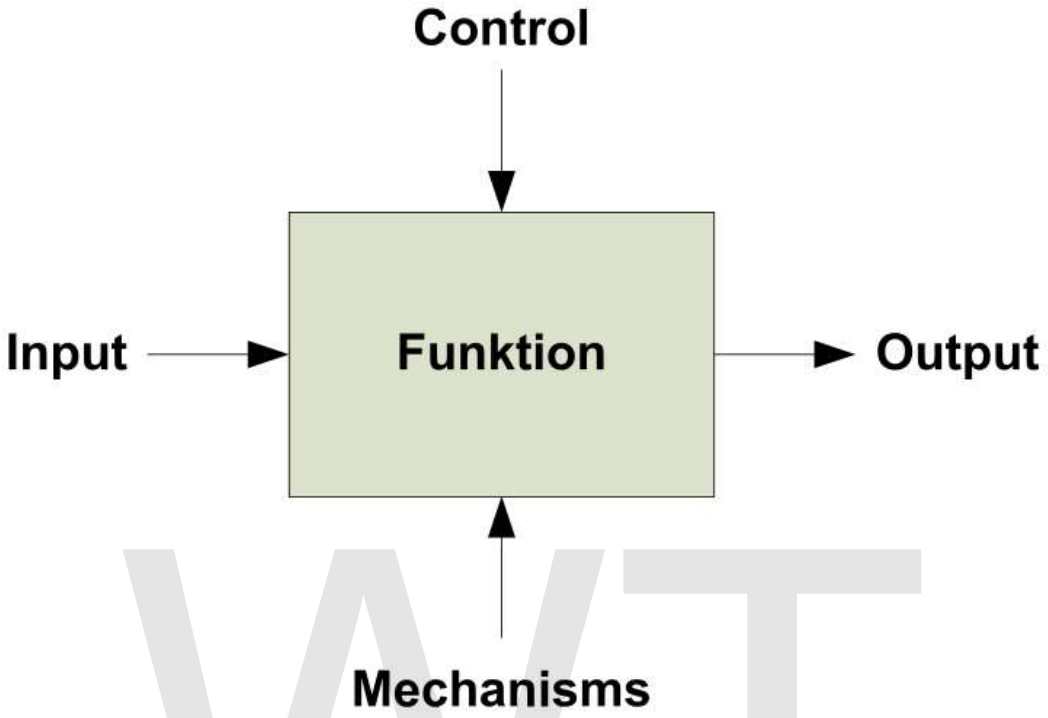


Figure 2. N2 chart definition.

The N2 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 N2 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 N2 chart is shown in Figure 2. The system functions are placed on the diagonal; the remainder of the squares in the N x N matrix represent the interface inputs and outputs.

Structured Analysis and Design Technique

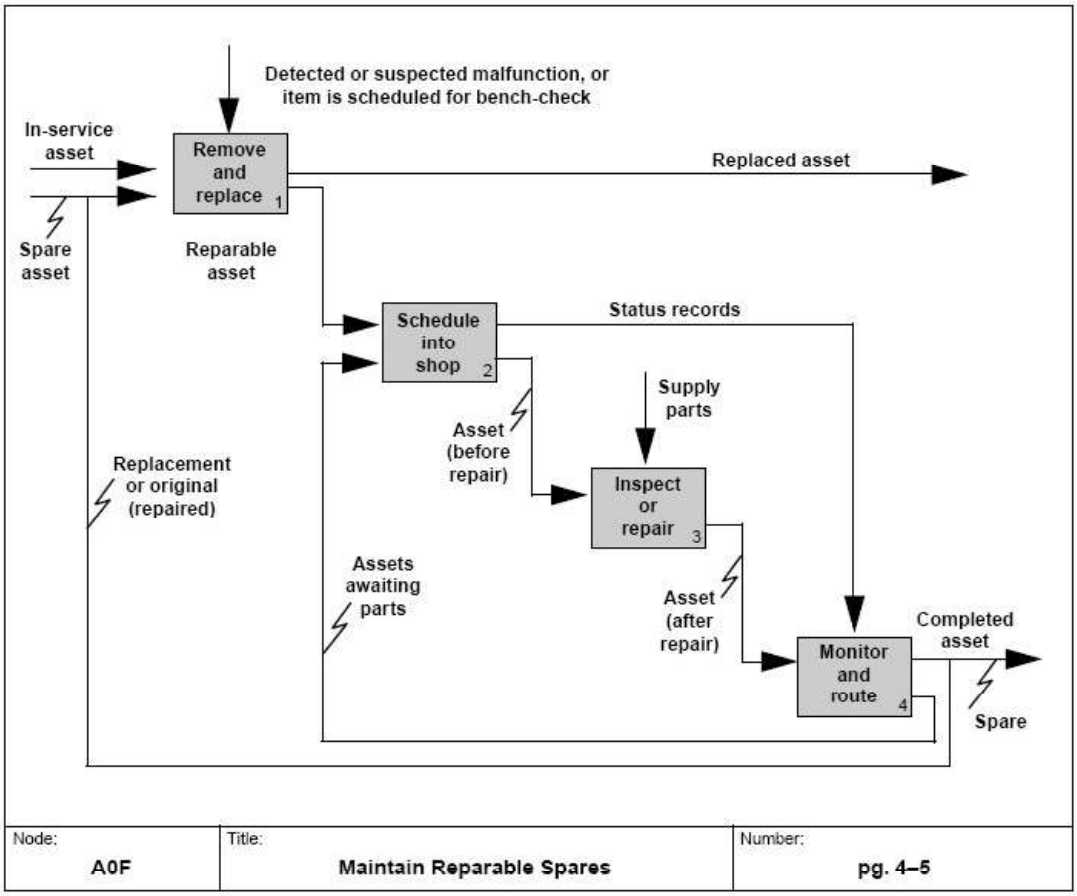


SADT basis element.

Structured Analysis and Design Technique (SADT) is a software engineering methodology for describing systems as a hierarchy of functions, a diagrammatic notation for constructing a sketch for a software application. It offers building blocks to represent entities and activities, and a variety of arrows to relate boxes. These boxes and arrows have an associated informal semantics. SADT can be used as a functional analysis tool of a given process, using successive levels of details. The SADT method allows to define user needs for IT developments, which is used in industrial Information Systems, but also to explain and to present an activity's manufacturing processes, procedures.

The SADT supplies a specific functional view of any enterprise by describing the functions and their relationships in a company. These functions fulfill the objectives of a company, such as sales, order planning, product design, part manufacturing, and human resource management. The SADT can depict simple functional relationships and can reflect data and control flow relationships between different functions. The IDEF0 formalism is based on SADT, developed by Douglas T. Ross in 1985.

IDEF0

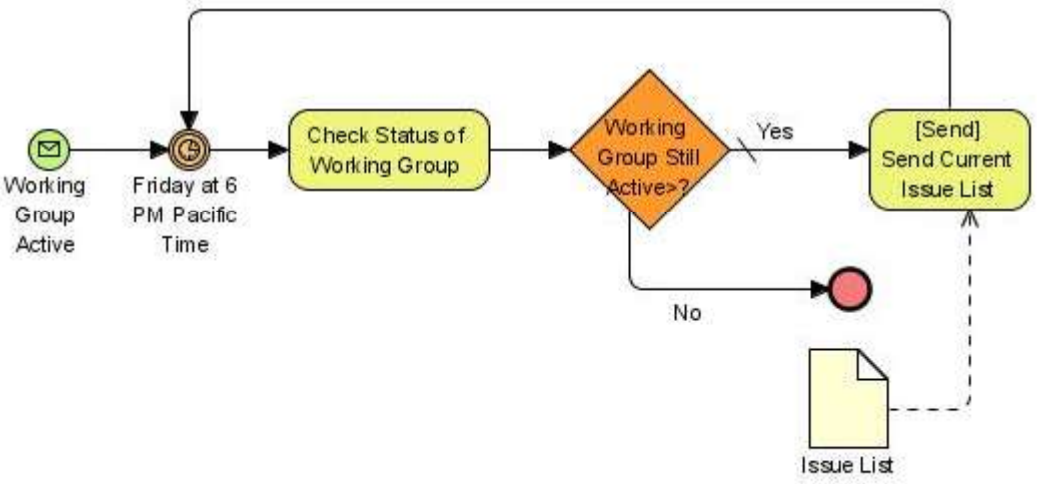


IDEF0 Diagram Example

IDEF0 is a function modeling methodology for describing manufacturing functions, which offers a functional modeling language for the analysis, development, re-engineering, and integration of information systems; business processes; or software engineering analysis. It is part of the IDEF family of modeling languages in the field of software engineering, and is built on the functional modeling language building SADT.

The IDEF0 Functional Modeling method is designed to model the decisions, actions, and activities of an organization or system. It was derived from the established graphic modeling language Structured Analysis and Design Technique (SADT) developed by Douglas T. Ross and SofTech, Inc.. In its original form, IDEF0 includes both a definition of a graphical modeling language (syntax and semantics) and a description of a comprehensive methodology for developing models. The US Air Force commissioned the SADT developers to develop a function model method for analyzing and communicating the functional perspective of a system. IDEF0 should assist in organizing system analysis and promote effective communication between the analyst and the customer through simplified graphical devices.

Business Process Modeling Notation



Business Process Modeling Notation Example.

Business Process Modeling Notation (BPMN) is a graphical representation for specifying business processes in a workflow. BPMN was developed by Business Process Management Initiative (BPMI), and is currently maintained by the Object Management Group since the two organizations merged in 2005. The current version of BPMN is 1.1, and a major revision process for BPMN 2.0 is in progress.

The Business Process Modeling Notation (BPMN) specification provides a graphical notation for specifying business processes in a Business Process Diagram (BPD). The objective of BPMN is to support business process management for both technical users and business users by providing a notation that is intuitive to business users yet able to represent complex process semantics. The BPMN specification also provides a mapping between the graphics of the notation to the underlying constructs of execution languages, particularly BPEL4WS.

Axiomatic Design

Axiomatic design is a top down hierarchical functional decomposition process used as a solution synthesis framework for the analysis, development, re-engineering, and integration of products, information systems, business processes or software engineering solutions. Its structure is suited mathematically to analyze coupling between functions in order to optimize the architectural robustness of potential functional solution models.

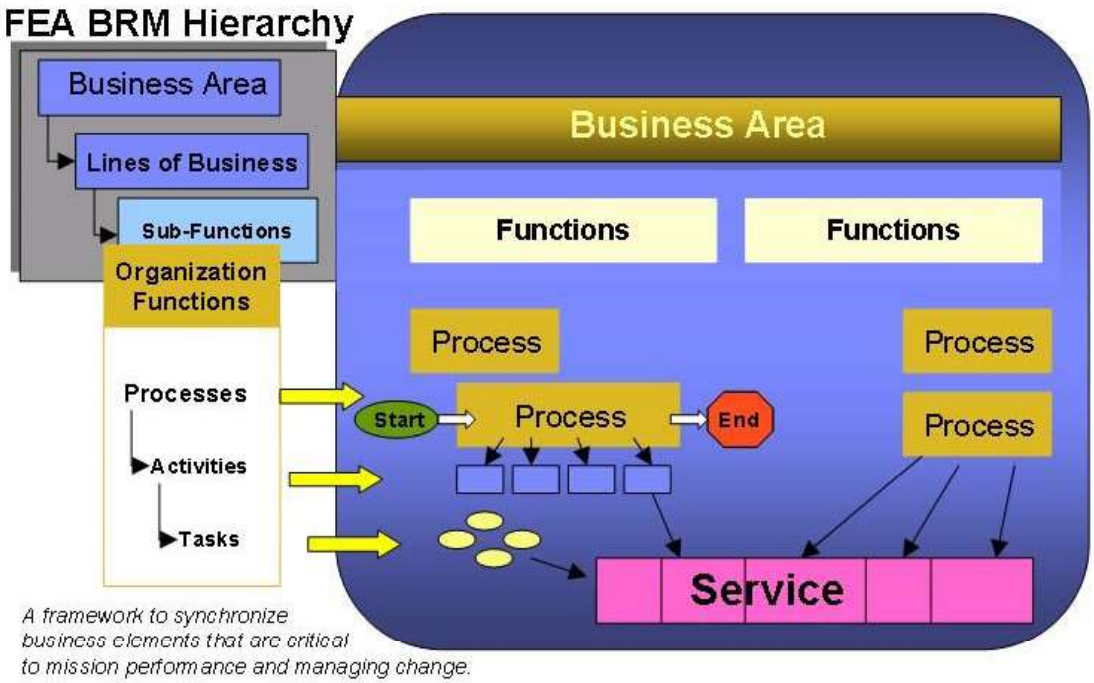
Other types of function models

In the field of systems and software engineering numerous specific function and functional models have been defined. Here only a few general types will be explained.

Business function model

A *Business Function Model* (BFM) is a general description or category of operations performed routinely to carry out an organization's mission. It can show the critical business processes in the context of the business area functions. The processes in the business function model must be consistent with the processes in the value chain models. Processes are a group of related business activities performed to produce an end product or to provide a service. Unlike business functions that are performed on a continual basis, processes are characterized by the fact that they have a specific beginning and an end point marked by the delivery of a desired output. The figure on the right depicts the relationship between the business processes, business functions, and the business area's business reference model.

Business reference model



This FEA Business reference model depicts the relationship between the business processes, business functions, and the business area's business reference model.

A Business reference model is a reference model, concentrating on the functional and organizational aspects of the core business of an enterprise, service organization or government agency. In enterprise engineering a business reference model is part of an

Enterprise Architecture Framework or *Architecture Framework*, which defines how to organize the structure and views associated with an Enterprise Architecture.

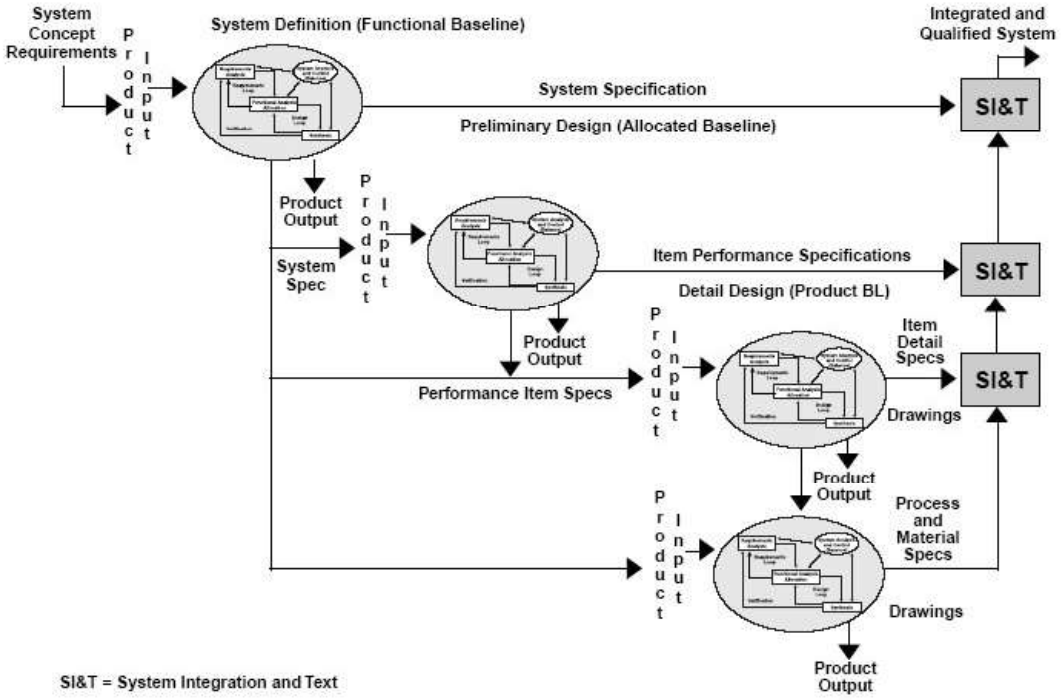
A reference model in general is a model of something that embodies the basic goal or idea of something and can then be looked at as a reference for various purposes. A business reference model is a means to describe the business operations of an organization, independent of the organizational structure that perform them. Other types of business reference model can also depict the relationship between the business processes, business functions, and the business area's business reference model. These reference model can be constructed in layers, and offer a foundation for the analysis of service components, technology, data, and performance.

Operator function model

The *Operator Function Model* (OFM) is proposed as an alternative to traditional task analysis techniques used by human factors engineers. An operator function model attempts to represent in mathematical form how an operator might decompose a complex system into simpler parts and coordinate control actions and system configurations so that acceptable overall system performance is achieved. The model represents basic issues of knowledge representation, information flow, and decision making in complex systems. Miller (1985) suggests that the network structure can be thought of as a possible representation of an operator's internal model of the system plus a control structure which specifies how the model is used to solve the decision problems that comprise operator control functions.

Chapter- 12

Functional Specification



Systems engineering model of Specification and Levels of Development. During system development a series of specifications are generated to describe the system at different levels of detail. These program unique specifications form the core of the configuration baselines. As shown here, in addition to referring to different levels within the system hierarchy, these baselines are defined at different phases of the design process.

A **functional specification** (also, *functional spec*, *specs*, *functional specifications document (FSD)*, or *Program specification*) in systems engineering and software development is the documentation that describes the requested behavior of an engineering system. The documentation typically describes what is needed by the system user as well as requested properties of inputs and outputs (e.g. of the software system).

Overview

In systems engineering a specification is a document that clearly and accurately describes the essential technical requirements for items, materials, or services including the procedures by which it can be determined that the requirements have been met. Specifications help avoid duplication and inconsistencies, allow for accurate estimates of necessary work and resources, act as a negotiation and reference document for engineering changes, provide documentation of configuration, and allow for consistent communication among those responsible for the eight primary functions of Systems Engineering. They provide a precise idea of the problem to be solved so that they can efficiently design the system and estimate the cost of design alternatives. They provide guidance to testers for verification (qualification) of each technical requirement.

A functional specification does not define the inner workings of the proposed system; it does not include the specification how the system function will be implemented. Instead, it focuses on what various outside agents (people using the program, computer peripherals, or other computers, for example) might "observe" when interacting with the system. A typical functional specification might state the following:

When the user clicks the OK button, the dialog is closed and the focus is returned to the main window in the state it was in before this dialog was displayed.

Such a requirement describes an interaction between an external agent (the user) and the software system. When the user provides input to the system by clicking the OK button, the program responds (or should respond) by closing the dialog window containing the OK button.

It can be *informal*, in which case it can be considered as a blueprint or user manual from a developer point of view, or *formal*, in which case it has a definite meaning defined in mathematical or programmatic terms. In practice, most successful specifications are written to understand and fine-tune applications that were already well-developed, although safety-critical software systems are often carefully specified prior to application development. Specifications are most important for external interfaces that must remain stable.

Functional specification topics

Purpose

There are many purposes for functional specifications. One of the primary purposes on team projects is to achieve some form of team consensus on what the program is to achieve before making the more time-consuming effort of writing source code and test cases, followed by a period of debugging. Typically, such consensus is reached after one or more reviews by the stakeholders on the project at hand after having negotiated a cost-effective way to achieve the requirements the software needs to fulfill.

Process

In the ordered industrial software engineering life-cycle (waterfall model), functional specification describes what has to be implemented. The next system specification document describes how the functions will be realized using a chosen software environment. In not industrial, prototypical systems development, functional specifications are typically written after or as part of requirements analysis.

When the team agrees that functional specification consensus is reached, the functional spec is typically declared "complete" or "signed off". After this, typically the software development and testing team write source code and test cases using the functional specification as the reference. While testing is performed the behavior of the program is compared against the expected behavior as defined in the functional specification.

Types of software development specifications

- Advanced Microcontroller Bus Architecture
- Bit specification
- Design specification
- Diagnostic design specification
- Multiboot Specification
- Product design specification
- Real-time specification for Java
- Software Requirements Specification

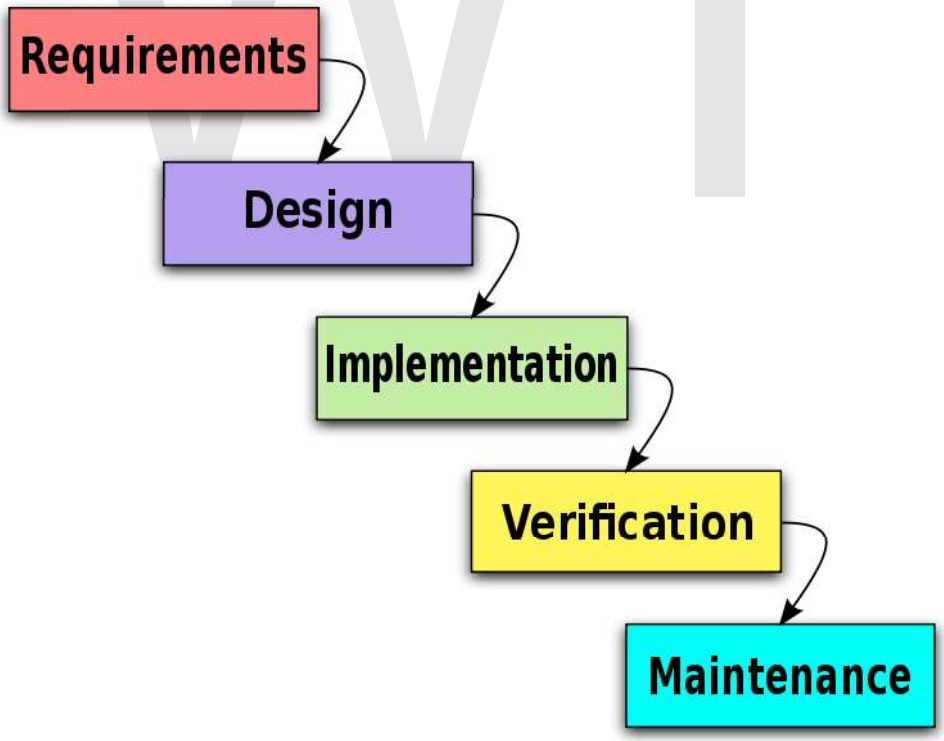
Chapter- 13

Dual Vee Model

The **Dual Vee Model** builds on the V-Model to cleanly depict the complexity associated with designing and developing systems. In systems engineering it defines a uniform procedure for product or project development. The model depicts concurrent development of a system's architecture as one Vee with each entity of that architecture as another Vee that intersects the architecture Vee. This clearly shows interactions and sequences in developing a complex system and a system of systems.

