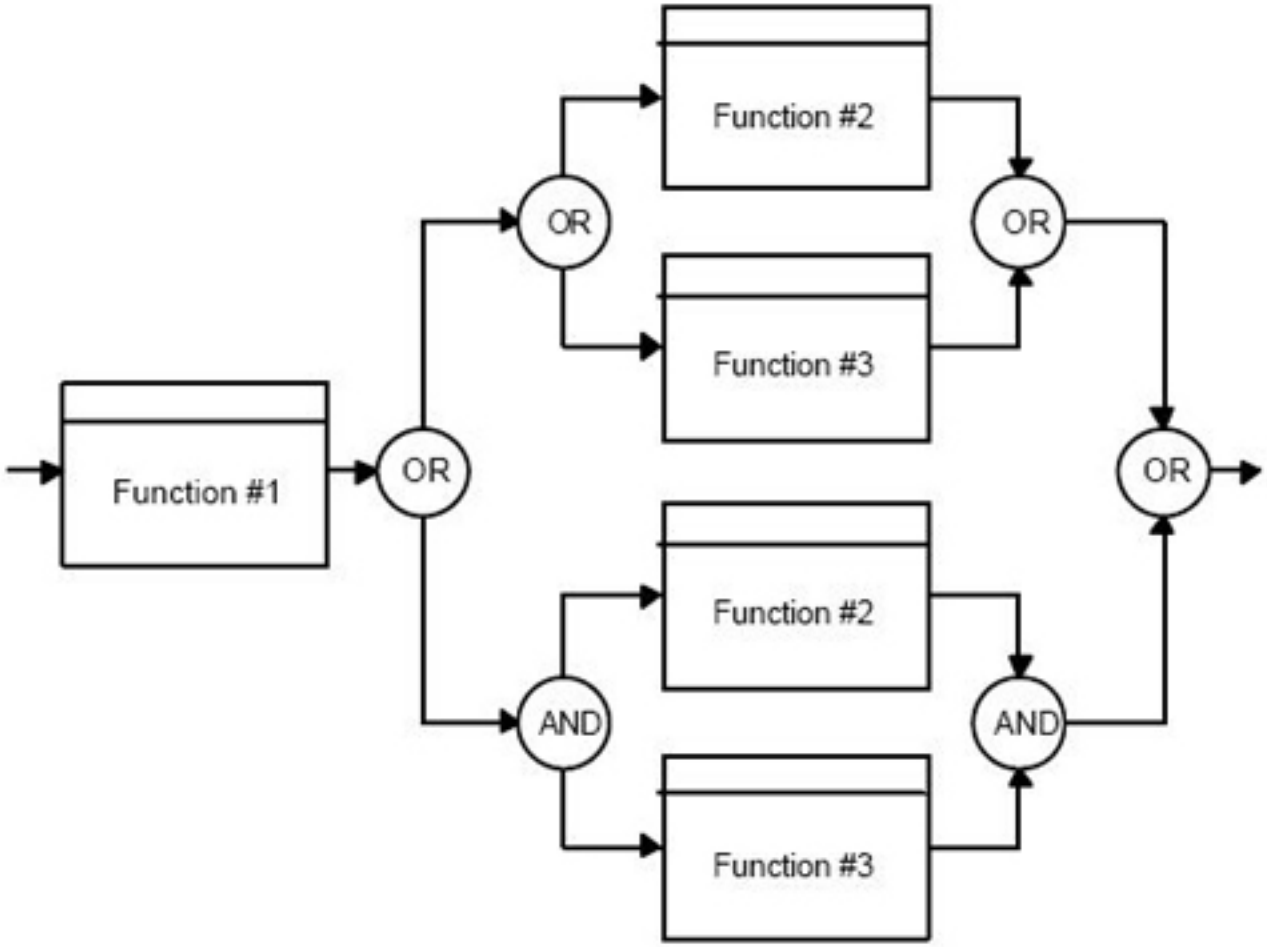
Systems
Analysis and Engineering



Irvin Crenshaw

First Edition, 2012

ISBN 978-81-323-0992-5

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

Table of Contents

Chapter 1

Event Partitioning & Data Flow Diagram

Event Partitioning

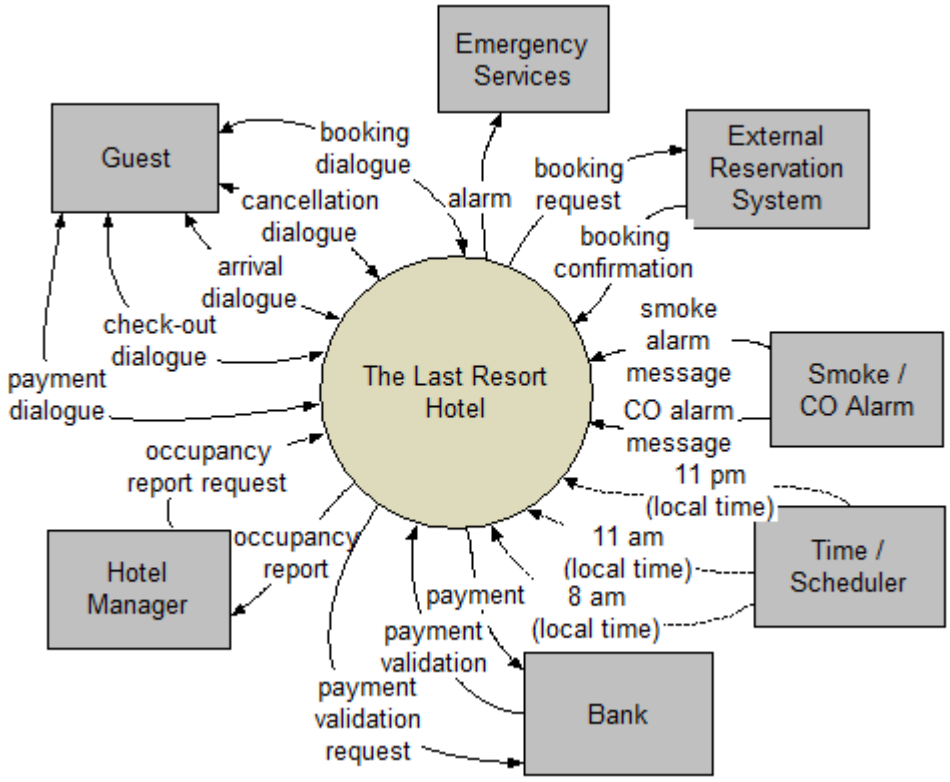
The goal of **event partitioning** is to be an easy-to-apply systems analysis technique that helps the analyst organise requirements for large systems into a collection of smaller, simpler, minimally-connected, easier-to-understand ‘mini systems’ / use cases. The approach is explained by Stephen M. McMenamin and John F. Palmer in *Essential Systems Analysis* . A brief version of the approach is described in the article on Data Flow Diagrams. A more complete discussion is in Edward Yourdon's *Just Enough Structured Analysis* . The description focuses on using the technique to create data flow diagrams, but it can be used to identify use cases as well.

The premise of event partitioning is that systems exist to respond to external events: identify what happens in the business environment that requires planned responses, then define and build systems to respond according to the rules of the business. In particular, a business system exists to service the requests of customers. A customer, in the jargon of the UML, is an ‘actor’.

Actor → Event → Detect → Respond

The method has the following steps.

- **1.** Identify the external systems by brainstorming a list of the ‘**actors**’ (external systems), which are the sources of external events. If you find a graphic to be helpful, create a context diagram showing the actors outside of the system under study and the flows/signals between them.



System context diagram for a fictitious hotel. (By convention, bidirectional flows, with arrows at both ends, are often used when a dialogue is initiated externally. For example, "booking dialogue" contains the flow "booking request", which is the initial trigger; "booking confirmation", the result, is sent back.)

- **2.** Putting oneself in the shoes of an 'actor' (or working with actor representatives), brainstorm a list of the '**external events**' / 'triggers' that they want the system to have a planned response to. (Note that the system cannot originate *external* events; only an actor can.)
- **3.** Identify what will enable the system to **detect** the external events:
  - the arrival of one or more pieces of **data** (possibly in the form of a message)
  - the arrival of one or more points in **time**
- **4.** Identify the **planned response(s)** that the system may carry out when the events occur. It's the response(s)/use case(s) that will enable the system to achieve its goals.

The technique was extended with 'non-event' events by Paul T. Ward and Stephen J. Mellor in *Structured Development for Real-Time Systems: Essential Modeling Techniques* .

> “Since the terminators [actors] are by definition outside the bounds of the system-building effort represented by the model, the implementors cannot modify the terminator [actor] technology at will to improve its reliability. Instead, they must build responses to terminator [actor] problems into the essential model of the system. A useful approach to modeling responses to terminator [actor] problems is to build a list of ‘normal’ events and then to ask, for each event, ‘Does the system need to respond if this event *fails to* occur as expected?’ ” [emphasis added] ”

Identifying Requirements and Their Reasons

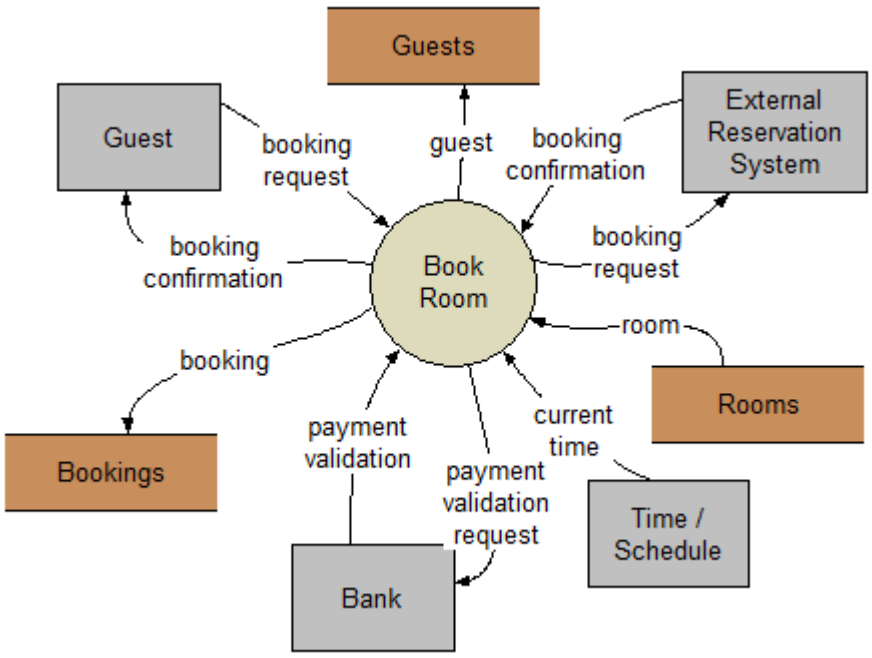
The event-response information may be captured in a table. The event is the *raison d’être* for the response, which gives ‘traceability’ from the response back to the environment.

1. Actor	2. External Event / Trigger	3. Detected by	4. Response(s) / Use Case(s)
Guest	Guest requests a room of a certain type, for a particular arrival date, departure date, at a certain rate, etc.	booking request + (payment validation) + (*external reservation system* booking confirmation)	Book Room (may include guaranteed booking, alternate hotel booking, waitlisted booking)
Guest	Guest asks to cancel room booking.	cancellation request	Cancel Booking
Guest	Guest arrives at hotel.	arrival message = * * = [guest name ; booking reference]	Check in Guest
Time / Scheduler	Guest *fails to* arrive at hotel. [This is a ‘non-event’ event.]	11 pm (local time) [A ‘non-event’ event is detected by the arrival of a point in time, a deadline.]	Create Guest Bill, Update Booking
Guest	Guest asks to check out of hotel.	check-out request = * * = [guest name ; room number]	Create Guest Bill, Update Room Occupancy
Time / Scheduler	Guest *fails to* check out of hotel. [This is a ‘non-event’ event.]	11 am (local time) [A ‘non-event’ event is detected by the arrival of a point in time, a deadline.]	Create Guest Bill

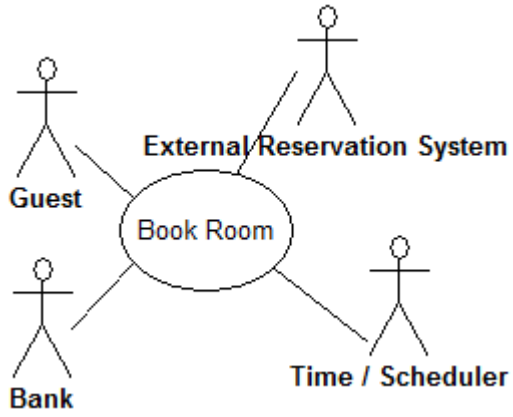
Guest	Guest offers payment of bill.	payment vehicle = * * = [cash ; cheque ; credit card ; debit card] + (guest id)	Accept Guest Payment
Time / Scheduler	Time to prepare Room Occupancy Report for previous night.	8 am (local time)	Report on Room Occupancy
Hotel Manager	Hotel Manager requests Room Occupancy Report.	occupancy report request	Report on Room Occupancy
Smoke / CO Alarm	Alarm detects smoke.	smoke alarm message	Report Smoke Alarm
Smoke / CO Alarm	Alarm detects CO (carbon monoxide).	CO alarm message	Report CO Alarm

Defining requirements

This approach helps the analyst to decompose the system into 'mentally bite-sized' mini-systems using events that require a planned response. The level of detail of each response is at the level of 'primary use cases'. Each planned response may be modelled using DFD notation or as a single use case using use case diagram notation.



Single process in a fictitious hotel using data flow diagram notation.



Single use case in a fictitious hotel using use case diagram notation.

The *basic flow* within a process or use case can usually be described in a relatively small number of steps, often fewer than twenty or thirty, possibly using something like 'structured English'. Ideally, all of the steps would be visible all at once (often a page or less). The intention is to reduce one of the risks associated with short-term memory, namely, forgetting what is not immediately visible ('out of sight, out of mind').

Alternatively, using the notations of structured techniques, an analyst could create a 'Nassi–Shneiderman diagram'. In the UML, the use case could be modelled using an activity diagram, a sequence diagram, or a communication diagram. This could be problematic if there are many complex scenarios of the use case; the analyst may wish to model all or most of the scenarios.

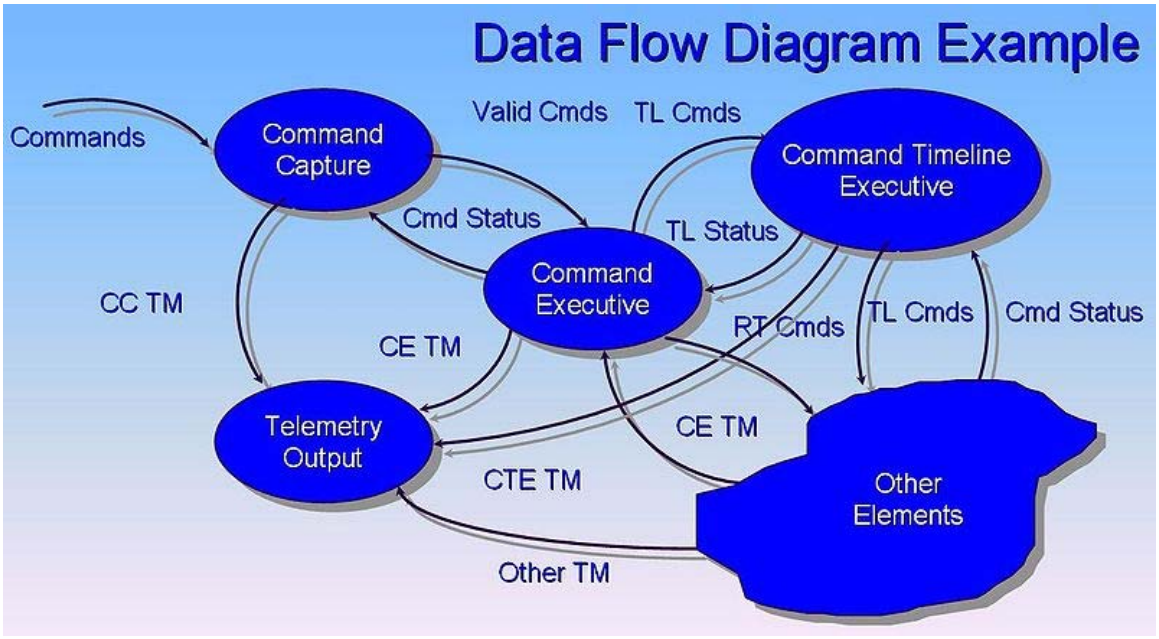
Complexity versus fragmentation

If the response is lengthy or complex (i.e., more than a page of text), an analyst may decompose ('factor out' or deduplicate) into smaller 'secondary use cases' to keep the 'parent' primary use case smaller and simpler. These secondary use cases may prove to be reusable as well. (In a UML use case diagram, they would be drawn as extended or included use cases, which are related to one or more primary use cases.)

While describing a use case, an analyst may also uncover 'business rules'. Some analysts suggest capturing business rules in a separate document using the Object Constraint Language or some other formal notation. Then when a business rule must be obeyed in a use case, the analyst makes reference to it. This minimises repetition within a specification, but risks fragmentation of a specification. One technique that may reduce this tension is to use hyperlinks in the specification document.

In addition to functional requirements captured in a use case description, an analyst may include such non-functional requirements as response time, learnability, etc.

Data Flow Diagram



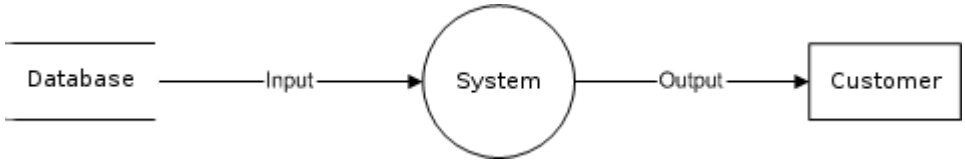
Data flow diagram example.

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).

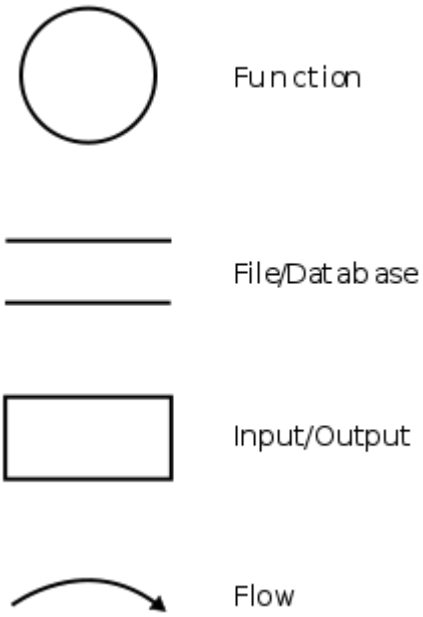
On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

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.

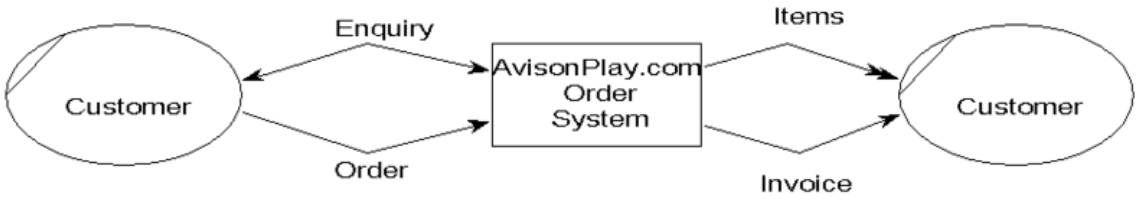
In the course of developing a set of *levelled* data flow diagrams the analyst/designers is forced to address how the system may be decomposed into component sub-systems, and to identify the transaction data in the data model.

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.

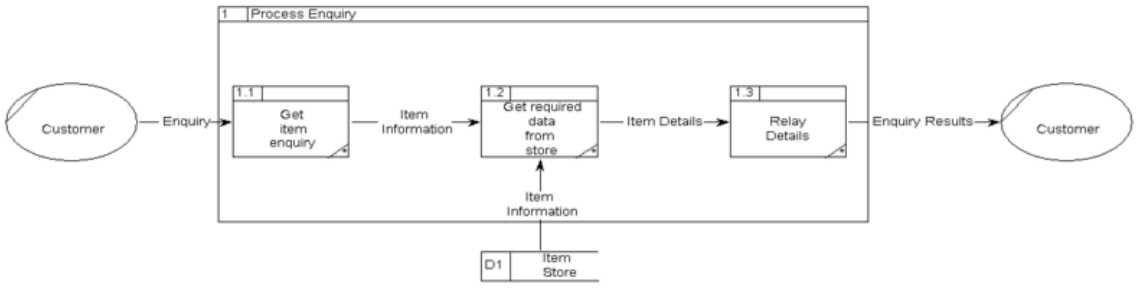
Number the Processes As a convenient way of referencing the processes in a DFD, most systems analysts number each bubble. It doesn't matter very much how you go about doing this — left to right, top to bottom, or any other convenient pattern will do -- as long as you are consistent in how you apply the numbers. The only thing that you must keep in mind is that the numbering scheme will imply, to some casual readers of your DFD, a certain sequence of execution. That is, when you show the DFD to a user, he may ask, "Oh, does this mean that bubble 1 is performed first, and then bubble 2, and then bubble 3?" Indeed, you may get the same question from other systems analysts and programmers; anyone who is familiar with a flowchart may make the mistake of assuming that numbers attached to bubbles imply a sequence. This is not the case at all. The DFD model is a network of communicating, asynchronous processes, which is, in fact, an accurate representation of the way most systems actually operate. Some sequence may be implied by the presence or absence of data (e.g., it may turn out that bubble 2

cannot carry out its function until it receives data from bubble 1), but the numbering scheme has nothing to do with this. So why do we number the bubbles at all? Partly, as indicated above, as a convenient way of referring to the processes; it's much easier in a lively discussion about a DFD to say "bubble 1" rather than "EDIT TRANSACTION AND REPORT ERRORS." But more importantly, the numbers become the basis for a hierarchical numbering scheme when we introduce leveled dataflow diagrams in Section 9.3.

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 interrelation. A process 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. So why do we number the bubbles at all? Partly, as indicated above, as a convenient way of referring to the processes; it's much easier in a lively discussion about a DFD to say "bubble 1" rather than "EDIT TRANSACTION AND REPORT ERRORS."

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 children 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.

Chapter 2

Control-Feedback-Abort Loop

Too often **systems** fail, sometimes leading to significant loss of life, fortunes and confidence in the provider of a product or service. It was determined that a simple and useful tool was needed to help in the analysis of interactions of groups and systems to determine possible unexpected consequences. The tool didn't need to provide every possible outcome of the interactions but needed to provide a means for analysts and product/service development stakeholders to evaluate the potential risks associated with implementing new functionality in a system. They needed a brainstorming tool to help ascertain if a concept was viable from a business perspective. The ***Control-Feedback-Abort Loop*** and the analysis diagram is one such tool that has helped organizations analyze their system workflows and workflow exceptions.

The concept of the Control-Feedback-Abort (CFA) loop is based upon another concept called the '*Control – Feedback Loop'*. The Control – Feedback Loop has been around for many years and was the key concept in the development of many electronic designs such as *Phase-Lock Loops*. The core of the CFA loop concept was based on a major need that corporate executives and staff can anticipate the operation of systems, processes, products and services they use and create before they are developed.

History of CFA Loop concept

The concept of the CFA loop was developed by T. James LeDoux, 'Jim', a Senior Consultant and software QA / test expert and owner of Alpha Group 3 LLC, a test management consulting company. In 1986, Mr. LeDoux, with assistance from Mr. Warren Yates, a former engineer from General Dynamics, Inc., found that using a Control and Feedback concept for analyzing group and system dynamics was not providing them with the full picture when systems were going out of control. In 1996, Jim LeDoux and Dr. Larry W. Smith, Ph.D., president of Remote Testing Services, Inc., discussed the issue at length and came to the conclusion that some other form of control must be present when a system goes out of control, even if the control is unintended.

In 1997, Mr. LeDoux used the change of behavior a person exhibits when driving a car at the time a police car pulls in behind them to describe how a change of control occurs. He demonstrated this phenomenon at a 2003 Product Development and Management Association (PDMA) meeting in Denver by showing the action of the first control (traffic, signs and speed) being aborted by the driver and a second control (police car,

signs and speed) becoming the primary control. In 2004, Mr. LeDoux worked with Dr. Susan Wheeler, Ed. D., a former Instructional Design Consultant with Nims, Inc. and the present Director of Technology Services at Illinois Central College, to identify the range of uses for the CFA Loop. The CFA Loop is now being used to analyze system activities in several Fortune 100 companies. A discussion of its use is also included in the management book "Takeoff!: The Introduction to Project Management Book that Will Make Your Projects Take Off and Fly!" by Dr. Dan Price, D.M. ISBN 9780970746115

It was found that strong similarities existed between the concept of '*Control Charts*' and the CFA Loop. The difference in the two concepts was that control charting is used as a dynamic measurement of present conditions. The CFA Loop is used to analyze how a closed-loop system is supposed to work and what are the expectations when alternate controls take over by either intent or accident. A comparison of the CFA Loop and its relationship to Control Charts are presented in a later section of this discussion.

The Control-Feedback concept

The Control – Feedback concept consisted of a 'Control' that gave information on the way the component was to perform and then adjustments to the control's present operation based upon the feedback. It used a concept called 'Sampling' to determine how often the 'Control' used the 'Feedback' information so that the 'Control' could modify instructions to the component.

What is the CFA Loop

Figure 1 shows a model of the CFA loop. The CFA loop consists of three main elements - The Control element, the Feedback element and the Abort element. Within any system, the lack of any one of these three elements will result in the system failing at some point in time. The term 'system' used in this document can represent any environment, task, process, procedure or system in a physical, organizational or natural structure where an entity will respond to influences. It has been found, through experience, that even trees appear to follow the CFA model. The diagram in Figure 1 can be used as an analysis diagram by inserting functions of the controls, feedbacks and aborts in each of the related circles defining the system being analyzed. (Example: Control - Workflow requests, Feedback - Results of requests, Aborts - Requests that failed, workflow exception path)

The CFA model can be used effectively with 3-sigma Control Charts. CFA loops and Control Charts share the same functionality, which will be discussed later in this document.

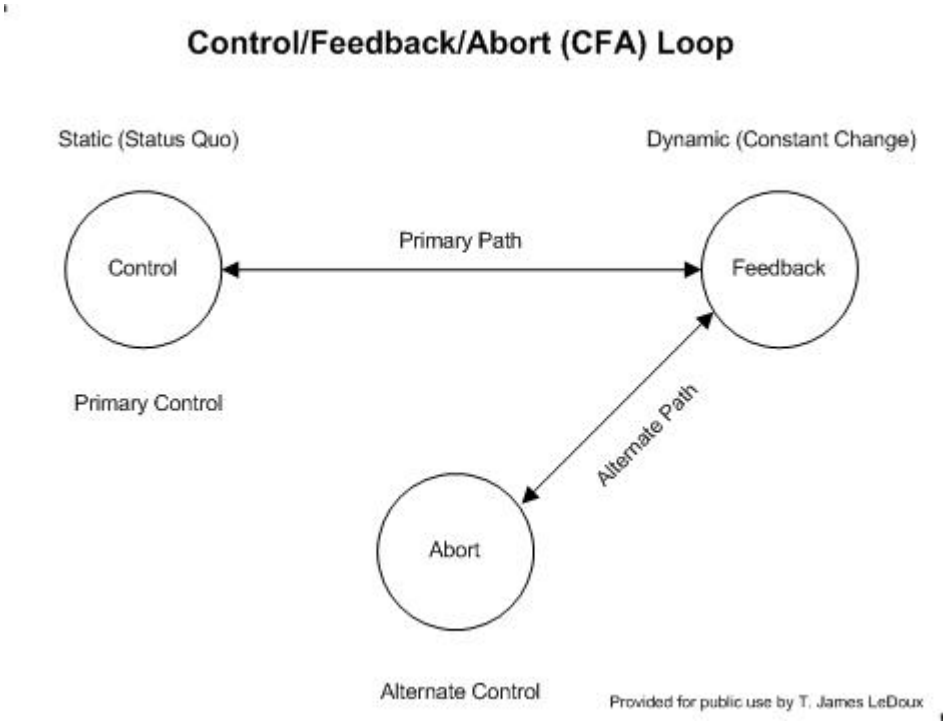


Figure 1 – The CFA Loop

A description of the Control-Feedback-Abort (CFA) Loop

As mentioned, the CFA Loop consists of three elements – Control, Feedback and Abort. First, we will discuss the Control element of the loop.

The control element

The Control element of the CFA loop, as highlighted in Figure 2, controls the activity of the system in question. A basic characteristic of the Control element is that it is always in a static state until it receives new information from the feedback. This static state is, in reality, the Control element holding the system in a status quo condition. Using an automobile as an example, if the previous instruction provided by the Control to the auto was to accelerate, it would continue to accelerate until a feedback reading would indicate to the Control that the Control should issue an instruction to stop accelerating.

Remember, the idea of the static condition is not saying that nothing is happening but rather to say that nothing is changing in the instructions given to the system since the last instruction from the Control. If the last instruction by the Control is to accelerate, the system will continue to accelerate until told otherwise.

The Control element is the ‘primary control’ for the system. While everything is operating within a ‘normal’ operational mode, the Control element remains the primary control.

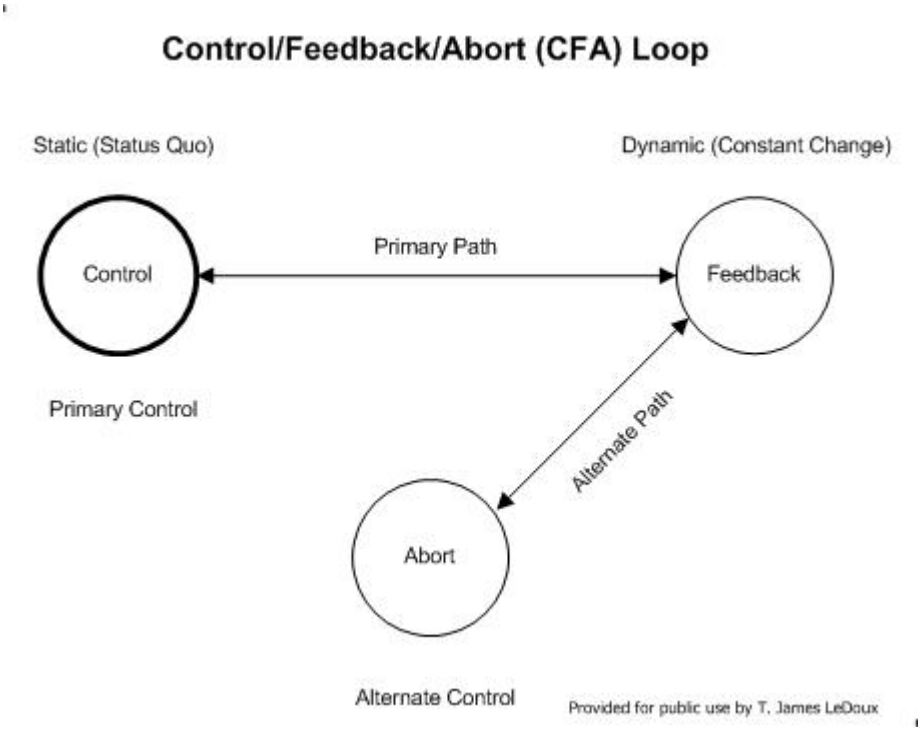


Figure 2 – CFA Loop – Control Element

The feedback element

The Feedback element feeds back information on the present state of the system. Due to the fact that the Feedback element is always reading the present state of the system, the feedback element has the basic characteristic of always being in a 'dynamic' state. This means that the feedback is reading constantly changing conditions. No system is ever in a non-changing condition except if it is off, no longer functioning or dead. Look at a computer in a wait state. It is still performing administrative activities even while it waits for some activity to happen. Change is the constant state of the Feedback element.

For this reason, the Feedback element needs to provide information to the Control element at intervals necessary to provide the Control element time to adequately respond to the changing environment.

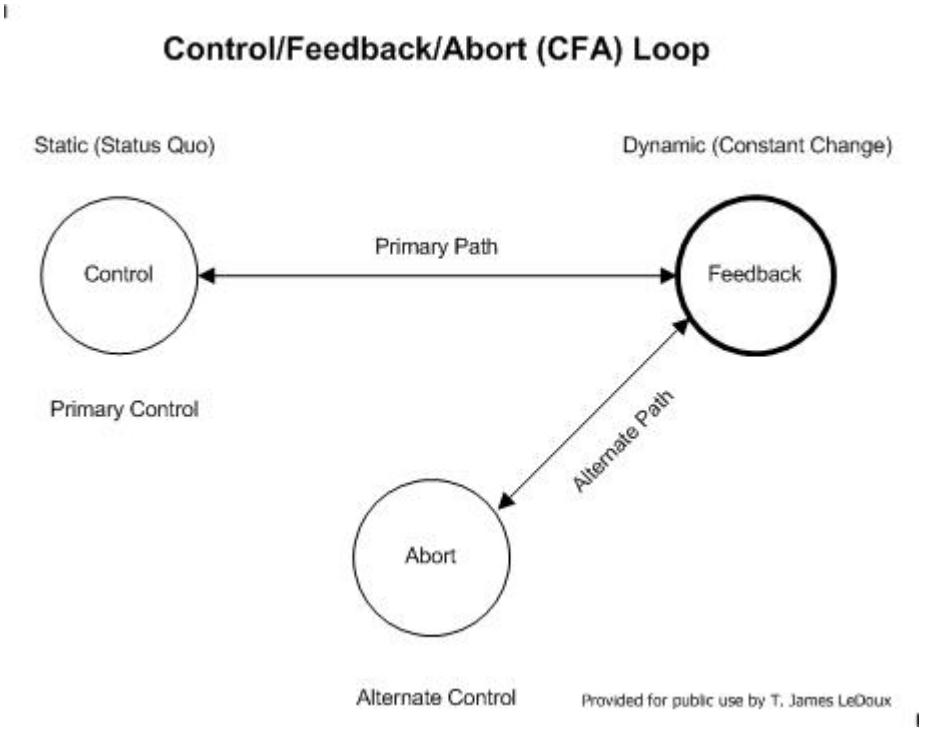


Figure 3 – CFA Loop – Feedback Element

The communication between the Control element and the Feedback element is performed by way of the 'Primary Path' (see Figure 4). The Primary Path is a bidirectional path that allows for the Control Element to request a sample of the information and the Feedback element to respond.

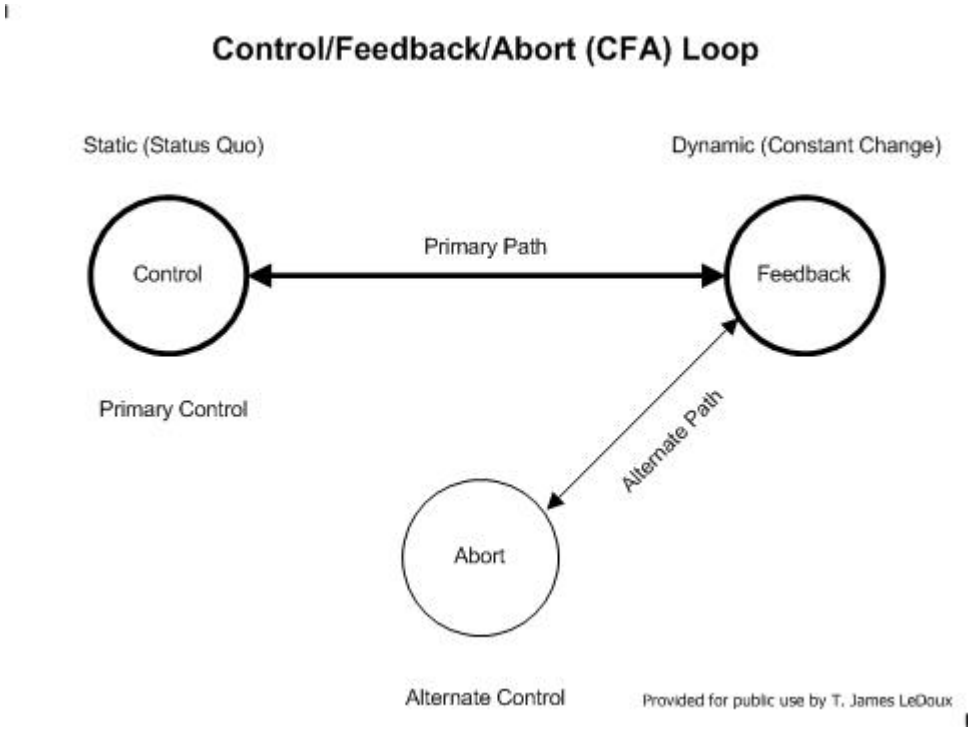


Figure 4 – CFA Loop – Primary Path

The abort element

The Abort element (see Figure 5) is so named because it responds to conditions that resulted in the primary path being 'aborted'. The Abort element then takes over the act of control until conditions can be brought back into acceptable parameters.

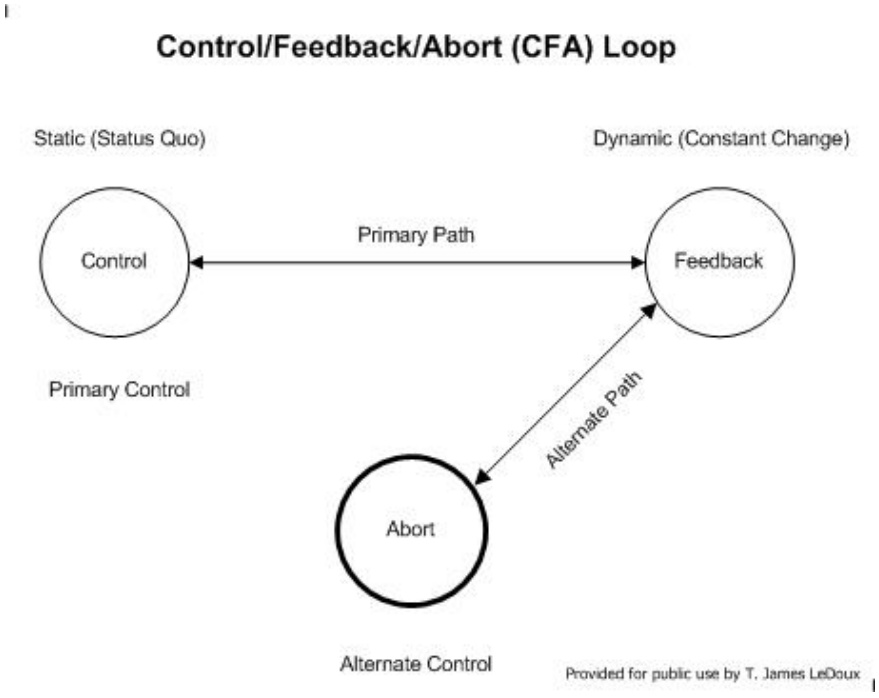


Figure 5 – CFA Loop – Abort Element

The 'Alternate Path' (see Figure 6) is used for communication between the Alternate Control (Abort) and the Feedback. The Feedback at this point may be a different set of feedbacks than was defined for the primary path.

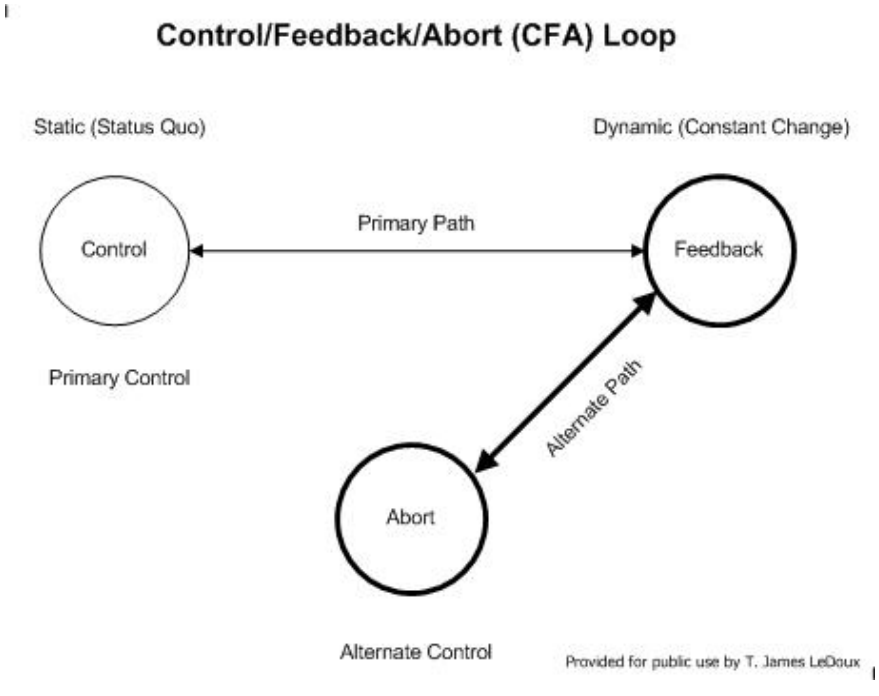


Figure 6 – CFA Loop – Alternate Path

To demonstrate that the Feedback may be another set of feedback elements, we look at the following example.