Background

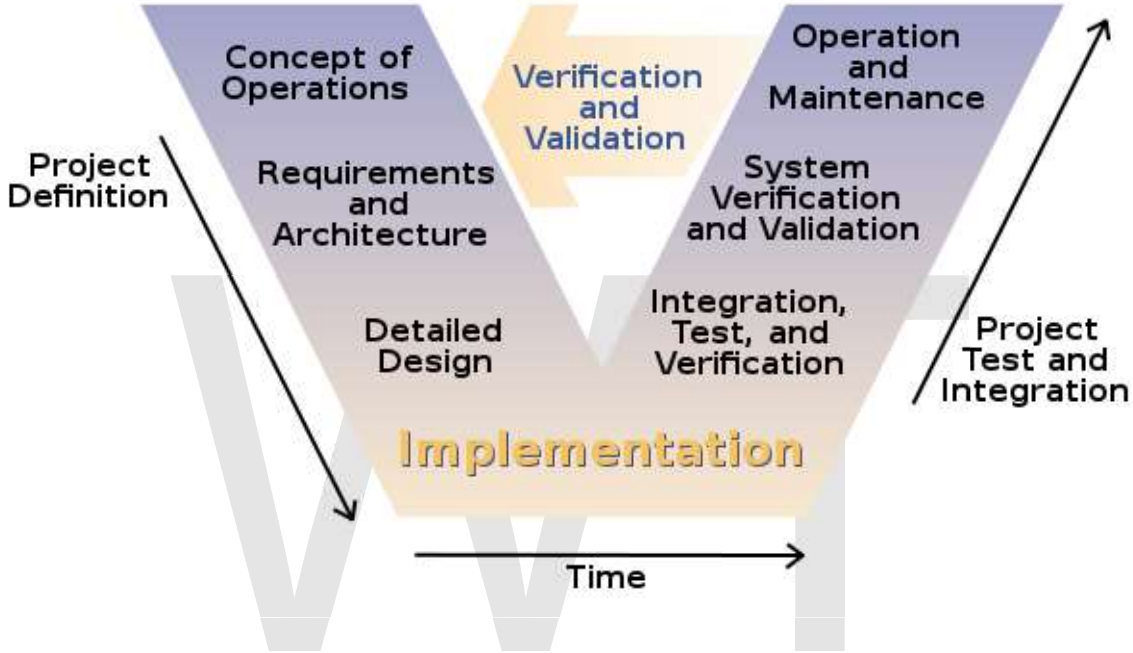
Waterfall Model



The unmodified "waterfall model". Progress flows from the top to the bottom, like a waterfall.

The Waterfall Model breaks up the development process into development phases. The name implies that design decisions flow from the requirements, implementation decisions flow from the design, etc. On a large project, many different people only work on each part. So a designer may ask, "What am I trying to design?" and the response would be, "You're trying to design something that will satisfy the product requirements." The builder may ask, "What am I trying to build?" and the response would be, "You're trying to build what was designed." etc.

Vee Model

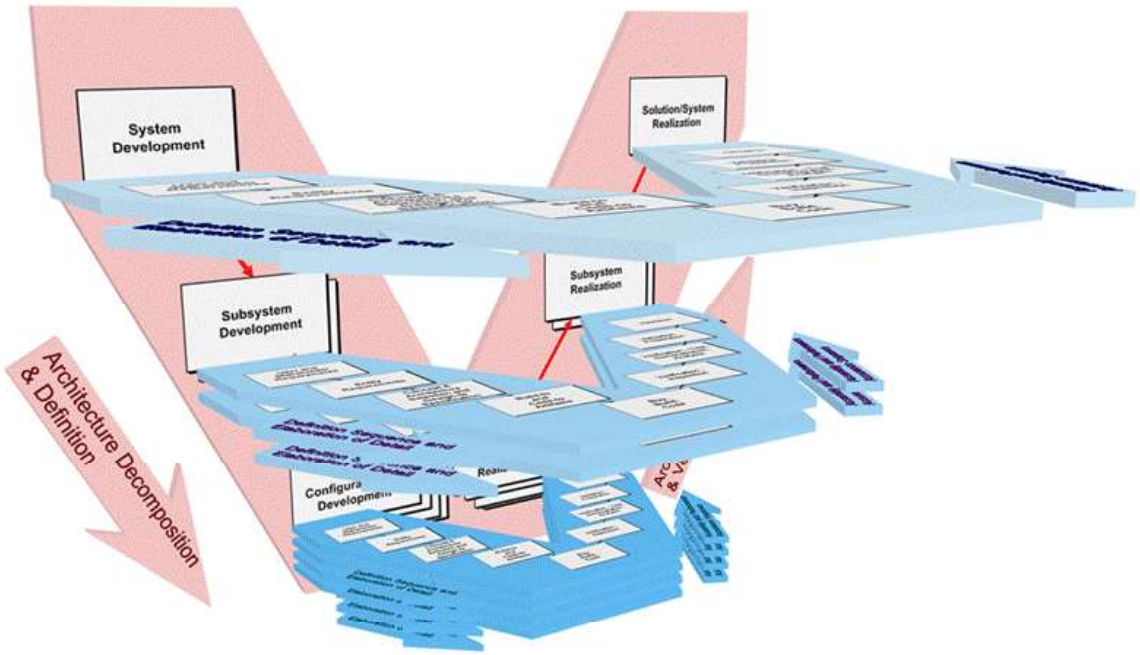


The V-model of the Systems Engineering Process.

The Vee Model organizes development phases into levels of complexity with the most complex item on top and least complex item on bottom (a.k.a. Lowest Configuration Item). This places the requirements at the beginning next to the product's operation at the end, and the design next to verification. This makes sense because when developer delivers a product to the customer, the customer may ask, "Why should I accept this product?" and the developer should answer, "Because it meets your (the customer's) requirements." The requirements are connect to the operation. When performing product testing, the test engineer may ask, "What tests should I conduct?" and the designer should answer, "You should conduct tests to show the product that was built matches the design." Verification is connected to the design.

The Vee Model can be expanded in several ways to meet system requirements. It can include the seven INCOSE layers of system complexity (i.e. system, element, subsystem, assembly, subassembly, component, and part). It can include decomposition, definition, integration, and verification. It may also include user/stakeholder participation, opportunity and risk management, and problem resolution.

Dual Vee Model



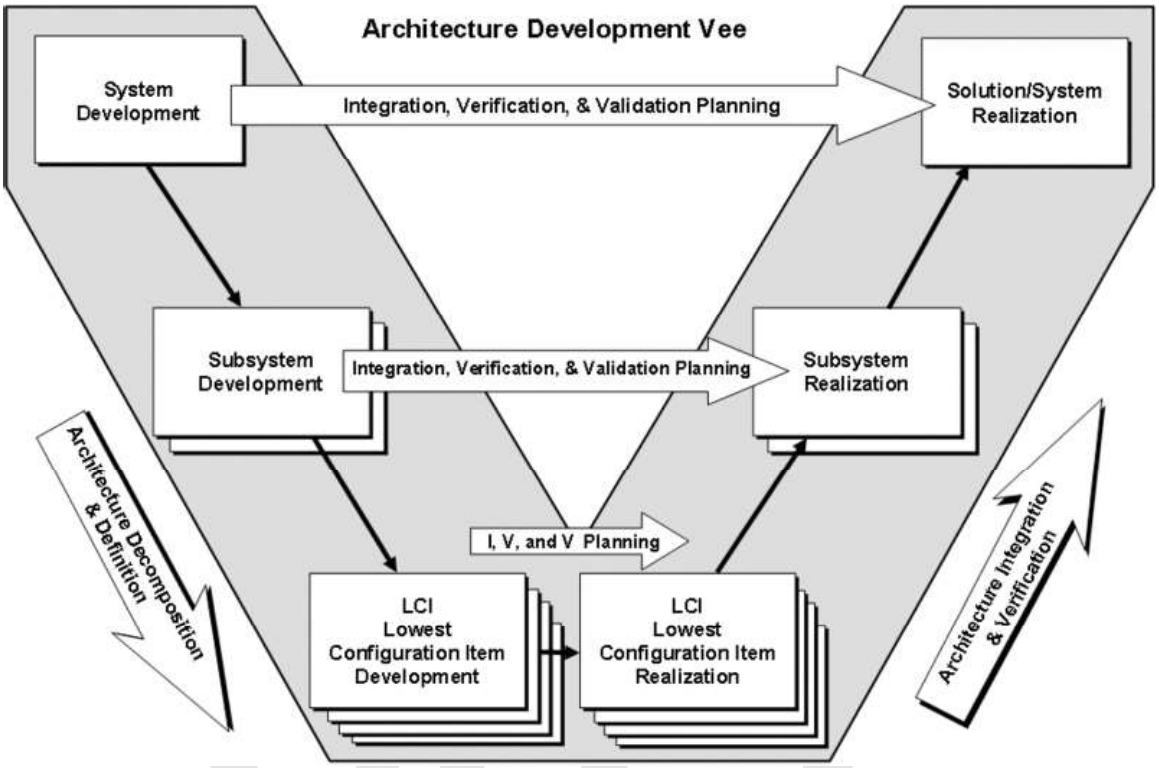
Architecture and Entity Vees Intersecting. Source - Kevin Forsberg and Hal Mooz 2006.

The Dual Vee Model builds on the Vee Model to manage a system of systems. An architecture Vee manages the system and entity Vees branch off the architecture Vee to manage sub-systems.

For example, GPS includes a constellation of satellites, a ground control network, and millions of users worldwide. Each satellite, ground control center, and GPS receiver is a complex system that could be managed by a separate Entity Vee. Development of a satellite can affect the design, manufacturing, or cost of receivers. Similarly, development of a receiver can affect design, manufacturing, or cost of satellites. So everything must be integrated into a system of systems that are developed within a larger Architecture Vee.

The Architecture Vee

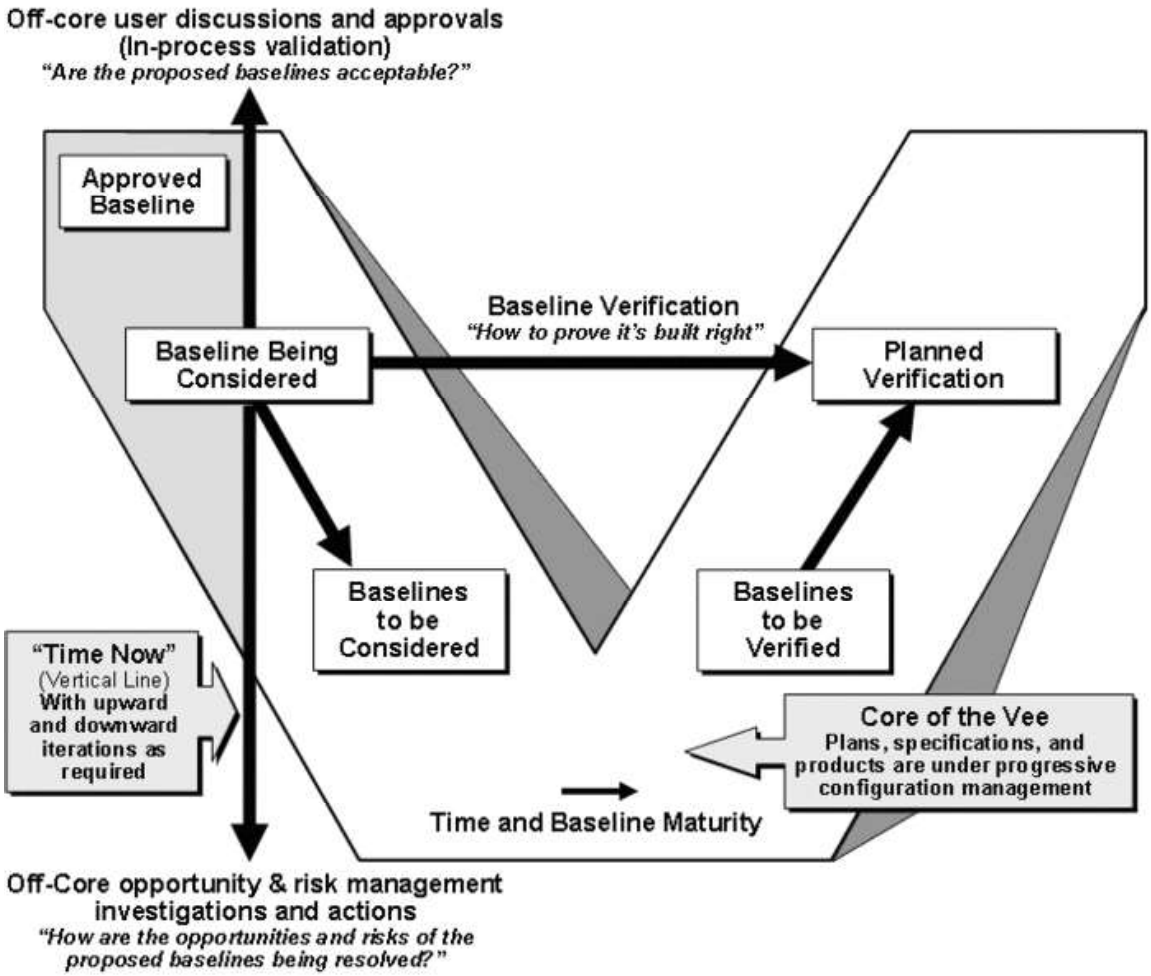
When developing complicated systems, a system engineer must manage a system baseline configuration from start to finish. The baseline can include design documents, user manuals, the product itself, and should answer every What?, Why?, and Who? for a system's architecture. At each development phase, there will be changes to the system, which will change the baseline.



Architecture Development Vee Model (Provides What, Why, and Who). Source - Kevin Forsberg and Hal Mooz 2006.

The core of the Vee is the evolving architecture baseline from initial requirements to a delivered system. The Architecture Vee produces the what, why, and who (which entity level) is responsible for a system's architecture.

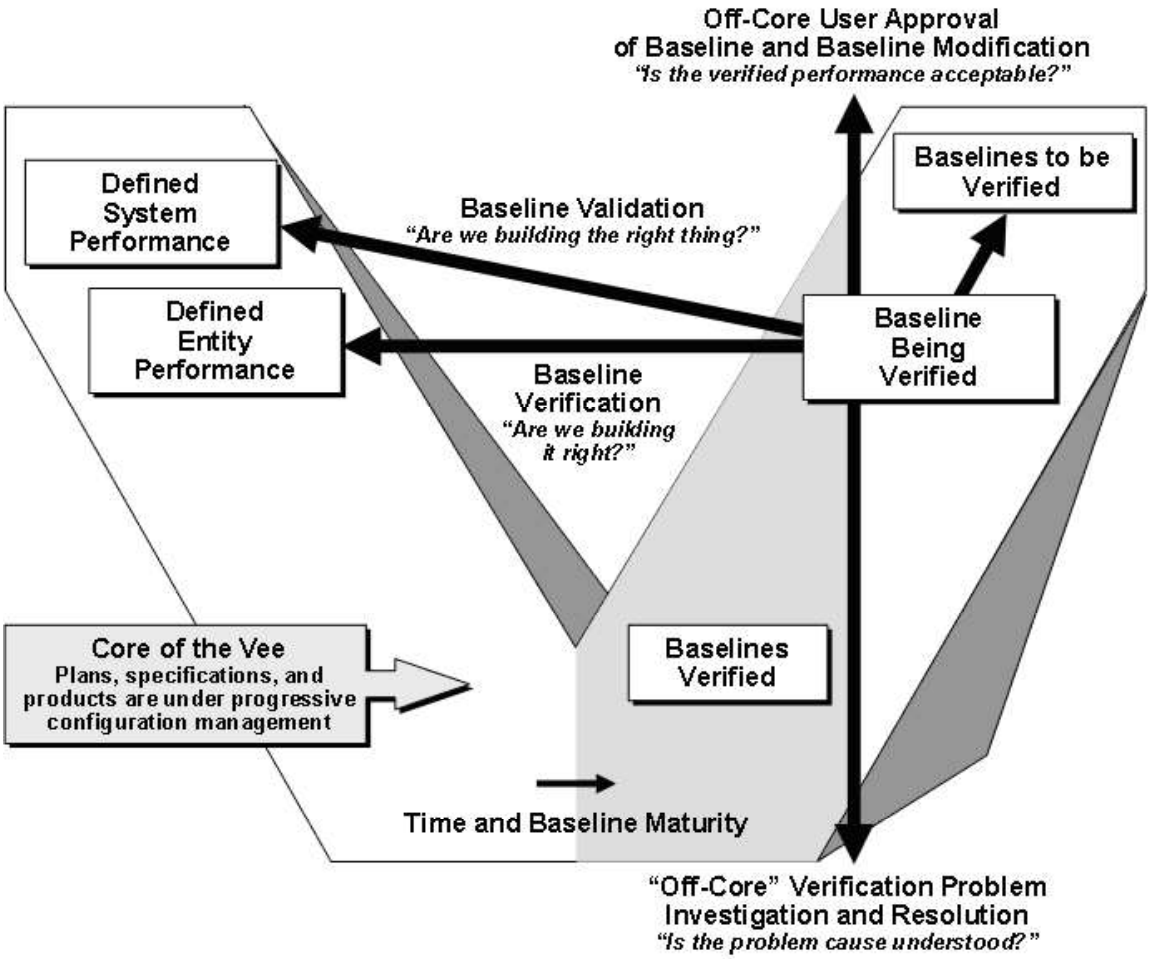
Downward off-Vee core investigations (figure - below) facilitate gaining knowledge to justify architecture baseline decisions made on the Vee core. Upward off core communication with customers and users facilitates in-process validation keeping the stakeholders abreast of and committed to the evolving baseline. Note that in all Vee representations time and maturity move from left to right. Just as we cannot move backward in time, so too one cannot move from right to left in the Vee model representation. Iteration is essential in system development, and all iteration is done vertically off-core, upward to users and customers (which is in-process validation), and downward for opportunity and risk management, as shown in the following figure.



Vee Model - Opportunity and Risk Investigations. Source - Kevin Forsberg and Hal Mooz 2006.

The left leg off-Vee core investigations center around what concept is best and what architecture is best for that concept. For example, commercial products usually face the dilemma as to whether batteries should be standard, unique, replaceable, or not. Downward off-Vee core investigations and analysis can facilitate determining the most desirable approach that would then be baselined on the Vee core if the stakeholders agree. Similar investigations can prove the viability and technical feasibility of candidate concepts.

Right leg off-Vee core downward investigations (figure - below) are directed at investigating integration anomalies to determine their root cause and to correct them. Upward communication with stakeholders determines if they can live with the as-integrated and as-verified performance.



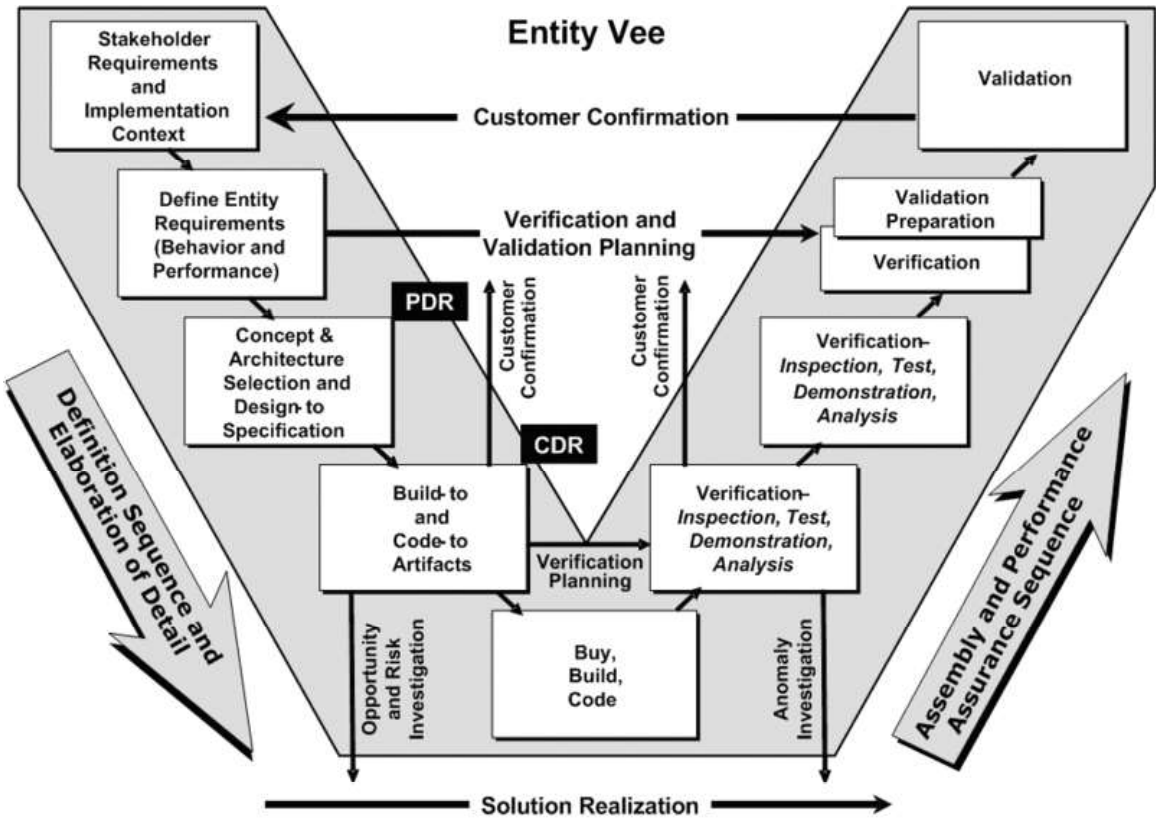
Vee Model - Integration and Verification. Source - Kevin Forsberg and Hal Mooz 2006.

At each decomposition level there is a direct correlation between activities on the left and right sides of the Vee. This is deliberate. For example, the method of integration, verification, and validation to be used on the right must be determined on the left as concepts are defined at each decomposition level. This minimizes the chances that concepts are conceived in a way that cannot be carried out.

The Entity Vee Model

The Entity Vee illustrates the entity development and realization process which describes how each entity will be obtained (development, purchase, reuse, etc.). An Entity Vee (figure - below) exists for every entity of the architecture from the system, down to the lowest configuration items (LCIs), such as computer software units or hardware components. All activities within an Entity Vee reside at the same architecture level (System, Subsystem, LCI). The left Vee leg represents entity definition elaboration from very sketchy user requirements, through concept determination and on to design-to specifications and fully detailed build-to artifacts. The right Vee leg represents the

sequence of entity assembly and performance assurance on through verification and validation of the entity.



Entity Vee Model (Provides How). Source - Kevin Forsberg and Hal Mooz 2006.

At each elaboration, there is a direct correlation between activities on the left and right legs of the Entity Vee. Again this is deliberate. The method of verification to be used on the right Vee leg must be defined as requirements are developed on the left, otherwise requirements might be created that could not be verified. For example “user friendly” is a valid requirement, but it is unverifiable. Instead, a requirement that a computer screen display have “no more than five lines of 14-point text” defines one user's view of “user friendly” in measurable terms. Verification plans should be baselined to ensure verification requirements and methods are known and planned for at the design-to decision gate, commonly called Preliminary Design Review (PDR). Draft verification procedures based on the verification requirements, verification plan, and proposed entity design should be available at the build-to and code-to decision gate, commonly called Critical Design Review (CDR). This reduces the chances that verification as specified cannot in fact be performed.

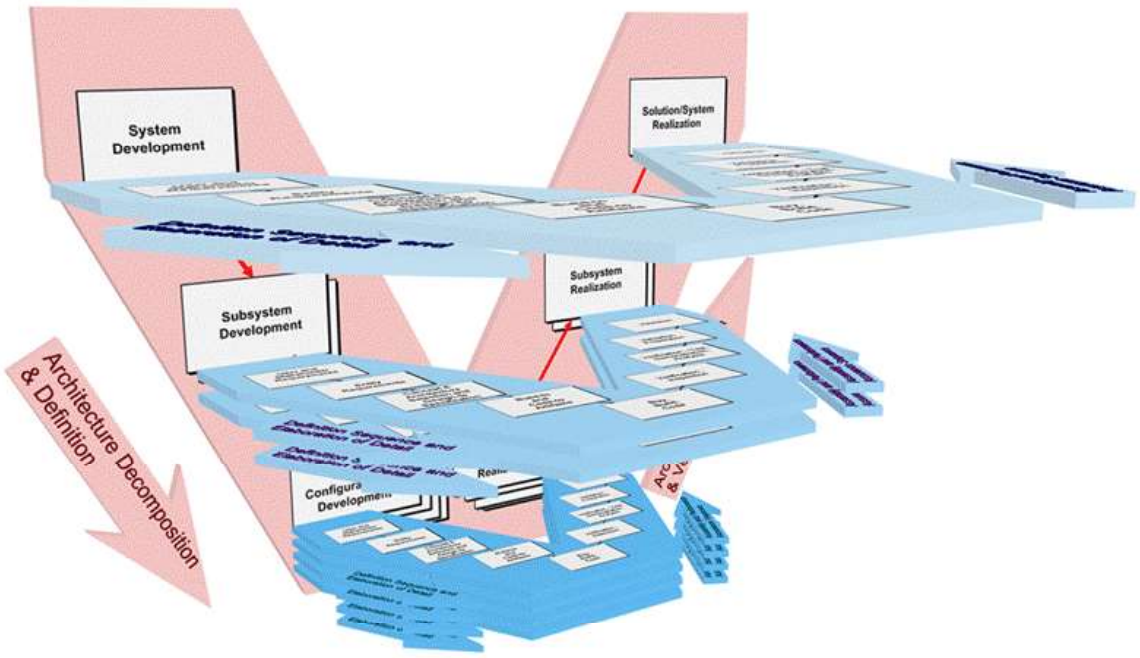
The vertical dimension of the Entity Vee is baseline elaboration at the selected architecture level and the core of the Entity Vee represents entity baseline elaboration progression. Also included (similar to the Architecture Vee) are the activities associated with opportunity and risk management, pursued downward and off-core to the level of

detail necessary for issue evaluation and resolution. For example, laboratory test of a computer chip or of software code may be necessary to confirm technical feasibility. Unlike the commonly held view of the Waterfall Model, there is no prohibition against doing exploratory design and analysis at any point in the project cycle to investigate or prove performance or feasibility. Unlike the Spiral Model, the Vee opportunity and risk investigations may be performed either in series or in parallel with the on-core development work, rather than being conducted sequentially and prior to the design development process. Hardware and software requirements-understanding models or technical feasibility models are encouraged early in the project cycle to pursue opportunities, such as new technologies, and to reduce risk. For instance, to evaluate a concept of a manual override versus full automation, technical feasibility of the two concepts could be modeled with selection based on response time versus cost. Customer confirmation could then provide valuable in-process validation of the preferred approach.

In the right leg, downward off-core investigations are applied to resolve assembly and verification anomalies. This may require descending to design errors, a cold solder joint, or operator error and the like. Upward off-core user interactions obtain user and customer confirmation or rejection of the realized performance. Note that in the entity Vee these interactions address individual entity solutions and not the integration of the architecture which is conducted on the Architecture Vee. At any level of decomposition, the customer of an entity is the manager of the next higher level of decomposition. For example, the power subsystem manager is the customer of the battery and is responsible for battery validation.

Dual Vees: Intersecting Architecture and Entity Vees

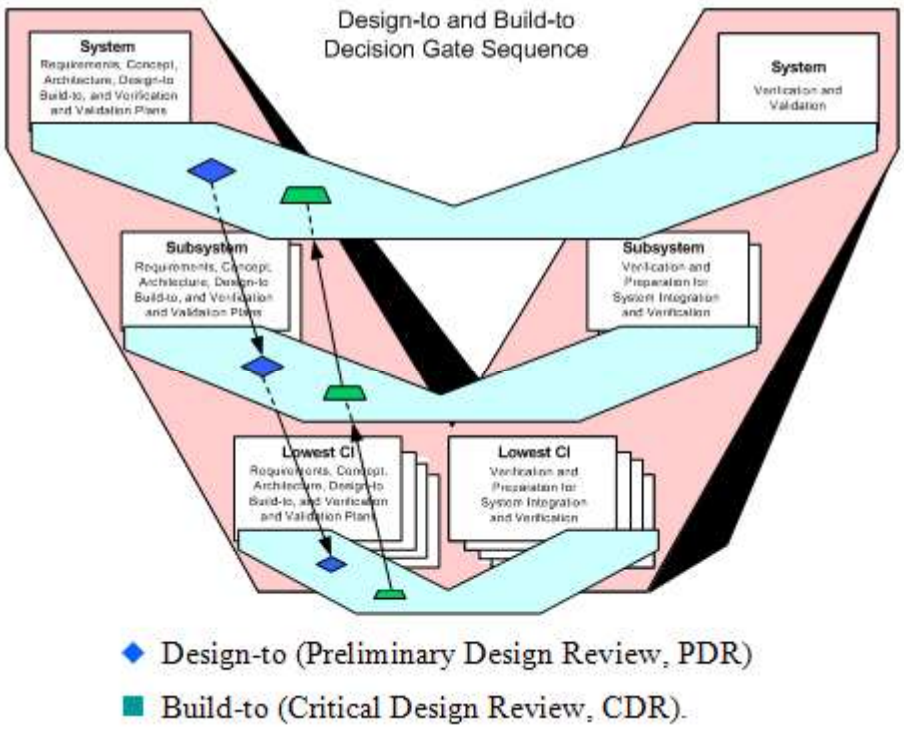
To evolve user needs into a system that satisfies those needs requires a best value solution for every entity of the architecture. This can be visualized by positioning Entity Vees orthogonal to the Architecture Vee as shown in the figure below. For each entity of the Architecture Vee there is a corresponding Entity Vee that addresses the entity development and realization. For example, the Architecture Vee of the figure below contains two subsystems hence there are two Entity Vees to represent the concurrent development of those two subsystems.



Architecture and Entity Vees Intersecting. Source - Kevin Forsberg and Hal Mooz 2006.

Phasing of decision gates

Architecture entities are developed and integrated into the system architecture in a phased sequence consistent with systems engineering best practices. The figure below provides a three dimensional view of Design-to and Build-to Decision Gate phasing



Design-to and Build-to Decision Gate Sequence. Source - Kevin Forsberg and Hal Mooz 2006.

For simplicity of illustration, only one Entity Vee is shown intersecting the Architecture Vee at each decomposition level. Note that the design-to sequence is top down, starting at the system level and proceeding consistent with decomposition to the lowest configuration item level (LCI). This sequence ensures that there is proper top down requirements flowdown and traceability.

When build-to and code-to artifacts, including draft verification procedures, are ready for baselining, the build-to decision gate sequence is conducted bottom up to prove the feasibility of building or coding the designs. The decision gate also confirms that, if the solution is built according to the build-to artifacts, the required performance will be achieved. This sequence ensures that, if the entity designs satisfy their design-to requirements, the entities will integrate into the next higher level entity, will perform as expected, and will meet user requirements.

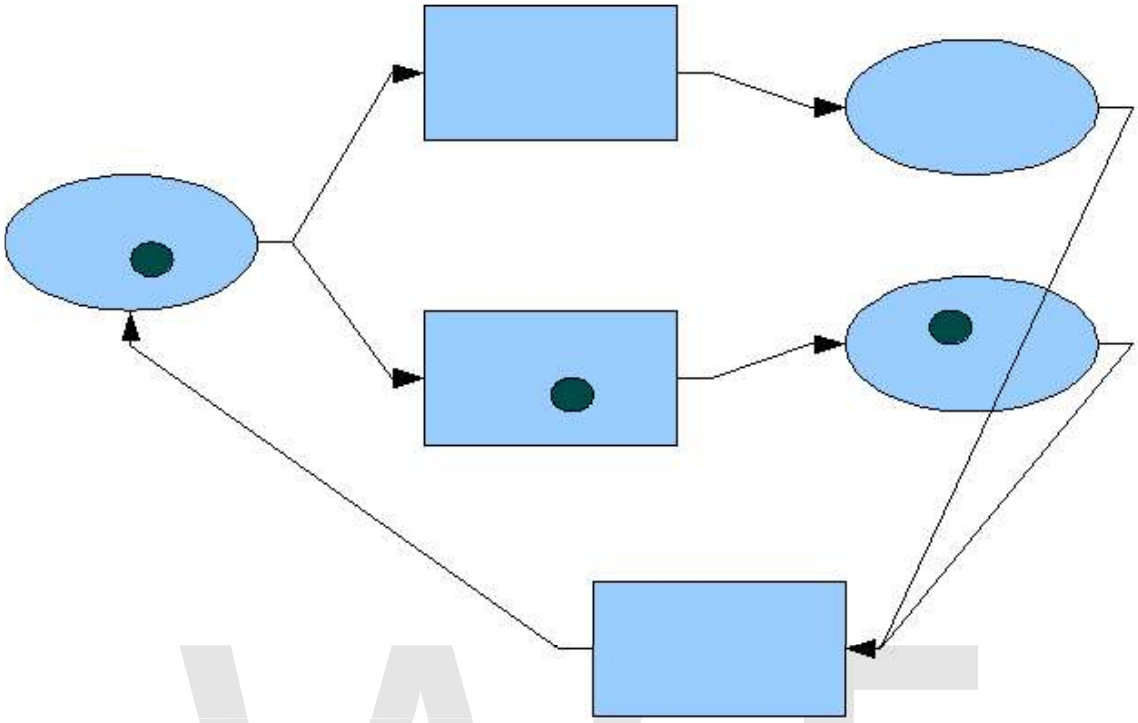
Chapter- 14

Dualistic Petri Nets

Dualistic Petri nets (dPNs) are a process-class variant of Petri nets. Like Petri nets in general and many related formalisms and notations, they are used to describe and analyze process architecture.

Process Modeling with dPNs

A simple, yet powerful way to model process architecture is using the dualistic extension of Petri nets called dualistic Petri nets (dPNs). A Petri net (PN) is a graphical, bipartite modeling language that intuitively and mathematically represent theoretical relationships of moving objects in a network of interconnected constructs. Classical Place/Transition PNs can represent theoretical processes, where the movement of objects implies their transformation, but is too absolute to be pragmatic in representing real-world processes. The real world is dualistic in nature and process is a dualistic phenomenon, this can not be easily represented using a digital-type modeling system. Instead, dualistic extensions to Place/Transition PNs have been introduced and used successfully in modeling the architecture of computer-based systems and business processes.



Dualistic Petri Net Simulation: Rectangles = Transformations, Ovals = Places

Among the distinctions of dPNs from classical PNs is space and time (due to energy use) in both the place construct and transformation construct. This results in the simulation effect of **marked transformations** that allow for the explicit representation of parallel processing, multiprocessing, and the implicit representation of deterioration of objects – all unique to dualistic Petri nets.

Architecture

Besides a propensity to modeling dualistic real-world behavior, PNs also offer a way to manage complex process systems hierarchically. Using classical PN construction rules, Petri nets of Petri nets can be built and a hierarchical conception of a complex process system can be studied. This structure of hierarchical abstraction is the heart of process architecture!

Bottom-Up: Starting with the manifested process

Dualistic Petri nets are capable of modeling any process system at its manifested level. When reverse engineering a manifested process, dPNs have a one-to-one correspondence of dPN construct to any manifested process piece, that is, it is isomorphic to the implementation language of the manifested process. For example, several lines of software code could be represented by one dPN transformation construct. Once the manifested process is completely represented by a network of dPNs, small, well-coupled groups of dPN constructs can be lumped together to form higher level dPN constructs – creating a network of dPNs at a higher level of hierarchical abstraction. Each level of

abstraction is consistent with its adjacent levels of abstraction and the rules governing them at each level are the exactly the same because PN abstractions are homomorphic. Now the process design can be considered at various levels of abstraction as deemed appropriate by the process architect, allowing for studies in its dynamic behavior and performance.

A typical use of reverse engineering using dPNs in the business world is in the documentation of processes for quality certification against standards like ISO 9000. In this case, dPNs are used to model pieces of the business process, which are then combined to form an overall enterprise architecture. The process system can be studied to determine each piece's capability and show where risks occur. Requirements are then reverse-engineered and applied at corresponding dPN constructs. Trouble spot processes can be identified and slated for reengineering. The overall dPN map not only gives quality entities the necessary information about a business's current process, but it also gives the process architect a blue print from which to manage and improve said processes. This is a major portion of quality engineering.

Top-down: From Idea to Implementation

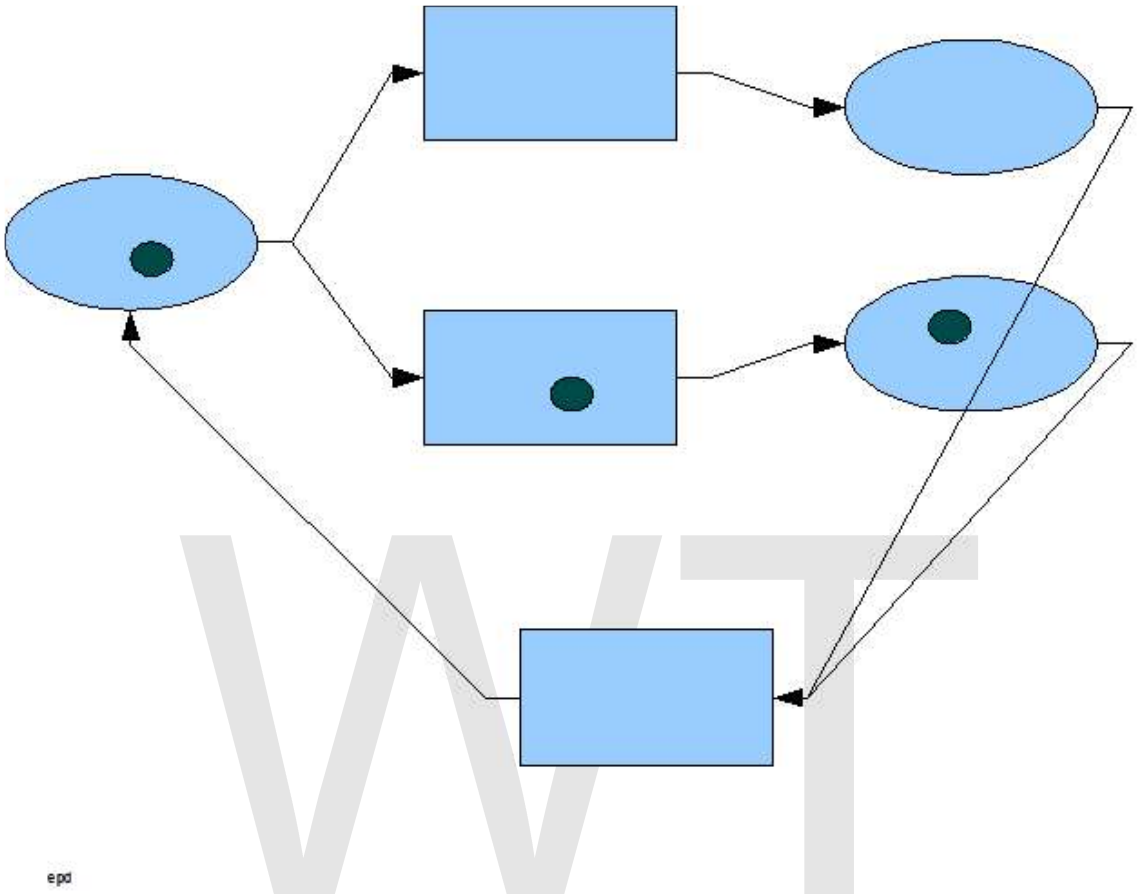
dPN modeling of a new process system starts at a high level of hierarchical abstraction. To design a complex process system, such as a sophisticated hardware component or a major project, the process architect must first define the problem space. Since the problem space is itself a process system, dPNs can be used for its modeling. Abstract dPNs that are yet to be implemented are specified within the context of the problem space. These constructs define the solution space within its context network. It is now up to the process architect to traverse down the hierarchical abstraction dimension, proposing new process designs for the solution space in an iterative manner until specifying the actual implementation in the specific implementation language.

This method for designing a complex process system is reflected in the general software development methodology known as the waterfall model. Actually, this method is not well-suited for the development of complex software without adjusting it to handle the step-wise decomposition of the process architecture. This decomposition occurs entirely within the domain of dPNs from the problem space context model down to the final mapping of the implementation language.

Process Structure

Whether a dPN hierarchical network map was created from the bottom up or from the top down, it shows the structure of the process system. Complex process systems, such as large software programs, will have several layers of hierarchical abstraction. At the top of the structure is one process represented by a couple of dPN constructs. Each subsequent layer below this process is the decomposition of the dPN constructs made up of more dPNs that are in turn decomposed. The “parent” dPN of a set of decomposed dPNs has associated with it requirements that apply to the decomposed network. These requirements were determined by studying the parent dPN's **suprastructure** or the

hierarchical structure *above* the construct. The decomposed "children" dPNs form the **infrastructure** or the hierarchical structure *below* the parent dPN.



dPNet-modeled Process Architecture

In complex computer design, requirements are generated and infrastructures proposed. Chosen infrastructures are then further decomposed by determining the new constructs' requirements and decomposing them further in this iterative fashion until the dPNs are decomposed into the implementation language of software or hardware specification. The final hierarchical dPN map documents the architectural decisions that were accepted and a specification is in place that can be used to maintain the system's future evolution.