Let's use the act of driving the auto once more for our example (see Figure 7). When a driver is driving the car, the primary path is the Control element (gas pedal) and the Feedback element (speedometer and street signs). Once a stop sign is detected ahead, the driver will take the foot off the gas pedal (primary control) and press the brake pedal (alternate control). Note that the driver is no longer looking at the speedometer or street signs once the auto gets to the stop sign. The driver is looking for other cars that may cross his path. In other words, the driver is looking for a different set of feedback sources. Once he feels it is safe to go, he will go back to the primary control and feedback and the primary path.

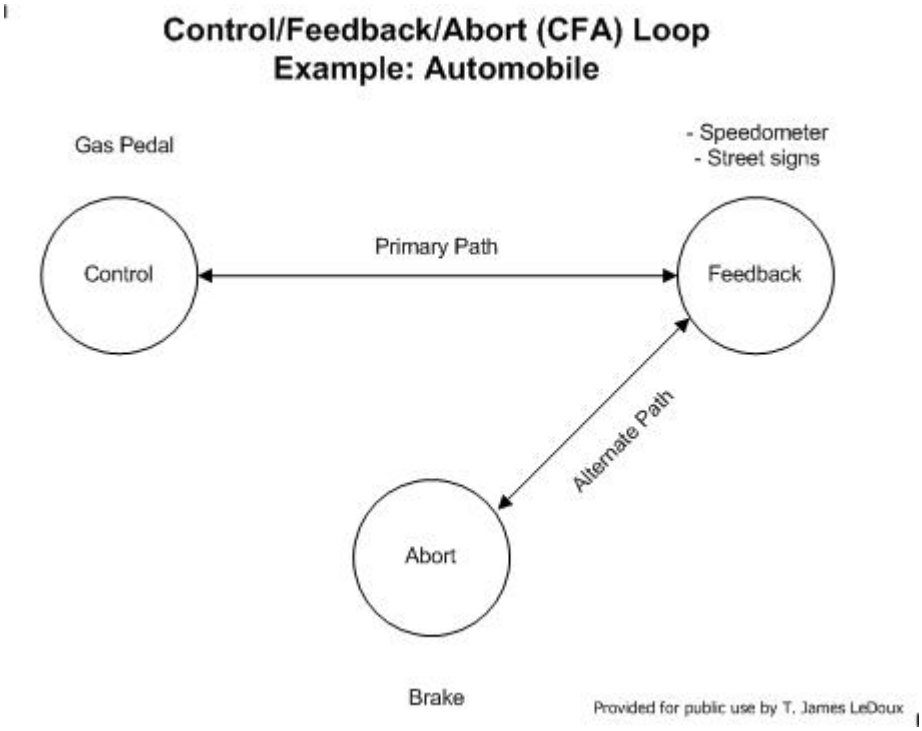


Figure 7 – CFA for Automobile Speed Adjustment

Sampling and the feedback element

In order for the Control element to be able to give proper instructions on what the system needs to do next, the information provided by the Feedback needs to be a true representation of the present conditions. If the feedback information is sampled by the Control element too often, it can put unnecessary demands on the system. If the information is not read often enough, considerable error can occur resulting in system failure. The solution to this dilemma is to sample when needed, at a rate that allows us to have confidence that we can still maintain control over the system.

Going back to our auto. The rate we sample the street signs for information is going to be different than when we look at the speedometer. We may also change our sampling rate when certain outside influences introduce themselves into the feedback mix. If we have a police car behind us, odds are that we will be sampling the speedometer much more often than if a police car was not there.

Creating the control loop diagram using the CFA Loop

The Control Loop Diagram is a chart that provides a list of each of the conditions we discover during the analysis of the interaction of the specific item in question. A basic Control Loop Diagram is shown in Table 1.

Control Element Conditions	Feedback Element Conditions	Abort Element Conditions
Control Element Name	Feedback Element Name	Abort Element Name
Numbered list	Numbered list	Numbered list

Table 1 – Control Loop Diagram Template

The Control Loop Diagram provides a vehicle for the CFA Loop to be used effectively. The following is a sequence that allows for us to create the CFA Loop analysis information and convert it into a Control Loop Diagram. The process is:

A. Identify the perspective of the CFA Loop.

It is important to know what the perspective is. We may be looking at the environment from a specific perspective (i.e. from the viewpoint of a Test Manager looking at defects or a Development Manager looking at versions.) The perspective will determine what is to be the Control and what is providing the Feedback for the analysis.

B. Identify what is controlling the environment.

C. Identify the Feedback components.

By identifying the controlling environment and the feedback elements, we can identify the parameters of the primary path.

D. Identify the conditions that would lead to an abort of the primary path.

The abort conditions can give us an insight into the limitations and boundaries the primary path must operate within.

E. Identify the processes the Control will use to manage the environment.

The interaction between the control and feedback elements can now be analyzed and the resulting information can be mapped into the Control Loop Diagram.

F. Identify the processes used when the Abort is given control.

An Example of the CFA Loop – Control loop diagram relationship

The following CFA Loop and Control Loop Diagram exhibit the relationship between a Version Control/Defect Reporting CFA Loop (Figure 8) and its associated Control Loop Diagram (Table 2).

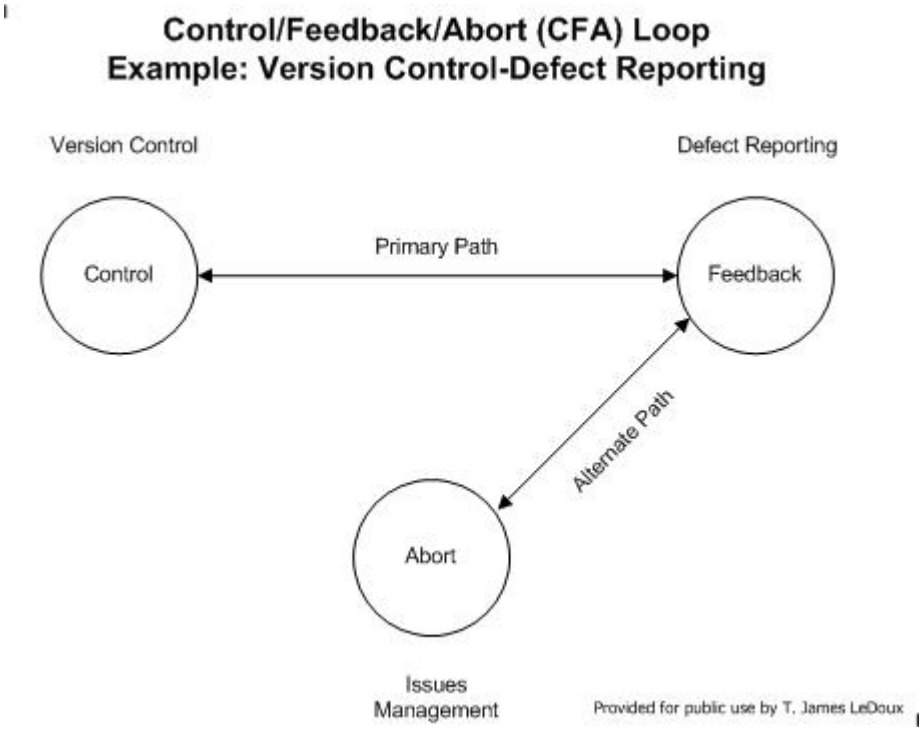


Figure 8 – CFA Loop for Version Control/Defect Reporting

A Control Loop Diagram for the CFA Loop with a focus on Version Control as the Control element (see Table 2) should look similar to the following table (mapped during an analysis brainstorming session):

Control Element Conditions	Feedback Element Conditions	Abort Element Conditions
Version Control (A)	Defect Reporting (B)	Issues Management (C)

1. Version must match the incremental sub-version number expected to fix the next set of defects	1. Defects reported back by critical levels	1. Defects that cannot be fixed within a predefined time must be escalated
2. Defects that are fixed, tested and passed cause the sub-version count to be incremented	2. Critical defect count	2. Defects violating the critical defect count or the age limits on critical defects will automatically create an Abort
3. Defects not above a predefined count		

Table 2 – Control Loop Diagram

Control charts

Control Charts have a very close relationship to the CFA Loop. Control Charts are used to provide a means of tracking the trend and condition of a specific measured item. The Control Chart (see Figure 9) uses the standard deviation of sampled items to determine whether the item is in-bounds (within acceptable conditions) or out of bounds (outside of acceptable conditions). The +3s is also identified as the Upper Defined-Control Limit or UDL. The -3s is also known as the Lower Defined-Control Limit or LDL.

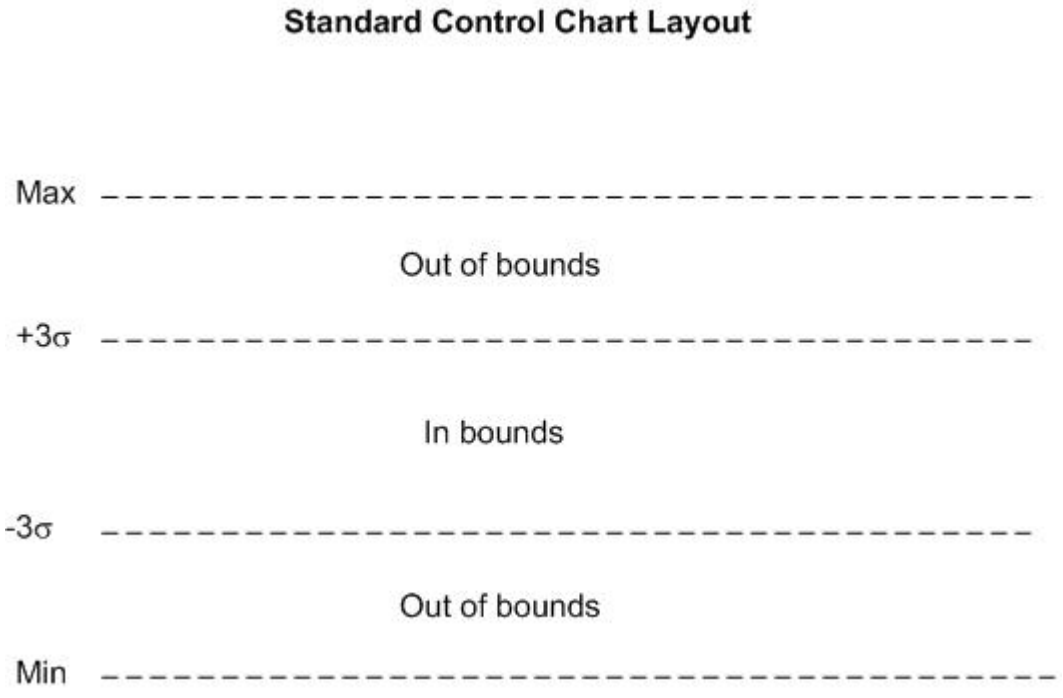


Figure 9 – Control Chart Limits

Those items that are in bounds are considered to be in control (see Figure 10). They can be the Control element of the CFA Loop.

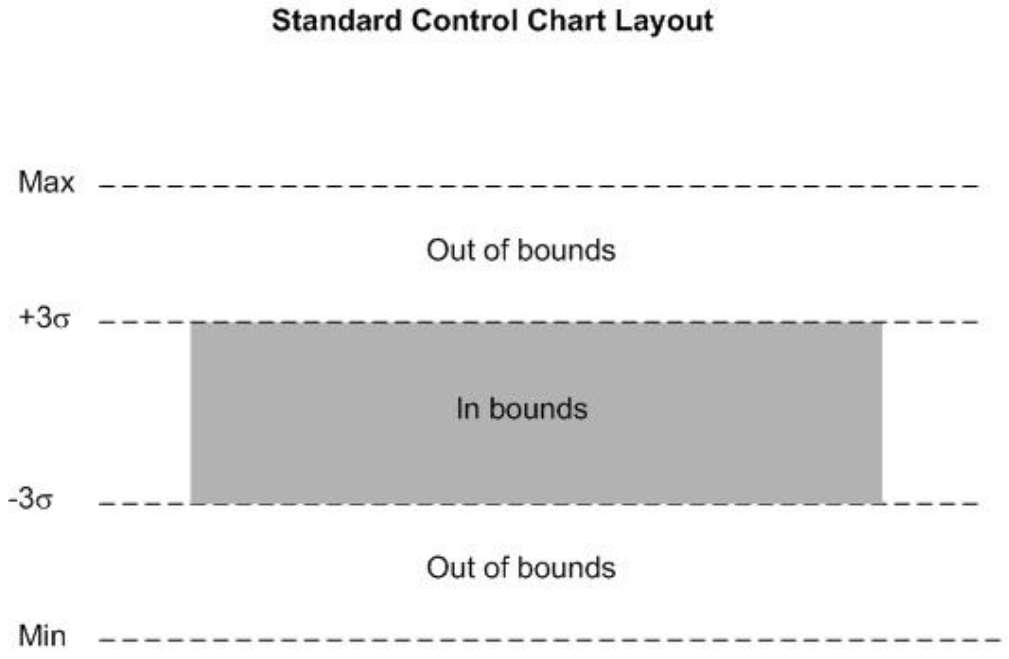


Figure 10 – Control Chart In-Bounds Area

Those items that are out of bounds are said to be out of control (see Figure 11). The out of bounds areas can also be identified as the Abort element of the CFA Loop.

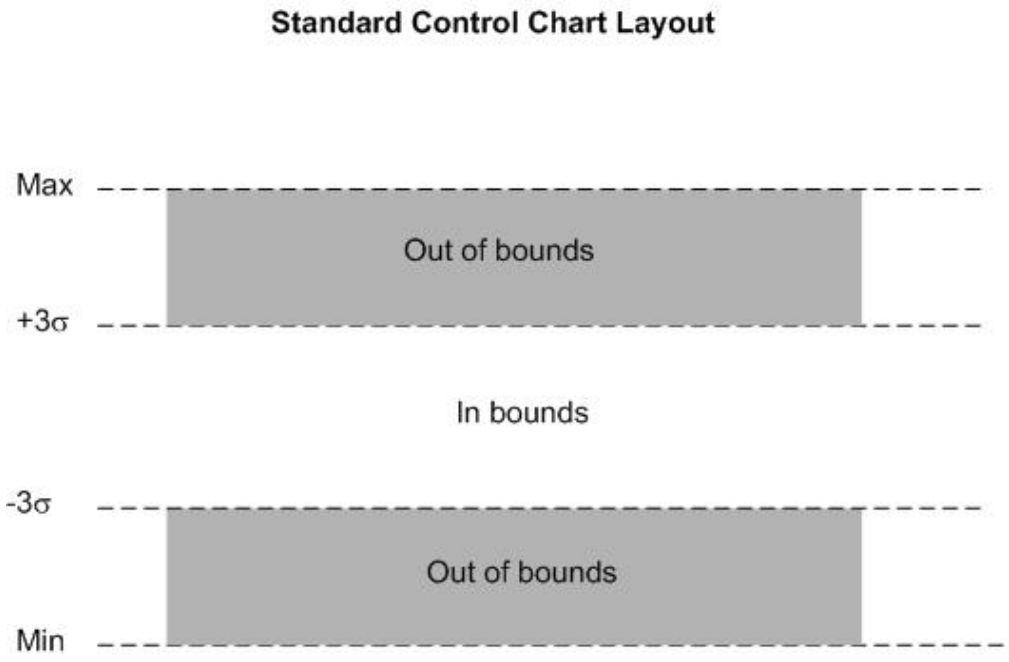


Figure 11 – Control Chart Out-Of-Bounds Area

Remember that it was mentioned earlier in this document that the CFA Loop and the Control Chart share the similar functions, the difference is in the use and objectives. We have already seen the Control and Abort similarities.

Let's look at a Control Chart (see Figure 12) and compare the information in the Control Chart with the CFA Loop elements.

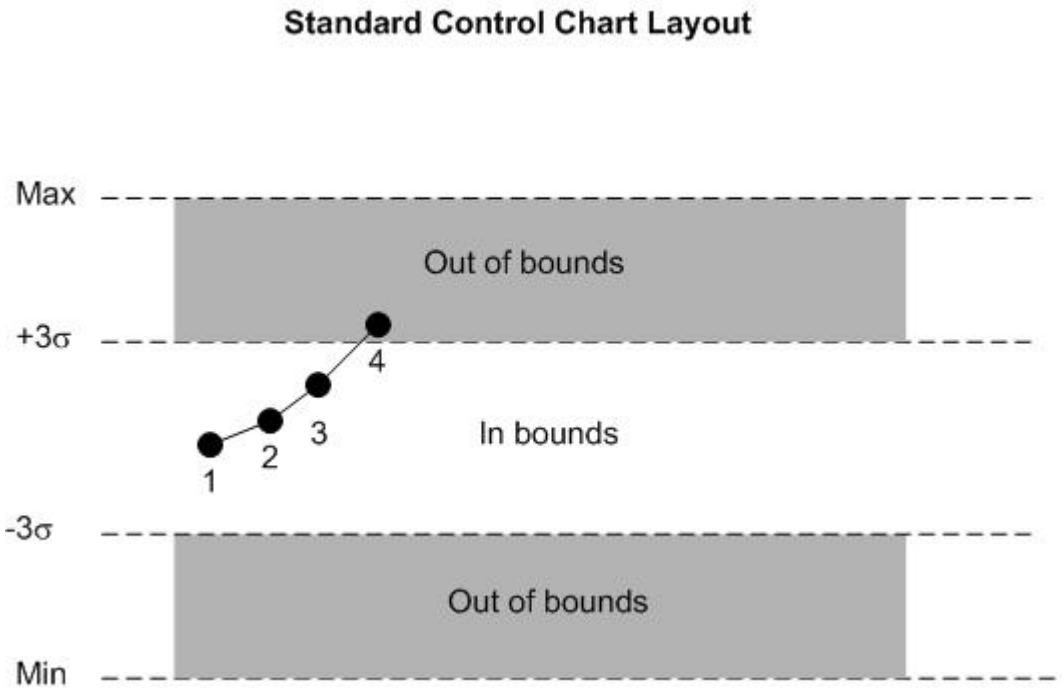


Figure 12 – Control Chart Showing Use

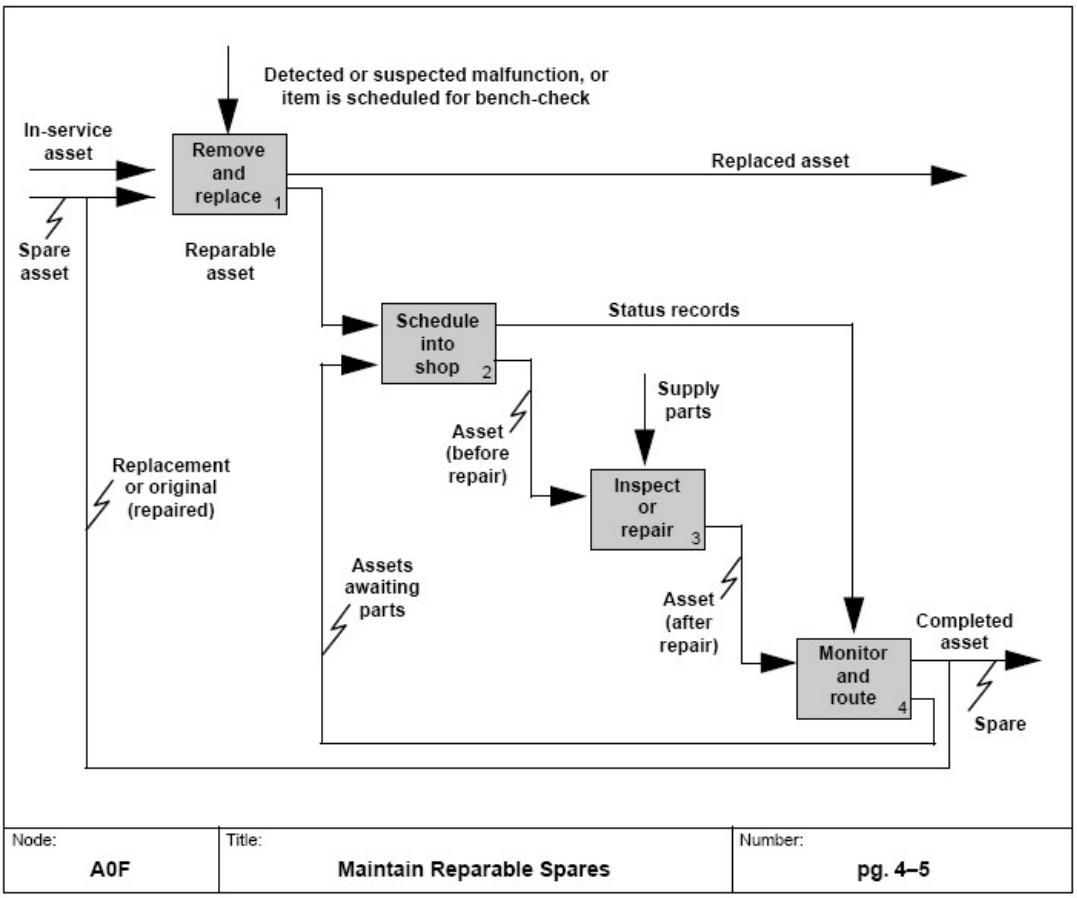
The 'in bounds' area is our Control element. As long as our data points, sometimes called items, are within the 'in bounds' area, we are said to be in control. The data points are the Feedback element. The 'out of bounds' areas are the Abort elements. Notice that data point 4 is in the 'out of bounds' area, which should lead to control being passed to the Abort element in order to take action to bring the future data points back into control. During the analysis of the system operation using the CFA Loop, the abort mechanism should have been clearly identified so that when the system goes out of bounds during operation, the alternate control should have been activated and the alternate action should be no surprise to the system designers.

The benefit of using control charts is due to its ability to report dynamic conditions of a system in operation. By data point 2, we should be able to see that if the data follows the trend set by the previous data points, the data will go out of control at some point. This ability to see the trend allows for the chart user to take early action to ensure that the system stays in control or to monitor automated abort processes used to bring the system back into control.

Chapter 3

IDEF0 & IDEF1X

IDEF0



IDEF0 Diagram Example

IDEF0 (*Integration Definition for Function Modeling*) is a function modeling methodology for describing manufacturing functions, which offers a functional modeling language for the analysis, development, reengineering, and integration of information systems; business processes; or software engineering analysis.

IDEF0 is part of the IDEF family of modeling languages in the field of software engineering, and is built on the functional modeling language Structured Analysis and Design Technique (SADT).

Overview

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".

Where the Functional flow block diagram is used to show the functional flow of a product, IDEF0 is used to show data flow, system control, and the functional flow of life cycle processes. IDEF0 is capable of graphically representing a wide variety of business, manufacturing and other types of enterprise operations to any level of detail. It provides rigorous and precise description, and promotes consistency of usage and interpretation. It is well-tested and proven through many years of use by government and private industry. It can be generated by a variety of computer graphics tools. Numerous commercial products specifically support development and analysis of IDEF0 diagrams and models.

An associated technique, Integration Definition for Information Modeling (IDEF1x), is used to supplement IDEF0 for data intensive systems. The IDEF0 standard, Federal Information Processing Standards Publication 183 (FIPS 183), and the IDEF1x standard (FIPS 184) are maintained by the National Institute of Standards and Technology (NIST).

History

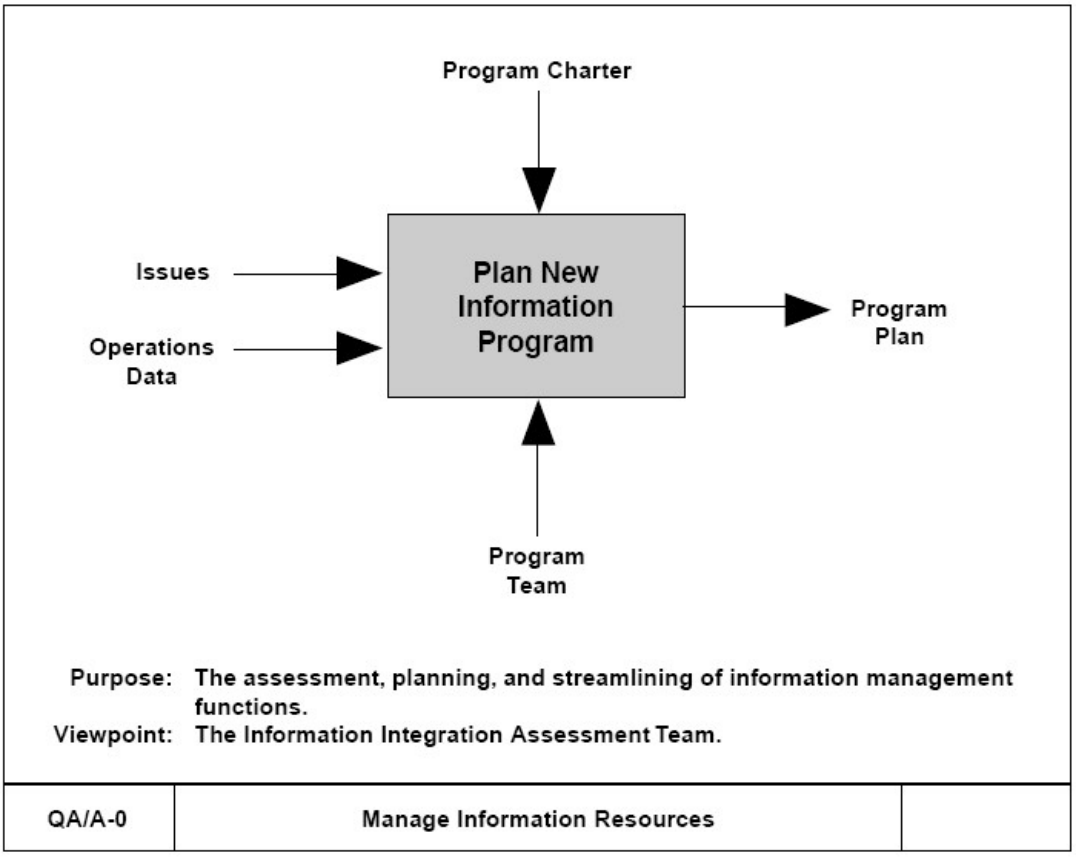
During the 1970s, the U.S. Air Force Program for Integrated Computer Aided Manufacturing (ICAM) sought to increase manufacturing productivity through systematic application of computer technology. The ICAM program identified the need for better analysis and communication techniques for people involved in improving manufacturing productivity. As a result, in 1981 the ICAM program developed a series of techniques known as the IDEF (ICAM Definition) techniques which included the following:

- IDEF0, used to produce a "function model". A function model is a structured representation of the functions, activities or processes within the modeled system or subject area.
- IDEF1, used to produce an "information model". An information model represents the structure and semantics of information within the modeled system or subject area.

- IDEF2, used to produce a "dynamics model". A dynamics model represents the time-varying behavioral characteristics of the modeled system or subject area.

In 1983, the U.S. Air Force Integrated Information Support System program enhanced the IDEF1 information modeling technique to form IDEF1X (IDEF1 Extended), a semantic data modeling technique. By the 1990s, IDEF0 and IDEF1X techniques are widely used in the government, industrial and commercial sectors, supporting modeling efforts for a wide range of enterprises and application domains. In 1991 the National Institute of Standards and Technology (NIST) received support from the U.S. Department of Defense, Office of Corporate Information Management (DoD/CIM), to develop one or more Federal Information Processing Standard (FIPS) for modeling techniques. The techniques selected were IDEF0 for function modeling and IDEF1X for information modeling. These FIPS documents are based on the IDEF manuals published by the U.S. Air Force in the early 1980s.

IDEF0 Topics

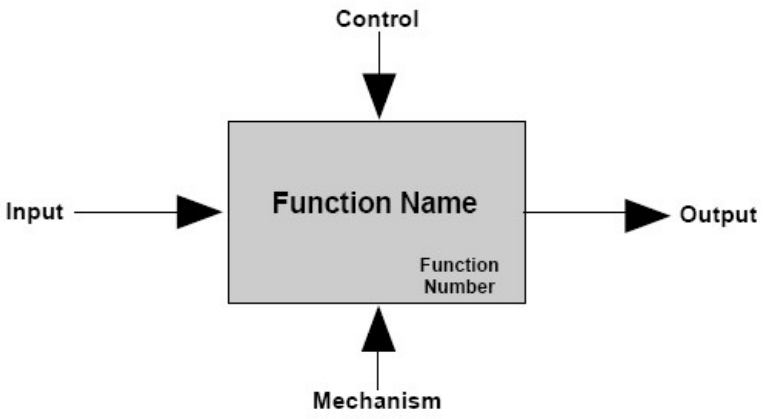


Top-Level Context Diagram

The IDEF0 Approach

IDEF0 may be used to model a wide variety of automated and non-automated systems. For new systems, it may be used first to define the requirements and specify the functions, and then to design an implementation that meets the requirements and performs the functions. For existing systems, IDEF0 can be used to analyze the functions the system performs and to record the mechanisms (means) by which these are done. The result of applying IDEF0 to a system is a model that consists of a hierarchical series of diagrams, text, and glossary cross-referenced to each other. The two primary modeling components are functions (represented on a diagram by boxes) and the data and objects that inter-relate those functions (represented by arrows).

IDEF0 Building blocks

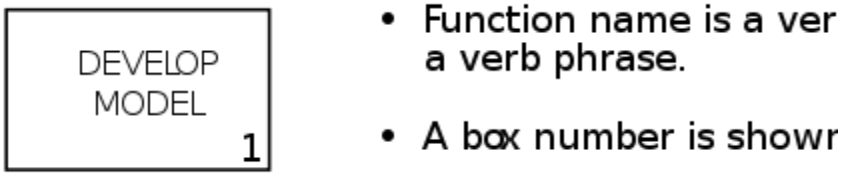


Integration Definition for Function Modeling (IDEF0) Box Format

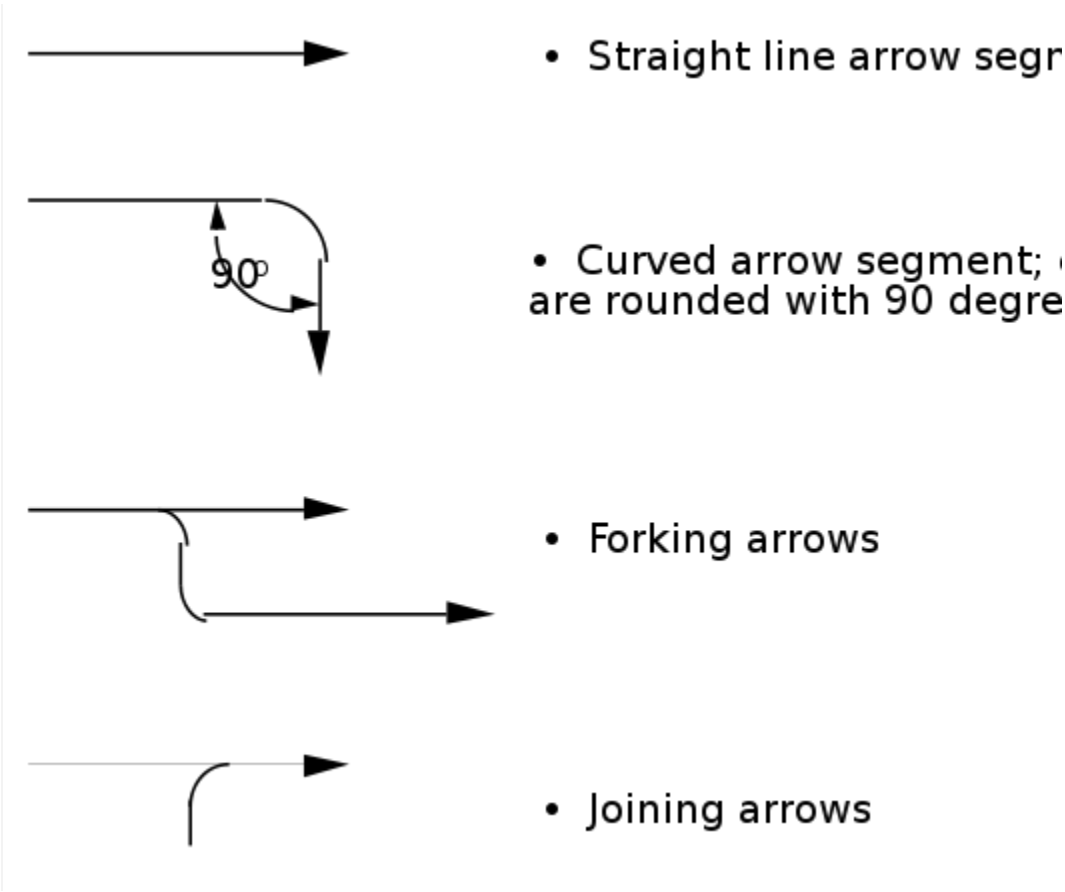
The IDEF0 model displayed here on the left is based on a simple syntax. Each activity is described by a verb based label placed in a box. Inputs are shown as arrows entering the left side of the activity box while output are shown as exiting arrows on the right side of the box. Controls are displayed as arrows entering the top of the box and mechanisms are displayed as arrows entering from the bottom of the box. Inputs, Controls, Outputs, and Mechanisms are all referred to as concepts.

- *Arrow* : A directed line, composed of one or more arrow segments, that models an open channel or conduit conveying data or objects from source (no arrowhead) to use (with arrowhead). There are 4 arrow classes: Input Arrow, Output Arrow, Control Arrow, and Mechanism Arrow (includes Call Arrow).

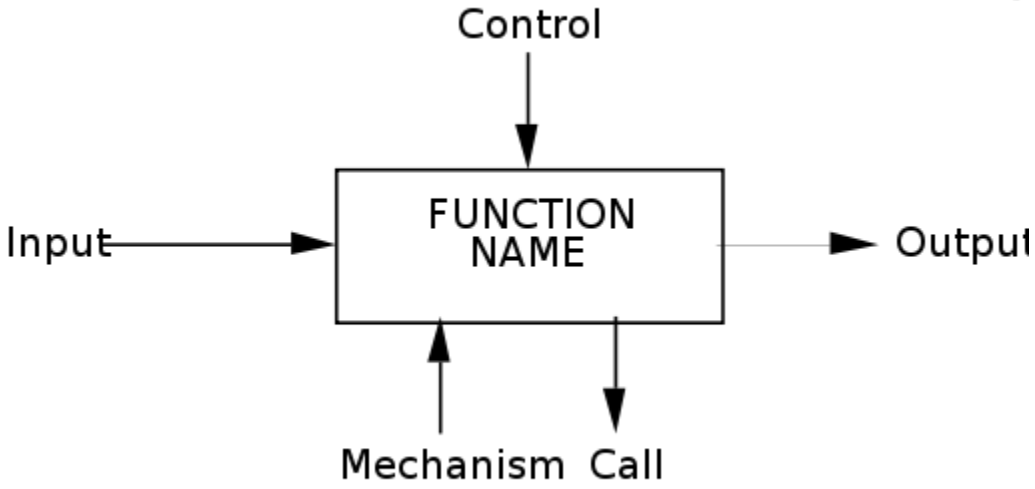
- *Box* : A rectangle, containing a name and number, used to represent a function.



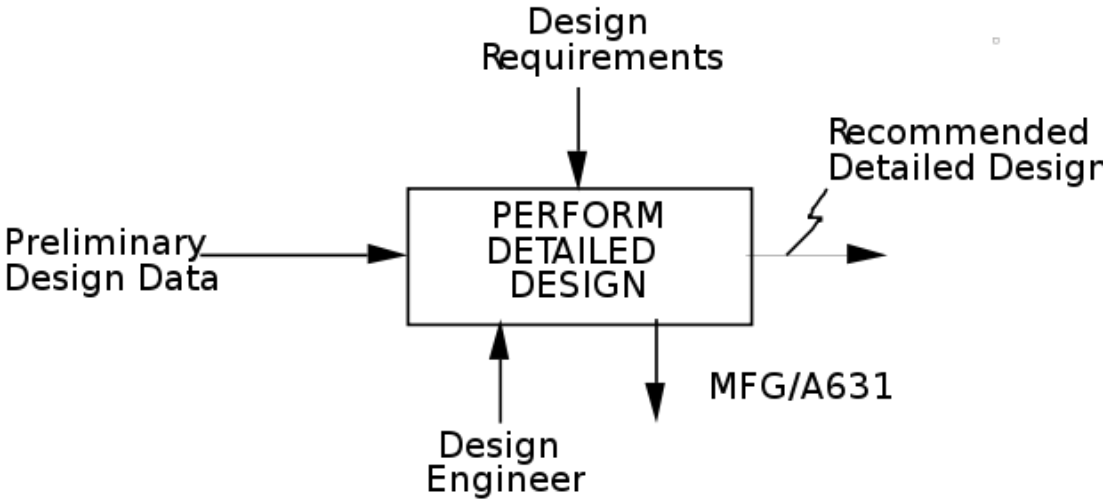
Box Syntax



Arrow Syntax

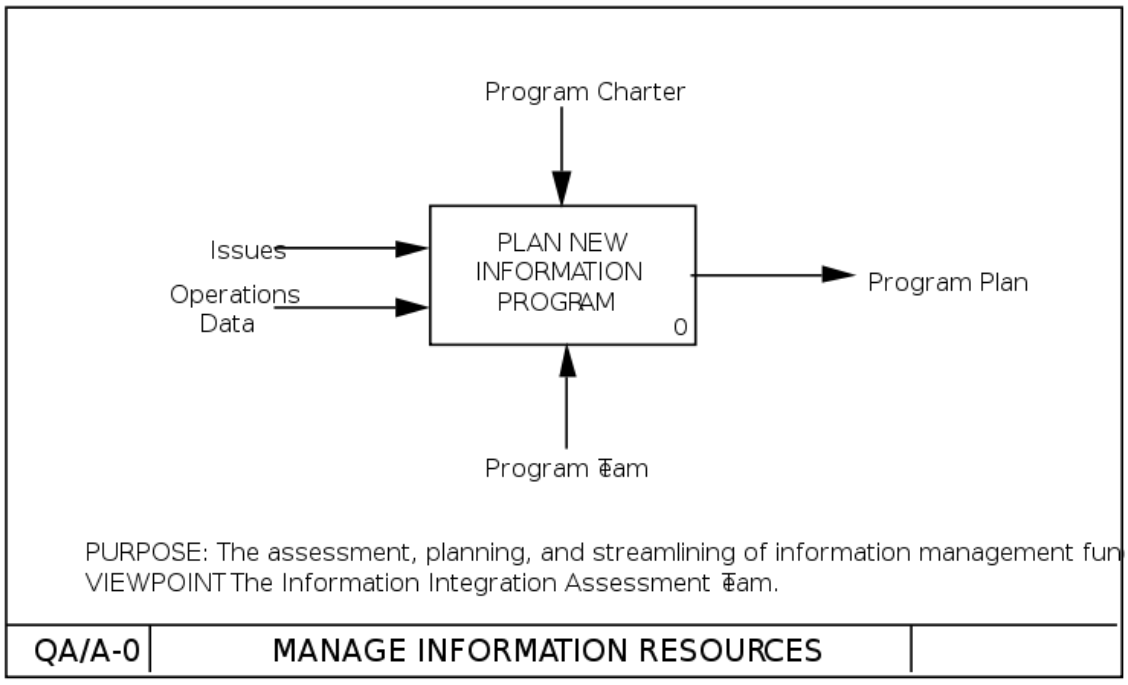


Arrow Positions and Roles

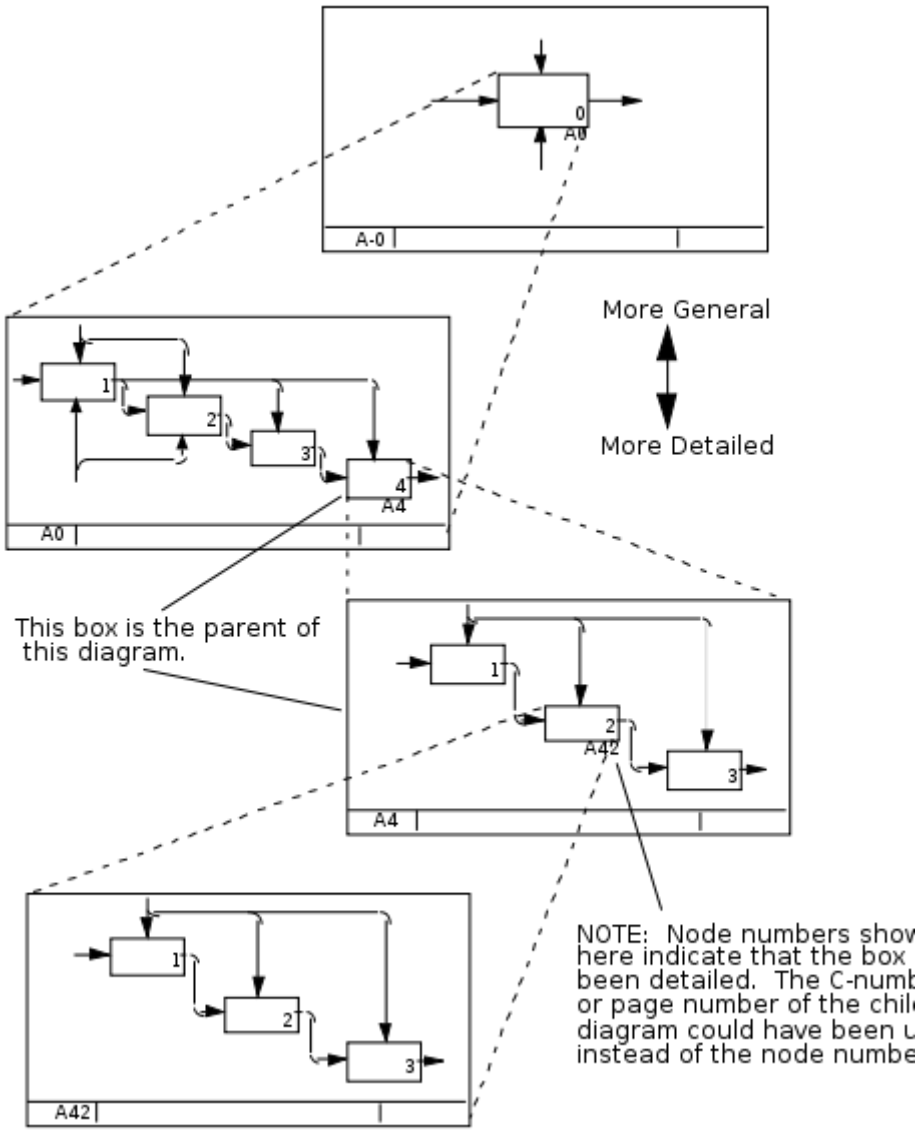


Label and Name Semantics

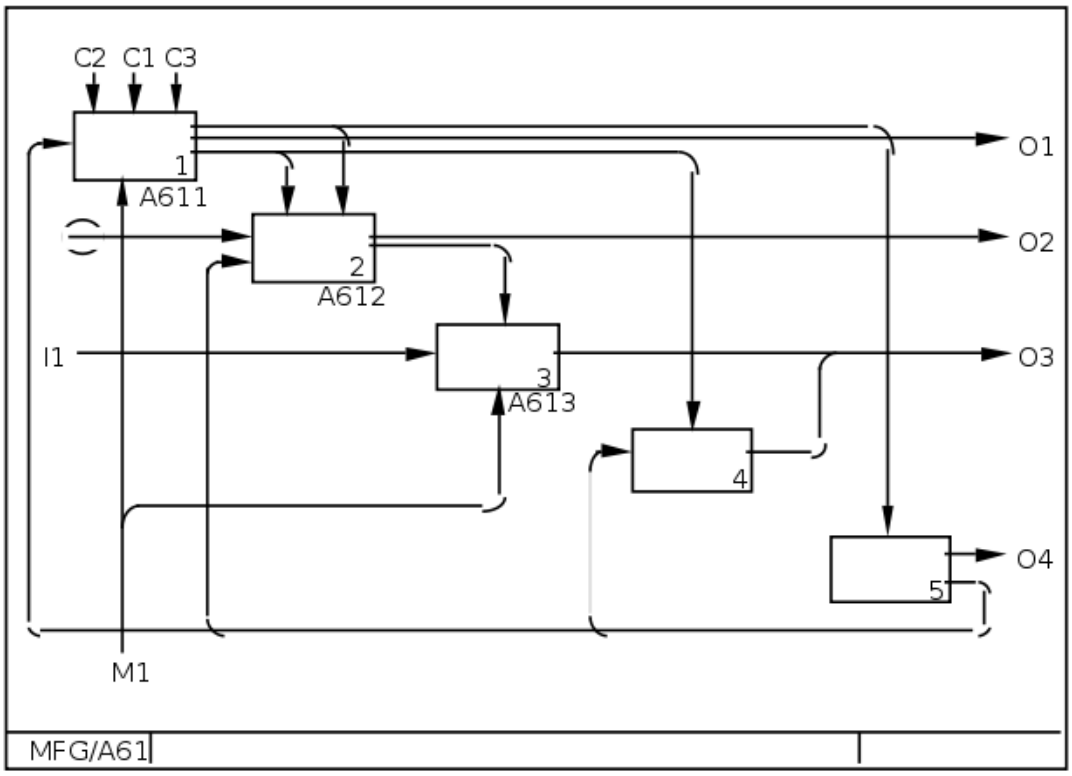
- *Context* : The immediate environment in which a function (or set of functions on a diagram) operates.
- *Decomposition* : The partitioning of a modeled function into its component functions.