In business processes, process requirements are policies that must be fulfilled by acceptable procedures. Complex procedures can be specified by simpler procedures. Since business processes are processes, dPNs are an ideal modeling language for them – especially when considering complex business processes like logistics.

Conclusion

The entire network of dualistic Petri nets becomes the architecture specification of the process system. If the problem and solution space are entirely in software, it is known as software architecture. If the problem and solution space are business processes, it is

known as enterprise architecture. If the problem and solution space are networked equipment, it is known as network architecture. What is important to each of these applications and to any other process system of varying complexity is that the hierarchical map of the system's structure created by the network of dPNs allows the process architect to study the behavior and performance of the system, keeps architectural design decisions documented, and organizes process requirements along the architectural structure.

Chapter- 15

Change Management (Engineering)

The **change management** process in systems engineering is the process of requesting, determining attainability, planning, implementing, and evaluating of changes to a system. It has two main goals: supporting the processing of changes – which is mainly discussed here – and enabling traceability of changes, which should be possible through proper execution of the process described here.

Introduction

There is considerable overlap and confusion between change management, change control and configuration management. The definition below does not yet integrate these areas.

Change management is an important process, because it can deliver vast benefits (by improving the system and thereby satisfying "customer needs"), but also enormous problems (by ruining the system and/or mixing up the change administration). Furthermore, at least for the Information Technology domain, more funds and work are put into system maintenance (which involves change management) than to the initial creation of a system. Typical investment by organizations during initial implementation of large ERP systems is 15-20% of overall budget.

In the same vein, Hinley describes two of Lehman's laws of software evolution: the law of continuing change (i.e. systems that are used must change or automatically become less useful) and the law of increasing complexity (i.e. through changes the structure of a system becomes ever more complex and more resources are needed to simplify it).

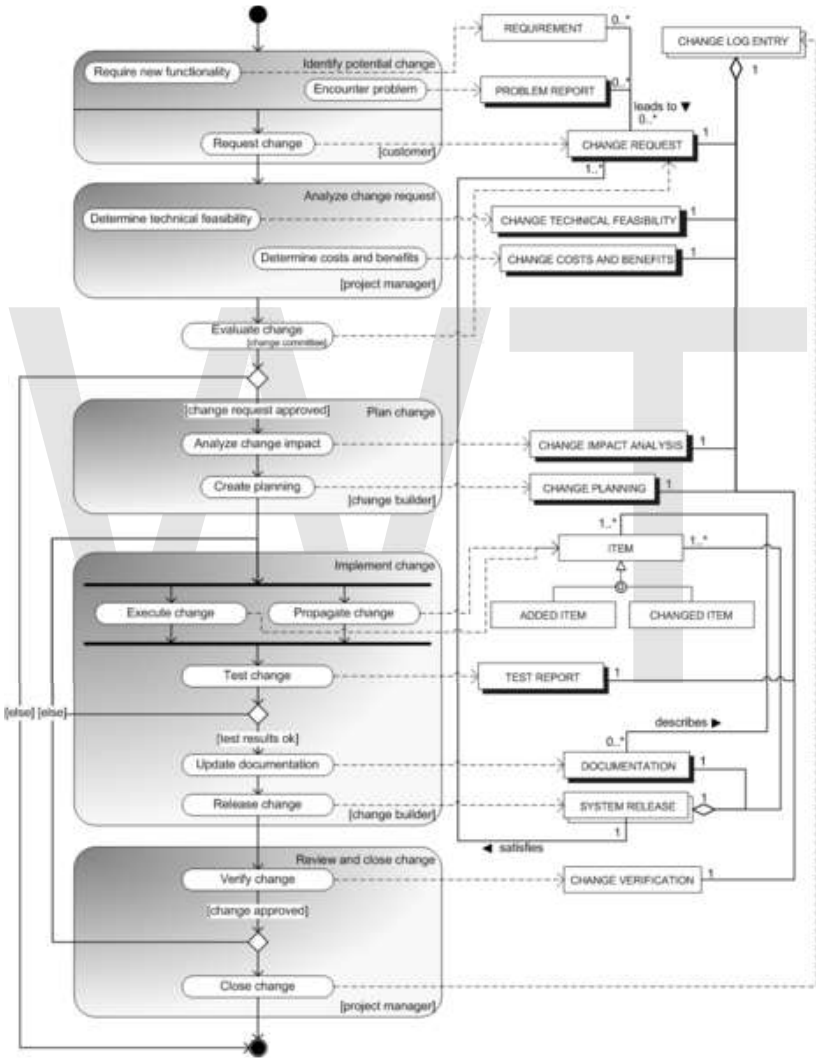
The field of manufacturing is nowadays also confronted with many changes due to increasing and worldwide competition, technological advances and demanding customers. Therefore, (efficient and effective) change management is also of great importance in this area.

It is not unthinkable that the above statements are true for other domains as well, because usually, systems tend to change and evolve as they are used. Below, a generic change management process and its deliverables are discussed, followed by some examples of instances of this process.

Notes: In the process below, it is arguable that the change committee should be responsible not only for accept/reject decisions, but also prioritization, which influences how change requests are batched for processing.

The process and its deliverables

For the description of the change management process, the meta-modeling technique is used. Figure 1 depicts the process-data diagram.



Activities

There are six main activities, which jointly form the change management process. They are: Identify potential change, Analyze change request, Evaluate change, Plan change, Implement change and Review and close change. These activities are executed by four different roles, which are discussed in Table 1. The activities (or their sub-activities, if applicable) themselves are described in Table 2.

Table 1: Role descriptions for the change management process

Role	Description
Customer	The customer is the role that requests a change due to problems encountered or new functionality requirements; this can be a person or an organizational entity and can be in- or external to the company that is asked to implement the change.
Project manager	The project manager is the owner of the project that the CHANGE REQUEST concerns. In some cases there is a distinct change manager, who in that case takes on this role.
Change committee	The change committee decides whether a CHANGE REQUEST will be implemented or not. Sometimes this task is performed by the project manager as well.
Change builder	The change builder is the person who plans and implements the change; it could be argued that the planning component is (partially) taken on by the project manager.

Table 2: Activity descriptions for the change management process

Activity	Sub-activity	Description
Identify potential change	Require new functionality	A customer desires new functionality and formulates a REQUIREMENT.
	Encounter problem	A customer encounters a problem (e.g. a bug) in the system and this leads to a PROBLEM REPORT.
	Request change	A customer proposes a change through creation of a CHANGE REQUEST.
Analyze change request	Determine technical feasibility	The project manager determines the technical feasibility of the proposed CHANGE REQUEST, leading to a CHANGE TECHNICAL FEASIBILITY.
	Determine costs and benefits	The project manager determines the costs and benefits of the proposed CHANGE REQUEST, resulting in CHANGE COSTS AND BENEFITS. This and the above sub-activity can be done in any order and they are independent of each other, hence the modeling as unordered activities.
Evaluate change		Based on the CHANGE REQUEST, its CHANGE TECHNICAL FEASIBILITY and CHANGE COSTS AND BENEFITS, the change committee makes the go/no-go decision. This is modeled as a separate activity because it is an important process step and has another role performing it. It is modeled as a sub-activity (without any activity containing it) as recommended by Remko Helms

(personal communication).

Plan change	Analyze change impact	The extent of the change (i.e. what other items the change effects) is determined in a CHANGE IMPACT ANALYSIS. It could be argued that this activity leads to another go/no-go decision, or that it even forms a part of the Analyze change request activity. It is modeled here as a planning task for the change builder because of its relationship with the activity Propagate change.
	Create planning	A CHANGE PLANNING is created for the implementation of the change. Some process descriptions (e.g. Mäkäräinen, 2000) illustrate that is also possible to 'save' changes and process them later in a batch. This activity could be viewed as a good point to do this.
Implement change	Execute change	The change is 'programmed'; this activity has a strong relationship with Propagate change, because sometimes the change has to be adapted to other parts of the system (or even other systems) as well.
	Propagate change	The changes resulting from Execute change have to be propagated to other system parts that are influenced by it. Because this and the above sub-activity are highly dependent on each other, they have been modeled as concurrent activities.
	Test change	The change builder tests whether what (s)he has built actually works and satisfies the CHANGE REQUEST. As depicted in the diagram, this can result in an iterative process together with the above two sub-activities.
	Update documentation	The DOCUMENTATION is updated to reflect the applied changes.
	Release change	A new SYSTEM RELEASE, which reflects the applied change, is made public.
Review and close change	Verify change	The implementation of the change in the new SYSTEM RELEASE is verified for the last time, now by the project manager. Maybe this has to happen before the release, but due to conflicting literature sources and diagram complexity

	considerations it was chosen to model it this way and include this issue.
Close change	This change cycle is completed, i.e. the CHANGE LOG ENTRY is wrapped up.

Deliverables

Besides activities, the process-data diagram (Figure 1) also shows the deliverables of each activity, i.e. the data. These deliverables or concepts are described in Table 3; in this context, the most important concepts are: CHANGE REQUEST and CHANGE LOG ENTRY.

A few concepts are defined by the author (i.e. lack a reference), because either no (good) definitions could be found, or they are the obvious result of an activity. These concepts are marked with an asterisk ('*'). Properties of concepts have been left out of the model, because most of them are trivial and the diagram could otherwise quickly become too complex. Furthermore, some concepts (e.g. CHANGE REQUEST, SYSTEM RELEASE) lend themselves for the versioning approach as proposed by Weerd , but this has also been left out due to diagram complexity constraints.

Table 3: Concept descriptions for the change management process

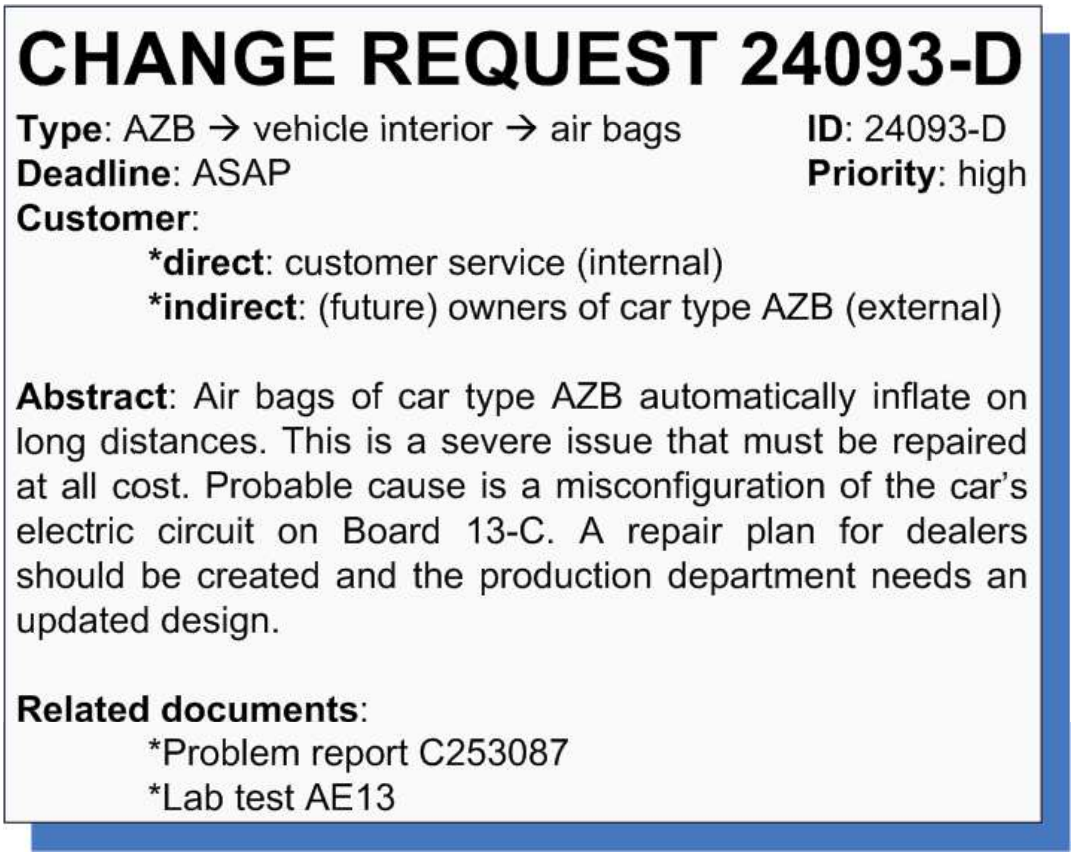
Concept	Description
REQUIREMENT	A required functionality of a component (or item; NASA, 2005).
PROBLEM REPORT	Document describing a problem that cannot be solved by a level 1 help desk employee; contains items like date, contact info of person reporting the problem, what is causing the problem, location and description of the problem, action taken and disposition, but this is not depicted in the diagram (Dennis, et al., 2002).
CHANGE REQUEST	Document that describes the requested change and why it is important; can originate from PROBLEM REPORTS, system enhancements, other projects, changes in underlying systems and senior management, here summarized as REQUIREMENTS (Dennis, et al., 2002). Important attribute: 'go/no-go decision', i.e. is the change going to be executed or not?
CHANGE LOG ENTRY*	Distinct entry in the collection of all changes (e.g. for a project); consists of a CHANGE REQUEST, CHANGE TECHNICAL FEASIBILITY, CHANGE COSTS AND BENEFITS, CHANGE IMPACT ANALYSIS, CHANGE PLANNING, TEST REPORT and CHANGE VERIFICATION. Not all these have to be included if the process is terminated earlier (i.e. if the change is not implemented).

CHANGE TECHNICAL FEASIBILITY	Concept that indicates whether or not "reliable hardware and software, technical resources capable of meeting the needs of a proposed system [i.e. change request] can be acquired or developed by an organization in the required time" (Vogl, 2004).
CHANGE COSTS AND BENEFITS	The expected effort required to implement and the advantages (e.g. cost savings, increased revenue) gained by implementing the change. Also named economic feasibility (Vogl, 2004).
CHANGE IMPACT ANALYSIS	An assessment of the extent of the change (Rajlich, 1999).
CHANGE PLANNING	"A scheme, method or design for the attainment of some objective or to achieve something [i.e. the change]" (Georgetown University, n.d.), in this case the change.
ITEM	"A non-specific term used to denote any product, including systems, subsystems, assemblies, subassemblies, units, sets, accessories, computer programs, computer software or parts" (Rigby, 2003); has (overlapping) subtypes ADDED ITEM and CHANGED ITEM.
ADDED ITEM*	Self-explanatory: a newly created ITEM; subtype of ITEM.
CHANGED ITEM*	Self-explanatory: an ITEM that already existed, but has been altered; subtype of ITEM.
TEST REPORT	"A document that describes the conduct and results of the testing carried out for a system or component [affected by the change]" (IEEE, 1991).
DOCUMENTATION	According to the Pennsylvania State University Libraries (2004) definition, DOCUMENTATION is "[p]rinted material which accompanies other materials (usually non-book), and which explains, gives instructions for use, or otherwise functions as a guide to the major materials." In this context, it can also be digital materials or even training, as long as it relates to (pieces of) the system.
SYSTEM RELEASE	"[M]erchandise issued for sale or public showing" (Princeton University, 2003). Consists of one or more ITEMS and the accompanying DOCUMENTATION.
CHANGE VERIFICATION	A determination of whether or not the result of the change implementation fulfills the requirements established earlier (Rigby, 2003).

Besides just 'changes', one can also distinguish deviations and waivers. A deviation is an authorization (or a request for it) to depart from a requirement of an item, prior to the creation of it. A waiver is essentially the same, but than during or after creation of the item. These two approaches can be viewed as minimalistic change management (i.e. no real solution to the problem at hand).

Examples

A good example of the change management process in action can be found in software development. Often users report bugs or desire new functionality from their software programs, which leads to a change request. The product software company then looks into the technical and economical feasibility of implementing this change and consequently it decides whether the change will actually be realized. If that indeed is the case, the change has to be planned, for example through the usage of function points. The actual execution of the change leads to the creation and/or alteration of software code and when this change is propagated it probably causes other code fragments to change as well. After the initial test results seem satisfactory, the documentation can be brought up to date and be released, together with the software. Finally, the project manager verifies the change and closes this entry in the change log.



Another typical area for change management in the way it is treated here, is the manufacturing domain. Take for instance the design and production of a car. If for example the vehicle's air bags are found to automatically fill with air after driving long distances, this will without a doubt lead to customer complaints (or hopefully problem reports during the testing phase). In turn, these produce a change request (see Figure 2 on the right), which will probably justify a change. Nevertheless, a – most likely simplistic – cost and benefit analysis has to be done, after which the change request can be approved. Following an analysis of the impact on the car design and production schedules, the

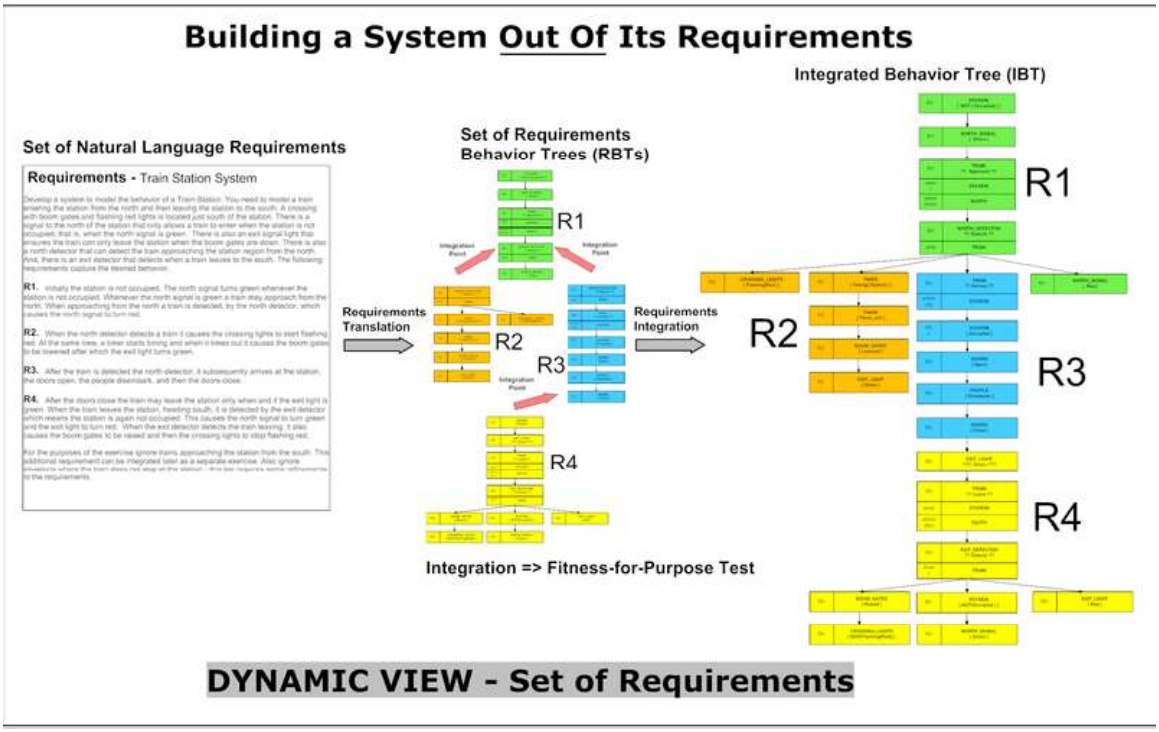
planning for the implementation of the change can be created. According to this planning, the change can actually be realized, after which the new version of the car is hopefully thoroughly tested before it is released to the public.