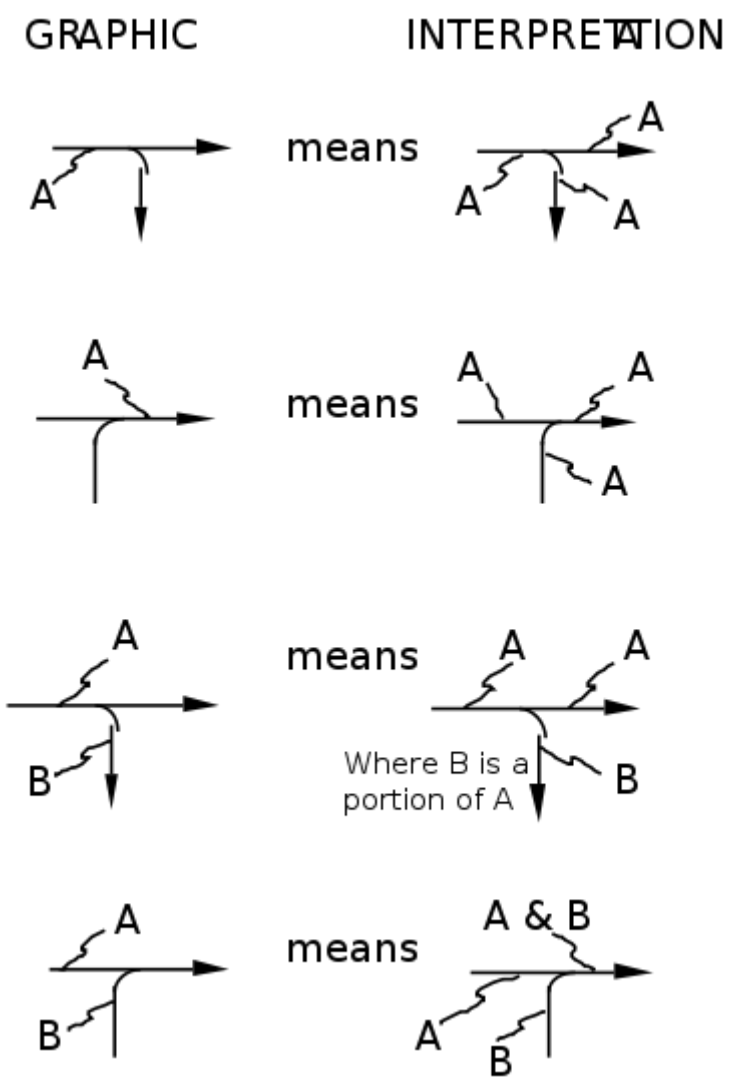
Example Top-level Diagram



Decomposition Structure

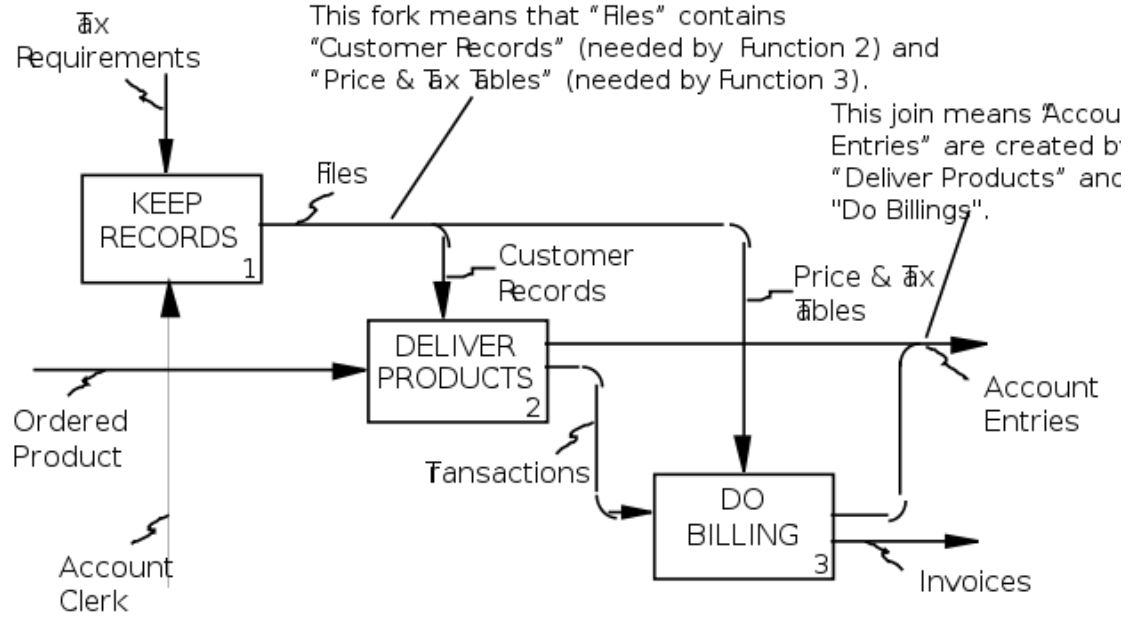


Detail Reference Expression Use

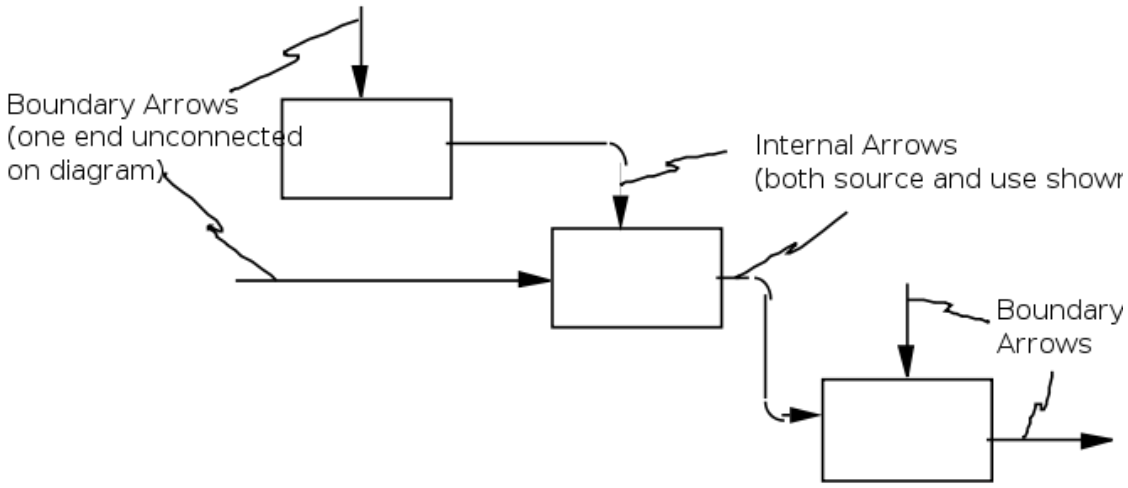


Arrow Fork and Join Structures

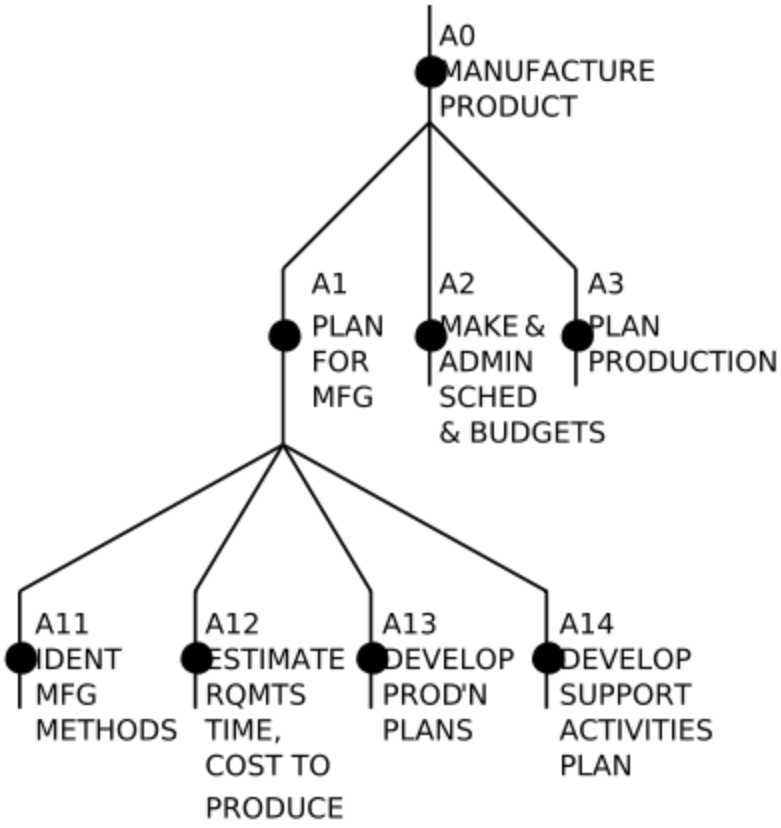
- *Fork* : The junction at which an IDEF0 arrow segment (going from source to use) divides into two or more arrow segments. May denote unbundling of meaning.



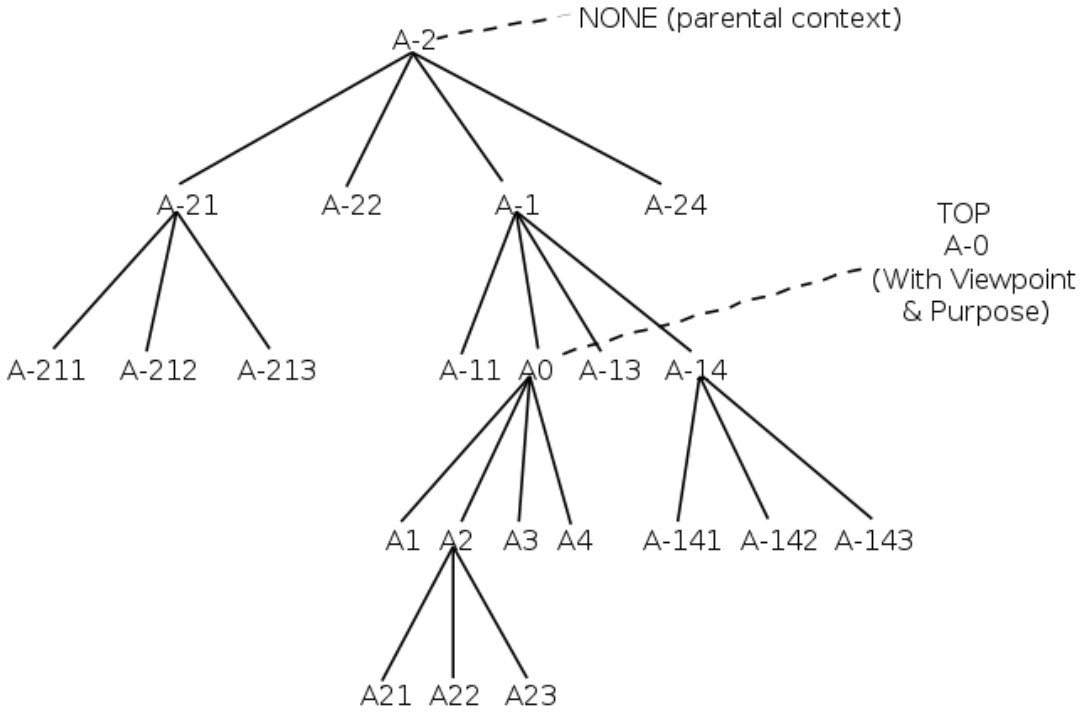
Connections Between Boxes



Boundary and Internal Arrows



Typical Node Tree

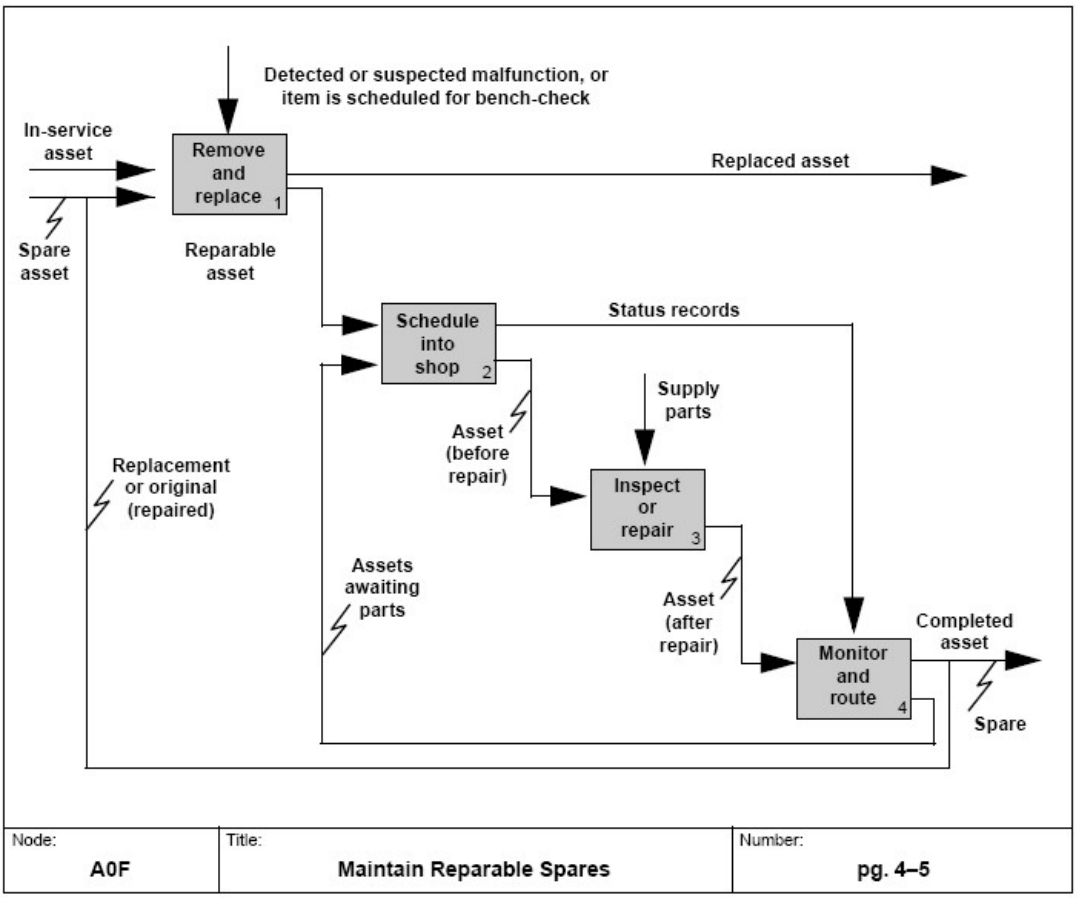


Negative Node-Numbered Context

- *Function* : An activity, process, or transformation (modeled by an IDEF0 box) identified by a verb or verb phrase that describes what must be accomplished.

- *Join* : The junction at which an IDEF0 arrow segment (going from source to use) merges with one or more other arrow segments to form a single arrow segment. May denote bundling of arrow segment meanings

- *Node* : A box from which child boxes originate; a parent box.



IDEF0 Diagram Example

Diagrammic notation

IDEF0 is a model that consists of a hierarchical series of diagrams, text, and glossary cross referenced to each other. The two primary modeling components are:

- functions (represented on a diagram by boxes), and
- data and objects that interrelate those functions (represented by arrows).

As shown by Figure 3 the position at which the arrow attaches to a box conveys the specific role of the interface. The controls enter the top of the box. The inputs, the data or objects acted upon by the operation, enter the box from the left. The outputs of the operation leave the right-hand side of the box. Mechanism arrows that provide supporting means for performing the function join (point up to) the bottom of the box.

The IDEF0 process

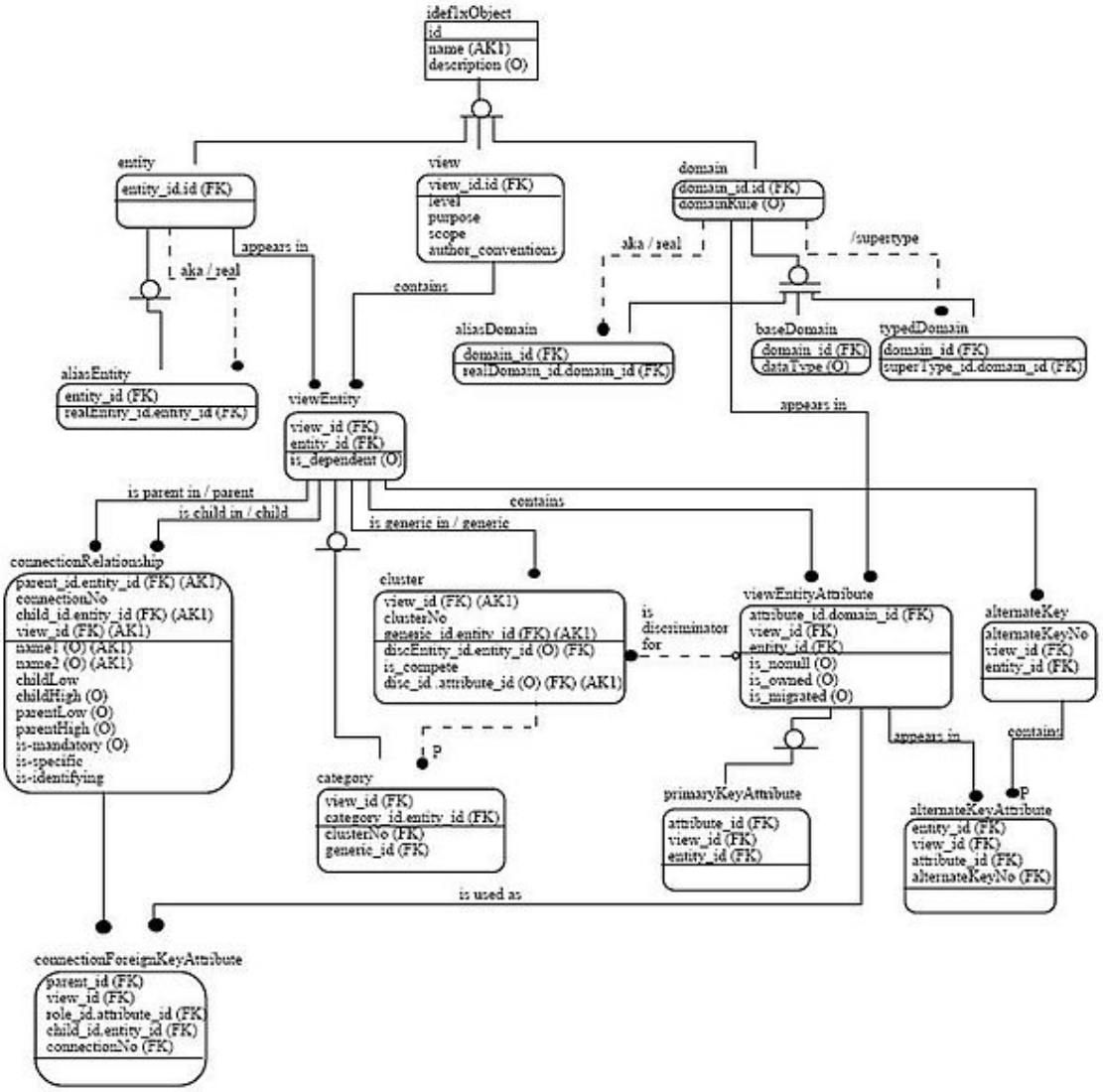
The IDEF0 process starts with the identification of the prime function to be decomposed. This function is identified on a “Top Level Context Diagram,” that defines the scope of

the particular IDEF0 analysis. An example of a Top Level Context Diagram for an information system management process is shown in Figure 3. From this diagram lower-level diagrams are generated. An example of a derived diagram, called a “child” in IDEF0 terminology, for a life cycle function is shown in Figure 4.

Federal Information Processing Standards

In Dec 1993 the National Institute of Standards and Technology announcing the standard for Integration Definition for Function Modeling (IDEF0) in the category Software Standard, Modeling Techniques. This publication announces the adoption of the IDEF0 as a Federal Information Processing Standard (FIPS). This standard was based on the Air Force Wright Aeronautical Laboratories Integrated Computer-Aided Manufacturing (ICAM) Architecture from June 1981.

IDEF1X



Example of an IDEF1X Diagram.

IDEF1X (**Integration Definition for Information Modeling**) is a data modeling language for the developing of semantic data models. IDEF1X is used to produce a graphical information model which represents the structure and semantics of information within an environment or system.

Use of the IDEF1X permits the construction of semantic data models which may serve to support the management of data as a resource, the integration of information systems, and

the building of computer databases. This standard is part of the IDEF family of modeling languages in the field of software engineering.

Overview

A data modeling technique is used to model data in a standard, consistent, predictable manner in order to manage it as a resource. It can be used in projects requiring a standard means of defining and analyzing the data resources within an organization. Such projects include the incorporation of a data modeling technique into a methodology, managing data as a resource, integrating information systems, or designing computer databases. The primary objectives of the IDEF1X standard are to provide:

- Means for completely understanding and analyzing an organization's data resources;
- Common means of representing and communicating the complexity of data;
- A technique for presenting an overall view of the data required to run an enterprise;
- Means for defining an application-independent view of data which can be validated by users and transformed into a physical database design; and
- A technique for deriving an integrated data definition from existing data resources.

A principal objective of IDEF1X is to support integration. The approach to integration focuses on the capture, management, and use of a single semantic definition of the data resource referred to as a “Conceptual schema”. The “conceptual schema” provides a single integrated definition of the data within an enterprise which is unbiased toward any single application of data and is independent of how the data is physically stored or accessed. The primary objective of this conceptual schema is to provide a consistent definition of the meanings and interrelationship of data which can be used to integrate, share, and manage the integrity of data. A conceptual schema must have three important characteristics. It must be:

- Consistent with the infrastructure of the business and be true across all application areas.
- Extendible, such that, new data can be defined without altering previously defined data.
- Transformable to both the required user views and to a variety of data storage and access structures.

History

The need for semantic data models was first recognized by the U.S. Air Force in the mid-1970s as a result of the Integrated Computer Aided Manufacturing (ICAM) Program. The objective of this program was to increase manufacturing productivity through the systematic application of computer technology. The ICAM Program identified a need for better analysis and communication techniques for people involved in improving

manufacturing productivity. As a result, the ICAM Program developed a series of techniques known as the IDEF (ICAM Definition) Methods which included the following:

- IDEF0 used to produce a "function model" which is a structured representation of the activities or processes within the environment or system.
- IDEF1 used to produce an "information model" which represents the structure and semantics of information within the environment or system.
- IDEF2 used to produce a "dynamics model"

The initial approach to IDEF information modeling (IDEF1) was published by the ICAM program in 1981, based on current research and industry needs. The theoretical roots for this approach stemmed from the early work of Edgar F. Codd on relational theory and Peter Chen on the entity-relationship model. The initial IDEF1 technique was based on the work of Dr. R.R. Brown and Mr. T.L. Ramey of Hughes Aircraft and Mr. D.S. Coleman of D. Appleton Company (DACOM), with critical review and influence by Charles Bachman, Peter Chen, Dr. M.A. Melkanoff, and Dr. G.M. Nijssen.

In 1983, the U.S. Air Force initiated the Integrated Information Support System (I2S2) project under the ICAM program. The objective of this project was to provide the enabling technology to logically and physically integrate a network of heterogeneous computer hardware and software. As a result of this project, and industry experience, the need for an enhanced technique for information modeling was recognized.

From the point of view of the contract administrators of the Air Force IDEF program, IDEF1X was a result of the ICAM IISS-6201 project and was further extended by the IISS-6202 project. To satisfy the data modeling enhancement requirements that were identified in the IISS-6202 project, a sub-contractor, DACOM, obtained a license to the Logical Database Design Technique (LDDT) and its supporting software (ADAM). From the point of view of the technical content of the modeling technique, IDEF1X is a renaming of LDDT.

Logical Database Design Technique

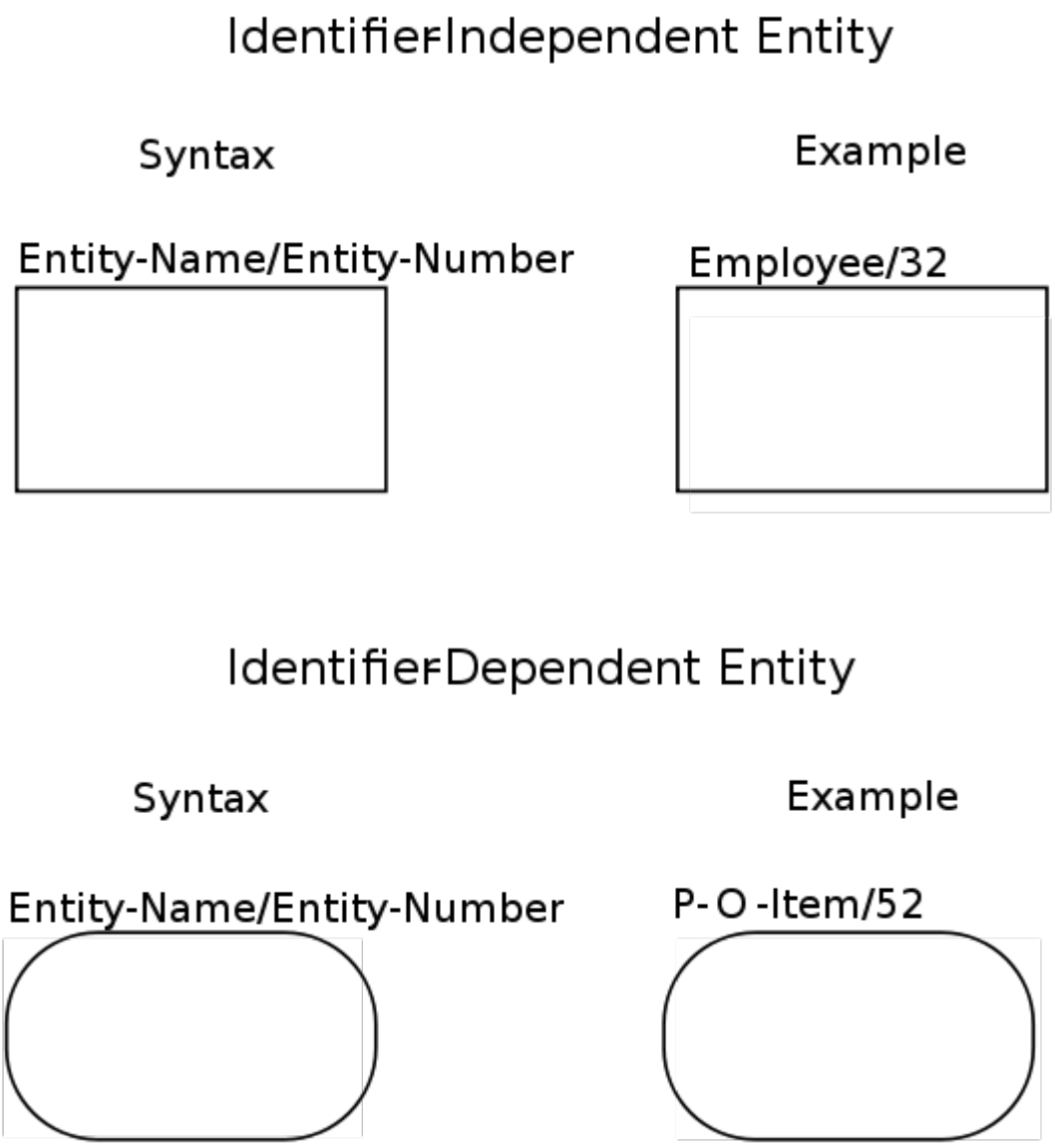
LDDT had been developed in 1982 by Robert G. Brown of The Database Design Group entirely outside the IDEF program and with no knowledge of IDEF1. Nevertheless, the central goal of IDEF1 and LDDT was the same: to produce a database neutral model of the persistent information needed by an enterprise by modeling the real-world entities involved. LDDT combined elements of the relational data model, the E-R model, and data generalization in a way specifically intended to support data modeling and the transformation of the data models into database designs.

LDDT included multiple levels of model, the modeling of generalization/specialization, and the explicit representation of relationships by primary and foreign keys, supported by a well defined role naming facility. The primary keys and unambiguously role-named foreign keys expressed sometimes subtle uniqueness and referential integrity constraints

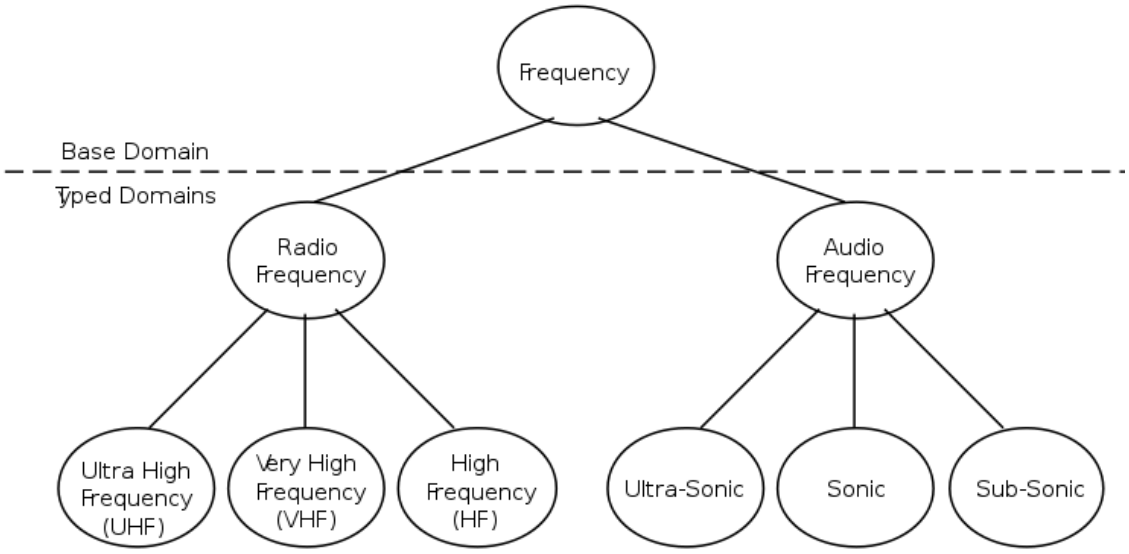
that needed to be known and honored by whatever type of database was ultimately designed. Whether the database design used the integrity constraint based keys of the LDDT model as database access keys or indexes was an entirely separate decision. The precision and completeness of the LDDT models was an important factor in enabling the relatively smooth transformation of the models into database designs. Early LDDT models were transformed into database designs for IBM's hierarchical database, IMS. Later models were transformed into database designs for Cullinet's network database, IDMS, and many varieties of relational database.

The graphic syntax of LDDT differed from that of IDEF1 and, more importantly, LDDT contained interrelated modeling concepts not present in IDEF1. Therefore, instead of extending IDEF1, Mary E. Loomis of DACOM wrote a concise summary of the syntax and semantics of a substantial subset of LDDT, using terminology compatible with IDEF1 wherever possible. DACOM labeled the result IDEF1X and supplied it to the ICAM program, which published it in 1985. (IEEE 1998, p. iii) (Bruce 1992, p. xii)

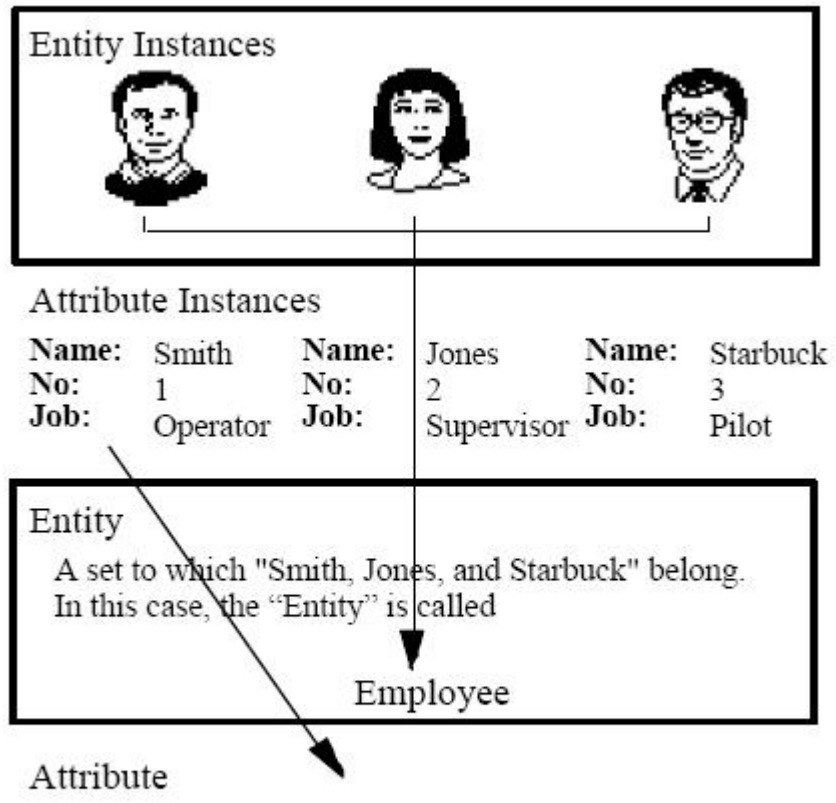
IDEF1X Building blocks



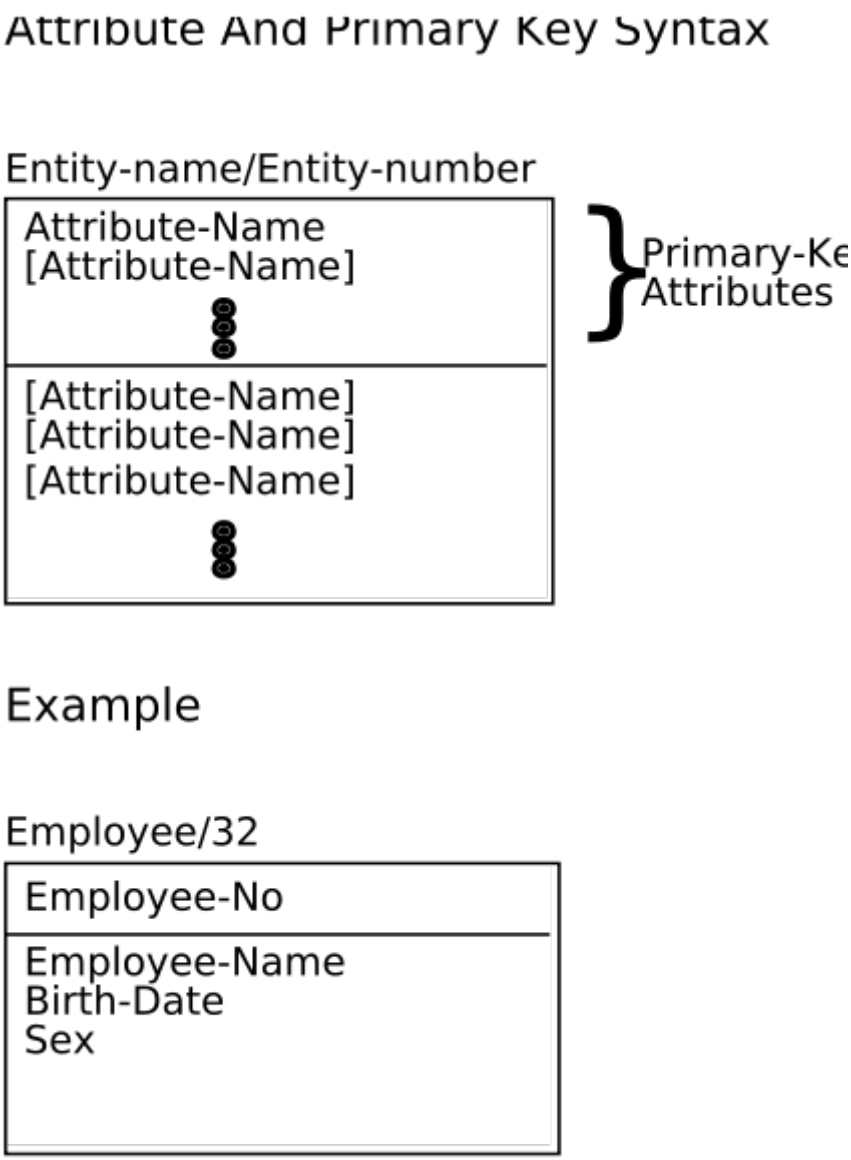
Entity Syntax



Domain Hierarchy



Attribute example



Primary Key Syntax

Entities

The representation of a set of real or abstract things (people, objects, places, events, ideas, combination of things, etc.) that are recognized as the same type because they share the same characteristics and can participate in the same relationships.

Domains

A named set of data values (fixed, or possibly infinite in number) all of the same data type, upon which the actual value for an attribute instance is drawn. Every attribute must be defined on exactly one underlying domain. Multiple attributes may be based on the same underlying domain.

Attributes

A property or characteristic that is common to some or all of the instances of an entity. An attribute represents the use of a domain in the context of an entity.