In industrial plants

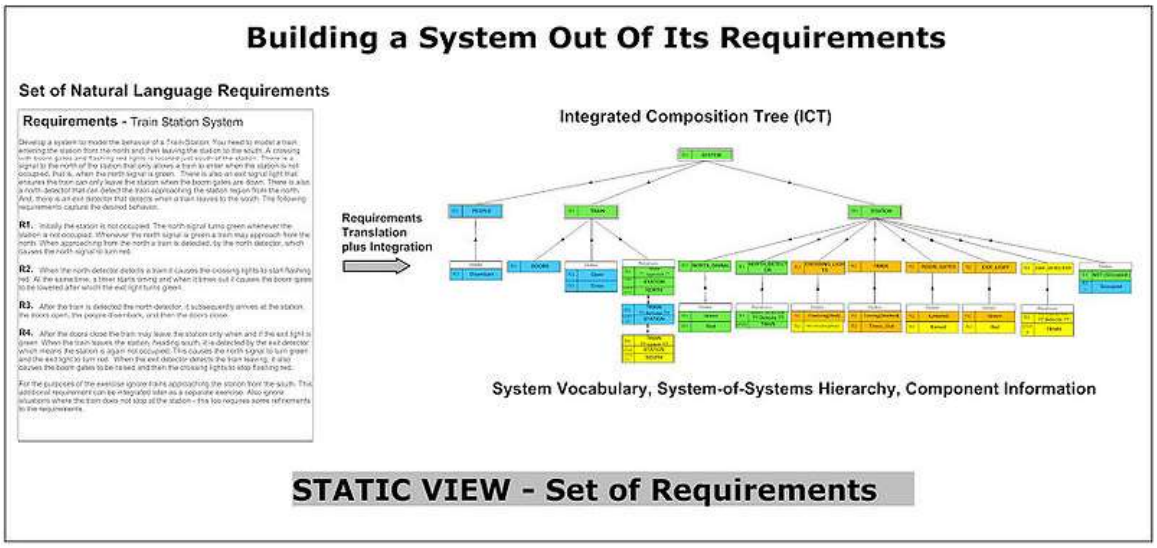
Since complex processes can be very sensitive to even small changes, proper management of change to industrial facilities and processes is recognized as critical to safety. In the US, OSHA has regulations that govern how changes are to be made and documented. The main requirement is that a thorough review of a proposed change be performed by a multi-disciplinary team to ensure that as many possible viewpoints are used to minimize the chances of missing a hazard. In this context, change management is known as Management of Change, or MOC. It is just one of many components of Process Safety Management, section 1910.119(l).1

Chapter- 16

Behavior Trees



Building a System Out of its Requirements - Dynamic View



Building a System Out of its Requirements - Static View

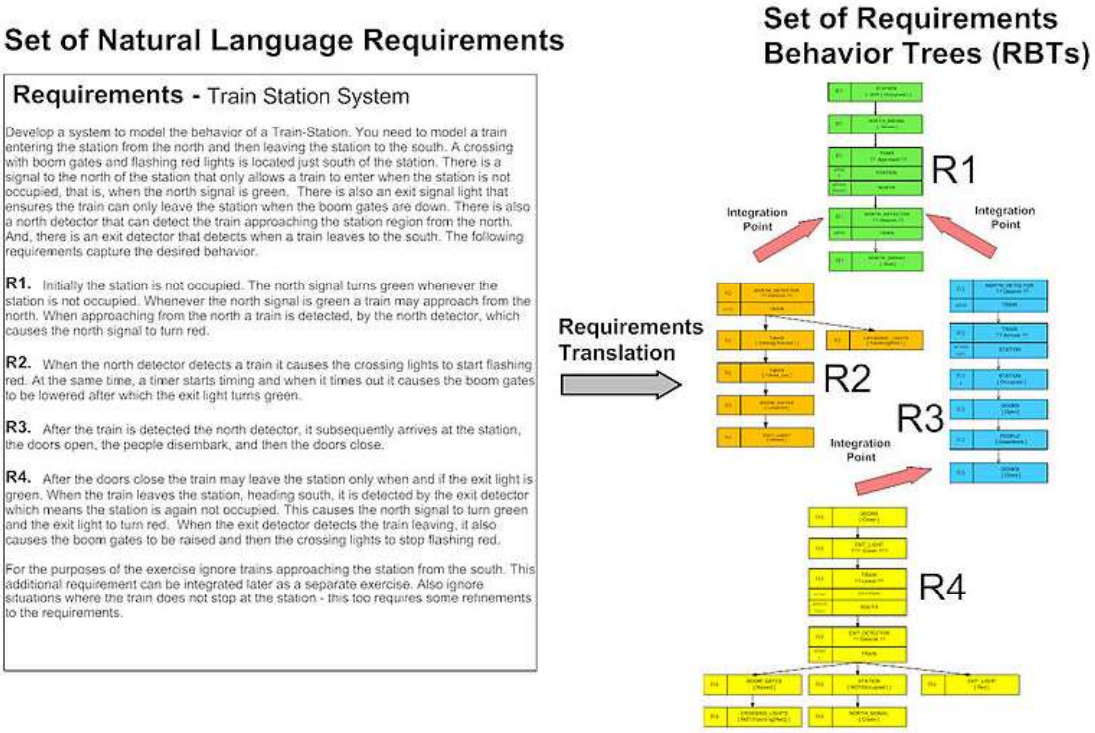
Behavior Trees are a formal, graphical modelling language used primarily in systems and software engineering. Behavior trees employ a well defined notation to unambiguously represent the hundreds or even thousands of natural language requirements that are typically used to express the stakeholder needs for a large-scale software-integrated system.

Overview

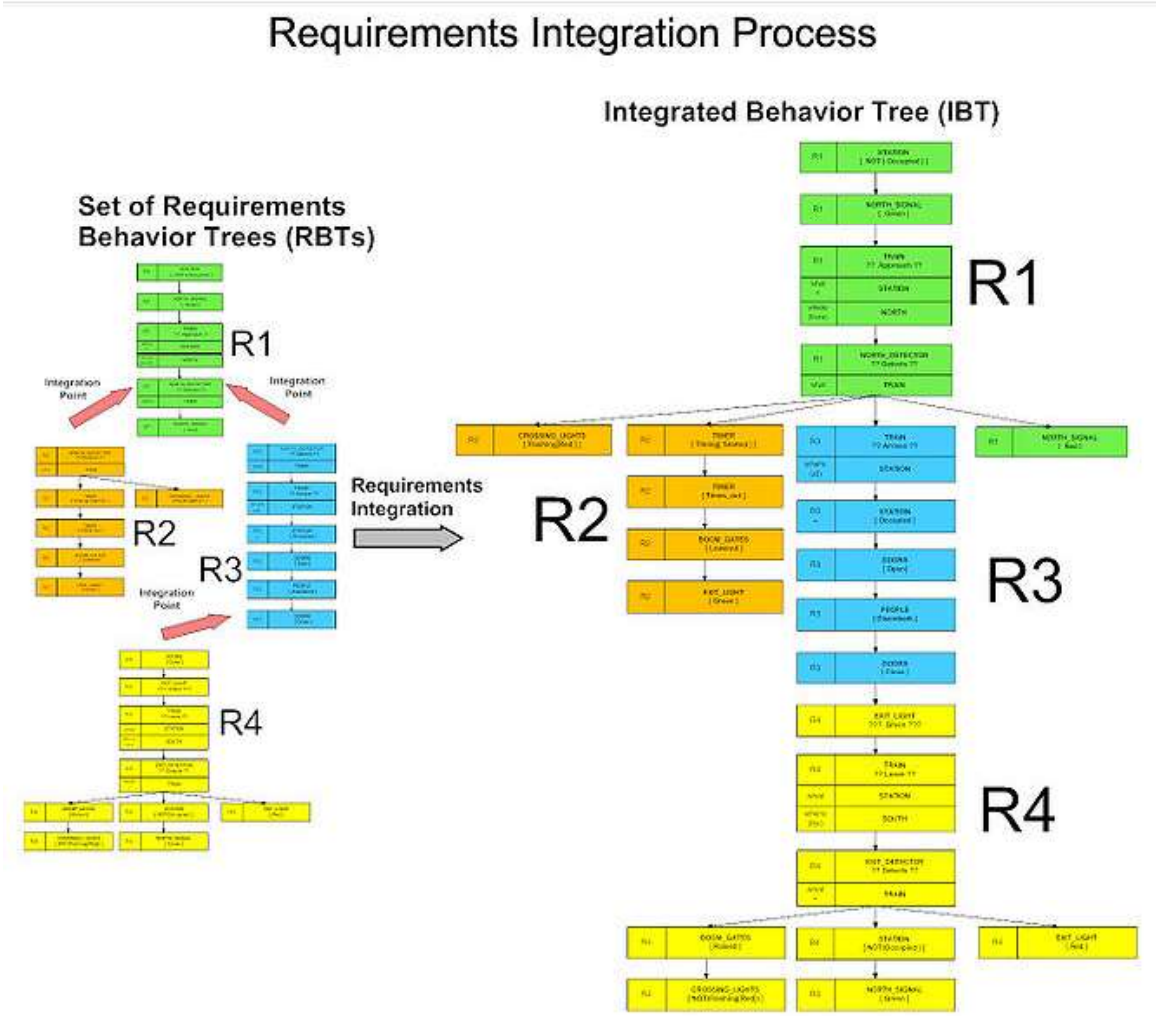
The amount of detail in the large number of natural language requirements for a large-scale system causes short-term memory overload and may create a barrier that prevents anyone from gaining a deep, accurate and holistic understanding of the system needs. Also, because of the use of natural language, there are likely to be many ambiguities, aliases, inconsistencies, redundancies and incompleteness problems associated with the requirements information. This adds further to the uncertainty and complexity. Generally, at best, a few people understand parts of the system or situation well, but no one has other than a superficial understanding of the whole – that is, the detailed integrated behavior of the system.

The Behavior Tree representation, (with the help of the Composition Tree representation that resolves alias and other vocabulary problems with large sets of requirements) allows people to avoid short-term memory overload and produce a deep, accurate, holistic representation of system needs that can be understood by all stakeholders because it strictly uses the vocabulary of the original requirements. Because the Behavior Tree Notation uses a formal semantics, for any given example, it already is, or can be made executable.

Behavior tree forms



Set of four Requirements Behavior Trees.



Requirements Integration Process

Single and composite or integrated behavior tree forms are both important in the application of behavior trees in systems and software engineering.

- Requirement Behavior Trees : Initially, individual requirement behavior trees (RBTs) are used to capture all the fragments of behavior in each individual natural language requirement by a process of rigorous, intent-preserving and vocabulary-preserving translation. The translation process can uncover a range of defects in original natural language requirements.
- Integrated Behavior Tree : Because a set of requirements imply the integrated behavior of a system, all the individual requirement behavior trees can be composed to construct an integrated behavior tree (IBT) that provides a single holistic view of the emergent integrated behavior of the system. This enables the building of the integrated behavior of a system **out of** its requirements. An analogy to help describe this process is the transition from a randomly arranged set of jigsaw puzzle pieces to putting each of the pieces in its appropriate place.

When we do this, we see each piece of information in its intended context and we see the pieces of information as a whole and the emergent properties of the whole.

Having all the requirements converted to behavior trees (RBTs) is similar to having all the pieces for a jigsaw puzzle randomly spread out on a table - until we put all the pieces together we cannot see the emergent picture and whether any pieces are missing or do not fit. Constructing an Integrated Behavior Tree (IBT) allows us to do this.

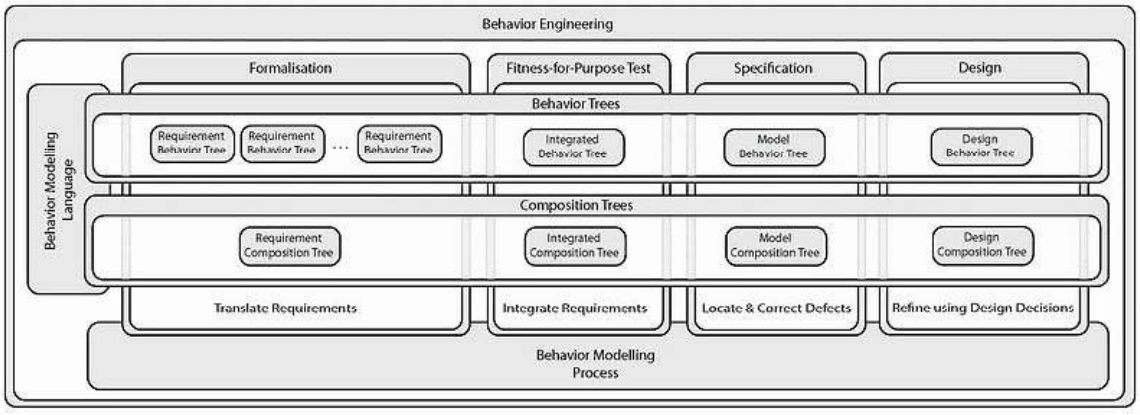
Behavior engineering process

Representation Used - (critical)

- BEHAVIOR TREES provide a vehicle for growing a shared understanding of a complex system.
- The role of the COMPOSITION TREE in the overall process is to provide a vehicle for overcoming the imperfect knowledge associated with the large set of requirements for a system.

Process Used - (critical)

- BEHAVIOR ENGINEERING uses Behavior Trees to control complexity while growing a shared understanding of a complex system.
- That shared, holistic understanding of a complex system, because it integrates the requirements, shows the emergent behavior of the system implied by requirements.



Phases of the Behavior Modelling Process

History

Behavior Trees and the concepts for their application in systems and software engineering were originally developed by Dromey with first publication of some of the key ideas in 2001. Early publications on this work used the terms “genetic software engineering” and “genetic design” to describe the application of behavior trees. The reason for originally using the word genetic was because sets of genes, sets of jigsaw

puzzle pieces and sets of requirements represented as behavior trees all appeared to share several key properties:

- they contained enough information as a set to allow them to be composed – with behavior trees this allows a system to be built out of its requirements
- the order in which the pieces were put together was not important – with requirements this aids coping with complexity
- when all the members of the set were put together the resulting integrated entity exhibited a set of important emergent properties.

For behavior trees important emergent properties include

- the integrated behavior of the system implied by the requirements
- the coherent behavior of each component referred to in the requirements.

These genetic parallels, in another context, were originally spelled by Woolfson, (A. Woolfson, Living Without Genes, Flamingo, 2000)

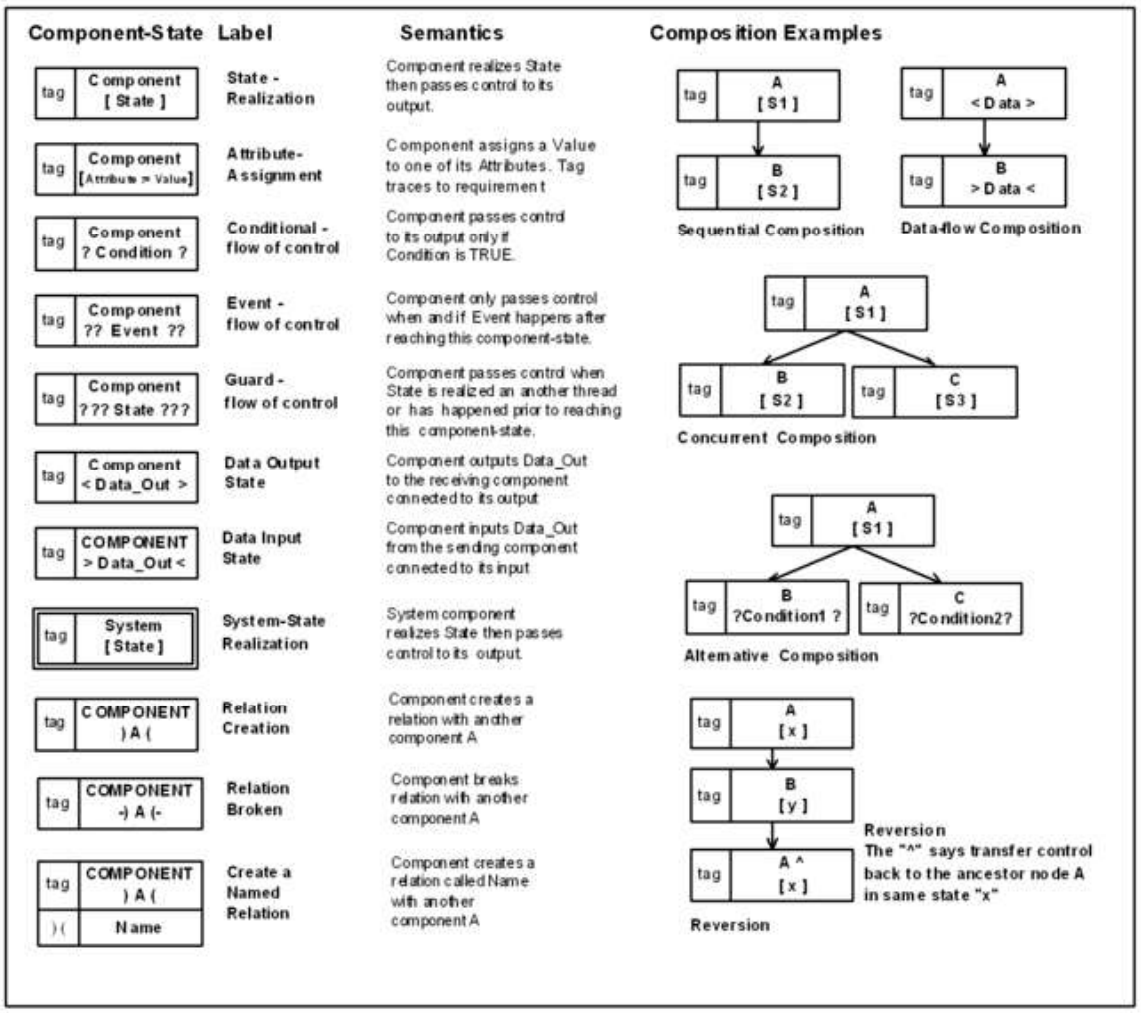
Further weight for use of the term genetic came from eighteenth century thinker Giambattista Vico, who said, “To understand something, and not merely be able to describe it, or analyse it into its component parts, is to understand how it came into being – its genesis, its growth … true understanding is always genetic”. Despite these legitimate genetic parallels it was felt that this emphasis led to confusion with the concept of genetic algorithms. As a result the term Behavior Engineering was introduced to describe the processes that exploit behavior trees to construct systems. The term "behavior engineering" has previously been used in a specialized area of Artificial Intelligence - robotics research. The present use embraces a much broader rigorous formalization and integration of large sets of behavioral and compositional requirements needed to model large-scale systems.

Since the Behavior Tree Notation was originally conceived a number of people from the DCCS (Dependable Complex Computer-based Systems Group – a joint University of Queensland, Griffith University research group) have made important contributions to the evolution and refinement of the notation and to the use of Behavior Trees. Members of this group include: David Carrington, Rob Colvin, Geoff Dromey, Lars Grunske, Ian Hayes, Diana Kirk, Peter Lindsay, Toby Myers, Dan Powell, Cameron Smith, Larry Wen, Nisansala Yatapanage, Kirsten Winter, Saad Zafar, Forest Zheng.

Probabilistic Timed Behavior Trees have recently been developed by Colvin, Grunske and Winter so that reliability, performance and other dependability properties can be expressed.

Key concepts

Behavior tree notation



Core Elements of the Behavior Tree Notation

A behavior tree is used to formally represent the *fragment of behavior* in each individual requirement. Behavior for a large-scale system in general, where concurrency is admitted, appears abstractly as a set of communicating sequential processes. The Behavior Tree Notation captures these composed component-states in a simple tree-like form.

Behavior is expressed in terms of components realizing states and components creating and breaking relations. Using the logic and graphic forms of conventions found in programming languages, components can support actions, composition, events, control-flow, data-flow, and threads.

Traceability tags (see Section 1.2 of Behavior Tree Notation) in behavior tree nodes link the formal representation to the corresponding natural language requirement. Behavior trees accurately capture behavior expressed in the natural language representation of functional requirements. Requirements Behavior Trees strictly use the vocabulary of the natural language requirements but employ graphical forms for behavior composition in order to eliminate risk of ambiguity. By doing this they provide a direct and clearly traceable relationship between what is expressed in the natural language representation and its formal specification.

A basis of the notation is that behavior is always associated with some component. Component-states which represent nodes of behavior are composed sequentially or concurrently to construct a behavior tree that represents the behavior expressed in the natural language requirements. A behavior tree with leaf nodes may revert back (symbolized by adding the caret operator ^) to an ancestor node to repeat behavior, or start a new thread (symbolized by two carets ^^).

A Behavior Tree specifies state changes in components, how data and control is passed between components and how threads interact. There are constructs for creating and breaking relations. There are also constructs for setting and testing states of components as well as mechanisms for inter-process communication that include message passing (events), shared variable blocking and synchronization.