Keys

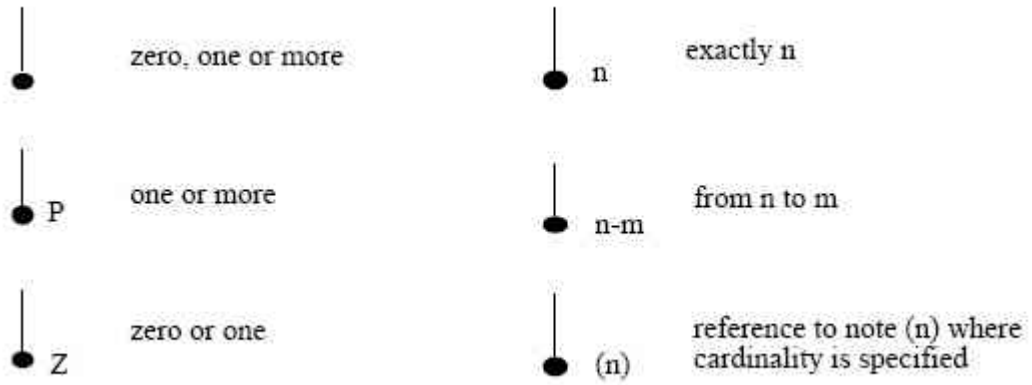
An attribute, or combination of attributes, of an entity whose values uniquely identify each entity instance.

Primary Keys

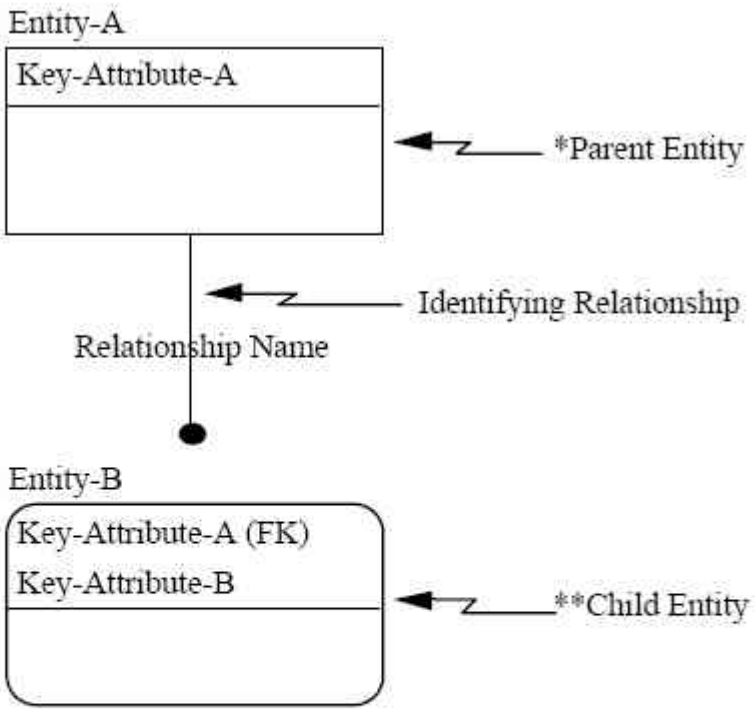
The candidate key selected as the unique identifier of an entity.

Foreign Keys

An attribute, or combination of attributes of a child or category entity instance whose values match those in the primary key of a related parent or generic entity instance. A foreign key results from the migration of the parent or generic entities primary key through a specific connection or categorization relationship.



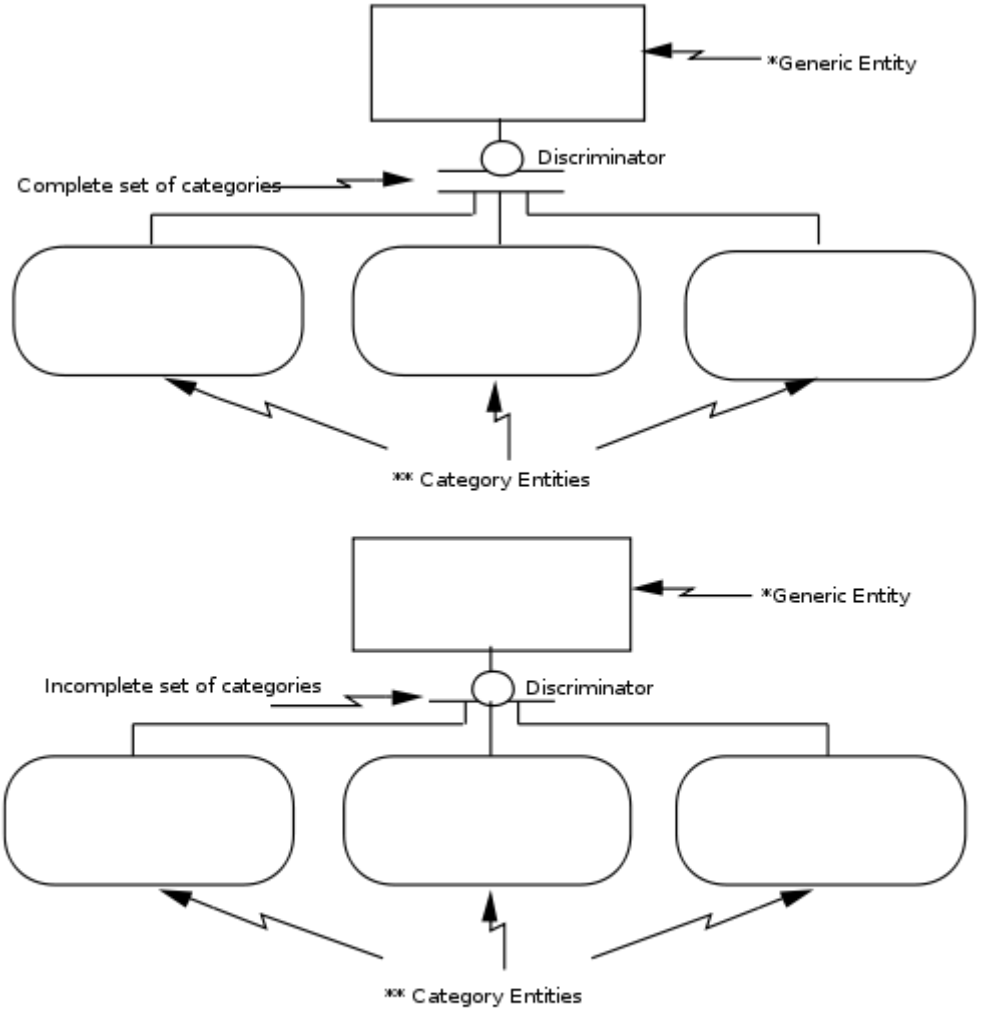
Relationship Cardinality Syntax



* The Parent Entity in an Identifying Relationship may be an Identifier-Independent Entity (as shown) or an Identifier-Dependent Entity depending upon other relationships.

** The Child Entity in an Identifying Relationship is always an Identifier-Dependent Entity.

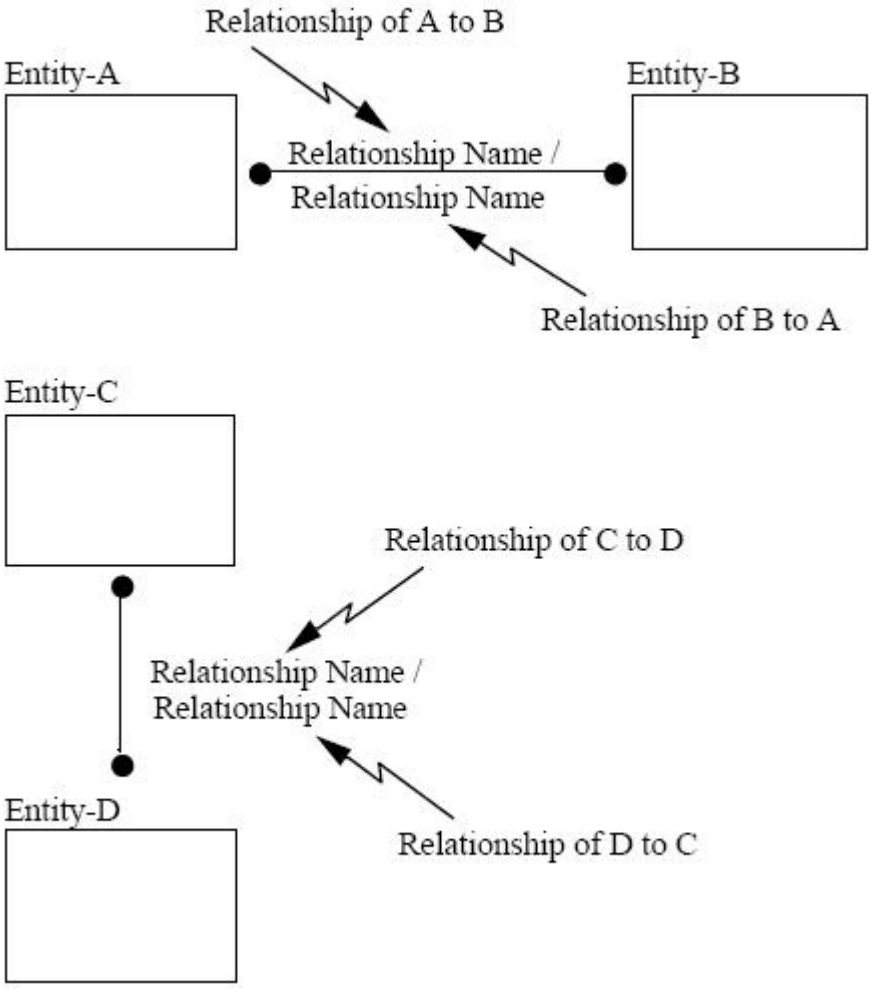
Identifying Relationship Syntax



* The Generic Entity may be an Identifier-Independent Entity (as shown) or a Identifier-Dependent Entity depending upon other relationships.

** Category Entities will always be Identifier-Dependent Entities.

Categorization Relationship Syntax



Non-Specific Relationship Syntax

Relationships

An association between two entities or between instances of the same entity.

Connection Relationships

The number of entity instances that can be associated with each other in a relationship.

Categorization Relationships

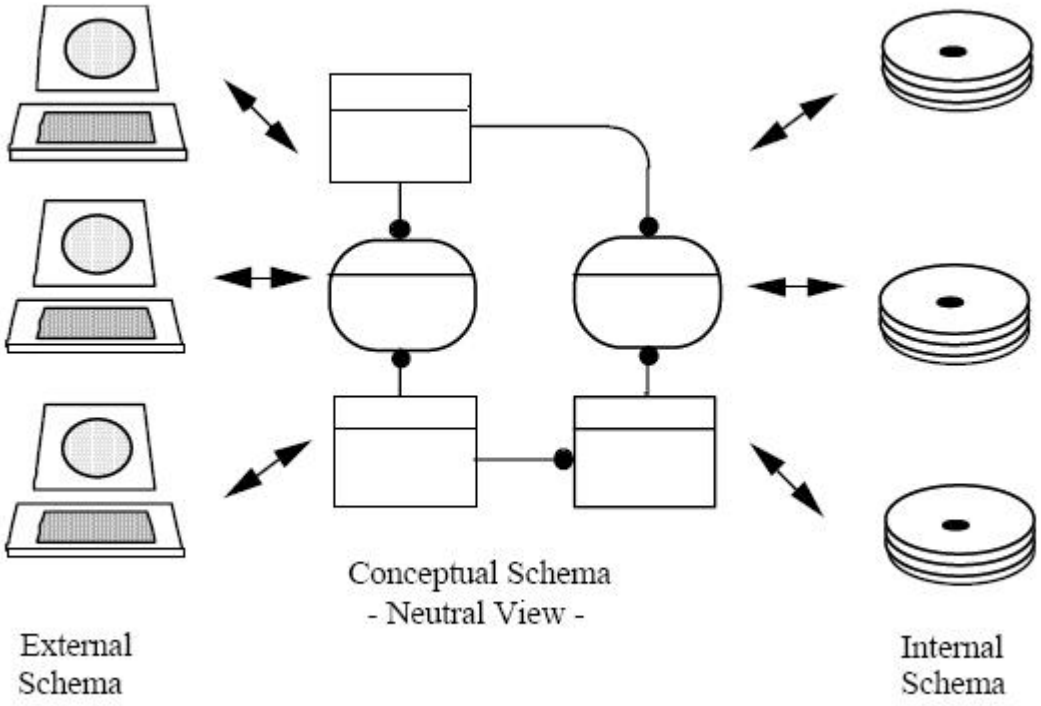
A relationship in which instances of both entities represent the same real or abstract thing. One entity (generic entity) represents the complete set of things, the other (category entity) represents a sub-type or sub-classification of those things. The category entity may have one or more characteristics, or a relationship with instances of another entity not shared by all generic entity instances. Each instance of the category entity is simultaneously an instance of the generic entity.

Non-Specific Relationships

A relationship in which an instance of either entity can be related to a number of instances of the other.

IDEF1X Topics

The Three Schema Approach



The three schema approach .

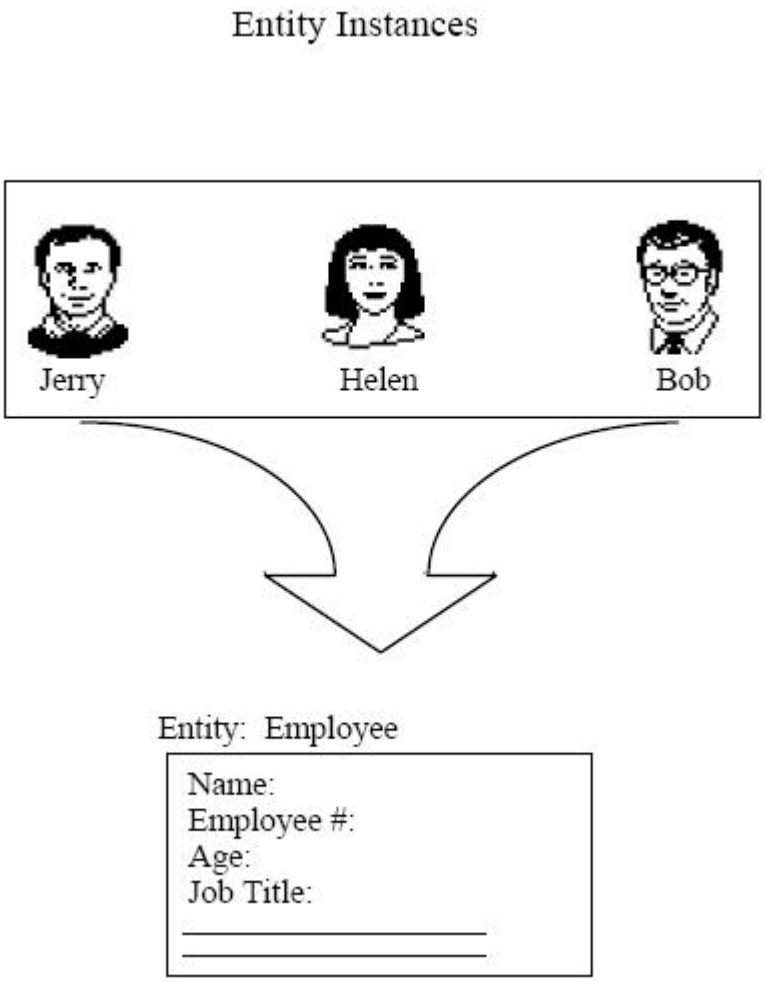
The three-schema approach in software engineering is an approach to building information systems and systems information management, that promotes the conceptual model as the key to achieving data integration.

A schema is a model, usually depicted by a diagram and sometimes accompanied by a language description. The three-schema approach has three types of schemas:

- External schema for user views
- Conceptual schema integrates external schemata
- Internal schema that defines physical storage structures

At the center, the conceptual schema defines the ontology of the concepts as the users think of them and talk about them. The physical schema describes the internal formats of the data stored in the database, and the external schema defines the view of the data presented to the application programs. The framework attempted to permit multiple data models to be used for external schemata.

Modeling Guidelines



Synthesizing an Entity in Phase One – Entity Definition

The modeling process can be divided into five stages of model developing.

Phase Zero - Project Initiation

The objectives of the project initiation phase include:

- Project definition — a general statement of what has to be done, why, and how it will get done.
- Source material — a plan for the acquisition of source material, including indexing and filing.
- Author conventions — a fundamental declaration of the conventions (optional methods) by which the author chooses to make and manage the model.

Phase One – Entity Definition

The objective of this phase is to identify and define the entities that fall within the problem domain being modeled. The first step in this process is the identification of entities.

Phase Two – Relationship Definition

The objective of Phase Two is to identify and define the basic relationships between entities. At this stage of modeling, some relationships may be non-specific and will require additional refinement in subsequent phases. The primary outputs from Phase Two are:

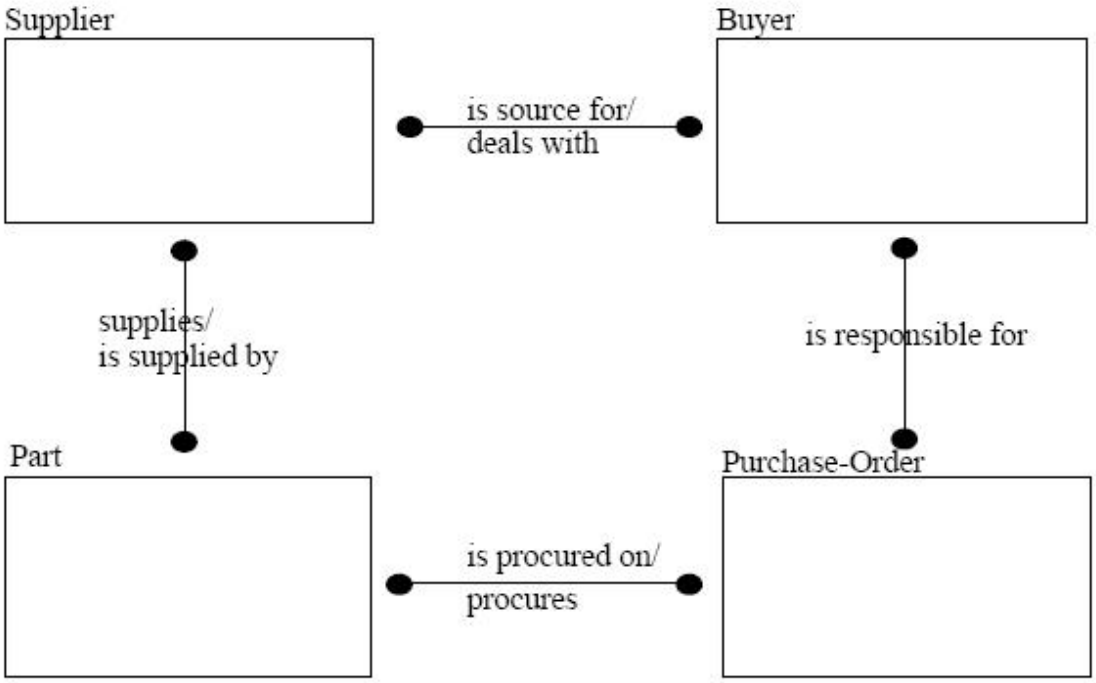
- Relationship matrix
- Relationship definitions
- Entity-level diagrams

Entity-Relationship Matrix Example

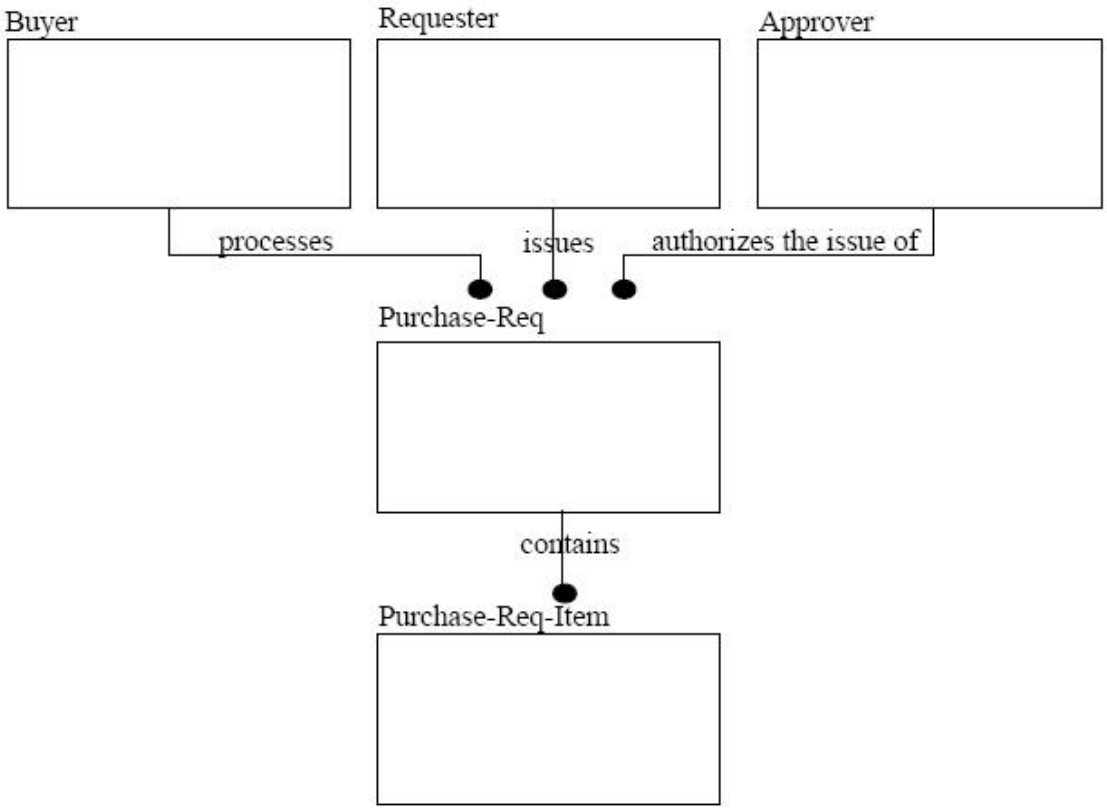
	Buyer	Requester	Approver	Purchase Requisition	Purchase Req. Item
Buyer		X		X	
Requester	X			X	
Approver				X	
Purchase Requisition	X	X	X		X
Purchase Req. Item				X	

An Entity-Relationship Matrix only reflects that a relationship of some kind may exist.

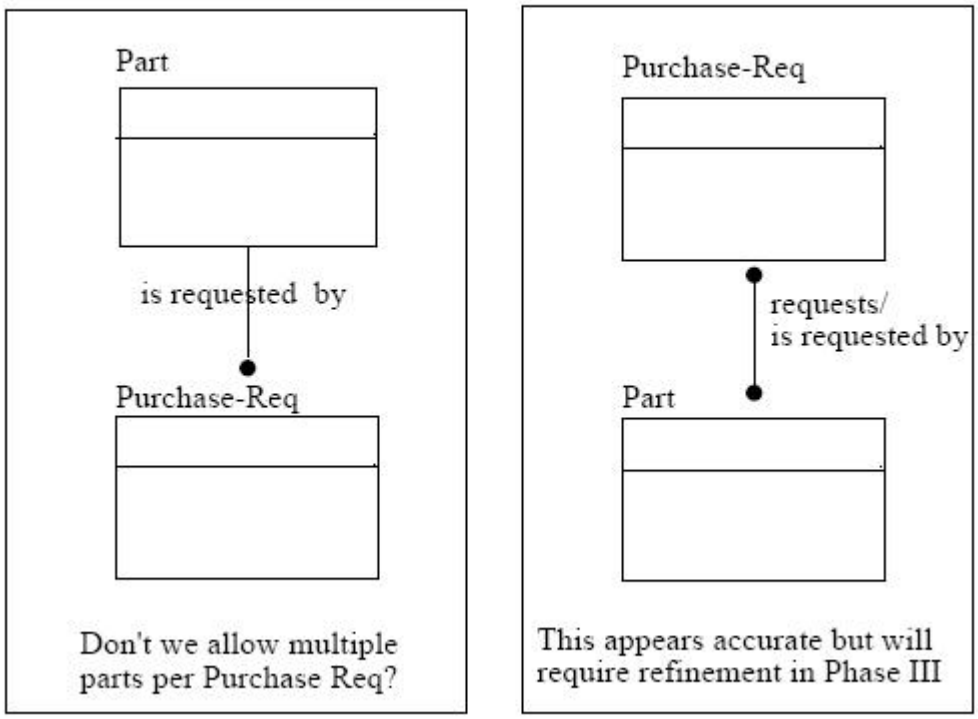
Entity Relationship Matrix



Entity Level Diagram



Entity Level Diagram Example



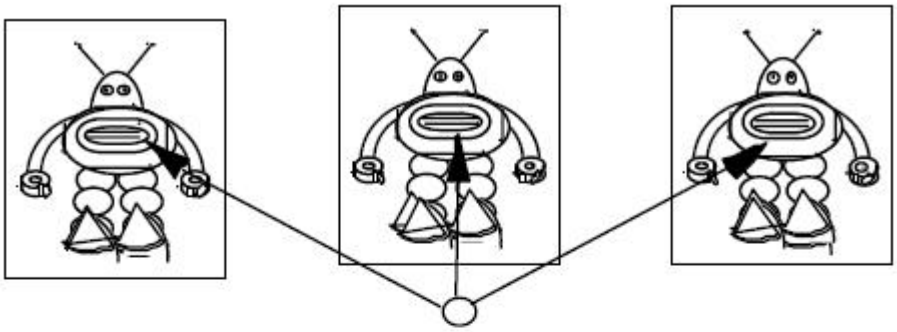
Reference Diagram

Phase Three - Key Definitions

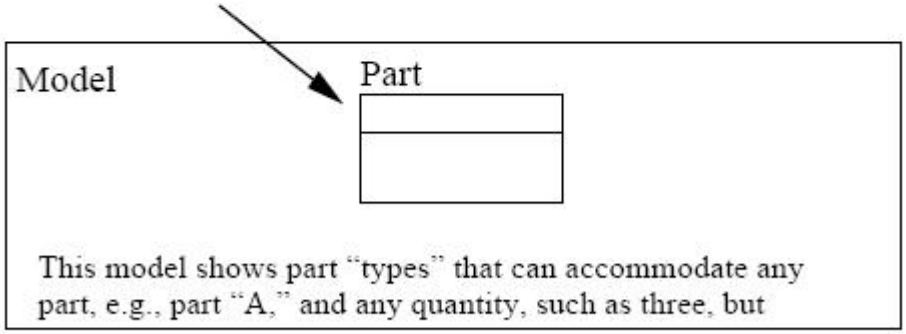
The objectives of Phase Three are to:

- Refine the non-specific relationships from Phase Two.
- Define key attributes for each entity.
- Migrate primary keys to establish foreign keys.
- Validate relationships and keys

Real World

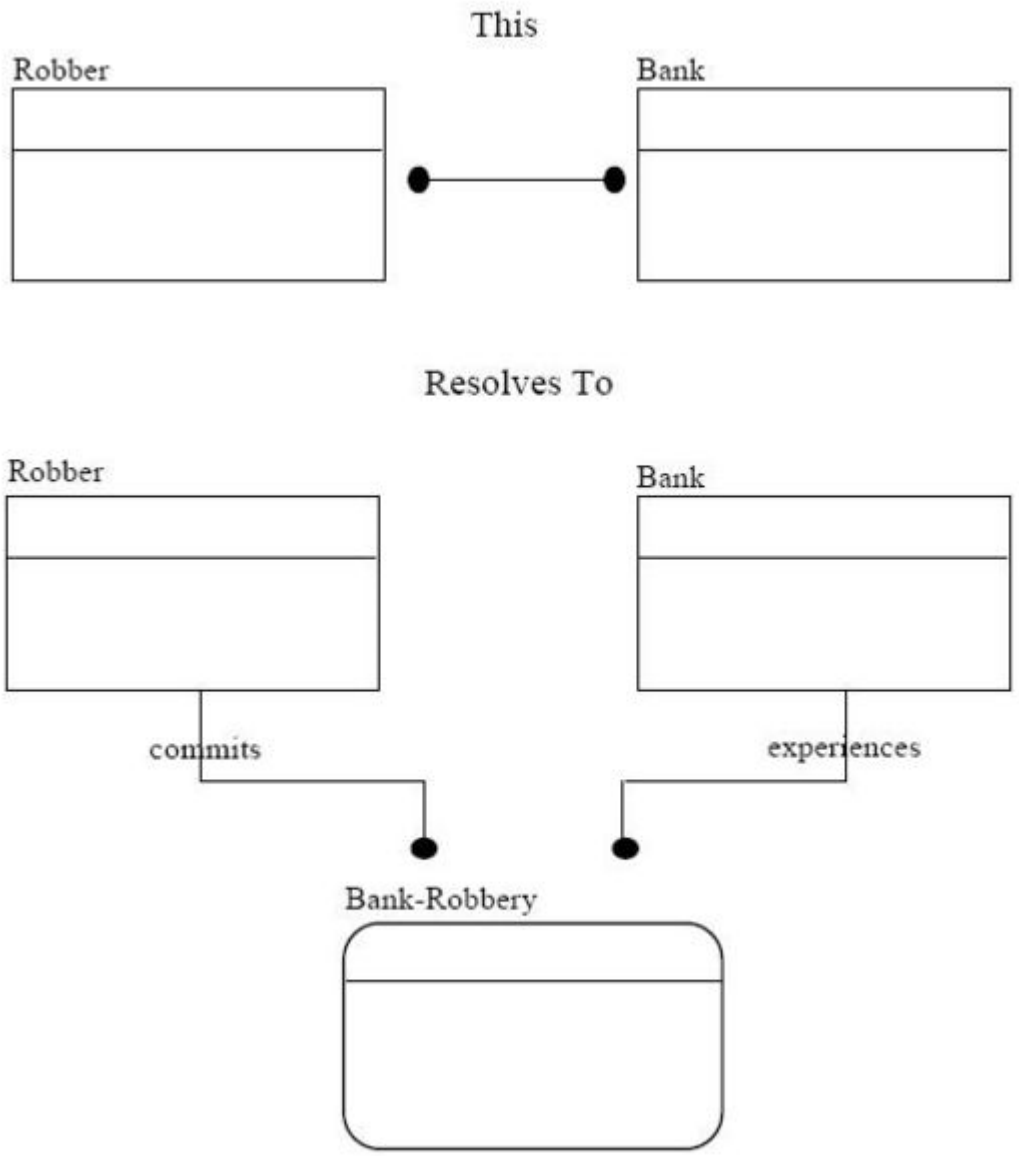


These three parts on the manufactured robot are of the same type. Let's call them part "A." There is no way that we can really tell one part "A" from any other part "A." That is, part "A's"are not uniquely identifiable This situation is modeled as follows:

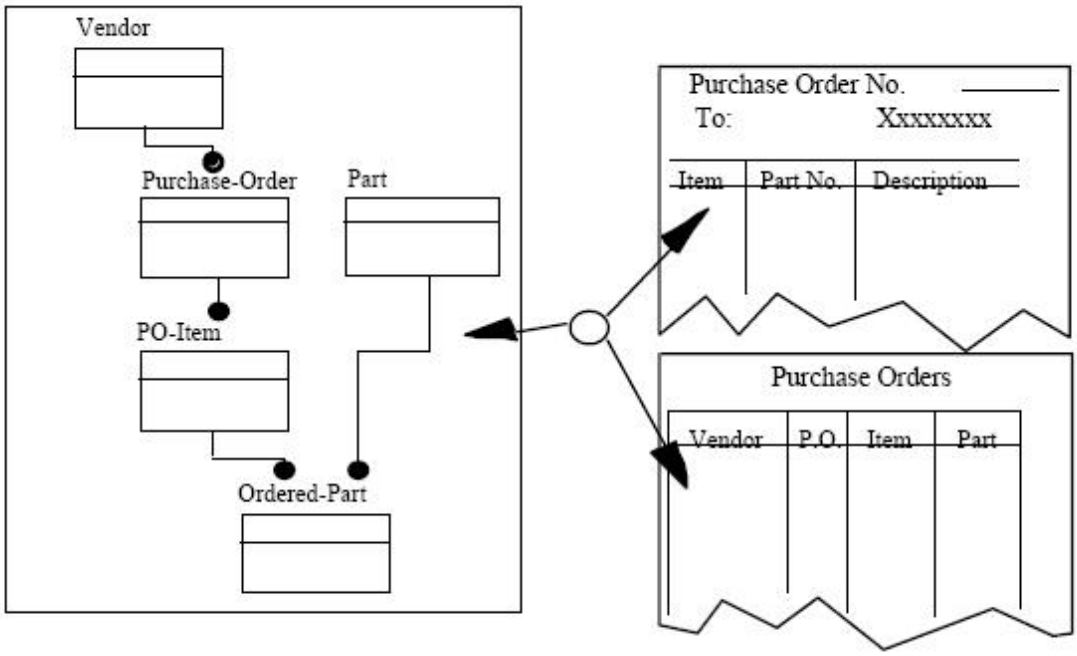


Problem What would happen to the model if serialized control of parts became a requirement?

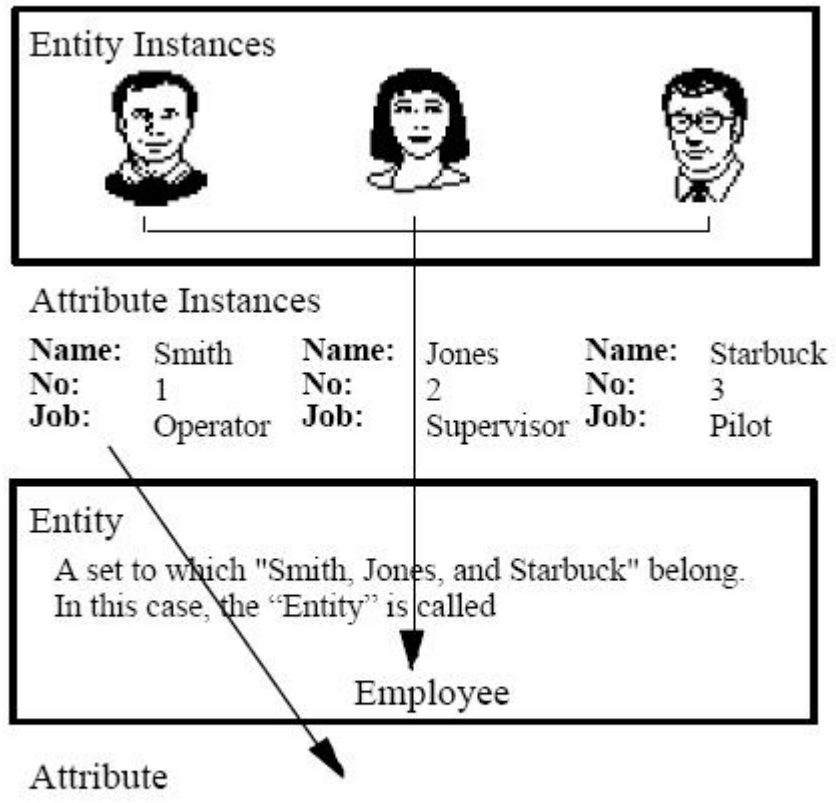
Example Reference Diagram



Non-Specific Relationship Refinement

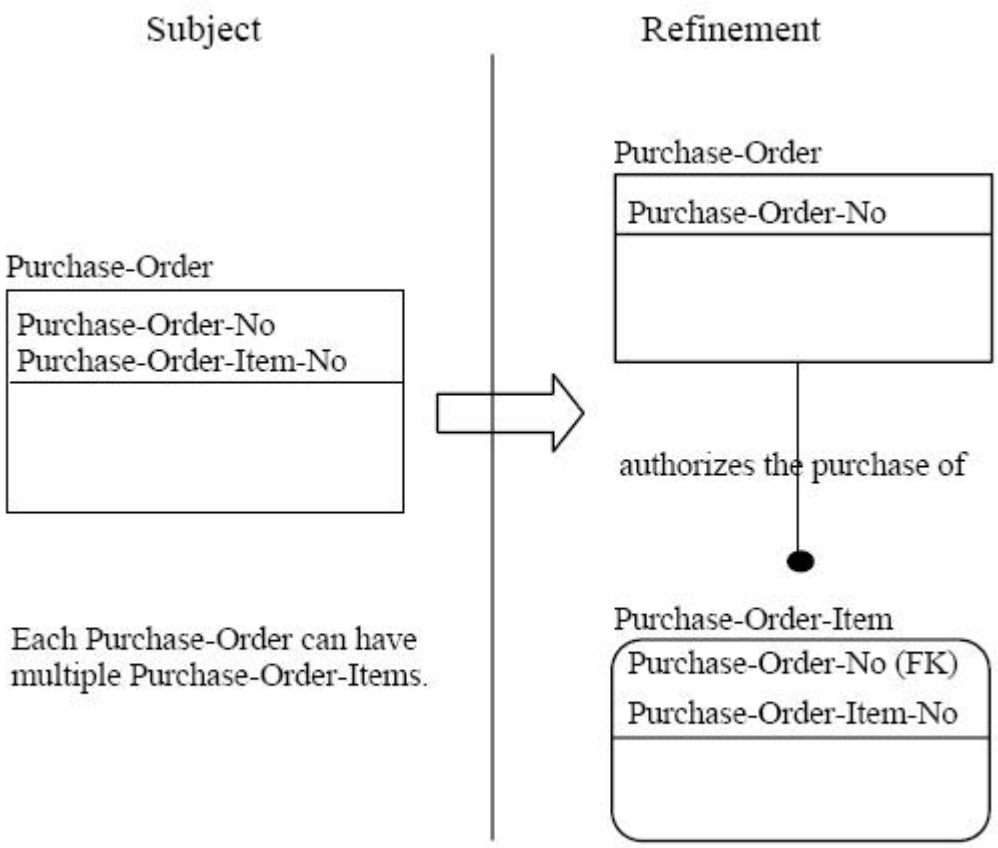


Scope of a Function View

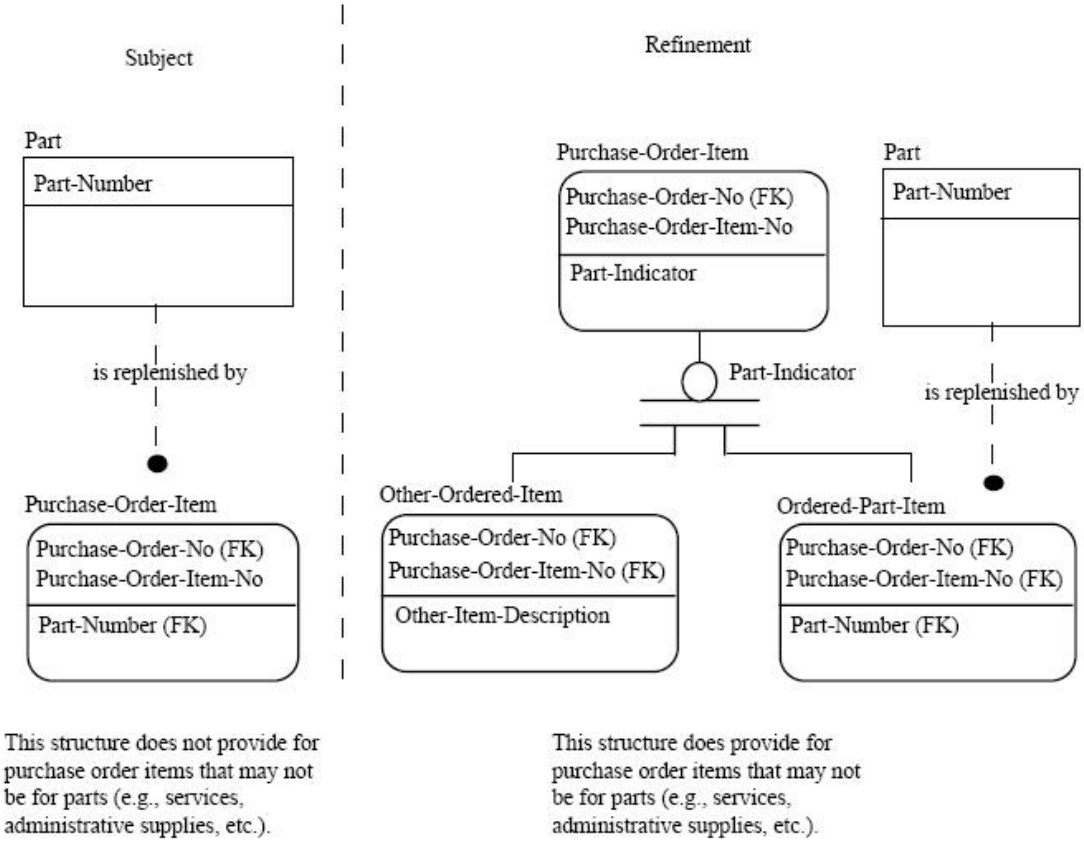


The "items" that commonly describe an Entity, e.g., Employee. In this case, the Attributes "Name, No., and Job" commonly describe each Employee.

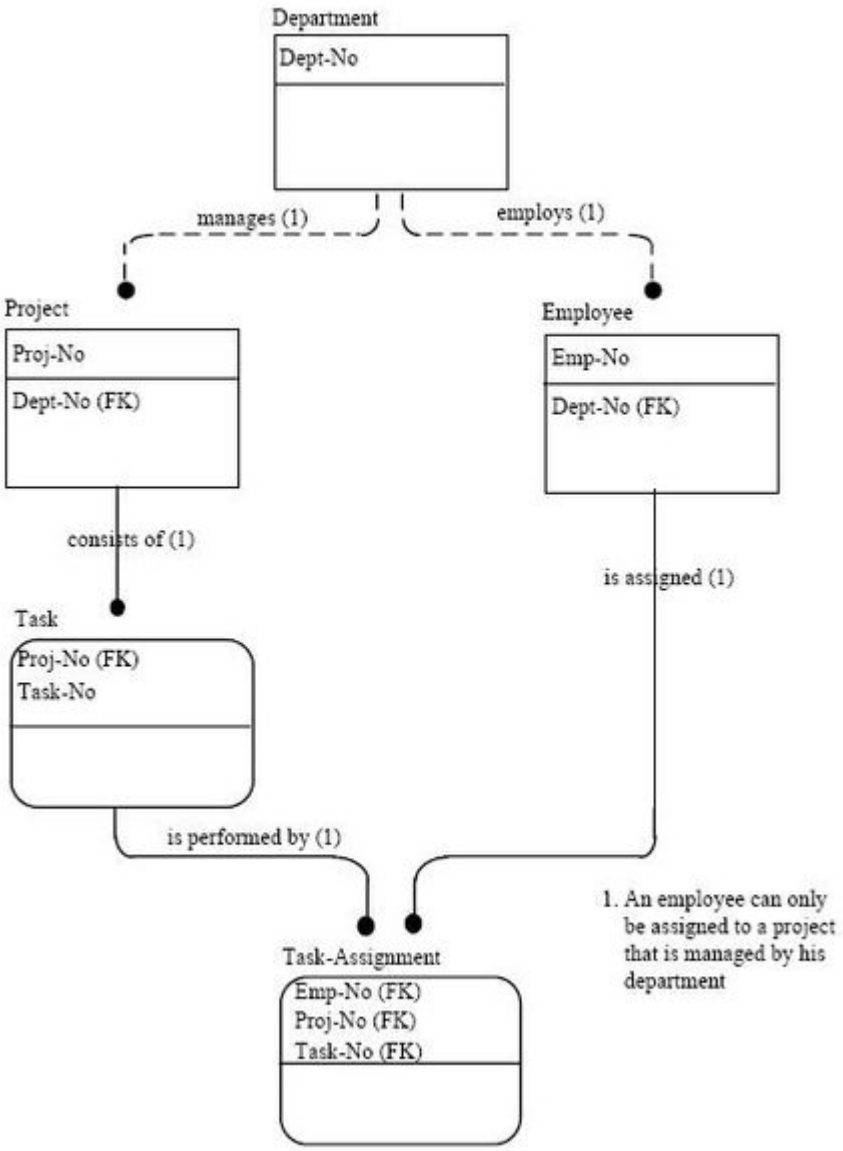
Attribute Examples



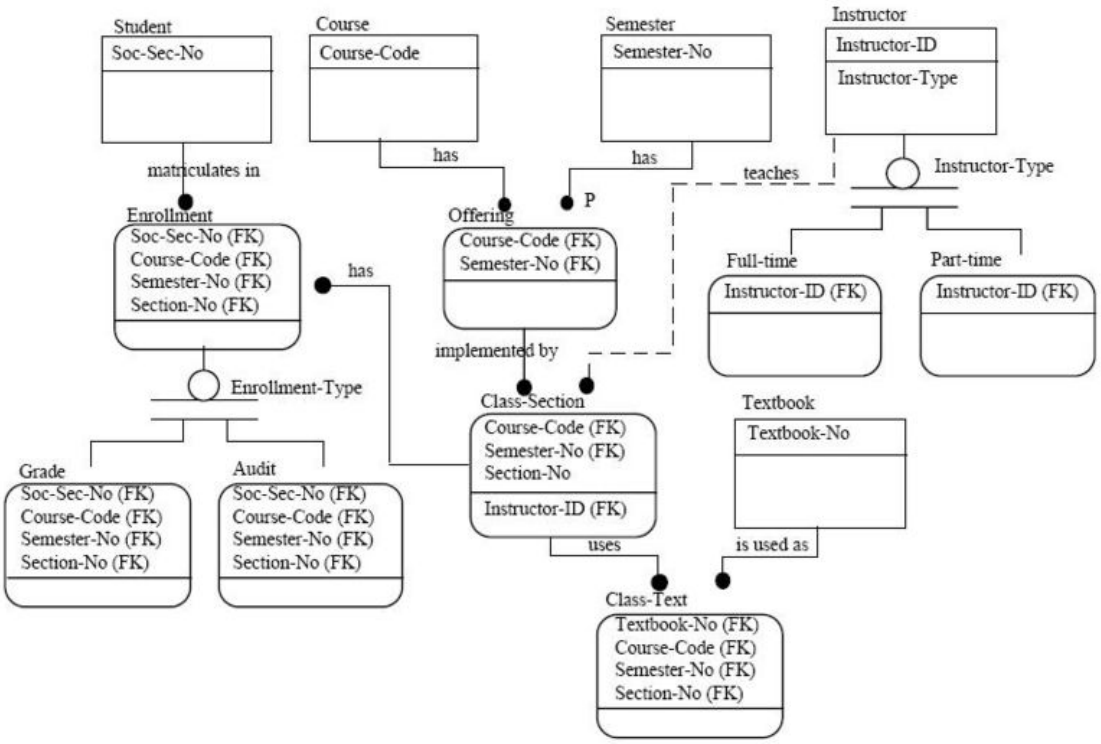
No-Repeat Rule Refinement



Rule Refinement.jpg



Path Assertions

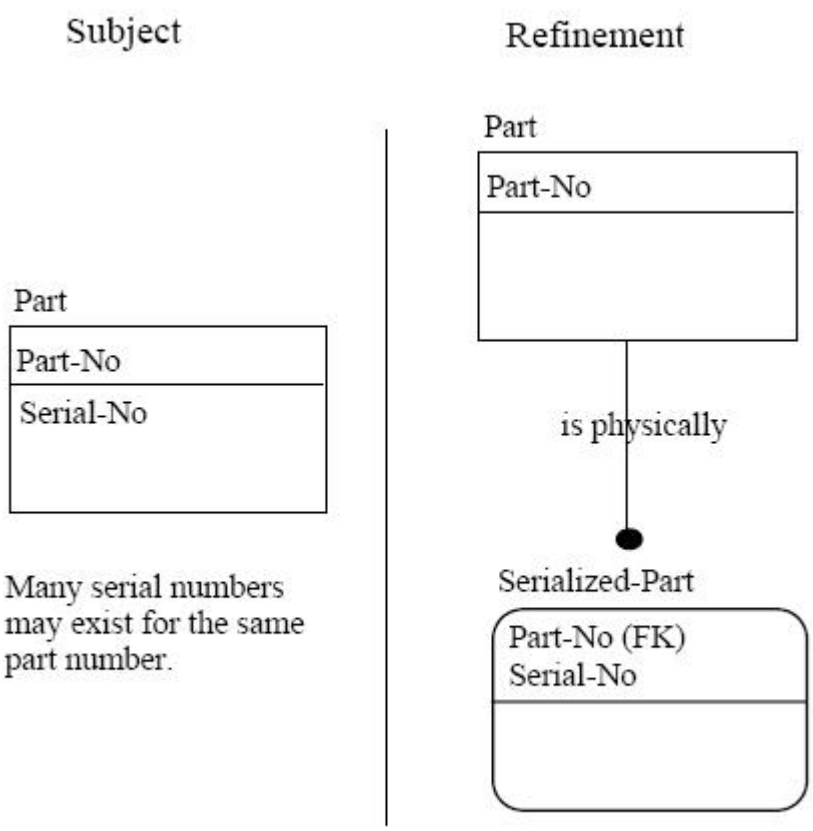


Example of Phase Three Function View Diagram

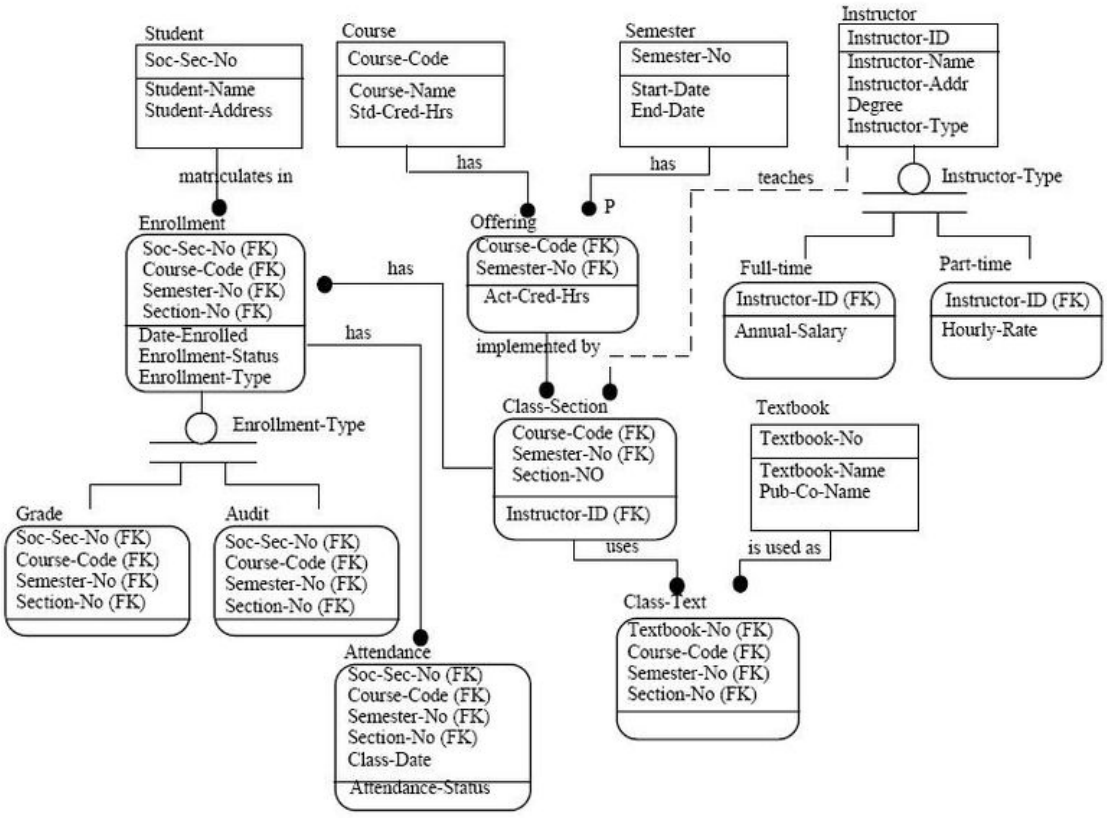
Phase Four - Attribute Definition

Phase Four is the final stage of model developing. The objectives of this plan are to:

- Develop an attribute pool
- Establish attribute ownership
- Define nonkey attributes
- Validate and refine the data structure

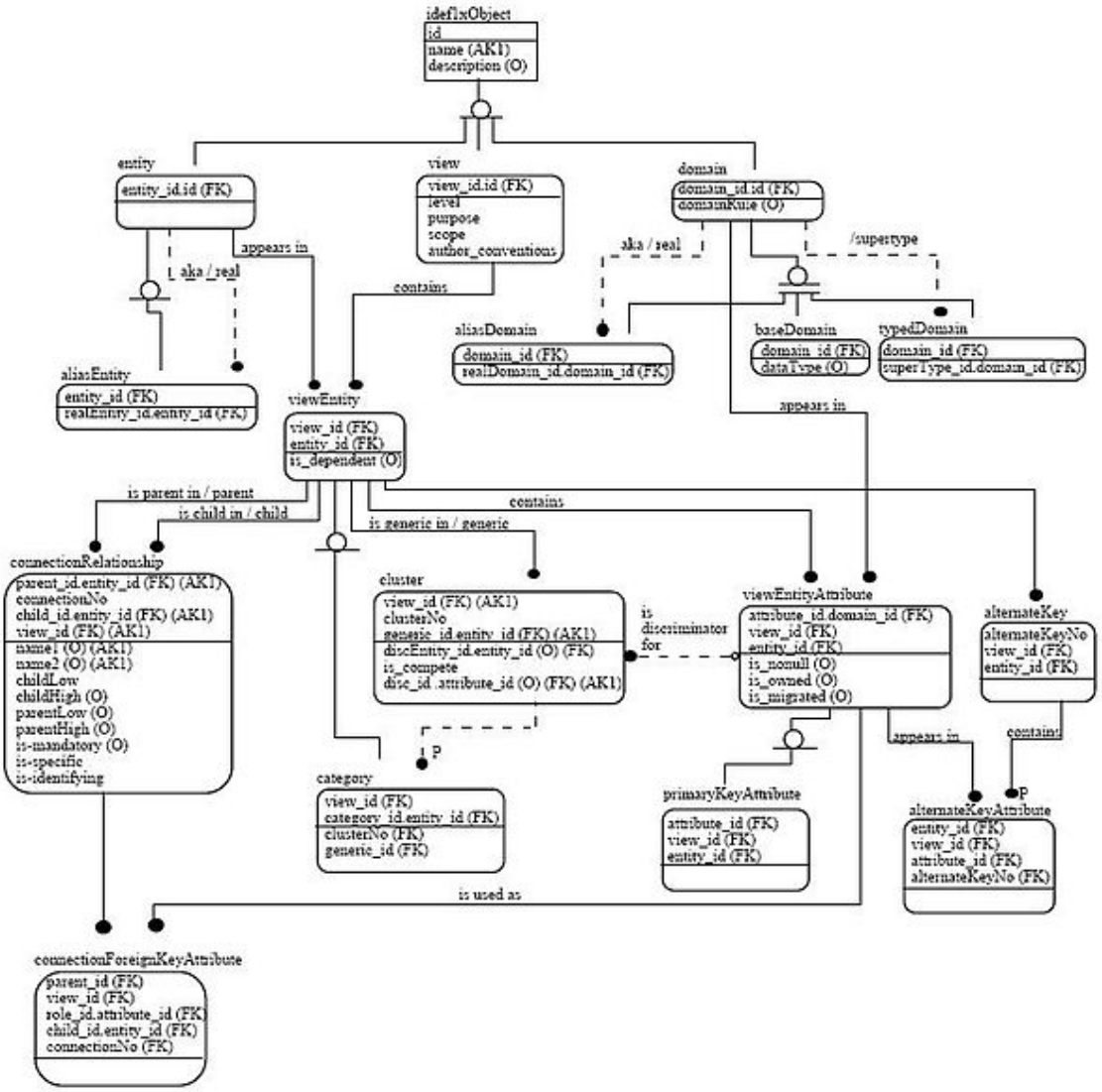


Applying the No Repeat Rule



Example of Phase Four Function

IDEF1X Meta Model



Meta Model of IDEF1X.

IDEF1X can be used to model IDEF1X itself. Such meta models can be used for various purposes, such as repository design, tool design, or in order to specify the set of valid IDEF1X models. Depending on the purpose, somewhat different models result. There is no “one right model”. For example, a model for a tool that supports building models incrementally must allow incomplete or even inconsistent models. The meta model for formalization emphasizes alignment with the concepts of the formalization. Incomplete or inconsistent models are not provided for. There are two important limitations on meta models. First, they specify syntax, not semantics. Second, a meta model must be supplemented with constraints in natural or formal language. The formal theory of IDEF1X provides both the semantics and a means to precisely express the necessary constraints.

A meta model for IDEF1X is given here. The name of the view is mm. The domain hierarchy and constraints are also given. The constraints are expressed as sentences in the formal theory of the meta model. The meta model informally defines the set of valid IDEF1X models in the usual way. The meta model also formally defines the set of valid IDEF1X models in the following way. The meta model, as an IDEF1X model, has a corresponding formal theory. The semantics of the theory are defined in the standard way. That is, an interpretaion of a theory consists of a domain of individuals and a set of assignments:

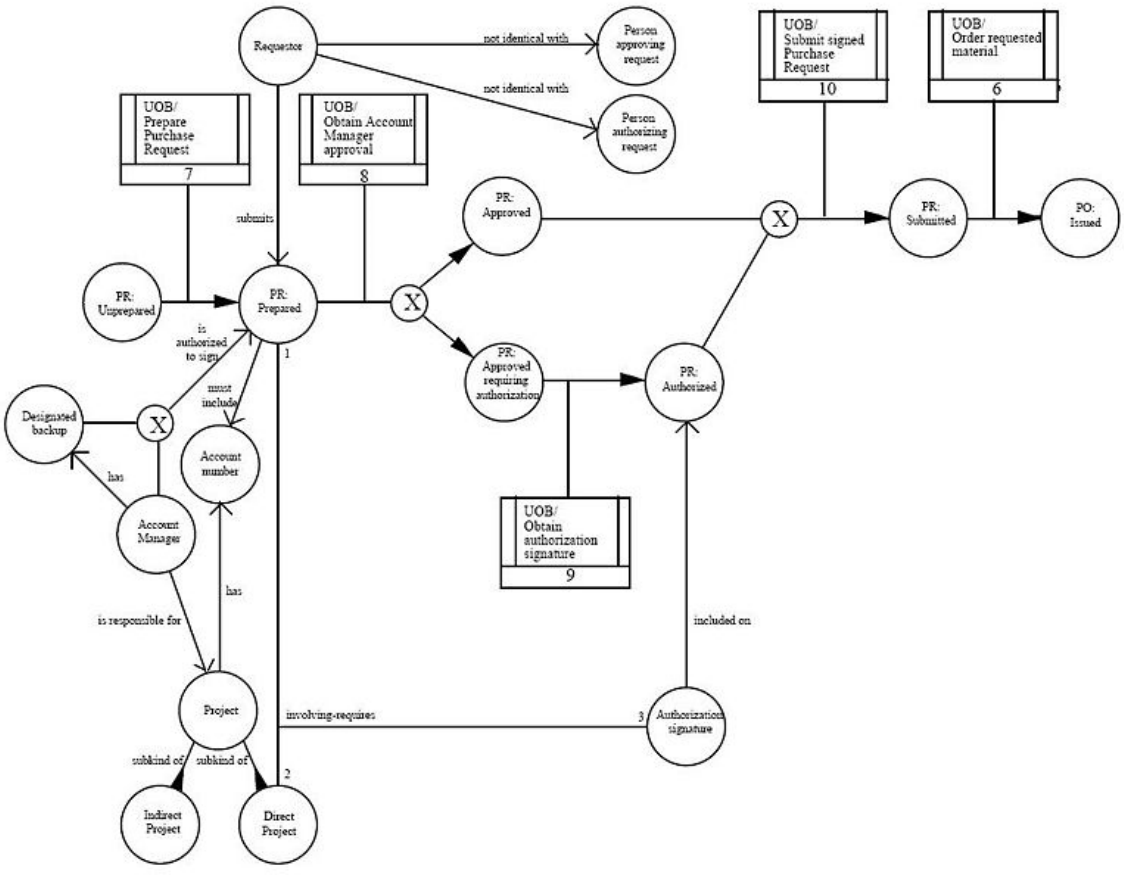
> To each constant in the theory, an individual in the domain is assigned.
> To each n-ary function symbol in the theory, an n-ary function over the domain is assigned.
> To each n-ary predicate symbol in the theory, an n-ary relation over the domain is assigned.

In the intended interpretation, the domain of individuals consists of views, such as production; entities, such as part and vendor; domains, such as qty_on_hand; connection relationships; category clusters; and so on. If every axiom in the theory is true in the interpretation, then the interpretation is called a model for the theory. Every model for the IDEF1X theory corresponding to the IDEF1X meta model and its constraints is a valid IDEF1X model.

Chapter 4

IDEF3 & IDEF5

IDEF3



Example of an Enhanced Transition Schematic, modelled with IDEF3.

IDEF3, officially named a *Integrated DEFinition for Process Description Capture Method*, is a business process modelling method complementary to IDEF0. The IDEF3

method is a scenario-driven process flow description capture method intended to capture the knowledge about how a particular system works.

The IDEF3 method provides modes to represent both

- Process Flow Descriptions to capture the relationships between actions within the context of a specific scenario, and
- Object State Transition to capture the description of the allowable states and conditions.

This method is part of the IDEF family of modeling languages in the field of systems and software engineering.

Overview

One of the primary mechanisms used for descriptions of the world is relating a story in terms of an ordered sequence of events or activities. The IDEF3 Process Description Capture Method was created to capture descriptions of sequences of activities, which is considered the common mechanisms to describe a situation or process. The primary goal of IDEF3 is to provide a structured method by which a domain expert can express knowledge about the operation of a particular system or organization. Knowledge acquisition is enabled by direct capture of assertions about real-world processes and events in a form that is most natural for capture. IDEF3 supports this kind of knowledge acquisition by providing a reliable and wellstructured approach for process knowledge acquisition, and an expressively, yet easy-to-use, language for information capture and expression.

Motives for the development of IDEF3 were the need:

- to speed up the process of business systems modeling,
- to provides mechanisms to describe this data life cycle information,
- to supported project management techniques by an automated tool,
- to provide the concepts, syntax, and procedures for building system requirements descriptions, and
- to work well both independently and jointly with other methods which address different areas of concentration (e.g., the IDEF0 Function Modeling method) as a complementary addition to the IDEF method family.

History

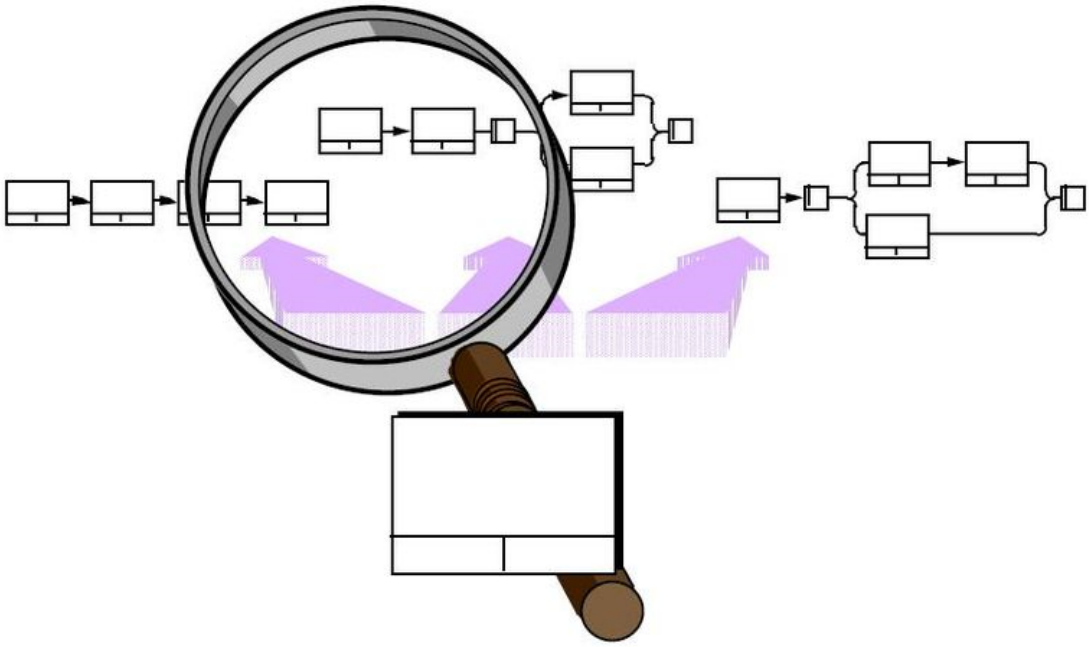
The original IDEFs were developed since the mid-1970s for the purpose of enhancing communication among people who needed to decide how their existing systems were to be integrated. IDEF0 was designed to allow a graceful expansion of the description of a systems' functions through the process of function decomposition and categorization of the relations between functions (i.e., in terms of the Input, Output, Control, and Mechanism classification). IDEF1 was designed to allow the description of the

information that an organization deems important to manage in order to accomplish its objectives.

The third IDEF (IDEF2) was originally intended as a user interface modeling method. However, since the Integrated Computer-Aided Manufacturing (ICAM) Program needed a simulation modeling tool, the resulting IDEF2 was a method for representing the time varying behavior of resources in a manufacturing system, providing a framework for specification of math model based simulations. It was the intent of the methodology program within ICAM to rectify this situation but limitation of funding did not allow this to happen. As a result, the lack of a method which would support the structuring of descriptions of the user view of a system has been a major shortcoming of the IDEF system. The basic problem from a methodology point of view is the need to distinguish between a description of what a system (existing or proposed) is supposed to do and a representative simulation model that will predict what a system will do. The latter was the focus of IDEF2, the former is the focus of IDEF3.

IDEF3 basic concepts

Descriptions and models



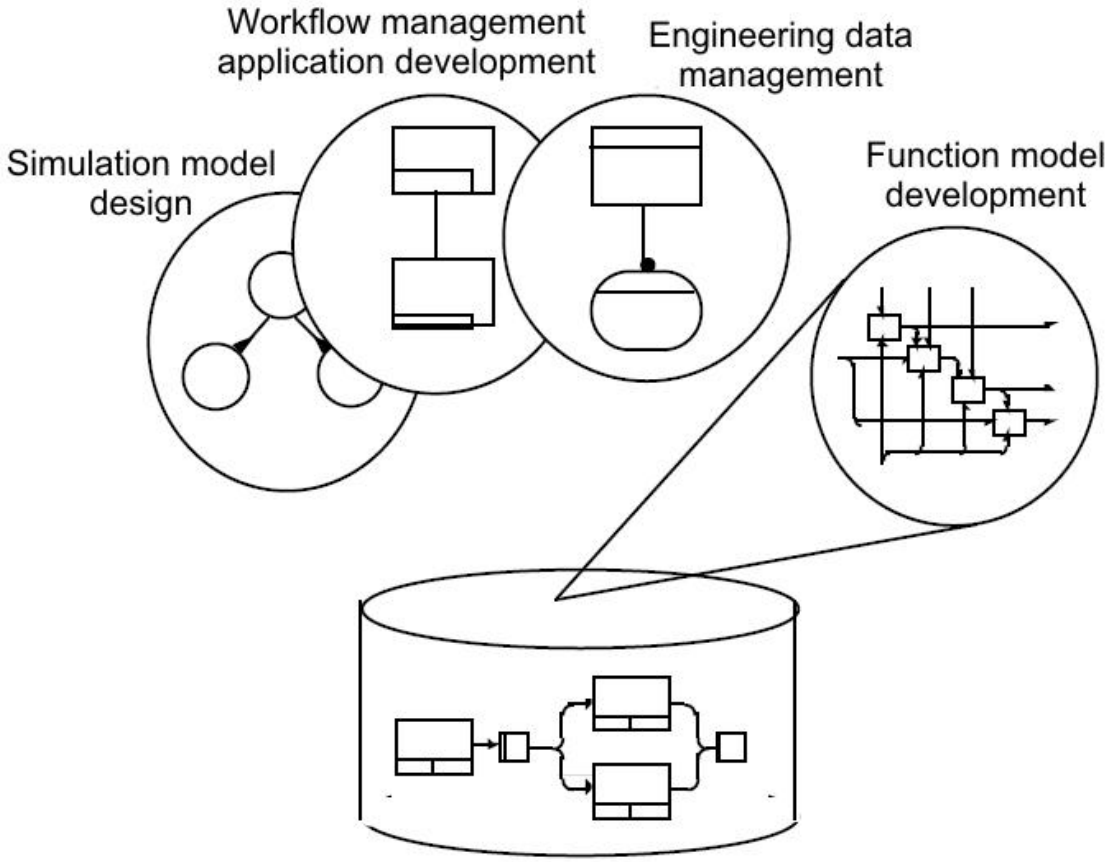
IDEF3 want to offer alternative descriptions of the same process from multiple viewpoints on the process.

The distinction between descriptions and models, though subtle, is an important one in IDEF3, and both have a precise technical meaning.

- The term description is used as a reserved technical term to mean records of empirical observations; that is, descriptions record knowledge that originates in or is based on observations or experience.
- The term model is used to mean an idealization of an entity or state of affairs. That is, a model constitutes an idealized system of objects, properties, and relations that is designed to imitate, in certain relevant respects, the character of a given real-world system. Frictionless planes, perfectly rigid bodies, the assumption of point mass, and so forth are representative examples of models.

The power of a model comes from its ability to simplify the real-world system it represents and to predict certain facts about that system by virtue of corresponding facts within the model. Thus, a model is a designed system in its own right. Models are idealized systems known to be incorrect but assumed to be close enough to provide reliable predictors for the predefined areas of interest within a domain. A description, on the other hand, is a recording of facts or beliefs about something within the realm of an individual's knowledge or experience. Such descriptions are generally incomplete; that is, the person giving a description may omit facts that he or she believes are irrelevant, or which were forgotten in the course of describing the system. Descriptions may also be inconsistent with respect to how others have observed situations within the domain. IDEF3 accommodates these possibilities by providing specific features enabling the capture and organization of alternative descriptions of the same scenario or process.

Description capture



IDEF focus on Description Capture, that enables reuse.

Modeling necessitates taking additional steps beyond description capture to resolve conflicting or inconsistent views. This, in turn, generally requires modelers to select or create a single viewpoint and introduce artificial modeling approximations to fill in gaps where no direct knowledge or experience is available. Unlike models, descriptions are not constrained by idealized, testable conditions that must be satisfied, short of simple accuracy.

The purpose of description capture may be simply to record and communicate process knowledge or to identify inconsistencies in the way people understand how key processes actually operate. By using a description capture method users need not learn and apply conventions forcing them to produce executable models (e.g., conventions ensuring accuracy, internal consistency, logical coherence, non-redundancy, completeness). Forcing users to model requires them to adopt a model design perspective and risk producing models that do not accurately capture their empirical knowledge of the domain.

Scenarios

The notion of a scenario or story is used as the basic organizing structure for IDEF3 Process Descriptions. A scenario can be thought of as a recurring situation, a set of situations that describe a typical class of problems addressed by an organization or system, or the setting within which a process occurs. Scenarios establish the focus and boundary conditions of a description. Using scenarios in this way exploits the tendency of humans to describe what they know in terms of an ordered sequence of activities within the context of a given scenario or situation. Scenarios also provide a convenient vehicle to organize collections of process-centered knowledge.

Process-Centered Views

IDEF3 Process Schematics are the primary means for capturing, managing, and displaying process-centered knowledge. These schematics provide a graphical medium that helps domain experts and analysts from different application areas communicate knowledgeabout processes. This includes knowledge about events and activities, the objects that participate in those occurrences, and the constraining relations that govern the behavior of an occurrence.

Object-Centered Views

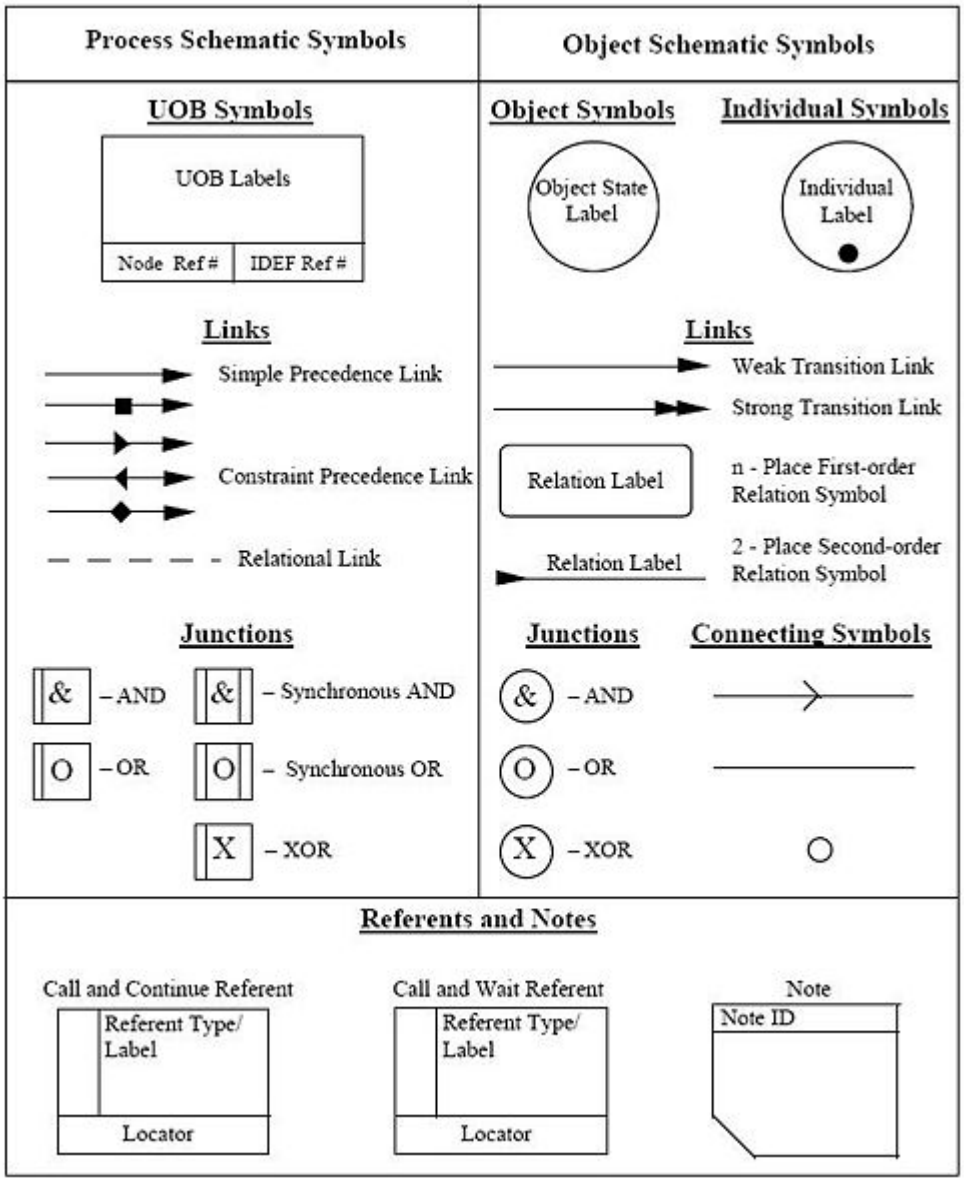
IDEF3 Object Schematics capture, manage, and display object-centered descriptions of a process—that is, information about how objects of various kinds are transformed into other kinds of things through a process, how objects of a given kind change states through a process, or context-setting information about important relations among objects in a process.

IDEF3 process description language

IDEF3 descriptions are developed from two different perspectives: process-centered and object-centered. Because these approaches are not mutually exclusive, IDEF3 allows cross-referencing between them to represent complex process descriptions.

Process Schematics

Process schematics tend to be the most familiar and broadly used component of the IDEF3 method. These schematics provide a visualization mechanism for processcentered descriptions of a scenario. The graphical elements that comprise process schematics include Unit of Behavior (UOB) boxes, precedence links, junctions, referents, and notes. The building blocks here are:



Symbols Used for IDEF3 Process Description Schematics.

- Unit of Behavior (UOB) boxes
- Links : Links are the glue that connect UOB boxes to form representations of dynamic processes.
- Simple Precedence Links : Precedence links express temporal precedence relations between instances of one UOB and those of another.
- Activation Plots : Activation plots are used to represent activations.
- Dashed Links : Dashed links carry no predefined semantics.
- Link Numbers : All links have an elaboration and unique link numbers.
- Activation Semantics for Nonbranching Process Schematics.
- Junctions : Junctions in IDEF3 provide a mechanism to specify the logic of process branching.

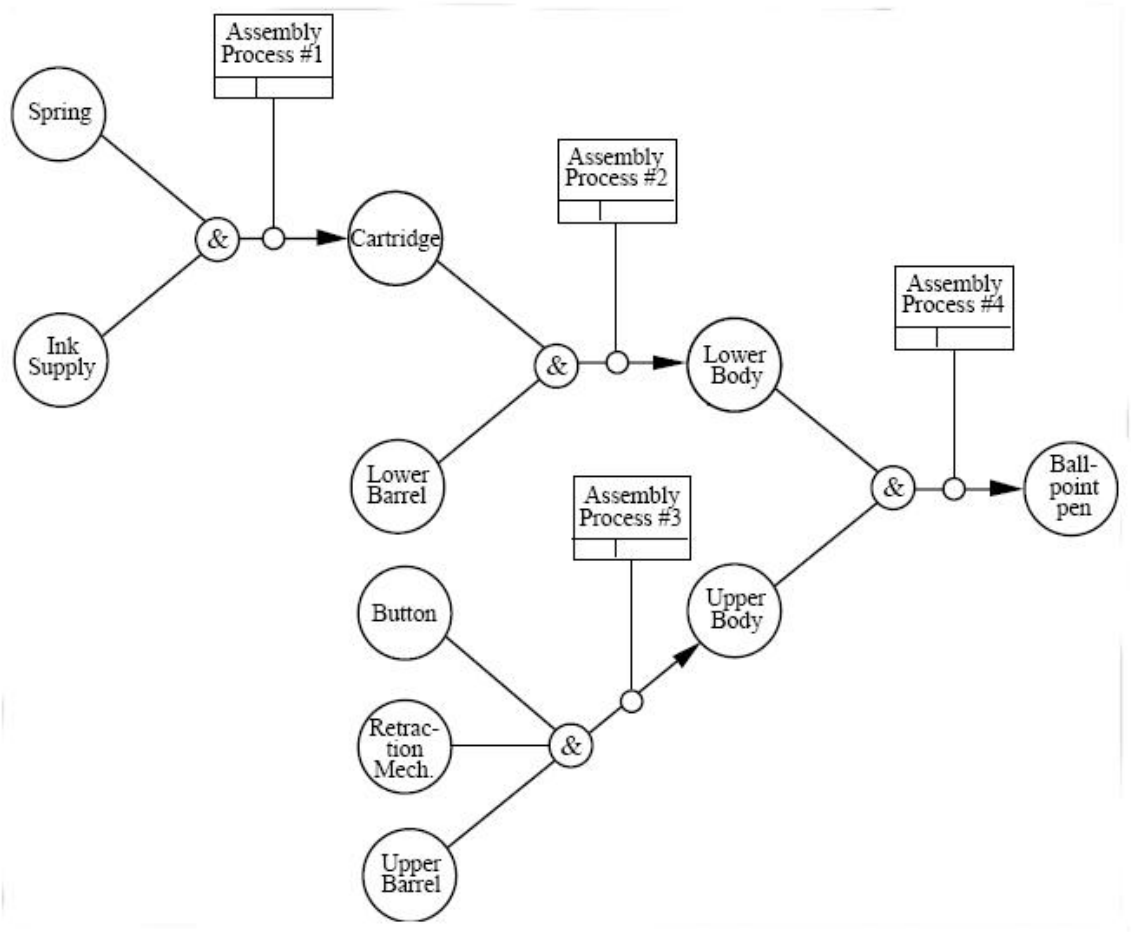
- UOB Decompositions : Elaborations capture and structure detailed knowledge about processes.
- UOB Reference Numbering Scheme : A UOB box number is assigned to each UOB box in an IDEF3 Process Description.
- Partial Descriptions : UOB boxes are joined together by links. Because of the description capture focus of IDEF3, it is possible to conceive of UOBs without links to other parts of an IDEF3 schematic.
- Referents : Referents enhance understanding, provide additional meaning, and simplify the construction (i.e., minimize clutter) of both process schematics and object schematics.

Object Schematics

IDEF offers a series of building blocks to express detailed object-centered process information; that is, information about how objects of various kinds are transformed into other kinds of things through a process, or how objects of a given kind change states through a process.

- Objects : An object of a certain kind, like a chassis, will be represented simply by a circle containing an appropriate label.
- Object States : A certain kind of object being in a certain state will be represented by a circle with a label that captures the kind itself and a corresponding state, representing thereby the type, or class, of objects that are in that state.
- Object schematics : The construction of complex representations built from kind symbols and object state symbols.
- Transition Schematics : The first and most basic construct is the basic state transition schematic or simply, transition schematic.

IDEF5



Example of an IDEF5 Composition Schematic for a Ballpoint Pen.

IDEF5 (*Integrated Definition for Ontology Description Capture Method*) is a software engineering method to develop and maintain usable, accurate, domain ontologies. This standard is part of the IDEF family of modeling languages in the field of software engineering.

Overview

In the field of computer science ontologies are used to capture the concept and objects in a specific domain, along with associated relationships and meanings. In addition, ontology capture helps coordinate projects by standardizing terminology and creates opportunities for information reuse. The lDEF5 Ontology Capture Method has been

developed to reliably construct ontologies in a way that closely reflects human understanding of the specific domain.

In the IDEF5 method, an ontology is constructed by capturing the content of certain assertions about real-world objects, their properties, and their interrelationships and representing that content in an intuitive and natural form. The IDEF5 method has three main components:

- A graphical language to support conceptual ontology analysis
- A structured text language for detailed ontology characterization, and
- A systematic procedure that provides guidelines for effective ontology capture.

IDEF5 Topics

Ontology

In IDEF5 the meaning of the term "ontology" is characterized to include a catalog of terms used in a domain, the rules governing how those terms can be combined to make valid statements about situations in that domain, and the “sanctioned inferences” that can be made when such statements are used in that domain. In every domain, there are phenomena that the humans in that domain discriminate as (conceptual or physical) objects, associations, and situations. Through various language mechanisms, we associate definite descriptors (e.g., names, noun phrases, etc.) to that phenomena.

Central Concepts of Ontology

The construction of ontologies for human engineered systems is the focus of the IDEF5. In the context of such systems, the nature of ontological knowledge involves several modifications to the more traditional conception. The first of these modifications has to do with the notion of a kind. Historically, a kind is an objective category of objects that are bound together by a common nature, a set of properties shared by all and only the members of the kind.

While there is an attempt to divide the world at its joints in the construction of enterprise ontologies, those divisions are not determined by the natures of things in the enterprise so much as the roles those things are to play in the enterprise from some perspective or other. Because those roles might be filled in any of a number of ways by objects that differ in various ways, and because legitimate perspectives on a domain can vary widely, it is too restrictive to require that the instances of each identifiable kind in an enterprise share a common nature, let alone that the properties constituting that nature be essential to their bearers. Consequently, enterprise ontologies require a more flexible notion of kind.