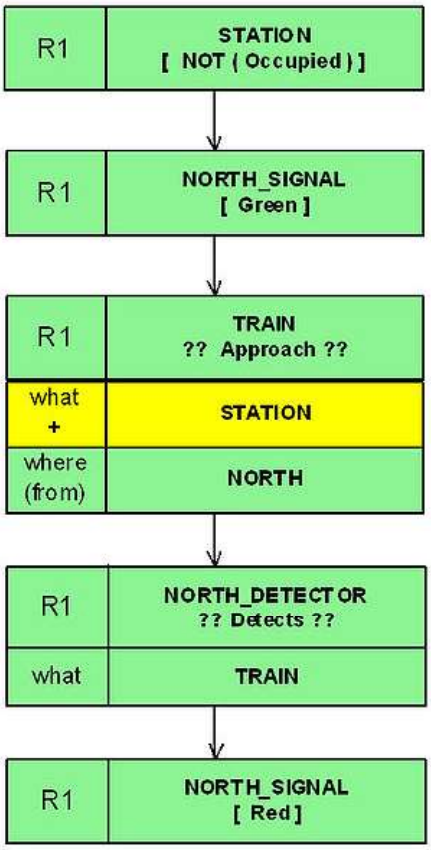
For a complete reference to Behavior Tree notation, version 1.0, see: Behavior Tree Notation v1.0 (2007)

Semantics

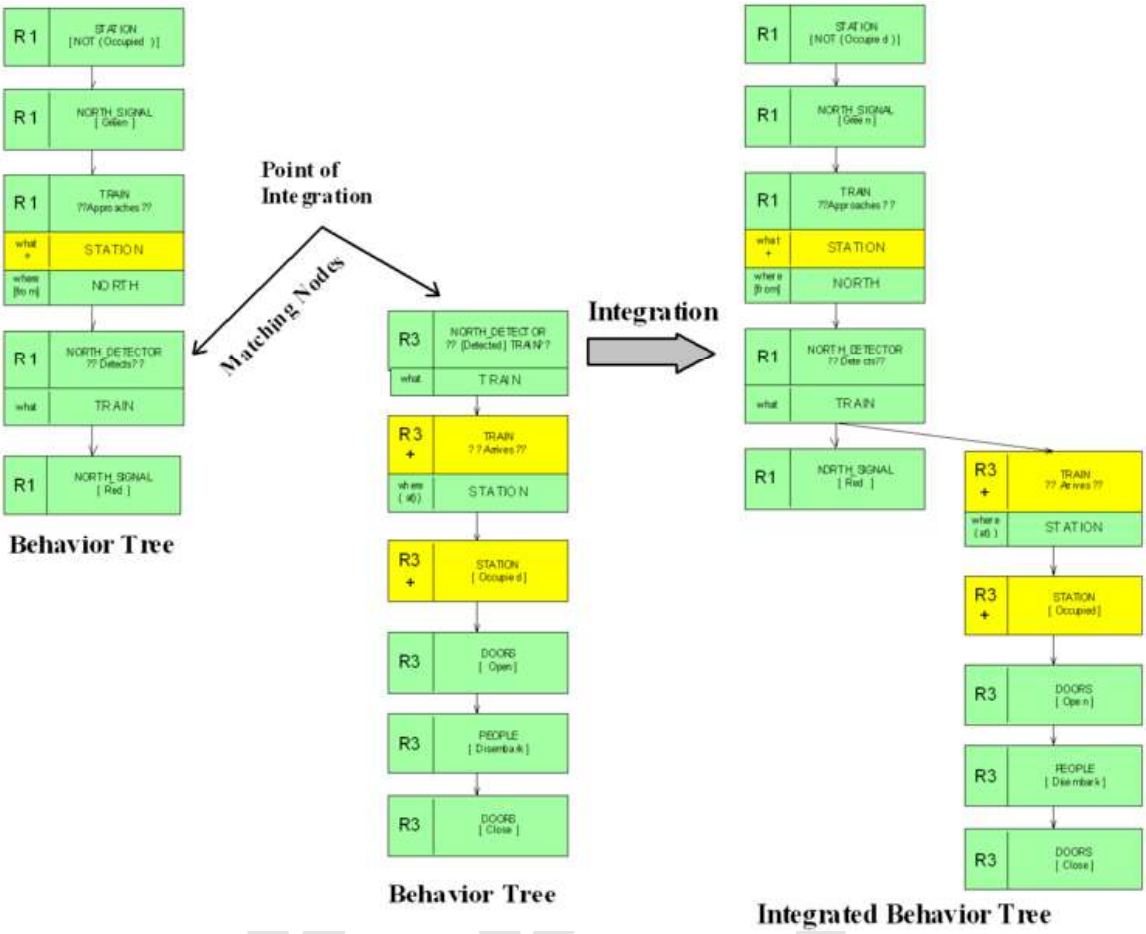
The formal semantics of Behavior Trees is given via a process algebra and its operational semantics. The semantics has been used as the basis for developing simulation, model checking and failure modes and effects analysis.

Requirements translation

Requirement-R1
Initially the **station** is not occupied. The **north signal** turns green *whenever (seq)* the **station** is not occupied. *Whenever* the **north signal** is green *(seq)* a **train** may approach from the north. *When* approaching from the north, *(seq)* a **train** is detected by the **north detector**, which *causes (seq)* the **north_signal** to turn red.



Example Requirement Translation



Requirements Behavior Tree Integration

Requirements translation is the vehicle used to cross the informal-formal barrier. Consider the process of translation for requirement R1 below. The first tasks are to identify the components (**bold**), identify the behaviors (underline) and identify indicators of the order (*italics*) in which behaviors take place. The corresponding behavior tree can then be constructed.

What is clear from the outcome of this process is that apart from pronouns, definite articles etc., essentially all the words in the sentences that contribute to the behavior they describe have been accounted for and used.

Requirements integration

Once the set of requirements are formalized as individual requirement behavior trees, two joint properties of systems and requirements need to be exploited in order to proceed with composing the integrated behavior tree:

- In general, a fragment of behavior expressed by a requirement always has associated with it a precondition which needs to be satisfied before the behavior

can take place (this precondition may or may not be expressed in the requirement).
- If the requirement is really part of the system then some other requirement in the set must establish the precondition needed in (1).

  For requirements represented as behavior trees this amounts to finding where the root node of one tree occurs in some other behavior tree and integrating the two trees at that node.

The example below illustrates requirements integration for two requirements, R1 and R3. In other words, it shows how these two requirements interact.

Operations on integrated behavior trees

Once an integrated behavior tree has been composed, there are a number of important operations that can be performed upon it.

Inspection: defect detection and correction

In general, many defects become much more visible when there is an integrated view of the requirements and each requirement has been placed in the behavior context where it needs to execute. For example, it is much easier to tell whether a set of conditions or events emanating from a node is complete and consistent. The traceability tags also make it easy to refer back to the original natural language requirements. There is also the potential to automate a number of defect and consistency checks on an integrated behavior tree.

When all defects have been corrected and the IBT is logically consistent and complete it becomes a Model Behavior Tree (MBT) which serves as a Formal Specification for the System's behavior that has been constructed out the original requirements. This is the clearly defined stopping point for the analysis phase. With other modelling notations and methods (for instance, with UML) it is less clear-cut when modelling can stop. In some cases, parts of a Model Behavior Tree may need to be transformed to make the specification executable. Once an MBT has been made executable it is possible to carry out a number of other dependability checks.

Simulation

A Model Behavior Tree can be readily simulated in order to explore the dynamic properties of the system. Both a symbolic tool and a graphics tool have been constructed to support these activities.

Model-checking

A translator has been written to convert a Model Behavior Tree into the "Actions Systems" language. This input can then be fed into the SAL Model-checker in order to allow checks to be made as to whether certain safety and security properties are satisfied.

Failure Mode and Effects Analysis (FMEA)

Model-checking has often been applied to system models to check that hazardous states cannot be reached during normal operation of the system. It is possible to combine model-checking with behavior trees to provide automated support for failure mode and effects analysis (FMEA). The advantage of using Behavior Trees for this purpose is that they allow the formal method aspects of the approach to be hidden from non-expert users.

Requirements change

The ideal that is sought when responding to a change in the functional requirements for a system is that it can be quickly determined:

- where to make the change,
- how the change affects the architecture of the existing system,
- which components of the system are affected by the change, and
- what behavioral changes will need to be made to the components (and their interfaces) that are affected by the change of requirements.

Because a system is likely to undergo many sets of changes over its service time, there is also a need to record, manage and optimize the system's evolution driven by the change sequence.

A traceability model, which uses behavior trees as a formal notation to represent functional requirements, reveals change impacts on different types of design constructs (documents) caused by the changes of the requirements. The model introduces the concept of evolutionary design documents that record the change history of the designs. From these documents, any version of a design document as well as the difference between any two versions can be retrieved. An important advantage of this model is that the major part of the procedure to generate these evolutionary design documents can be supported by automated tools.

Code generation and execution

The Behavior Tree representation of the integrated behavior of the system affords several important advantages as an executable model. It clearly separates the tasks of *component integration* from the task of individual *component implementation*. The integrated behavior of the system that emerges from integrating the requirements can be used as a foundation to create a design by applying design decisions. The result is a Design

Behavior Tree (DBT): an executable multithreaded component integration specification that has been built out of the original requirements.

Behavior Tree models are executed in a virtual machine called the Behavior Run-time Environment (BRE). The BRE links together components using middleware, allowing components to be independent programs written in one of several languages that can be executed in a distributed environment. The BRE also contains an expression parser that automatically performs simple operations to minimize the amount of code required to be manually implemented in the component.

The implementation of components is supported by views that are automatically extractable from the DBT. These views provide the component behavior trees (CBTs) of individual components together with the interfaces of individual components. This information, together with the information in the integrated composition tree (ICT) captured about each individual component, provides the information that is needed to implement each individual component.

Several BRE's can be linked together to form complex systems using a system-of-systems construct and the Behavior Engineering Component Integration Environment (BECIE). BECIE is also used to monitor and control the Behavior Tree models being executed within a BRE, similar to supervisory control and data acquisition (SCADA) systems used in industrial process control.

Executable Behavior Trees have been developed for case studies including automated train protection, mobile robots with dynamic object following, an ambulatory infusion pump and traffic light management systems. A version of the BRE suited for embedded systems (eBRE) is also available that has reduced functionality to tailor it to small-footprint microcontrollers.

Applications

Behavior Tree modelling can and has been applied to a diverse range of applications over a number of years. Some of the main application areas are described below.

Large-scale systems

Modeling large-scale systems with large sets of natural language requirements has always been the major focus for trialling Behavior Trees and the overall Behavior Engineering process. Conducting these evaluations and trials of the method has involved work with a number of industry partners and government departments in Australia. The systems studied have included a significant number of defense systems, enterprise systems, transportation systems, information systems, health systems and sophisticated control systems with stringent safety requirements. The results of these studies have all been commercial-in-confidence. However the results of the extensive industry trails with Raytheon Australia are presented below in the Industry Section. What all this work has consistently shown is that by translating requirements and creating dynamic and static

integrated views of requirements a very significant number major defects are discovered early, over and above the defects that are found by current industry best-practice.

Embedded systems

Failure of a design to satisfy a system's requirements can result in schedule and cost overruns. If there are also critical dependability issues, not satisfying system requirements can have life threatening consequences. However in current approaches, ensuring requirements are satisfied is often delayed until late in the development process during a cycle of testing and debugging. This work describes how the system development approach, Behavior Engineering, can be used to develop software for embedded systems. The result is a model-driven development approach that can create embedded system software that satisfies its requirements, as a result of applying the development process.

Hardware-software systems

Many large-scale systems consist of a mixture of co-dependent software and hardware. The different nature of software and hardware means they are often modelled separately using different approaches. This can subsequently lead to integration problems due to incompatible assumptions about hardware/software interactions. These problems can be overcome by integrating Behavior Trees with the Modelica, mathematical modelling approach. The environment and hardware components are modelled using Modelica and integrated with an executable software model that uses Behavior Trees.

Role-based access control

To ensure correct implementation of complex access control requirements, it is important that the validated and verified requirements are effectively integrated with the rest of the system. It is also important that the system can be validated and verified early in the development process. An integrated, role-based access control model has been developed. The model is based on the graphical Behavior Tree notation, and can be validated by simulation, as well as verified using a model checker. Using this model, access control requirements can be integrated with the rest of the system from the outset, because: a single notation is used to express both access control and functional requirements; a systematic and incremental approach to constructing a formal Behavior Tree specification can be adopted; and the specification can be simulated and model checked. The effectiveness of the model has been evaluated using a case study with distributed access control requirements.

Biological systems

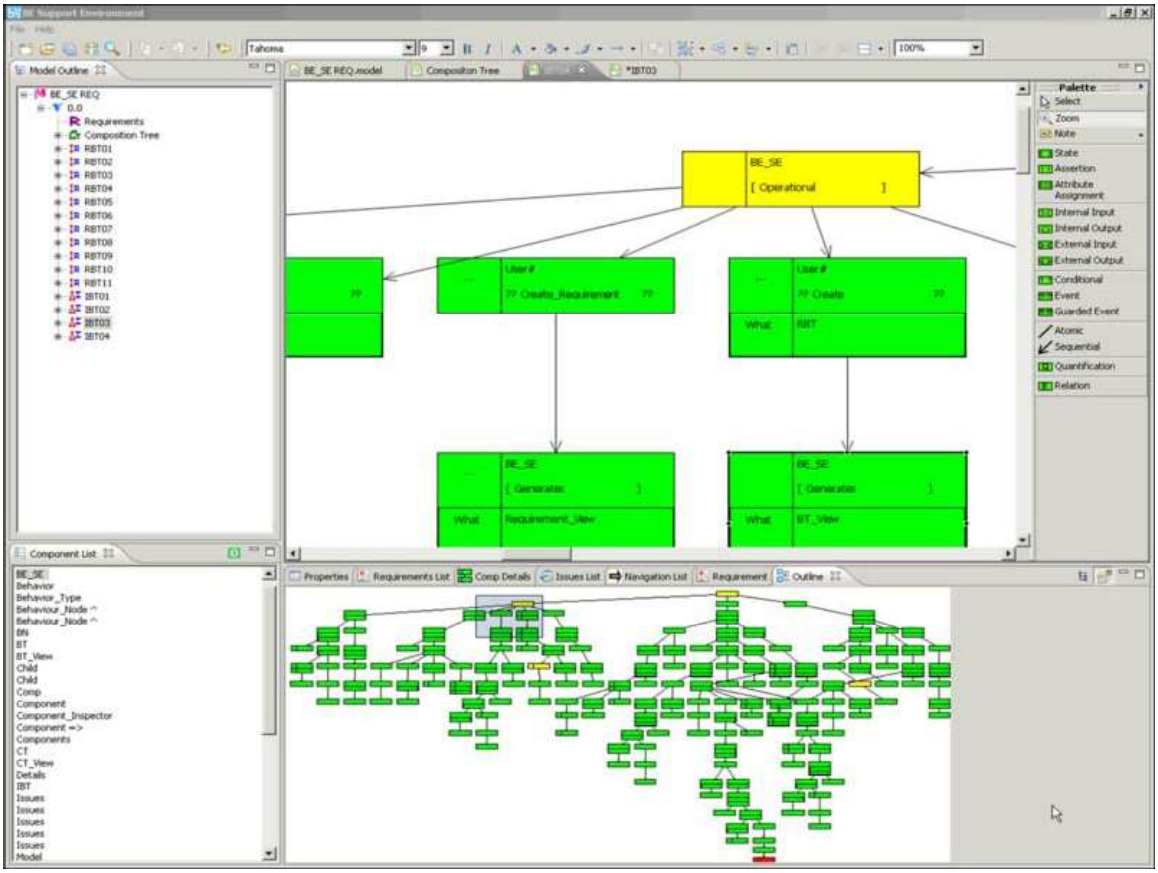
Because Behavior Trees describe complex behavior, they can be used for describing a range of systems not limited to those that are computer-based. In a biological context, BTs can be used to piece together a procedural interpretation of biological functions described in research papers, treating the papers as the requirements documents as

described above. This can help to construct a more concrete description of the process than is possible from reading only, and can also be used as the basis for comparing competing theories in alternative papers. In ongoing research, the Behavior Trees notation is being used to develop models of the brain function in rats under fear conditioning.

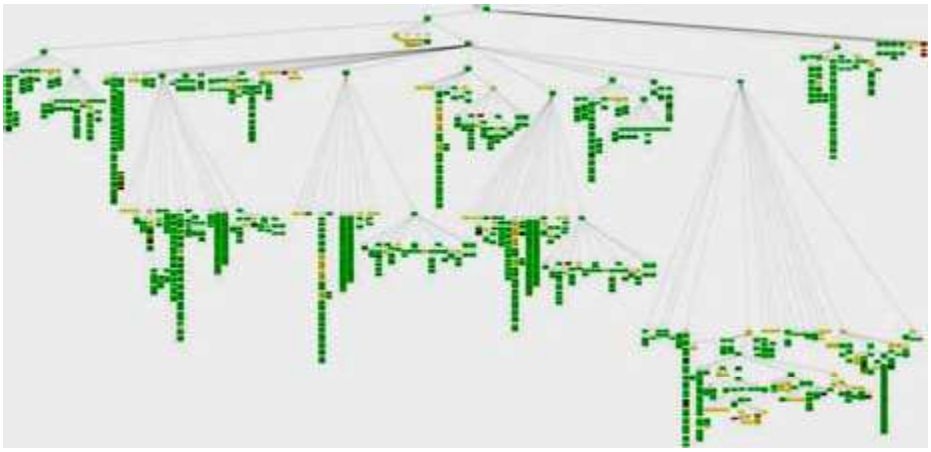
Game A.I Modeling

While BTs have become popular for modeling the Artificial Intelligence in computer games such as Halo and Spore, these types of trees are very different than the ones described on this page, and are closer to a combination of hierarchical finite state machines or hierarchical task network planners. Soccer-player modeling has also been a successful application of BTs..

Scalability and industry applications



Screen-shot of Behavior Engineering Support Environment Tool



Integrated Behavior Tree - Larger System (more than 1000 requirements)

The first industry trials to test the feasibility of the method and refine its capability were conducted in 2002. Over the last three years a number of systematic industry trials on large-scale defence, transportation and enterprise systems have been conducted. This work has established that the method scales to systems with large numbers of requirements but also that it is important to use tool support in order to efficiently navigate and edit such large integrated views of graphical data. Several main results have come out of this work with industry. On average, over a number of projects, 130 confirmed major defects per 1000 requirements have consistently been found after normal reviews and corrections have been made. With less mature requirements sets much higher defect rates have been observed.

Raytheon Australia supports pioneering systems research

An important part of this work with industry has involved applying the analysis part of the method to six large-scale defence projects for Raytheon Australia. They see the method as “a key risk mitigation strategy, of use in both solution development and as a means of advising the customer on problems with acquisition documentation”. An outcome of these industry trials has been the joint development with Raytheon Australia of an industry-strength tool to support the analysis, editing and display of large integrated sets of requirements. More extensive details of industry findings can be found on the Behavior Engineering website.

Dr Terry Stevenson (Chief Technical Officer, Raytheon Australia) and Mr. Jim Boston (Senior Project Manager Raytheon Australia), Mr. Adrian Pitman from the Australian Defence Materiel Organization, Dr Kelvin Ross (CEO, K.J.Ross & Associates) and Christine Cornish (Bushell & Cornish) have provided the special opportunities needed to support this research and to conduct the industry trials and live project work. This work has been supported by the Australian Research Council – ARC Centre for Complex Systems and funds received from industry.

Benefits, advantages

As a behavior modelling representation, Behavior Trees have a number of significant benefits and advantages:

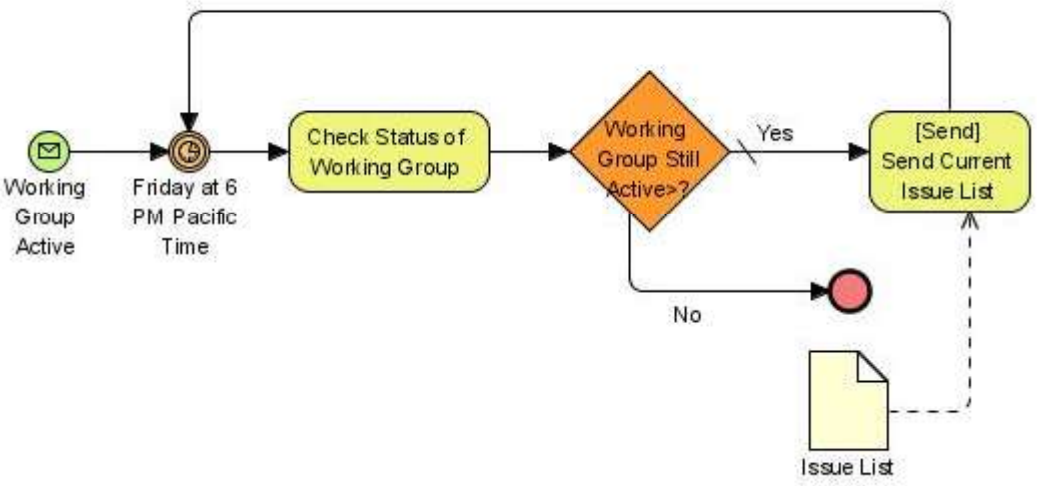
- They employ a well-defined and effective strategy for dealing with requirements complexity, particularly where the initial needs of a system are expressed using hundreds or thousands of requirements written in natural language. This significantly reduces the risk on large-scale projects.
- By rigorously translating then integrating requirements at the earliest possible time they provide a more effective means for uncovering requirements defects than competing methods.
- They employ a single, simple notation for analysis, specification and to represent the behavior design of a system.
- They represent the system behavior as an executable integrated whole.
- They build the behavior of a system out of its functional requirements in a directly traceable way which aids verification and validation.
- They can be understood by stakeholders without the need for formal methods training. By strictly retaining the vocabulary of the original requirements this eases the burden of understanding.
- They have a formal semantics, they support concurrency, they are executable and they can be simulated, model-checked and used to undertake failure mode and effects analysis.
- They can be used equally well to model human processes, to analyse contracts, to represent forensic information, to represent biological systems, and numerous other applications. In each case they deliver the same benefits in terms of managing complexity, and seeing things as a whole. They can also be used for safety critical systems, embedded systems and real-time systems.