Ontology development process

Ontology development requires extensive iterations, discussions, reviews, and introspection. Knowledge extraction is usually a discovery process and requires considerable introspection. It requires a process that incorporates both significant expert involvement as well as the dynamics of a group effort. Given the open-ended nature of ontological analyses, it is not prudent to adopt a "cookbook" approach to ontology development. In brief, the IDEF5 ontology development process consists of the following five activities:

1. *Organizing and Scoping*: This activity involves establishing the purpose, viewpoint, and context for the ontology development project and assigning roles to the team members.
2. *Data Collection*: This activity involves acquiring the raw data needed for ontology development.
3. *Data Analysis*: This activity involves analyzing the data to facilitate ontology extraction.
4. *Initial Ontology Development*: This activity involves developing a preliminary ontology from the acquired data.
5. *Ontology Refinement and Validation*: This activity involves refining and validating the ontology to complete the development process.

Although the above activities are listed sequentially, there is a significant amount of overlap and iteration between the activities.

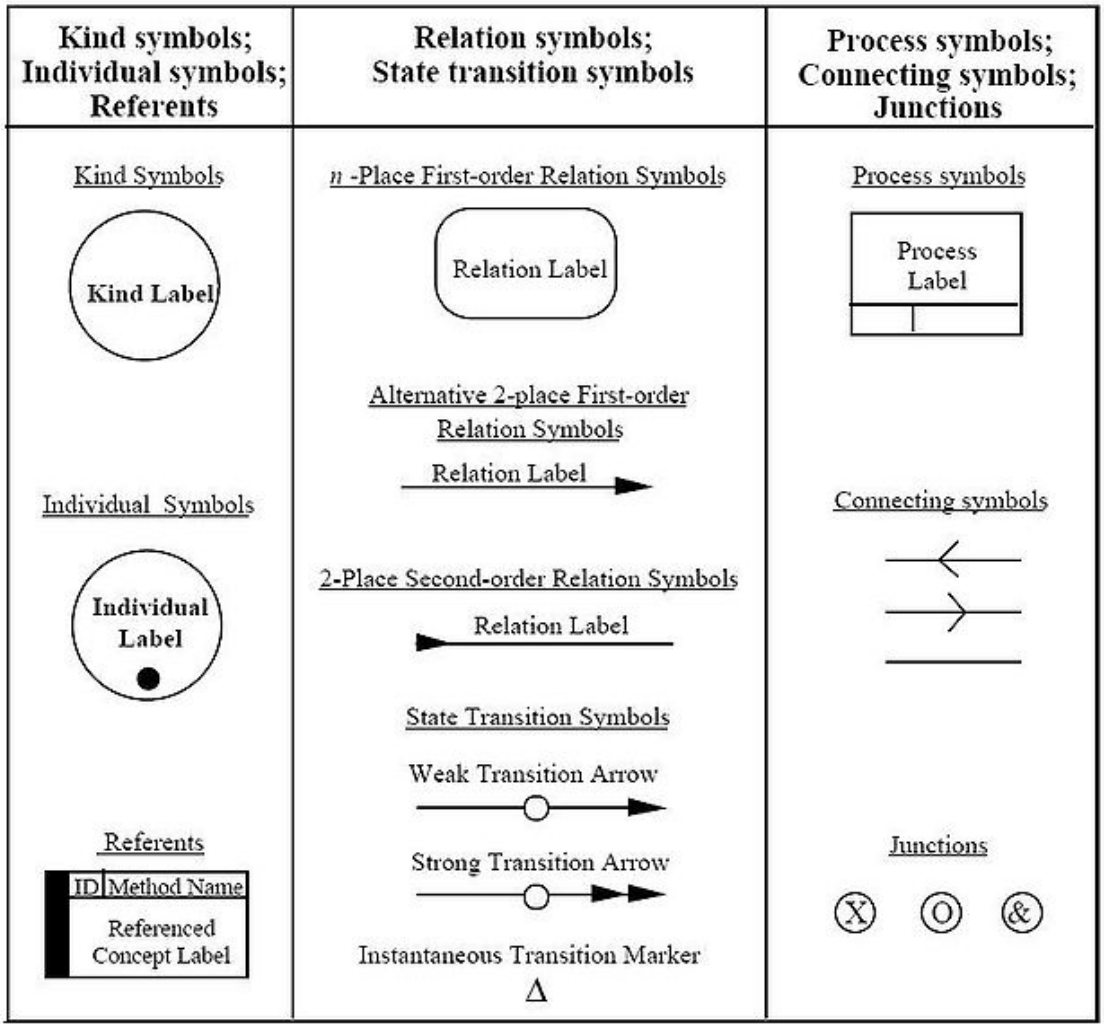
Ontological Analysis

Ontological analysis is accomplished by examining the vocabulary that is used to discuss the characteristic objects and processes that compose the domain, developing rigorous definitions of the basic terms in that vocabulary, and characterizing the logical connections among those terms. The product of this analysis, an ontology, is a domain vocabulary complete with a set of precise definitions, or axioms, that constrain the meanings of the terms sufficiently to enable consistent interpretation of the data that use that vocabulary.

IDEF5 Building blocks

Definitions

Some of the key terms in IDEF5 and the basic IDEF5 Schematic Language Symbols, see figure.:



Kind

Informally, a group of individuals that share some set of distinguished characteristics. More formally, kinds are properties typically expressed by common nouns such as ‘employee’, ‘machine’, and ‘lathe’.

Individual

The most logically basic kind of real world object. Prominent examples include human persons, concrete physical objects, and certain abstract objects such as programs. Unlike objects of higher logical orders such as properties and relations, individuals essentially are not multiply instantiable. Individuals are also known as first-order objects.

Referent

A construct in the IDEF5 elaboration language used to refer to a kind, object, property, relation, or process kind in another ontology or an IDEF model.

Relation

An abstract, general association or connection that holds between two or more objects. Like properties, relations are multiply instantiable. The objects among which a relation holds in a particular instance are known as its arguments.

State

A property, generally indicated by an adjective rather than a common noun, that is characteristic of objects of a certain kind at a certain point within a process. For example, water can be in frozen, liquid, or gaseous states.

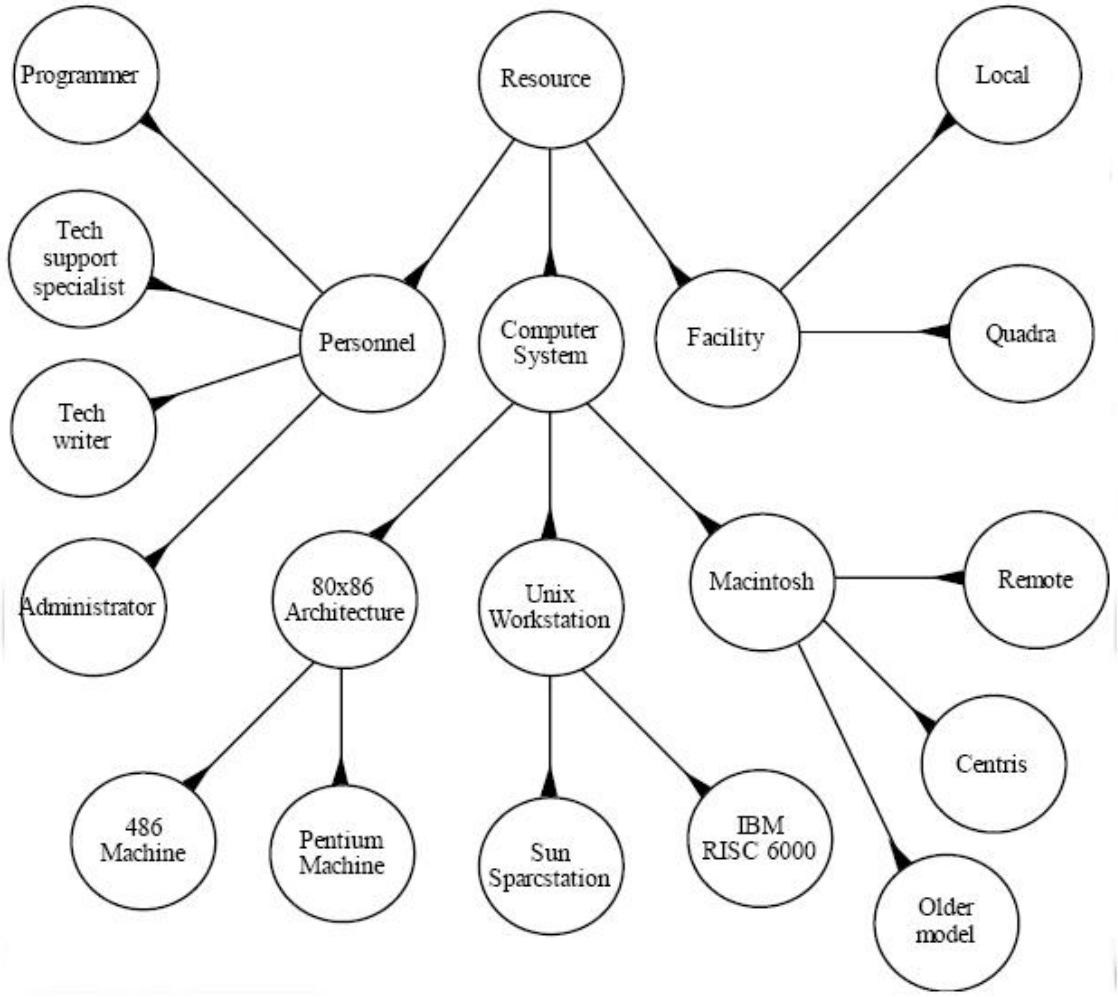
Process

A real world event or state of affairs involving one or more individuals over some (possibly instantaneous) interval of time. Typically, a process involves some sort of change in the properties of one or more of the individuals within the process. Because of the ambiguity in the term “process”, sometimes referred to as process instance.

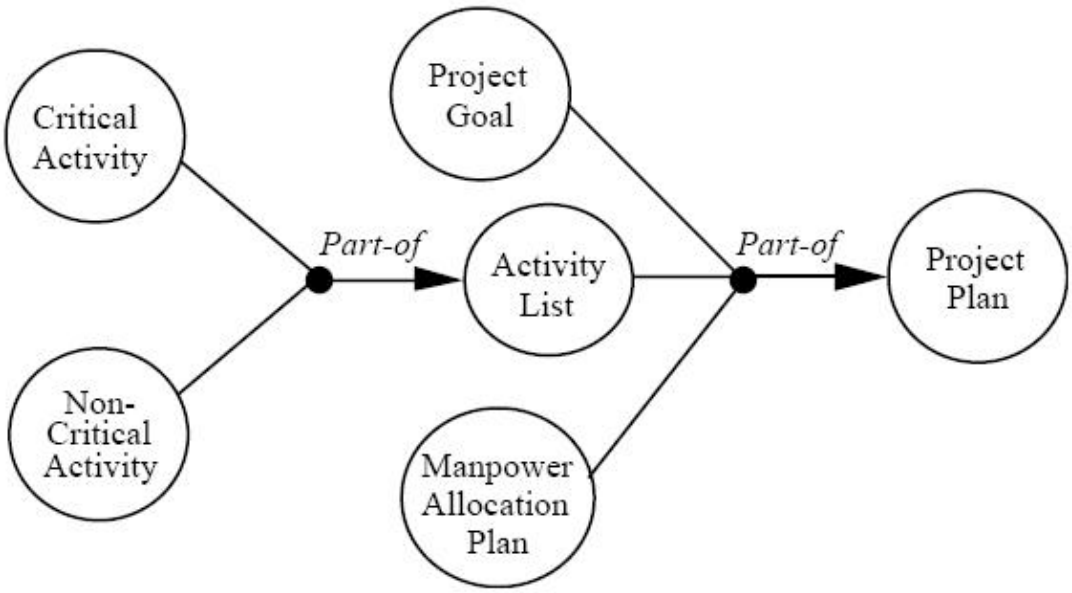
Diagram types

Various diagram types, or schematics, can be constructed in the IDEF5 Schematic Language. The purpose of these schematics, like that of any representation, is to represent information visually. Thus, semantic rules must be provided for interpreting every possible schematic. These rules are provided by outlining the rules for interpreting the most basic constructs of the language, then applying them recursively to more complex constructs. There are four primary schematic types derived from the basic IDEF5 Schematic Language which can be used to capture ontology information directly in a form that is intuitive to the domain expert.

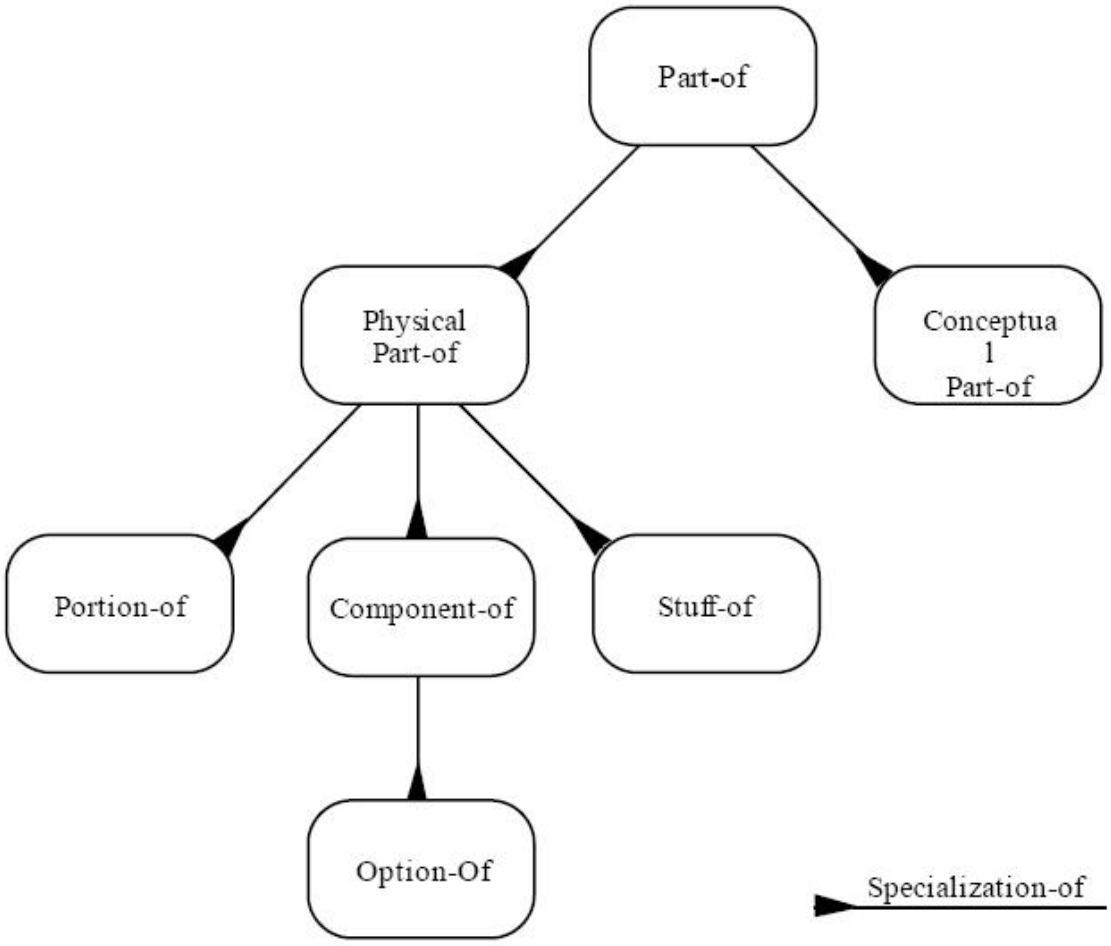
- *Classification Schematics* : Classification schematics provide mechanisms for humans to organize knowledge into logical taxonomies. Of particular merit are two types of classification: description subsumption and natural kind classification.
- *Composition Schematics* : Composition schematics serve as mechanisms to represent graphically the "part-of" relation that is so common among components of an ontology.
- *Relation Schematics* : Relation schematics allow ontology developers to visualize and understand relations among kinds in a domain, and can also be used to capture and display relations between first-order relations.
- *Object State Schematics* : Because there is no clean division between information about kinds and states and information about processes, the IDEF5 schematic language enables modelers to express fairly detailed object-centered process information (i.e., information about kinds of objects and the various states they can be in relative to certain processes). Diagrams built from these constructs are known as Object-State Schematics.



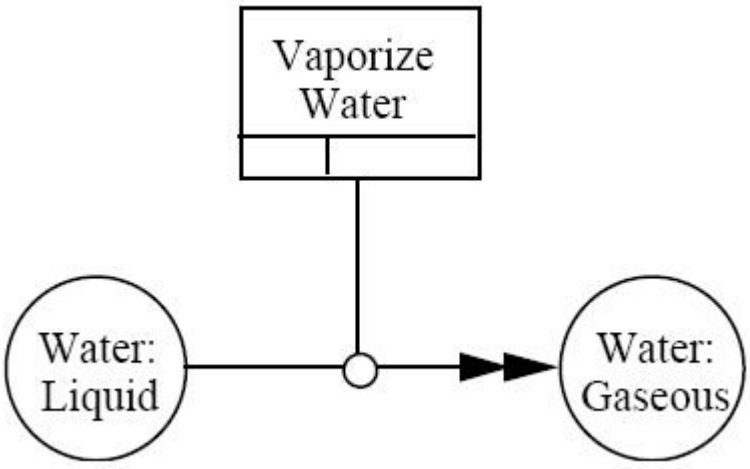
Classification Schematics



Composition Schematics



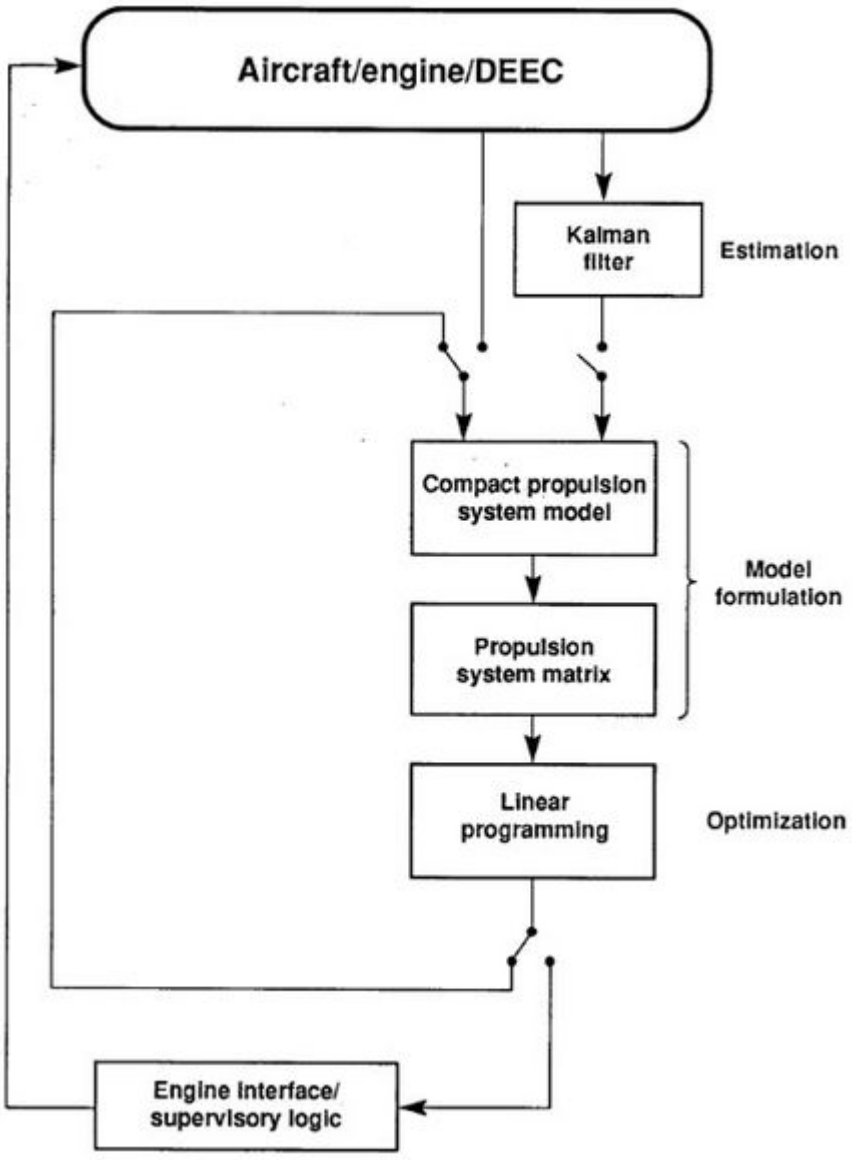
Relation Schematics



Object State Schematics

Chapter 5

Control Flow Diagram



Example of a so called "performance seeking control flow diagram".

A **control flow diagram** (**CFD**) is a diagram to describe the control flow of a business process, process or program.

Control flow diagrams were developed in the 1950s, and are widely used in multiple engineering disciplines. They are one of the classic business process modeling methodologies, along with flow charts, data flow diagrams, functional flow block diagram, Gantt charts, PERT diagrams, and IDEF.

Overview

A control flow diagram can consist of a subdivision to show sequential steps, with if-then-else conditions, repetition, and/or case conditions. Suitably annotated geometrical figures are used to represent operations, data, or equipment, and arrows are used to indicate the sequential flow from one to another.

There are several types of control flow diagrams, for example:

- Change control flow diagram, used in project management
- Configuration decision control flow diagram, used in configuration management
- Process control flow diagram, used in process management
- Quality control flow diagram, used in quality control.

In software and systems development control flow diagrams can be used in control flow analysis, data flow analysis, algorithm analysis, and simulation. Control and data flow analysis are most applicable for real time and data driven systems. These flow analyses transform logic and data requirements text into graphic flows which are easier to analyze than the text. PERT, state transition, and transaction diagrams are examples of control flow diagrams.

Types of Control Flow Diagrams

Process Control Flow Diagram

A flow diagram can be developed for the process control system for each critical activity. Process control is normally a closed cycle in which a sensor provides information to a process control software application through a communications system. The application determines if the sensor information is within the predetermined (or calculated) data parameters and constraints. The results of this comparison are fed to an actuator, which controls the critical component. This feedback may control the component electronically or may indicate the need for a manual action.

This closed-cycle process has many checks and balances to ensure that it stays safe. The investigation of how the process control can be subverted is likely to be extensive because all or part of the process control may be oral instructions to an individual monitoring the process. It may be fully computer controlled and automated, or it may be a hybrid in which only the sensor is automated and the action requires manual

intervention. Further, some process control systems may use prior generations of hardware and software, while others are state of the art.

Performance seeking control flow diagram

The figure presents an example of a performance seeking control flow diagram of the algorithm. The control law consists of estimation, modeling, and optimization processes. In the Kalman filter estimator, the inputs, outputs, and residuals were recorded. At the compact propulsion system modeling stage, all the estimated inlet and engine parameters were recorded.

In addition to temperatures, pressures, and control positions, such estimated parameters as stall margins, thrust, and drag components were recorded. In the optimization phase, the operating condition constraints, optimal solution, and linear programming health status condition codes were recorded. Finally, the actual commands that were sent to the engine through the DEEC were recorded. dfd(data float diagam)is network manen ment system

Chapter 6

Policy Analysis

Policy analysis is "determining which of various alternative policies will most achieve a given set of goals in light of the relations between the policies and the goals". However, policy analysis can be divided into two major fields. Analysis **of** policy is analytical and descriptive—i.e., it attempts to explain policies and their development. Analysis **for** policy is prescriptive—i.e., it is involved with formulating policies and proposals (e.g., to improve social welfare). The area of interest and the purpose of analysis determines what type of analysis is conducted. A combination of policy analysis together with program evaluation would be defined as Policy studies.

Policy Analysis is frequently deployed in the public sector, but is equally applicable to other kinds of organizations. Policy analysis has its roots in systems analysis as instituted by United States Secretary of Defense Robert McNamara during the Vietnam War.

Approaches

Although various approaches to policy analysis exist, three general approaches can be distinguished: the analycentric, the policy process, and the meta-policy approach.

The **analycentric** approach focuses on individual problems and its solutions; its scope is the micro-scale and its problem interpretation is usually of a technical nature. The primary aim is to identify the most effective and efficient solution in technical and economic terms (e.g. the most efficient allocation of resources).

The **policy process** approach puts its focal point onto political processes and involved stakeholders; its scope is the meso-scale and its problem interpretation is usually of a political nature. It aims at determining what processes and means are used and tries to explain the role and influence of stakeholders within the policy process. By changing the relative power and influence of certain groups (e.g., enhancing public participation and consultation), solutions to problems may be identified.

The **meta-policy approach** is a systems and context approach; i.e., its scope is the macro-scale and its problem interpretation is usually of a structural nature. It aims at explaining the contextual factors of the policy process; i.e., what are the political, economic and socio-cultural factors influencing it. As problems may result because of

structural factors (e.g., a certain economic system or political institution), solutions may entail changing the structure itself.

Methodology

Policy analysis is methodologically diverse using both qualitative methods and quantitative methods, including case studies, survey research, statistical analysis, and model building among others. One common methodology is to define the problem and evaluation criteria; identify all alternatives; evaluate them; and recommend the best policy agenda per favor.

Models

Many models exist to analyze the creation and application of public policy. Analysts use these models to identify important aspects of policy, as well as explain and predict policy and its consequences.

Some models are:

Institutional model

Public policy is determined by political institutions, which give policy legitimacy. Government universally applies policy to all citizens of society and monopolizes the use of force in applying policy. The legislature, executive and judicial branches of government are examples of institutions that give policy legitimacy.

Process model

Policy creation is a process following these steps:

- Identification of a problem and demand for government action.
- Formulation of policy proposals by various parties (e.g., congressional committees, think tanks, interest groups).
- Selection and enactment of policy; this is known as **Policy Legitimation**.
- Implementation of the chosen policy.
- Evaluation of policy.

This model, however, has been criticized for being overly linear and simplistic. In reality, stages of the policy process may overlap or never happen. Also, this model fails to take the multiple actors attempting the process itself as well as each other, and the complexity this entails.

Rational model

The rational model of decision-making is a process for making logically sound decisions in policy making in the public sector, although the model is also widely used in private corporations. Herbert Simon, the father of rational models, describes rationality as "*a style of behavior that is appropriate to the achievement of given goals, within the limits imposed by given conditions and constraints*". It is important to note the model makes a series of assumptions in order for it to work, such as:

- The model must be applied in a system that is stable,
- The government is a rational and unitary actor and that its actions are perceived as rational choices,
- The policy problem is unambiguous,
- There are no limitations of time or cost.

Indeed, some of the assumptions identified above are also pin pointed out in a study written by the historian H.A. Drake, as he states:

In its purest form, the Rational Actor approach presumes that such a figure [as Constantine] has complete freedom of action to achieve goals that he or she has articulated through a careful process of rational analysis involving full and objective study of all pertinent information and alternatives. At the same time, it presumes that this central actor is so fully in control of the apparatus of government that a decision once made is as good as implemented. There are no staffs on which to rely, no constituencies to placate, no generals or governors to cajole. By attributing all decision making to one central figure who is always fully in control and who acts only after carefully weighing all options, the Rational Actor method allows scholars to filter out extraneous details and focus attention on central issues.

Furthermore, as we have seen, in the context of policy rational models are intended to achieve maximum social gain. For this purpose, Simon identifies an outline of a step by step mode of analysis to achieve rational decisions. Ian Thomas describes Simon's steps as follows:

1. Intelligence gathering— data and potential problems and opportunities are identified, collected and analyzed.
2. Identifying problems
3. Assessing the consequences of all options
4. Relating consequences to values— with all decisions and policies there will be a set of values which will be more relevant (for example, economic feasibility and

environmental protection) and which can be expressed as a set of criteria, against which performance (or consequences) of each option can be judged.
5. Choosing the preferred option— given the full understanding of all the problems and opportunities, all the consequences and the criteria for judging options.

In similar lines, Wiktorowicz and Deber describe through their study on 'Regulating biotechnology: a rational-political model of policy development' the rational approach to policy development. The main steps involved in making a rational decision for these authors are the following:

1. The comprehensive organization and analysis of the information
2. The potential consequences of each option
3. The probability that each potential outcome would materialize
4. The value (or utility) placed on each potential outcome.

The approach of Wiktorowicz and Deber is similar to Simon and they assert that the rational model tends to deal with "the facts" (data, probabilities) in steps 1 to 3, leaving the issue of assessing values to the final step. According Wiktorowicz and Deber values are introduced in the final step of the rational model, where the utility of each policy option is assessed.

Many authors have attempted to interpret the above mentioned steps, amongst others, Patton and Sawicki who summarize the model as presented in the following figure:

File:Interpretation of the rational model of decision making.PNG

1. Defining the problem by analyzing the data and the information gathered.
2. Identifying the decision criteria that will be important in solving the problem. The decision maker must determine the relevant factors to take into account when making the decision.
3. A brief list of the possible alternatives must be generated; these could succeed to resolve the problem.
4. A critical analyses and evaluation of each criterion is brought through. For example strength and weakness tables of each alternative are drawn and used for comparative basis. The decision maker then weights the previously identified criteria in order to give the alternative policies a correct priority in the decision.
5. The decision-maker evaluates each alternative against the criteria and selects the preferred alternative.
6. The policy is brought through.

The model of rational decision-making has also proven to be very useful to several decision making processes in industries outside the public sphere. Nonetheless, many criticism of the model arise due to claim of the model being impractical and lying on unrealistic assumptions. . For instance, it is a difficult model to apply in the public sector because social problems can be very complex, ill-defined and interdependent. The problem lies in the thinking procedure implied by the model which is linear and can face

difficulties in extra ordinary problems or social problems which have no sequences of happenings. This latter argument can be best illustrated by the words of Thomas R. Dye, the president of the Lincoln Center for Public Service, who wrote in his book `Understanding Public Policy´ the following passage:

There is no better illustration of the dilemmas of rational policy making in America than in the field of health…the first obstacle to rationalism is defining the problem. Is our goal to have good health — that is, whether we live at all (infant mortality), how well we live (days lost to sickness), and how long we live (life spans and adult mortality)? Or is our goal to have good medical care — frequent visits to the doctor, wellequipped and accessible hospitals, and equal access to medical care by rich and poor alike?

The problems faced when using the rational model arise in practice because social and environmental values can be difficult to quantify and forge consensus around. Furthermore, the assumptions stated by Simon are never fully valid in a real world context.

However, as Thomas states the rational model provides a good perspective since in modern society rationality plays a central role and everything that is rational tends to be prized. Thus, it does not seem strange that "we ought to be trying for rational decision-making".

Decision Criteria for Policy Analysis — Step 2

As illustrated in Figure 1, rational policy analysis can be broken into 6 distinct stages of analysis. Step 2 highlights the need to understand which factors should be considered as part of the decision making process. At this part of the process, all the economic, social, and environmental factors that are important to the policy decision need to be identified and then expressed as policy decision criteria. For example, the decision criteria used in the analysis of environmental policy is often a mix of —

- Ecological impacts — such as biodiversity, water quality, air quality, habitat quality, species population, etc.
- Economic efficiency — commonly expressed as benefits and costs.
- Distributional equity — how policy impacts are distributed amongst different demographics. Factors that can affect the distribution of impacts include location, ethinicity, income, and occupation.
- Social/Cultural acceptability — the extent to which the policy action may be opposed by current social norms or cultural values.
- Operational practicality — the capacity required to actually operationalize the policy. For example,

- Legality — the potential for the policy to be implemented under current legislation versus the need to pass new legislation that accommodates the policy.

- Uncertainty — the degree to which the level of policy impacts can be known.

Some criteria, such as economic benefit, will be more easily measurable or definable, while others such as environmental quality will be harder to measure or express quantitatively. Ultimately though, the set of decision criteria needs to embody all of the policy goals, and overemphasising the more easily definable or measurable criteria, will have the undesirable impact of biasing the analysis towards a subset of the policy goals.

The process of identifying a suitably comprehensive decision criteria set is also vulnerable to being skewed by pressures arising at the political interface. For example, decision makers may tend to give "*more weight to policy impacts that are concentrated, tangible, certain, and immediate than to impacts that are diffuse, intangible, uncertain, and delayed*."^8. For example, with a cap-and-trade system for carbon emissions the net financial cost in the first five years of policy implementation is a far easier impact to conceptualise than the more diffuse and uncertain impact of a country's improved position to influence global negotiations on climate change action.

Decision Methods for Policy Analysis — Step 5

Displaying the impacts of policy alternatives can be done using a policy analysis matrix (PAM) such that shown in Table 1. As shown, a PAM provides a summary of the policy impacts for the various alternatives and examination of the matrix can reveal the tradeoffs associated with the different alternatives.

Table 1. Policy analysis matrix (PAM) for SO2 emissions control.

Once policy alternatives have been evaluated, the next step is to decide which policy alternative should be implemented. This is shown as step 5 in Figure 1. At one extreme, comparing the policy alternatives can be relatively simple if all the policy goals can be measured using a single metric and given equal weighting. In this case, the decision method is an exercise in benefit cost analysis (BCA).

At the other extreme, the numerous goals will require the policy impacts to be expressed using a variety of metrics that are not readily comparable. In such cases, the policy analyst may draw on the concept of utility to aggregate the various goals into a single score. With the utility concept, each impact is given a weighting such that 1 unit of each weighted impact is considered to be equally valuable (or desirable) with regards to the collective well-being.

Weimer and Vining also suggest that the "*go, no go*" rule can be a useful method for deciding amongst policy alternatives^8. Under this decision making regime, some or all policy impacts can be assigned thresholds which are used to eliminate at least some of the policy alternatives. In their example, one criterion "*is to minimize SO2 emissions*" and so

a threshold might be a reduction SO2 emissions "of at least 8.0 million tons per year". As such, any policy alternative that does not meet this threshold can be removed from consideration. If only a single policy alternative satisfies all the impact thresholds then it is the one that is considered a "go" for each impact. Otherwise it might be that all but a few policy alternatives are eliminated and those that remain need to be more closely examined in terms of their trade-offs so that a decision can be made.

Case Study Example of Rational Policy Analysis Approach

To demonstrate the rational analysis process as described above, let's examine the policy paper "Stimulating the use of biofuels in the European Union: Implications for climate change policy" by Lisa Ryan where the substitution of fossil fuels with biofuels has been proposed in the European Union (EU) between 2005–2010 as part of a strategy to mitigate greenhouse gas emissions from road transport, increase security of energy supply and support development of rural communities.

Considering the steps of Patton and Sawicki model as in Figure 1 above, this paper only follows components 1 to 5 of the rationalist policy analysis model:

1. Defining The Problem – the report identifies transportation fuels pose two important challenges for the European Union (EU). First, under the provisions of the Kyoto Protocol to the Climate Change Convention, the EU has agreed to an absolute cap on greenhouse gas emissions; while, at the same time increased consumption of transportation fuels has resulted in a trend of increasing greenhouse gas emissions from this source. Second, the dependence upon oil imports from the politically volatile Middle East generates concern over price fluctuations and possible interruptions in supply. Alternative fuel sources need to be used & substituted in place of fossil fuels to mitigate GHG emissions in the EU.
2. Determine the Evaluation Criteria – this policy sets Environmental impacts/benefits (reduction of GHG's as a measure to reducing climate change effects) and Economical efficiency (the costs of converting to biofuels as alternative to fossil fuels & the costs of production of biofuels from its different potential sources)as its decision criteria. However, this paper does not exactly talk about the social impacts, this policy may have. It also does not compare the operational challenges involved between the different categories of biofuels considered.
3. Identifying Alternative Policies – The European Commission foresees that three alternative transport fuels: hydrogen, natural gas, and biofuels, will replace transport fossil fuels, each by 5% by 2020.
4. Evaluating Alternative Policies – Biofuels are an alternative motor vehicle fuel produced from biological material and are promoted as a transitional step until more advanced technologies have matured. By modelling the efficiency of the biofuel options the authors compute the economic and environmental costs of each biofuel option as per the evaluation criteria mentioned above.

5. Select The Preferred Policy – The authors suggest that the overall best biofuel comes from the sugarcane in Brazil after comparing the economic & the environmental costs. The current cost of subsidising the price difference between European biofuels and fossil fuels per tonne of CO_2 emissions saved is calculated to be €229–2000. If the production of European biofuels for transport is to be encouraged, exemption from excise duties is the instrument that incurs the least transactions costs, as no separate administrative or collection system needs to be established. A number of entrepreneurs are producing biofuels at the lower margin of the costs specified here profitably, once an excise duty rebate is given. It is likely that growth in the volume of the business will engender both economies of scale and innovation that will reduce costs substantially.

Group model

The political system's role is to establish and enforce compromise between various, conflicting interests in society.

Elite model

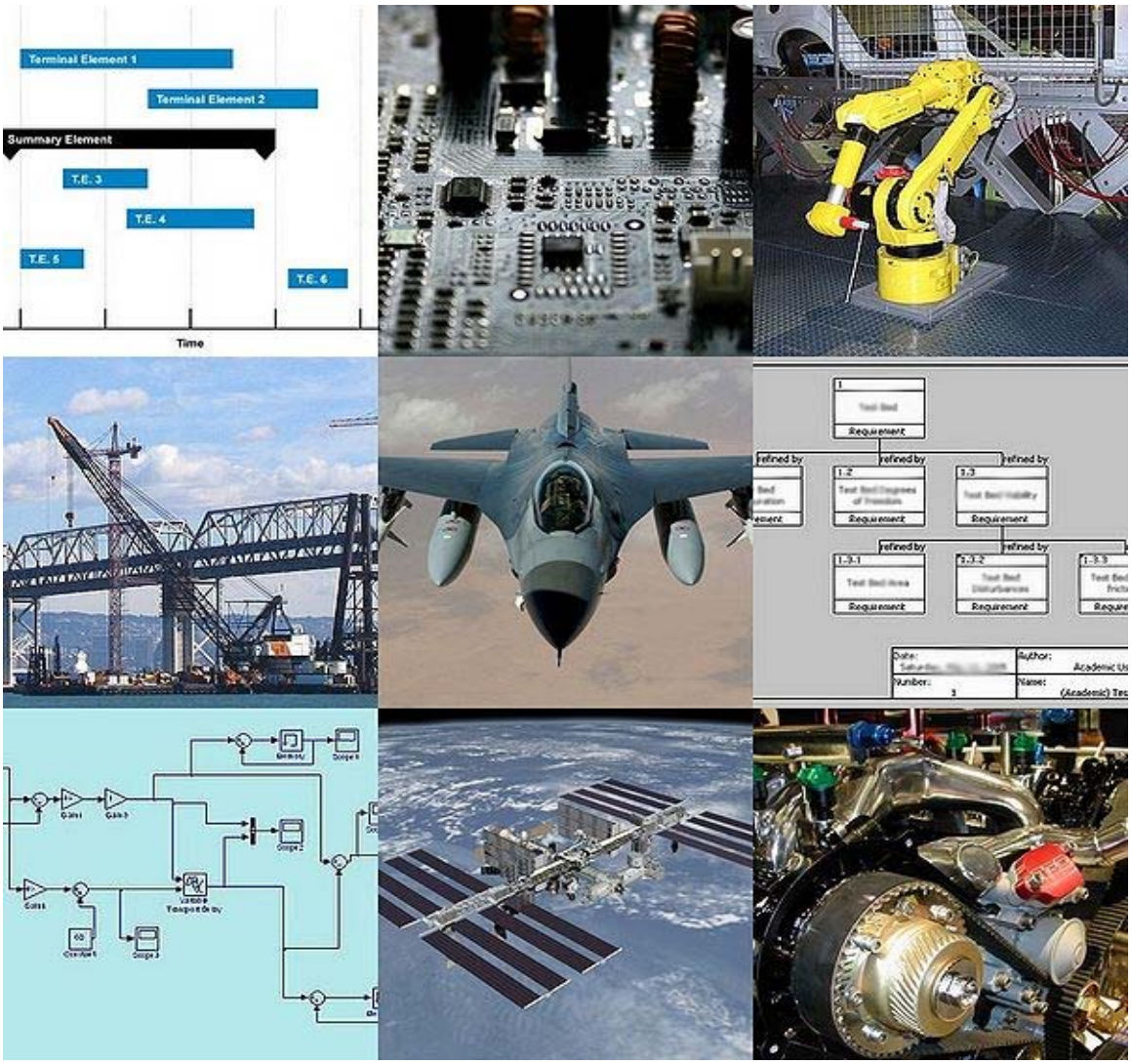
Policy is a reflection of the interests of those individuals within a society that have the most power, rather than the demands of the mass.

Six-step model

1. Verify, define and detail the problem
2. Establish evaluation criteria
3. Identify alternative policies
4. Evaluate alternative policies
5. Display and distinguish among alternative policies
6. Monitor the implemented policy

Chapter 7

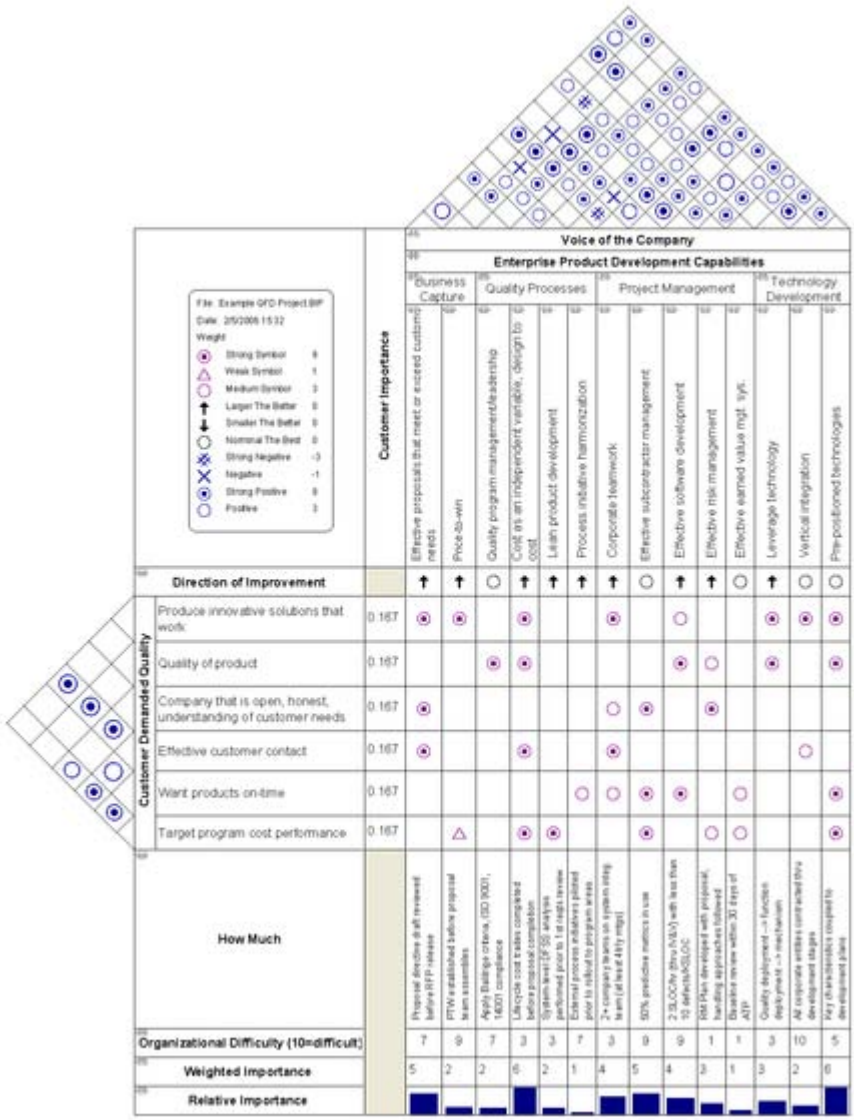
Systems Engineering



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

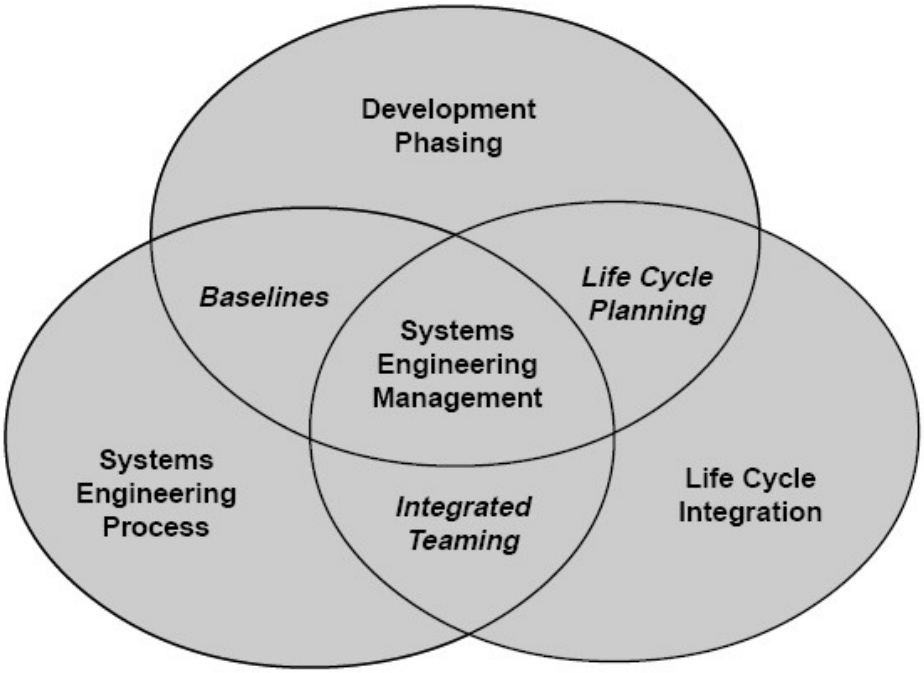
The need for systems engineering arose with the increase in complexity of systems and projects. When speaking in this context, complexity incorporates not only engineering systems, but also the logical human organization of data. At the same time, a system can become more complex due to an increase in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

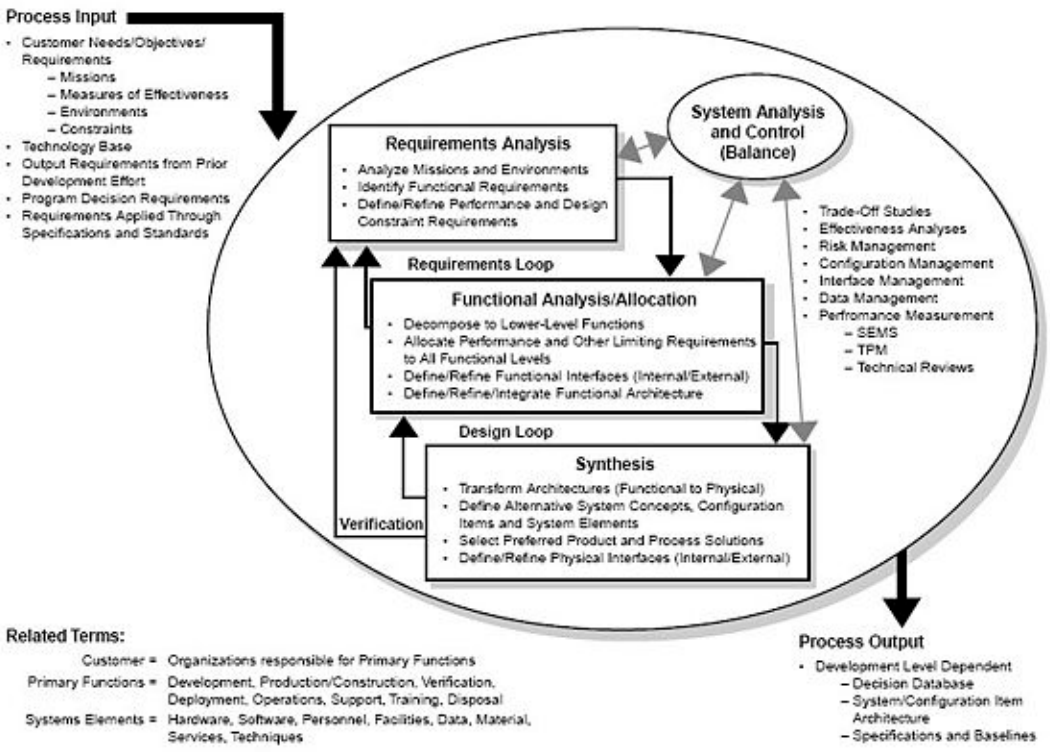
System

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 8

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 9

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 10

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.

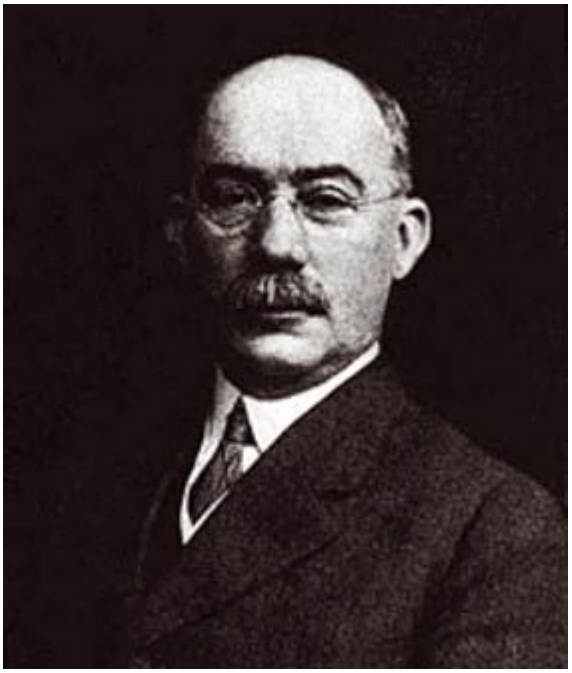
A project is a temporary endeavor, having a defined beginning and end (usually constrained by date, but can be by funding or deliverables), undertaken to meet unique goals and objectives, usually to bring about beneficial change or added value. The temporary nature of projects stands in contrast to business 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

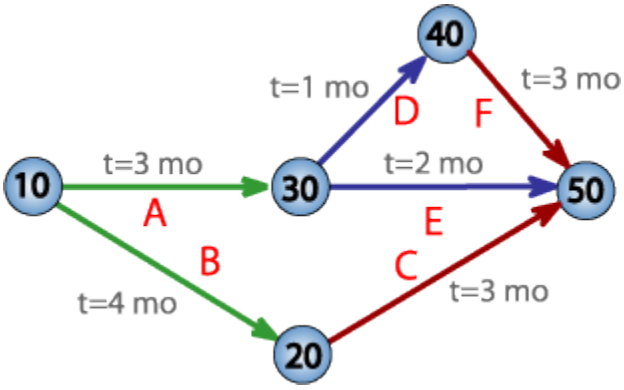
Project management has been practiced since early civilization. Until 1900 civil engineering projects were generally managed by creative architects and engineers themselves, among those for example Vitruvius (1st century BC), Christopher Wren (1632–1723), Thomas Telford (1757–1834) and Isambard Kingdom 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.

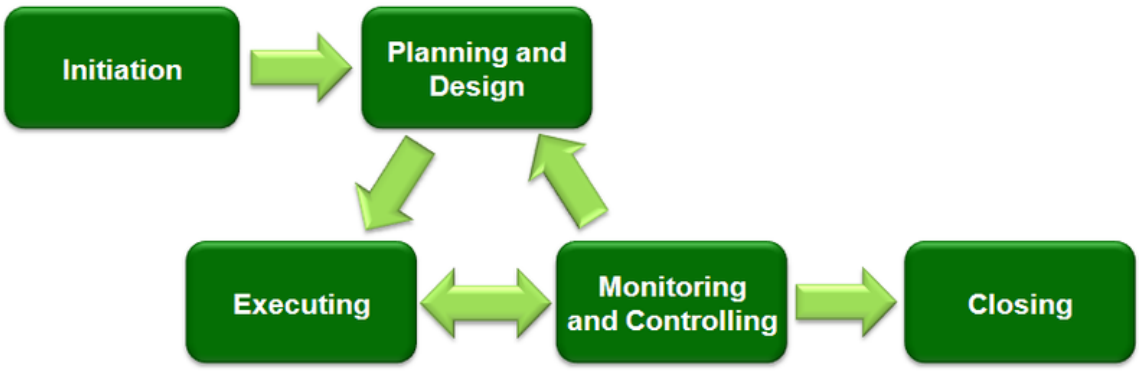
Approaches

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.

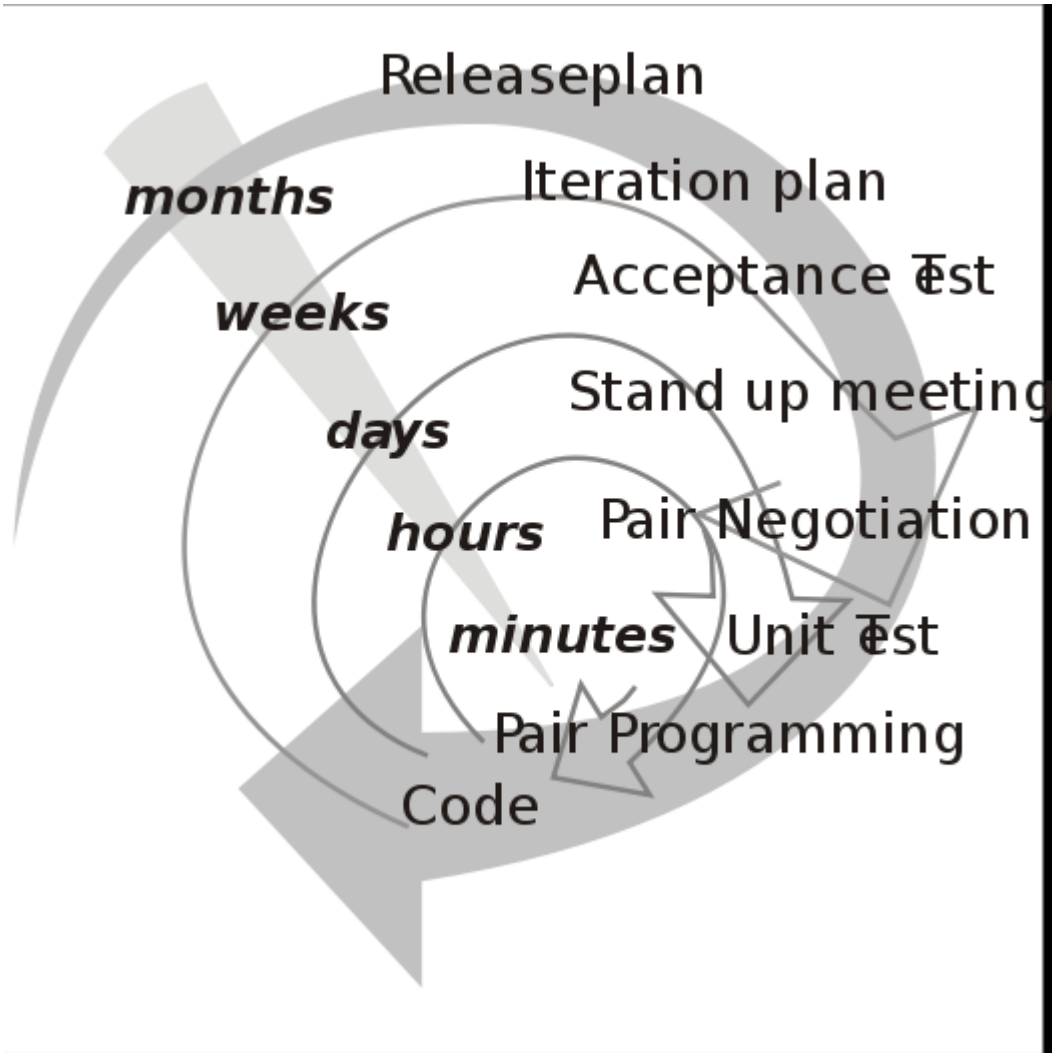
Many industries use variations on these project stages. For example, when working on a brick and mortar design and construction, projects will typically progress through stages like Pre-Planning, Conceptual Design, Schematic Design, Design Development, Construction Drawings (or Contract Documents), and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

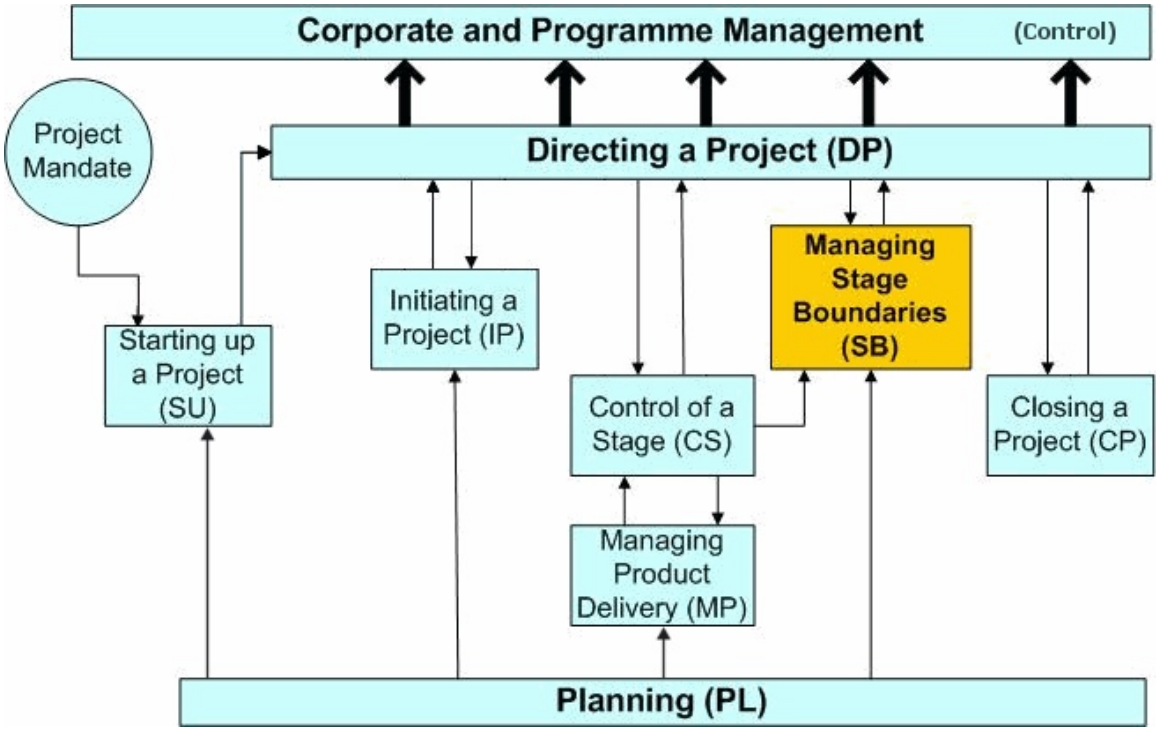
The generalization of Extreme Programming to other kinds of projects is extreme project management, which may be used in combination with the process modeling and management principles of human interaction management.

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

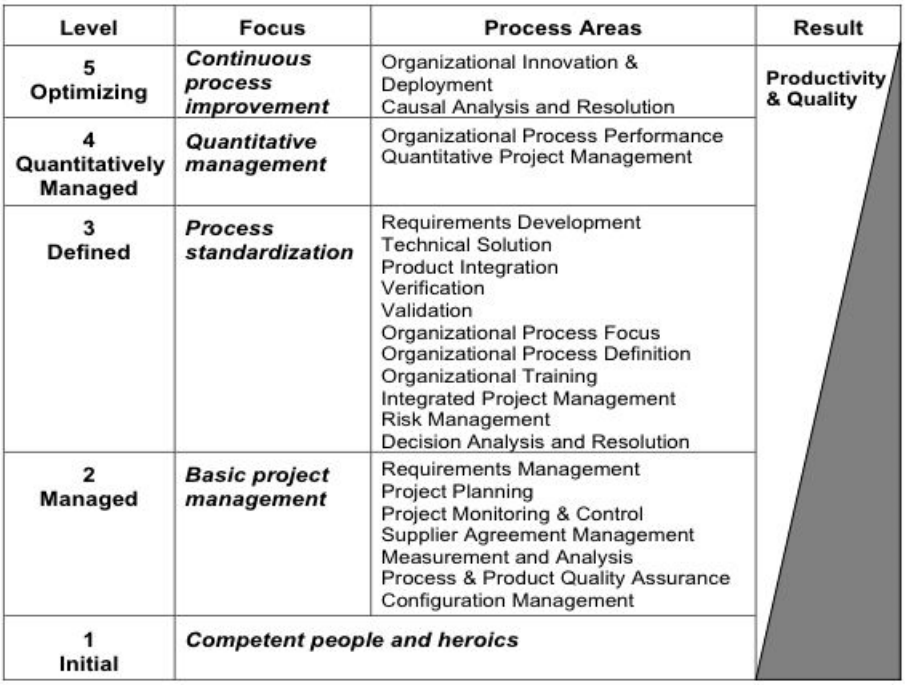
PRINCE2

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

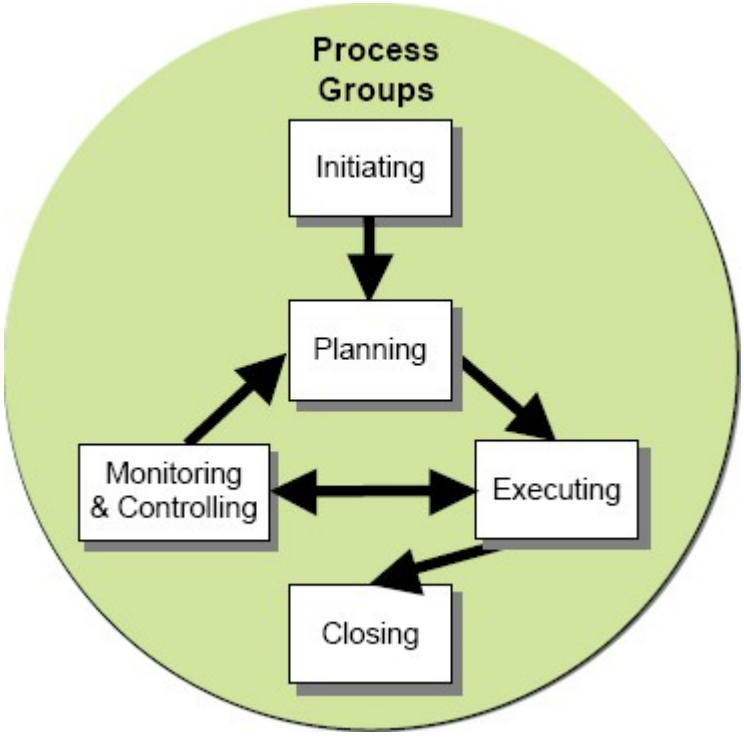
Also furthering the concept of project control is the incorporation of process-based management. This area has been driven by the use of Maturity models such as the CMMI (Capability Maturity Model Integration) and ISO/IEC15504 (SPICE - Software Process Improvement and Capability Estimation).

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.

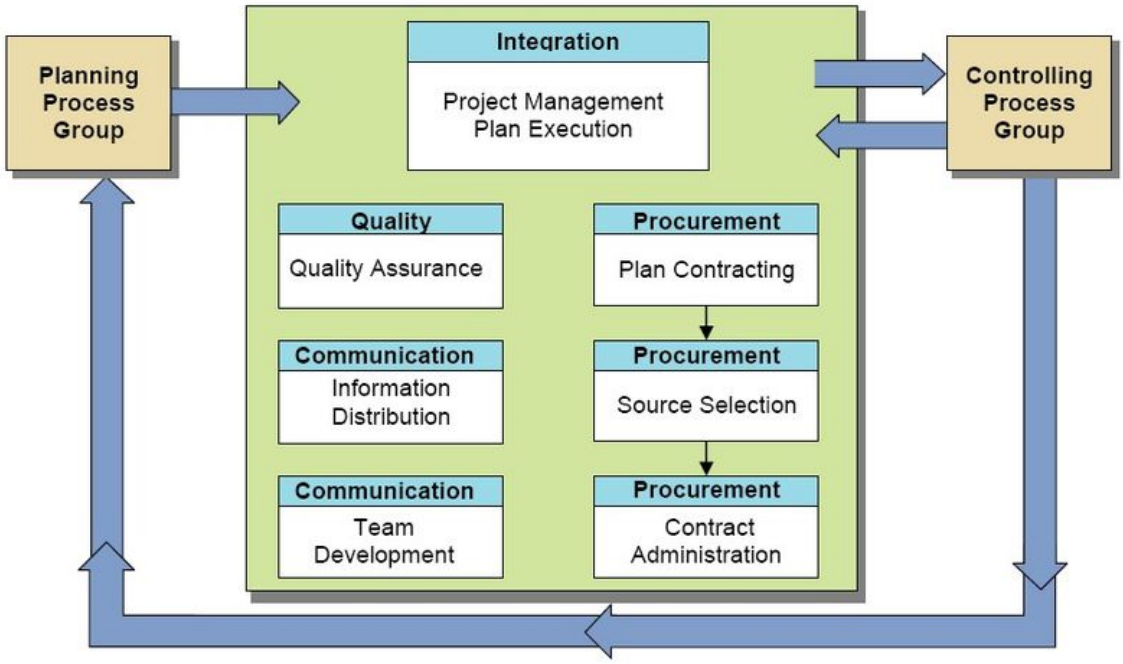
Project planning generally consists of

- 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

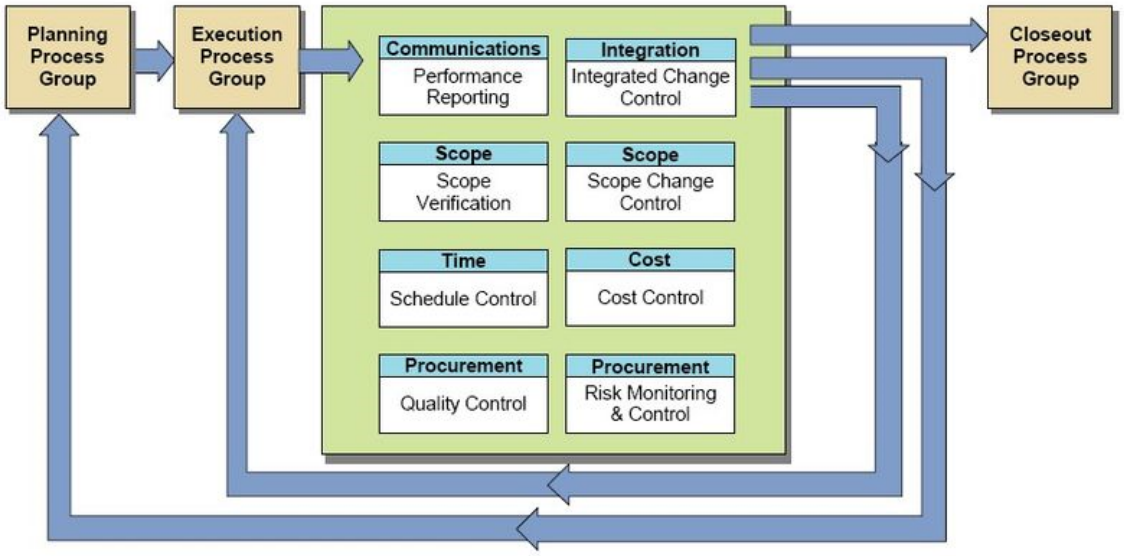


Executing Process Group Processes

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

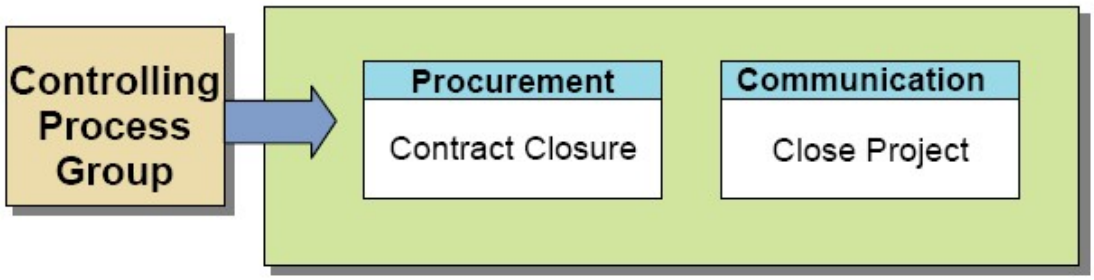
Monitoring and Controlling cycle

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

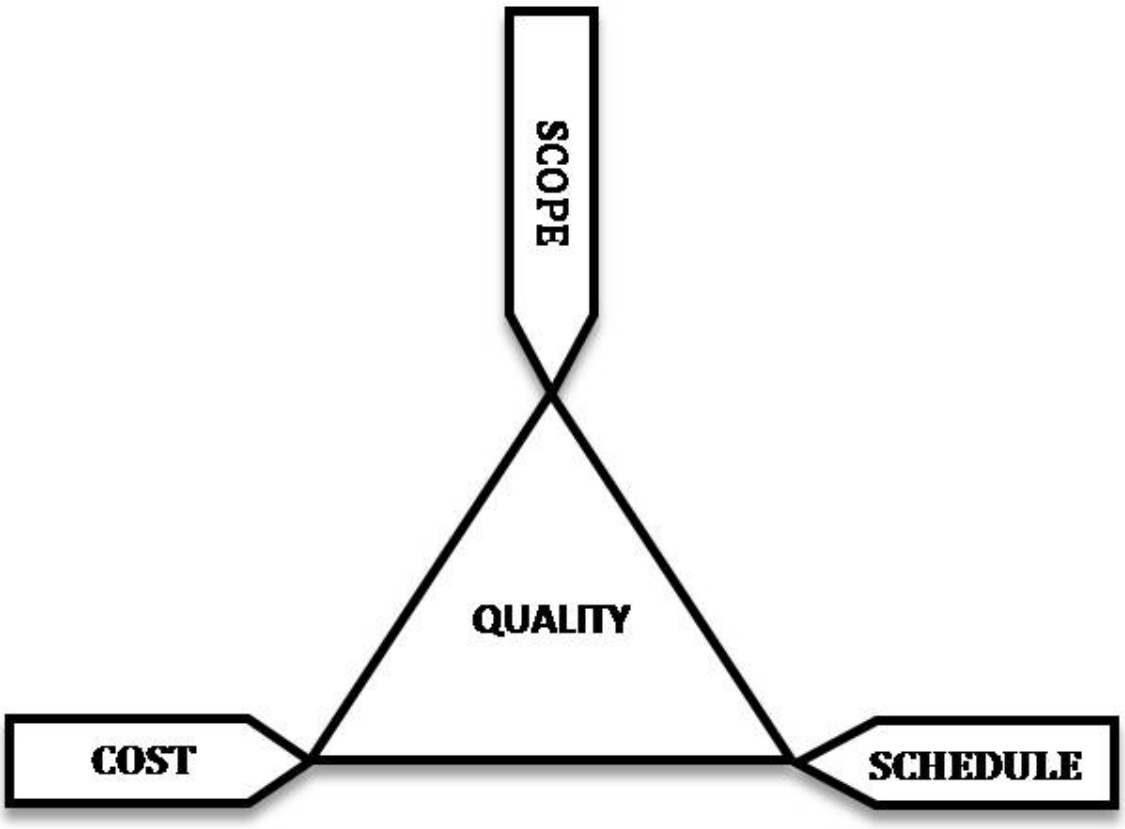
Project managers

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



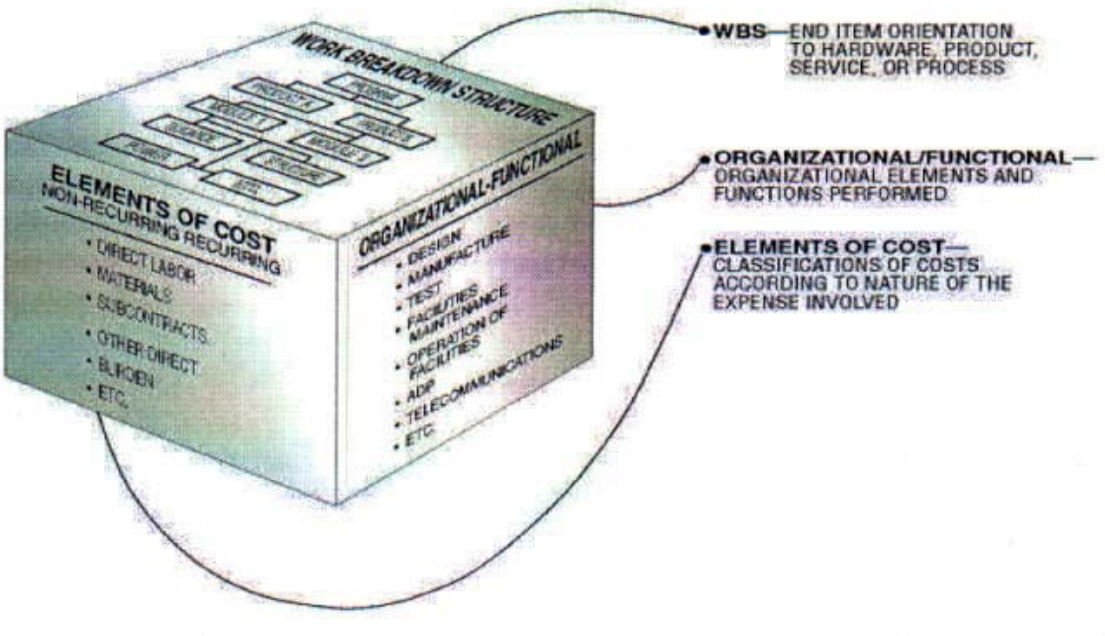
The Project Management Triangle

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



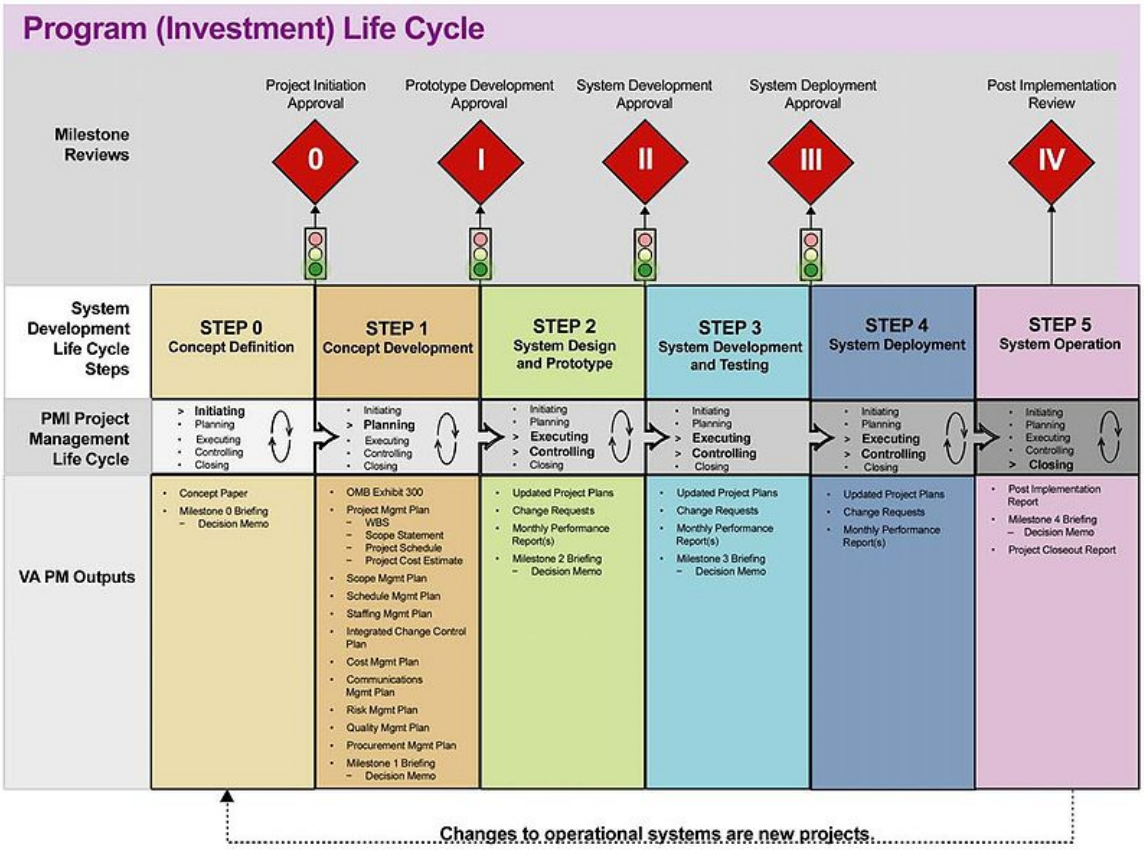
Example of a Work breakdown structure applied in a NASA reporting structure

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.

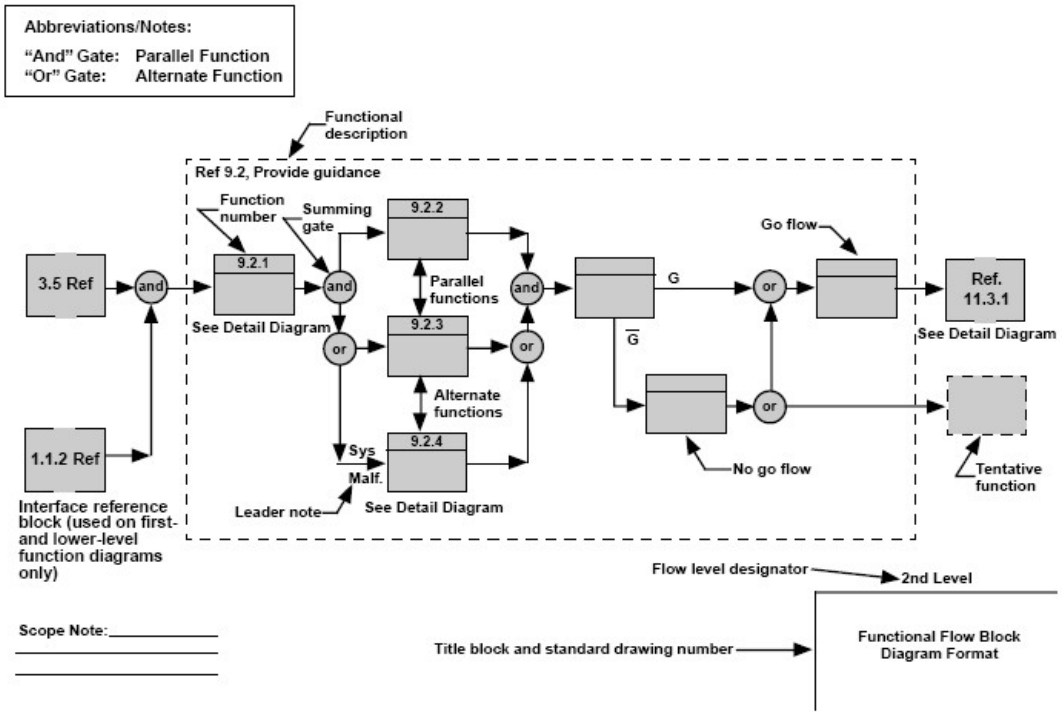
Project portfolio management

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 11

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

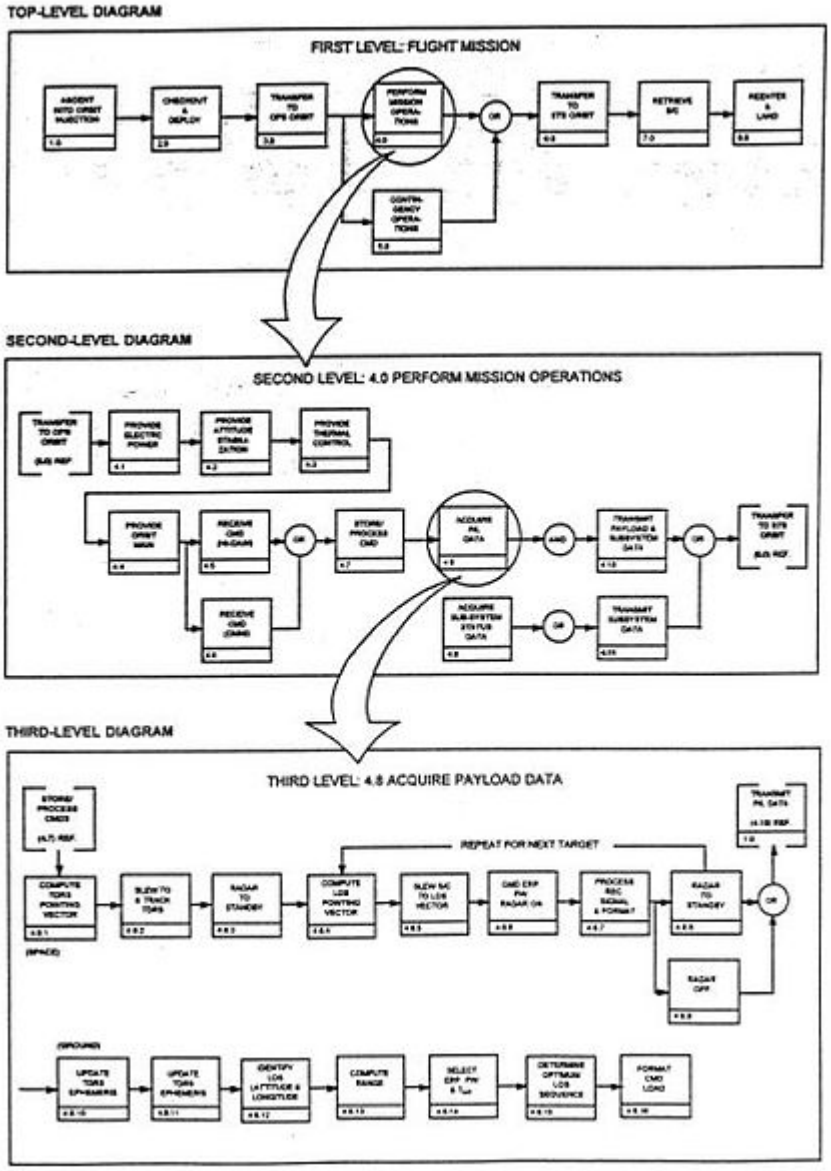


Figure 2: Development of Functional Flow Block Diagrams

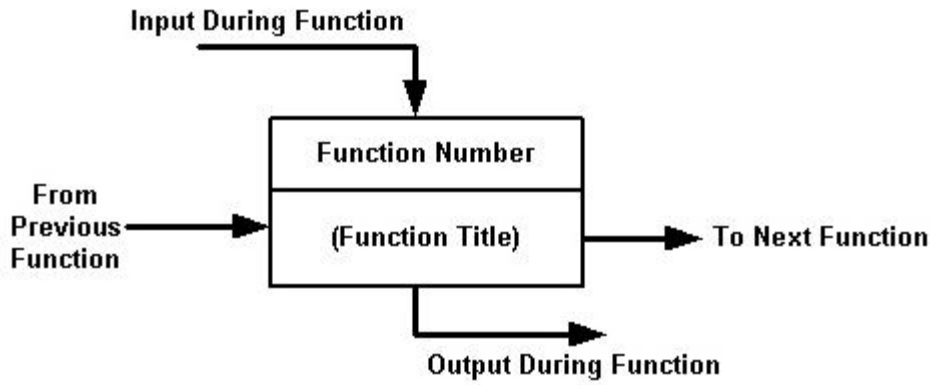
FFBDs can be developed in a series of levels. FFBDs show the same tasks identified through functional decomposition and display them in their logical, sequential relationship. For example, the entire flight mission of a spacecraft can be defined in a top level FFBD, as shown in Figure 2. Each block in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

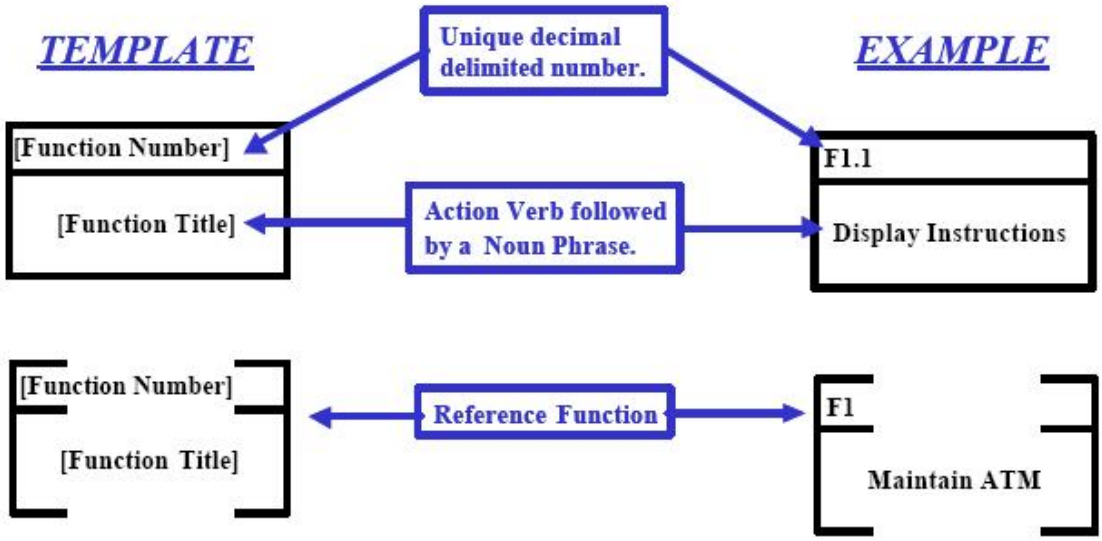


Figure 3. Function Symbol

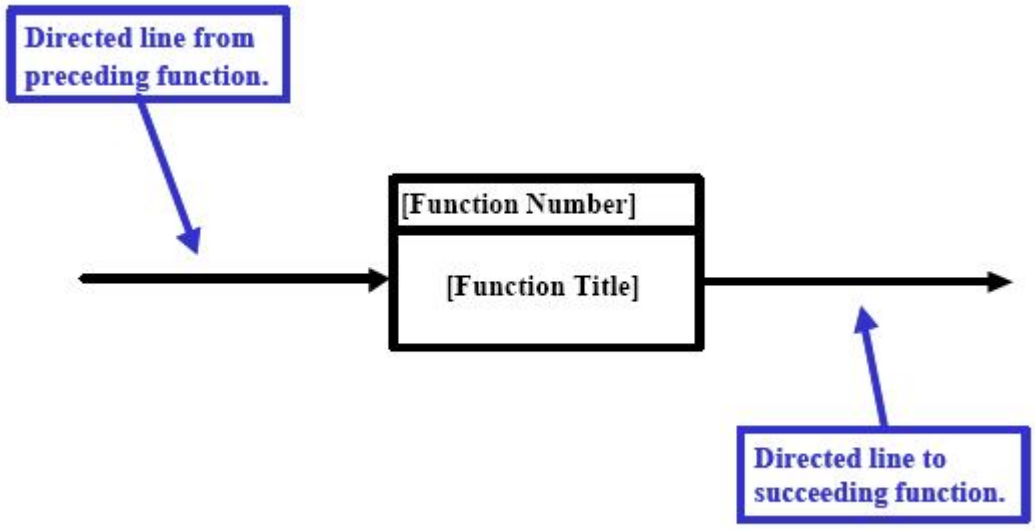


Figure 4. Directed Lines

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.