Criticisms, disadvantages

- For small textbook level examples, their tree-like nature means that the graphic models produced are sometimes not as compact as Statechart or State Machine behavior specifications.
- The process of requirements translation is very demanding. It is not possible to spend more than three or four hours in a day doing translation even with good tool support.
- Tool support is needed to navigate the very large integrated behavior trees for systems that have hundreds or thousands of requirements.
- For group walkthroughs of very large systems good display facilities are needed.
- There is a need to provide additional sophisticated tool support to fully exploit integrated behavior tree models.

Chapter- 17

Business Process Modeling



Example of business process modeling of a process with a normal flow with the Business Process Modeling Notation.

Business process modeling (BPM) in systems engineering and hardware engineering is the activity of representing processes of an enterprise, so that the current process may be analyzed and improved. BPM is typically performed by business analysts and managers who are seeking to improve process efficiency and quality. The process improvements identified by BPM may or may not require Information Technology involvement, although that is a common driver for the need to model a business process, by creating a process master.

Change management programs are typically involved to put the improved business processes into practice. With advances in technology from large platform vendors, the vision of BPM models becoming fully executable (and capable of simulations and round-trip engineering) is coming closer to reality every day.

History

Techniques to model business process such as the flow chart, functional flow block diagram, control flow diagram, Gantt chart, PERT diagram, and IDEF have emerged since the beginning of the 20th century. The Gantt chart were among the first to arrive around 1999, the flow charts in the 1920s, Functional Flow Block Diagram and PERT in the 1987, Data Flow Diagrams and IDEF in the 1970s. Among the modern methods are Unified Modeling Language and Business Process Modeling Notation. Still these represent just a fraction of the methodologies used over the years to document business processes. The term "business process modeling" itself was coined in the 1960s in the field of systems engineering by S. Williams in his 1967 article "Business Process Modeling Improves Administrative Control". His idea was that techniques for obtaining a better understanding of physical control systems could be used in a similar way for business processes. It took until the 1990s before the term became popular.

In the 1990s the term "process" became a new productivity paradigm. Companies were encouraged to think in *processes* instead of *functions* and *procedures*. Process thinking looks at the chain of events in the company from purchase to supply, from order retrieval to sales etc. The traditional modeling tools were developed to picture time and costs, while modern methods focus on cross-function activities. These cross-functional activities have increased severely in number and importance due to the growth of complexity and dependencies. New methodologies such as business process redesign, business process innovation, business process management, integrated business planning among others all "aiming at improving processes across the traditional functions that comprise a company".

In the field of software engineering the term "business process modeling" opposed the common software process modeling, aiming to focus more on the state of the practice during software development. In that time early 1990s all existing and new modeling techniques to picture business processes were considered and called "business process modeling languages." In the Object Oriented approach, it was considered to be an essential step in the specification of Business Application Systems. Business process modeling became the base of new methodologies, that for example also supported data collection, data flow analysis, process flow diagrams and reporting facilities. Around 1995 the first visually oriented tools for business process modeling and implementation were being presented.

BPM topics

Business model

A business model is a framework for creating economic, social, and/or other forms of value. The term 'business model' is thus used for a broad range of informal and formal descriptions to represent core aspects of a business, including purpose, offerings, strategies, infrastructure, organizational structures, trading practices, and operational processes and policies.

In the most basic sense, a business model is the method of doing business by which a company can sustain itself. That is, generate revenue. The business model spells-out how a company makes money by specifying where it is positioned in the value chain.

Business process

A business process is a collection of related, structured activities or tasks that produce a specific service or product (serve a particular goal) for a particular customer or customers. There are three main types of business processes:

1. Management processes, the processes that govern the operation of a system. Typical management processes include "Corporate Governance" and "Strategic Management".
2. Operational processes, processes that constitute the core business and create the primary value stream. Typical operational processes are Purchasing, Manufacturing, Marketing, and Sales.
3. Supporting processes, which support the core processes. Examples include Accounting, Recruitment, Technical support.

A business process can be decomposed into several sub-processes, which have their own attributes, but also contribute to achieving the goal of the super-process. The analysis of business processes typically includes the mapping of processes and sub-processes down to activity level. A business process model is a model of one or more business processes, and defines the ways in which operations are carried out to accomplish the intended objectives of an organization. Such a model remains an abstraction and depends on the intended use of the model. It can describe the workflow or the integration between business processes. It can be constructed in multiple levels.

A workflow is a depiction of a sequence of operations, declared as work of a person, work of a simple or complex mechanism, work of a group of persons, work of an organization of staff, or machines. Workflow may be seen as any abstraction of real work, segregated in workshare, work split or whatever types of ordering. For control purposes, workflow may be a view on real work under a chosen aspect.

Artifact-centric Business process

The Artifact-centric business process model has emerged as a new promising approach for modeling business processes, as it provides a highly flexible solution to capture operational specifications of business processes. It particularly focuses on describing the data of business processes, known as “artifacts”, by characterizing business-relevant data objects, their lifecycles, and related services. The artifact-centric process modelling approach fosters the automation of the business operations and supports the flexibility of the workflow enactment and evolution

Business process modeling tools

Business process modeling tools provide business users with the ability to model their business processes, implement and execute those models, and refine the models based on as-executed data. As a result, business process modeling tools can provide transparency into business processes, as well as the centralization of corporate business process models and execution metrics.

Modeling and simulation

Modeling and simulation functionality allows for pre-execution "what-if" modeling and simulation. Post-execution optimization is available based on the analysis of actual as-performed metrics.

Business process modeling diagrams are:

- Use case diagrams created by Ivar Jacobson, 1992. Currently integrated in UML
- Activity diagrams, also currently adopted by UML

Some business process modeling techniques are:

- Business Process Modeling Notation (BPMN)
- Cognition enhanced Natural language Information Analysis Method (CogNIAM)
- Extended Business Modeling Language (xBML)
- Event-driven process chain (EPC)
- ICAM DEFinition (IDEF0)
- Unified Modeling Language (UML), extensions for business process such as Eriksson-Penker's
- Riva method using Role Activity Diagrams (RAD)

Programming languages tools for BPM

BPM suite software provides programming interfaces (web services, application program interfaces (APIs)) which allow enterprise applications to be built to leverage the BPM engine. This component is often referenced as the *engine* of the BPM suite.

Programming languages that are being introduced for BPM include: Some standards:

- BPMN
- Business Process Execution Language (BPEL),
- Web Services Choreography Description Language (WS-CDL).
- XML Process Definition Language (XPDL),

Some vendor-specific languages:

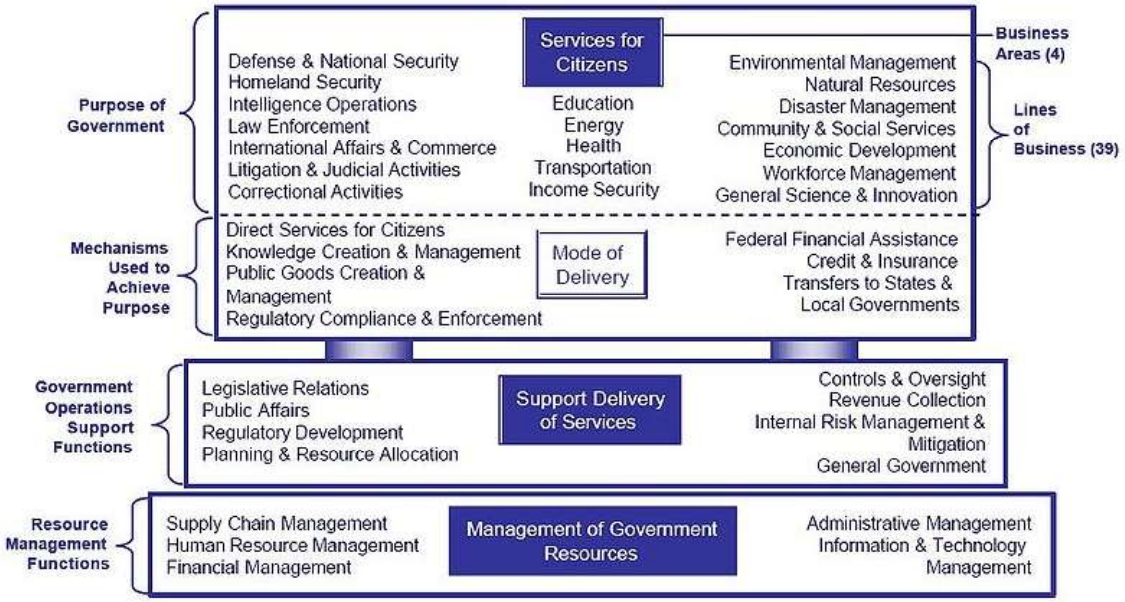
- Architecture of Integrated Information Systems (ARIS) supports EPC,

- Java Process Definition Language (JBPM),

Other technologies related to business process modeling include model-driven architecture and service-oriented architecture.

Related topics

Business reference model

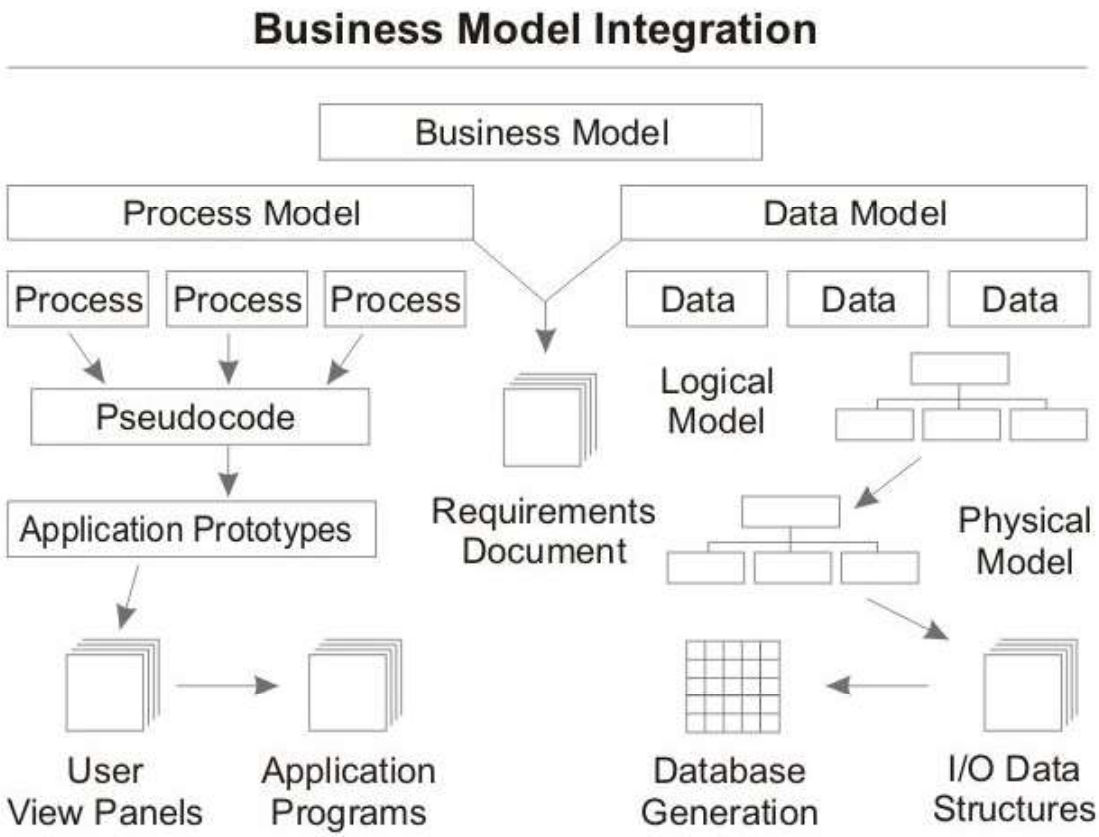


Example of the US Federal Government Business Reference Model.

A business reference model is a reference model, concentrating on the functional and organizational aspects of an enterprise, service organization or government agency. In general a reference model is a model of something that embodies the basic goal or idea of something and can then be looked at as a reference for various purposes. A business reference model is a means to describe the business operations of an organization, independent of the organizational structure that perform them. Other types of business reference model can also depict the relationship between the business processes, business functions, and the business area's business reference model. These reference models can be constructed in layers, and offer a foundation for the analysis of service components, technology, data, and performance.

The most familiar business reference model is the Business Reference Model of the US Federal Government. That model is a function-driven framework for describing the business operations of the Federal Government independent of the agencies that perform them. The Business Reference Model provides an organized, hierarchical construct for describing the day-to-day business operations of the Federal government. While many models exist for describing organizations - organizational charts, location maps, etc. - this model presents the business using a functionally driven approach.

Business process integration



Example of the interaction between business process and data models.

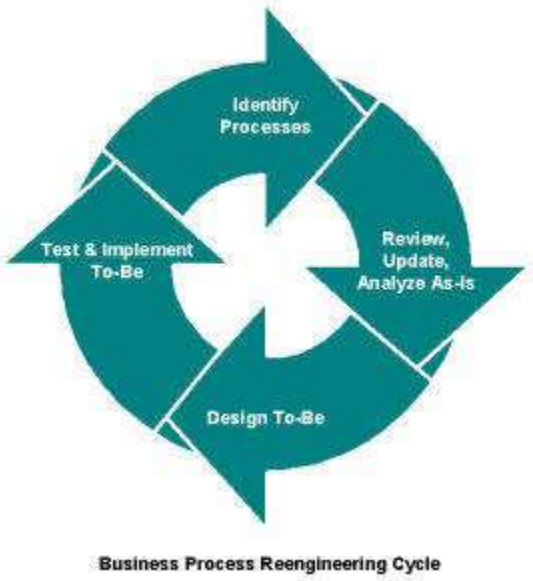
A business model, which may be considered an elaboration of a business process model, typically shows business data and business organizations as well as business processes. By showing business processes and their information flows a business model allows business stakeholders to define, understand, and validate their business enterprise. The data model part of the business model shows how business information is stored, which is useful for developing software code. See the figure on the right for an example of the interaction between business process models and data models.

Usually a business model is created after conducting an interview, which is part of the business analysis process. The interview consists of a facilitator asking a series of questions to extract information about the subject business process. The interviewer is referred to as a facilitator to emphasize that it is the participants, not the facilitator, who provide the business process information. Although the facilitator should have some knowledge of the subject business process, but this is not as important as her mastery of a pragmatic and rigorous method interviewing business experts. The method is important because for most enterprises a team of facilitators is needed to collect information across

the enterprise, and the findings of all the interviewers must be compiled and integrated once completed.

Business models are developed as defining either the current state of the process, in which case the final product is called the "as is" snapshot model, or a concept of what the process should become, resulting in a "to be" model. By comparing and contrasting "as is" and "to be" models the business analysts can determine if the existing business processes and information systems are sound and only need minor modifications, or if reengineering is required to correct problems or improve efficiency. Consequently, business process modeling and subsequent analysis can be used to fundamentally reshape the way an enterprise conducts its operations.

Business process reengineering



Business Process Reengineering Cycle.

Business process reengineering (BPR) is an approach aiming at improvements by means of elevating efficiency and effectiveness of the processes that exist within and across organizations. The key to business process reengineering is for organizations to look at their business processes from a "clean slate" perspective and determine how they can best construct these processes to improve how they conduct business.

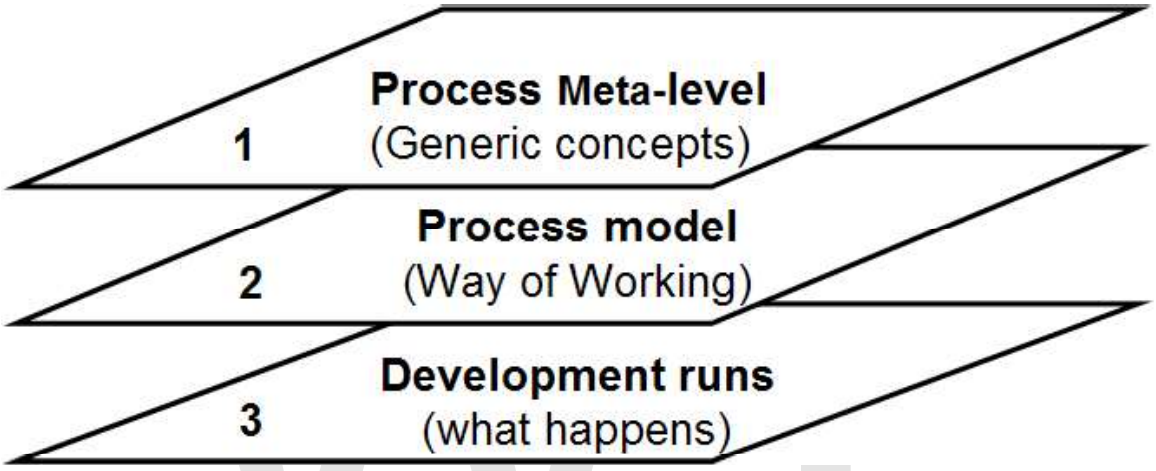
Business process reengineering (BPR) began as a private sector technique to help organizations fundamentally rethink how they do their work in order to dramatically improve customer service, cut operational costs, and become world-class competitors. A key stimulus for reengineering has been the continuing development and deployment of sophisticated information systems and networks. Leading organizations are becoming bolder in using this technology to support innovative business processes, rather than refining current ways of doing work.

Business process management

Business process management is a field of management focused on aligning organizations with the wants and needs of clients. It is a holistic management approach that promotes business effectiveness and efficiency while striving for innovation, flexibility and integration with technology. As organizations strive for attainment of their objectives, business process management attempts to continuously improve processes - the process to define, measure and improve your processes – a "process optimization" process.

Chapter- 18

Meta-Process Modeling



Abstraction level for processes.

Meta-process modeling is a type of metamodeling used in software engineering and systems engineering for the analysis and construction of models applicable and useful to some predefined problems.

Meta-process modeling supports the effort of creating flexible process models. The purpose of process models is to document and communicate processes and to enhance the reuse of processes. Thus, processes can be better taught and executed. Results of using meta-process models are an increased productivity of process engineers and an improved quality of the models they produce.

Overview

Meta-process modeling focuses on and supports the process of constructing process models. Its main concern is to improve process models and to make them evolve, which in turn, will support the development of systems. This is important due to the fact that "processes change with time and so do the Process Models underlying them. Thus, new processes and models may have to be built and existing ones improved". "The focus has been to increase the level of formality of process models in order to make possible their enactment in process-centred software environments". referring to:

A process meta-model is a meta model, "a description at the type level of a process model. A process model is, thus, an instantiation of a process meta-model. [..] A meta-model can be instantiated several times in order to define various process models. A process meta-model is at the meta-type level with respect to a process."
There exist standards for several domains:

- Software Engineering
- Software Process Engineering Metamodel (SPEM) which is defined as a Profile (UML) by the Object Management Group.

Topics in metadata modeling

There are different techniques for constructing process models. "Construction techniques used in the Information Systems area have developed independently of those in Software Engineering. In information systems, construction techniques exploit the notion of a meta-model and the two principal techniques used are those of *instantiation* and *assembly*. In software engineering the main construction technique used today is language-based. However, early techniques in both, information systems and software engineering were based on the experience of process engineers and were, therefore, *ad-hoc* in nature."

Ad-hoc

"Traditional process models are expressions of the experiences of their developers. Since this experience is not formalised and is, consequently, not available as a fund of knowledge, it can be said that these process models are the result of an ad-hoc construction technique. This has two major consequences: it is not possible to know how these process models were generated, and they become dependent on the domain of experience. If process models are to be domain independent and if they are to be rapidly generable and modifiable, then we need to go away from experience based process model construction. Clearly, generation and modifiability relate to the process management policy adopted (see Usage World). Instantiation and assembly, by promoting modularization, facilitate the capitalisation of good practice and the improvement of given process models."

Assembly

The assembly technique is based on the idea of a process repository from which process components can be selected. Rolland 1998 lists two selection strategies:

1. Promoting a *global* analysis of the project on hand based on contingency criteria (Example Van Slooten 1996)
2. Using the notion of descriptors as a means to describe process chunks. This eases the retrieval of components meeting the requirements of the user / matching with the situation at hand.

(Example Plihon 1995 in NATURE () and repository of scenario based approaches accessible on Internet in the CREWS project)

For the assembly technique to be successful, it is necessary that process models are modular. If the assembly technique is combined with the instantiation technique then the meta-model must itself be modular.

Instantiation

For reusing processes a meta-process model identifies "the common, generic features of process models and represents them in a system of concepts. Such a representation has the potential to 'generate' all process models that share these features. This potential is realised when a generation technique is defined whose application results in the desired process model."

Process models are then derived from the process meta-models through *instantiation*. Rolland associates a number of advantages with the instantiation approach:

1. The exploitation of the meta-model helps to define a wide range of process models.
2. It makes the activity of defining process models systematic and versatile.
3. It forces to look for and introduce, in the process meta-model, generic solutions to problems and this makes the derived process models inherit the solution characteristics.

"The instantiation technique has been used, for example, in NATURE, Rolland 1993, Rolland 1994, and Rolland 1996. The process engineer must define the instances of contexts and relationships that comprise the process model of interest."

Language

Rolland 1998 lists numerous languages for expressing process models used by the software engineering community:

- E3
- Various Prolog dialects for EPOS , Oikos , and PEACE
- PS-Algol for PWI)

as well as further computational paradigms:

- Petri nets in EPOS and SPADE
- Rule based paradigm in MERLIN
- ALF
- Marvel
- EPOS
- Triggers in ADELE and MVP-L).