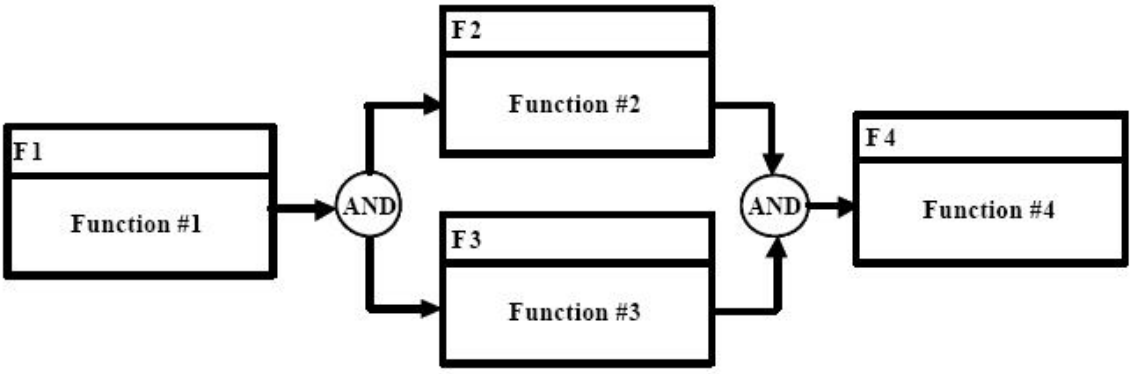


Figure 5. "AND" Symbol

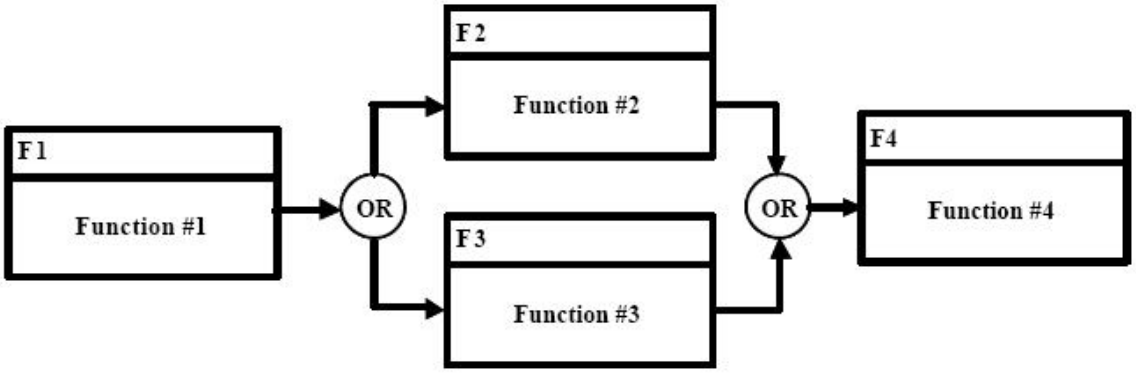


Figure 6. "Exclusive OR" Symbol

- 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

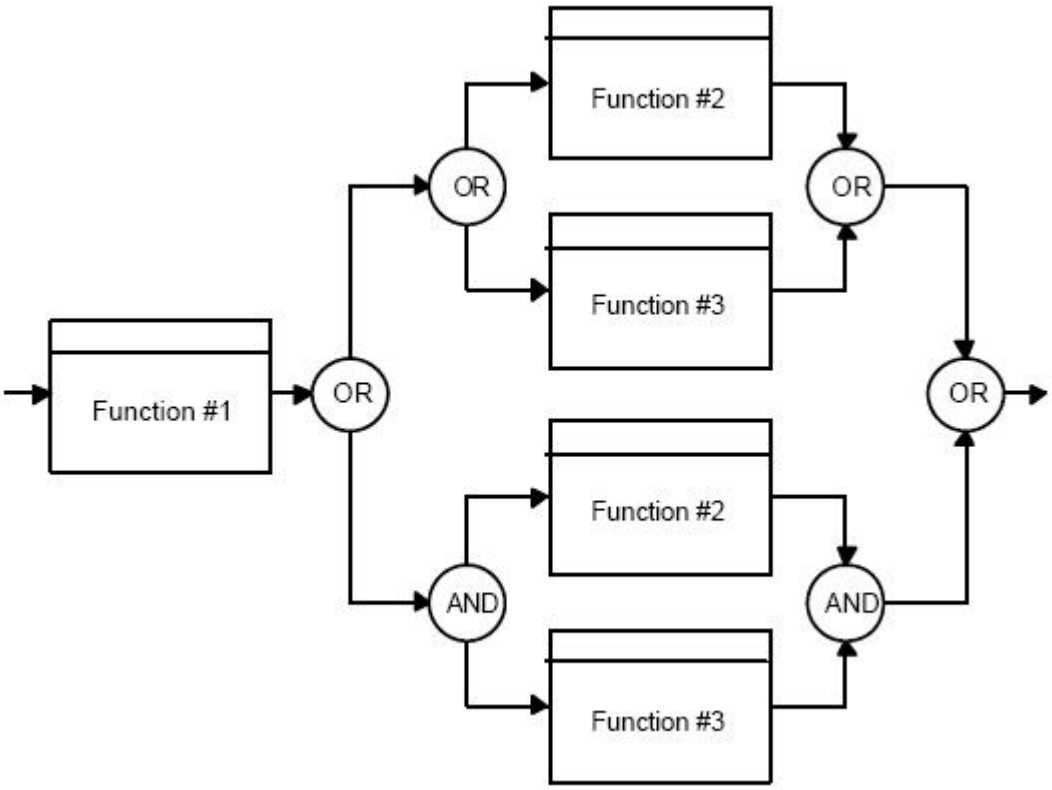


Figure 7. “Inclusive OR” Logic

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.

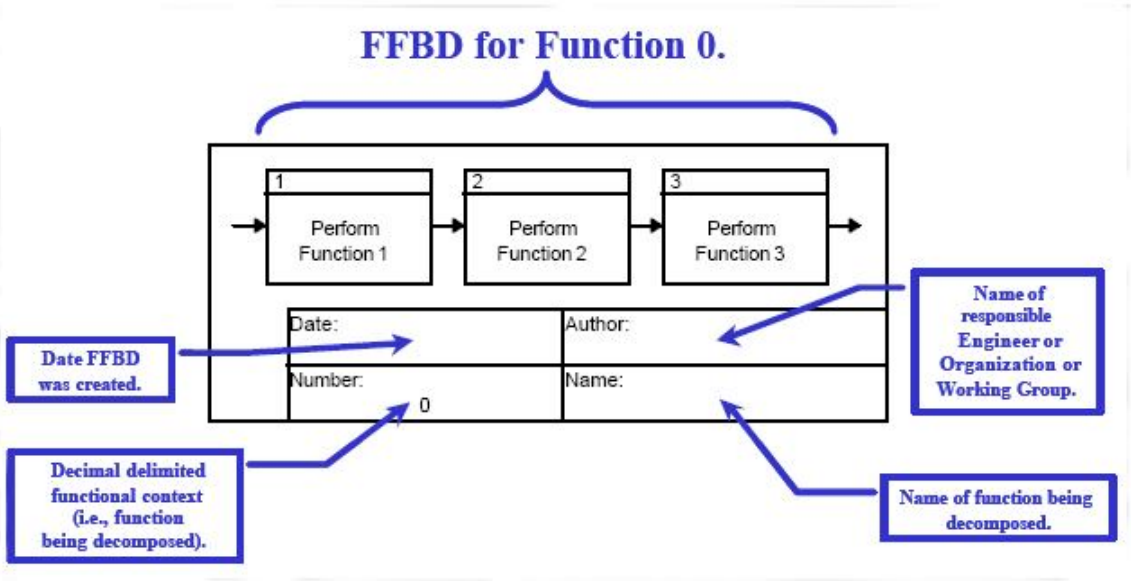


Figure 8. FFBD Function 0 Illustration

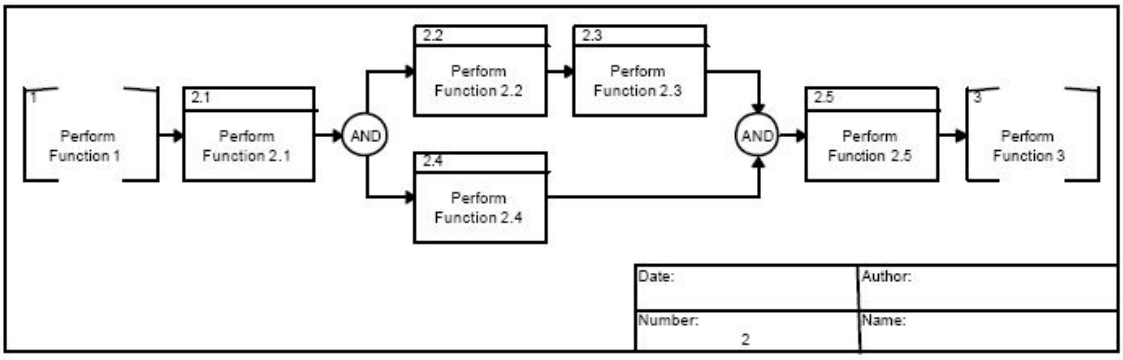
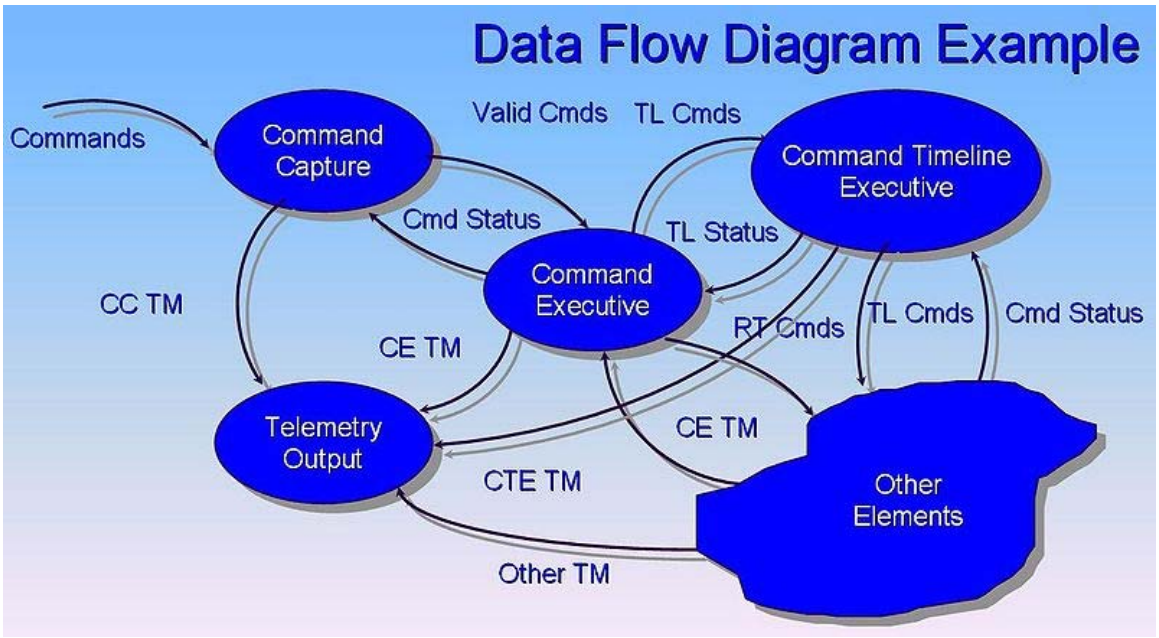


Figure 9. FFBD Function 2 Illustration

Chapter 12

Data Flow Diagram and N2 Chart

Data flow diagram



Data flow diagram example

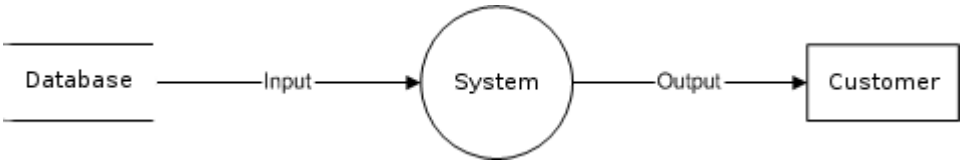
A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

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.

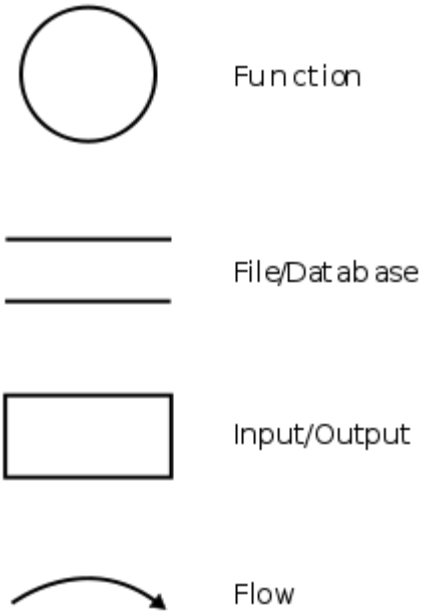
A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in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

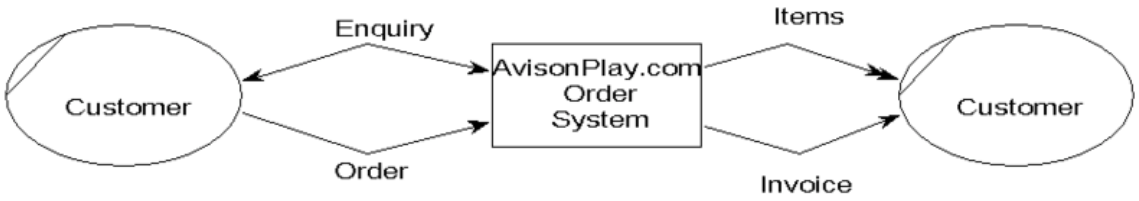
In the course of developing a set of *levelled* data flow diagrams the analyst/designers is forced to address how the system may be decomposed into component sub-systems, and to identify the transaction data in the data model.

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*.



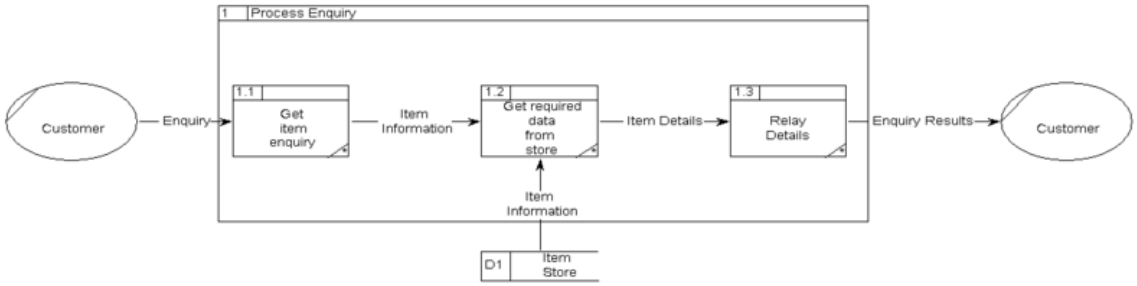
A context level Data flow diagram created using Select SSADM

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.

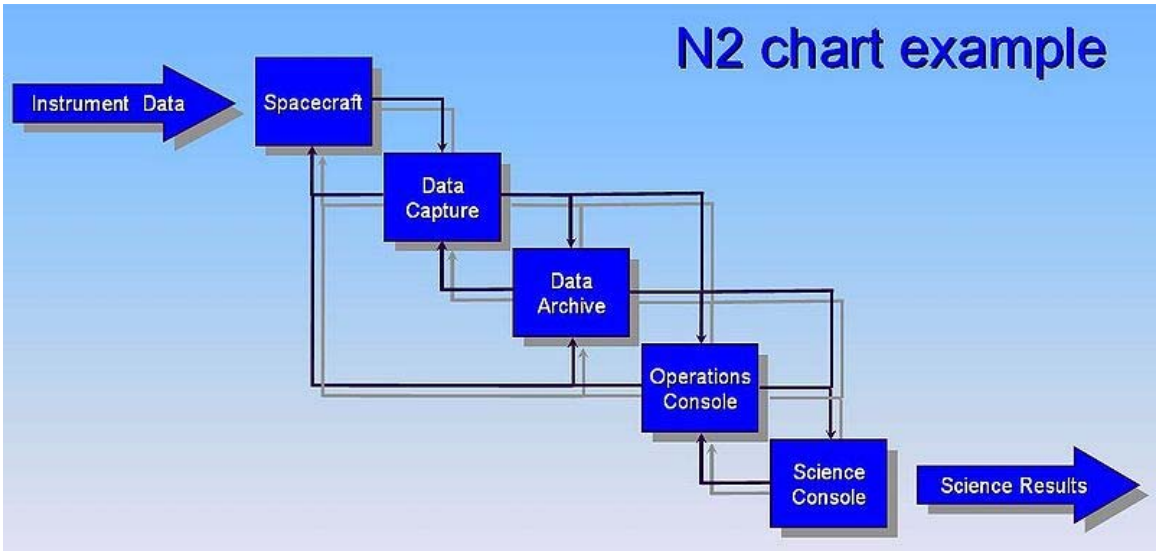
Level 2 (low level diagram)



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.

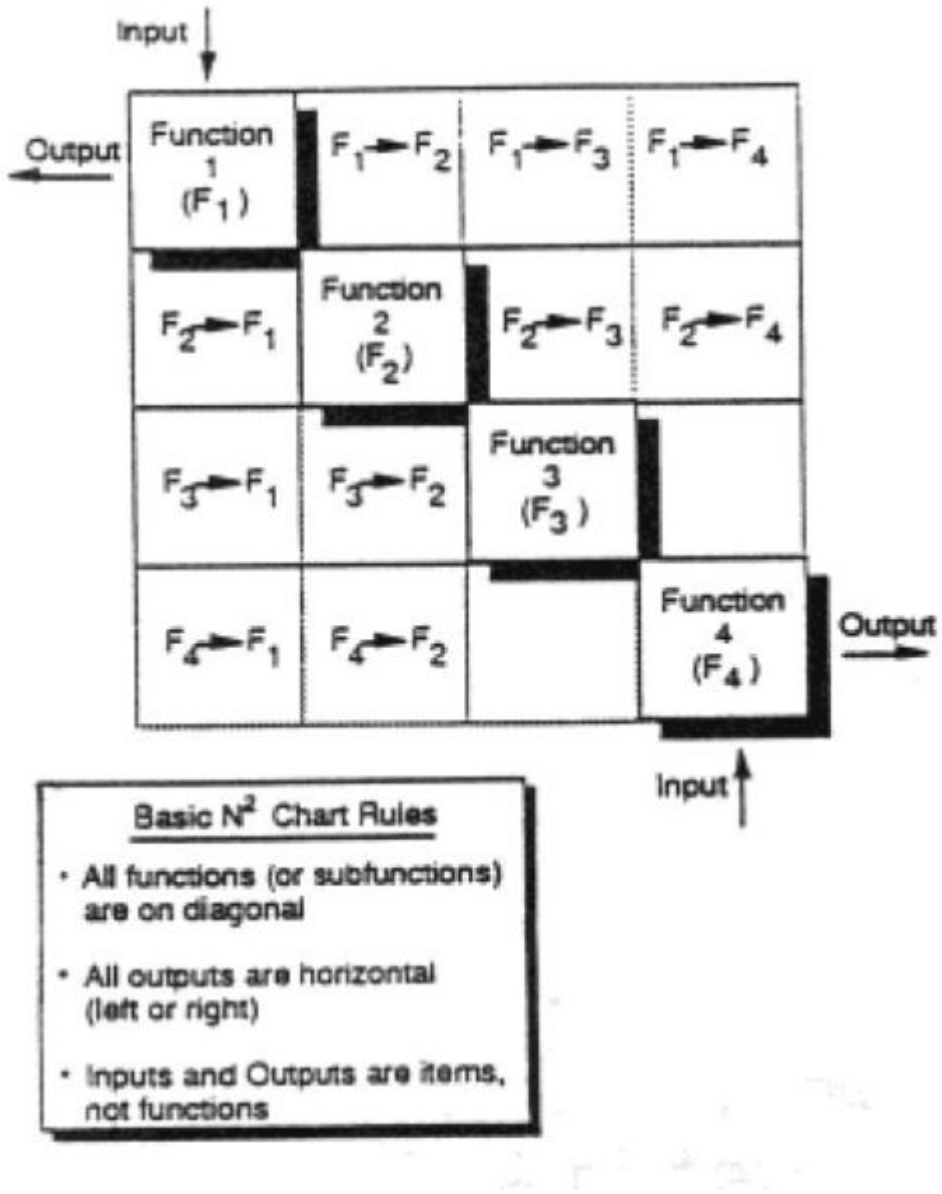


Figure 2. N2 chart definition

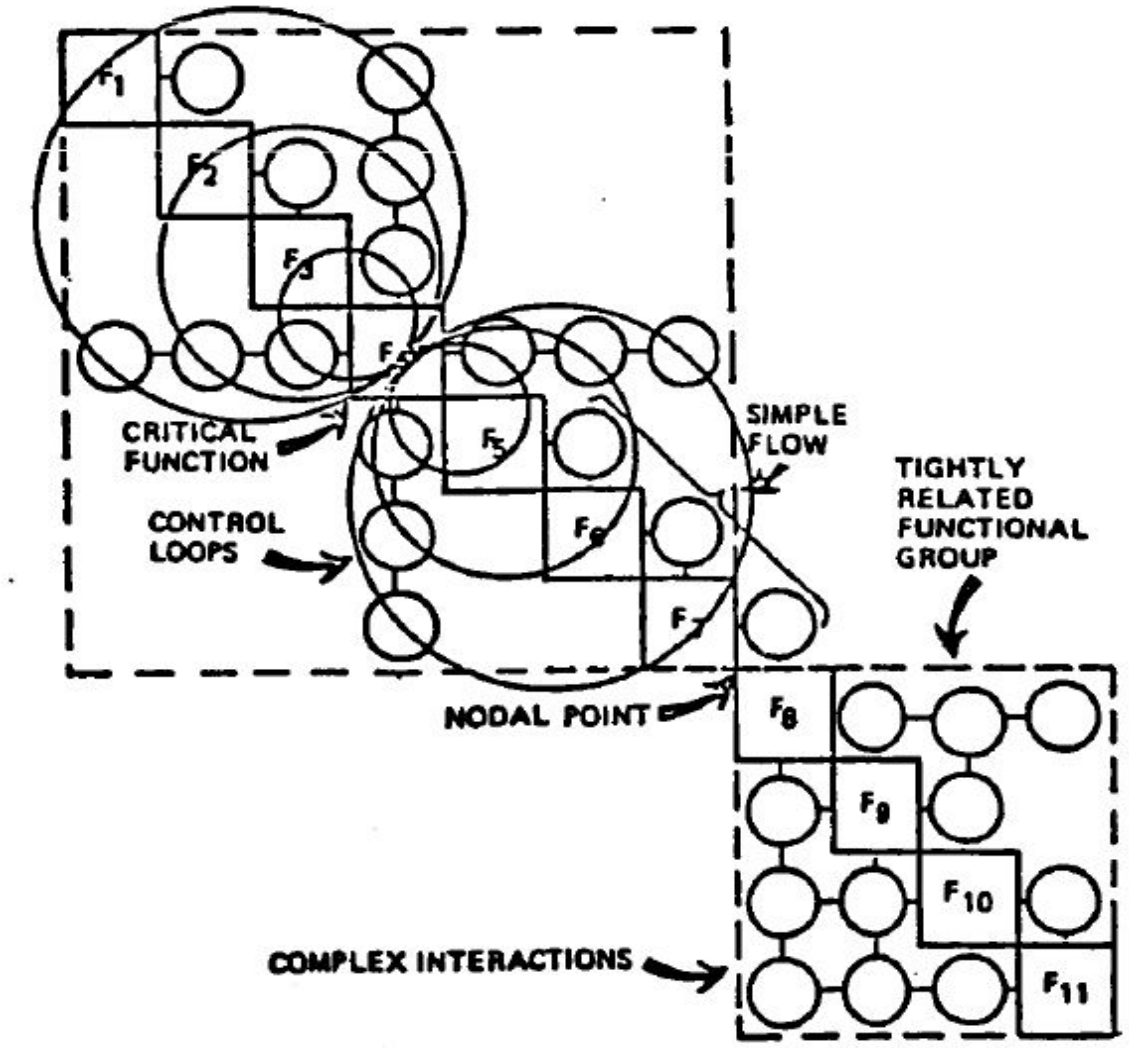


Figure 3. N2 Chart Key Features

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

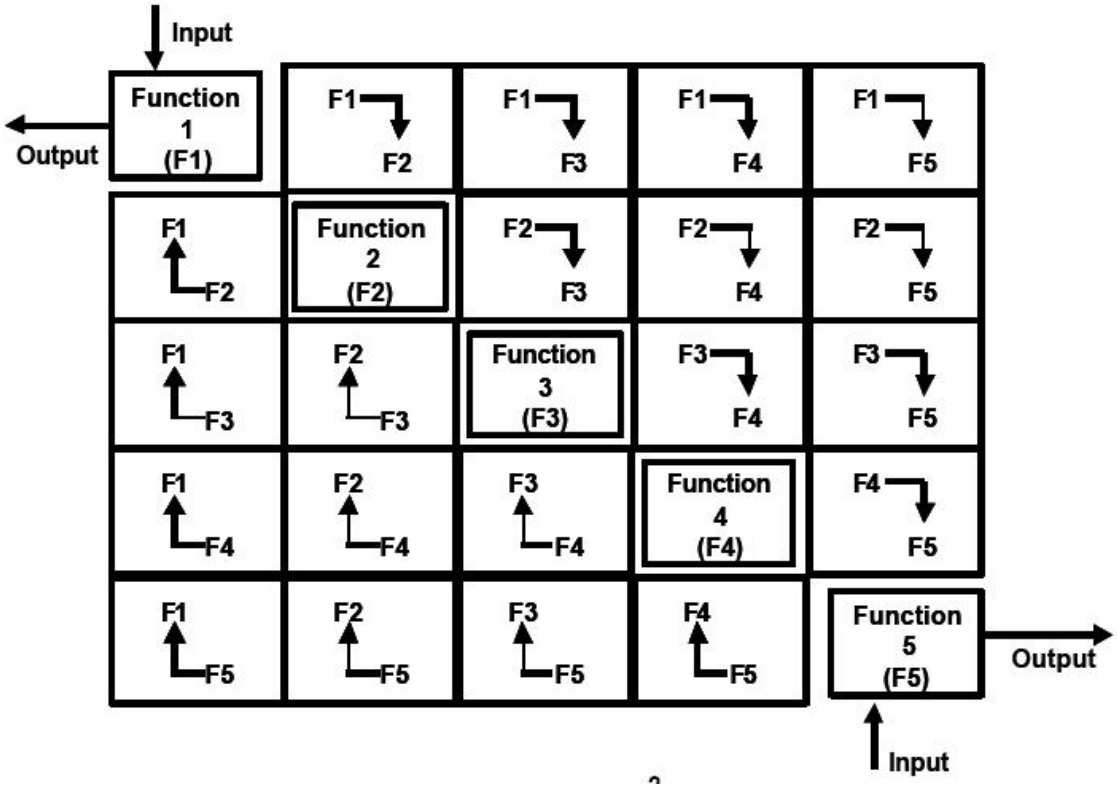


Figure 4. N^2 diagram

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.

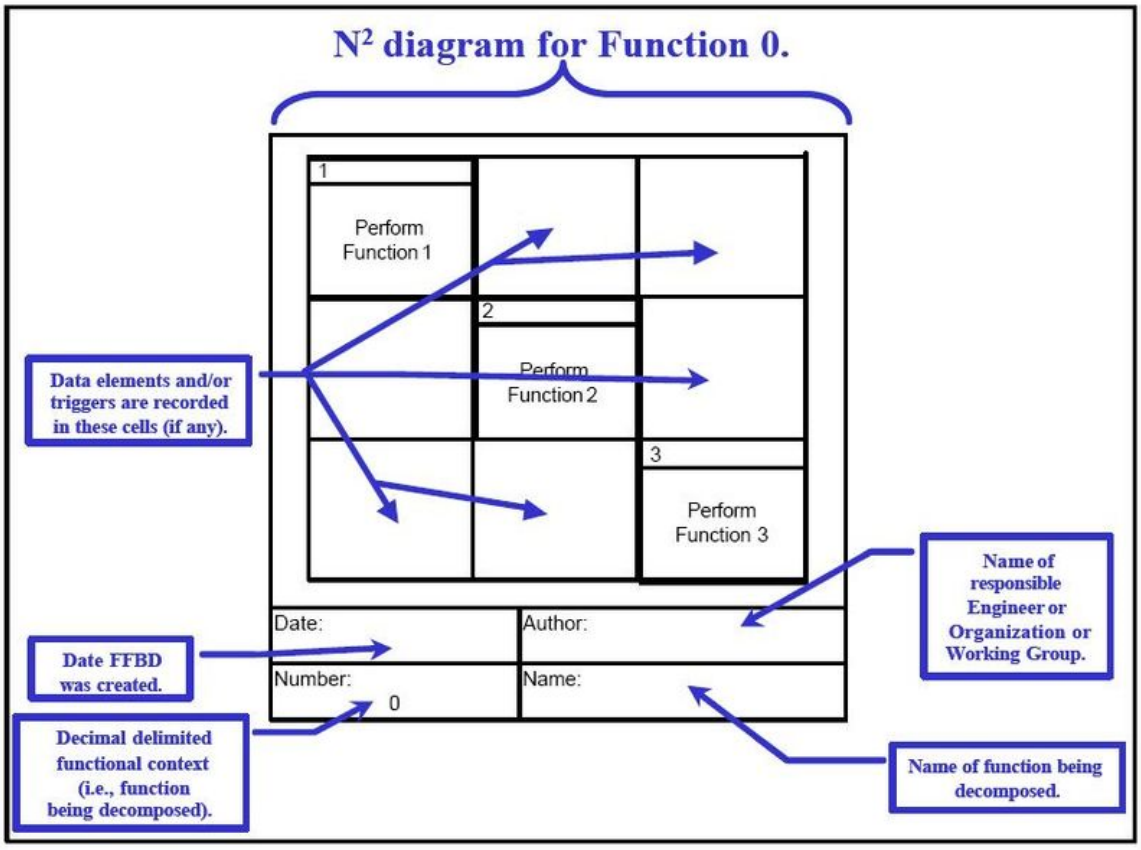
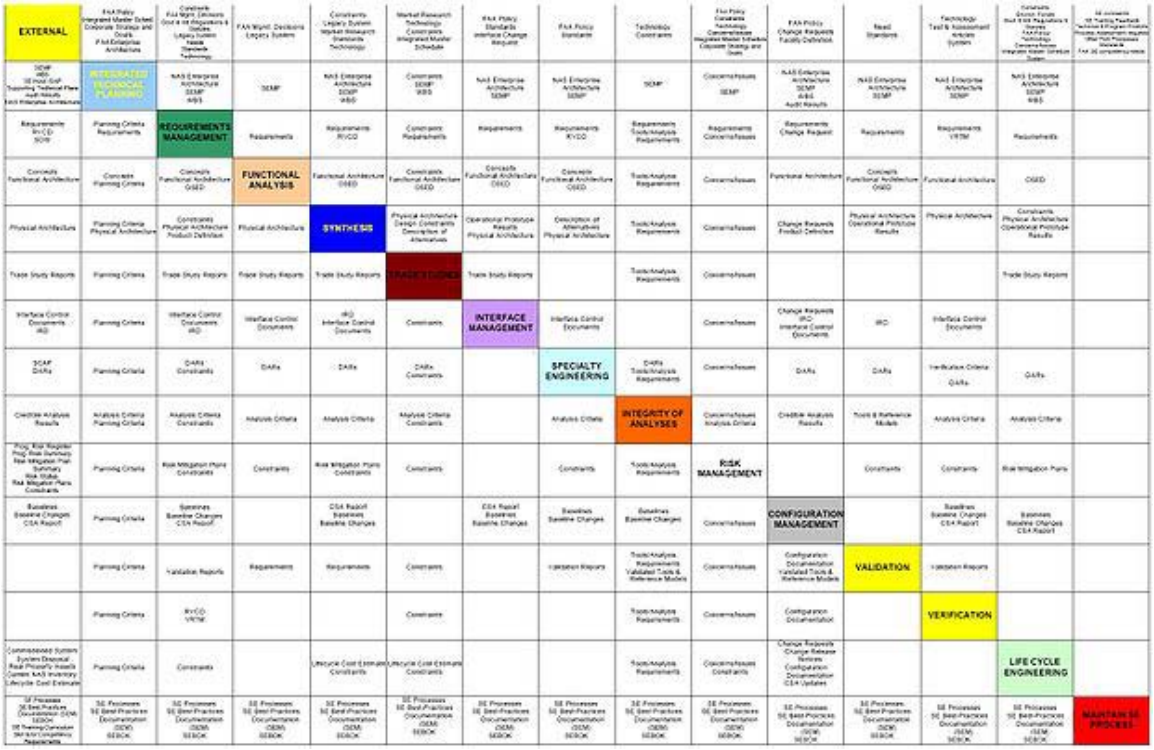


Figure 5 presents information in an N2 diagram, which complements the Functional flow block diagram. Notice that in this illustration, there are no data elements or triggers. The figure illustrates the context between functions at different levels of the model.

Examples

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



Chapter 13

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 14

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.

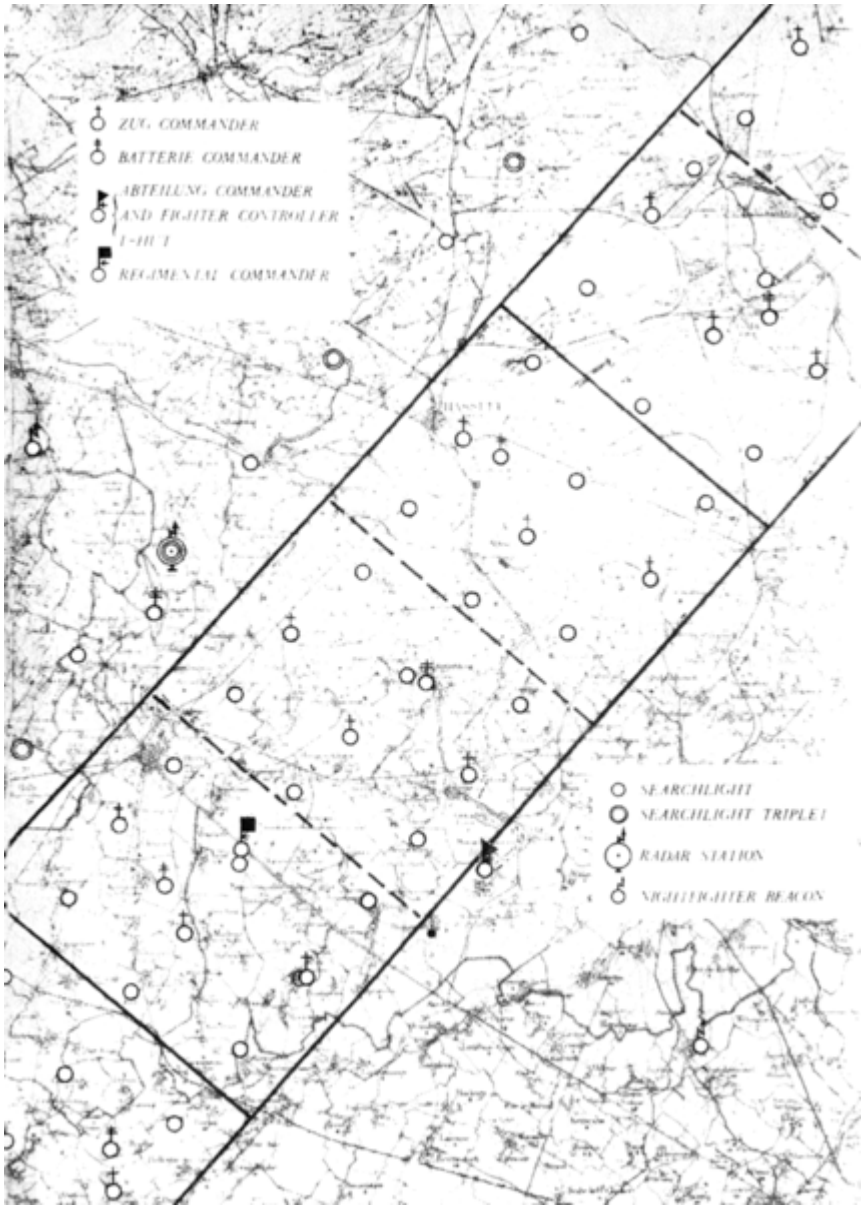
In 1941 Blackett moved from the RAE to the Navy, first to the Royal Navy's Coastal Command, in 1941 and then early in 1942 to the Admiralty. Blackett's team at Coastal Command's Operational Research Section (CC-ORS) included two future Nobel prize winners and many other people who went on to be preeminent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