Languages are typically related to process programs whereas instantiation techniques have been used to construct process scripts.

Tool support

The Meta-modeling process is often supported through software tools, called CAME tools (Computer Aided Method Engineering) or Meta-CASE tools (Computer Assisted Software Engineering tools on a Meta-level). Often the instantiation technique "has been utilised to build the repository of Computer Aided Method Engineering environments" (referring to).

Example tools for meta-process modeling are:

- Maestro II
- MetaEdit+
- Mentor

Example: "Multi-model view"

Colette Rolland (1999) provides an example of a meta-process model which utilizes the instantiation and assembly technique. In the paper the approach is called "Multi-model view" and was applied on the CREWS-L'Ecritoire method. The CREWS-L'Ecritoire method represents a methodical approach for Requirements Engineering, "the part of the IS development that involves investigating problems and requirements of the users community and developing a specification of the future system, the so-called conceptual schema.".

Besides the CREWS-L'Ecritoire approach, the multi-model view has served as a basis for representing :

> (a) the three other requirements engineering approaches developed within the CREWS project, Real World Scenes approach , SAVRE approach for scenario exceptions discovery , and the scenario animation approach
> (b) for integrating approaches one with the other and with the OOSE approach

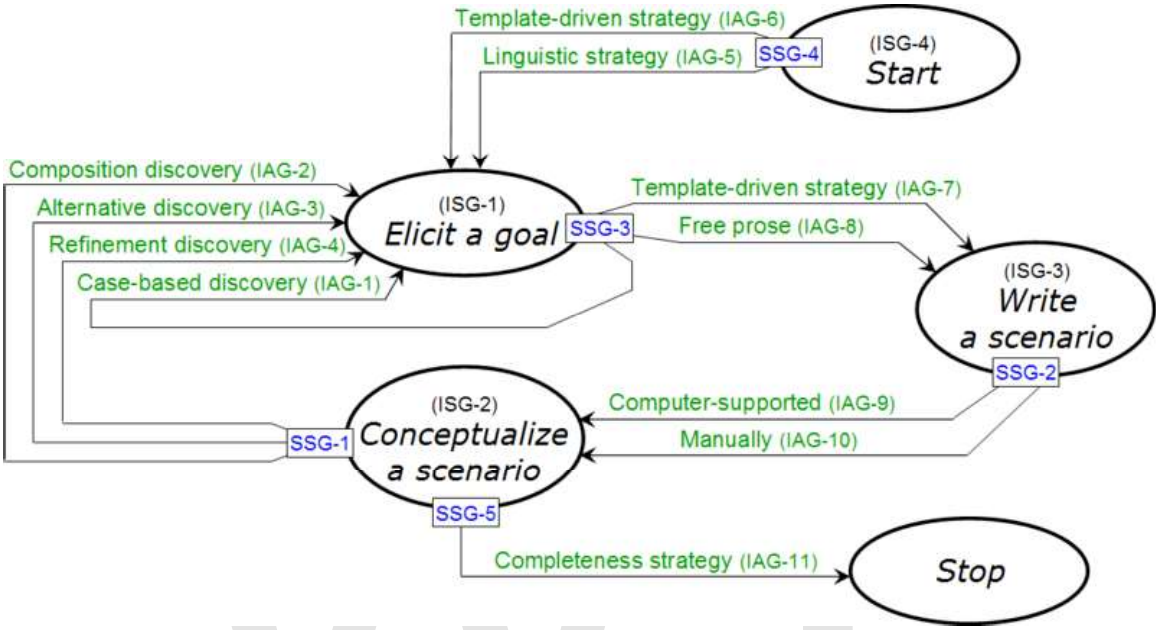
Furthermore, the CREWS-L'Ecritoire utilizes Process Models and Meta-Process Models in order to achieve flexibility for the situation at hand. The approach is based on the notion of a labelled graph of intentions and strategies called a *map* as well as its associated *guidelines* . Together, map (process model) and the guidelines form the method. The main source of this explanation is the elaboration of Colette Rolland in .

Process model / Map

The map is "a navigational structure which supports the dynamic selection of the intention to be achieved next and the appropriate strategy to achieve it"; it is "a process model in which a nondeterministic ordering of intentions and strategies has been

included. It is a labelled directed graph with intentions as nodes and strategies as edges between intentions. The directed nature of the graph shows which intentions can follow which one."

The map of the CREWS-L'Ecritoire method looks as follow:



Process model of the CREWS-L'Ecritoire method

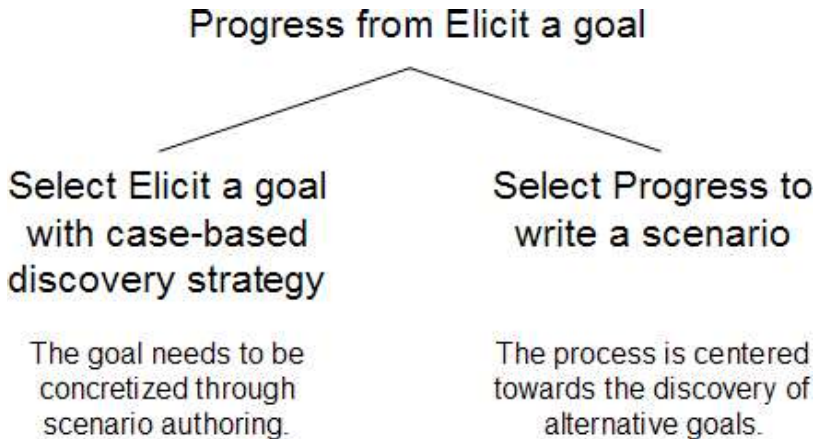
The map consists of goals / *intentions* (marked with ovals) which are connected by *strategies* (symbolized through arrows). An *intention* is a goal, an objective that the application engineer has in mind at a given point of time. A *strategy* is an approach, a manner to achieve an intention. The connection of two goals with a strategy is also called *section*.

A map "allows the application engineer to determine a path from Start intention to Stop intention. The map contains a finite number of paths, each of them prescribing a way to develop the product, i.e. each of them is a process model. Therefore the map is a multi-model. It embodies several process models, providing a multi-model view for modeling a class of processes. None of the finite set of models included in the map is recommended 'a priori'. Instead the approach suggests a dynamic construction of the actual path by navigating in the map. In this sense the approach is sensitive to the specific situations as they arise in the process. The next intention and strategy to achieve it are selected dynamically by the application engineer among the several possible ones offered by the map. Furthermore, the approach is meant to allow the dynamic adjunction of a path in the map, i.e. adding a new strategy or a new section in the actual course of the process. In such a case guidelines that make available all choices open to handle a given situation are of great convenience. The map is associated to such guidelines" .

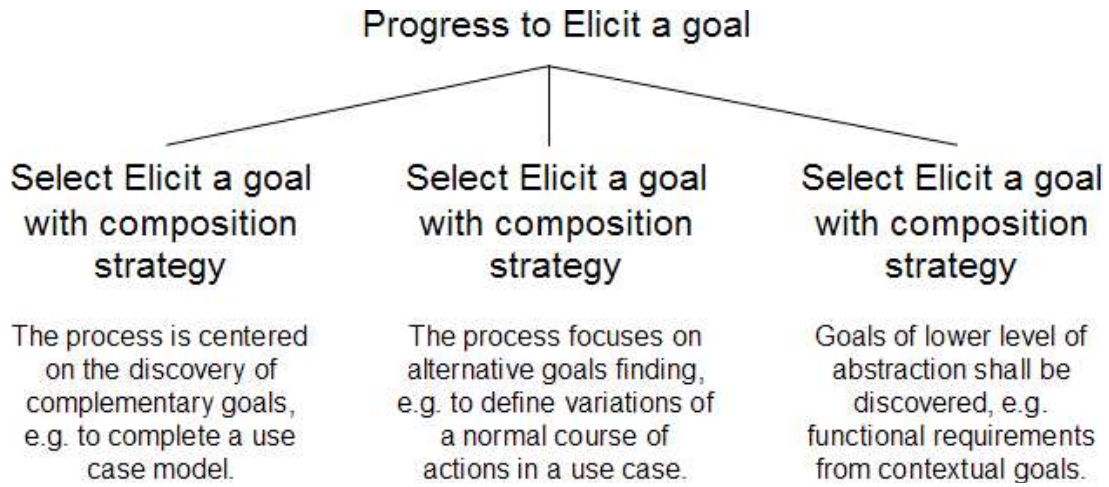
Guidelines

A guideline "helps in the operationalisation of the selected intention" ; it is "a set of indications on how to proceed to achieve an objective or perform an activity." The description of the guidelines is based on the NATURE project's contextual approach and its corresponding enactment mechanism. Three types of guidelines can be distinguished:

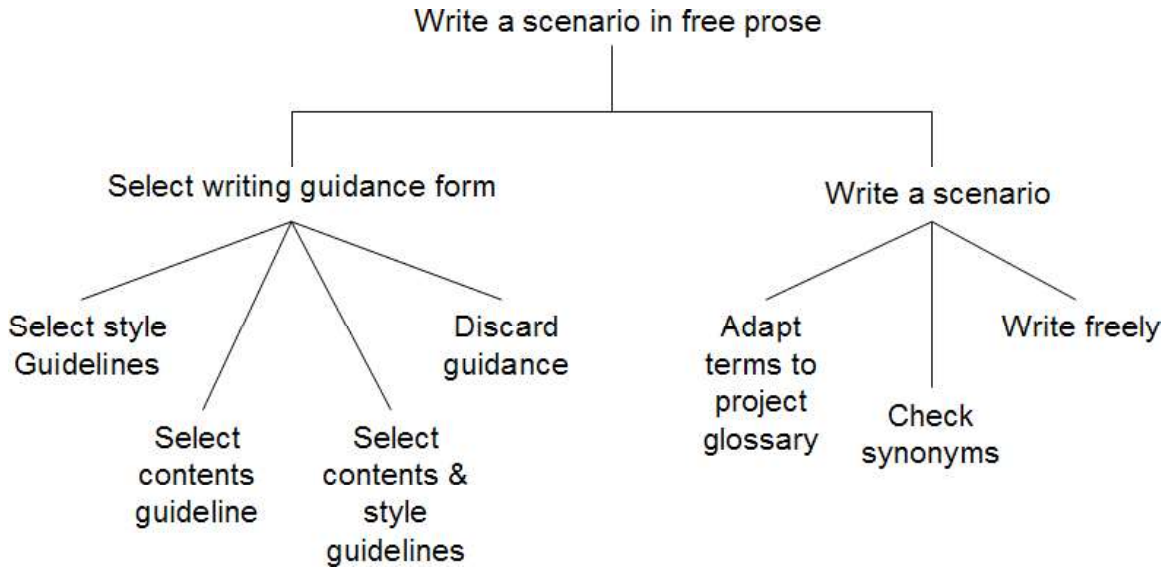
- *Intention Selection Guidelines (ISG)* identify the set of intentions that can be achieved in the next step and selects the corresponding set of either IAGs (only one choice for an intention) or SSGs (several possible intentions).
- *Strategy Selection Guidelines (SSG)* guide the selection of a strategy, thereby leading to the selection of the corresponding IAG.
- *Intention Achievement Guidelines (IAG)* aim at supporting the application engineer in the achievement of an intention according to a strategy, are concerned with the tactics to implement these strategies, might offer several tactics, and thus may contain alternative operational ways to fulfil the intention.



Example of an Intention Selection Guideline 1 (ISG-1)



Example of an Strategy Selection Guideline 1 (SSG-1)



Example of an Intention Achievement Guideline 8 (IAG-8)

In our case, the following guidelines – which correspond with the map displayed above – need to be defined:

Intention Selection Guidelines (ISG)

1. ISG-1 Progress from Elicit a goal
2. ISG-2 Progress from Conceptualize a Scenario
3. ISG-3 Progress from Write a scenario
4. ISG-4 Progress from Start

Strategy Selection Guidelines (SSG)

1. SSG-1 Progress to Elicit a goal
2. SSG-2 Progress to Conceptualize a Scenario
3. SSG-3 Progress to Write a scenario
4. SSG-4 Progress to Elicit a goal
5. SSG-5 Progress to Stop

Intention Achievement Guidelines (IAG)

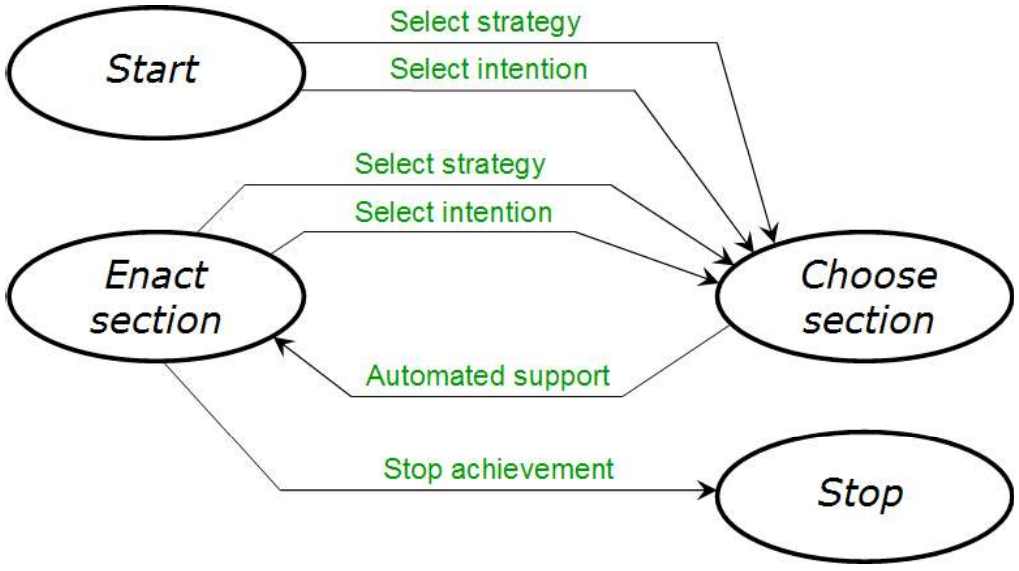
1. IAG-1 Elicit a goal with case-based strategy
2. IAG-2 Elicit a goal with composition strategy
3. IAG-3 Elicit a goal with alternative strategy
4. IAG-4 Elicit a goal with refinement strategy
5. IAG-5 Elicit a goal with linguistic strategy
6. IAG-6 Elicit a goal with template-driven strategy
7. IAG-7 Write a scenario with template-driven strategy

8. IAG-8 Write a scenario in free prose
9. IAG-9 Conceptualize a Scenario with computer support strategy
10. IAG-10 Conceptualize a Scenario manually
11. IAG-11 Stop with completeness strategy

The following graph displays the details for the Intention Achievement Guideline 8 (IAG-8).

Meta-process map

In the multi-model view as presented in the paper of C. Rolland, the meta-process (the instance of the meta-process model) is "a process for the generation of a path from the map and its instantaneous enactment for the application at hand." While the meta-process model can be represented in many different ways, a map was chosen again as a means to do so. It is not to be mixed up with the map for the process model as presented above.



Meta-process model of the CREWS-L'Ecritoire method

Colette Rolland describes the meta-model as follow : (Meta-intentions are in bold, meta-strategies in italic – in green in the map).

"The **Start** meta-intention starts the construction of a process by selecting a section in the method map which has map intention Start as source. The **Choose Section** meta-intention results in the selection of a method map section. The **Enact Section** meta-intention causes the execution of the method map section resulting from **Choose Section**. Finally, the **Stop** meta-intention stops the construction of the application process. This happens when the **Enact Section** meta-intention leads to the enactment of the method map section having Stop as the target. As already explained in the previous sections, there are two ways in which a section of a method map can be selected, namely by selecting an

intention or by selecting a strategy. Therefore, the meta-intention **Choose Section** has two meta-strategies associated with it, *select intention* and *select strategy* respectively. Once a method map section has been selected by **Choose Section**, the IAG to support its enactment must be retrieved; this is represented in [the graph] by associating the meta-strategy *automated support* with the meta-intention, **Enact Section**."

Sample process

The sample process "Eliciting requirements of a Recycling Machine" is about a method for designing the requirements of recycling facilities. The recycling facilities are meant for customers of a supermarket. The adequate method is obtained though instantiation of the meta-process model on the process model.

The following table displays the stepwise trace of the process to elicit requirements for the recycling machine (from):

Step	Guideline	Meta-process	Process	Product (Goal = Gxx)
1.1	SSG-4	Choose section with select strategy	SSG4 suggests two strategies. The template-driven strategy is chosen because it is the most appropriate way to become familiar with the goal formalisation proposed by the CREWS-L'Ecritoire method	
1.2	IAG-6	Enact section with automated support	IAG6 displays a goal statement template and explains the meaning of each parameter. The requirement Engineer (RE) chooses a loose statement having only a verb and a target	G1: Provideverb (Recycling Facilities*) target *RF
2.1	ISG-1	Choose section with select intention	ISG1 provides RE with arguments to advise him on choosing one of the two possible intentions from **'Elicit a Goal'**, namely to **'Elicit a Goal or to '**Write a Scenario'**. The former is selected so as to generate alternative design solutions**	

2.2	IAG-1	Enact section with automated support	IAG1 uses the goal statement structure and parameter values supplied to generate alternative goals. This leads to 21 alternative goals to G1 which are ORed to G1. After discussion with stakeholders, G4 is selected	G2: Provide bottle RF to our customers with a card-based machine; G3: Provide paper RF to our customers with a card-based machine; G4: Provide bottle and box RR to our customers with a card-based machine; . . . G22: Provide bottle RF to all customers with money return machine
3.1	SSG-3	Choose section with select strategy	SSG3 offers two strategies from which the template-driven strategy is chosen. This is because there is uncertainty about what a scenario should be. The templates lead to some certainty	
3.2	IAG-7	Enact section with automated support	IAG7 proposes a template to be filled in. The template corresponds to a service scenario and contains actions that express services expected from the system	SC4: If the customer gets a card, he recycles objects
4.1	SSG-2	Choose section with select strategy	SSG2 offers two strategies to conceptualise a Scenario. Among the two strategies, manual and computer based, the former is chosen since the service scenario (SC4) is very simple and can be handled manually	
4.2	IAG-10	Enact section with automated support	IAG10 suggests two things: (1) to avoid anaphoric references such as he, she, etc. (2) to express atomic actions in an explicit ordering (3) to avoid ambiguities The scenario is rewritten accordingly	SC4: 1. The customer gets a card; 2. The customer recycles boxes and bottles

5.1	SSG-1	Choose section with select strategy	The RE knows that he wants to analyse the scenario SC4 to discover a new goal. Thus, he knows the target intention 'Elicit a Goal' and SSG1 is displayed. SSG1 offers three strategies to discover new goals from scenario analysis. The refinement strategy, is chosen because there is a need to discover the functional requirements of recycling machine	
5.2	IAG-4	Enact section with automated support	IAG4 guides in transforming actions of the service scenario SC4 into goals which express functional requirements. Two goals are generated and related together to G4 with an AND relationship. G24 is selected for further processing	G23: Get card from supermarket; G24: Recycle bottles and boxes from RM
6.1	SSG-3	Choose section with select strategy	The RE knows his target intention, namely 'Write a Scenario'. Thus SSG3 is displayed to help the RE in selecting the right strategy. The free prose strategy is selected because the text is likely to be long and the free prose facilitates this	
6.2	IAG-8	Enact section with automated support	IAG8 provides style and contents guidelines adapted to the type of scenario at hand, namely system interaction scenario	SC24-1: The customer inserts his card in the RM. The RM checks if the card is valid and then a prompt is given. The customer inputs the bottles and/or boxes in the RM. If the objects are not blocked, the RM ejects the card and prints a receipt
7.1	SSG-2	Choose	SSG2 is displayed. The	

		section with select strategy	automated support strategy is selected to take advantage of the powerful linguistic devices and obtain a scenario formulation which will be the basis for automated reasoning	
7.2	IAG-9	Enact section with automated support	IAG9 semi-automatically transforms the initial prose into a structured text whose semantics conform to the scenario model. The transformation includes disambiguation, completion and mapping onto the linguistic structures associated to the concepts of the scenario model. SC24-2 is the result of the transformation of SC24-1. (Underlined statements result of the transformation)	SC24-2: 1. The customer inserts the customer card in the RM, 2. The RM checks if the card is valid, 3. If the card is valid, 4. A prompt is given to the customer, 5. The customer inputs the bottles and the boxes in the RM, 6. The RM checks if the bottles and the boxes are not blocked, 7. If the bottles and the boxes are not blocked, 8. The RM ejects the card to the customer, 9. The RM prints a receipt to the customer
8.1	SSG-1	Choose section with select strategy	Of the three strategies proposed by SSG1, the alternative discovery strategy is chosen. This strategy suits the need to investigate variations and exceptions of the normal course of actions described in SC242	
8.2	IAG-3	Enact section with automated support	IAG3 proposes several tactics to discover alternative goals to G24. The one based on the analysis of conditions in the scenario is selected. This leads to discover G25 and G26	G25: Recycle box and bottles from RM with invalid card; G26: Recycle box and bottles with a deblocking phase